COMPUTER NETWORKS

4th edition

Andrew S. Tanenbaum



Prentice Hall PTR
Upper Saddle River, New Jersey 07458
www.phptr.com

КЛАССИКА COMPUTER SCIENCE

Э. ТАНЕНБАУМ

КОМПЬЮТЕРНЫЕ СЕТИ

4-Е ИЗДАНИЕ

ПИТЕР®

Москва · Санкт-Петербург · Нижний Новгород · Воронеж
Ростов-на-Дону · Екатеринбург · Самара
Киев · Харьков · Минск
2003

БК 32.973.202
ДК 681.324
18

Т18 **Компьютерные сети. 4-е изд.** / Э. Таненбаум. — СПб.: Питер, 2003. — 992 с.: ил. — (Серия «Классика computer science»).

ISBN 5-318-00492-X

В этой книге подробно и последовательно изложены основные концепции, определяющие современное состояние и тенденции развития компьютерных сетей. Это уже четвертое, значительно переработанное издание книги, и три предыдущих неизменно были бестселлерами и использовались в качестве учебного пособия во многих западных университетах.

Автор подробнейшим образом объясняет устройство и принципы работы аппаратного и программного обеспечения, рассматривает все аспекты и уровни организации сетей, от физического до уровня прикладных программ. Изложение теоретических принципов дополняется яркими, показательными примерами функционирования Интернета, сетей ATM и беспроводных сетей.

ББК 32.973.202
УДК 681.324

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 0-13-066102-3 (англ.)
ISBN 5-318-00492-X

© 2003, 1996 Pearson Education, Inc.
© Перевод на русский язык, ЗАО Издательский дом «Питер», 2003
© Издвние на русском языке, оформление, ЗАО Издательский дом «Питер», 2003

Краткое содержание

Об авторе 16
Предисловие 17
Глава 1. Введение 21
Глава 2. Физический уровень 114
Глава 3. Уровень передачи данных 222
Глава 4. Подуровень управления доступом к среде 291
Глава 5. Сетевой уровень 399
Глава 6. Транспортный уровень 551
Глава 7. Прикладной уровень 658
Глава 8. Безопасность в сетях 814
Глава 9. Библиография 941
Алфавитный указатель 971

Содержание

Об авторе 16

Предисловие 17

От издательства 20

Глава 1. Введение 21

Применение компьютерных сетей 23

Сети в организациях 23
Использование сетей частными лицами 26
Использование беспроводных сетей 31
Социальный аспект 35

Сетевое оборудование 37

Локальные сети 39
Муниципальные сети 40
Глобальные сети 42
Беспроводные сети 44
Домашние сети 46
Объединения сетей 49

Сетевое программное обеспечение 50

Иерархия протоколов 50
Разработка уровней 54
Службы на основе соединений
и службы без установления соединений 56
Примитивы служб 58
Службы и протоколы 61

Эталонные модели 62

Эталонная модель OSI 62
Эталонная модель TCP/IP 66
Сравнение эталонных моделей OSI и TCP 69
Критика модели и протоколов OSI 70
Критика эталонной модели TCP/IP 73

Примеры сетей 74

Интернет 75
Сети на основе соединений: X.25, ретрансляция кадров, ATM 86

Ethernet . . . 92
Беспроводные ЛВС: 802.11 . . . 95

Стандартизация сетей . . . 98
Кто есть кто в мире телекоммуникаций . . . 99
Кто есть кто в мире международных стандартов . . . 102
Кто есть кто в мире стандартов Интернета . . . 104

Единицы измерения . . . 106

Краткое содержание следующих глав . . . 107

Резюме . . . 108

Вопросы . . . 110

Глава 2. Физический уровень . . . 114

Теоретические основы передачи данных . . . 115
Ряды Фурье . . . 115
Сигналы с ограниченным спектром . . . 115
Максимальная скорость передачи данных через канал . . . 118

Управляемые носители информации . . . 119
Магнитные носители . . . 119
Витая пара . . . 120
Коаксиальный кабель . . . 121
Волоконная оптика . . . 122

Беспроводная связь . . . 129
Электромагнитный спектр . . . 130
Радиосвязь . . . 133
Связь в микроволновом диапазоне . . . 134
Инфракрасные и миллиметровые волны . . . 138
Связь в видимом диапазоне . . . 138

Спутники связи . . . 140
Геостационарные спутники . . . 141
Средневысотные спутники . . . 145
Низкоорбитальные спутники . . . 145
Спутники против оптоволокна . . . 148

Коммутируемая телефонная сеть общего пользования . . . 149
Структура телефонной системы . . . 150
Политика телефонии . . . 153
Местные линии связи: модемы, ADSL, беспроводная связь . . . 156
Магистрали и уплотнение . . . 171
Коммутация . . . 182

Мобильная телефонная система . . . 187
Мобильные телефоны первого поколения: аналоговая передача речи . . . 189
Второе поколение мобильных телефонов: цифровая передача голоса . . . 193
Мобильные телефоны третьего поколения: цифровая речь и данные . . . 203

Кабельное телевидение 206
Абонентское телевидение 207
Кабельный Интернет 208
Распределение спектра 209
Кабельные модемы 211
ADSL или кабель? 214
Резюме 215
Вопросы 216

Глава 3. Уровень передачи данных 222

Ключевые аспекты организации уровня передачи данных 223
Сервисы, предоставляемые сетевому уровню 224
Формирование кадра 227
Обработка ошибок 230
Управление потоком 231
Обнаружение и исправление ошибок 232
Корректирующее кодирование 233
Коды с обнаружением ошибок 236
Элементарные протоколы передачи данных 240
Неограниченный симплексный протокол 245
Симплексный протокол с ожиданием 246
Симплексный протокол для зашумленных каналов 248
Протоколы скользящего окна 252
Протокол однобитового скользящего окна 254
Протокол с возвратом на *n* 257
Протокол с выборочным повтором 264
Верификация протоколов 270
Модели конечных автоматов 270
Сети Петри 273
Примеры протоколов передачи данных 276
HDLC — высокоуровневый протокол управления каналом 276
Уровень передачи данных в Интернете 280
Резюме 285
Вопросы 286

Глава 4. Подуровень управления доступом к среде 291

Проблема распределения канала 292
Статическое распределение канала в локальных и региональных сетях 292
Динамическое распределение каналов в локальных и региональных сетях 294
Протоколы коллективного доступа 295
Система ALOHA 295
Протоколы множественного доступа с контролем несущей 300

Протоколы без столкновений . . . 304
Протоколы с ограниченной конкуренцией . . . 307
Протоколы множественного доступа со спектральным разделением . . . 310
Протоколы беспроводных локальных сетей . . . 313

Сеть Ethernet . . . 317
Кабели Ethernet . . . 317
Манчестерский код . . . 321
Протокол подуровня управления доступом к среде в Ethernet . . . 322
Алгоритм двоичного экспоненциального отката . . . 325
Производительность сети стандарта 802.3 . . . 326
Коммутируемые сети Ethernet . . . 329
Быстрый Ethernet . . . 330
Гигабитная сеть Ethernet . . . 334
Стандарт IEEE 802.2: протокол LLC . . . 339
Ретроспектива Ethernet . . . 340

Беспроводные локальные сети . . . 341
Стандарт 802.11: стек протоколов . . . 341
Стандарт 802.11: физический уровень . . . 342
Стандарт 802.11: протокол подуровня управления доступом к среде . . . 345
Стандарт 802.11: структура кадра . . . 350
Сервисы . . . 351

Широкополосные беспроводные сети . . . 353
Сравнение стандартов 802.11 и 802.16 . . . 354
Стандарт 802.16: стек протоколов . . . 355
Стандарт 802.16: физический уровень . . . 356
Стандарт 802.16: протокол подуровня MAC . . . 358
Стандарт 802.16: структура кадра . . . 360

Bluetooth . . . 361
Архитектура Bluetooth . . . 362
Приложения Bluetooth . . . 363
Bluetooth: набор протоколов . . . 365
Bluetooth: уровень радиосвязи . . . 367
Bluetooth: уровень немодулированной передачи . . . 367
Bluetooth: уровень L2CAP . . . 368
Bluetooth: структура кадра . . . 369

Коммутация на уровне передачи данных . . . 370
Мосты между 802.x и 802.y . . . 372
Локальное межсетевое взаимодействие . . . 375
Мосты связующего дерева . . . 377
Удаленные мосты . . . 378
Повторители, концентраторы, мосты, коммутаторы, маршрутизаторы и шлюзы . . . 379
Виртуальные локальные сети . . . 382

Резюме . . . 391

Вопросы . . . 394

Глава 5. Сетевой уровень 399

Вопросы проектирования сетевого уровня 400
Метод коммутации пакетов с ожиданием 400
Сервисы, предоставляемые транспортному уровню 401
Реализация сервиса без установления соединения 402
Реализация сервиса с установлением соединения 404
Сравнение подсетей виртуальных каналов и дейтаграммных подсетей 405

Алгоритмы маршрутизации 406
Принцип оптимальности маршрута 408
Выбор кратчайшего пути 409
Заливка 412
Маршрутизация по вектору расстояний 413
Маршрутизация с учетом состояния линий 417
Иерархическая маршрутизация 424
Широковещательная маршрутизация 426
Многоадресная рассылка 428
Алгоритмы маршрутизации для мобильных хостов 430
Маршрутизация в специализированных сетях 433
Поиск узла в равноранговых сетях 439

Алгоритмы борьбы с перегрузкой 444
Общие принципы борьбы с перегрузкой 446
Стратегии предотвращения перегрузки 448
Борьба с перегрузкой в подсетях виртуальных каналов 450
Борьба с перегрузкой в дейтаграммных подсетях 451
Сброс нагрузки 454
Борьба с флуктуациями 456

Качество обслуживания 458
Требования 458
Методы достижения хорошего качества обслуживания 460
Интегральное обслуживание 472
Дифференцированное обслуживание 475
Коммутация меток и MPLS 478

Объединение сетей 481
Различия сетей 483
Способы объединения сетей 484
Сцепленные виртуальные каналы 486
Дейтаграммное объединение сетей 487
Туннелирование 489
Маршрутизация в объединенных сетях 490
Фрагментация 492

Сетевой уровень в Интернете 495
Протокол IP 498
IP-адреса 501
Управляющие протоколы Интернета 515
Протокол внутреннего шлюза OSPF 520

Протокол внешнего шлюза BGP . . . 526
Многоадресная рассылка в Интернете . . . 528
Мобильный IP . . . 529
Протокол IPv6 . . . 532

Резюме . . . 542

Вопросы . . . 543

Глава 6. Транспортный уровень . . . 551

Транспортная служба . . . 551
Услуги, предоставляемые верхним уровнем . . . 552
Примитивы транспортной службы . . . 554
Сокеты Беркли . . . 557
Пример программирования сокета: файл-сервер для Интернета . . . 559

Элементы транспортных протоколов . . . 563
Адресация . . . 564
Установка соединения . . . 567
Разрыв соединения . . . 573
Управление потоком и буферизация . . . 577
Мультиплексирование . . . 582
Восстановление после сбоев . . . 583

Простой транспортный протокол . . . 585
Служебные примитивы примера транспортного протокола . . . 585
Транспортная сущность примера транспортного протокола . . . 587
Пример протокола как конечного автомата . . . 595

Транспортные протоколы Интернета: UDP . . . 598
Основы UDP . . . 598
Вызов удаленной процедуры . . . 600
Транспортный протокол реального масштаба времени . . . 603

Транспортные протоколы Интернета: TCP . . . 607
Основы TCP . . . 607
Модель службы TCP . . . 608
Протокол TCP . . . 610
Заголовок TCP-сегмента . . . 611
Установка TCP-соединения . . . 614
Разрыв соединения TCP . . . 616
Модель управления TCP-соединением . . . 616
Управление передачей в TCP . . . 619
Борьба с перегрузкой в TCP . . . 623
Управление таймерами в TCP . . . 626
Беспроводные протоколы TCP и UDP . . . 629
Транзакционный TCP . . . 632

Вопросы производительности . . . 633
Причины снижения производительности компьютерных сетей . . . 634
Измерение производительности сети . . . 637

Проектирование производительных систем . . . 640
Быстрая обработка TPDU-модулей . . . 644
Протоколы для гигабитных сетей . . . 648
Резюме . . . 652
Вопросы . . . 653
Глава 7. Прикладной уровень . . . 658
Служба имен доменов DNS . . . 658
Пространство имен DNS . . . 659
Записи ресурсов . . . 662
Серверы имен . . . 665
Электронная почта . . . 668
Архитектура и службы . . . 670
Пользовательский агент . . . 672
Форматы сообщений . . . 675
Пересылка писем . . . 683
Доставка сообщений . . . 686
Всемирная паутина (WWW) . . . 693
Представление об архитектуре . . . 694
Статические веб-документы . . . 712
Динамические веб-документы . . . 727
HTTP — протокол передачи гипертекста . . . 735
Повышение производительности . . . 741
Беспроводная Паутина . . . 748
I-mode . . . 750
Мультимедиа . . . 760
Основы цифровой обработки звука . . . 761
Сжатие звука . . . 764
Потоковое аудио . . . 767
Интернет-радио . . . 771
Передача речи поверх IP . . . 774
Видео . . . 782
Сжатие видеоданных . . . 786
Видео по заказу . . . 795
Система MBone . . . 803
Резюме . . . 807
Вопросы . . . 808
Глава 8. Безопасность в сетях . . . 814
Криптография . . . 818
Основы криптографии . . . 819
Метод подстановки . . . 821
Метод перестановки . . . 823
Одноразовые блокноты . . . 824
Два фундаментальных принципа криптографии . . . 830

Алгоритмы с симметричным криптографическим ключом 832
Стандарт шифрования данных DES 834
Тройное шифрование с помощью DES 836
Улучшенный стандарт шифрования AES 837
Режимы шифрования 841
Другие шифры 847
Криптоанализ 848

Алгоритмы с открытым ключом 849
Алгоритм RSA 850
Другие алгоритмы с открытым ключом 852

Цифровые подписи 853
Подписи с симметричным ключом 853
Подписи с открытым ключом 855
Профили сообщений 856
Задача о днях рождения 861

Управление открытыми ключами 863
Сертификаты 864
X.509 866
Инфраструктуры систем с открытыми ключами 867

Защита соединений 871
IPsec 871
Брандмауэры 876
Виртуальные частные сети 879
Безопасность в беспроводных сетях 881

Протоколы аутентификации 886
Аутентификация, основанная на общем секретном ключе 887
Установка общего ключа:
протокол обмена ключами Диффи — Хеллмана 892
Аутентификация с помощью центра распространения ключей 894
Аутентификация при помощи протокола Kerberos 897
Аутентификация с помощью шифрования с открытым ключом 900

Конфиденциальность электронной переписки 901
PGP — довольно неплохая конфиденциальность 901
PEM — почта повышенной секретности 906
S/MIME 907

Защита информации во Всемирной паутине 907
Возможные опасности 908
Безопасное именование ресурсов 909
SSL — протокол защищенных сокетов 916
Защита переносимых программ 920

Социальный аспект 923
Конфиденциальность 924
Свобода слова 927
Защита авторских прав 931

Резюме 934
Вопросы 935
Глава 9. Библиография 941
Литература для дальнейшего чтения 941
Введение и неспециализированная литература 942
Физический уровень 943
Уроаень передачи данных 945
Подуровень управления доступом к носителю 946
Сетевой уровень 947
Транспортный уровень 949
Прикладной уровень 950
Безопасность в сетях 951
Алфавитный список литературы 952
Алфавитный указатель 971

Сьюзан, Барбаре, Марвину,
а также памяти Брэма и Крошки π

Об авторе

Эндрю Таненбаум получил степень бакалавра естественных наук в Массачусетском технологическом институте и степень доктора в Калифорнийском университете в Беркли. В настоящее время является профессором Амстердамского университета, где возглавляет группу разработчиков компьютерных систем. Кроме того, Э. Таненбаум возглавляет факультет вычислительной техники (межвузовскую аспирантуру, занимающуюся исследованиями в области современных параллельных систем, распределенных систем и систем обработки и формирования изображений). Тем не менее он всеми силами старается не превратиться в бюрократа.

В прошлом Эндрю Таненбаум занимался компиляторами, операционными системами, компьютерными сетями и локальными распределенными системами. В настоящее время его внимание сосредоточено на разработке глобальных распределенных систем, пользователями которых являются миллионы людей. С результатами этих исследований можно познакомиться на сайте *www.cs.vu.nl/globe*. Кроме того, в издательстве Питер, в 2003 году вышла книга «Распределенные системы», написанная Э. Таненбаумом совместно с профессором Маартеном ван Стееном.

Профессор Таненбаум разработал также значительный объем программного обеспечения. Он был главным архитектором «амстердамского пакета разработки компиляторов» (Amsterdam Compiler Kit), широко известного пакета для создания переносимых компиляторов, а также MINIX, миниатюрного клона UNIX, предназначенного для студенческих лабораторных работ по программированию. Вместе со своими аспирантами и программистами он способствовал созданию распределенной операционной системы Amoeba, высокопроизводительной распределенной операционной системы на базе микроядра. Системы MINIX и Amoeba бесплатно распространяются через Интернет.

Его аспиранты, многие из которых получили степень доктора, достигли больших успехов. Он очень ими гордится.

Профессор Таненбаум является членом ACM и IEEE, действительным членом Королевской академии наук и искусств Нидерландов, обладателем премии ACM 1994 года за заслуги в преподавательской деятельности, премии ACM/SIGCSE 1997 года за впечатляющий вклад в преподавание компьютерных дисциплин и премии Texty в 2002 году за свои великолепные учебники. Его имя упоминается в мировом справочнике «Кто есть кто в мире». Его домашнюю страницу в Интернете можно найти по адресу http://www.cs.vu.nl/~ast/.

Предисловие

Вот и вышло в свет уже четвертое издание этой книги. Каждое издание соответствует своему периоду развития компьютерных сетей. Так, в 1980 году, когда появилось первое из них, сети были лишь академической диковинкой. Второе издание (1988 год) пришлось на те времена, когда сетевые технологии стали применяться в университетах и большом бизнесе. В 1996 году появилось третье издание, и уже тогда сети, особенно Интернет, стали ежедневной реальностью для миллионов людей. Вы держите в руках книгу, соответствующую периоду стремительного развития технологий беспроводных сетей.

С тех пор, как вышло третье издание, ситуация в этой области изменилась самым радикальным образом. В середине 90-х существовало огромное количество разнообразных сетей LAN и WAN с соответствующим числом наборов протоколов. К 2003 году единственной широко распространенной системой, построенной на технологии проволочной LAN, остается Ethernet, а большинство WAN составляет Сеть сетей, то есть Интернет. В новом издании это учтено — устаревший материал убран.

Как вы понимаете, работа шла не только и не столько над удалением старого, но и над добавлением нового, поскольку со времен третьего издания появилось очень много новых технологий и решений. Наиболее резкий рывок вперед наблюдается в беспроводных сетях: стандарт 802.11, беспроводные местные линии связи, сотовые сети 2G и 3G, Bluetooth, WAP, i-mode — все это достижения последних лет, описанию которых уделено много места в новой книге. Все более важным становится вопрос защиты информации, этому посвящена целая глава.

Глава 1, как и в предыдущих изданиях, играет роль введения, однако ее содержимое было пересмотрено и дополнено. Например, в ней рассказывается об основах технологий Интернет, Ethernet, беспроводных локальных сетей, излагается их история. Вкратце рассматриваются домашние сети.

Глава 2 была несколько реорганизована. После небольшого введения, в котором рассказывается о принципах передачи данных, следуют три основных раздела (управляемая среда, беспроводная передача данных, спутниковая передача данных). Затем приводятся три очень важных примера: традиционная телефонная система, мобильная телефонная система, кабельное телевидение. Добавлены новые параграфы, касающиеся ADSL, широкополосных беспроводных сетей, глобальных беспроводных сетей, доступа в Интернет по кабелю и с помощью DOCSIS.

Глава 3 всегда была посвящена фундаментальным принципам работы двухточечных (point-to-point) протоколов. Идеям этим не суждено устареть. Не устарели и примеры, они без изменений были перенесены из третьего издания.

Зато подуровень управления доступом к среде (MAC) за последние годы претерпел существенные изменения, они отражены в главе 4. Раздел, касавшийся Ethernet, был расширен и теперь включает описание гигабитного Ethernet. Добавлены совершенно новые очень важные разделы, в которых рассказывается о беспроводных локальных сетях, широкополосных беспроводных сетях, Bluetooth, коммутации на канальном уровне (включая MPLS).

Глава 5 тоже была пересмотрена, из нее полностью исключен материал по ATM, зато добавлены новые материалы, касающиеся Интернета. В связи с этим рассматриваются вопросы интегрированных и дифференцированных служб. В этой главе тоже не обошлось без обсуждения беспроводных технологий, в частности, проблем маршрутизации в специализированных сетях. Добавлены темы, касающиеся NAT и равноранговых сетей.

Глава 6 по-прежнему посвящена транспортному уровню, но и в нее были внесены изменения. Например, добавлен пример программирования сокетов. Одностраничный клиент и одностраничный сервер запрограммированы на C, пример снабжен подробными пояснениями. Его смысл состоит в построении примитивного файл- или веб-сервера, с которым можно было бы поэкспериментировать. Листинги программ можно скачать с веб-сайта книги, затем откомпилировать и запустить. Среди других обновлений необходимо отметить вопросы RTP, транзакций/TCP и удаленного вызова процедур.

Глава 7, посвященная прикладному уровню, стала чуть более конкретной. После небольшого введения в DNS следует основная часть, в которой обсуждаются всего три вопроса: e-mail, Web и мультимедиа. Но каждый из них рассматривается очень подробно. Например, объяснение принципа работы Web занимает около 60 страниц, на которых вы найдете следующие темы: статические и динамические веб-страницы, HTTP, CGI-скрипты, сети доставки содержимого, cookie-файлы, веб-кэширование. Здесь же рассказывается о принципах создания современных сайтов, включая основы XML, XSL, XHTML, PHP и т. д. Приводятся примеры, которые можно запустить и изучить «живьем». Что касается беспроводных сетей, то в этой главе вы найдете описание i-mode и WAP. Раздел, посвященный мультимедиа, включает теперь описание формата MP3, потокового звука, интернет-радио, а также технологий передачи речи по IP.

Защита информации стала настолько актуальной темой, что ее обсуждение теперь занимает целую главу объемом свыше 100 страниц. В главе 8 описываются как теоретические принципы безопасности в сетях передачи данных (алгоритмы симметричного шифрования и шифрования с открытыми ключами, цифровые подписи, сертификаты X.509), так и их практические приложения (аутентификация, защита от несанкционированного доступа к e-mail, защита информации в Web). Можно сказать, что эта глава простирается как вширь (от квантовой криптографии до правительственной цензуры), так и вглубь (взять хотя бы детальное рассмотрение работы SHA-1).

Глава 9 содержит полностью обновленный список литературы для дальнейшего чтения, библиографию более 350 ссылок, встречающихся в тексте. Около 200 из этих источников представляют собой газеты и книги, написанные после 2000 года.

В изданиях по компьютерной тематике всегда много сокращений. Данная книга не стала исключением. После ее прочтения вы будете легко оперировать следующими аббревиатурами: ADSL, AES, AMPS, AODV, ARP, ATM, BGP, CDMA, CDN, CGI, CIDR, DCF, DES, DHCP, DMCA, FDM, FHSS, GPRS, GSM, HDLC, HFC, HTML, HTTP, ICMP, IMAP, ISP, ITU, LAN, LMDS, MAC, MACA, MIME, MPEG, MPLS, MTU, NAP, NAT, NSA, NTSC, OFDM, OSPF, PCF, PCM, PGP, PHP, PKI, POTS, PPP, PSTN, QAM, QPSK, RED, RFC, RPC, RSA, RSVP, RTP, SSL, TCP, TDM, UDP, URL, UTP, VLAN, VPN, VSAT, WAN, WAP, WDMA, WEP, WWW и XML. Каждая из них будет расшифрована, так что волноваться не стоит. Нужно лишь внимательно читать книгу.

В помощь преподавателям, использующим эту книгу в качестве основы для своих курсов, автор подготовил различные дополнительные пособия:

- руководство по решению возникающих проблем;
- рисунки, таблицы и графики в различных электронных форматах;
- презентации в Power Point для построения лекций на основе этой книги;
- симулятор примеров протоколов из главы 3 (написанный на языке C);
- веб-страницу со ссылками на множество ресурсов по тематике книги (организации, самоучители, ответы на часто задаваемые вопросы и т. д.).

Сборник задач с решениями преподаватели (но не студенты!) могут получить в издательстве Prentice Hall. Весь прочий материал можно найти на сайте книги по адресу http://www.prenhall.com/tanenbaum. Там нужно щелкнуть на изображении обложки книги.

В процессе создания четвертого издания книги мне помогало множество замечательных людей, которых я хотел бы поблагодарить. Это Росс Андерсон (Ross Anderson), Элизабет Белдинг-Ройер (Elizabeth Belding-Royer), Стив Белловин (Steve Bellovin), Чачик Биздикян (Chatschik Bisdikian), Киз Бот (Kees Bot), Скотт Брэднер (Scott Bradner), Дженнифер Брэй (Jennifer Bray), Пэт Кейн (Pat Cain), Эд Фелтен (Ed Felten), Варвик Форд (Warwick Ford), Кевин Фю (Kevin Fu), Рон Фулль (Ron Fulle), Джим Джейер (Jim Geier), Марио Джерла (Mario Gerla), Натали Жиру (Nathalie Giroux), Стив Ханна (Steve Hanna), Джеф Хейес (Jeff Hayes), Амир Херцберг (Amir Herzberg), Филип Хомбург (Philip Homburg), Филипп Хошка (Philipp Hoschka), Дэвид Грин (David Green), Барт Джекобс (Bart Jacobs), Франц Кашук (Frans Kaashoek), Стив Кент (Steve Kent), Роджер Кермоуд (Roger Kermode), Роберт Киницки (Robert Kinicki), Шей Каттен (Shay Kutten), Роб Ланфье (Rob Lanphier), Маркус Лич (Marcus Leech), Том Мофер (Tom Maufer), Брент Миллер (Brent Miller), Шивакант Мишра (Shivakant Mishra), Томас Надо (Thomas Nadeau), Шлёмо Овадья (Shlomo Ovadia), Кавех Пахлаван (Kaveh Pahlavan), Радья Перлман (Radia Perlman), Гильом Пьер (Guilaumme Pierre), Уэйн Плезент (Wayne Pleasant), Патрик Пауэлл (Patrick Powell), Томас Робертацци (Thomas Robertazzi), Меди Санадиди (Medy

Sanadidi), Кристиан Шмутцер (Christian Schmutzer), Хеннинг Шульцринне (Henning Schulzrinne), Поль Севинк (Paul Sevinc), Михаил Сичитью (Mihail Sichitiu), Бернард Склар (Bernard Sklar), Эд Шкодиус (Ed Skodius), Боб Страдер (Bob Strader), Джордж Суоллоу (George Swallow), Джордж Сирувафукал (George Thiruvathukal), Петер Томсу (Peter Tomsu), Патрик Веркайк (Patrick Verkaik), Дэйв Виттали (Dave Vittali), Спирос Вулгарис (Spyros Voulgaris), Жан-Марк Вэмс (Jan-Mark Wams), Ройдигер Вайс (Ruediger Weis), Берт Вийнен (Bert Wijnen), Джозеф Уилкс (Joseph Wilkes), Лендерт ван Дорн (Leendert van Doorn) и Маартен ван Стеен (Maarten van Steen).

Особая благодарность Труди Левин (Trudy Levine), которая доказала, что бабушка с успехом может быть редактором технической литературы. Шивакант Мишра (Shivakant Mishra) приложил много усилий для решения проблем, возникавших в конце разделов. Энди Дорнэн (Andy Dornan) предложил добавить в библиографический список несколько замечательных источников для дальнейшего чтения. Жан Луен (Jan Looyen) предоставил необходимую аппаратуру в критический момент. Доктор Ф. де Ни (Dr. F. de Nies) мастерски проделывал работу по удалению и вставке слов в нужных местах. Мой редактор Мэри Франц (Mary Franz) из издательства «Prentice Hall», как обычно, снабдила меня таким количеством литературы для чтения, которое я не сумел бы прочитать за предыдущие 7 лет, и оказывала всяческую поддержку.

Наконец, самые главные люди: Сьюзан (Suzanne), Барбара (Barbara) и Марвин (Marvin). Сьюзан, спасибо тебе за любовь, заботу и замечательные пикники. Барбара и Марвин, вы всегда так веселы и жизнерадостны (кроме тех моментов, когда вы жалуетесь на ужасные учебники из колледжа), что и меня заряжаете этим же настроением. Спасибо вам.

Эндрю С. Таненбаум

От издательства

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

Подробную информацию о наших книгах вы найдете на web-сайте издательства http://www.piter.com.

Глава 1
Введение

- Применение компьютерных сетей
- Сетевое оборудование
- Сетевое программное обеспечение
- Эталонные модели
- Примеры сетей
- Стандартизация сетей
- Единицы измерения
- Краткое содержание следующих глав
- Резюме
- Вопросы

Каждое из трех прошедших столетий ознаменовалось преобладанием своей господствующей технологии. XVIII столетие было веком индустриальной революции и механизации. В XIX веке наступила эпоха паровых двигателей. В течение XX века главной технологией стали сбор, обработка и распространение информации. Среди прочих разработок следует отметить создание глобальных телефонных сетей, изобретение радио и телевидения, рождение и небывалый рост компьютерной индустрии, запуск спутников связи.

Благодаря высокой скорости технологического прогресса эти области очень быстро проникают друг в друга. При этом различия между сбором, транспортировкой, хранением и обработкой информации продолжают быстро исчезать. Организации с сотнями офисов, разбросанных по всему миру, должны иметь возможность получать информацию о текущем состоянии своего самого удаленного офиса мгновенно, нажатием кнопки. По мере роста нашего умения собирать, обрабатывать и распространять информацию потребности в средствах еще более сложной обработки информации растут все быстрее.

Хотя компьютерная индустрия еще довольно молода по сравнению с другими производствами (например, автомобильной или авиационной промышленностью), прогресс в сфере производства компьютеров был весьма впечатляющим. В первые два десятилетия своего существования компьютерные системы были сильно централизованными и располагались, как правило, в пределах одного помещения. Часто эта комната оборудовалась стеклянными стенами, сквозь которые посетители могли полюбоваться на великое электронное чудо. Компания среднего размера или университет могли позволить себе один-два компьютера, тогда как у крупных организаций их число могло достигать нескольких десятков. Сама мысль о том, что через какие-нибудь 20 лет столь же мощные компьютеры будут иметь размеры почтовой марки и производиться миллионами, тогда казалась чистой фантастикой.

Объединение компьютеров и средств связи оказало глубокое влияние на принцип организации компьютерных систем. Концепция «компьютерного центра» в виде комнаты, в которой помещался большой компьютер и куда пользователи приносили свои программы, сегодня полностью устарела. Модель, в которой один компьютер выполнял всю необходимую работу по обработке данных, уступила место модели, состоящей из большого количество отдельных, но связанных между собой компьютеров. Такие системы называются компьютерными сетями. Разработке и устройству сетей и посвящена данная книга.

На протяжении всей книги мы будем использовать термин «компьютерная сеть» для обозначения набора связанных между собой автономных компьютеров. Два компьютера называются связанными между собой, если они могут обмениваться информацией. Связь не обязательно должна осуществляться при помощи медного провода. Могут использоваться самые разнообразные средства связи, включая волоконную оптику, радиоволны высокой частоты и спутники связи. Сети могут отличаться друг от друга также размерами и принципами построения. Как ни странно, ни Интернет, ни так называемая Мировая паутина (WWW) не являются сетями. К концу книги вы поймете, что это всего лишь расхожее заблуждение. Сейчас я дам только короткое объяснение этому: Интернет — это сеть сетей, а Веб — распределенная система на базе Интернета.

В литературе существует путаница между понятиями «компьютерная сеть» и «распределенная система». Основное их различие заключается в том, что в распределенной системе наличие многочисленных автономных компьютеров незаметно для пользователя. С его точки зрения, это единая связанная система. Обычно имеется набор программного обеспечения на определенном уровне (над операционной системой), которое называется связующим ПО и отвечает за реализацию этой идеи. Хорошо известный пример распределенной системы — это Мировая паутина (World Wide Web), в которой, с точки зрения пользователя, все выглядит как документ (веб-страница).

В компьютерных сетях нет никакой единой модели, нет и программного обеспечения для ее реализации. Пользователи имеют дело с реальными машинами, и со стороны вычислительной системы не осуществляется никаких попыток связать их воедино. Скажем, если компьютеры имеют разное аппаратное и программное обеспечение, пользователь не сможет этого не заметить. Если он хочет

запустить программу на удаленной машине, ему придется явно зарегистрироваться на ней и явно дать задание на запуск.

На самом деле распределенная система является программной системой, построенной на базе сети. Эта программная система обеспечивает высокую степень связности и прозрачности элементов. Таким образом, различие между компьютерной сетью и распределенной системой заключается в программном обеспечении (особенно в операционной системе), а не в аппаратуре.

Тем не менее эти два понятия имеют много общего. Например, как компьютерная сеть, так и распределенная система занимаются перемещением файлов. Разница заключается в том, кто вызывает эти перемещения — система или пользователь.

Хотя основной темой этой книги являются сети, многие разделы будут касаться и распределенных систем. Дополнительную информацию о распределенных системах см: Таненбаум Э., ван Стеен М. Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003 (Tannenbaum and van Steen, 2002).

Применение компьютерных сетей

Прежде чем приступить к изучению технических подробностей, стоит посвятить некоторое время обсуждению вопросов, почему люди интересуются компьютерными сетями и для чего эти сети могут быть использованы. В конце концов, если бы никто не был заинтересован в развитии этих технологий, то не было бы построено такого огромного количества самых разных сетей. Мы начнем с обсуждения таких традиционных вещей, как корпоративные сети и вообще сети в организациях, использование сетей частными лицами, затем перейдем к новым технологиям, связанным с мобильной связью и домашними сетями.

Сети в организациях

Многие современные организации используют большое количество компьютеров, зачастую значительно удаленных друг от друга. Например, могут быть отдельные компьютеры для слежения за производственным процессом, учета товаров и начисления заработной платы. Поначалу все эти компьютеры нередко работают изолированно друг от друга, однако в какой-то момент администрация может принять решение соединить их, чтобы иметь возможность быстрого доступа к информации по всей компании.

Если посмотреть на эту проблему с более общих позиций, то вопросом здесь является совместное использование ресурсов, а целью — предоставление доступа к программам, оборудованию и особенно данным любому пользователю сети, независимо от физического расположения ресурса и пользователя. В качестве примера можно привести сетевой принтер, то есть устройство, доступ к которому может осуществляться с любой рабочей станции сети. Это выгодное решение, поскольку нет никакой необходимости в том, чтобы свое печатающее устройство было у каждого служащего, к тому же содержание и обслуживание одного принтера, очевидно, обходится дешевле.

Но, наверное, даже более важной проблемой, нежели совместное использование физических ресурсов (принтеров, сканеров, устройств записи компакт-дисков), является совместное использование информации. В наше время любая компания, независимо от ее размеров, просто немыслима без данных, представленных в электронном виде. Большинство фирм старается вести базу данных клиентов, товаров, счетов, финансовых операций, очень часто требуется налоговая информация и многое другое. Если бы вдруг отказали все компьютеры какого-нибудь банка, даже самого крупного, он обанкротился бы минут за пять, не более. Любое автоматизированное производство с использованием вычислительной техники в этом случае не продержалось бы и столько. Да что там говорить, если даже маленькое туристическое агентство, весь штат которого состоит из трех человек, очень сильно зависит от компьютерных сетей, позволяющих получать доступ к необходимой информации и документам.

В маленьких компаниях все компьютеры обычно собраны в пределах одного офиса или, в крайнем случае, одного здания. Если же речь идет о больших фирмах, то и вычислительная техника, и служащие могут быть разбросаны по десяткам представительств в разных странах. Несмотря на это продавец, находящийся в Нью-Йорке, может запросить и сразу же получить информацию о товарах, имеющихся на складе в Сингапуре. Другими словами, тот факт, что пользователь удален от физического хранилища данных на 15 тысяч километров, никак не ограничивает его возможности доступа к этим данным. Можно сказать, что одной из целей сетей является борьба с «тиранией географии».

Проще всего информационную систему компании можно представить себе как совокупность одной или более баз данных и некоторого количества работников, которым удаленно предоставляется информация. В этом случае данные хранятся на мощном компьютере, называемом **сервером**. Довольно часто сервер располагается в отдельном помещении и обслуживается системным администратором. С другой стороны, компьютеры служащих могут быть менее мощными, они идентифицируются в сети как **клиенты**, могут в большом количестве располагаться даже в пределах одного офиса и иметь удаленный доступ к информации и программам, хранящимся на сервере. (Иногда мы будем называть клиентом пользователя такой машины. Я думаю, вы сможете по контексту догадаться, когда речь идет о компьютере, а когда — о человеке.) Клиентская и серверная машины объединены в сеть, как показано на рис. 1.1. Обратите внимание: пока что мы показываем сеть просто в виде овала, не вдаваясь в детали. Такое представление мы будем использовать при ведении абстрактного разговора о компьютерных сетях. При обсуждении того или иного аспекта их функционирования мы будем «раскрывать» этот овал, узнавая о нем все новые подробности.

Такая система называется **клиент-серверной моделью**. Она используется очень широко и зачастую является основой построения всей сети. Она применима и тогда, когда клиент и сервер находятся в одном здании, и когда они расположены далеко друг от друга. Скажем, когда пользователь получает доступ к интернет-сайту, работает та же модель. При этом веб-сервер играет роль серверной машины, а пользовательский компьютер — клиентской. В большинстве случаев один сервер занимается обслуживанием большого числа клиентов.

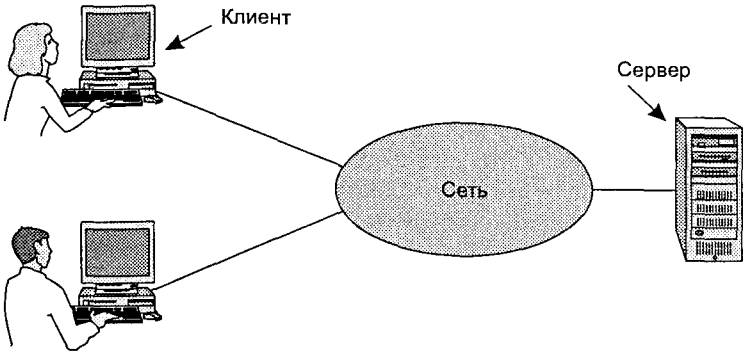


Рис. 1.1. Сеть, состоящая из двух клиентов и одного сервера

Если мы посмотрим на модель «клиент-сервер» чуть пристальнее, то станет очевидно, что в работе сети можно всегда выделить два процесса: серверный и клиентский. Обмен информацией чаще всего происходит так. Клиент посылает запрос серверу через сеть и начинает ожидать ответ. При принятии запроса сервер выполняет определенные действия или ищет запрашиваемые данные, затем отсылает ответ. Все это показано на рис. 1.2.

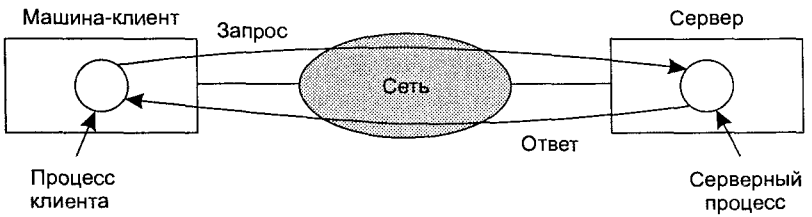


Рис. 1.2. В модели «клиент-сервер» различают запросы и ответы

Вторая цель работы компьютерной сети связана в большей степени с людьми, чем с информацией или вычислительными машинами. Дело в том, что сеть — это замечательная **коммуникационная среда** для работников предприятия. Почти в любой компании найдется хотя бы один компьютер, умеющий принимать и отправлять **электронную почту (e-mail)**, а ведь именно ее большинство людей предпочитает использовать для общения. На самом деле обычное ворчание начальства на тему того, что люди проводят много времени за чтением и написанием электронных писем, совершенно беспочвенно: многие руководители давно уже поняли, что они и сами могут рассылать своим подчиненным электронные послания — это удобно и просто.

Однако e-mail — это далеко не единственное средство связи, предоставляемое сетевыми технологиями. При помощи сети два или более удаленных друг от друга сотрудника могут легко составить совместный отчет. Если один из сотрудников изменяет документ, находящийся на сервере, в подключенном режиме (on-line), остальные сотрудники могут немедленно увидеть эти изменения, а не ждать пись-

ма в течение нескольких дней. Подобное ускорение передачи информации делает возможным сотрудничество удаленных друг от друга групп людей.

Еще одним способом связи между сотрудниками является видеоконференция. Используя эту технологию, можно устраивать встречи, причем собеседники, находящиеся за тысячи километров друг от друга, будут не только слышать, но и видеть друг друга. Кроме того, можно оставлять записи на виртуальной доске, являющейся разделяемым ресурсом (доступным обеим сторонам). Видеоконференции иногда способны заменить реальные встречи, что позволяет сэкономить деньги и время, которые пришлось бы потратить на поездку. Уже идут разговоры о том, что виртуальное общение конкурирует с перемещением в пространстве. Как только что-то одно победит, другое сразу начнет устаревать.

Третья цель применения компьютерных сетей становится очевидна все большему числу компаний — это возможность электронного делового общения с другими компаниями. Особенно это касается взаимоотношений типа «поставщик—клиент». Например, производители автомобилей, летательных аппаратов, компьютеров закупают комплектующие и детали у огромного числа поставщиков, а затем занимаются сборкой конечной продукции. С помощью компьютерных сетей процесс составления и отправки заказов можно автоматизировать. Более того, заказы могут формироваться строго в соответствии с производственными нуждами, что позволяет резко повысить эффективность.

Четвертая цель — это интернет-коммерция. Эта область сейчас является очень перспективной и быстро развивающейся. Через Интернет уже можно приобретать, например, билеты на самолет, книги или музыкальные компакт-диски. Компании, занимающиеся торговлей, в какой-то момент поняли, что многим клиентам удобнее совершать покупки, не выходя из дома. В Интернете начали появляться каталоги продукции и услуг, а заказы пользователь смог осуществлять прямо в подключенном (on-line) режиме. Вся эта технология называется **электронным бизнесом**.

Использование сетей частными лицами

В 1977 году Кен Олсен (Ken Olsen) был президентом корпорации DEC (Digital Equipment Corporation), которая на тот момент была второй по величине (после IBM) компанией, производящей компьютерную технику. Когда у него спросили, почему DEC не поддерживает идею создания персональных компьютеров, он сказал: «Я не вижу никакого смысла в том, чтобы в каждом доме стоял компьютер». Возможно, он и был прав, но исторический факт заключается в том, что все оказалось как раз наоборот, а корпорация DEC вообще прекратила свое существование. Зачем люди устанавливают компьютеры у себя дома? Изначально основными целями были редактирование текстов и электронные игры. Однако за последние несколько лет ситуация радикальным образом изменилась, и теперь основная причина, по которой многие люди приобретают компьютеры, — это доступ в Интернет. Дома его можно использовать, например, в таких целях:

- ✦ доступ к удаленной информации;
- ✦ общение;

- ✦ интерактивные развлечения;
- ✦ электронный бизнес.

Доступ к удаленной информации может осуществляться в различной форме. Можно бродить по Сети в поисках нужной или просто интересной информации. При этом практически невозможно найти такую область знаний, которая не была бы представлена в Интернете. Там есть искусство, бизнес, кулинария, политика, здоровье, история, различные хобби, отдых, наука, спорт, путешествия и многое-многое другое. Развлекательных тем безумное количество, причем некоторые из них не стоит даже упоминать.

Многие газеты стали доступны в электронном виде, их можно персонализировать. Например, можно заказать себе все статьи, касающиеся коррупции среди политических деятелей, больших пожаров, скандалов, связанных со знаменитостями, эпидемий, но отказаться от статей о футболе. Можно сделать и так, чтобы ночью газета загружалась на ваш жесткий диск или распечатывалась на принтере, — тогда с утра, перед завтраком, вы ее с удовольствием прочтете. Конечно, это все может привести к массовой безработице среди подростков, продающих на улицах газеты, но сами редакции довольны таким поворотом событий, поскольку распространение всегда было самым слабым звеном в их производственной цепочке.

Следующий шаг после создания электронных версий газет и журналов — это онлайновые библиотеки. Многие профессиональные организации, такие как ACM (www.acm.org) и даже объединение IEEE (www.computer.org), уже занимаются этим. Да и другие фирмы и частные лица выкладывают свои коллекции самых разнообразных материалов в Интернете. Если учесть, что цены на ноутбуки падают, а их размер и вес уменьшаются, не исключено, что в скором времени печатные издания начнут морально устаревать. Скептики уже сейчас сравнивают это с эффектом от появления в средние века печатного станка, который заменил ручное письмо.

Все приведенные возможности применения сетей включают взаимодействие между пользователем и удаленной базой данных. Следующей категорией применения сетей является общение между частными лицами, что можно назвать ответом XXI века веку XIX с его изобретением телефона. Электронная почта уже широко используется миллионами людей, и скоро включение в письмо изображений и звука наравне с текстом станет обычным делом. Несколько большего периода времени, по-видимому, потребует достижение совершенства в передаче запахов.

Огромное количество подростков обожает так называемую **систему диалоговых сообщений (чатов)**. А все, между прочим, начиналось с программы *talk*, написанной под UNIX еще примерно в 1970 году. Она позволяла двум пользователям обмениваться сообщениями в реальном времени. Когда пользователей, принимающих участие в разговоре, становится больше, это превращается в то, что называется чатом.

Всемирные конференции, в которых обсуждаются всевозможные темы, уже вполне обычны для некоторых групп людей и вскоре, возможно, станут привычными для большинства населения планеты. Подобные обсуждения, в которых

один человек посылает сообщение, а прочитать его могут все остальные подписчики конференции, охватывают весь спектр тем, от юмористических до самых серьезных. В отличие от чатов, конференции не проводятся в реальном времени, а сохраняются в виде сообщений, чем-то напоминающих электронную почту. Поэтому, скажем, вернувшись из отпуска, подписчик может прочитать все дискуссии, которые велись во время его отсутствия.

Еще один тип сетевого общения основан на технологии **равноранговых сетей (peer-to-peer)**. Эта модель принципиально отличается от модели «клиент-сервер» (Parameswaran и др., 2001). Люди, входящие в некоторую группу пользователей, могут общаться друг с другом. В принципе, каждый может связаться с каждым, разделение на клиентские и серверные машины в этом случае отсутствует. Это показано на рис. 1.3.

Такого рода коммуникации стали очень популярны примерно в 2000 году, они были реализованы с помощью службы Napster. В апогее своего развития равноранговая сеть насчитывала порядка 50 миллионов (!) любителей музыки, которые обменивались записями, и это, пожалуй, было самое масштабное нарушение закона об авторских правах за всю историю звукозаписи. Идея была очень проста. Члены группы вносили информацию об имеющихся на их жестких дисках аудиозаписях в специальную базу данных, хранившуюся на сервере Napster. Если кто-то хотел скачать какую-нибудь песню или альбом, он смотрел по этой базе, у кого есть соответствующие файлы, и обращался напрямую к их обладателю. Поскольку на самом сервере Napster никогда не хранился ни один аудиофайл, компания утверждала, что никакие законы она не нарушает. Тем не менее суд постановил прекратить деятельность этой службы.

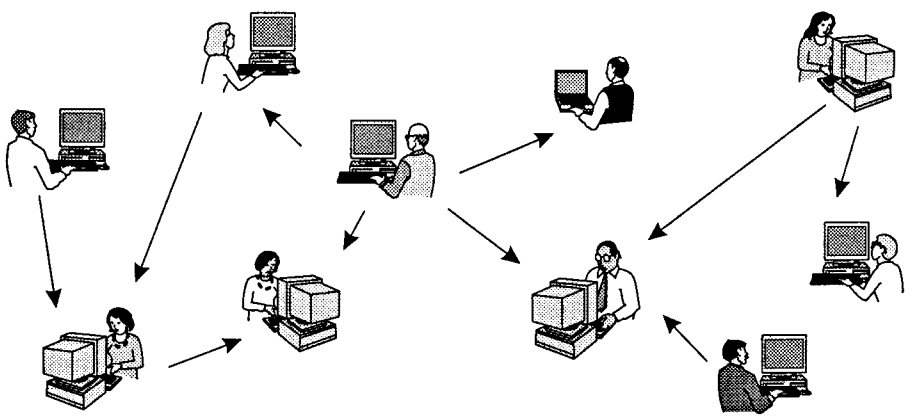


Рис. 1.3. В равноранговой сети нет четко определенных клиентов и серверов

Пользователи этой системы, видимо, приняли все это к сведению, и следующее поколение равноранговых сетей отличалось отсутствием централизованной базы данных. Теперь каждый член группы стал поддерживать свою базу и вести список группы. Новый пользователь узнавал, что есть у какого-то одного члена бывшей группы Napster, а также получал список всех остальных членов, у кото-

рых мог запросить какие-то другие музыкальные файлы. Процесс поиска, таким образом, мог продолжаться бесконечно, и каждый пользователь мог составить собственную довольно объемную базу данных. Такой род деятельности, конечно, утомляет человека, но компьютер при соответствующей настройке программного обеспечения прекрасно с этим справляется.

Между тем существуют и легальные равноранговые сети. Например, теми же записями можно обмениваться, если музыканты выпустили какие-то треки для публичного воспроизведения и не претендуют на полноценные авторские права. Бывают семейные сети, в которых люди обмениваются фотографиями, любительскими фильмами, информацией, касающейся генеалогического древа, и т. п. Существуют равноранговые сети, организованные подростками, увлекающимися онлайновыми играми. Между прочим не стоит забывать и о том, что самая популярная интернет-технология — e-mail — выросла именно из идеи равноранговых сетей. Такой вид коммуникаций вообще является перспективным, и в будущем он еще будет довольно активно развиваться.

Преступления с применением компьютерных сетей, увы, не ограничиваются только нарушением закона об авторских правах. Еще одна сфера деятельности мошенников — это электронные азартные игры. Компьютеры вот уже несколько десятилетий с успехом занимаются симуляцией самых разных вещей. Почему бы не научить их симулировать игровые автоматы, рулетки и т. п.? Все это замечательно, но во многих странах действует закон, запрещающий играть в азартные игры. Причем основная проблема заключается в том, что существуют места, где это официально разрешено (например, Англия), и тамошние владельцы казино, почуяв, что их деятельностью может заинтересоваться новый огромный сектор веб-клиентуры, стали развивать это направление. А что будет, если игрок и виртуальное казино окажутся в разных странах, в которых действуют противоречивые законы? Хороший вопрос.

Среди других прикладных направлений, использующих средства Интернета, стоит отметить IP-телефонию, виртуальные видеофоны, интернет-радио. Все это сегодня находится в стадии стремительного развития. Еще одна большая прикладная область — электронные образовательные программы. Например, можно посещать виртуальную школу, и для того чтобы успеть на первый урок, начинающийся в 8 утра, не придется предварительно вставать с постели. В дальнейшем именно сетевое общение станет фаворитом среди всех других средств коммуникации.

Наша третья категория — развлечения — является гигантской индустрией, продолжающей развиваться. Основным направлением сейчас является видео по заказу. Лет через десять будет можно выбрать любой фильм или телевизионную передачу из когда-либо и где-либо снятых и увидеть ее мгновенно на своем экране. Новые фильмы могут стать интерактивными, где пользователю периодически будет предлагаться выбор сюжетной линии (должен ли Макбет убить Дункана сейчас или немного подождать?) с альтернативными сценариями, заготовленными для всех случаев. Телевидение тоже может стать интерактивным — с участием аудитории в викторинах, с возможностью выбора победителя и т. п.

С другой стороны, может быть видео по заказу и не станет главным направлением индустрии сетевых развлечений. Возможно, им станут сетевые игры. Уже сейчас существуют игры-симуляторы в реальном времени с большим количеством участников — например, прятки в виртуальном средневековом замке или симуляторы полетов, в которых одна команда пытается сбить игроков команды противника. Если в такие игры играть в специальных стереоскопических очках при фотографическом качестве двигающихся в реальном времени трехмерных образов, мы будем иметь что-то вроде всемирной виртуальной реальности совместного доступа.

Четвертой выделенной нами категорией является электронная коммерция в самом широком смысле слова. Закупка продуктов и вещей для дома через Интернет уже давно стала привычным делом. У покупателя появился богатый выбор, поскольку компаний, предлагающих приобрести что-либо по веб-каталогу, с каждым днем становится все больше, и счет уже идет на тысячи. Некоторые из этих каталогов в скором времени наверняка будут предлагать рекламные видеоролики, которые можно будет посмотреть, щелкнув на названии товара. Если клиент купил некий продукт, но не знает, что с ним делать, ему поможет сетевая служба поддержки.

Электронная коммерция развивается и в другом серьезном направлении: организуется доступ к финансовым учреждениям. Многие уже давно оплачивают свои векселя, управляют банковскими счетами и работают с вкладами с помощью соответствующих сетевых служб. По мере прогресса в области защиты передаваемой информации эти службы, безусловно, будут развиваться.

Еще одна область, которую вряд ли кто-то мог предвидеть, — это электронные «блошиные» рынки. Онлайновые аукционы, на которых распродают подержанные вещи, стали довольно мощной сетевой индустрией. В отличие от традиционных форм электронной коммерции, использующих модель «клиент-сервер», такие разновидности бизнеса ближе к равноранговым сетям (как бы по формуле «клиент — клиенту»). Некоторые формы сетевой коммерции со временем обрели сокращенные обозначения, которые приведены в табл. 1.1. (Английский предлог *to*, отвечающий на вопросы *«кому?»*, *«чему?»*, произносится так же, как *2* (*two*). Исходя из этого и выбирались обозначения.)

Таблица 1.1. Некоторые формы электронной коммерции

Обозначение	Полное название	Пример
B2C	Коммерсант клиенту (Business-to-Consumer)	Заказ книг в режиме on-line
B2B	Коммерсант коммерсанту (Business-to-Business)	Производитель автомобилей заказывает покрышки у поставщика
G2C	Государство клиенту (Government-to-Client)	Распространение бланков квитанций через Интернет
C2C	Клиент клиенту (Client-to-Client)	Продажа подержанных вещей
P2P	Равноранговые сети (Peer-to-Peer)	Совместное пользование файлами

Вне всякого сомнения, возможности компьютерных сетей будут со временем расширяться. Причем направления развития могут быть самыми неожиданными. Ну кто 10 лет назад мог предвидеть, что подростки будут с удовольствием заниматься таким скучным делом, как обмен SMS-сообщениями, а автобусные компании, перевозящие этих подростков, будут способствовать получению телефонными компаниями немалой прибыли?

Компьютерные сети объединили людей, которые оказались в силу разных обстоятельств отделены друг от друга своим географическим расположением. Люди, живущие далеко от цивилизации, но имеющие компьютер и доступ в Интернет, могут почувствовать себя городскими жителями, быть в курсе всех событий и жить полной современной жизнью. Электронные образовательные программы могут сильно повлиять на образование вообще; университеты могут запросто обслуживать учащихся со всей страны или планеты. Телекоммуникации в области медицины находятся на начальной стадии развития (хотя уже существуют системы, позволяющие удаленно контролировать состояние пациента), но и здесь они могут со временем начать играть большую роль. Однако настоящей приманкой для пользователей должно быть что-нибудь приземленное и в то же время сногсшибательное, например использование веб-камеры в холодильнике для определения момента, когда количество пакетов молока в нем дойдет до минимума.

Использование беспроводных сетей

Мобильные компьютеры — ноутбуки и карманные компьютеры (PDA) — это еще одна область, в которой сейчас наблюдается бурное развитие. У их владельцев обычно имеется настольный компьютер на работе, но они хотят поддерживать постоянную связь с домашней «базой». Так как обычные сети, в которых информация передается по проводам, невозможно использовать в машине или самолете, люди задумываются о беспроводных сетях. В этом разделе мы вкратце рассмотрим вопросы их применения.

Итак, для чего же реально требуются беспроводные сети? Самое популярное применение — мобильный офис. Многим людям бывает необходимо, находясь в пути, совершать телефонные звонки, отправлять и принимать факсы и электронную почту, иметь доступ в Интернет и к удаленным машинам. Все эти возможности должны быть предоставлены независимо от местонахождения пользователя. Например, организаторы компьютерных конференций зачастую используют мобильные сети. Имея ноутбук и беспроводный модем, можно подключиться к Интернету и получить доступ к информации в том же объеме, что и при использовании настольного компьютера. В некоторых университетах в студенческих городках организуются беспроводные сети, и тогда студент, сидя в сквере под деревом, может посещать виртуальную академическую библиотеку или читать электронную почту.

Такого рода сети уже давно и с успехом применяются в больших компаниях, занимающихся грузоперевозками, в таксопарках, службах доставки почты и ремонта. Там это жизненно необходимо — как для отслеживания пути передвиже-

ния транспорта и груза, так и для поддержания постоянной связи с диспетчерами. Например, во многих местах водители такси являются частными предпринимателями, не относящимися к какому-либо таксопарку. У них в машинах имеется электронное табло. Когда на централизованный пульт поступает заявка, диспетчер вводит данные о местонахождении пассажира и требуемом месте назначения. Информация появляется одновременно на экранах у всех водителей и сопровождается звуковым сигналом. Тот водитель, который первым нажмет кнопку, считается принявшим заявку.

Беспроводные сети находят широкое применение и в военном деле. При ведении боевых действий в совершенно произвольном месте планеты не приходится рассчитывать на инфраструктуру местных сетей связи и нужно организовывать свою сеть.

Хотя мобильные компьютеры и беспроводные сети тесно связаны между собой, все же это не одно и то же. Это отражено в табл. 1.2. Можно заметить, что есть разница между **стационарными** и **мобильными беспроводными сетями**. Даже ноутбуки иногда подключают к обычной компьютерной сети. Если пользователь, скажем, в номере гостиницы подключит его к телефонной розетке, он получит мобильность при отсутствии беспроводной сети.

С другой стороны, и наличие беспроводной сети еще не говорит о наличии мобильности. В качестве примера можно привести компанию, арендующую помещение в старом здании, в котором отсутствует проводка, необходимая для компьютерной сети. Тем не менее, этой компании нужно как-то связать между собой компьютеры, вот и приходится организовывать беспроводную сеть. А это несколько сложнее, чем покупка маленькой коробочки, напичканной электроникой, и включение ее в розетку. Как ни странно, беспроводная сеть в этом случае может обойтись дешевле, поскольку прокладка кабеля — довольно дорогое удовольствие.

Таблица 1.2. Сочетания беспроводных сетей и мобильных компьютеров

Беспроводная сеть	Мобильность	Применение
Нет	Нет	Настольные компьютеры в офисах
Нет	Есть	Ноутбук в номере гостиницы
Есть	Нет	Сети в старых зданиях, в которых не проложена проводка
Есть	Есть	Мобильный офис; PDA, хранящий информацию о товарах

Разумеется, есть и полноценные мобильные применения беспроводных сетей — от мобильных офисов до PDA, которые не дают людям в супермаркете забыть, за какими товарами они сюда пришли. Во многих крупных и загруженных аэропортах служащие, принимающие на стоянках арендованные машины, зачастую используют портативные компьютеры. Они набирают государственный номер машины, связываются с главным компьютером, получают от него информацию, а встроенный в PDA принтер сразу же распечатывает счет.

По мере развития подобных технологий возникают все новые их применения. Рассмотрим некоторые из них. Электронные счетчики, установленные на город-

ских стоянках автотранспорта, выгодны как владельцам автомобилей, так и государству. Такие счетчики могут работать с кредитными карточками, запрашивая их подтверждение по беспроводной сети. Когда оплаченное время стоянки заканчивается, проверяется наличие на стоянке машины (принимается отраженный от нее сигнал), и если она все еще там, об этом сообщается в полицию. Подсчитано, что таким образом в американский бюджет ежегодно поступает до 10 миллиардов долларов (Harte и др., 2000). Более того, это благотворно влияет на окружающую среду, так как водитель, уверенный в том, что он, как всегда, опоздает к окончанию времени стоянки и ему снова придется платить штраф, лучше воспользуется общественным транспортом.

Торговые автоматы, в которых продается какая-нибудь еда или напитки, можно встретить повсеместно. Как ни странно, продукты в них не появляются сами по себе. Время от времени приезжает служащий и заполняет их. Раз в день автомат отправляет по беспроводной линии связи на центральный компьютер электронный отчет, в котором сообщается, какая продукция требуется в каком количестве. Конечно, для этого можно использовать и обычную телефонную линию, но, знаете ли, подводить к каждому торговому автомату свою телефонную линию ради одного звонка в день — это слишком шикарно.

Еще одна область, в которой использование беспроводных сетей могло бы сэкономить деньги, — это снятие показаний счетчиков коммунальных услуг. Действительно, если бы счетчики электричества, газа, воды и т. д. самостоятельно отправляли данные, не пришлось бы оплачивать труд сотрудников, которые ходят по домам и записывают показания. Дымоуловители могут позвонить в пожарную службу вместо того, чтобы включать сирену, которая становится особенно бессмысленной, если дома никого нет. Стоимость радиопередающих и принимающих устройств снижается, как и плата за эфирное время, а значит, все больше и больше разнообразных применений можно реализовать без существенных финансовых затрат.

Совершенно иной прикладной областью, связанной с радиопередачей цифрового сигнала, является интеграция мобильных телефонов и PDA. Первой реализацией этой идеи стали крохотные PDA, которые могли отображать веб-странички на своих еще более крохотных экранах. Эта система основывалась на совершенно новом протоколе **WAP 1.0** (протокол, предназначенный для распространения информационных материалов по сети Интернет). Но он канул в Лету из-за того, что экраны были слишком маленькими, пропускная способность канала была низкой, да и качество предоставляемых услуг оставляло желать лучшего. Новые устройства подобного рода уже используют протокол WAP 2.0.

Одна из областей, в которой такая технология может иметь большой успех, это **мобильная коммерция (m-commerce)**. Движущей силой развития этого направления является стремление производителей PDA и сетевых операторов ухватить кусок пирога электронной коммерции. Одна из надежд, лелеемых ими, — использование беспроводных PDA для осуществления банковских операций и покупок в интернет-магазинах. Такие устройства и впрямь можно использовать в качестве электронного бумажника, оплачивая покупки даже в обычных магазинах. Это уже новое слово по сравнению с кредитными картами и даже смарт-кар-

тами. Реальная оплата при этом может выражаться в увеличении суммы счета за мобильный телефон. Магазину это выгодно, поскольку на данный момент такая схема позволяет избежать оплаты услуг за совершение операций с кредитными картами. Впрочем, есть и оборотная сторона вопроса, вовсе не выгодная для магазина: клиент, прежде чем совершить покупку, с помощью своего PDA может узнать, где выбранные им товары можно купить дешевле. Более того, PDA могут быть снабжены небольшим встроенным сканером для чтения штрихкода продукции и получения детальной информации о том, где еще и по какой цене она продается.

Поскольку сетевой оператор знает, где находится пользователь, он может предложить некоторые услуги, опирающиеся на это знание. Например, можно узнать, где находится ближайший книжный магазин или китайский ресторан. Еще одна замечательная возможность — мобильные карты местности. А как вам нравятся очень локализованные прогнозы погоды? Удобное, наверное, спросить: «Когда же, наконец, перестанет лить дождь у меня во дворе?» Нет никаких сомнений в том, что со временем появятся и другие интересные применения мобильных сетей.

У операторов мобильных сетей, а значит, и у м-коммерции есть одно замечательное преимущество. В отличие от пользователей Интернета, абоненты мобильных телефонов привыкли за все платить. Если на каком-нибудь сайте промелькнет упоминание о том, что за оплату с помощью кредитной карты будут взиматься какие-то сборы, то посетители поднимут большой шум. Если же оператор мобильной связи за небольшую плату любезно предоставит возможность оплатить покупки в магазине с помощью телефона, наверное, это будет воспринято нормально. Впрочем, время покажет.

В будущем, вероятно, будут развиваться технологии, основанные на всеобщей тенденции миниатюризации вычислительной техники. Это касается персональных сетей и носимых компьютеров. Фирмой IBM уже разработаны наручные часы со встроенной операционной системой Linux (включая систему управления окнами X11). Они имеют выход в Интернет, с их помощью можно отправлять и получать электронную почту (Narayanaswami и др., 2002). В будущем можно будет обмениваться визитными карточками, просто показывая друг другу свои часы. Подобные устройства могут служить персональными электронными идентификаторами для прохода в различные секретные помещения с ограниченным доступом. Такую роль сейчас играют специальные магнитные карты. Идентификацию можно будет дополнительно осуществлять с помощью ввода PIN-кода или снятия каких-нибудь биометрических показаний. Такие хитрые часы можно будет использовать для определения собственного местоположения и поиска каких-нибудь близлежащих объектов (например, ресторанов). Возможности ограничены только человеческой фантазией.

К умным часам со встроенным радиоприемником мы начали привыкать тогда, когда о них впервые рассказал в юмористическом монологе Дик Трэйси (Dick Tracy) в 1946 году. Но привыкнуть к тому, что уже сейчас появляется «умная пыль», очень непросто. Ученые из Беркли смогли разместить беспроводной компьютер в кубике со стороной 1 мм (Warneke и др., 2001). Потенциальные применения таких микроскопических устройств включают в себя сопровождение товаров, маленьких птичек, грызунов и даже насекомых!

Социальный аспект

Широкое распространение сетей вызвало новые социальные, этические и политические проблемы. Рассмотрим кратко лишь некоторые из них, так как подробное обсуждение потребует, как минимум, отдельной книги. Популярной чертой многих сетей являются конференции или электронные доски объявлений, где люди могут обмениваться сообщениями по самым различным темам. До тех пор, пока обсуждаемый предмет не выходил за рамки техники или увлечений вроде возделывания огородов, проблем, которые могут возникнуть, было не так уж много.

Проблемы начались с возникновением конференций, посвященных темам, по-настоящему волнующим людей, таким как политика, религия или секс. Взгляды, излагаемые одними людьми, могут оказаться оскорбительными для других. Они и в самом деле зачастую далеки от норм политкорректности. Кроме того, сетевые технологии, как известно, не ограничены только лишь передачей текста. Без особых проблем по Сети ходят фотографии с высоким разрешением и даже видеофрагменты. Некоторые люди придерживаются позиции «живи и дай жить другим», однако другие считают, что помещение в сети некоторых материалов (например, детской порнографии) просто недопустимо. Законодательства разных стран имеют разные взгляды на эту проблему; таким образом, страсти накаляются.

Люди подают в суд на сетевых операторов, считая их ответственными за содержимое сайтов, подобно тому как газеты и журналы несут ответственность за содержимое своих страниц. В ответ же операторы сетей утверждают, что сеть подобна телефонной компании или почтовому отделению и они не могут отвечать за то, что говорят их клиенты, а тем более управлять содержанием этих разговоров. Более того, если бы операторы были обязаны подвергать сообщения цензуре, им пришлось бы удалять все сообщения, которые оставляют даже малейшую возможность судебного иска, и, таким образом, нарушают права пользователей на свободу слова. Можно с уверенностью сказать, что подобные дебаты будут тянуться еще довольно долго.

Еще одной областью конфликтов оказались права наемных работников, вступившие в противоречие с правами нанимателей. Некоторые наниматели считают себя вправе читать и, возможно, подвергать цензуре сообщения своих работников, включая сообщения, посланные с домашних терминалов после работы. Не все с этим согласны.

Если даже наниматель имеет право обращаться подобным образом с корреспонденцией своих работников, то как же быть с государственными университетами и их студентами? А как насчет школ и учеников? В 1994 году университет Карнеги—Меллона (Carnegie—Mellon University) решил не пропускать через сеть входящий поток сообщений для некоторых конференций, посвященных вопросам секса, поскольку университет счел данный материал неприемлемым для студентов младших курсов (то есть для студентов моложе 18 лет). Пройдет не один год, пока эти события забудутся.

Очень серьезной проблемой являются взаимоотношения государства и граждан. Известно, что ФБР установило на серверах многих поставщиков услуг Интернета системы слежения за всеми входящими и исходящими сообщениями, что должно, по мнению США, служить интересам государственной безопасности

(Blaze, Bellovin, 2000; Sobel, 2001; также Zacks, 2001). Система изначально называлась **Carnivore** (что в переводе означает «хищник». — *Примеч. перев.*), однако такое зловещее название обращало на себя слишком много внимания общественности. Было решено переименовать систему и назвать ее невинным именем — DCS1000. Но ее цель от этого не изменилась. «Хищник» следит за миллионами людей в надежде найти в их письмах намек на незаконную деятельность. К сожалению, Четвертая поправка к Конституции США запрещает не обеспеченную какими-либо доказательствами слежку. Вопрос о том, имеют ли эти 54 слова какой-либо вес в XXI веке, судя по всему, будет обсуждаться еще и в XXII веке.

Хотелось бы подчеркнуть, что государство не имеет права угрожать частной жизни людей. Впрочем, подобными нелицеприятными действиями занимается не только ФБР, но и самые обыкновенные веб-дизайнеры. Взять хотя бы cookie-файлы, содержащие информацию о том, что пользователь делал в Сети, и позволяющие нечистым на руку компаниям узнавать конфиденциальную информацию и передавать через Интернет номера кредитных карт и другие важные идентификаторы (Berghel, 2001).

Компьютерные сети предоставляют возможности для посылки анонимных сообщений. В некоторых ситуациях такая необходимость есть. Например, таким образом студенты, солдаты, служащие и граждане могут пожаловаться на незаконные действия профессоров, офицеров, начальства и политиков, не опасаясь репрессий. С другой стороны, в США и во многих других демократических странах законом особо предусмотрено право обвиняемой стороны на очную ставку со своим обвинителем в суде, а также на встречный иск. Поэтому анонимные обвинения не могут рассматриваться в качестве свидетельств в суде.

Итак, подобно печатному станку 500 лет назад, компьютерные сети предоставляют новые способы распространения гражданами их взглядов среди самой различной аудитории. Новая свобода распространения информации несет с собой и новые нерешенные политические, социальные и нравственные проблемы.

Все хорошее имеет свою оборотную сторону. Похоже, так устроена жизнь. Интернет позволяет с огромной скоростью находить нужную информацию, однако кто проверит ее качество и достоверность? Совет врача, которого так ждет тяжелобольной человек, может на самом деле исходить как от лауреата Нобелевской премии в области медицины, так и от разгильдяя, которого выгнали из школы и которому нечем заняться. Компьютерные сети породили новые виды антиобщественных преступлений. Электронная макулатура, «спам», стала, увы, частью нашей жизни, потому что есть способы собрать миллионы адресов e-mail и рассылать по ним все что угодно. Более того, списки этих адресов продаются на пиратских компакт-дисках, которые покупают горе-коммерсанты. Сообщения, передаваемые по электронной почте, могут также содержать злобные разрушительные вирусы.

Воровство конфиденциальной информации, к сожалению, тоже стало очень распространенным явлением. Новые воры ничего не взламывают физически. Они крадут лишь несколько казалось бы ничего не значащих символов. Эти символы оказываются, например, номерами кредитных карт, с которых вдруг таинственным образом начинают исчезать деньги. Наконец, возможность передачи через Интернет достаточно качественной аудио- и видеоинформации позволило заин-

тересованным лицам нарушать все мыслимые законы об авторских правах. А вычислить нарушителей оказалось делом очень непростым.

Многие из этих проблем могут быть решены, если компьютерная индустрия всерьез займется вопросами защиты информации. Если бы все сообщения передавались в зашифрованном виде, это позволило бы избежать огромных убытков, понесенных как частными лицами, так и крупными компаниями. Более того, системы кодирования уже давно разработаны, и мы подробно изучим их в главе 8. Проблема в том, что производители аппаратного и программного обеспечения прекрасно знают, каких денег стоит внедрение защитных систем, и понимают, что попытки продать такую дорогостоящую продукцию обречены. Немало хлопот доставляют и «баги» (ошибки в программах), «дыры» в защите и т. д. Они возникают потому, что производители добавляют все новые и новые функции, а это приводит к увеличению числа неполадок. Возможным выходом из такой ситуации является взимание платы за расширенные версии программ, однако поди заставь конторы, занимающиеся разработкой ПО, отказаться от такого хорошего рекламного хода, как бесплатные обновления. Можно, конечно, обязать фирмы возмещать убытки, нанесенные выпущенными ими дефектными программами, однако это приведет к банкротству практически всей программной индустрии в первый же год.

Сетевое оборудование

Теперь пора от вопросов применения сетей и социальных аспектов перейти к рассмотрению технической стороны разработки сетей. Единой общепринятой системы, которой удовлетворяют все сети, не существует, однако есть два важнейших параметра: технология передачи и размеры. Рассмотрим оба параметра по очереди.

Если смотреть в общих чертах, существует два типа технологии передачи:

- широковещательные сети;
- сети с передачей от узла к узлу.

Широковещательные сети обладают единым каналом связи, совместно используемым всеми машинами сети. Короткие сообщения, называемые в некоторых случаях пакетами, которые посылаются одной машиной, получают все машины. Поле адреса в пакете указывает, кому направляется сообщение. При получении пакета машина проверяет его адресное поле. Если пакет адресован этой машине, она его обрабатывает. Пакеты, адресованные другим машинам, игнорируются.

В качестве иллюстрации представьте себе человека, стоящего в конце коридора с большим количеством комнат и кричащего: «Ватсон, идите сюда. Вы мне нужны». И хотя это сообщение может быть получено (услышано) многими людьми, ответит только Ватсон. Остальные просто не обратят на него внимания. Другим примером может быть объявление в аэропорту, предлагающее всем пассажирам рейса 644 подойти к выходу номер 12.

Широковещательные сети также позволяют адресовать пакет одновременно всем машинам с помощью специального кода в поле адреса. Когда передается пакет с таким кодом, его получают и обрабатывают все машины сети. Такая опера-

ция называется **широковещательной передачей**. Некоторые широковещательные системы также предоставляют возможность посылать сообщения подмножеству машин, и это называется **многоадресной передачей**. Одной из возможных схем реализации этого может быть резервирование одного бита для признака многоадресной передачи. Оставшиеся $n-1$ разрядов адреса могут содержать номер группы. Каждая машина может «подписаться» на одну, несколько или все группы. Когда пакет посылается определенной группе, он доставляется всем машинам, являющимся членами этой группы.

Сети с передачей от узла к узлу, напротив, состоят из большого количества соединенных пар машин. В сети подобного типа пакету, чтобы добраться до пункта назначения, необходимо пройти через ряд промежуточных машин. Часто при этом существует несколько возможных путей от источника до получателя, поэтому алгоритмы вычисления таких путей играют очень важную роль в сетях с передачей от узла к узлу. Обычно (хотя имеются и исключения) небольшие, географически локализованные в одном месте сети используют широковещательную передачу, тогда как в более крупных сетях применяется передача от узла к узлу. В последнем случае имеется один отправитель и один получатель, и такую систему иногда называют **однонаправленной передачей**.

Другим признаком классификации сетей является их размер. На рис. 1.4 приведена классификация мультипроцессорных систем в зависимости от их размеров. В верхней строке таблицы помещаются **персональные сети**, то есть сети, предназначенные для одного человека. Примером может служить беспроводная сеть, соединяющая компьютер, мышь, клавиатуру и принтер. Устройство типа PDA, контролирующее работу слухового аппарата или являющееся кардиостимулятором, тоже попадает в эту категорию. Далее в таблице следуют более протяженные сети. Их можно разделить на локальные, муниципальные и глобальные сети. И замыкают таблицу объединения двух и более сетей.

Расстояние между процессорами	Процессоры расположены	Пример
1 м	На одном квадратном метре	Персональная сеть
10 м	Комната	Локальная сеть
100 м	Здание	
1 км	Кампус	
10 км	Город	Муниципальная сеть
100 км	Страна	Глобальная сеть
1000 км	Континент	
10 000 км	Планета	Интернет

Рис. 1.4. Классификация многопроцессорных систем по размеру

Хорошо известным примером такого объединения выступает Интернет. Размеры сетей являются весьма важным классификационным фактором, поскольку в сетях различного размера применяется различная техника. В данной книге мы рассмотрим сети всех размеров, а также их объединения. Далее мы дадим краткое описание сетевого оборудования.

Локальные сети

Локальными сетями называют частные сети, размещающиеся, как правило, в одном здании или на территории какой-либо организации площадью до нескольких квадратных километров. Их часто используют для объединения компьютеров и рабочих станций в офисах компании или предприятия для предоставления совместного доступа к ресурсам (например, принтерам) и обмена информацией. Локальные сети отличаются от других сетей тремя характеристиками: размерами, технологией передачи данных и топологией.

Локальные сети ограничены в размерах — это означает, что время пересылки пакета ограничено сверху и этот предел заранее известен. Знание этого предела позволяет применять определенные типы разработки, которые были бы невозможны в противоположном случае. Кроме того, это упрощает управление локальной сетью.

В локальных сетях часто применяется технология передачи данных, состоящая из единственного кабеля, к которому присоединены все машины. Это подобно тому, как раньше в сельской местности использовались телефонные линии. Обычные локальные сети имеют пропускную способность канала связи от 10 до 100 Мбит/с, невысокую задержку (десятые доли микросекунды) и очень мало ошибок. Наиболее современные локальные сети могут обмениваться информацией на более высоких скоростях, доходящих до 10 Гбит/с. В этой книге мы будем придерживаться традиции и указывать скорость линий в мегабитах в секунду (1 Мбит состоит из 1 000 000 бит) и в гигабитах в секунду (1 Гбит равен 1 000 000 000 бит).

В широковещательных локальных сетях могут применяться различные топологические структуры. На рис. 1.5 показаны две из них. В сети с общей шиной (линейный кабель) в каждый момент одна из машин является хозяином шины (master) и имеет право на передачу.

Все остальные машины должны в этот момент воздержаться от передачи. Если две машины захотят что-нибудь передавать одновременно, то возникнет конфликт, для разрешения которого требуется специальный механизм. Этот механизм может быть централизованным или распределенным. Например, стандарт IEEE 802.3, называемый **Ethernet**, описывает широковещательную сеть с топологией общей шины с децентрализованным управлением, работающую на скоростях от 10 Мбит/с до 10 Гбит/с. Компьютеры в сети Ethernet могут выполнять передачу в любое время. При столкновении двух или более пакетов каждый компьютер просто ждет в течение случайного интервала времени, после которого снова пытается передать пакет.

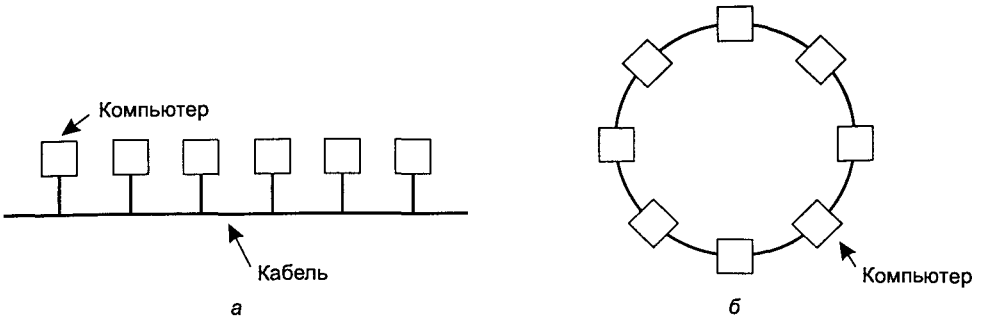


Рис. 1.5. Широковещательные сети: шина (*а*); кольцо (*б*)

Вторым типом широковещательных сетей является кольцо. В кольце каждый бит передается по цепочке, не ожидая остальной части пакета. Обычно каждый бит успевает обойти все кольцо, прежде чем будет передан весь пакет. Как и во всех широковещательных сетях, требуется некая система арбитража для управления доступом к линии. Применяемые для этого методы будут описаны далее в этой книге. Стандарт IEEE 802.5 (маркерное кольцо) описывает популярную кольцевую локальную сеть, работающую на скоростях 4 и 16 Мбит/с. Еще одним примером кольцевой сети является FDDI (оптоволоконная сеть).

В зависимости от способа назначения канала широковещательные сети подразделяются на статические и динамические. При статическом назначении используется циклический алгоритм и все время делится между всеми машинами на равные интервалы, так что машина может передавать данные только в течение выделенного ей интервала времени. При этом емкость канала расходуется неэкономно, так как временной интервал предоставляется машинам независимо от того, есть им что сказать или нет. Поэтому чаще используется динамическое (то есть по требованию) предоставление доступа к каналу.

Методы динамического предоставления доступа к каналу также могут быть централизованными либо децентрализованными. При централизованном методе предоставления доступа к каналу должен существовать арбитр шины, определяющий машину, получающую право на передачу. Арбитр должен принимать решение на основании получаемых запросов и некоего внутреннего алгоритма. При децентрализованном методе каждая машина должна сама решать, передавать ей что-нибудь или нет. Можно подумать, что подобный метод обязательно приводит к беспорядку, однако это не так. Далее мы рассмотрим различные алгоритмы, специально созданные для внесения порядка в возможный хаос.

Муниципальные сети

Муниципальные сети (metropolitan area network, MAN) объединяют компьютеры в пределах города. Самым распространенным примером муниципальной сети является система кабельного телевидения. Она стала правопреемником обычных антенных телесетей в тех местах, где по тем или иным причинам качество эфира

было слишком низким. Общая антенна в этих системах устанавливалась на вершине какого-нибудь холма, и сигнал передавался в дома абонентов.

Вначале стали появляться специализированные, разработанные прямо на объектах сетевые структуры. Затем компании-разработчики занялись продвижением своих систем на рынке, начали заключать договоры с городским правительством и в итоге охватили целые города. Следующим шагом стало создание телевизионных программ и даже целых каналов, предназначенных только для кабельного телевидения. Зачастую они представляли какую-то область интересов. Можно было подписаться на новостной канал, спортивный, посвященный кулинарии, саду-огороду и т. д. До конца 90-х годов эти системы были предназначены исключительно для телевизионного приема.

Когда Интернет стал привлекать к себе массовую аудиторию, операторы кабельного телевидения поняли, что, внеся небольшие изменения в систему, можно сделать так, чтобы по тем же каналам в неиспользуемой части спектра передавались (причем в обе стороны) цифровые данные. С этого момента кабельное телевидение стало постепенно превращаться в муниципальную компьютерную сеть. В первом приближении систему MAN можно представить себе такой, как она изображена на рис. 1.6. На этом рисунке видно, что по одним и тем же линиям передается и телевизионный, и цифровой сигналы. Во **входном устройстве** они смешиваются и передаются абонентам. Мы еще вернемся к этому вопросу позднее.

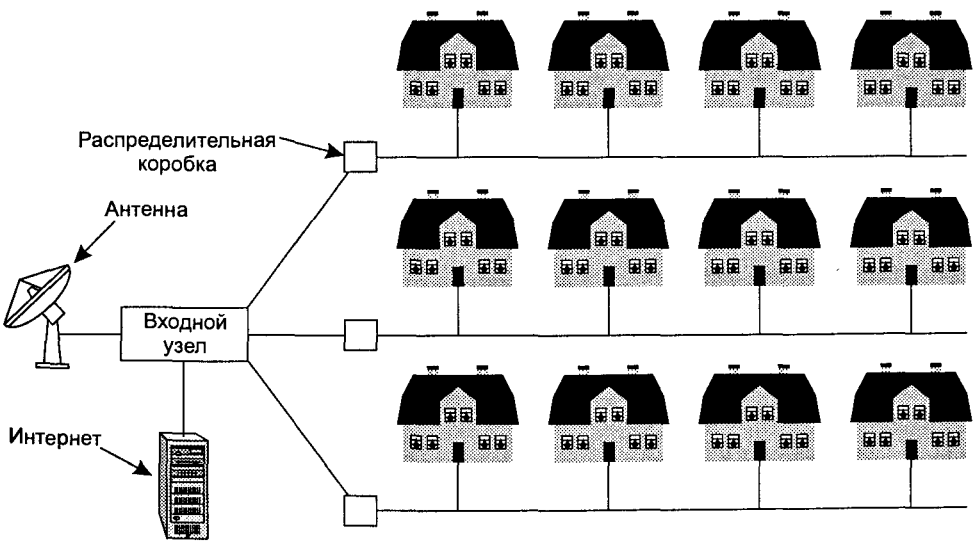


Рис. 1.6. Муниципальная сеть на базе кабельного ТВ

Впрочем, муниципальные сети — это не только кабельное телевидение. Недавние разработки, связанные с высокоскоростным беспроводным доступом в Интернет, привели к созданию других MAN, которые описаны в стандарте IEEE 802.16. Мы рассмотрим их более подробно в главе 2.

Глобальные сети

Глобальная сеть (wide area network, WAN) охватывает значительную географическую область, часто целую страну или даже континент. Она объединяет машины, предназначенные для выполнения программ пользователя (то есть приложений). Мы будем следовать традиционной терминологии и называть эти машины хостами. Хосты соединяются коммуникационными подсетями, называемыми для краткости просто подсетями. Хосты обычно являются собственностью клиентов (то есть просто клиентскими компьютерами), в то время как коммуникационной подсетью чаще всего владеет и управляет телефонная компания или поставщик услуг Интернета. Задачей подсети является передача сообщений от хоста хосту, подобно тому как телефонная система переносит слова от говорящего слушающему. Таким образом, коммуникативный аспект сети (подсеть) отделен от прикладного аспекта (хостов), что значительно упрощает структуру сети.

В большинстве глобальных сетей подсеть состоит из двух раздельных компонентов: линий связи и переключающих элементов. Линии связи, также называемые **каналами** или **магистралями**, переносят данные от машины к машине. Переключающие элементы являются специализированными компьютерами, используемыми для соединения трех или более линий связи. Когда данные появляются на входной линии, переключающий элемент должен выбрать выходную линию — дальнейший маршрут этих данных. В прошлом для названия этих компьютеров не было стандартной терминологии. Сейчас их называют маршрутизаторами (router), однако читателю следует знать, что по поводу терминологии в данном случае единого мнения не существует. К сожалению, многие остряки-самоучки любят рифмовать «router» с «doubter», что в переводе означает «скептик», а некоторые вместо «router» пишут «rooter» («корчеватель»).

В модели, показанной на рис. 1.7, каждый хост соединен с локальной сетью, в которой присутствует маршрутизатор, хотя в некоторых случаях хост может быть связан с маршрутизатором напрямую. Набор линий связи и маршрутизаторов (но не хостов) образует подсеть.

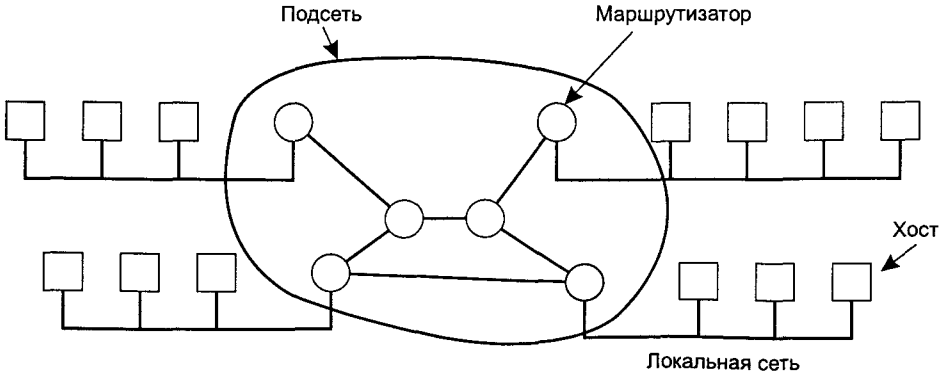


Рис. 1.7. Связь хостов и подсети в ЛВС

Следует также сделать замечание по поводу термина «подсеть» (subnet). Изначально его *единственным* значением являлся набор маршрутизаторов и линий связи, используемый для передачи пакета от одного хоста к другому. Однако спустя несколько лет этот термин приобрел второй смысл, связанный с адресацией в сети (что будет обсуждаться в главе 5). Таким образом, имеется некая двусмысленность, связанная с термином «подсеть». К сожалению, этому термину в его изначальном смысле нет никакой альтернативы, поэтому нам придется использовать его в обоих смыслах. По контексту всегда будет ясно, что имеется в виду.

Большинство глобальных сетей содержат большое количество кабелей или телефонных линий, соединяющих пару маршрутизаторов. Если какие-либо два маршрутизатора не связаны линией связи напрямую, то они должны общаться при помощи других маршрутизаторов. Когда пакет посылается от одного маршрутизатора другому через несколько промежуточных маршрутизаторов, он получается каждым промежуточным маршрутизатором целиком, хранится на нем, пока требуемая линия связи не освободится, а затем пересылается дальше. Подсеть, работающая по такому принципу, называется **подсетью с промежуточным хранением (store-and-forward)** или **подсетью с коммутацией пакетов (packet-switched)**. Почти у всех глобальных сетей (кроме использующих спутники связи) есть подсети с промежуточным хранением. Небольшие пакеты фиксированного размера часто называют **ячейками (cell)**.

О принципе организации сетей с коммутацией пакетов стоит сказать еще несколько слов, поскольку они используются очень широко. В общем случае, когда у процесса какого-нибудь хоста появляется сообщение, которое он собирается отправить процессу другого хоста, первым делом отправляющий хост разбивает последовательность на пакеты, каждый из которых имеет свой порядковый номер. Пакеты один за другим направляются в линию связи и по отдельности передаются по сети. Принимающий хост собирает пакеты в исходное сообщение и передает процессу. Продвижение потока пакетов наглядно показано на рис. 1.8.

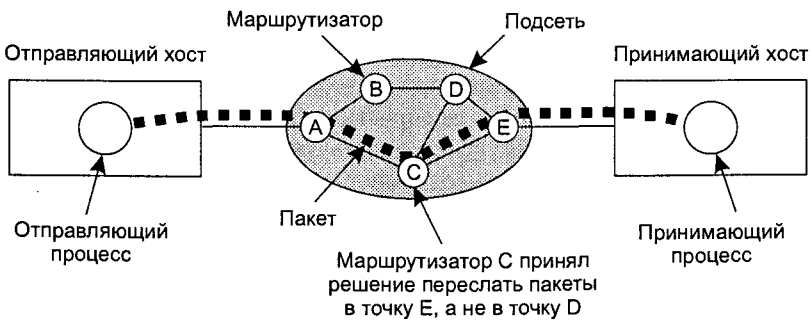


Рис. 1.8. Поток пакетов от отправляющего процесса к принимающему

На рисунке видно, что все пакеты следуют по пути ACE, а не ABDE или ACDE. В некоторых сетях путь всех пакетов данного сообщения вообще является строго определенным. В других сетях путь пакетов может прокладываться независимо.

Решения о выборе маршрута принимается на локальном уровне. Когда пакет приходит на маршрутизатор A, именно последний решает, куда его перенаправить — на B или на C. Метод принятия решения называется **алгоритмом маршрутизации**. Их существует огромное множество, и некоторые из них мы изучим в главе 5.

Не все глобальные сети используют коммутацию пакетов. Второй возможностью соединить маршрутизаторы глобальной сети является радиосвязь с использованием спутников. Каждый маршрутизатор снабжается антенной, при помощи которой он может принимать и посылать сигнал. Все маршрутизаторы могут принимать сигналы со спутника, а в некоторых случаях они могут также слышать передачи соседних маршрутизаторов, передающих данные на спутник. Иногда все маршрутизаторы соединяются обычной двухточечной подсетью, и только некоторые из них снабжаются спутниковой антенной. Спутниковые сети являются широковещательными и наиболее полезны там, где требуется широковещание.

Беспроводные сети

Идея цифровой беспроводной связи не нова. Уже в 1901 году итальянский физик Гульельмо Маркони (Guglielmo Marconi) продемонстрировал телеграфную связь между кораблем и берегом при помощи азбуки Морзе, состоящей из точек и тире, что весьма похоже на двоичный код. Сегодняшние цифровые радиосистемы обладают более высокой производительностью, однако в их основе лежит та же идея.

В первом приближении беспроводные сети можно разбить на следующие три категории:

- взаимодействующие системы;
- беспроводные ЛВС (LAN);
- беспроводные глобальные сети (WAN).

Под взаимодействующими системами понимается прежде всего связывание между собой компонентов компьютера с использованием радиоволн малого радиуса действия. Почти любой компьютер состоит из нескольких частей: монитора, клавиатуры, мыши, принтера... Каждое из этих внешних устройств, как известно, подсоединяется к системному блоку с помощью кабелей. А знаете, сколько проблем с подключением этой техники возникает у новичков, несмотря на маркировку разъемов и подробные руководства по эксплуатации? Недаром же большинство фирм, торгующих компьютерами, предлагают услуги технической службы, заключающиеся только лишь в соединении компонентов системы. Несколько компаний одна за другой пришли к идее создания беспроводной системы **Bluetooth**, предназначенной для того, чтобы избавить компоненты компьютера от кабелей и разъемов. Кроме стандартных устройств, с помощью Bluetooth можно подключать к компьютеру цифровые камеры, гарнитуры, сканеры и др. То есть теперь практически любые цифровые устройства, располагающиеся недалеко от системного блока, можно соединить с ним беспроводной сетью. Никаких проводов, никаких разъемов, никаких драйверов. Нужно просто принести устройство, вклю-

чить его, и оно будет работать. Для многих начинающих пользователей такая простота — большой плюс.

В простейшем случае взаимодействие внутри системы подчиняется принципу «главный – подчиненный». Системный блок чаще всего выступает в роли главного устройства, а все прочие — в роли подчиненных. В чем заключается это главенство? Именно системный блок назначает адреса устройств, определяет моменты, в которые они могут «вещать», ограничивает время передачи, задает диапазоны рабочих частот и т. д. Мы обсудим технологию Bluetooth более подробно в главе 4.

Следующим шагом в развитии этого направления стали беспроводные ЛВС (локальные вычислительные сети). В них каждый компьютер оборудован радиомодемом и антенной, с их помощью он может обмениваться данными с другими компьютерами. Иногда есть общая антенна, расположенная на потолке, и передача данных происходит через нее, но если рабочие станции сети расположены достаточно близко, то обычно используют одноранговую конфигурацию. Беспроводные сети все шире используются в бизнесе и для домашних целей, где прокладывать Ethernet нет никакого смысла, а также в старых зданиях, арендуемых под офисы, в кафетериях, в офисных центрах, конференц-залах и других местах. Стандарт беспроводных сетей имеет маркировку **IEEE 802.11**, и именно он чаще всего реализуется и становится все более популярным. Его мы тоже обсудим в главе 4.

Третий тип беспроводных сетей используется в глобальных сетях. Примером может служить система сотовой связи, являющаяся на самом деле низкопроизводительной цифровой беспроводной сетью. Выделяют уже целых три поколения сотовой связи. Первые сотовые сети были аналоговыми и предназначались только для передачи речи. Второе поколение было уже цифровым, но ничего, кроме речи, передавать по-прежнему было нельзя. Наконец, нынешнее, третье поколение — цифровое, причем появилась возможность передачи как голоса, так и других данных. В некотором смысле, сотовые сети — это те же беспроводные ЛВС, разница лишь в зоне охвата и более низкой скорости передачи. Если обычные беспроводные сети могут работать со скоростью до 50 Мбит/с на расстоянии десятков метров, то сотовые системы передают данные на скорости 1 Мбит/с, но расстояние от базовой станции до компьютера или телефона исчисляется километрами, а не метрами. В главе 2 мы будем подробно обсуждать эти сети.

Сейчас развиваются не только низко-, но и высокопроизводительные глобальные беспроводные сети. Изначальная установка такова: необходимо организовать доступ в Интернет с нормальной скоростью, чтобы при этом не был задействован телефон. Такую услугу иногда называют локальной многоузловой системой распределения. Мы изучим ее несколько позже. Она уже даже имеет свой стандарт, IEEE 802.16, речь о котором пойдет в главе 4.

Почти все беспроводные сети в каком-то месте имеют шлюз, обеспечивающий связь с обычными компьютерными сетями, иначе просто невозможно было бы организовать, допустим, доступ в Интернет. Такие комбинации могут использоваться в самых разных видах и ситуациях. Скажем, на рис. 1.9, *а* изобра-

жен самолет, в котором несколько человек используют модемы и телефоны, установленные на спинках сидений, для связи с офисом. Каждый звонок производится независимо. Однако значительно эффективнее использовать летающую локальную сеть (рис. 1.9, *б*). В этом случае каждое кресло оборудуется розеткой Ethernet, к которой пассажир может подключить свой компьютер. Радиосвязь осуществляется единым маршрутизатором самолета, который связывается с одним из маршрутизаторов, находящихся на земле. Последние сменяют друг друга по мере перемещения самолета. Конфигурация представляет собой обычную локальную сеть, с той разницей, что связь с внешним миром осуществляется не по кабелю, а по радио.

Многие верят в то, что именно за беспроводными технологиями будущее сетевых технологий (см., например, Bi и др., 2001; Leeper, 2001; Varshey и Vetter, 2000), однако есть по крайней мере один человек, который с этим не согласен. Так, например, изобретатель Ethernet Боб Меткалф (Bob Metcalfe) писал в 1995 году: «Переносные беспроводные компьютеры подобны переносным душевым кабинкам без труб. Они будут весьма популярны на транспортных средствах, стройплощадках и рок-концертах. Но я советую вам протянуть кабель к своему дому и оставаться там». История, возможно, сохранит эту запись в том же разделе, что и высказывание главы IBM Т. Дж. Уотсона (T. J. Watson), который в 1945 году объяснял, почему его фирма не стремится заниматься компьютерами: «По крайней мере, до 2000 года миру хватит для удовлетворения его потребностей четырех-пяти компьютеров».

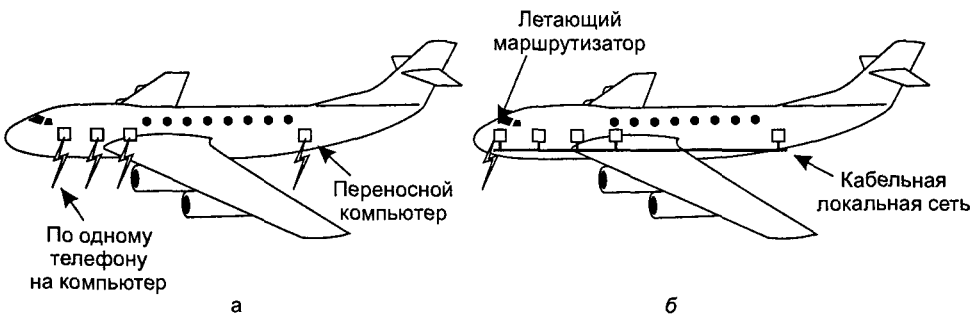


Рис. 1.9. Индивидуальные переносные компьютеры (а); летающая ЛВС (*б*)

Домашние сети

Домашние сети постепенно входят в нашу жизнь. В будущем, скорее всего, для этого окажутся приспособлены практически все жилища и почти все бытовые приборы, не говоря уже о компьютерах, смогут обмениваться данными между собой и будут доступны через Интернет. Это будет очередная концепция, которая, как сейчас кажется, никому не нужна (как, например, удаленное управление телевизором или мобильным телефоном), но когда она станет реальностью, все начнут недоумевать, как же они до сих пор жили без этого.

В сеть можно объединить больше устройств, чем кажется на первый взгляд. Вот самые очевидные из них:

- компьютеры (настольные ПК, ноутбуки, PDA, периферийные устройства совместного доступа);
- приборы для развлечений (телевизоры, DVD, VCR, различные видеокамеры, аудиосистемы, MP3-проигрыватели);
- телекоммуникации (телефоны, мобильные телефоны, системы двухсторонней связи, факсы);
- бытовые приборы (микроволновые печи, холодильники, часы, отопительные системы, кондиционеры, системы освещения);
- измерительные приборы (счетчики, системы пожарной сигнализации, термостаты, камеры).

Домашние сети уже реально существуют, правда, широкое распространение пока что получают только более или менее традиционные решения. Во многих домах установлены устройства для подключения компьютеров к общей линии высокоскоростного доступа в Интернет. Развлекательные устройства пока еще в сети не объединяют, но чем больше цифровых аудио- и видеозаписей можно будет найти в Интернете, тем больше будет потребность в таких сетях. К тому же человек устроен так, что ему нужно делиться информацией, поэтому, например, снятую самостоятельно видеозапись в любом случае захочется отправить друзьям или родственникам, а значит, данные должны уметь ходить по Сети в обе стороны. Средства телекоммуникаций, в принципе, организованы в виде сетей и уже имеют связь с внешним миром, но не исключено, что скоро они станут полностью цифровыми и будут работать через Глобальную Сеть. В доме среднего обывателя сегодня насчитывается до дюжины часов разных калибров (в том числе встроенных в бытовые приборы). Два раза в год их приходится переставлять на час вперед или назад. Если бы все они были подключены к Интернету, то это могло бы происходить автоматически. Наконец, многим людям бывает необходимо наблюдать за своим домом и за тем, что в нем происходит. Несомненно, нашлись бы родители, готовые потратить некоторую сумму денег на то, чтобы установить систему, позволяющую во время ужина в ресторане наблюдать за своим спящим дома малышом с помощью PDA. Можно помечтать не только об отдельных сетях, созданных для конкретных применений, но и о единой интегрированной сети, и это, вероятно, лучшее, что можно придумать.

Домашние сети должны обладать некоторыми специфическими свойствами, присущими им одним. Во-первых, вся сеть и отдельные ее компоненты должны устанавливаться предельно просто. Автору в течение многих лет пришлось заниматься установкой различных устройств и программного обеспечения на разные компьютеры; результаты были всякие. После определенного количества телефонных звонков в службу поддержки удалось создать определенную классификацию ответов, получаемых на любые вопросы. Практически все ответы сводятся к следующему:

- прочитайте инструкцию;
- перезагрузите компьютер;

- отключите от компьютера все оборудование, кроме нашего, и попробуйте еще раз;
- скачайте с нашего веб-сайта свежие драйверы.
- если же все это не помогает, то отформатируйте жесткий диск и переустановите Windows с лицензионного компакт-диска.

Если продавцы холодильников начнут рассказывать своим клиентам по телефону про какие-то драйверы, боюсь, их могут неправильно понять. Это пользователи компьютеров привыкли, купив какую-нибудь железку, идти из компьютерного магазина с замиранием сердца, гадая, будет она работать или нет. Публика, покупающая только автомобили, телевизоры и холодильники, не столь толерантна. Она всегда рассчитывает только на стопроцентную вероятность того, что купленная вещь будет работать.

Во-вторых, сетевые устройства должны иметь достаточно мощную «защиту от дурака». Кондиционеры, имеющие всего один переключатель с четырьмя позициями (ВЫКЛЮЧЕН, СЛАБО, СРЕДНЕ и СИЛЬНО), снабжаются инструкцией на тридцати страницах. Если такое устройство будет сетевым, то только на раздел, касающийся техники безопасности, придется отвести тридцать страниц. Только в этом случае можно гарантировать, что большинство пользователей поймут, как безопасно пользоваться кондиционером.

В-третьих, для успеха важна низкая цена. По крайней мере, добавление сетевых возможностей не должно существенно влиять на конечную стоимость продукции. Никто не станет платить $50 сверх обычной цены за интернет-термостат, потому что людям тяжело будет объяснить необходимость наблюдения за температурой в квартире с работы. Если же надбавка составит долларов пять, то такое устройство найдет своего покупателя.

В-четвертых, основное применение домашних сетей, скорее всего, будет связано с мультимедиа. Это означает, что скорость передачи должна быть достаточно высока. Интернет-телевизорам, которые смогут показывать кино с разрешением 320×240 при средней скорости 10 кадров в секунду, путь на рынок закрыт. Быстрый Ethernet — рабочая лошадка офисных компьютерных сетей — не справится с потоковой передачей мультимедиа. Следовательно, технологии домашних сетей должны развиваться в двух противоречивых направлениях: они должны быть одновременно быстрее и дешевле, чем офисные сети. Только тогда их можно представлять как массовую продукцию.

В-пятых, домашние сети должны обладать свойством наращиваемости, чтобы можно было начать с одного-двух устройств, а затем иметь возможность без проблем увеличивать их количество. Это означает, что не должно быть даже речи о «войнах форматов». В этом секторе рынка не пройдет такой номер, когда можно всем сказать, что самый главный стандарт — это IEEE 1394 (FireWire), а через пару лет вдруг заявить, что USB 2.0 — это самый популярный интерфейс месяца и все должны срочно переходить на устройства с его поддержкой. Нет. Здесь интерфейсы должны оставаться неизменными в течение многих лет, а кабели (если таковые будут) должны прокладываться с расчетом на работу десятилетиями.

В-шестых, очень важны защита информации и надежность подобных сетей. Одно дело, если несколько файлов исчезнут из-за вируса, присланного по e-mail, и совсем другое, если вору удастся с помощью своего PDA взломать защиту вашего жилища и разорить его.

Интересен вопрос о том, будут ли в домашних сетях применяться кабели или же сети будут беспроводными. В большинстве домов и так уже установлено шесть сетей: электрическая, телефонная, кабельного телевидения, газовая, водопроводная и канализационная. Добавить седьмую во время постройки здания труда не составит, однако переделывать уже существующие дома тяжеловато и дороговато. С точки зрения экономии средств привлекательнее беспроводные сети, но с точки зрения защиты информации — кабельные. Еще одно неприятное обстоятельство, касающееся беспроводных систем, заключается в том, что радиоволны легко проходят сквозь преграды и могут создавать помехи. Не всем понравится слушать по радио, как пищит, устанавливая соединение, модем соседей. В главе 8 мы обсудим методы шифрования, обеспечивающие защиту информации, но в случае домашних сетей эта защита должна быть одновременно «защитой от дурака», которая сработает даже в том случае, если с техникой играет ребенок или неопытный пользователь. Об этом, конечно, легко рассуждать, но не так просто сделать, даже если рассчитывать на достаточно разумного пользователя.

В общем, у домашних сетей еще все впереди. Здесь много возможностей и сложностей, которые предстоит преодолеть. От большинства из домашних сетей требуются простота управления, надежность, защищенность, особенно в руках пользователей, далеких от техники. В то же время необходима высокая производительность при низкой цене.

Объединения сетей

Существующие ныне сети часто используют различное оборудование и программное обеспечение. Люди, связанные с одной сетью, хотят общаться с людьми, подключенными к другой. Для выполнения этого желания необходимо объединить вместе различные и часто несовместимые сети. С этой целью иногда используются машины, называемые **шлюзами**, обеспечивающие соединение и необходимое преобразование в терминах как аппаратуры, так и программного обеспечения. Набор соединенных сетей называется **объединенной сетью** или просто **интерсетью**. Обратите внимание на то, что слово «интерсеть» (internet, написанный со строчной буквы) всегда будет использоваться в этой книге в его исконном смысле, в отличие от слова «Интернет» (с прописной буквы).

Обычной формой объединенных сетей является набор локальных сетей, объединенных при помощи глобальной сети. Действительно, если заменить надпись «подсеть» на рис. 1.7 «глобальной сетью», то в этом рисунке больше ничего не надо будет менять. Единственное техническое различие между подсетью и глобальной сетью заключается в наличии хостов. Если система внутри овала содержит только маршрутизаторы, то это подсеть. Если же она содержит как маршрутизаторы, так и хосты, то это глобальная сеть. Реальные различия заключаются в том, кто владеет сетью и пользуется ею.

Часто путают подсети, сети и интерсети. Термин «подсети» обычно употребляется в контексте глобальных сетей, где он означает набор маршрутизаторов и линий связи, принадлежащих одному сетевому оператору. Аналогично этому телефонная система состоит из телефонных станций, соединенных друг с другом высокоскоростными каналами, а с домами и офисами — низкоскоростными каналами. Эти каналы и оборудование принадлежат телефонным компаниям, являющимся аналогами подсетей. Сами телефонные аппараты (аналоги хостов) не являются частью подсетей. Вместе с хостами подсеть образует сеть. В случае локальной сети сеть состоит из кабеля и хостов. Подсетей там нет.

Интерсеть образуется путем объединения нескольких сетей. С нашей точки зрения, объединение локальной и глобальной сетей или объединение двух локальных сетей образует интерсеть, однако в индустрии нет единого мнения по поводу терминологии в данной области. Можно использовать следующее мнемоническое правило: если создание и поддержку сети оплачивают разные организации, то мы имеем дело с интерсетью, а не единой сетью. Также если работа основана на применении нескольких технологий (например, широковещательная в одной ее части и двухузловая — в другой), значит, и сетей несколько.

Сетевое программное обеспечение

Когда собирались первые сети, то основное внимание уделялось аппаратуре, а вопросы программного обеспечения откладывались на будущее. Подобная стратегия больше не работает. Современное сетевое программное обеспечение в высокой степени структурировано. В следующих разделах мы узнаем, как осуществляется эта структуризация. Описанный метод является краеугольным камнем всей книги и будет часто встречаться и далее.

Иерархия протоколов

Для упрощения структуры большинство сетей организуются в наборы **уровней** или **слоев**, каждый последующий из которых возводится над предыдущим. Количество уровней, их названия, содержание и назначение разнятся от сети к сети. Однако во всех сетях целью каждого уровня является предоставление неких сервисов для вышестоящих уровней. При этом от них скрываются детали реализации предоставляемого сервиса.

Такая концепция не нова и используется в computer science уже давно. Ее вариации известны как сокрытие информации, абстрактные типы данных, свойство инкапсуляции и объектно-ориентированное программирование. Фундаментальной идеей является предоставление неким программным или аппаратным уровнем сервисов своим пользователям без раскрытия деталей своего внутреннего состояния и подробностей алгоритмов.

Уровень n одной машины поддерживает связь с уровнем n другой машины. Правила и соглашения, используемые в данном общении, называются **протоко-**

лом уровня *n*. По сути протокол является договоренностью общающихся сторон о том, как должно происходить общение. По аналогии, когда женщину представляют мужчине, она может протянуть ему свою руку. Он, в свою очередь, может принять решение либо пожать, либо поцеловать эту руку в зависимости от того, является ли эта женщина американским адвокатом на деловой встрече или же европейской принцессой на официальном балу. Нарушение протокола создаст затруднения в общении, а может, и вовсе сделает общение невозможным.

На рис. 1.10 показана пятиуровневая сеть. Объекты, включающие в себя соответствующие уровни на разных машинах, называются равноранговыми, или равноправными, узлами, или сущностями, сети. Именно они общаются при помощи протокола.

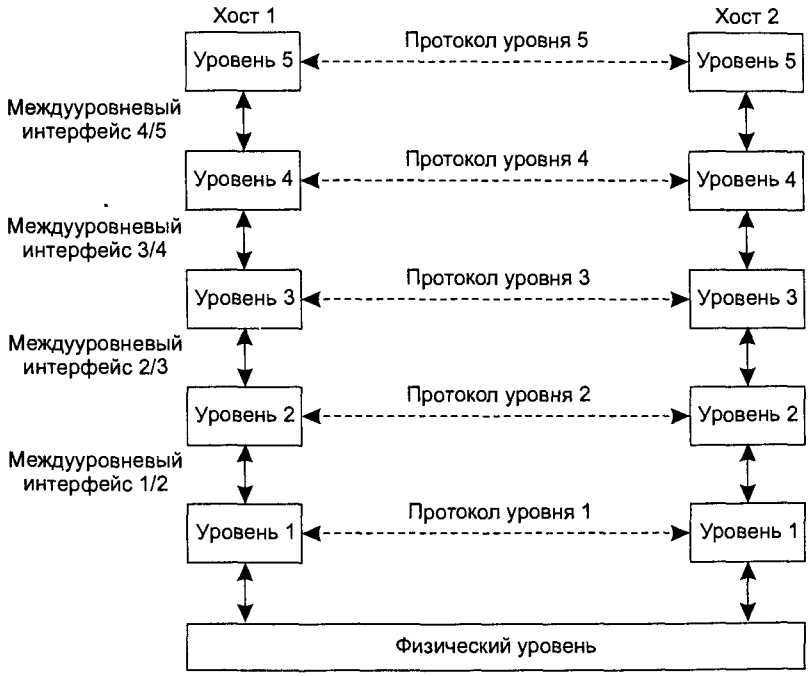


Рис. 1.10. Уровни, протоколы и интерфейсы

В действительности данные не пересылаются с уровня *n* одной машины на уровень *n* другой машины. Вместо этого каждый уровень передает данные и управление уровню, лежащему ниже, пока не достигается самый нижний уровень. Ниже первого уровня располагается **физическая среда**, по которой и производится обмен информацией. На рис. 1.10 виртуальное общение показано пунктиром, тогда как физическое — сплошными линиями.

Между каждой парой смежных уровней находится **интерфейс**, определяющий набор примитивных операций, предоставляемых нижним уровнем верхнему. Когда разработчики сетей решают, сколько уровней включить в сеть и что

на каждом уровне по мере продвижения сообщения. Заголовки нижних уровней более высоким уровням не передаются.

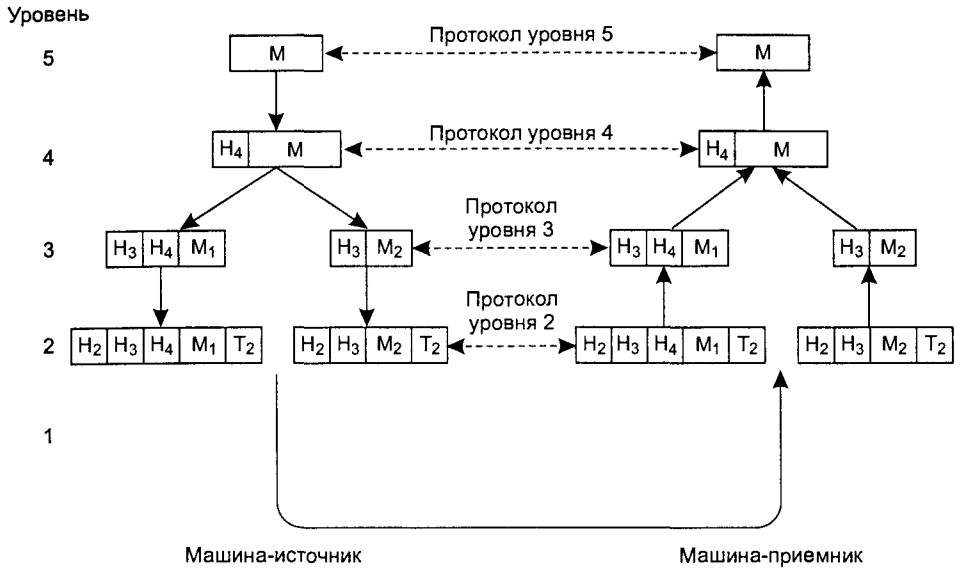


Рис. 1.12. Пример потока информации на уровне 5

Необходимо понять соотношение между виртуальным и реальным общением и разницу между протоколом и интерфейсом. Одноранговые процессы уровня 4, например, считают свое общение горизонтальным, использующим протокол 4-го уровня. У каждого из них имеется процедура с названием вроде *SendToOtherSide* (Отправить другой стороне) и *GetFromOtherSide* (Получить от другой стороны), даже если на самом деле эти процедуры общаются не друг с другом, а с нижними уровнями при помощи интерфейсов 3/4.

Абстракция одноранговых процессов является ключевой для проектирования сетей. С ее помощью невыполнимая задача разработки целой сети может быть разбита на несколько меньших по размеру и вполне разрешимых проблем разработки, а именно разработки индивидуальных уровней.

Хотя этот раздел называется «Сетевое программное обеспечение», следует отметить, что нижние уровни в иерархии протоколов часто реализуются аппаратно или программно-аппаратно. Тем не менее при этом используются сложные алгоритмы протоколов, хотя они и внедряются в аппаратуру частично или целиком.

Разработка уровней

Некоторые из ключевых аспектов разработки, возникающие при создании компьютерных сетей, присутствуют на нескольких уровнях. Далее мы кратко опишем наиболее важные из них.

Каждый уровень нуждается в механизме идентификации отправителей и получателей. В сети обычно работает довольно много компьютеров, на них одновременно могут выполняться сразу несколько процессов, каждому из которых необходимо средство, позволяющее указать, с кем он хочет поговорить. Следовательно, нужна система **адресации**.

Также необходимо выработать правила для переноса данных. В некоторых системах данные могут перемещаться только в одном направлении, в других — в любом направлении. Протокол также должен определять количество логических каналов, относящихся к соединению, и их приоритеты. Многие сети обеспечивают минимум по два логических канала на соединение: один для обычных данных и еще один — для срочных.

Важным аспектом является контроль ошибок, поскольку физические каналы связи несовершенны. Известно множество кодов, опознающих и исправляющих ошибки, однако обе стороны соединения должны договориться между собой о том, какой именно код будет выбран. Кроме того, получатель должен иметь возможность сообщить отправителю, какие из сообщений были получены правильно, а какие — нет.

Не все каналы связи сохраняют последовательность посылаемых по ним сообщений. Чтобы исправить возможную потерю порядка сообщений, протокол должен явно снабжать получателя номерами пакетов, чтобы получаемые фрагменты сообщений могли быть собраны в правильном порядке. Очевидным решением проблемы является нумерация пакетов, однако остается открытым вопрос: что делать с пакетами, приходящими в неверном порядке?

Кроме того, на каждом уровне возникает вопрос: как организовать пересылку данных так, чтобы быстрая передающая сторона не завалила пакетами медленную принимающую сторону? Для разрешения данной проблемы существуют различные решения, которые будут обсуждаться далее. Некоторые из них предполагают прямые или косвенные ответы получателя посылающей стороне, информирующие ее о текущем состоянии получателя. Другим решением может быть ограничение скорости передачи до некоторого договорного уровня. В целом это называется **управлением потоком**.

Еще одна проблема, которую необходимо разрешать на различных уровнях, — это неспособность всех процессов принимать сколь угодно длинные сообщения. С этим может быть связан вопрос: что делать, если процесс настаивает на передаче данных столь малыми порциями, что передача становится неэффективной? Для решения подобной проблемы можно объединять посылаемые сообщения в один большой пакет и снова разбивать его после пересылки на отдельные сообщения.

Когда неудобно или неэффективно устанавливать отдельное соединение для каждой пары общающихся процессов, располагающийся ниже уровень может принять решение использовать одно и то же соединение для различных не связанных друг с другом разговоров. Пока это **уплотнение каналов**, или **мультиплексирование**, происходит прозрачно, оно может использоваться любым уровнем. Мультиплексирование, например, необходимо на физическом уровне, где вся связь должна осуществляться по ограниченному числу контуров.

Когда между отправителем и получателем существует несколько возможных путей следования сообщения, возникает задача выбора пути. Иногда она может быть разделена между несколькими уровнями. Например, при посылке сообщения из Лондона в Рим верхний уровень может выбрать путь через Францию или Германию, основывая свой выбор на знании законов, касающихся тайны переписки в данных странах, тогда как выбор нижнего уровня может основываться на текущей загруженности линий связи. Эта задача называется **маршрутизацией**.

Службы на основе соединений и службы без установления соединений

Уровни могут предлагать вышестоящим уровням услуги двух типов: с наличием или отсутствием установления соединения. В этом разделе мы рассмотрим, что означает каждый из этих типов и в чем состоит разница между ними.

Типичным примером сервиса с установлением соединения является телефонная связь. Чтобы поговорить с кем-нибудь, необходимо поднять трубку, набрать номер, а после окончания разговора положить трубку. Нечто подобное происходит и в компьютерных сетях: при использовании сервиса с установлением соединения абонент сперва устанавливает соединение, а после окончания сеанса разрывает его. Это напоминает трубу: биты сообщения влетают в один ее конец, а вылетают с другого. В большинстве случаев не возникает путаницы с последовательностью передачи этих битов.

В некоторых случаях перед началом передачи отправляющая и получающая машины обмениваются приветствиями, отсылая друг другу приемлемые параметры соединения: максимальный размер сообщения, необходимое качество сервиса и др. В большинстве случаев одна из сторон посылает запрос, а другая его принимает, отвергает или же выставляет встречные условия.

Что касается сервисов без установления соединеня, то типичный пример такой технологии — почтовые системы. Каждое письмо содержит полный адрес назначения и проходит по некоему маршруту, который совершенно не зависит от других писем. Обычно то письмо, которое отправлено раньше, в место назначения приходит раньше.

Каждая служба характеризуется **качеством обслуживания**. Некоторые службы являются надежными в том смысле, что они никогда не теряют данных. Обычно надежная служба реализуется при помощи подтверждений, посылаемых получателем в ответ на каждое принятое сообщение, так что отправитель знает, дошло очередное сообщение или нет. Процесс пересылки подтверждений требует некоторых накладных расходов и снижает пропускную способность канала. Впрочем, подобные затраты обычно не очень велики и окупаются, хотя иногда могут быть нежелательными.

Типичным примером необходимости надежной службы на основе соединений является пересылка файлов. Владелец файла хочет быть уверенным, что все биты файла прибыли без искажений и в том же порядке, в котором были отправле-

ны. Вряд ли кто-нибудь отдаст предпочтение службе, которая случайным образом искажает информацию, даже если передача происходит значительно быстрее.

Надежные службы на основе соединений бывают двух типов: последовательности сообщений и байтовые потоки. В первом варианте сохраняются границы между сообщениями. Когда посылаются два сообщения размером по 1 Кбайт, то они прибывают в виде двух сообщений размером по 1 Кбайт и никогда — как одно двухкилобайтное сообщение. При втором варианте связь представляет собой просто поток байтов, без разделения на отдельные сообщения. Когда 2048 байт прибывают к получателю, то нет никакой возможности определить, было это одно сообщение длиной 2 Кбайт, два сообщения длиной 1 Кбайт или же 2048 однобайтных сообщений. Если страницы книги посылаются по сети фотонаборной машине в виде отдельных сообщений, то, возможно, необходимо сохранить границы между сообщениями. С другой стороны, при регистрации с удаленного терминала в системе разделения времени вполне достаточно потока байтов с терминального компьютера.

Как уже упоминалось ранее, существуют системы, для которых задержки, связанные с пересылкой подтверждений, неприемлемы. В качестве примера такой системы можно назвать цифровую голосовую связь. В данном случае предпочтительнее допустить шумы на линии или искаженные слова, нежели большие паузы, вызванные отсылкой подтверждений и повторной передачей блоков данных. Аналогично, при проведении видеоконференции отдельные неправильные пикселы окажутся меньшей проблемой, нежели дергающиеся и останавливающиеся кадры.

Не все приложения требуют установки соединения. Например, при рассылке рекламы по электронной почте установка связи для пересылки каждого отдельного сообщения нежелательна. Также не требуется в этом случае и 100-процентная надежность, особенно, если это существенно увеличит стоимость. Все, что нужно, — это способ переслать сообщение с высокой вероятностью его получения, но без гарантии. Ненадежная (то есть без подтверждений) служба без установления соединения часто называется **службой дейтаграмм**, или дейтаграммной службой — по аналогии с телеграфной службой, также не предоставляющей подтверждений отправителю.

В других ситуациях бывает желательно не устанавливать соединение для пересылки коротких сообщений, но надежность, тем не менее, существенна. Такая служба называется **службой дейтаграмм с подтверждениями**. Она подобна отправке заказного письма с подтверждением получения. Получив подтверждение, отправитель уверен, что письмо доставлено адресату, а не потеряно по дороге.

Кроме того, существует **служба запросов и ответов**, в которой отправитель посылает дейтаграммы, содержащие запросы, и получает ответы от получателя. Например, к данной категории можно отнести вопрос к библиотеке о том, где говорят по-уйгурски. Обычно модель запросов и ответов применяется для реализации общения в модели «клиент-сервер»: клиент посылает запрос, а сервер отвечает на него. Обсуждавшиеся ранее типы служб сведены в таблицу на рис. 1.13.

	Служба	Пример
Ориентированная на соединение	Надежный поток сообщений	Последовательность страниц
	Надежный поток байт	Удаленная регистрация
	Ненадежное соединение	Цифровая голосовая связь
Без установления соединения	Ненадежная дейтаграмма	Рассылка рекламы электронной почтой
	Дейтаграмма с подтверждениями	Заказные письма
	Запрос — ответ	Запрос к базе данных

Рис. 1.13. Шесть типов служб

Концепция использования ненадежной связи поначалу может показаться несколько странной. В самом деле, почему это может возникать такая ситуация, когда выгоднее предпочесть ненадежную связь надежной? Во-первых, надежное соединение (в том смысле, который был оговорен ранее, то есть с подтверждением) не всегда можно установить. Скажем, Ethernet не является «надежным» средством коммуникации. Пакеты при передаче могут искажаться, но решать эту проблему должны протоколы более высоких уровней. Во-вторых, задержки, связанные с отсылкой подтверждения, в некоторых случаях неприемлемы, особенно при передаче мультимедиа в реальном времени. Именно благодаря этим факторам продолжают сосуществовать надежные и ненадежные соединения.

Примитивы служб

Служба (сервис) формально описывается набором **примитивов** или операций, доступных пользователю или другой сущности для получения сервиса. Эти примитивы заставляют службу выполнять некоторые действия или служат ответами на действия сущности того же уровня. Если набор протоколов входит в состав операционной системы (как часто и бывает), то примитивы являются системными вызовами. Они приводят к возникновению системных прерываний в привилегированном режиме, в результате чего управление машиной передается операционной системе, которая и отсылает нужные пакеты.

Набор доступных примитивов зависит от природы сервиса. Скажем, примитивы сервисов с установлением соединения и без него различаются. В табл. 1.3 приведен минимальный набор примитивов, обеспечивающий надежную передачу битового потока в среде типа «клиент-сервер».

Эти примитивы могут использоваться следующим образом. Вначале сервер исполняет LISTEN, показывая тем самым, что он готов устанавливать входящие соединения. Этот примитив обычно реализуется в виде блокирующего системного вызова. После его исполнения процесс сервера приостанавливается до тех пор, пока не будет установлено соединение.

Таблица 1.3. Пять сервисных примитивов, обеспечивающих простую передачу с установлением соединения

Примитив	Значение
LISTEN (ожидание)	Блок ожидает входящего соединения
CONNECT (соединение)	Установка соединения с ожидающей сущностью того же ранга
RECEIVE (прием)	Блок ожидает входящего сообщения
SEND (отправка)	Отправка сообщения ожидающей сущности того же ранга
DISCONNECT (разрыв)	Разрыв соединения

Затем процесс клиента выполняет примитив CONNECT, устанавливая соединение с сервером. В системном вызове должно быть указано, с кем именно необходимо установить связь. Для этого может вводиться специальный параметр, содержащий адрес сервера. Далее операционная система клиента посылает равноранговой сущности пакет с запросом на соединение, как показано на рис. 1.14 стрелочкой с пометкой (1). Процесс клиента приостанавливается в ожидании ответа. Пакет, пришедший на сервер, обрабатывается его операционной системой. Если в пакете обнаруживается запрос на соединение, начинается поиск того клиента, который отправил этот запрос. При его обнаружении производятся два действия: клиент разблокируется и ему отсылается подтверждение (2). Реальное разблокирование происходит по прибытии подтверждения на клиентскую машину. Начиная с этого момента считается, что сервер и клиент установили соединение. Важно отметить здесь то, что подтверждение (2) генерируется самим кодом протокола и не является ответом на примитив пользователя, содержащий запрос. Может возникнуть ситуация, когда запрос на соединение есть, а клиента нет. В этом случае результат будет неопределенным. В некоторых системах пакет может быть отложен на короткое время, в течение которого ожидается LISTEN.

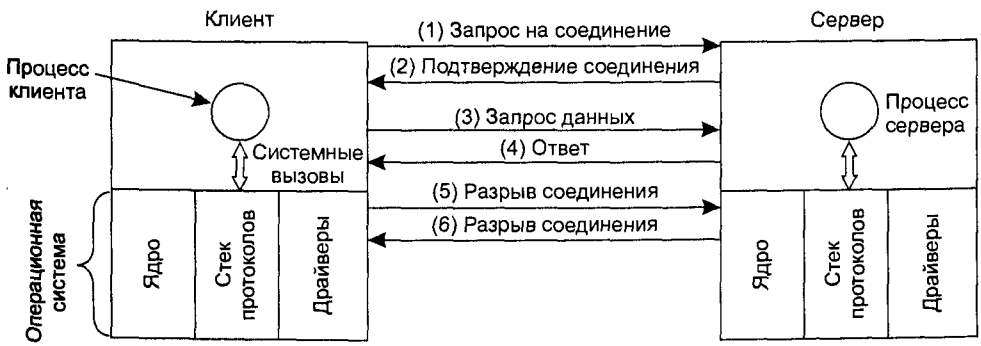


Рис. 1.14. Простейшее взаимодействие клиента и сервера при передаче пакетов по сети с установлением соединения

Самым очевидным жизненным примером такого взаимодействия может служить звонок покупателя (клиента) в сервисный центр компании. Менеджер сер-

висного центра должен находиться у телефона, чтобы иметь возможность ответить в том случае, если он зазвонит. Клиент совершает звонок. Когда менеджер поднимает трубку, считается, что соединение установлено.

Следующим шагом будет выполнение сервером примитива RECEIVE, подготавливающего систему к принятию первого запроса. В нормальной ситуации это происходит сразу же после прекращения ожидания (LISTEN), даже до того, как клиент получает подтверждение соединения. Системный вызов RECEIVE вновь блокирует сервер.

Клиент выполняет SEND, передает запрос (3) и сразу же выполняет RECEIVE, ожидая ответ.

Прием пакета с запросом разблокирует процесс сервера, благодаря чему он может обработать запрос. По окончании обработки выполняется примитив SEND, и ответ отсылается клиенту (4). Прием пакета разблокирует клиента, теперь наступает его очередь обрабатывать пакет. Если у клиента есть еще запросы к серверу, он может отослать их. В противном случае соединение разрывается с помощью DISCONNECT. Обычно первый примитив DISCONNECT отсылает пакет, уведомляющий сервер об окончании сеанса, и блокирует клиента (5). В ответ сервер генерирует свой примитив DISCONNECT, являющийся подтверждением для клиента и командой, разрывающей связь. Клиент, получив его, разблокируется, и соединение считается окончательно разорванным. Именно так в двух словах можно описать схему коммуникации с установлением соединения.

Конечно, жизнь не настолько проста. Описанный алгоритм работы весьма схематичен, а кое-что просто неправильно (например, CONNECT на самом деле выполняется до LISTEN). При этом пакеты, бывает, теряются, возникают и другие проблемы. Позднее мы рассмотрим все это гораздо более подробно, но на данный момент можно получить лишь общее представление о работе клиент-серверной системы с установлением соединения. Для этого полезно внимательно изучить рис. 1.14.

Увидев эти шесть пакетов, необходимых для работы протокола, можно удивиться, почему же не используется протокол без установления соединения? Ответ таков: в идеальном мире, где нужны всего два пакета — один для запроса и один для ответа, — это, возможно, имело бы смысл. Но стоит представить себе передачу большого сообщения (скажем, мегабайтного файла), причем в обе стороны, причем с ошибками при передаче, потерянными пакетами и т. д., как ситуация меняется. Если ответ сервера состоит из нескольких сотен пакетов, парочка из которых затерялась по пути, то как клиент узнает, что он получил сообщение не в полном объеме? Как он узнает о том, что последний принятый пакет является действительно последним? Допустим, клиент запросил второй файл. Как он отличит пакет 1 из второго файла от потерянного пакета 1 из первого файла, который вдруг нашелся? Короче говоря, в реальном мире простой протокол запросов-ответов без подтверждений часто не подходит. В главе 3 мы обсудим протоколы, позволяющие решать самые разные проблемы, возникающие при передаче данных. А сейчас поверьте на слово: наличие надежной связи с упорядоченным байтовым потоком между процессами — это удобно.

Службы и протоколы

Службы и протоколы являются различными понятиями, хотя часто эти понятия смешиваются. Различие между ними, однако, столь важно, что мы хотели бы еще раз обратить на него ваше внимание. *Служба* (или *сервис*) — это набор примитивов (операций), которые более низкий уровень предоставляет более высокому. Служба определяет, какие именно операции уровень будет выполнять от лица своих пользователей, но никак не оговаривает, как должны реализовываться эти операции. Служба описывает интерфейс между двумя уровнями, в котором нижний уровень является поставщиком сервиса, а верхний — его потребителем.

Напротив, *протокол* — это набор правил, описывающих формат и назначение кадров, пакетов или сообщений, которыми обмениваются одноранговые сущности внутри уровня. Сущности используют протокол для реализации определений их служб. Они могут менять протокол по желанию, при условии что при этом остаются неизменными службы, предоставляемые ими своим пользователям. Таким образом, служба и протокол оказываются практически независимыми.

Другими словами, службы — это нечто связанное с межуровневыми интерфейсами, тогда как протоколы связаны с пакетами, передающимися сущностями одного уровня, расположенными на разных машинах. Это показано на рис. 1.15. Важно не путать эти два понятия.

Стоит провести аналогию с языками программирования. Службу можно уподобить абстрактному типу данных или объекту в объектно-ориентированных языках программирования. Он определяет операции, которые могут выполняться с объектом, но не описывает, как реализованы эти операции. В этом случае протокол относится к *реализации* службы и, таким образом, невидим для пользователей службы.

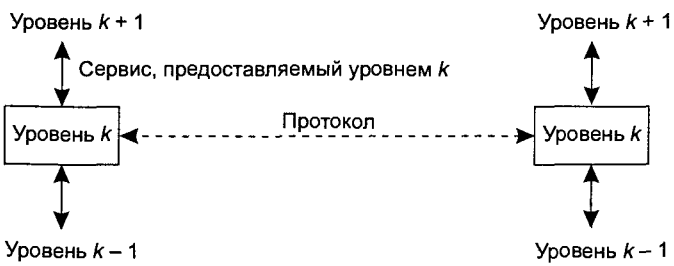


Рис. 1.15. Связь между службой и протоколом

Во многих старых системах служба не отделялась от протокола. В результате типичный уровень мог содержать примитив службы SEND PACKET, в котором пользователь должен был указать ссылку на полностью собранный пакет. Это означало, что любые изменения протокола тут же становились видимыми для пользователей. Большинство разработчиков сетей сегодня считают подобный подход серьезнейшей ошибкой.

Эталонные модели

Обсудив многоуровневые сети в общих чертах, пора рассмотреть несколько примеров. В следующих двух разделах описываются два важных архитектурных типа — эталонные модели OSI и TCP/IP. Несмотря на то что *протоколы*, связанные с эталонной моделью OSI, используются сейчас очень редко, сама *модель* до сих пор весьма актуальна, а свойства ее уровней, которые будут обсуждаться в этом разделе, очень важны. В эталонной модели TCP/IP все наоборот — сама модель сейчас почти не используется, а ее протоколы являются едва ли не самыми распространенными. Исходя из этого, мы обсудим подробности, касающиеся обеих моделей. К тому же иногда приходится больше узнавать из поражений, чем из побед.

Эталонная модель OSI

Эталонная модель OSI (за исключением физической среды) показана на рис. 1.16. Эта модель основана на разработке Международной организации по стандартизации (International Organization for Standardization, ISO) и является первым шагом к международной стандартизации протоколов, используемых на различных уровнях (Day и Zimmerman, 1983). Затем она была пересмотрена в 1995 году (Day, 1995). Называется эта структура эталонной моделью взаимодействия открытых систем ISO (**ISO OSI (Open System Interconnection) Reference Model**), поскольку она связывает открытые системы, то есть системы, открытые для связи с другими системами. Для краткости мы будем называть эту модель просто «модель OSI».

Модель OSI имеет семь уровней. Появление именно такой структуры было обусловлено следующими соображениями.

1. Уровень должен создаваться по мере необходимости отдельного уровня абстракции.
2. Каждый уровень должен выполнять строго определенную функцию.
3. Выбор функций для каждого уровня должен осуществляться с учетом создания стандартизированных международных протоколов.
4. Границы между уровнями должны выбираться так, чтобы поток данных между интерфейсами был минимальным.
5. Количество уровней должно быть достаточно большим, чтобы различные функции не объединялись в одном уровне без необходимости, но не слишком высоким, чтобы архитектура не становилась громоздкой.

Далее мы обсудим каждый уровень модели, начиная с самого нижнего. Обратите внимание: модель OSI не является сетевой архитектурой, поскольку она не описывает службы и протоколы, используемые на каждом уровне. Она просто определяет, что должен делать каждый уровень. Тем не менее ISO также разработала стандарты для каждого уровня, хотя эти стандарты не входят в саму эталонную модель. Каждый из них был опубликован как отдельный международный стандарт.

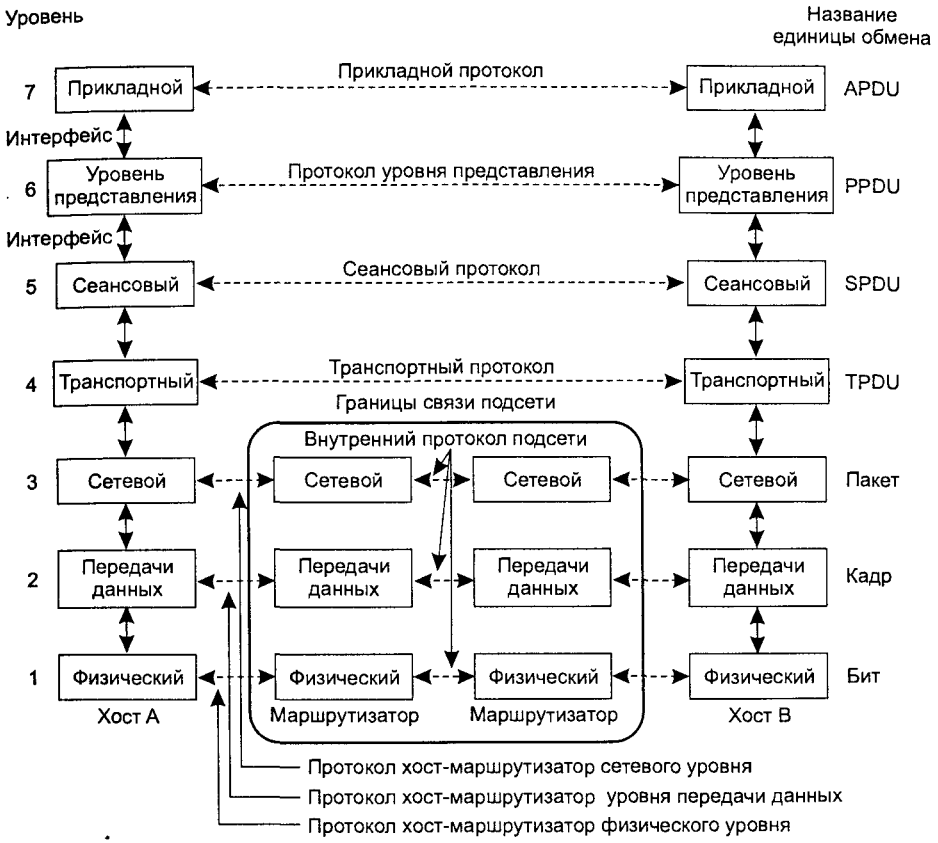


Рис. 1.16. Эталонная модель OSI

Физический уровень

Физический уровень занимается реальной передачей необработанных битов по каналу связи. При разработке сети необходимо убедиться, что когда одна сторона передает единицу, то принимающая сторона получает также единицу, а не ноль. Принципиальными вопросами здесь являются следующие: какое напряжение должно использоваться для отображения единицы, а какое — для нуля; сколько микросекунд длится бит; может ли передача производиться одновременно в двух направлениях; как устанавливается начальная связь и как она прекращается, когда обе стороны закончили свои задачи; из какого количества проводов должен состоять кабель и какова функция каждого провода. Вопросы разработки в основном связаны с механическими, электрическими и процедурными интерфейсами, а также с физическим носителем, лежащим ниже физического уровня.

Уровень передачи данных

Основная задача **уровня передачи данных** — быть способным передавать «сырые» данные физического уровня по надежной линии связи, свободной от необ-

наруженных ошибок с точки зрения вышестоящего сетевого уровня. Уровень выполняет эту задачу при помощи разбиения входных данных на **кадры**, обычный размер которых колеблется от нескольких сотен до нескольких тысяч байт. Кадры данных передаются последовательно с обработкой **кадров подтверждения**, отсылаемых обратно получателем.

Еще одна проблема, возникающая на уровне передачи данных (а также и на большей части более высоких уровней), — как не допустить ситуации, когда быстрый передатчик заваливает приемник данными. Должен быть предусмотрен некий механизм регуляции, который информировал бы передатчик о наличии свободного места в буфере приемника на текущий момент. Часто подобное управление объединяется с механизмом обработки ошибок.

В широковещательных сетях существует еще одна проблема уровня передачи данных: как управлять доступом к совместно используемому каналу. Эта проблема разрешается введением специального дополнительного подуровня уровня передачи данных — подуровня доступа к носителю.

Сетевой уровень

Сетевой уровень занимается управлением операциями подсети. Важнейшим моментом здесь является определение маршрутов пересылки пакетов от источника к пункту назначения. Маршруты могут быть жестко заданы в виде таблиц и редко меняться. Кроме того, они могут задаваться в начале каждого соединения, например терминальной сессии. Наконец, они могут быть в высокой степени динамическими, то есть вычисляемыми заново для каждого пакета с учетом текущей загруженности сети.

Если в подсети одновременно присутствует слишком большое количество пакетов, то они могут закрыть дорогу друг другу, образуя заторы в узких местах. Недопущение подобной закупорки также является задачей сетевого уровня. В более общем смысле сетевой уровень занимается предоставлением определенного уровня сервиса (это касается задержек, времени передачи, вопросов синхронизации).

При путешествии пакета из одной сети в другую также может возникнуть ряд проблем. Так, способ адресации, применяемый в одной сети, может отличаться от принятого в другой. Сеть может вообще отказаться принимать пакеты из-за того, что они слишком большого размера. Также могут различаться протоколы, и т. д. Именно сетевой уровень должен разрешать все эти проблемы, позволяя объединять разнородные сети.

В широковещательных сетях проблема маршрутизации очень проста, поэтому в них сетевой уровень очень примитивный или вообще отсутствует.

Транспортный уровень

Основная функция **транспортного уровня** — принять данные от сеансового уровня, разбить их при необходимости на небольшие части, передать их сетевому уровню и гарантировать, что эти части в правильном виде прибудут по назначению. Кроме того, все это должно быть сделано эффективно и таким образом, чтобы

изолировать более высокие уровни от каких-либо изменений в аппаратной технологии.

Транспортный уровень также определяет тип сервиса, предоставляемого сеансовому уровню и, в конечном счете, пользователям сети. Наиболее популярной разновидностью транспортного соединения является защищенный от ошибок канал между двумя узлами, поставляющий сообщения или байты в том порядке, в каком они были отправлены. Однако транспортный уровень может предоставлять и другие типы сервисов, например пересылку отдельных сообщений без гарантии соблюдения порядка их доставки или одновременную отправку сообщения различным адресатам по принципу широковещания. Тип сервиса определяется при установке соединения. (Строго говоря, полностью защищенный от ошибок канал создать невозможно. Говорят лишь о таком канале, уровень ошибок в котором достаточно мал, чтобы ими можно было пренебречь на практике.)

Транспортный уровень является настоящим сквозным уровнем, то есть доставляющим сообщения от источника адресату. Другими словами, программа на машине-источнике поддерживает связь с подобной программой на другой машине при помощи заголовков сообщений и управляющих сообщений. На более низких уровнях для поддержки этого соединения устанавливаются соединения между всеми соседними машинами, через которые проходит маршрут сообщений. Различие между уровнями с 1-го по 3-й, действующими по принципу звеньев цепи, и уровнями с 4-го по 7-й, являющимися сквозными, проиллюстрировано на рис. 1.16.

Сеансовый уровень

Сеансовый уровень позволяет пользователям различных компьютеров устанавливать **сеансы связи** друг с другом. При этом предоставляются различные типы сервисов, среди которых **управление диалогом** (отслеживание очередности передачи данных), **управление маркерами** (предотвращение одновременного выполнения критичной операции несколькими системами) и **синхронизация** (установка служебных меток внутри длинных сообщений, позволяющих после устранения ошибки продолжить передачу с того места, на котором она оборвалась).

Уровень представления

В отличие от более низких уровней, задача которых — достоверная передача битов и байтов, **уровень представления** занимается по большей части синтаксисом и семантикой передаваемой информации. Чтобы было возможно общение компьютеров с различными представлениями данных, необходимо преобразовывать форматы данных друг в друга, передавая их по сети в неком стандартизированном виде. Уровень представления занимается этими преобразованиями, предоставляя возможность определения и изменения структур данных более высокого уровня (например, записей баз данных).

Прикладной уровень

Прикладной уровень содержит набор популярных протоколов, необходимых пользователям. Одним из наиболее распространенных является протокол пере-

дачи гипертекста **HTTP** (HyperText Transfer Protocol), который составляет основу технологии Всемирной Паутины. Когда браузер запрашивает веб-страницу, он передает ее имя (адрес) и рассчитывает на то, что сервер будет использовать HTTP. Сервер в ответ отсылает страницу. Другие прикладные протоколы используются для передачи файлов, электронной почты, сетевых рассылок.

Эталонная модель TCP/IP

Рассмотрим теперь эталонную модель, использовавшуюся в компьютерной сети ARPANET, которая является бабушкой нынешних сетей, а также в ее наследнице, всемирной сети Интернет. Хотя краткую историю сети ARPANET мы рассмотрим чуть позднее, некоторые ключевые моменты ее следует отметить прямо сейчас. ARPANET была исследовательской сетью, финансируемой Министерством обороны США. В конце концов она объединила сотни университетов и правительственных зданий при помощи выделенных телефонных линий. Когда впоследствии появились спутниковые сети и радиосети, возникли большие проблемы при объединении с ними других сетей с помощью имеющихся протоколов. Понадобилась новая эталонная архитектура. Таким образом, возможность объединять различные сети в единое целое являлась одной из главных целей с самого начала. Позднее эта архитектура получила название **эталонной модели TCP/IP** в соответствии со своими двумя основными протоколами. Первое ее описание встречается в книге Cerf и Kahn (1974). Из более поздних описаний можно выделить книгу, написанную Leiner и др. в 1985 году. Конструктивные особенности модели обсуждаются в издании Clark, 1988.

Поскольку Министерство обороны беспокоилось, что ценные хосты, маршрутизаторы и межсетевые шлюзы могут быть мгновенно уничтожены, другая важная задача состояла в том, чтобы добиться способности сети сохранять работоспособность при возможных потерях подсетевого оборудования, так, чтобы при этом связь не прерывалась. Другими словами, Министерство обороны требовало, чтобы соединение не прерывалось, пока функционируют приемная и передающая машины, даже если некоторые промежуточные машины или линии связи внезапно вышли из строя. Кроме того, от архитектуры нужна была определенная гибкость, поскольку предполагалось использовать приложения с различными требованиями, от переноса файлов до передачи речи в реальном времени.

Интернет-уровень

Все эти требования обусловили выбор модели сети с коммутацией пакетов, в основе которой лежал не имеющий соединений межсетевой уровень. Этот уровень, называемый **интернет-уровнем** или **межсетевым уровнем**, является основой всей архитектуры. Его задача заключается в обеспечении возможности для каждого хоста посылать в любую сеть пакеты, которые будут независимо двигаться к пункту назначения (например, в другой сети). Они могут прибывать не в том порядке, в котором были отправлены. Если требуется соблюдение порядка отправления, эту задачу выполняют более верхние уровни. Обратите внимание, что сло-

во «интернет» здесь используется в своем первоначальном смысле несмотря на то, что этот уровень присутствует в сети Интернет.

Здесь можно увидеть аналогию с почтовой системой. Человек может бросить несколько международных писем в почтовый ящик в одной стране, и если повезет, большая часть из них будет доставлена по правильным адресам в других странах. Вероятно, письма по дороге пройдут через несколько международных почтовых шлюзов, однако это останется тайной для корреспондентов. В каждой стране (то есть в каждой сети) могут быть свои марки, свои предпочитаемые размеры конвертов и правила доставки, незаметные для пользователей почтовой службы.

Межсетевой уровень определяет официальный формат пакета и протокол, называемый **IP** (Internet Protocol). Задачей межсетевого протокола является доставка IP-пакетов к пунктам назначения. Основными аспектами здесь являются выбор маршрута пакета и недопущение закупорки транспортных артерий. Поэтому можно утверждать, что межсетевой уровень модели TCP/IP функционально близок сетевому уровню модели OSI. Это соответствие показано на рис. 1.17.

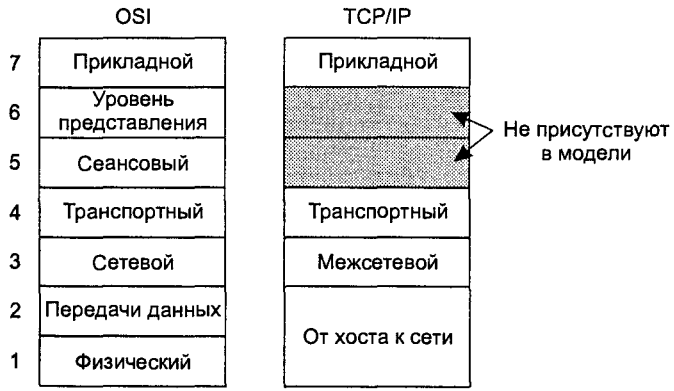


Рис. 1.17. Эталонная модель TCP/IP

Транспортный уровень

Уровень, расположенный над межсетевым уровнем модели TCP/IP, как правило, называют **транспортным**. Он создан для того, чтобы одноранговые сущности на приемных и передающих хостах могли поддерживать связь, подобно транспортному уровню модели OSI. На этом уровне должны быть описаны два сквозных протокола. Первый, **TCP** (Transmission Control Protocol — протокол управления передачей), является надежным протоколом с установлением соединений, позволяющим без ошибок доставлять байтовый поток с одной машины на любую другую машину объединенной сети. Он разбивает входной поток байтов на отдельные сообщения и передает их межсетевому уровню. В пункте назначения получающий TCP-процесс собирает из полученных сообщений выходной поток. Кроме того, TCP осуществляет управление потоком, чтобы быстрый отправитель не завалил информацией медленного получателя.

Второй протокол этого уровня, **UDP** (User Data Protocol — пользовательский протокол данных), является ненадежным протоколом без установления соединения, не использующим последовательное управление потоком протокола TCP, а предоставляющим свое собственное. Он также широко используется в одноразовых клиент-серверных запросах и приложениях, в которых оперативность важнее аккуратности, например, при передаче речи и видео. Взаимоотношения протоколов IP, TCP и UDP показаны на рис. 1.18. Со времени создания протокола IP этот протокол был реализован во многих других сетях.

Прикладной уровень

В модели TCP/IP нет сеансового уровня и уровня представления. В этих уровнях просто не было необходимости, поэтому они не были включены в модель. Опыт работы с моделью OSI доказал правоту этой точки зрения: большинство приложений в них мало нуждаются.

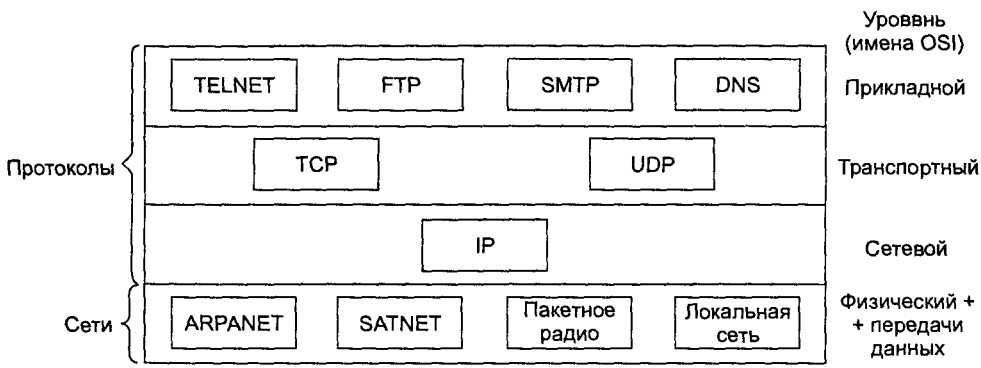


Рис. 1.18. Протоколы и сети в модели TCP/IP

Над транспортным уровнем располагается прикладной уровень. Он содержит все протоколы высокого уровня. К старым протоколам относятся протокол виртуального терминала (TELNET), протокол переноса файлов (FTP) и протокол электронной почты (SMTP), как показано на рис. 1.18. Протокол виртуального терминала позволяет пользователю регистрироваться на удаленном сервере и работать на нем. Протокол переноса файлов предоставляет эффективный способ перемещения информации с машины на машину. Электронная почта изначально представляла собой разновидность переноса файлов, однако позднее для нее был разработан специальный протокол. С годами было добавлено много других протоколов, таких как DNS (Domain Name Service — служба имен доменов), позволяющая преобразовывать имена хостов в сетевые адреса, NNTP (Network News Transfer Protocol — сетевой протокол передачи новостей), HTTP, протокол, используемый для создания страниц на World Wide Web, и многие другие.

Хост-сетевой уровень

В эталонной модели TCP/IP не описывается подробно, что располагается ниже межсетевого уровня. Сообщается только, что хост соединяется с сетью при помощи

какого-нибудь протокола, позволяющего ему посылать по сети IP-пакеты. Этот протокол никак не определяется и может меняться от хоста к хосту и от сети к сети. В книгах и статьях, посвященных модели TCP/IP, этот вопрос обсуждается редко.

Сравнение эталонных моделей OSI и TCP

У моделей OSI и TCP имеется много общих черт. Обе модели основаны на концепции стека независимых протоколов. Функциональность уровней также во многом схожа. Например, в обеих моделях уровни, начиная с транспортного и выше, предоставляют сквозную, не зависящую от сети транспортную службу для процессов, желающих обмениваться информацией. Эти уровни образуют поставщика транспорта. Также в каждой модели уровни выше транспортного являются прикладными потребителями транспортных сервисов.

Несмотря на это фундаментальное сходство, у этих моделей имеется и ряд отличий. В данном разделе мы обратим внимание на ключевые различия. Обратите внимание на то, что мы сравниваем именно *эталонные модели*, а не соответствующие им *стеки протоколов*. Сами протоколы будут обсуждаться несколько позднее. Существует книга (Piscitello и Chapin, 1993), которая целиком посвящена сравнению моделей TCP/IP и OSI.

Для модели OSI центральными являются три концепции:

1. Службы.
2. Интерфейсы.
3. Протоколы.

Вероятно, наибольшим вкладом модели OSI стало явное разделение этих трех концепций. Каждый уровень предоставляет некоторые сервисы для расположенного выше уровня. *Сервис* определяет, что именно делает уровень, но не то, как он это делает и каким образом сущности, расположенные выше, получают доступ к данному уровню.

Интерфейс уровня определяет способ доступа к уровню для расположенных выше процессов. Он описывает параметры и ожидаемый результат. Он также ничего не сообщает о внутреннем устройстве уровня.

Наконец, равноранговые *протоколы*, применяемые в уровне, являются внутренним делом самого уровня. Для выполнения поставленной ему задачи (то есть предоставления сервиса) он может использовать любые протоколы. Кроме того, уровень может менять протоколы, не затрагивая работу приложений более высоких уровней.

Эти идеи очень хорошо соответствуют современным идеям объектно-ориентированного программирования. Уровень может быть представлен в виде объекта, обладающего набором методов (операций), к которым может обращаться внешний процесс. Семантика этих методов определяет набор служб, предоставляемых объектом. Параметры и результаты методов образуют интерфейс объекта. Внутреннее устройство объекта можно сравнить с протоколом уровня. За пределами объекта оно никого не интересует и никому не видно.

Изначально в модели TCP/IP не было четкого разделения между службами, интерфейсом и протоколом, хотя и производились попытки изменить это, чтобы

сделать ее более похожей на модель OSI. Так, например, единственными настоящими сервисами, предоставляемыми межсетевым уровнем, являются SEND IP PACKET (послать IP-пакет) и RECEIVE IP PACKET (получить IP-пакет).

В результате в модели OSI протоколы скрыты лучше, чем в модели TCP/IP, и при изменении технологии они могут быть относительно легко заменены. Возможность проводить подобные изменения — одна из главных целей многоуровневых протоколов.

Эталонная модель OSI была разработана *прежде*, чем были изобретены протоколы для нее. Такая последовательность событий означает, что эта модель не была настроена на какой-то конкретный набор протоколов, что сделало ее универсальной. Обратной стороной такого порядка действий было то, что у разработчиков было мало опыта в данной области и не было четкого представления о том, какие функции должен выполнять каждый уровень.

Например, уровень передачи данных изначально работал только в сетях с передачей от узла к узлу. С появлением широковещательных сетей в модель потребовалось ввести новый подуровень. Когда же на базе модели OSI начали строить реальные сети с использованием существующих протоколов, обнаружилось, что они не соответствуют требуемым спецификациям служб. Поэтому в модель пришлось добавить подуровни для устранения несоответствия. Наконец, изначально ожидалось, что в каждой стране будет одна сеть, управляемая правительством и использующая протоколы OSI, поэтому никто и не думал об объединении различных сетей. В действительности все оказалось не так.

С моделью TCP/IP было все наоборот: сначала появились протоколы, а уже затем была создана модель, описывающая существующие протоколы. Таким образом, не было проблемы с соответствием протоколов модели. Они ей соответствовали прекрасно. Единственной проблемой было то, что *модель* не соответствовала никаким другим стекам протоколов. В результате она не использовалась для описания каких-нибудь других сетей, отличных от TCP/IP.

Если взглянуть на эти две модели поближе, то прежде всего обратит на себя внимание различие в количестве уровней: в модели OSI семь уровней, в модели TCP/IP — четыре. В обеих моделях имеются межсетевой, транспортный и прикладной уровни, а остальные уровни различные.

Еще одно различие между моделями лежит в сфере возможности использования связи на основе соединений и связи без установления соединения. Модель OSI на сетевом уровне поддерживает оба типа связи, а на транспортном уровне — только связь на основе соединений (поскольку транспортные службы являются видимыми для пользователя). В модели TCP/IP на сетевом уровне есть только один режим связи (без установления соединения), но на транспортном уровне он поддерживает оба режима, предоставляя пользователям выбор. Этот выбор особенно важен для простых протоколов «запрос — ответ».

Критика модели и протоколов OSI

Ни описанные ранее модели (OSI и TCP/IP), ни их протоколы не являются совершенными. Довольно много критики было высказано по поводу обеих моделей.

Некоторые критические замечания мы рассмотрим в данном и в следующем разделах. Сначала проанализируем модель OSI, а затем TCP/IP.

В то время, когда вышло второе (английское. — *Примеч. ред.*) издание этой книги (1989), многим экспертам в данной области казалось, что модель OSI и ее протоколы завоюют весь мир и вытеснят все остальное. Этого не случилось. Почему? Может быть, полезно оглянуться и учесть некоторые из уроков этой истории. Основных причин неудачи модели OSI было четыре:

- несвоевременность;
- неудачная технология;
- неудачная реализация;
- неудачная политика.

Несвоевременность

Прежде всего рассмотрим причину номер один: несвоевременность. Для успеха стандарта чрезвычайно важно, в какое время он устанавливается. У Дэвида Кларка (David Clark) из M.I.T. есть теория стандартов, которую он называет *апокалипсисом двух слонов* (рис. 1.19).

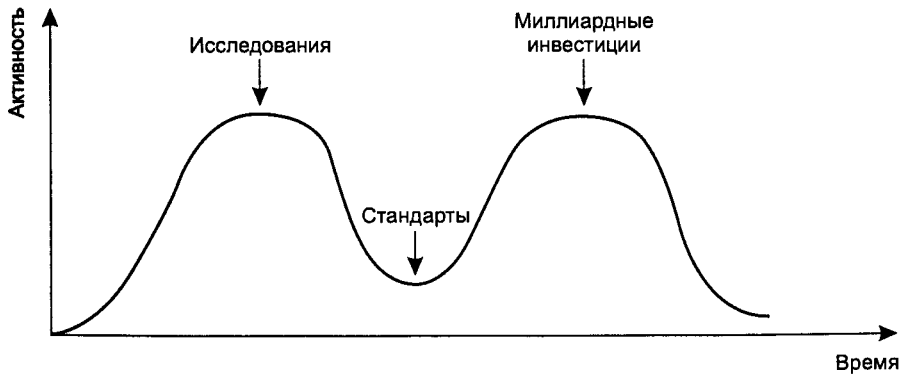


Рис. 1.19. Апокалипсис двух слонов

На этом рисунке изображена активность, сопровождающая любую новую разработку. Открытие новой темы вначале вызывает всплеск исследовательской активности в виде дискуссий, статей и собраний. Через некоторое время наступает спад активности, эту тему открывают для себя корпорации, и в результате в нее инвестируются миллиарды долларов.

Существенным является то, что стандарты пишутся именно в период между двумя «слонами». Если их создавать слишком рано, прежде чем закончатся исследования, предмет может оказаться еще слишком мало изучен и понят, что повлечет принятие плохих стандартов. Если создавать их слишком поздно, компании могут успеть вложить деньги в несколько отличные от стандартов технологии, так что принятые стандарты могут оказаться проигнорированными. Если интервал между двумя пиками активности будет слишком коротким (а все стре-

мятся делать деньги как можно быстрее), разработчики стандартов могут просто не успеть их выработать.

Теперь становится ясно, почему стандартные протоколы OSI потерпели неудачу. К моменту их появления среди исследовательских университетов уже получили широкое распространение конкурирующие с ними протоколы TCP/IP. И хотя волна инвестиций еще не обрушилась на данную область, рынок университетов был достаточно широк для того, чтобы многие разработчики стали осторожно предлагать продукты, поддерживающие протоколы TCP/IP. Когда же появился OSI, разработчики не захотели поддерживать второй стек протоколов; таким образом, начальных предложений не было. Каждая компания выжидала, пока первым начнет кто-нибудь другой, поэтому OSI так никто и не стал поддерживать.

Плохая технология

Второй причиной, по которой модель OSI не была реализована, оказалось несовершенство как самой модели, так и ее протоколов. Выбор семиуровневой структуры стал больше политическим решением, чем техническим. В результате два уровня (сеансовый и уровень представления) почти пусты, тогда как два других (сетевой и передачи данных) перегружены.

Эталонная модель OSI вместе с соответствующими определениями служб и протоколами оказалась невероятно сложной. Если сложить в стопку распечатку официального описания стандартов, получится кипа бумаги высотой в один метр. Модель тяжело реализуема и неэффективна в работе. В этом контексте вспоминается шутка Пола Мокапетриса (Paul Mockapetris), процитированная в издании Rose, 1993.

Вопрос. Что получится, если скрестить гангстера с международным стандартом?

Ответ. Человек, делающий вам предложения, которые вы не способны понять.

Еще одна проблема, помимо невозможности понять стандарты OSI, заключалась в том, что некоторые функции, такие как адресация, управление потоком и обработка ошибок, повторялись снова и снова в каждом уровне. Так, например, в книге Saltzer и др. (1984) указывается, что для того, чтобы контроль за ошибками был эффективным, он должен осуществляться на самом верхнем уровне, поэтому повторение его снова и снова на каждом уровне часто оказывается излишним и неэффективным.

Неудачная реализация

Учитывая огромную сложность модели и протоколов, громоздкость и медлительность первых реализаций не стали неожиданностью. Неудачу потерпели все, кто попытался реализовать эту модель. Поэтому вскоре понятие «OSI» стало ассоциироваться с плохим качеством. И хотя со временем продукты улучшились, ассоциации остались.

Первые реализации TCP/IP, основанные на Berkley UNIX, напротив, были достаточно хороши (не говоря уже о том, что они были открытыми). Они довольно быстро вошли в употребление, что привело к появлению большого сооб-

щества пользователей. Это вызвало исправления и улучшения реализации, в результате чего сообщество пользователей еще выросло. В данном случае обратная связь явно была положительной.

Неудачная политика

Из-за особенностей первоначальной реализации многие, особенно в университетских кругах, считали TCP/IP частью системы UNIX. А к системе UNIX в университетских кругах в 80-е годы испытывали чувства, средние между родительскими (в те времена некорректно по отношению к правам мужского населения называемые материнскими) и чувствами к яблочному пирогу.

С другой стороны, OSI считался детищем европейских телекоммуникационных министерств, Европейского сообщества и (позднее) правительства США. Все это было лишь отчасти верным, однако сама мысль о группе правительственных чиновников, пытающихся протолкнуть неудачный в техническом отношении стандарт в глотки бедных исследователей и программистов, прокладывавших компьютерные сети в траншеях, не способствовала продвижению этой модели. Кое-кто рассматривал это развитие в том же свете, что и заявления корпорации IBM, сделанные в 1960 году, о том, что PL/I будет языком будущего, или Министерства обороны, поправлявшего позднее это утверждение своим заявлением, что в действительности таким языком будет Ada.

Критика эталонной модели TCP/IP

У модели TCP/IP и ее протоколов также имеется ряд недостатков. Во-первых, в этой модели нет четкого разграничения концепций служб, интерфейса и протокола. При разработке программного обеспечения желательно провести четкое разделение между спецификацией и реализацией, что весьма тщательно делает OSI и чего не делает TCP/IP. В результате модель TCP/IP довольно бесполезна при разработке сетей, использующих новые технологии.

Во-вторых, модель TCP/IP отнюдь не является общей и довольно плохо описывает любой стек протоколов, кроме TCP/IP. Так, например, описать технологию Bluetooth с помощью модели TCP/IP совершенно невозможно.

В-третьих, хост-сетевой уровень в действительности не является уровнем в том смысле, который обычно используется в контексте уровневых протоколов. Это скорее интерфейс между сетью и уровнями передачи данных. Различие между интерфейсом и уровнем является чрезвычайно важным, и здесь не следует быть небрежным.

В-четвертых, в модели TCP/IP не различаются физический уровень и уровень передачи данных. Об этом различии даже нет упоминания. Между тем они абсолютно разные. Физический уровень должен иметь дело с характеристиками передачи информации по медному кабелю, оптическому волокну и по радио, тогда как задачей уровня передачи данных является определение начала и конца кадров и передача их с одной стороны на другую с требуемой степенью надежности. Правильная модель должна содержать их как два различных уровня. В модели TCP/IP этого нет.

И наконец, хотя протоколы IP и TCP были тщательно продуманы и неплохо реализованы, многие другие протоколы были созданы несколькими студентами, работавшими над ними, пока это занятие им не наскучило. Реализации этих протоколов свободно распространялись, в результате чего они получили широкое признание, глубоко укоренились, и теперь их трудно заменить на что-либо другое. Некоторые из них в настоящее время оказались серьезным препятствием на пути прогресса. Например, протокол виртуального терминала TELNET, созданный еще для механического терминала типа Teletype, работавшего с огромной скоростью 10 символов в секунду. Ему ничего не известно о графических интерфейсах пользователя и о мышках. Тем не менее сейчас, 25 лет спустя, он все еще широко используется.

Подытожим сказанное. Несмотря на все недостатки, *модель* OSI (кроме сеансового уровня и уровня представления) показала себя исключительно полезной для теоретических дискуссий о компьютерных сетях. *Протоколы* OSI, напротив, не получили широкого распространения. Для TCP/IP верно обратное: *модель* практически не существует, тогда как *протоколы* чрезвычайно популярны. Поскольку ученые-компьютерщики любят получать свою часть пирога, в этой книге мы будем использовать модифицированную модель OSI, но рассматривать будем в основном протоколы TCP/IP и родственные им, а также новые протоколы вроде 802, SONET и Bluetooth. В результате в качестве точки отсчета для всей книги мы будем использовать гибридную модель, изображенную на рис. 1.20.

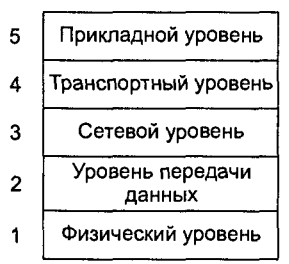


Рис. 1.20. Гибридная эталонная модель

Примеры сетей

Компьютерные сети бывают очень разными: большими и маленькими, всемирно известными и почти никому не известными. Они преследуют в своей работе разные цели, имеют разные масштабы, используют разные технологии. В этом разделе мы рассмотрим несколько примеров, помогающих осознать, насколько многообразен мир сетей. Первым примером будет самая известная сеть сетей, Интернет. Вы узнаете, как она появилась, как эволюционировала и какие технологии при этом использовались. Затем мы обратимся к технологии ATM, которая часто служит ядром больших (телефонных) сетей. Технически ATM довольно сильно от-

личается от Интернета, их даже можно в некотором смысле противопоставить. Мы представим также основную технологию, использующуюся при создании локальных вычислительных сетей, — Ethernet. Наконец, последним примером в этом разделе будет стандарт беспроводных локальных сетей IEEE 802.11

Интернет

Для начала следует еще раз напомнить о том, что Интернет вообще не является сетью, это собирательное название разных сетей, использующих определенные общие протоколы и предоставляющие определенные сервисы. Эта система необычна тем, что ее никто специально не планировал и не контролировал. Чтобы лучше понять, почему так получилось, мы начнем с самых истоков существования Интернета. В качестве прекрасного пособия по истории Интернета можно порекомендовать книгу, которую написал Джон Нотон (John Naughton) в 2000 году. Это редкое издание, потому что оно не только легко читается, но и содержит двадцатистраничный библиографический список параллельных мест и цитат, которые будут полезны людям, всерьез занимающимся историей. Часть материала, представленного далее, основывается именно на этой книге.

ARPANET

История глобальных сетей началась в конце 50-х годов. В самый разгар холодной войны Министерство обороны США пожелало иметь сеть, которая могла бы пережить даже ядерную войну. В то время все военные телекоммуникации базировались на общественной телефонной сети, которая была сочтена слишком уязвимой. Графически эта уязвимость демонстрируется на рис. 1.21. Здесь черными точками обозначены коммутационные станции, с каждой из которых были связаны тысячи абонентов. Эти коммутаторы, в свою очередь, являлись абонентами для станций более высокого уровня — междугородных. Междугородные станции формировали национальные сети. При этом степень резервной избыточности была минимальной. Уязвимость заключалась в том, что потеря всего одного ключевого коммутатора или междугородной станции разделила бы сеть на изолированные участки.

Для решения этой проблемы Министерство обороны обратилось к корпорации RAND. Один из ее работников, Пол Бэрен (Paul Baran), разработал проект высоконадежной распределенной сети (рис. 1.21, *б*). Поскольку по линиям такой большой длины тяжело было бы передать аналоговый сигнал с допустимым уровнем искажений, Бэрен предложил передавать цифровые данные и использовать технологию коммутации пакетов. Им было написано несколько отчетов для Министерства обороны, в которых описывались подробности реализации его идей. Пентагону понравилась предложенная концепция, и компании AT&T (тогдашнему монополисту в США по части телефонных сетей) было поручено разработать прототип. AT&T сразу же отклонила идеи Бэрена. Конечно, богатейшая и крупнейшая компания не могла позволить какому-то мальчишке указывать ей, как следует строить телефонные сети. Было заявлено, что бэреновскую сеть построить невозможно, и на этом проект был закрыт.

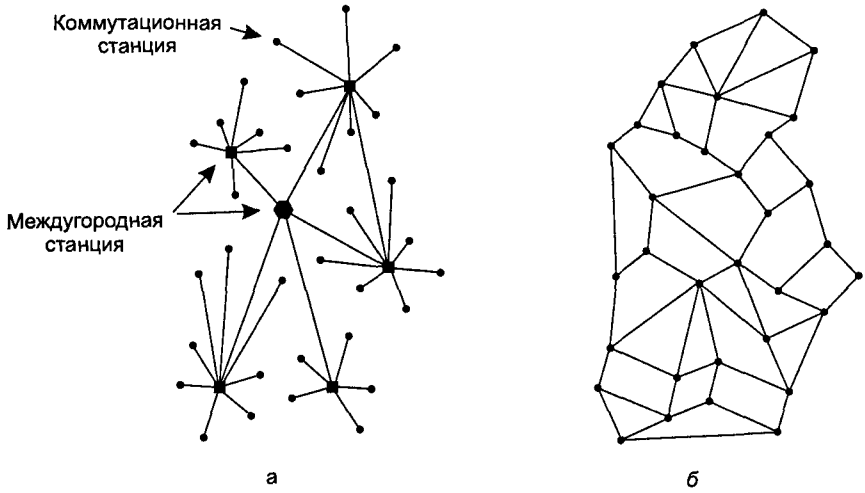


Рис. 1.21. Структура телефонной сети (*а*); предложенная Бэреном архитектура распределенной сети (*б*)

Прошло еще несколько лет, но Министерству обороны так и не было предложено никакой замены существующей системе оперативного управления. Чтобы понять, как развивались события дальше, мы вспомним октябрь 1957 года, когда в СССР был запущен первый в мире искусственный спутник Земли и тем самым основной соперник США получил преимущество в космосе. Тогда президент Эйзенхауэр задумался о том, кто же допустил такой прокол. И выяснилось, что армия, флот и ВВС США только зря проедают деньги, отпущенные Пентагоном на научные исследования. Было немедленно решено создать единую научную организацию под покровительством Министерства обороны, **ARPA** (Advanced Research Projects Agency, Управление перспективного планирования научно-исследовательских работ). У ARPA не было ни ученых, ни лабораторий. У нее вообще практически ничего не было, за исключением небольшого офиса и скромного (по меркам Пентагона) бюджета. ARPA занималось тем, что выбирало из множества предлагаемых университетами и компаниями проектов наиболее перспективные и организовывало выделение грантов под эти проекты и заключение контрактов с этими организациями.

Все первые годы своего существования ARPA пыталось определиться с направлением своей деятельности, пока внимание ее директора Ларри Робертса (Larry Roberts) не привлекли компьютерные сети. Он наладил контакты с различными экспертами, пытаясь понять, какие разработки могут представлять наибольший интерес для Министерства обороны. Один из экспертов, Весли Кларк (Wesley Clark), предложил построить подсеть с коммутацией пакетов, где каждый хост имел бы собственный маршрутизатор, как показано на рис. 1.8.

После преодоления собственного скептицизма Робертс все же решился приобрести эту идею и представил некий смутный отчет, касающийся этого, на симпозиуме ACM SIGOPS, посвященном принципам работы операционных систем.

Симпозиум состоялся в Гетлингбурге, штат Теннесси, в конце 1967 года (Roberts, 1967). К большому удивлению Робертса, он услышал доклад, в котором описывалась очень похожая система, причем эта система была не только спроектирована, но и реализована под руководством Дональда Дэвиса (Donald Davis) в Национальной физической лаборатории (NPL) Англии. Разработанная NPL сеть, конечно, не охватывала всю страну — она вообще соединяла лишь несколько компьютеров на территории организации, но ее реализация доказала, что пакетная коммутация может с успехом применяться на практике. Более того, то, что услышал Робертс, практически цитировало отвергнутую когда-то разработку Бэрена! Директор ARPA уехал из Гетлингбурга с твердым намерением создать в Америке то, что позднее будет названо **ARPANET**.

Подсеть должна была состоять из специализированных мини-компьютеров, называемых **IMP** (Interface Message Processor), соединенных линиями связи, передающими информацию со скоростью 56 Кбит/с. Для повышения надежности каждый IMP должен был соединяться как минимум с двумя другими IMP. Подсеть должна была быть дейтаграммной, чтобы в случае если какие-либо линии и IMP разрушатся, сообщения могли бы автоматически выбрать альтернативный путь.

Каждый узел сети должен был состоять из IMP и хоста, находящихся в одной комнате и соединенных коротким проводом. Хост мог пересылать своему IMP сообщения длиной до 8063 бит, которые IMP разбивал на пакеты, как правило, по 1008 бит, и пересылал их далее, независимо друг от друга, к пункту назначения. Пакет пересылался дальше только после того, как он был получен целиком, — таким образом, это была первая электронная коммутирующая пакеты сеть с промежуточным хранением.

Затем агентство ARPA предложило тендер на строительство подсети. В тендере участвовали двенадцать компаний. Оценив предложения, агентство ARPA выбрало BBN, консалтинговую фирму в Кембридже, штат Массачусетс, и в декабре 1968 года подписало с ней контракт на постройку подсети и создание для нее программного обеспечения. BBN решило использовать специально модифицированные мини-компьютеры Honeywell DDP-316 с 12 Кбайт 16-разрядных слов оперативной памяти в качестве IMP. У IMP не было дисков, поскольку движущиеся детали были сочтены ненадежными. Их соединили линиями с пропускной способностью по 56 Кбит/с, арендованными у телефонных компаний. Хотя в наше время 56 Кбит/с — это выбор подростков, которые еще не могут позволить себе ADSL или прокладку качественного кабеля, в 1968 году ничего более высокоскоростного просто не существовало.

Программное обеспечение было разбито на две части: для подсети и хостов. Подсетевое программное обеспечение состояло из части соединения хост — IMP со стороны IMP, протокола IMP—IMP и протокола между IMP-источником и IMP-приемником, разработанного для улучшения надежности. Оригинальная структура сети ARPANET показана на рис. 1.22.

Вне подсети также требовалось программное обеспечение, а именно: соединение хост—IMP со стороны хоста, протокол хост—хост и прикладные программы. Как вскоре выяснилось, фирма BBN полагала, что ее задача ограничивается приемом сообщения на линии хост—IMP и передачей его на линии IMP—хост приемника.

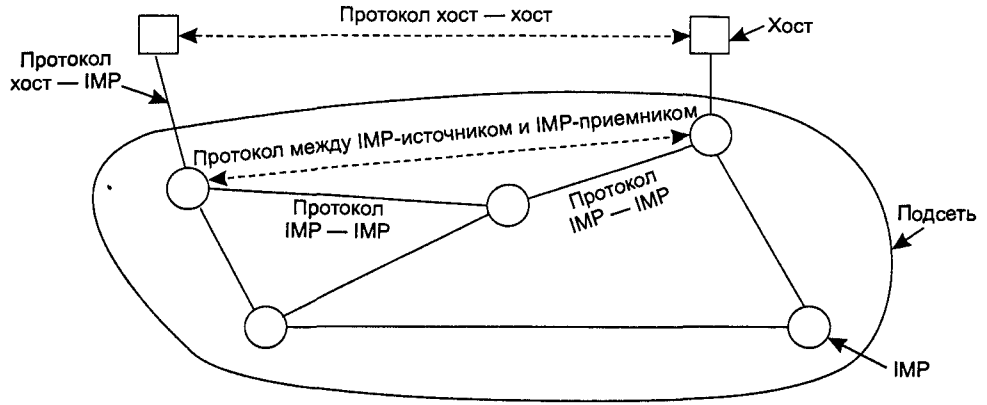


Рис. 1.22. Оригинальная структура сети ARPANET

Чтобы решить проблему программного обеспечения для хостов, Ларри Робертс летом 1969 года созвал совещание сетевых исследователей, большей частью аспирантов, в городе Сноуберд (Snowbird), штат Юта. Аспиранты ожидали, что какой-нибудь эксперт в области сетей объяснит им устройство сети и его программное обеспечение, после чего распределит между ними работу. С изумлением они обнаружили, что не было ни специалиста по сетям, ни плана. Они должны были сами решать, что нужно сделать.

Тем не менее в декабре 1969 года удалось запустить экспериментальную сеть, состоящую из четырех узлов, расположенных в Калифорнийском университете в Лос-Анджелесе (UCLA), Калифорнийском университете в Санта-Барбаре (UCSB), Исследовательском институте Стэнфорда (SRI, Stanford Research Institute) и университете штата Юта. Были выбраны эти четыре университета, поскольку у них был большой опыт общения с агентством ARPA; кроме того, у всех имелись различные и совершенно несовместимые компьютеры-хосты (чтобы было веселее). Сеть быстро росла по мере создания и установки новых IMP. Вскоре она охватила все Соединенные Штаты. На рис. 1.23 показано, как быстро росла сеть ARPANET в первые три года.

Помимо помощи развивающейся сети ARPANET, агентство ARPA также финансировало исследовательские работы по спутниковым сетям и разработку мобильных пакетных радиосетей. На одной знаменитой демонстрации грузовик, который ездил по Калифорнии, посылал сообщения по пакетной радиосети в SRI, которые затем передавались по ARPANET на Атлантическое побережье США и по спутниковой сети транслировались в University College в Лондоне. Таким образом, исследователь в грузовике мог работать с компьютером, находящимся в Лондоне.

При этой демонстрации также выяснилось, что имеющиеся протоколы сети ARPANET непригодны для работы с объединенными сетями. В результате были произведены дополнительные исследования в области протоколов, завершившиеся изобретением модели и протоколов TCP/IP (Cerf и Kahn, 1974). TCP/IP

был специально разработан для управления обменом данными по интерсетям, что становилось все более и более важным по мере подключения все новых сетей к ARPANET.

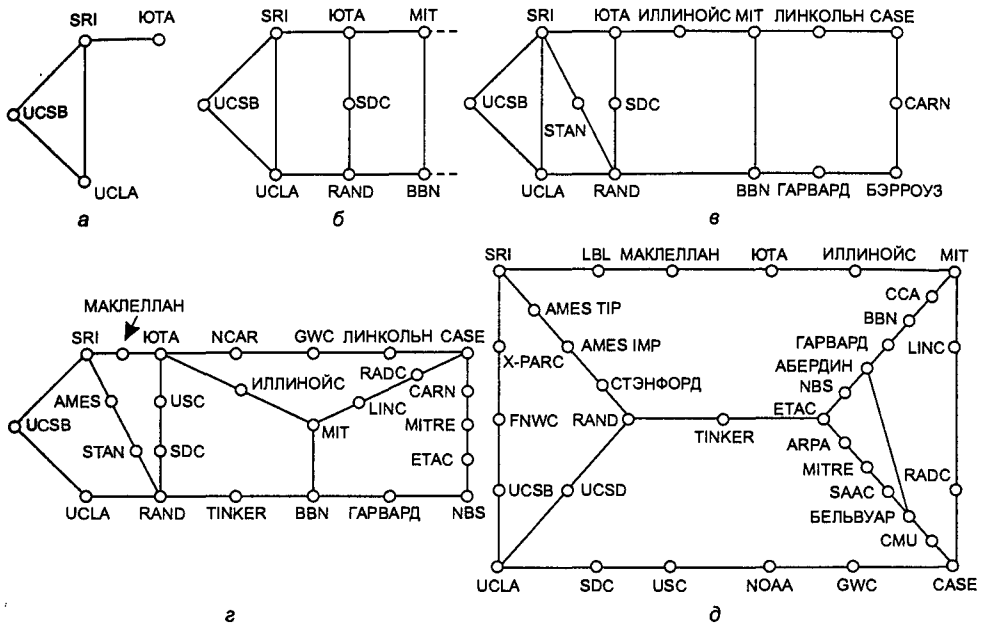


Рис. 1.23. Рост сети ARPANET: декабрь 1969 (*а*); июль 1970 (*б*); март 1971 (*в*); апрель 1972 (*г*); сентябрь 1972 (*д*)

Чтобы поощрить принятие новых протоколов, ARPA заключило несколько контрактов с BBN и Калифорнийским университетом в Беркли для интеграции этих протоколов в Berkeley UNIX. Исследователи в Беркли разработали удобный программный интерфейс для выхода в сеть (сокеты), а также написали множество приложений, утилит и управляющих программ, чтобы упростить работу с сетью.

Время было выбрано прекрасно. Многие университеты только что приобрели второй или третий компьютер VAX и ЛВС, чтобы их соединить, но у них не было сетевого программного обеспечения. С появлением системы UNIX 4.2 BSD, в которую вошли TCP/IP, сокеты и большое количество сетевых утилит, полный пакет был принят немедленно. Кроме того, TCP/IP позволял легко соединить локальную сеть с ARPANET, что многие и делали.

В течение 80-х годов к ARPANET были подсоединен еще ряд сетей, в основном ЛВС. По мере роста размеров глобальной сети задача поиска хостов становилась все сложнее. В результате была создана система **DNS** (Domain Name System — служба имен доменов), позволившая организовать компьютеры в домены и преобразовывать имена хостов в IP-адреса. С тех пор DNS стала обоб-

щенной распределенной системой баз данных, хранящей имена хостов и доменов. Мы рассмотрим ее более подробно в главе 7.

NSFNET

В конце 70-х Национальный научный фонд США (NSF, National Science Foundation) пришел к выводу, что сеть ARPANET оказывает огромное влияние на исследовательские работы университетов, позволяя ученым всей страны обмениваться информацией и совместно работать над проектами. Однако для получения доступа к ARPANET университет должен был заключить контракт с Министерством обороны, которого у многих университетов не было. Ответом NSF стала идея создания сети-преемника ARPANET, которая была бы открыта для всех университетских исследовательских групп. Чтобы начать с чего-нибудь конкретного, Национальный научный фонд решил построить сетевую магистраль, соединив ею шесть суперкомпьютерных центров в Сан-Диего, Боулдере, Шампейне, Питтсбурге, Итаке и Принстоне. К каждому суперкомпьютеру был присоединен небольшой микрокомпьютер LSI-11, называемый **фаззбол** (fuzzball). Эти мини-компьютеры соединили выделенными линиями по 56 Кбит/с и сформировали подсеть по той же аппаратной технологии, которая использовалась в ARPANET. Однако программная технология была другой — мини-компьютеры с самого начала использовали протокол TCP/IP, составляя, таким образом, первую в мире глобальную сеть на основе протокола TCP/IP.

Национальный научный фонд также профинансировал создание нескольких (всего около 20) региональных локальных сетей, соединенных с магистралью, что позволило пользователям в тысячах университетов, исследовательских лабораторий, библиотек и музеев получить доступ к суперкомпьютерам. Вся сеть, состоящая из магистрали и региональных сетей, получила имя **NSFNET**. Она соединялась с ARPANET через линию между IMP и микрокомпьютером в компьютерном зале университета Карнеги — Меллона (Carnegie — Mellon University). Первоначальная магистраль сети NSFNET изображена на рис. 1.24.

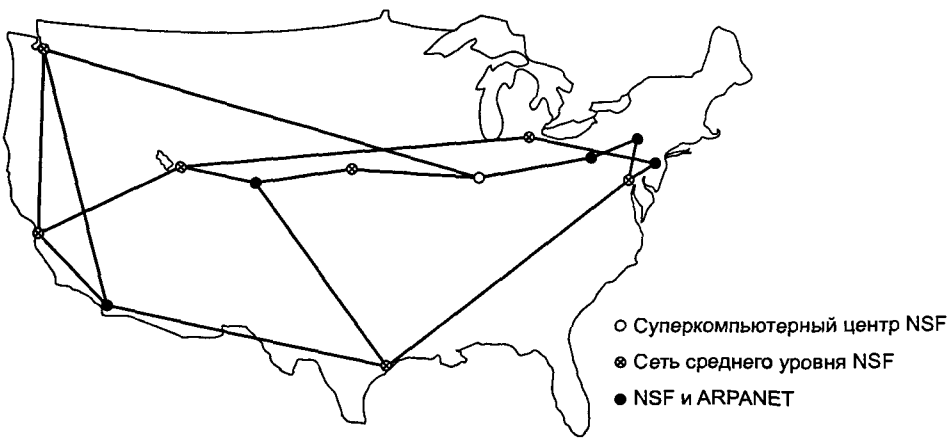


Рис. 1.24. Магистраль сети NSFNET в 1988 году

Сеть NSFNET имела мгновенный успех, ей предсказывали большое будущее. Национальный научный фонд сразу же после завершения работы над NSFNET начал планировать следующую сеть и с этой целью подписал контракт с базирующимся в штате Мичиган консорциумом MERIT. Для создания второй версии магистрали сети у оператора междугородной связи MCI (Microwave Communications, Inc. — компания, объединившаяся с тех пор с WorldCom) были арендованы волоконно-оптические каналы с пропускной способностью в 448 Кбит/с. В качестве маршрутизаторов использовались IBM PC-RT (RT-PC — RISC Technology Personal Computer — персональный компьютер на основе процессора с сокращенным набором команд). Вскоре и этого стало недостаточно, и вторая магистраль была ускорена до 1,5 Мбит/с.

Рост отрасли продолжался, но Национальный научный фонд понимал, что правительство не сможет финансировать развитие сетей постоянно. Кроме того, коммерческие организации выражали желание поучаствовать в общем деле, но уставом фонда им было запрещено использовать сети, за которые заплатил Национальный научный фонд. Впоследствии Национальный научный фонд поддержал создание компаниями MERIT, MCI и IBM некоммерческой корпорации ANS (Advanced Networks and Services, Inc.) в качестве первого шага на пути коммерциализации. В 1990 году ANS вступила во владение сетью NSFNET и усовершенствовала линии со 1,5 Мбит/с до 45 Мбит/с, сформировав **ANSNET**. Эта сеть проработала пять лет, после чего была продана компании America Online. Но к тому времени уже появилось множество коммерческих фирм, предлагающих свои услуги в области IP-коммуникаций. Стало понятно, что государству не удастся выдержать конкуренцию с ними и оно должно уйти из этого бизнеса.

В декабре 1991 года Конгресс США утвердил закон, разрешающий создание сети **NREN** (National Research and Education Network — государственная научно-исследовательская и образовательная сеть), являвшейся преемницей сети NSFNET, но работающей на гигабитных скоростях. Целью было создание государственной сети, работающей на скорости 3 Гбит/с, до конца тысячелетия. Эта сеть должна была служить прототипом для многократно обсуждавшейся информационной супермагистрали.

Для того чтобы облегчить переход с одних сетей на другие и гарантировать, что все региональные сети могут связаться друг с другом, Национальный научный фонд заключил контракт с четырьмя различными сетевыми операторами об организации пунктов доступа к сети (NAP, Network Access Point). Этими операторами были компании PacBell (Сан-Франциско), Ameritech (Чикаго), MFS (Вашингтон) и Sprint (Нью-Йорк, с которым для удобства NAP были объединены Пеннсаукен и Нью-Джерси). Каждый сетевой оператор, который хотел предоставлять услуги по соединению региональных сетей NSF, должен был подключиться ко всем пунктам NAP.

Таким образом, пакет, пересылаемый с одной сети в другую, мог выбирать, по какому каналу перемещаться от одного пункта NAP до другого. Из-за этого операторы были вынуждены соперничать друг с другом в области цен и предоставляемых услуг, как, собственно, и было задумано. Концепция единой магистрали была заменена коммерчески управляемой конкурентной инфраструктурой. Мно-

гие любят критиковать государственные структуры США за их консерватизм, а между тем не кто иные, как Министерство обороны и государственный Национальный научный фонд, создали все необходимые условия для развития Интернета, а затем передали свои закрытые разработки массовому пользователю.

В 90-х годах в других странах и регионах также были построены сети, сравнимые с NSFNET. Так, в Европе EuropaNET является IP-магистралью для исследовательских организаций, а EBONE представляет собой коммерчески ориентированную сеть. Обе сети соединяют большое число европейских городов. Скорость каналов изначально составляла 2 Мбит/с, но впоследствии была увеличена до 34 Мбит/с. В конечном счете сетевая инфраструктура в Европе, как и в США, превратилась в промышленную отрасль.

Применение Интернета

После того как 1 января 1983 года TCP/IP стал единственным официальным протоколом, количество сетей, машин и пользователей, соединенных с ARPANET, быстро увеличивалось. Когда сети NSFNET и ARPANET объединились, рост стал экспоненциальным. Присоединились многочисленные региональные сети, была установлена связь с сетями в Канаде, Европе и сетями стран Тихоокеанского региона.

Примерно в середине 80-х это множество сетей стали называть интерсетью (internet), а впоследствии Интернетом (Internet), хотя официального открытия с каким-нибудь политиком, разбивающим бутылку шампанского о маршрутизатор или хост, не было.

Силами, удерживающими части Интернета вместе, являются эталонная модель TCP/IP и стек протоколов TCP/IP. TCP/IP, благодаря которому стали возможными глобальные службы, можно уподобить единой телефонной системе или принятию в XIX веке единого стандарта ширины железнодорожных путей.

Какой смысл вкладывается в понятие подключения к Интернету? По нашему определению машина считается находящейся в Интернете, если на ней действует стек протоколов TCP/IP, у нее есть IP-адрес и возможность посылать IP-пакеты на все остальные машины в Интернете. При этом, скажем, одной возможности отправки и получения e-mail недостаточно, поскольку электронная почта при помощи шлюзов может выходить за рамки Интернета. Кроме того, вопрос осложняется тем фактом, что очень много персональных компьютеров имеют возможность позвонить интернет-провайдеру при помощи модема, получить временный IP-адрес и возможность посылать IP-пакеты другим хостам Интернета. Есть смысл считать такие машины находящимися в Интернете на тот период, пока они подключены к маршрутизатору поставщика услуг.

Традиционно (имеется в виду период примерно с 1970 по 1990 год) у Интернета было четыре основные сферы применения:

1. **Электронная почта.** Возможность создавать, посылать и получать электронную почту появилась с первых дней существования сети ARPANET и до сих пор чрезвычайно популярна. Многие люди получают десятки сообщений в день и рассматривают это как свой основной способ взаимодействия с внешним миром, а также уверены, что e-mail гораздо важнее телефона и обычной

почты. На сегодняшний день почтовые программы могут работать на компьютерах любых типов.

2. **Новости.** Конференции являются специализированными форумами, в которых пользователи могут обмениваться сообщениями по какой-нибудь определенной теме. Существуют тысячи телеконференций по техническим и нетехническим вопросам, включая компьютеры, науку, отдых и политику. В каждой конференции есть свои правила этикета, свой стиль и обычаи. Горе тому, кто их нарушит.
3. **Удаленный доступ.** При помощи таких программ, как Telnet, Rlogin, ssh и т. п., пользователи Интернета могут регистрироваться на любой машине, на которой у них имеется учетная запись.
4. **Перенос файлов.** С помощью программ FTP можно копировать файлы с одной машины, подключенной к Интернету, на другую. Подобным образом доступно огромное количество статей, баз данных и другой информации.

Вплоть до начала 90-х Интернет был весьма популярен среди академических, государственных и промышленных исследователей. Однако одно новое приложение, WWW (World Wide Web — Всемирная паутина), поставило все с ног на голову, приведя в сеть миллионы новых, далеких от академического мира пользователей. Это приложение, созданное физиком из Европейского совета по ядерным исследованиям (CERN, Conseil Européen pour la Recherche Nucleaire) Тимом Бернерс-Ли (Tim Berners-Lee), не изменило возможности Сети, но облегчило их использование. Вместе с программой Mosaic viewer, написанной в Национальном центре по применению суперкомпьютеров (NCSA, National Center for Supercomputing Applications), приложение WWW сделало возможным размещение на сайте нескольких страниц информации, содержащей текст, изображения, звук и даже видео, со встроенными ссылками на другие страницы. При щелчке на ссылке пользователь сразу перемещается на страницу, на которую указывает эта ссылка. Например, у многих компаний имеется домашняя страница, содержащая ссылки на другие страницы, на которых представлена информация о продуктах, ценах, торговле, технической поддержке, а также информация для общения с сотрудниками, для акционеров и многое другое.

За очень короткое время в Интернете появляются многочисленные страницы, содержащие карты, таблицы бирж, каталоги библиотек, записанных радиопрограмм и даже страницы со ссылками на полные тексты книг, чьи авторские права потеряли силу (Марк Твен, Чарльз Диккенс и т. д.). Все больше появляется личных (домашних) страниц.

Разумеется, столь бурный спрос не мог не вызвать соответствующее предложение, и вот в 90-е годы появляется огромное количество **фирм — провайдеров услуг Интернета**. Эти коммерческие компании обеспечивают доступ в Интернет компьютеров, принадлежащих как частным лицам, так и корпоративным клиентам. Наиболее распространен «наборный доступ» (dial-up), то есть доступ с помощью модема и обычных коммутируемых телефонных сетей. При этом клиенты могут пользоваться электронной почтой, WWW и любыми другими службами Сети. В конце последнего десятилетия XX века к Интернету подключались де-

сятки миллионов новых пользователей в год, что совершенно изменило характер сети. Интернет из закрытой сети для военных и ученых превратился в некую коммунальную услугу, подобную телефонной сети. Нынешнее число пользователей Сети подсчитать невозможно, но оно, несомненно, исчисляется сотнями миллионов и скоро перевалит за миллиард.

Архитектура Интернета

В этом разделе мы попытаемся дать краткое описание сегодняшней структуры сети Интернет. Благодаря слиянию многих телефонных компаний с компаниями, предоставляющими доступ в Интернет, их сферы деятельности несколько смешались, и сейчас уже тяжело сказать, кто чем занимается. Поэтому приведенное описание оказывается, на самом деле, упрощенным по сравнению с реальной жизнью. В самом общем виде структура Интернета показана на рис. 1.25. Рассмотрим этот рисунок поподробнее.

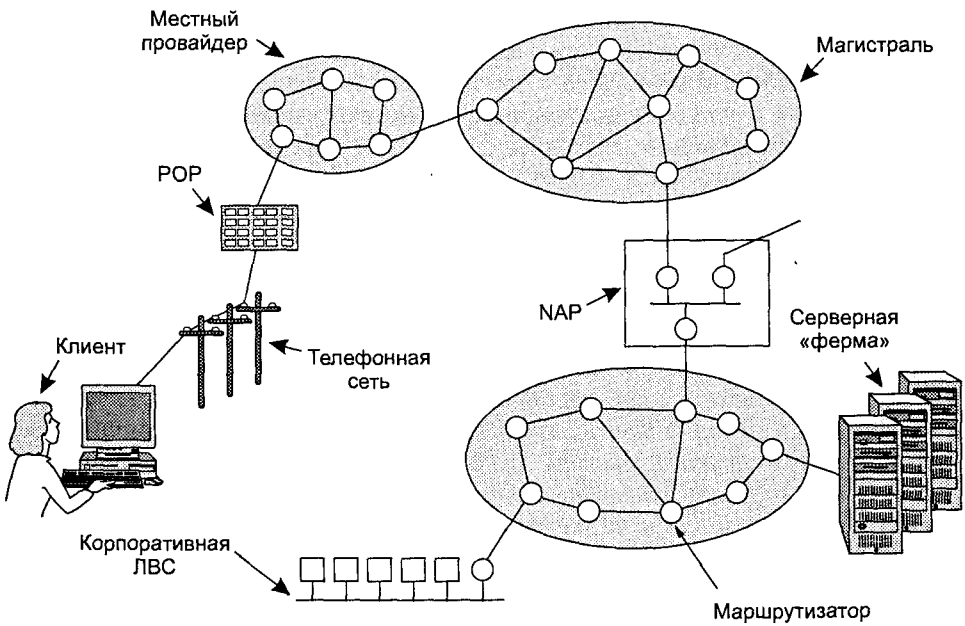


Рис. 1.25. Архитектура Интернета

Начнем с клиента, который сидит у себя дома. Предположим, он решил установить соединение с провайдером Интернета с помощью телефонной линии и модема. Модем — это специальное периферийное устройство ПК, которое преобразует цифровой сигнал компьютера в аналоговый сигнал, который можно передавать по телефонной сети. Итак, аналоговые сигналы приходят на **точку присутствия** (Point Of Presence, POP) провайдера, где они снимаются с телефонной линии и поступают в его региональную сеть. Начиная с этого момента, вся система работает только с цифровыми данными и использует коммутацию пакетов.

Если провайдером является местная телефонная компания, точка присутствия, скорее всего, будет расположена на телефонной станции — как раз там, где заканчивается линия, идущая напрямую от абонента. В ином случае точки входа могут располагаться на нескольких телефонных коммутационных станциях.

Региональная сеть провайдера Интернета состоит из взаимосвязанных маршрутизаторов в различных городах, которые он обслуживает. Если место назначения пакета — хост, обслуживаемый данным провайдером, то пакет просто доставляется туда. Если же это не так, то пакет передается оператору сетевой магистрали.

На самом верхнем уровне всей этой цепочки находится ряд магистральных операторов. В качестве таковых выступают компании типа AT&T, Sprint. В их ведении находятся крупные международные магистральные сети, образованные тысячами маршрутизаторов, соединенных волоконно-оптическими линиями с очень высокой пропускной способностью. Некоторые большие корпорации и хостинговые компании, имеющие собственные серверные системы («фермы»), обслуживающие тысячи запросов веб-страниц в секунду, соединяются с магистралями напрямую. Операторы магистральных сетей реализуют такую возможность путем сдачи в аренду помещений (так называемых **отелей для операторов связи**) для установки оборудования серверных «ферм». Обычно в этих же помещениях располагаются магистральные маршрутизаторы, что обеспечивает небольшую длину линии, а значит, высокую скорость обмена.

Если пакет предназначен для провайдера или другой компании, обслуживаемой магистральным оператором, он передается с магистрали на ближайший маршрутизатор, где происходит его отчуждение. Однако место назначения не обязательно будет относиться к данной магистрали — в мире их довольно много. Для того чтобы можно было перебросить пакет на другую магистраль, между ними существуют **точки входа в сеть** (Network Access Point, NAP). Они представляют собой специальные помещения, заполненные маршрутизаторами: каждую магистраль должен представлять по меньшей мере один маршрутизатор. Внутри этого помещения расположена также локальная сеть, объединяющая все находящееся в ней оборудование. Благодаря этому пакеты могут передаваться с маршрутизатора на маршрутизатор, то есть, фактически, с магистрали на магистраль. Кроме того, наиболее крупные магистрали связаны друг с другом не только через точки входа, но и напрямую. Это называется **частной равноранговой связью**. Один из парадоксов интернет-технологий заключается в том, что зачастую провайдеры, которые открыто конкурируют друг с другом в борьбе за клиентов, в то же самое время организуют частную равноранговую связь между собой (Metz, 2001).

На этом мы закончим наш краткий обзор архитектуры Интернета. Впереди еще много разделов, посвященных изучению отдельных компонентов этого вопроса: проектирования, алгоритмов, протоколов. Хочется напоследок отметить еще один нюанс: некоторые компании, объединяющие свои внутренние сети, часто используют те же технологии, которые используются в глобальной сети Интернет. Доступ к данным этих **интрасетей** обычно ограничивается пределами компании, но во всем остальном это тот же самый Интернет, только в миниатюре.

Сети на основе соединений: X.25, ретрансляция кадров, ATM

С самого появления компьютерных сетей шла война между теми, кто поддерживал идею дейтаграммных подсетей без установления соединения, и теми, кто высказывался за подсети на основе соединений. Сторонники последних — это зачастую выходцы из сообщества пользователей ARPANET/Internet. Не следует забывать о том, что изначальная идея Министерства обороны США состояла в создании такой сети, которая, даже будучи наполовину разрушенной в результате взрыва ядерной бомбы, могла бы функционировать бесперебойно. То есть основным параметром при создании таких сетей была устойчивость к повреждениям, а получением доходов от клиентов никто не был озабочен. Данный подход нашел свою реализацию в создании сетей без установления соединений, в которых маршрут каждого отдельно взятого пакета не зависит от остальных. Таким образом, выход из строя некоторого количества маршрутизаторов не мог привести к прекращению работы сети, поскольку при динамической перенастройке все равно находился какой-нибудь путь к месту назначения пакетов.

Самые ярые энтузиасты сетей на основе установления соединений — это люди, связанные с телефонными системами и компаниями. Как известно, абонент телефонной сети, прежде чем начать разговор, должен набрать номер и дождаться ответа. Примерно те же действия нужно совершить, чтобы соединиться с провайдером и передать или получить какие-нибудь данные. Соединение происходит при помощи телефонной сети. Оно является временным и поддерживается лишь до тех пор, пока абонент не повесил трубку или не отключился от провайдера. При этом все произносимые слова, как и передаваемые пакеты, следуют одним и тем же маршрутом. Если выходит из строя линия или коммутатор между начальным и конечным пунктом, то соединение разрывается. Именно это неприятное свойство не устраивало в свое время Министерство обороны.

Почему же тогда используются такие системы? Есть две причины этого:

- качество обслуживания;
- доход операторов таких сетей.

При предварительной установке соединения подсеть может зарезервировать для него некоторое количество ресурсов, например, буферное пространство или часть мощностей центрального процессора. Если другим абонентом производится попытка установить соединение, а ресурсы уже на исходе, то вызов отвергается и выдается сигнал, равнозначный сигналу «занято». Таким образом, если уж соединение установлено, значит, будет обеспечено надлежащее качество обслуживания. В сетях без установления соединения такого добиться сложно. Например, если на один и тот же маршрутизатор придет слишком большое количество пакетов, то он может «задохнуться» и потерять часть из них. Конечно, система организована так, что отправитель узнает о потерях и перешлет нужные пакеты, но общее качество обслуживания оставит тягостное впечатление. Такую сеть нельзя будет использовать для передачи аудио- и видеоинформации, если только она не будет практически свободна. Естественно, телефонные компании заботятся об

обеспечении нормальной передачи голоса, то есть аудиоинформации, отсюда их очевидные предпочтения одних технологий другим.

Вторая причина, по которой телефонные компании продолжают использовать сети, основанные на установлении соединения, — это возможность получать прибыль за выделяемое для работы пользователей время. Известно, что звонки на большие расстояния приходится оплачивать поминутно. Когда стали распространяться компьютерные сети, система поминутной оплаты вполне естественно переняла такую схему. Если вам приходится установить соединение, прежде чем начать передачу данных, то этот момент может стать началом отсчета времени работы в Сети. Если же такого момента соединения не существует, то ни о какой поминутной оплате не может быть и речи.

Между прочим, поддержка системы тарифных сборов обходится очень дорого. Если бы телефонные компании просто установили ежемесячную абонентскую плату за неограниченное время пребывания в Интернете, они сэкономили бы огромную сумму денег несмотря на то, что при этом возросла бы нагрузка на линии и оборудование. Однако некоторые политические и другие факторы не могут допустить введения такой системы оплаты. Между тем в других секторах услуг с населения взимается именно абонентская плата. Скажем, кабельное телевидение обязывает пользователей вносить ежемесячную фиксированную сумму, которая не зависит от того, сколько программ они посмотрели. А ведь здесь тоже можно было бы применить принцип оплаты просмотра каждой программы. Так сделано не было отчасти из-за дороговизны внедрения тарифной системы. Аналогично этому, многие парки развлечений продают только входные билеты, которые не ограничивают количество посещений разных аттракционов. А передвижные ярмарки с аттракционами, наоборот, берут деньги за каждую поездку на пластмассовой лошадке.

Таким образом, становится понятно, почему все сети, связанные с телефонной индустрией, никогда не откажутся от идеи подсетей с установлением соединений. Несколько удивляет, что и Интернет движется в том же направлении. Видимо, это объясняется попыткой повысить качество обслуживания, чтобы не возникало проблем с передачей мультимедийной информации. Мы еще вернемся к этому вопросу в главе 5. А теперь рассмотрим некоторые примеры сетей на основе соединений.

X.25 и ретрансляция кадров

Первым примером сети с установлением соединений является **X.25**. Это была первая общественная сеть передачи данных. Родилась она в 70-е годы, когда во всем мире действовали телефонные монополии. В каждой стране имелась только одна сеть передачи данных — телефонная. Чтобы подключиться к X.25, компьютер устанавливал соединение с удаленным компьютером, то есть совершал телефонный звонок. Этому соединению присваивался уникальный номер, по которому и определялось место назначения передаваемых пакетов (в такой сети одновременно могло быть много соединений). Пакеты имели очень простой формат. Они состояли из трехбайтного заголовка, после которого следовало поле данных размером до 128 байт. Заголовок состоял из 12-битного номера соединения, порядкового

номера пакета, номера для подтверждения доставки и еще некоторой служебной информации. Сеть X.25 работала с переменным успехом в течение одного десятилетия.

В 80-е годы на смену X.25 пришла **технология ретрансляции кадров** (Frame Relay). На ее основе была построена сеть без контроля ошибок и передачи. Поскольку она была ориентирована на установление соединения, то пакеты доставлялись в нужном порядке (если, конечно, доставлялись вообще). Эти три свойства (отсутствие контроля передачи, отсутствие контроля ошибок и доставка в нормальном порядке) сделали сеть с ретрансляцией кадров похожей на некую глобальную по охвату и локальную по принципу действия сеть. Она широко применялась для соединения ЛВС разных офисов одного предприятия. Данная технология имела умеренный успех, но до сих пор кое-где используется.

Асинхронный режим передачи (ATM)

Наконец, еще одна очень важная технология с установлением соединения носит название **ATM** (Asynchronous Transfer Mode — асинхронный режим передачи). Причиной столь странного наименования является то, что в большинстве систем, связанных с телефонной сетью, передача осуществляется с синхронизацией (то есть жестко привязана ко времени), а в ATM это не так.

Появилась эта технология в начале 90-х и вошла в жизнь компьютерных сетей во многом благодаря бурной рекламной кампании (Ginsburg, 1996; Goralsky, 1995; Ibe, 1997; Kim и др., 1994; Stallings, 2000). Было заявлено, что ATM решит абсолютно все мировые проблемы, связанные с компьютерными сетями и телекоммуникациями, объединив передачу голоса и данных, кабельное телевидение, телеграф, телетайп, почтовых голубей и консервные банки, чем бы они ни соединялись — проводами, дымовыми сигналами, звуками тамбуринов, — в единую интегрированную систему, которая умеет все и удовлетворит всех. Как ни странно, этого не произошло. По большей части, проблемы ATM были похожи на обсуждавшиеся ранее проблемы OSI: несвоевременность, неудачные технологии, реализация и политика. Только что отбившись от давления телефонных компаний, Интернет-сообщество увидело и в ATM борца против телефонных сетей — своего, так сказать, сподвижника. Но на самом деле все оказалось не совсем так, и на этот раз даже самые ярые фанатики дейтаграммных сетей убедились в том, что качество обслуживания при предоставлении такого доступа в Интернет оставляет желать лучшего.

В целом технология ATM имела больший успех, чем OSI, и продолжает использоваться глубоко в недрах телефонной системы и поныне. Занимается она там передачей IP-пакетов. Поскольку ее использование можно заметить, только очень сильно углубившись в подробности передачи данных, рядовые пользователи даже не подозревают о существовании ATM, а между тем она живет и здравствует.

Виртуальные каналы ATM

Поскольку ATM является технологией, основанной на предварительном соединении, перед посылкой данных необходимо отправить пакет для установки связи.

По мере прохождения этого установочного пакета по узлам подсети все маршрутизаторы делают записи в своих внутренних таблицах, отмечая тем самым наличие соединения и резервируя для него определенные ресурсы. Устанавливаемые соединения в терминологии ATM часто называют **виртуальными каналами**, по аналогии с физическими каналами, имеющимися в телефонной системе. В большинстве сетей ATM также имеются **постоянные виртуальные каналы**, представляющие собой постоянные соединения между двумя удаленными друг от друга хостами. Они напоминают выделенные телефонные линии. У каждого соединения (как временного, так и постоянного) есть свой уникальный идентификатор. Виртуальный канал показан на рис. 1.26.

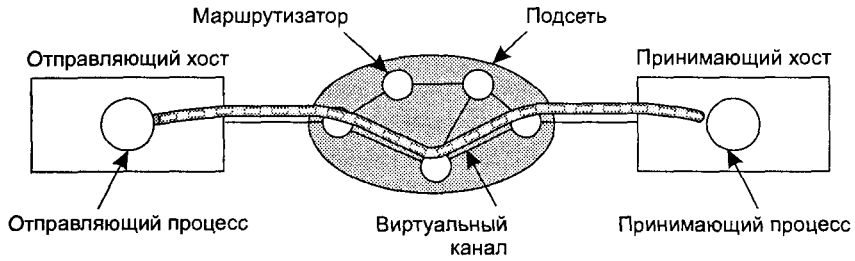


Рис. 1.26. Виртуальный канал

После установки соединения любая из сторон может начать передачу данных. В основе технологии ATM лежит идея передачи информации небольшими пакетами фиксированной длины, называемых **ячейками**. Ячейки имеют размер 53 байта, из которых 5 байт составляют заголовок, а 48 несут полезную нагрузку, как показано на рис. 1.27. Частью заголовка является идентификатор соединения, поэтому как хосты (отправляющий и принимающий), так и все маршрутизаторы, встречающиеся по пути, знают, какая ячейка принадлежит какому соединению. Эта информация позволяет маршрутизатору направить каждый входящий пакет в нужном направлении. Определение маршрута производится на аппаратном уровне с высокой скоростью. На самом деле, основным аргументом в пользу ячеек фиксированной длины является как раз тот факт, что можно легко организовать аппаратную обработку маршрутизаторами коротких пакетов известной длины. Маршрутизацию IP-пакетов можно осуществить только программно, что гораздо медленнее. Еще одним плюсом ATM является то, что можно настроить аппаратуру на размножение входящего пакета на множество выходных линий. Такое свойство требуется для организации некого подобия телевизионной программы, которая методом широковещания пересылается большому количеству абонентов. Наконец, небольшие ячейки вряд ли смогут заблокироваться линию надолго, что гарантирует определенный уровень качества обслуживания.

Все ячейки следуют по одному и тому же маршруту, поэтому верный порядок их доставки гарантируется, хотя сама доставка — нет. Если отправляется вначале ячейка № 1, затем ячейка № 2, то (если они обе прибывают на принимающий хост) не может возникнуть такой ситуации, когда придет сначала № 2, затем № 1. Однако не исключено, что один или даже оба пакета по пути потеряются. Вос-

станавливать их должны протоколы более высоких уровней. И все же обратите внимание: хотя эта технология не дает абсолютных гарантий доставки, это лучше, чем то, что может обеспечить Интернет. Там пакеты могут не только потеряться, но может быть перепутан порядок их следования. ATM же обеспечивает правильный порядок доставки ячеек.

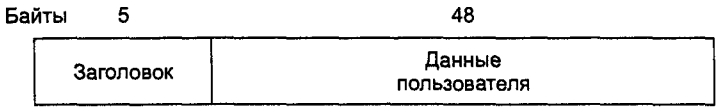


Рис. 1.27. Ячейка ATM

Сети ATM организованы по принципу традиционных региональных сетей, с линиями и коммутаторами (маршрутизаторами). Наиболее распространенные скорости работы ATM — это 155 и 622 Мбит/с, но можно использовать и более высокие скорости. Скорость 155 Мбит/с была выбрана потому, что примерно такая скорость нужна для передачи телевидения высокого разрешения. Выбор скорости в 155,52 Мбит/с был сделан для совместимости с передающей системой фирмы SONET AT&T (речь о ней пойдет в главе 2). Скорость 622 Мбит/с была выбрана для одновременной передачи четырех 155-мегабитных каналов.

Эталонная модель ATM

У ATM есть своя эталонная модель, отличная от моделей OSI и TCP/IP. Эта модель показана на рис. 1.28. Она состоит из трех уровней: физического, ATM и адаптационного ATM-уровня, — а также включает верхние уровни, используемые пользователем.

Физический уровень взаимодействует с физической средой передачи данных: напряжениями, длительностями битов и т. п. ATM не предписывает на этот счет каких-нибудь правил. Напротив, сообщается, что ячейки ATM могут передаваться по проводам или по оптоволоконным кабелям сами собой, а также в упакованном виде в качестве данных по любому другому носителю. Другими словами, ATM разрабатывалась как независимая от физического носителя система.

ATM-уровень имеет дело с ячейками и их передачей. Он описывает схему ячейки и значение полей заголовков. Кроме того, этот уровень занимается установлением и освобождением виртуальных каналов связи. Здесь же осуществляется управление перегрузкой.

Поскольку большинство приложений не желают работать напрямую с ячейками (хотя некоторые могут это делать), уровень, расположенный над уровнем ATM, был разработан для предоставления пользователям возможности посылать пакеты, превосходящие размер ячейки. Интерфейс ATM разбивает такой пакет на ячейки, передает их по отдельности, после чего вновь собирает из них пакет на противоположной стороне. Этот уровень называется **уровнем адаптации ATM**, или **уровнем AAL**.

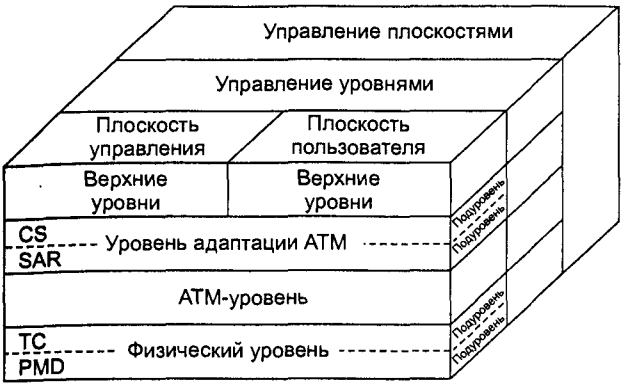


Рис. 1.28. Эталонная модель ATM

В отличие от более ранних двумерных эталонных моделей, модель ATM является трехмерной, как показано на рис. 1.28. **Плоскость пользователя** занимается транспортировкой данных, управлением потоками, исправлением ошибок и другими функциями. Задачей же **плоскости управления** является управление соединениями. Управляющие функции уровней и плоскостей занимаются управлением ресурсами и межуровневой координацией.

Физический уровень и уровень AAL подразделяются на два подуровня; нижний подуровень выполняет работу, а верхний (подуровень конвергенции) предоставляет вышестоящему уровню соответствующий интерфейс. Функции уровней и подуровней приведены на рис. 1.29.

Уровень OSI	Уровень ATM	Подуровень ATM	Функция
3/4	AAL	CS	Предоставление стандартного интерфейса (конвергенция)
		SAR	Сегментация и повторная сборка
2/3	ATM		Управление потоком Генерация/извлечение заголовков ячеек Управление виртуальным каналом/путем Объединение/разъединение ячеек
2	Физический	TC	Разборка ячеек Подсчет и проверка контрольных сумм в заголовках Генерация ячеек Упаковка ячеек в конверты или распаковка Генерация кадров
1		PMD	Управление временными параметрами сигналов Физический доступ к сети

Рис. 1.29. Уровни и подуровни ATM, их функции

Зависящий от физического носителя подуровень **PMD** (Physical Medium Dependent) обеспечивает интерфейс с физическим кабелем. Он пересылает биты и управляет временными параметрами сигналов. Этот уровень будет различным для различных носителей и кабелей.

Другим подуровнем физического уровня является подуровень **конвергенции передачи** (Transmission Convergence, TC). Уровень конвергенции передачи пересылает ячейки в виде последовательности битов уровню PMD. Это несложно. С другой стороны, подуровень конвергенции передачи принимает поток битов от подуровня PMD. Его задачей является преобразование этого битового потока в поток ячеек для уровня ATM. Он выполняет все задачи, связанные с определением начала и конца ячеек во входном потоке. В модели ATM эти функции относятся к физическому уровню. В модели OSI, а также во многих старых сетях превращение сырого потока битов в последовательность кадров или ячеек выполнялось уровнем передачи данных.

Как уже упоминалось, уровень ATM управляет ячейками, включая их создание и транспортировку. Именно здесь располагается большая часть интересных аспектов модели ATM. Этот уровень представляет собой смесь уровня передачи данных модели OSI и сетевого уровня, однако он не разбит на отдельные подуровни.

Уровень AAL состоит из двух подуровней: **подуровня сегментации и повторной сборки** (Segmentation And Reassembly, SAR) и **подуровня конвергенции** (Convergence Sublayer, CS). Нижний подуровень разбивает пакеты на ячейки при передаче и восстанавливает пакеты из ячеек при приеме. Верхний подуровень обеспечивает предоставление системами ATM определенных услуг различным приложениям (например, перенос файлов и видео по заказу предъявляют ряд требований, включая обработку ошибок, временные параметры и т. д.).

Поскольку сейчас наметилось явное устаревание технологии ATM, больше мы не будем обсуждать ее в этой книге. Тем не менее, поскольку многие системы построены с ее использованием, она продержится, вероятно, еще несколько лет. Дополнительную информацию об ATM можно почерпнуть из следующих источников: Dobrowski и Grise, 2001; Gadecki и Heckart, 1997.

Ethernet

Интернет и ATM — это глобальные сети. Однако в различных компаниях, университетах и других организациях имеется множество компьютеров, которые необходимо соединять между собой. Эта необходимость привела к возникновению и развитию технологий локальных вычислительных сетей. В этом разделе мы вкратце рассмотрим наиболее популярную технологию ЛВС под названием Ethernet.

Ее история началась на девственно чистых, ничего не знающих о высоких технологиях Гавайских островах. Под «высокими технологиями» имеется в виду хотя бы самая обычная телефонная сеть. Даже ее там тогда еще не было. Это избавляло от назойливых звонков отдыхающих там туристов, но заметно усложняло жизнь исследователю Норману Абрамсону (Norman Abramson) и его коллегам из

Гавайского университета, которые пытались соединить пользователей, разбросанных по островам, с центральным компьютером в Гонолулу. Протягивание собственных кабелей по дну Тихого океана не входило в планы ученых, поэтому необходимо было найти какое-то альтернативное решение.

И оно нашлось. За техническую основу было принято радио ближнего радиуса действия. Каждый терминал был оборудован небольшой радиостанцией, работавшей на двух частотах, одна из которых использовалась для передачи исходящего потока (к центральному компьютеру), а другая — для входящего (все от того же центрального компьютера). Если нужно было соединиться с компьютером, радиостанция передавала пакет данных по исходящему каналу. Если в этот момент никто больше не занимался пересылкой данных, то пакет успешно уходил в нужном направлении, что удостоверялось специальным подтверждением от входящего канала принимающей стороны. Если же пакет по какой-то причине не мог быть доставлен, терминал-отправитель замечал отсутствие подтверждения и пробовал снова. Поскольку заполнять данными входящий поток каждого терминала мог только один источник — центральный компьютер, то коллизий здесь возникнуть не могло. Система была названа ALOHANET и прекрасно работала в условиях низкого трафика, однако с повышением нагрузки на сеть она быстро задохнулась.

Примерно в то же время студент по имени Боб Меткаф (Bob Metcalfe) получил диплом бакалавра в Массачусетском технологическом институте (США) и встал на путь получения звания кандидата наук в Гарвардском университете. В процессе учебы он ознакомился с разработкой Абрамсона. Она настолько заинтересовала его, что он решил провести лето на Гавайских островах, работая с Абрамсоном, перед тем как начать свою запланированную деятельность в исследовательском центре Xerox. Когда же Меткаф приступил к работе в Xerox, он обнаружил, что ученые из этой компании разработали нечто, что позднее было названо персональным компьютером. Однако эти устройства никак не были связаны между собой. Используя свои знания разработок Абрамсона, Меткаф вместе со своим коллегой Дэвидом Боггсом (David Boggs) разработал и реализовал первую локальную вычислительную сеть (Metcalfe and Boggs, 1976).

Система была названа **Ethernet**. (От *luminiferous ether*, что означает «люминесцентный эфир» — вещество, в котором, как когда-то предполагалось, распространяется электромагнитное излучение. Когда в XIX веке английский физик Джеймс Максвелл (James Maxwell) открыл явление электромагнитного излучения, было не совсем понятно, в какой среде это излучение может существовать. Ученые того времени предположили, что такая среда должна быть заполнена специальным эфиром. Только известный эксперимент, проведенный Михельсоном и Морли (Michelson — Morley) в 1887 году, убедил физиков в том, что электромагнитное излучение может распространяться в вакууме.)

Тем не менее средой передачи данных в Ethernet был не вакуум, не эфир, а толстый коаксиальный кабель, длина которого могла достигать 2,5 км (с повторителями через каждые 500 м.) К системе с помощью приемопередатчиков (трансиверов), прикрученных к кабелю, можно было подключать до 256 машин. Кабель с вычислительными машинами, параллельно подключенными к нему, на-

зывается **моноканалом**. Скорость системы составляла 2,94 Мбит/с. Схема такой сети приведена на рис. 1.30. Наиболее существенное преимущество Ethernet по сравнению с ALOHANET заключается в том, что перед началом обмена данными каждый компьютер прослушивает линию, определяя ее состояние. Если по линии уже передаются данные, значит, она занята и собственную передачу следует отложить. Такой метод обеспечивает исключение наложения данных и является довольно эффективным. В ALOHANET ничего подобного не было, поскольку там была задействована радиосвязь, и терминалу, расположенному на одном далеком острове, было тяжело определить наличие передачи с терминала, расположенного на другом далеком острове. Когда же есть один кабель, проблем с этим не возникает.

Есть одна проблема, связанная с тем, что компьютеры прослушивают линию, — непонятно, как действовать, если два или более компьютера после определения свободного состояния линии захотят начать передачу? Решение было найдено в том, чтобы компьютеры прослушивали линию и во время собственной передачи и при обнаружении интерференции с чужими данными блокировали бы линию для всех отправителей. При этом тот, кто обнаружил коллизию, должен снять свои данные с линии и в течение случайного интервала времени ожидать повторной попытки. Если и после этого возникнет коллизия, время ожидания удваивается, и т. д. Это позволяет более свободно распределить во времени попытки передачи, чтобы один из компьютеров все-таки смог начать первым.

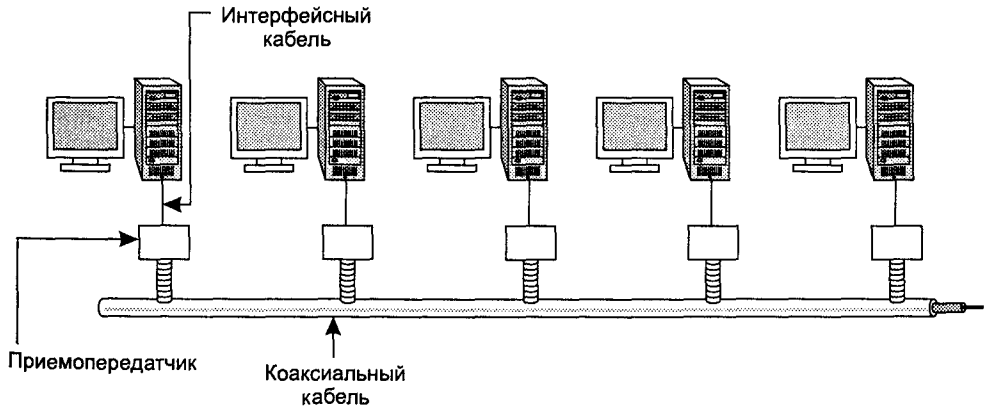


Рис. 1.30. Классическая архитектура Ethernet

Xerox Ethernet был настолько популярен, что фирмы DEC, Intel и Xerox в 1978 году совместно разработали **стандарт DIX**, описывающий Ethernet, работающий со скоростью 10 Мбит/с. С двумя небольшими изменениями DIX превратился в 1983 году в стандарт IEEE 802.3.

К сожалению, у фирмы Xerox уже был печальный опыт удачных начальных разработок (например, персональный компьютер) и неудачной маркетинговой политики — эта история описана в книге «*Fumbling the Future*» (Smith and Alexander, 1988). Когда Xerox проявила себя неспособной получить нормальный доход

от чего-либо связанного с Ethernet, кроме сопроводительных стандартов, Меткаф организовал собственную фирму 3Com и стал производить сетевые адаптеры Ethernet для ПК. Было продано более 100 миллионов экземпляров этих устройств.

Технология Ethernet продолжала развиваться, она развивается и сейчас. Уже вышли новые версии, работающие на 100 Мбит/с, 1000 Мбит/с и выше. Улучшилось качество кабелей и коммутаторов, были добавлены новые возможности. Более подробно мы обсудим Ethernet в главе 4.

Между прочим, Ethernet (IEEE 802.3) — это не единственный стандарт ЛВС. Комитетом по стандартизации зарегистрированы также технологии «маркерная шина» (802.4) и «маркерное кольцо» (802.5). Но необходимость в наличии трех более или менее совместимых стандартов вызвана причинами не столько техническими, сколько политическими. Во время стандартизации компания General Motors продвигала ЛВС с архитектурой, повторяющей Ethernet (линейный кабель). Отличие состояло только в способе определения очередности передачи. Для этого было предложено пересылать от компьютера к компьютеру специальный короткий пакет, называемый **маркером**. Начать передачу мог только тот, кто захватил маркер. Таким образом обеспечивалось отсутствие коллизий в моноканале. И все бы ничего, но General Motors заявила, что именно такая схема принципиальна для разработки автомобильной электроники. С этой позиции компанию ничто не могло сдвинуть. Несмотря на это стандарт 802.4 как-то очень быстро исчез из виду.

Аналогичным образом поступила фирма IBM, выдвинув в качестве своей новой разработки в области технологий ЛВС **маркерное кольцо**. Идея состояла в передаче маркера по кольцевому маршруту. Захватив маркер, компьютер мог осуществлять передачу и только после ее окончания передавал маркер далее. В отличие от 802.4, такая схема используется до сих пор, — правда, исключительно в вычислительных центрах IBM. Вне этих центров встретить маркерное кольцо сложно. Тем не менее, ведется разработка гигабитного маркерного кольца (802.5v), однако вряд ли эта технология сможет составить серьезную конкуренцию Ethernet. В общем, в соперничестве маркерной шины, маркерного кольца и Ethernet победила технология Ethernet, и во многом это объясняется тем, что она была первой, а все остальные разработки оказались вторичными как по времени создания, так и по уровню.

Беспроводные ЛВС: 802.11

Почти одновременно с появлением ноутбуков у людей появились мысли о том, что неплохо было бы иметь возможность, например, по пути на работу каким-нибудь волшебным образом войти в Интернет и почитать последние новости. Разумеется, многие компании стали заниматься разработкой соответствующих аппаратных решений. Вскоре был найден очень практичный метод. Он состоял в том, чтобы оборудовать ноутбук и настольный компьютер, находящийся в офисе, радиотрансиверами небольшого радиуса действия, что позволило бы им связаться между собой. И вскоре на рынке появились первые беспроводные локальные сети, созданные разными производителями.

Проблема была в том, что сети разных фирм были совершенно несовместимы между собой. Например, компьютер, оборудованный трансивером фирмы А, был не способен работать в помещении, в котором находилась базовая станция фирмы Б. Наконец было решено привести все беспроводные ЛВС к единому стандарту. Разобраться во всем многообразии существующих технологий и выработать единую концепцию было поручено институту IEEE, который уже имел опыт стандартизации обычных ЛВС. Результатом стал стандарт 802.11, который на профессиональном жаргоне получил прозвище **WiFi**. Но этот стандарт важен и заслуживает уважения, поэтому мы будем называть его как положено — 802.11.

Данный стандарт подразумевает возможность работы в двух режимах:

- с базовой станцией;
- без базовой станции.

В первом случае связь осуществляется посредством базовой станции, называемой в терминологии 802.11 **точкой доступа**. Во втором случае компьютеры общаются непосредственно друг с другом. Этот режим иногда называют **специальной сетью**. Типичным примером является случай, когда несколько людей создают беспроводную локальную сеть, объединяя в нее свои ноутбуки. При этом они обычно находятся в помещении, в котором отсутствует базовая станция. Оба режима показаны на рис. 1.31.

Самой простой задачей IEEE было дать стандарту название. Все номера с 802.1 по 802.10 были заняты, поэтому логично было присвоить ему следующий номер: 802.11. Все остальное оказалась сложнее.

В частности, необходимо было решить следующие проблемы: найти подходящую полосу частот, желательно, удовлетворяющую международным стандартам; преодолеть сильную ограниченность дальности передачи радиоволн; обеспечить защиту информации; принять во внимание ограниченный ресурс аккумуляторов ноутбуков; подумать и о медицинской стороне вопроса (стимулируют ли радиоволны развитие раковых опухолей?); осознать предпосылки компьютерной мобильности и последствия ее реализации; наконец, построить систему с достаточно высокой пропускной способностью и разумной ценой.



Рис. 1.31. Беспроводная сеть с базовой станцией (*а*); специальная сеть (*б*)

Ко времени начала процесса стандартизации (середина 90-х) Ethernet уже играл доминирующую роль среди технологий ЛВС, поэтому было решено сделать стандарт 802.11 совместимым с Ethernet поверх канального уровня. В частности, это коснулось возможности посылать IP-пакеты по беспроводной ЛВС таким же способом, как и по обычной сети Ethernet. Тем не менее, в физическом и канальном уровнях произошли некоторые изменения, которые следовало учесть в стандарте.

Во-первых, компьютер, подключенный к Ethernet, обязан прослушивать линию перед началом передачи, и может осуществить ее, только если линия свободна. Когда дело касается беспроводной сети, эта идея не проходит. Чтобы понять, почему, взгляните на рис. 1.32. Вполне возможна ситуация, когда компьютер *А* передает данные компьютеру *Б*, но радиуса действия трансивера *А* не хватает, чтобы его «услышал» компьютер *В*. Если последний захочет обменяться данными с *Б*, он, конечно, может вначале прослушать эфир, но результат окажется не соответствующим действительности. Стандарт 802.11 должен был как-то решить эту проблему.

Еще один недостаток радиоволн, с которым необходимо было бороться, заключается в том, что они замечательно отражаются от твердых поверхностей, а значит, могут быть приняты несколько раз с нескольких направлений. Такой вид интерференции породил понятие **замирания вследствие многолучевого распространения**.

Третьей проблемой явилось то, что большая часть программного обеспечения была не приспособлена к мобильности компьютера. Например, текстовые процессоры часто хранят списки доступных принтеров. Когда такой компьютер переносят с одного места на другое, этот список перестает быть актуальным.

Четвертая проблема заключается в следующем. При переносе ноутбука из зоны действия одной базовой станции в зону действия другой нужно организовать соответствующий переход в работе трансивера с базовыми станциями. Эта задача известна в мире сотовой связи, но она никогда не ставилась в Ethernet-технологии, и ее надо было как-то решить.

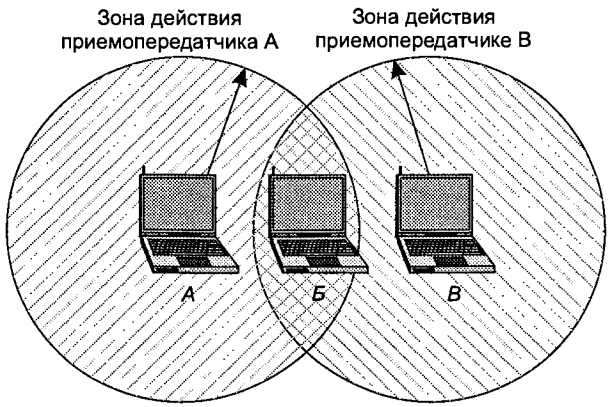


Рис. 1.32. Радиусы действия отдельных трансиверов могут не охватывать всю систему

В частности, можно представить себе сеть, состоящую из нескольких секций, в каждую из которых входит одна базовая станция, подключенная к Ethernet. Тогда в целом система выглядит как единая сеть Ethernet. Это показано на рис. 1.33. Элемент, соединяющий 802.11 с внешним миром, называется **порталом**.

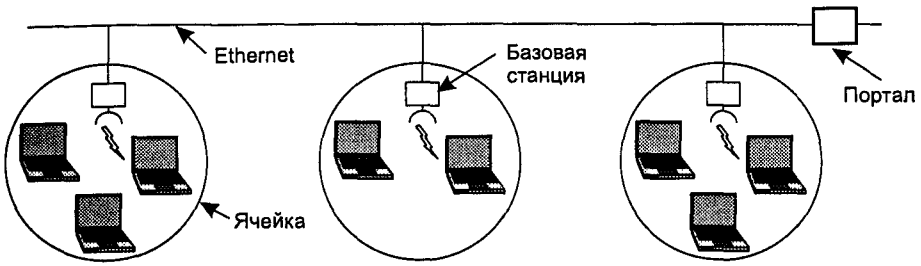


Рис. 1.33. Многосекционная сеть 802.11

Результатом работы комитета IEEE стал выпуск в 1997 году стандарта, который удовлетворял как перечисленным, так и многим другим требованиям. Беспроводная локальная сеть, которую он описывал, работала на скорости 1 Мбит/с или 2 Мбит/с. Практически сразу стали поступать жалобы пользователей на то, что сеть работала слишком медленно, поэтому была начата работа над более высокоскоростным стандартом. Разделение, произошедшее в комитете, привело к созданию в 1999 году сразу двух стандартов. IEEE 802.11a описывает сеть с расширенной полосой частот и скоростью обмена до 54 Мбит/с. 802.11b — это беспроводная сеть с такой же полосой частот, что и в базовом стандарте 802.11, но несколько иная технология модуляции позволила достигнуть скорости 11 Мбит/с. Многим это кажется важным психологическим фактором, поскольку 11 Мбит/с — это чуть быстрее, чем обычный Ethernet. Похоже, что дело идет к постепенному отмиранию изначального стандарта 802.11 со скоростью 1 Мбит/с, однако до сих пор непонятно, какое из двух расширений — 802.11a или 802.11b — выживет.

Чтобы еще больше усложнить жизнь, комитет 802 выработал еще один стандарт — 802.11g, который использует метод модуляции 802.11a и полосу частот 802.11b. Впрочем, мы еще вернемся к семейству стандартов 802.11 в главе 4.

Несомненно, 802.11 вскоре приведет к настоящей революции в компьютерном мире и в технологиях доступа в Интернет. Беспроводные сети уже устанавливаются в аэропортах, на железнодорожных станциях, в отелях, больших магазинах и университетах. Их можно найти даже в некоторых богатых кафе, где, сидя за чашечкой кофе, «золотая молодежь» может полазать по Интернету. Собственно говоря, стандарт 802.11 делает ноутбуки мобильными. Именно для этого они и были когда-то созданы.

Стандартизация сетей

В мире существует большое количество производителей сетей, каждый из которых имеет свои представления о способах реализации различных функций. Без

координации их действий наступила бы полная неразбериха, и пользователи сетей не смогли бы с ней справиться. Единственным способом борьбы с хаосом является достижение согласия по определенным вопросам на основе сетевых стандартов.

Стандарты не только обеспечивают возможность общения различных компьютеров, но также расширяют рынок для продукции, придерживающейся стандарта, что приводит к массовому выпуску совместимой друг с другом аппаратуры, очень широкомасштабной интеграции (VLSI, Very Large Scale Integration), удешевлению производства и, следовательно, к еще большему притоку потребителей на этот рынок. В следующих разделах мы рассмотрим чрезвычайно важный, но плохо изученный мир международных стандартов.

Стандарты делятся на две категории: de facto и de jure. Стандарты **de facto** (лат. «фактические») установились сами собой, без какого-либо предварительного плана. Так, например, персональные компьютеры IBM PC и их преемники стали стандартом de facto для небольших офисных компьютеров, поскольку десятки производителей решили довольно детально копировать машины фирмы IBM. Система UNIX стала стандартом de facto для операционных систем университетских компьютеров.

Стандарты **de jure** (лат. «юридические»), напротив, являются формальными, легитимными стандартами, принятыми некими авторитетными организациями по стандартизации. Международные организации по стандартизации обычно делятся на два класса: созданные на основе межправительственных договоров и добровольные организации. В области сетевых компьютерных стандартов существуют несколько организаций каждого типа, которые мы обсудим далее.

Кто есть кто в мире телекоммуникаций

Официальный статус телефонных компаний мира в разных странах различен. На одном полюсе находятся Соединенные Штаты, в которых имеется 1500 отдельных частных телефонных компаний. До разделения в 1984 году AT&T, самая большая на тот момент корпорация в мире, полностью доминировала в этом рыночном секторе. Ее услугами пользовались около 80 % абонентов Америки, что покрывало примерно половину территории страны, тогда как все остальные компании поставляли услуги остальным клиентам (в основном провинциям). После разделения AT&T продолжает предоставлять услуги по междугородной связи, теперь уже конкурируя с другими компаниями. Семь региональных телефонных компаний (RBOC, Regional Bell Operating Company), на которые была разбита AT&T, и 1500 независимых компаний предоставляют местные телефонные услуги, а также услуги сотовой связи. Благодаря постоянному процессу объединения и раскола компаний эта отрасль промышленности находится в состоянии непрерывного изменения.

Компании в США, предоставляющие общественные услуги в области связи, называются **операторами связи общего пользования** (common carriers) или просто **операторами связи**. Предоставляемые ими услуги и цены на эти услуги описываются в документе, называющемся **тарифом** (tariff), который должен быть одоб-

рен Федеральной комиссией связи США (FCC, Federal Communication Commission), если речь идет о международной связи и связи между штатами, и комиссиями по коммунальным услугам штатов — применительно к связи внутри штата.

На другом полюсе располагаются страны, в которых правительство обладает полной монополией на все средства связи, включая почту, телеграф, телефон, а часто также и радио, и телевидение. В эту категорию попадает большая часть мира. В некоторых случаях владельцем средств связи выступает национализированная компания, в других им является просто особая ветвь правительства, обычно называемая **Министерством связи** или PTT (Postal Telegraph and Telephone administration — Управление почтово-телеграфной и телефонной связи). Сейчас во всем мире наблюдается тенденция перехода от государственной монополии к либерализации и конкуренции. Большинство европейских стран уже передало целиком или частично эту отрасль в руки независимых собственников, но кое-где процесс пока еще идет слишком медленно.

При таком разнообразии поставщиков услуг, очевидно, имеется необходимость в обеспечении совместимости во всемирном масштабе. Совместимости, гарантирующей пользователям (и компьютерам) разных стран возможность связаться друг с другом. На самом деле, эта потребность существовала уже очень давно. В 1865 году представители многих европейских государств собрались, чтобы сформировать союз, который явился предшественником сегодняшнего Международного союза телекоммуникаций (ITU, International Telecommunications Union). Задачей этого союза стала стандартизация международных средств связи. В то время еще нечего было стандартизировать, кроме телеграфа, но уже тогда было ясно, что если половина стран будет использовать азбуку Морзе, а другая половина — какой-либо другой код, то возникнут проблемы. С появлением международной телефонной связи ITU занялся также разработкой стандартов в области телефонии. В 1947 году международный союз телекоммуникаций вошел в состав учреждений Организации Объединенных Наций.

Международный союз телекоммуникаций состоит из трех главных секторов:

- ✦ сектор радиосвязи (ITU-R);
- ✦ сектор стандартизации телекоммуникаций (ITU-T);
- ✦ сектор развития (ITU-D).

Сектор радиосвязи (ITU-R) занимается вопросами предоставления радиочастот конкурирующим компаниям во всем мире. Нас же в первую очередь интересует сектор стандартизации телекоммуникаций (ITU-T), рассматривающий вопросы, связанные с телефонными системами и системами передачи данных. С 1956 по 1993 год ITU-T назывался Консультативным комитетом по международной телефонной и телеграфной связи (CCITT, Comité Consultatif International Télégraphique et Téléphonique). 1 марта 1993 года CCITT был реорганизован и переименован в ITU-T, что больше отражало роль этой организации. Как CCITT, так и ITU-T выпускали рекомендации в области телефонных систем и систем передачи данных. Рекомендации CCITT встречаются еще довольно часто, как, например, CCITT X.25, хотя с 1993 года рекомендации носят эмблему ITU-T.

В ITU-T входят представители

- ✦ национальных административных структур;
- ✦ соответствующего промышленного сектора;
- ✦ организаций;
- ✦ регулятивных органов.

В ITU-T входят примерно 200 представителей различных ведомств, включая практически все организации ООН. В США отсутствует PTT, но кто-то же должен представлять государственные структуры. Эта роль выпала Государственному департаменту. Возможно, он участвует в решении вопросов взаимодействия ITU-T с другими государствами, что коррелирует с собственным направлением деятельности Госдепартамента. В состав ITU-T входит примерно 500 представителей соответствующего промышленного сектора, среди которых есть и телефонные компании (например, AT&T, Vodafone, WorldCom), и производители телекоммуникационной аппаратуры (например, Cisco, Nokia, Nortel), и производители компьютеров (например, Compaq, Sun, Toshiba), производители микросхем (например, Intel, Motorola, TI), медиа-компании (например, AOL Time Warner, CBS, Sony) и другие заинтересованные компании (например, Boeing, Samsung, Xerox). Различные некоммерческие научные организации и промышленные консорциумы (например, IFIP и IATA) также входят в состав представителей промышленного сектора. Другую группу составляют представители относительно небольших организаций, заинтересованных в конкретных направлениях стандартизации. Регулятивные органы наблюдают за развитием телекоммуникационного бизнеса. Среди них, например, Федеральная комиссия связи США.

Задачей ITU-T является разработка технических рекомендаций в области телефонии, телеграфа и интерфейсов передачи данных. Подобные рекомендации часто становятся международными стандартами, как, например, V.24 (известный также как EIA RS-232), описывающий расположение и назначение контактов разъема, используемого в большинстве асинхронных терминалов и внешних модемов.

Следует заметить, что рекомендации ITU-T являются всего лишь предложениями, которые правительство любой страны может принять или проигнорировать. (Потому что правительства подобны тринадцатилетним мальчишкам, которые терпеть не могут, когда им приказывают что-то делать. Лучше скромно порекомендовать.) На практике никто не может помешать стране принять свой собственный телефонный стандарт, отличающийся от всех других, однако тем самым эта страна сама себя отрежет от всего мира. Возможно, такой подход будет работать в Северной Корее, но во всем остальном мире это будет настоящей проблемой. Так что использование названия «рекомендации» для стандартов ITU-T можно считать выдумкой для умиротворения националистов.

Собственно работа в ITU-T осуществляется исследовательскими группами (Study Groups), состав которых часто достигает 400 человек. В настоящее время работают 14 таких групп, занимающихся самыми разными вопросами, начиная от концепции оплаты телефонных услуг и заканчивая службами мультимедиа. Чтобы сделать возможной какую-либо работу, исследовательские группы разделяются на рабочие группы (Working Parties), подразделяющиеся, в свою очередь,

на экспертные команды (Expert Teams), которые также делятся на группы. Бюрократия и здесь остается бюрократией.

Несмотря на все это ITU-T справляется с работой. Текущий выход этой организации — около 3000 рекомендаций в год, что составляет 60 000 страниц печатного текста. Многие из них широко применяются на практике. Например, популярный стандарт V-90 для модемов со скоростью 56 Кбит/с является ничем иным, как рекомендацией ITU.

По мере того, как телекоммуникации завершают начатый ими в 80-е годы переход от национальных систем к глобальным, стандарты становятся все более важным аспектом их развития, и все большее число организаций желает принять участие в их формировании. Дополнительную информацию об ITU см. Irner, 1994.

Кто есть кто в мире международных стандартов

Международные стандарты разрабатываются Международной организацией по стандартизации (International Organization for Standardization, ISO), добровольной организацией, созданной в 1946 году В нее входят национальные организации по стандартизации из 89 стран. Среди ее членов такие организации, как ANSI (США), BSI (Великобритания), AFNOR (Франция), DIN (Германия) и др.

ISO выпускает стандарты, касающиеся широкого спектра вопросов, начиная от болтов и гаек (буквально) до покраски телефонных столбов (не говоря уж о стандартах на какао-бобы (ISO 2451), рыболовные сети (ISO 1530), женское нижнее белье (ISO 4416)... Сложно придумать такую вещь, для которой не существовало бы стандарта ISO). К настоящему времени выпущено более 13 000 стандартов, включая стандарты OSI. В ISO входят почти 200 технических комитетов (Technical Committee, TC), нумеруемых последовательно, по мере их создания, каждый из которых занимается своим отдельным вопросом. Так, например, TC1 занимается болтами и гайками (стандартизацией диаметра и шага резьбы). Компьютеры и обработка информации входят в сферу деятельности TC97. Каждый технический комитет делится на подкомитеты (subcommittee, SC), которые, в свою очередь, состоят из рабочих групп (working group, WG).

Основная работа проводится в рабочих группах, в которые входит более 100 000 добровольцев по всему миру. Многие из этих «добровольцев» работают по найму в компаниях, чьи продукты должны быть стандартизованы. Другие являются государственными служащими, пытающимися сделать свой национальный стандарт международным. Ученые эксперты также принимают активное участие во многих рабочих группах.

В области телекоммуникационных стандартов ISO и ITU-T часто сотрудничают (ISO является членом ITU-T), чтобы не допустить возникновения двух несовместимых международных стандартов.

Представителем США в ISO является **Национальный институт стандартизации США (ANSI**, American National Standards Institute), который, несмотря на свое название, является частной неправительственной некоммерческой организацией. Ее членами являются производители, операторы связи и другие заинтересованные стороны. Стандарты ANSI часто принимаются ISO в качестве международных.

Для принятия стандартов Международной организацией по стандартизации ISO разработана процедура, позволяющая добиться принятия мнения, одобренного максимально возможным количеством сторон. Процесс начинается, когда одна из национальных организаций по стандартизации чувствует потребность в появлении международного стандарта в определенной области. Тогда формируется рабочая группа, которая вырабатывает **предварительный план** (Committee Draft, CD). Этот набросок передается всем остальным членам технического комитета. На критику проекта отводится шесть месяцев. Если проект получает одобрение значительного большинства, то откорректированный документ получает название **чернового международного стандарта** (DIS, Draft International Standard); после этого он опять циркулирует в различных группах, где за него голосуют и снабжают его комментариями. На основании этого подготавливается, утверждается и публикуется окончательный документ, называемый **международным** стандартом (IS, International Standard). В некоторых случаях черновым вариантам, CD или DIS приходится проходить через многократные изменения и голосования, пока они не наберут достаточного количества голосов. Этот процесс может длиться несколько лет.

Национальный институт стандартов и технологий США (NIST, National Institute of Standards and Technology) является агентством Министерства торговли США (U.S. Dept. of Commerce). Ранее он назывался Национальным бюро стандартов (National Bureau of Standards). Он выпускает стандарты, обязательные для закупок, проводимых правительством США, кроме закупок Министерства обороны, у которого имеются свои стандарты.

Одним из основных игроков на поле стандартизации является Институт инженеров по электротехнике и электронике (IEEE, Institute of Electrical and Electronics Engineers) — крупнейшая профессиональная организация в мире. Помимо выпуска ряда журналов и организации разнообразных конференций, IEEE также разрабатывает стандарты в области электротехники и электроники. Например, комитет IEEE 802 выпустил ряд ключевых стандартов в области локальных компьютерных сетей, часть из которых мы рассмотрим в этой книге. Реальная работа проводится, как правило, внутри рабочих групп, которые перечислены в табл. 1.4. Рабочие группы комитета 802 никогда не считались преуспевающими. Номер 802.x вовсе не был залогом успеха. Однако стремительный взлет популярности стандартов типа 802.3 и 802.11 изменил ситуацию.

Таблица 1.4. Рабочие группы комитета 802. Наиболее важные отмечены звездочкой (*). Те, что помечены галочкой (v), бездействуют. Отмеченные крестиком (†) самоликвидировались за ненадобностью

Номер	Тема разработок
802.1	Общее представление и архитектура ЛВС
802.2 v	Управление логическим каналом
802.3 *	Ethernet
802.4 v	Маркерная шина (одно время использовалась в промышленных сетях)

продолжение

Таблица 1.4 (*продолжение*)

Номер	Тема разработок
802.5	Маркерное кольцо (вклад фирмы IBM в технологии ЛВС)
802.6 v	Двойная двунаправленная шина (ранние региональные сети)
802.2 v	Техническая консультативная группа по широкополосным технологиям
802.8 †	Техническая консультативная группа по оптоволоконным технологиям
802.9 v	Изохронные ЛВС (для приложений реального времени)
802.10 v	Виртуальные ЛВС и защита информации
802.11 *	Беспроводные ЛВС
802.12 v	Приоритеты запросов (для AnyLAN фирмы Hewlett-Packard)
802.13	Счастливый номер. Почему-то его никто не выбрал.
802.14 v	Кабельные модемы (рабочая группа распалась: в области кабельных модемов ее опередил промышленный консорциум)
802.15 *	Персональные сети (Bluetooth)
802.16 *	Широкополосные беспроводные ЛВС
802.17	Гибкая технология пакетного кольца

Кто есть кто в мире стандартов Интернета

Всемирная сеть Интернет имеет свой механизм стандартизации, значительно отличающийся от ITU-T и ISO. В двух словах, основное отличие заключается в том, что сотрудники ITU и ISO носят деловые костюмы, тогда как стандарты Интернета разрабатывают в основном люди в джинсах (ну, кроме тех, кто работает в Сан-Диего — на них надеты шорты и футболки с коротким рукавом).

На совещания ITU-T и ISO собираются администраторы корпораций и государственные гражданские служащие, для которых стандартизация является работой. Они считают, что стандартизация — Очень Нужная Вещь, и посвящают ей свою жизнь. Для людей Интернета, напротив, анархия является делом принципа, однако когда сотни миллионов делают какое-то общее дело, иногда им все же приходится о чем-то договариваться, чтобы хоть что-то работало. Волей-неволей стандарты оказываются необходимыми.

Когда была запущена сеть ARPANET, Министерство обороны США создало неофициальный комитет для наблюдения за сетью. В 1983 году этот комитет был переименован в **Совет по деятельности Интернета** (Internet Activities Board, **IAB**). Перед советом были поставлены несколько расширенные задачи, а именно: удерживать исследователей, включенных в проекты ARPANET и Интернет, в более или менее одном направлении, что напоминало попытку выпаса стада кошек. Значение сокращения IAB было затем изменено на **Совет по архитектуре Интернета** (Internet Architecture Board).

Каждый из приблизительно десяти членов IAB возглавлял специальную комиссию по отдельному важному вопросу. Совет по архитектуре Интернета собирался несколько раз в год для обсуждения результатов работы и представления

отчета Министерству обороны и NSF, которые в то время осуществляли основное финансирование в этой области. Когда требовался какой-либо стандарт (например, новый алгоритм маршрутизации), члены совета прорабатывали этот вопрос, после чего объявляли об изменениях аспирантам, занимавшимся реализацией программного обеспечения сетей. Стандарты оформлялись в виде набора технических отчетов, называемых **RFC** (Requests for Comments). RFC доступны в Интернете для всех желающих (www.ietf.org/frc). Они пронумерованы в хронологическом порядке их создания. На сегодняшний день существует около 3000 этих документов. На многие из них мы будем ссылаться в этой книге.

К 1989 году Интернет вырос настолько, что подобный неформальный подход к его стандартам перестал работать. К тому моменту многие производители предлагали продукцию на основе протокола TCP/IP и не хотели ее менять просто потому, что десятку исследователей пришла в головы одна хорошая идея. Летом 1989 года IAB был снова реорганизован. Исследователи были переведены в **группу исследования Интернета** (Internet Research Task Force, IRTF), подконтрольную IAB, и в **группу проектирования Интернета** (Internet Engineering Task Force, IETF). В совете IAB появились люди, представляющие более широкий спектр организаций, чем исследовательское сообщество. Вначале это была группа, в которой члены работали в течение двух лет, после чего сами назначали своих преемников. Затем было создано **Общество Интернета** (Internet Society), в которое вошли люди, заинтересованные в Интернете. Таким образом, Интернет-сообщество в каком-то смысле сравнимо с Ассоциацией по вычислительной технике (ACM, Association for Computing Machinery) или IEEE. Оно управляется избираемыми доверенными лицами, которые утверждают состав IAB.

Идея этого разделения заключалась в том, чтобы сосредоточить IRTF на долгосрочных исследованиях, а IETF — на краткосрочных инженерных вопросах. Проблемная группа IETF была разделена на рабочие группы, каждая из которых решала свою задачу. Первое время председатели рабочих групп встречались друг с другом в составе руководящего комитета для координации совместных исследовательских усилий. Рабочие группы занимались такими вопросами, как новые приложения, информация для пользователей, OSI-интеграция, маршрутизация и адресация, безопасность, управление сетью и стандарты. В конце концов было сформировано так много рабочих групп (более 70), что их объединили по областям, после чего в руководящем комитете стали собираться председатели областей.

Кроме того, был принят более формальный процесс стандартизации по аналогии с процедурой, принятой в ISO. Чтобы стать **предлагаемым стандартом**, основная идея должна быть полностью изложена в RFC и должна представлять достаточный интерес, гарантирующий ее рассмотрение. Затем, чтобы стать **проектом стандарта**, должна быть создана работающая реализация, которую нужно тщательно протестировать минимум двумя независимыми сайтами в течение 4 месяцев. Если IAB уверен, что идея здравая и программное обеспечение работает, он может объявить RFC стандартом Интернета. Некоторые стандарты Интернета стали стандартами Министерства обороны США (MIL-STD), что сделало их обязательными к применению поставщиками министерства. Дэвид Кларк (David

Clark) как-то высказал замечание, ставшее ныне популярным, о стандартизации Интернета, состоящей из «грубого консенсуса и работающей программы».

Единицы измерения

Во избежание путаницы необходимо предварить дальнейшие рассуждения замечанием по поводу единиц измерения. В computer science традиционной английской системе обычно предпочитают десятичную систему мер. Основные префиксы, используемые при этом, приведены в табл. 1.5. Обычно они сокращаются по первым буквам их названий, причем, если префикс имеет вес, больший 1, то он пишется с заглавной буквы (Кбайт, Мбайт и т. д.). Единственное исключение исторически составляет сокращение Кбит/с. Таким образом, линия, работающая на скорости 1 Мбит/с, передает 10^6 бит в секунду, а таймер на 100 пс изменяет свое состояние каждую 10^{-10}-ю долю секунды. Поскольку «милли» и «микро» начинаются с одной и той же буквы, то принято обозначать «милли» буквой «м», а «микро» — буквами «мк» или греческой буквой «μ».

Таблица 1.5. Основные префиксы метрической системы

Степень	В явном виде	Префикс	Степень	В явном виде	Префикс
10^{-3}	0,001	Милли	10^{3}	1000	Кило
10^{-6}	0,000001	Микро	10^{6}	1 000 000	Мега
10^{-9}	0,000000001	Нано	10^{9}	1 000 000 000	Гига
10^{-12}	0,000000000001	Пико	10^{12}	1 000 000 000 000	Тера
10^{-15}	0,000000000000001	Фемто	10^{15}	1 000 000 000 000 000	Пета
10^{-18}	0,000000000000000001	Атто	10^{18}	1 000 000 000 000 000 000	Экза
10^{-21}	0,000000000000000000001	Цепто	10^{21}	1 000 000 000 000 000 000 000	Цетта
10^{-24}	0,000000000000000000000001	Йокто	10^{24}	1 000 000 000 000 000 000 000 000	Йотта

Также необходимо отметить, что единицы измерения, использующиеся для обозначения объемов памяти, емкости дисков, размеров файлов и баз данных, несколько отличаются от принятых в других областях. Например, «кило» означает не 1000 (10^3), а 1024 (2^{10}), что соответствует общей двоичной концепции computer science. Размеры памяти всегда представляют собой степени двойки. Так, в 1 Кбайте содержится 1024 байт, а не 1000 байт. Аналогично, в 1 Мбайте содержится 2^{20}, то есть 1 048 576 байт, а в 1 Гбайте, соответственно, — 2^{30} (1 073 741 824) байт. База данных на 1 Тбайт содержит 2^{40} (1 099 511 627 776) байт. Тем не менее, линия при скорости 1 Кбит/с передает 1000 бит/с, а ЛВС, работающая со скоростью 10 Мбит/с, может передавать 10 000 000 бит/с — скорости не измеряются степенями двойки. К сожалению, многие путают эти две системы счисления, особенно когда дело касается емкости дисков. Чтобы избежать двусмысленности, еще раз повторюсь: по крайней мере, в нашей книге сим-

волы Кбайт, Мбайт и Гбайт будут означать 2^{10}, 2^{20} и 2^{30} байт соответственно, а «биты» так и будут «битами». Поэтому символы Кбит/с, Мбит/с и Гбит/с будут означать соответственно 10^3, 10^6 и 10^9 бит в секунду.

Краткое содержание следующих глав

В этой книге обсуждаются как теоретические, так и практические вопросы построения компьютерных сетей. Большая часть глав начинается с обсуждения основных принципов, за которыми следует ряд примеров, иллюстрирующих эти принципы. На протяжении всей книги в качестве примеров используются две действующие сети — Интернет и беспроводные сети. Эти сети очень важны и актуальны, а кроме того, они очень разные. Там, где это важно, будут даны и другие примеры.

Структура книги соответствует гибридной модели, изображенной на рис. 1.20. В главе 2 мы рассмотрим иерархию протоколов, начиная с самого нижнего из них. В ней мы изучим основные положения в области обмена данными. Кроме того, в главе 2 обсуждаются проводная и беспроводная передача, а также спутниковые телекоммуникационные системы. Этот материал относится к физическому уровню, хотя мы будем обсуждать не столько аппаратные, сколько архитектурные аспекты. Также будут обсуждаться несколько примеров физического уровня, такие как коммутируемая телефонная сеть общего пользования, мобильная телефонная сеть, а также сеть кабельного телевидения.

В главе 3 рассматривается канальный уровень и его протоколы. Каждый последующий пример будет сложнее предыдущего. Далее проводится анализ этих протоколов и обсуждаются некоторые важные действующие протоколы, включая HDLC (используемый в сетях с низкими и средними скоростями) и PPP (используется в Интернете).

В главе 4 рассматривается подуровень доступа к носителям, являющийся частью канального уровня. В основном, он определяет следующего, кто может использовать сеть, состоящую из одного совместно используемого канала (моноканала). Такая архитектура применяется в большинстве локальных сетей, а также в некоторых спутниковых сетях. Приводится ряд примеров из области обычных ЛВС, беспроводных ЛВС (в частности, Ethernet), беспроводных региональных сетей, Bluetooth и спутниковых сетей. Также обсуждаются мосты и коммутаторы, используемые для объединения нескольких локальных сетей.

В главе 5 описывается сетевой уровень, в частности маршрутизация. Обсуждаются как статические, так и динамические алгоритмы маршрутизации. Даже очень хороший алгоритм рассчитан на сеть с определенным уровнем загрузки. При превышении этого уровня могут возникать заторы, их предотвращение и борьба с ними также обсуждается в этой главе. Однако более общим вопросом, чем предотвращение заторов, является обеспечение необходимого уровня обслуживания. Мы обсудим и это. Кроме того, затрагивается ряд проблем, возникающих при связи разнородных сетей. Особое внимание уделяется сетевому уровню применительно к Интернету.

Глава 6 посвящена транспортным уровням. Основное внимание уделяется протоколам, ориентированным на соединение, поскольку в них нуждаются многие приложения. Детально обсуждаются пример транспортной службы и ее реализация. Приводится реально работающий код для этого несложного примера, который дает представление о том, как программно реализуется данный уровень. Подробно описываются транспортные протоколы Интернета — TCP и UDP. Обсуждаются также вопросы их эффективности. Также в главе 6 будут упомянуты транспортные уровни беспроводных сетей.

В главе 7 описывается прикладной уровень, его протоколы и приложения. Первое, к чему мы обратимся, будет служба DNS — интернет-аналог телефонной книги. Затем речь пойдет об электронной почте и о протоколах, которые в ней используются. Много места уделено веб-технологиям. В разделах, посвященных им, подробно обсуждается статическое и динамическое содержимое страниц, рассказывается о том, что происходит на стороне клиента и сервера, рассматриваются протоколы, производительность, беспроводной Веб и др. Наконец, будут раскрыты и вопросы мультимедиа в Сети, включая потоковый звук, интернет-радио и видео.

Глава 8 посвящена вопросам защиты информации. В этой теме есть аспекты, касающиеся всех уровней, именно поэтому данная глава завершает книгу. Она начинается с объяснения основ криптографии. После этого приводятся примеры применения криптографии для налаживания безопасных соединений, защиты электронной почты и веб-страниц. Заканчивается глава обсуждением некоторых вопросов, в которых защита информации обеспечивает некоторые права человека. Речь идет о свободе слова, цензуре и других насущных социальных проблемах.

Глава 9 содержит аннотированный список предлагаемой литературы, организованный по главам. Его цель — помочь читателям, желающим продолжить изучение сетей. В главе также содержится алфавитный список литературы, упоминаемой в тексте книги.

Сайт автора в издательстве Prentice Hall (http://www.prenhall/tanenbaum) содержит страничку со ссылками на различные самоучители, ответы на часто задаваемые вопросы, а также ссылки на сайты компаний и промышленных консорциумов, профессиональных организаций, комитетов по стандартизациии, технологиям, документации и т. д.

Резюме

Компьютерные сети могут использоваться для оказания различных услуг как компаниям, так и частным лицам. В компаниях сети персональных компьютеров часто обеспечивают доступ к корпоративной информации. Обычно они строятся в соответствии с моделью «клиент-сервер», причем клиентские станции на рабочих местах в офисе связаны с мощными серверами, расположенными в специальном помещении. Частным лицам сеть предоставляет доступ к широкому спектру информационных и развлекательных ресурсов. Многие пользователи для входа

в Интернет используют модем, с помощью которого они соединяются со своим провайдером, хотя все больше людей устанавливает дома постоянное подключение. Новой и очень перспективной областью является технология беспроводных сетей, применяемая для таких целей как мобильный доступ к электронной почте и мобильная коммерция.

Грубо говоря, все сети могут быть разделены на локальные, региональные, глобальные и объединенные, каждая со своими характеристиками, технологиями, скоростями и областями применения. Локальные сети охватывают здание и работают с очень высокой скоростью. Региональные охватывают город — примером этого могут служить сети кабельного телевидения, используемые в последнее время для доступа в Интернет. Глобальные сети охватывают страну или континент. Локальные и региональные сети являются некоммутируемыми (то есть они не имеют маршрутизаторов), глобальные же сети являются коммутируемыми. Беспроводные сети становятся очень популярными, особенно беспроводные ЛВС. Сети можно соединять между собой, получая объединенные сети.

Сетевое программное обеспечение состоит из протоколов, или правил, по которым процессы обмениваются информацией. Протоколы могут быть ориентированными на соединение и не требующими соединения. Большая часть сетей поддерживает иерархию протоколов, в которой каждый уровень предоставляет услуги вышестоящему уровню, не раскрывая ему подробностей своей работы. Стек протоколов обычно базируется на модели OSI или модели TCP/IP. В обеих моделях имеется сетевой, транспортный и прикладной уровни, но они различаются в остальных уровнях. Вопросы разработки включают в себя уплотнение каналов, управление передачей, обнаружение ошибок и т. д. В книге уделяется много внимания протоколам и их проектированию.

Итак, сети предоставляют пользователям определенные услуги, реализуемые на основе установления соединений или без соединений. Есть сети, которые на одних уровнях используют первый из этих принципов, на других (более низких) — второй.

К хорошо известным сетям относятся Интернет, ATM, Ethernet и беспроводные ЛВС стандарта 802.11. Сеть Интернет выросла из ARPANET, к которой был добавлен ряд других сетей. Таким образом, Интернет является объединенной сетью, Сетью сетей, число которых сегодня измеряется тысячами. Интернет характеризуется, прежде всего, использованием стека протоколов TCP/IP. ATM широко используется в телефонной системе при передаче данных на большие расстояния. Ethernet — это наиболее популярная технология ЛВС, встречается в крупнейших компаниях и университетах. Наконец, беспроводные ЛВС с удивительно высокими скоростями (до 54 Мбит/с) тоже постепенно входят в нашу жизнь.

Для того чтобы миллионы компьютеров могли общаться друг с другом, нужны как аппаратные, так и программные стандарты. Разрабатываются они такими организациями, как ITU-T, ISO, IEEE и IAB. Каждая из них работает в своей области, и все вместе они реализуют процесс стандартизации компьютерных сетей.

Вопросы

1. Представьте, что вы научили свою собаку, сенбернара Берни, приносить вам коробку с тремя 8-миллиметровыми магнитными лентами вместо бутылки бренди. (Потому что с некоторых пор вы стали рассматривать заканчивающееся место на жестком диске как трагедию.) На каждой ленте помещается 7 Гбайт информации. Собака обучена бежать к вам, где бы вы ни находились, со скоростью 18 км/ч. В каком диапазоне расстояний скорость передачи данных собакой будет выше, чем у линии, чья фактическая скорость работы составляет 150 Мбит данных в секунду?
2. Альтернативой локальной сети является большая система разделения времени с терминалом для каждого пользователя. Приведите два преимущества клиент-серверной системы, использующей локальную сеть.
3. На производительность системы «клиент-сервер» влияют два параметра сети: пропускная способность (сколько бит в секунду она может передавать) и время ожидания (сколько секунд требуется на доставку первого бита от клиента до сервера). Приведите пример: а) сети с высокой пропускной способностью и большим временем ожидания; б) сети с низкой пропускной способностью и малым временем ожидания.
4. Какие еще характеристики, кроме пропускной способности и времени ожидания, нужно оптимизировать для получения высокого качества обслуживания в сетях цифровой передачи речи?
5. Одним из факторов, влияющих на время ожидания в сетях с коммутацией пакетов и промежуточным хранением, является задержка при сохранении и переадресации пакета коммутатором. Если время коммутации составляет 10 мкс, будет ли это основной задержкой в работе клиент-серверной системы, в которой клиент находится в Нью-Йорке, а сервер — в Калифорнии? Скорость распространения сигнала в медной линии принять равной 2/3 скорости света в вакууме.
6. Система «клиент-сервер» использует спутниковую сеть. Орбита вращения спутника удалена от поверхности Земли на 40 000 км. Какова будет минимально возможная задержка при ожидании ответа на запрос в такой системе?
7. В будущем, когда у всех дома будет терминал, подключенный к компьютерной сети, станут возможными мгновенные референдумы по важным законодательным вопросам. В конце концов существующие законодательные органы могут быть распущены, что позволит народу выражать свою волю напрямую. Позитивные аспекты такой прямой демократии очевидны. Обсудите некоторые негативные аспекты.
8. Пять маршрутизаторов необходимо соединить в подсеть с двухточечным соединением. Каждые два маршрутизатора разработчики могут соединить высокоскоростной, среднескоростной, низкоскоростной линией или никак не соединять. Предположим, компьютеру требуется 100 мс для моделирования

и анализа каждой топологии. Сколько компьютерного времени понадобится для выбора варианта, лучше всего соответствующего ожидаемой нагрузке?

9. Группа из $2^n - 1$ маршрутизаторов соединены в централизованное бинарное дерево с маршрутизатором в каждом узле. Маршрутизатор i общается с маршрутизатором j, посылая сообщения корню дерева. Затем корень дерева посылает это сообщение маршрутизатору j. Выведите приблизительную формулу числа прыжков сообщения для большого n, предполагая, что все пары маршрутизаторов одинаковы.
10. Отрицательной чертой широковещательной подсети является потеря мощности вследствие попытки одновременного доступа к каналу нескольких хостов. В качестве простейшего примера, предположим, что время делится на равные интервалы, в которые каждый из n хостов пытается использовать канал с вероятностью p. Какой процент интервалов будет потерян из-за конфликтов?
11. Назовите две причины использования многоуровневых протоколов.
12. Президент корпорации Specialty Paint Corp. решает объединить усилия с местной пивоваренной фабрикой для производства невидимой пивной банки (в качестве средства борьбы с мусором). Президент просит свой юридический отдел рассмотреть эту идею, а те, в свою очередь, обращаются за помощью в технический отдел. В результате начальник технического отдела звонит начальнику технического отдела пивоваренного завода, чтобы обсудить технические аспекты проекта. После этого оба инженера докладывают своим юридическим отделам, которые затем обсуждают друг с другом по телефону юридические вопросы. Наконец, два президента корпораций обсуждают финансовую сторону дела. Является ли данный пример иллюстрацией многоуровневого протокола в духе модели OSI?
13. Какова принципиальная разница между ориентированным на соединение и не требующим соединения обменом информации?
14. Представим себе две сети, предоставляющие надежные ориентированные на соединение службы. Одна из сетей обеспечивает надежный поток байтов, а другая — надежный поток сообщений. Идентичны ли эти сети? Если да, то почему проводится различие? Если нет, объясните на примере, чем они различаются.
15. Что означает термин «согласование» (negotiation) в контексте обсуждения сетевых протоколов? Приведите пример.
16. На рис. 1.15 изображен некоторый сервис. Подразумевается ли наличие других сервисов на этом рисунке? Если да, то где? Если нет, то почему?
17. В некоторых сетях уровень передачи данных обрабатывает ошибки передачи, требуя повторной передачи поврежденных кадров. Если вероятность повреждения кадра равна p, каким будет среднее число попыток, необходимых для передачи кадра, при условии, что подтверждения никогда не теряются?
18. Какой из уровней OSI выполняет следующее:
    1) разбивает передаваемый поток битов на кадры;
    2) определяет, какой из маршрутов подсети использовать?

Если данные, измененные на уровне передачи данных, называются кадрами, а измененные на сетевом уровне — пакетами, означает ли это, что в кадре содержатся пакеты или что в пакете содержатся кадры? Ответ поясните.

Система обладает n-уровневой иерархией протоколов. Приложения обмениваются сообщениями длиной M байт. На каждом из уровней добавляется заголовок из h байт. Какой процент пропускной способности занят заголовками?

Приведите две общие черты эталонных моделей OSI и TCP/IP. Приведите два различия этих моделей.

В чем основное различие между протоколами TCP и UDP?

Подсеть на рис. 1.21, *б* проектировалась таким образом, чтобы выстоять во время ядерной войны. Сколько бомб потребуется, чтобы разбить сеть на две изолированные части, если одна бомба разрушает узел со всеми линиями, подходящими к нему.

Интернет удваивается в размерах приблизительно каждые 18 месяцев. Точное число хостов неизвестно, но один аналитик в 2001 году назвал цифру в 100 млн. хостов. Сколько будет хостов в Интернете в 2010 году? Вы сами верите в это? Поясните свою точку зрения.

При передаче файла между двумя компьютерами возможны (как минимум) две стратегии подтверждений. В первом случае файл может быть разбит на отдельные пакеты, получение которых подтверждается получателем индивидуально, но получение всего файла не подтверждается. Во втором случае получение каждого пакета не подтверждается, а подтверждается получение всего файла. Обсудите оба варианта.

Почему в ATM используются короткие ячейки фиксированного размера?

Какой длины (в метрах) был один бит в соответствии со стандартом 802.3? Для вычислений примите скорость работы равной 10 Мбит/с, а скорость распространения сигнала равной 2/3 скорости света в вакууме.

Имеется несжатое изображение размером 1024×768 пикселов, 3 байта/пиксел. Сколько времени потребуется на его передачу с помощью модема, работающего со скоростью 56 Кбит/с? С помощью кабельного модема, работающего на 1 Мбит/с? По Ethernet со скоростью передачи 10 Мбит/с? По Ethernet со скоростью 100 Мбит/с?

Ethernet и беспроводные сети имеют много общего, но есть и различия. Одним из свойств Ethernet является возможность передачи только одного кадра в каждый момент времени. Унаследовал ли стандарт 802.11 это свойство?

Беспроводные сети легко устанавливаются, что делает их относительно недорогими, потому что стоимость установки обычной сети часто превышает стоимость оборудования. Тем не менее, у них есть некоторые недостатки. Назовите два из них.

Назовите два преимущества и два недостатка наличия международных стандартов для сетевых протоколов.

Когда у системы имеются постоянная и съемная части, как у проигрывателя компакт-дисков и компакт-диска, важно, чтобы система была стандартизиро-

вана, чтобы различные компании могли выпускать как постоянные, так и съемные части, стыкующиеся друг с другом. Приведите три примера международных стандартов вне компьютерной области. Теперь приведите три примера отсутствия международных стандартов вне компьютерной области.

33. Перечислите, какие действия вы производите ежедневно при помощи компьютерных сетей. Как изменится ваша жизнь, если сети вдруг перестанут существовать?

34. Узнайте, какие сети установлены в вашем учебном заведении или на работе. Опишите их типы, топологии, методы коммутации.

35. Программа *ping* позволяет отправлять тестовый пакет по указанному адресу и исследовать время его прохождения в одну и в другую сторону. Попробуйте воспользоваться этой программой, чтобы выяснить, сколько времени следуют данные по различным известным адресам. Используя полученные данные, постройте график зависимости времени прохождения пакета от расстояния. Лучше всего использовать для экспериментов серверы университетов, поскольку их расположение очень хорошо известно. Например, berkley.edu находится в Беркли, штат Калифорния, mit.edu — в Кембридже, Массачусетс, vu.nl — в Амстердаме, Голландия, www.usyd.edu.au — в Сиднее, Австралия и www.uct.ac.za — в Кейптауне, Южная Африка.

36. Посетите сайт IETF (www.ietf.org) и изучите, чем занимается эта организация. Возьмите в качестве примера любой понравившийся вам проект и напишите краткий отчет о том, что он собой представляет.

37. Стандартизация — это очень важный аспект развития сети. ITU и ISO — это ведущие организации, занимающиеся стандартизацией. Посетите их веб-сайты (www.itu.org и www.iso.org) и изучите, чем они занимаются. Напишите краткий отчет об областях стандартизации, в которых преуспели эти организации.

38. Интернет состоит из огромного числа сетей. Их взаимное расположение определяет топологию Интернета. Очень много информации на тему топологии Интернета можно найти на различных веб-сайтах. С помощью поисковых программ найдите соответствующую информацию и напишите краткий отчет по итогам исследования.

Глава 2
Физический уровень

- Теоретические основы передачи данных
- Управляемые носители информации
- Беспроводная связь
- Спутники связи
- Коммутируемая телефонная сеть общего пользования
- Мобильная телефонная система
- Кабельное телевидение
- Резюме
- Вопросы

В этой главе мы рассмотрим нижний уровень иерархии компьютерных сетей. Он определяет механические, электрические и временные характеристики сетей. Начнем с теоретического анализа передачи данных, чтобы с удивлением обнаружить, что природа накладывает определенные ограничения на то, что и как можно передавать с помощью физического носителя.

Затем мы обсудим три типа сред передачи — управляемую (медный провод и оптоволокно), радиоэфир (наземная радиосвязь) и радиоэфир, связанный со спутниковыми системами. Мы изучим основы ключевых технологий передачи данных, применяемых в современных сетях.

Оставшаяся часть главы посвящена трем примерам систем связи, которые используются на практике в глобальных сетях. Мы начнем с телефонной системы (стационарной); второй пример — мобильная телефонная система; третий — кабельное телевидение. Все они на уровне магистралей используют оптоволокно, но организованы по-разному, а по мере приближения к конечному пользователю в них применяются все более отличающиеся друг от друга технологии.

Теоретические основы передачи данных

Информация может передаваться по проводам за счет изменения какой-либо физической величины, например напряжения или силы тока. Представив значение напряжения или силы тока в виде однозначной функции времени $f(t)$, мы сможем смоделировать поведение сигнала и подвергнуть его математическому анализу. Этому анализу и посвящены следующие разделы.

Ряды Фурье

В начале XIX столетия французский математик Жан-Батист Фурье (Jean-Baptiste Fourier) доказал, что любая периодическая функция $g(t)$ с периодом T может быть разложена в ряд (возможно, бесконечный), состоящий из сумм синусов и косинусов:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft); \qquad (2.1),$$

где $f = 1/T$ — основная частота (гармоника), a_n и b_n — амплитуды синусов и косинусов n-й **гармоники**, а c — константа. Подобное разложение называется **рядом Фурье**. Разложенная в ряд Фурье функция может быть восстановлена по элементам этого ряда, то есть если период T и амплитуды гармоник известны, то исходная функция может быть восстановлена с помощью суммы ряда (2.1).

Информационный сигнал, имеющий конечную длительность (все информационные сигналы имеют конечную длительность), может быть разложен в ряд Фурье, если представить, что весь сигнал бесконечно повторяется снова и снова (то есть интервал от T до $2T$ полностью повторяет интервал от 0 до T, и т. д.).

Амплитуды a_n могут быть вычислены для любой заданной функции $g(t)$. Для этого нужно умножить левую и правую стороны уравнения (2.1) на $\sin(2\pi kft)$, а затем проинтегрировать от 0 до T. Поскольку:

$$\int_0^T \sin(2\pi kft)\sin(2\pi nft)dt = \begin{cases} 0 \text{ при } k \neq n, \\ T/2 \text{ при } k = n, \end{cases}$$

остается только один член ряда: a_n. Ряд b_n исчезает полностью. Аналогично, умножая уравнение (2.1) на $\cos(2\pi kft)$ и интегрируя по времени от 0 до T, мы можем вычислить значения b_n. Если проинтегрировать обе части уравнения, не изменяя его, то можно получить значение константы c. Результаты этих действий будут следующими:

$$a_n = \frac{2}{T}\int_0^T g(t)\sin(2\pi nft)\,dt;\ b_n = \frac{2}{T}\int_0^T g(t)\cos(2\pi nft)\,dt;\ c = \frac{2}{T}\int_0^T g(t)\,dt.$$

Сигналы с ограниченным спектром

Чтобы понять, какое отношение все вышеизложенное имеет к передаче данных, рассмотрим конкретный пример — передачу двоичного кода ASCII символа «b». Для этого потребуется 8 бит (то есть 1 байт). Задача — передать следующую после-

тельность бит: 01100010. На рис. 2.1, *а* слева изображена зависимость выход-
напряжения от времени на передающем компьютере. В результате анализа
ье для данного сигнала получаем следующие значения коэффициентов:

$$a_n = \frac{1}{\pi n}\left[\cos(\pi n/4) - \cos(3\pi n/4) + \cos(6\pi n/4) - \cos(7\pi n/4)\right];$$

$$b_n = \frac{1}{\pi n}\left[\sin(3\pi n/4) - \sin(\pi n/4) + \sin(7\pi n/4) - \sin(6\pi n/4)\right];$$

$$c = 3/4.$$

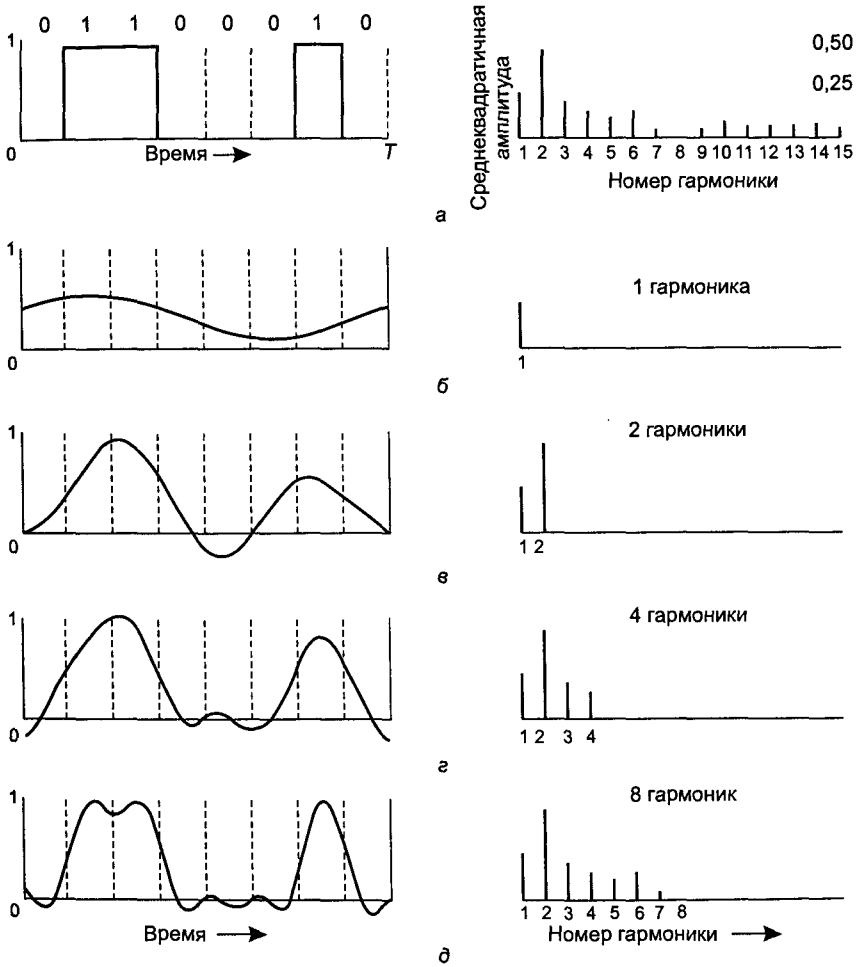


Рис. 2.1. Двоичный сигнал и его среднеквадратичные гармоники Фурье (*а*); последовательные приближения к оригинальному сигналу (*б*–*д*)

Среднеквадратичные амплитуды $\sqrt{a_n^2 + b_n^2}$ для нескольких первых гармоник
казаны на рис. 2.1, *а* справа. Эти значения представляют интерес, поскольку

их квадраты пропорциональны энергии, передаваемой на соответствующей частоте.

Ни один канал связи не может передавать сигналы без потери мощности. Если бы все гармоники ряда Фурье уменьшались при передаче в равной степени, то сигнал уменьшался бы по амплитуде, но не искажался (то есть у него была бы та же самая замечательная прямоугольная форма, как на рис. 2.1, *а*). К сожалению, все каналы связи уменьшают гармоники ряда Фурье в разной степени, тем самым искажая передаваемый сигнал. Как правило, амплитуды передаются без уменьшения в частотном диапазоне от 0 до некоей частоты f_c (измеряемой в периодах в секунду или герцах (Гц)), при этом высокочастотная составляющая сигнала (выше частоты f_c, называемой *частотой среза*) заметно ослабляется. Этот диапазон частот называется **полосой пропускания**. На практике срез вовсе не является таким резким, поэтому обычно в упомянутую полосу пропускания включают те частоты, которые передаются с потерей мощности, не превышающей 50 %.

Полоса пропускания является физической характеристикой среды передачи данных и зависит обычно от конструкции, толщины и длины носителя. Иногда для намеренного уменьшения полосы пропускания, доступной абонентам, в линию включается специальное устройство — *фильтр*. Например, кабель, используемый в телефонии при небольших расстояниях, имеет полосу пропускания, равную 1 МГц, однако телефонные компании с помощью частотных фильтров урезают ее, предоставляя пользователям лишь 3100 Гц. Такой полосы, впрочем, вполне хватает для отчетливой передачи речи, зато за счет уменьшения расходуемых каждым абонентом ресурсов повышается общая эффективность системы.

Теперь посмотрим, как будет выглядеть сигнал, изображенный на рис. 2.1, *а*, если полоса пропускания канала будет такой, что через него будут проходить только самые низкие частоты (то есть функция $g(t)$ будет аппроксимирована лишь несколькими первыми членами рядов уравнения (2.1)). На рис. 2.1, *б* показан сигнал на выходе канала, пропускающего лишь первую (основную, f) гармонику сигнала. Аналогично, рис. 2.1, *в–д показывают спектры и восстановленные сигналы для каналов с более широкой полосой пропускания.*

При заданной скорости передачи в битах, равной b бит/с, время, требуемое для передачи, скажем, 8 бит, будет равно $8/b$ секунд. Таким образом, частота первой гармоники равна $b/8$ Гц. Обычная телефонная линия, часто называемая **речевым каналом**, имеет искусственно созданную частоту среза около 3000 Гц. Это ограничение означает, что номер самой высокой гармоники, прошедшей сквозь телефонный канал, примерно (срез не очень крутой) равен $3000/(b/8)$ или $24\,000/b$.

Для некоторых скоростей передачи данных эти значения показаны в табл. 2.1. Из приведенных данных ясно, что попытка передать по речевому каналу данные на скорости 9600 бит/с превратит сигнал, показанный на рис. 2.1, *а*, в нечто подобное рис. 2.1, *в*, что сделает прием исходного потока битов с приемлемым качеством практически невозможным. Очевидно, что у сигналов, передаваемых со скоростью 38 400 бит/с и выше, нет никаких шансов пройти через речевой канал, даже при полном отсутствии помех на линии. Другими словами, ограничение по-

лосы пропускания частот канала ограничивает его пропускную способность для передачи *двоичных* данных, даже для идеальных каналов. Однако схемы, использующие несколько уровней напряжений, существуют и позволяют достичь более высоких скоростей передачи данных. Мы обсудим зто далее в этой главе.

Таблица 2.1. Соотношение между скоростью передачи данных и числом гармоник

Скорость, бит/с	T, мс	1-я гармоника, Гц	Количество пропускаемых гармсник
300	26,67	37,5	80
600	13,33	75	40
1200	6,67	150	20
2400	3,33	300	10
4800	1,67	600	5
9600	0,83	1200	2
19 200	0,42	2400	1
38 400	0,21	4800	0

Максимальная скорость передачи данных через канал

В 1924 году американский ученый Х. Найквист (H. Nyquist) из компании AT&T пришел к выводу, что существует некая предельная скорость передачи даже для идеальных каналов. Он вывел уравнение, позволяющее найти максимальную скорость передачи данных в бесшумном канале с ограниченной полосой пропускания частот. В 1948 году Клод Шеннон (Claude Shannon) продолжил работу Найквиста и расширил ее для случая канала со случайным (то есть термодинамическим) шумом. Мы кратко рассмотрим результаты работы Найквиста и Шеннона, ставшие сегодня классическими.

Найквист доказал, что если произвольный сигнал прошел через низкочастотный фильтр с полосой пропускания H, то такой отфильтрованный сигнал может быть полностью восстановлен по дискретным значениям этого сигнала, измеренным с частотой $2H$ в секунду. Производить измерения сигнала чаще, чем $2H$ в секунду, нет смысла, так как более высокочастотные компоненты сигнала были отфильтрованы. Если сигнал состоит из V дискретных уровней, то уравнение Найквиста будет выглядеть так:

$$\text{максимальная скорость передачи данных} = 2H \log_2 V, \text{ бит/с}$$

Так, например, бесшумный канал с частотой пропускания в 3 кГц не может передавать двоичные (то есть двухуровневые) сигналы на скорости, превосходящей 6000 Кбит/с.

Итак, мы рассмотрели случай бесшумных каналов. При наличии в канале случайного шума ситуация резко ухудшается. Уровень термодинамического шума в канале измеряется отношением мощности сигнала к мощности шума и называется **отношением сигнал/шум**. Если обозначить мощность сигнала S, а мощность

шума — N, то отношение сигнал/шум будет равно S/N. Обычно сама величина отношения не употребляется. Вместо нее используется ее десятичный логарифм, умноженный на 10: 10 lg S/N. Такая единица называется **децибелом** (decibel, dB, дБ). Таким образом, если отношение сигнал/шум 10, это соответствует 10 дБ, отношение 100 равно 20 дБ, отношение 1000 равно 30 дБ и т. д. Производители стереоусилителей часто указывают полосу частот (частотный диапазон), в котором их аппаратура имеет линейную амплитудно-частотную характеристику в пределах 3 дБ. Отклонение в 3 дБ соответствует ослаблению сигнала примерно в два раза (потому что $\log_{10}3 \approx 0{,}5$).

Главным результатом, который получил Шеннон, было утверждение о том, что максимальная скорость передачи данных в канале с полосой частот H Гц и отношением сигнал/шум, равным S/N, можно вычислить по формуле

$$\text{максимальная скорость передачи данных} = H \log_2(1+S/N).$$

Например, канал с частотной полосой пропускания в 3000 Гц и отношением мощностей сигнала и термального шума в 30 дБ (обычные параметры для аналоговой части телефонной системы) никогда не сможет передавать более 30 000 бит/с, независимо от способа модуляции сигнала, то есть количества используемых уровней сигнала, частоты дискретизации и т. д. Результат, полученный Шенноном и подкрепленный постулатами теории информации, применим к любому каналу с гауссовским (термальным) шумом. Попытки доказать обратное следует считать обреченными не провал. Однако следует заметить, что данная теорема описывает верхний, теоретический предел пропускной способности информационного канала, и реальные системы редко достигают его.

Управляемые носители информации

Назначением физического уровня сети является передача необработанного потока битов от одной машины к другой. Для передачи могут использоваться различные физические носители информации, называемые также средой распространения сигнала. Каждый из них имеет характерный набор полос пропускания, задержек, цен и простоты установки и использования. Носители можно разделить на две группы: управляемые носители, такие как медный провод и оптоволоконный кабель, и неуправляемые, например радиосвязь и передача по лазерному лучу без кабеля. Мы рассмотрим их в следующих разделах.

Магнитные носители

Один из самых простых способов перенести данные с одного компьютера на другой — записать их на магнитную ленту или другой съемный носитель (например, перезаписываемый DVD), физически перенести эти ленты и диски к пункту назначения и там прочитать их. Поскольку такой метод значительно проще применения, скажем, геостационарного спутника связи, он часто оказывается гораздо более эффективным в экономическом отношении, особенно для приложений,

в которых высокая пропускная способность или цена за бит являются ключевыми факторами.

Разобраться в данном вопросе нам помогут несложные вычисления. Стандартная кассета с лентой Ultrium вмещает 200 Гбайт. В коробку размером 60×60×60 помещается около 1000 таких кассет, что дает общую емкость 1600 Тбит (1,6 Пбит). Коробка с кассетами может быть доставлена в пределах США в течение 24 часов службой Federal Express или другой компанией. Эффективная полоса пропускания при такой передаче составляет 1600 Тбит/86 400 с, или 19 Гбит/с. Если же пункт назначения находится всего в часе езды, то пропускная способность составит свыше 400 Гбит/с. Ни одна компьютерная сеть пока не в состоянии даже приблизиться к таким показателям.

Если представить себе банк данных на много гигабайт, который должен ежедневно архивировать данные на запасном компьютере (чтобы иметь возможность продолжать работу даже в случае сильного наводнения или землетрясения), то похоже, что никакая технология передачи данных пока и не начала приближаться к производительности магнитных лент.

Если мы теперь взглянем на этот вопрос с экономической точки зрения, то получим сходную картину. Оптовая цена кассеты составляет около $40. Коробка с лентами обойдется в $4000, при этом одну и ту же ленту можно использовать десятки раз. Прибавим $1000 на перевозку (а на самом деле, гораздо меньше) и получим около $5000 за передачу 200 Тбайт или 3 цента за гигабайт. Ни одна сеть на земле не может соперничать с этим. Мораль этой истории такова:

Не думай свысока о скорости передачи данных автомобилем, полным кассет, с грохотом передвигающимся по дороге.

Витая пара

Хотя скорость передачи данных с помощью магнитных лент отличная, однако величина задержки при такой передаче очень велика. Время передачи измеряется минутами или часами, а не миллисекундами. Для многих приложений требуется мгновенная реакция удаленной системы (в подключенном режиме). Одним из первых и до сих пор часто применяемых средств передачи является **витая пара**. Этот носитель состоит из двух изолированных медных проводов, обычный диаметр которых составляет 1 мм. Провода свиваются один вокруг другого в виде спирали, чем-то напоминая молекулу ДНК. Это позволяет уменьшить электромагнитное взаимодействие нескольких расположенных рядом витых пар. (Два параллельных провода образуют простейшую антенну, витая пара — нет.)

Самым распространенным применением витой пары является телефонная линия. Почти все телефоны соединяются с телефонными компаниями при помощи этого носителя. Витая пара может передавать сигнал без ослабления мощности на расстояние, составляющее несколько километров. На более дальних расстояниях требуются повторители. Большое количество витых пар, тянущихся на большое расстояние в одном направлении, объединяются в кабель, на который надевается защитное покрытие. Если бы пары проводов, находящиеся внутри таких кабелей, не были свиты, то сигналы, проходящие по ним, накладывались бы друг

на друга. Телефонные кабели диаметром несколько сантиметров можно видеть протянутыми на столбах.

Витые пары могут использоваться для передачи как аналоговых, так и цифровых данных. Полоса пропускания зависит от диаметра и длины провода, но в большинстве случаев на расстоянии до нескольких километров может быть достигнута скорость несколько мегабит в секунду. Благодаря довольно высокой пропускной способности и небольшой цене витые пары широко распространены и, скорее всего, будут популярны и в будущем.

Витые пары применяются в нескольких вариантах, два из которых особенно важны в области компьютерных сетей. Витые пары **категории 3** состоят из двух изолированных проводов, свитых друг с другом. Четыре такие пары обычно помещаются вместе в пластиковую оболочку. До 1988 года большинство офисных зданий были оснащены кабелями третьей категории, тянущимися из кабельного центра на каждом этаже в отдельные офисы. Подобная схема позволяла соединять до четырех обычных телефонов или по два многоканальных телефона в каждом офисе с оборудованием телефонной компании, установленном в **кабельном центре**.

Начиная с 1988 года в офисах стали использоваться более новые витые пары **категории 5**. Они похожи на витые пары третьей категории, но имеют большее число витков на сантиметр длины проводов. Это позволяет еще сильнее уменьшить наводки между различными каналами и обеспечить улучшенное качество передачи сигнала на большие расстояния. Витые пары категории 5 более приемлемы для высокоскоростной компьютерной связи. Вскоре, вероятно, появятся кабели категорий 6 и 7, способные передавать сигнал с полосой пропускания соответственно 250 и 600 МГц (сравните с полосами в 16 и 100 МГц для категорий 3 и 5).

Все эти типы соединений часто называются **UTP** (unshielded twisted pair — неэкранированная витая пара), в противоположность громоздким дорогим экранированным кабелям из витых пар корпорации IBM, которые она представила на рынке в 1980 году, но которые так и не стали популярными за пределами фирмы IBM. Разные типы UTP схематично изображены на рис. 2.2.



а *б*

Рис. 2.2. UTP категории 3 (*а*); UTP категории 5 (*б*)

Коаксиальный кабель

Другим распространенным средством передачи данных является **коаксиальный кабель**. Он лучше экранирован, чем витая пара, поэтому может обеспечить передачу данных на более дальние расстояния с более высокими скоростями. Широко применяются два типа кабелей. Один из них, 50-омный, обычно используется для передачи исключительно цифровых данных. Другой тип кабеля, 75-омный,

часто применяется для передачи аналоговой информации, а также в кабельном телевидении. В основе такого разделения лежат скорее исторические, нежели технические факторы (например, первые дипольные антенны имели импеданс 300 Ом, и проще всего было использовать уже существующие преобразователи с отношением импеданса 4:1).

Коаксиальный кабель состоит из покрытого изоляцией твердого медного провода, расположенного в центре кабеля. Поверх изоляции натянут цилиндрический проводник, обычно выполненный в виде мелкой медной сетки. Он покрыт наружным защитным слоем изоляции (пластиковой оболочкой). Вид кабеля в разрезе показан на рис. 2.3.

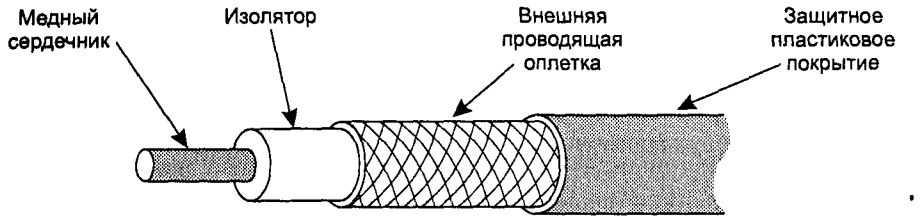


Рис. 2.3. Коаксиальный кабель

Конструкция и специальный тип экранирования коаксиального кабеля обеспечивают высокую пропускную способность и отличную помехозащищенность. Максимальная пропускная способность зависит от качества, длины и соотношения сигнал/шум линии. Современные кабели имеют полосу пропускания около 1 ГГц. Коаксиальные кабели широко применялись в телефонных системах, но теперь на линиях большой протяженности их все чаще заменяют оптоволоконными кабелями. Однако коаксиальные кабели все еще широко используются для кабельного телевидения, а также в некоторых региональных сетях.

Волоконная оптика

Быстрое развитие компьютерных технологий вызывает чувство гордости у многих представителей этой индустрии. Первый персональный компьютер фирмы IBM, созданный в 1981 году, работал с тактовой частотой 4,77 МГц. Спустя 20 лет этот показатель вырос до 2 ГГц. Прирост множителя составил 20 за декаду. Не так уж плохо.

За тот же период скорость передачи данных выросла с 56 Кбит/с (ARPANET) до 1 Гбит/с (современная оптическая связь), это означает рост в 125 раз за каждые 10 лет. При этом вероятность ошибки при передаче уменьшилась с 10^{-5} на бит почти до нуля.

В настоящее время процессоры начинают приближаться к своим физическим пределам. Скорость света преодолеть невозможно, непросто решить и проблему отвода тепловой энергии. Существующая ныне оптоволоконная технология, напротив, может развивать скорость передачи данных вплоть до 50 000 Гбит/с (50 Тбит/с), и при этом много специалистов занято поиском более совершенных

материалов. Сегодняшний практический предел в 10 Гбит/с обусловлен нашей неспособностью быстрее преобразовывать электрические сигналы в оптические и обратно, хотя в лабораторных условиях уже достигнута скорость 100 Гбит/с на одинарном волокне.

В гонке компьютеров и средств связи победили последние. Мысль о практически бесконечной полосе пропускания (при ненулевой стоимости, разумеется) еще не усвоена до конца поколением ученых-компьютерщиков, приученных мыслить в категориях низких ограничений Найквиста и Шеннона, накладываемых на медный провод. Новая точка зрения должна заключаться в том, что все компьютеры безнадежно медленны, и сетям следует любой ценой избегать вычислений независимо от того, какая часть полосы пропускания при этом будет потеряна. В данном разделе мы рассмотрим технологию передачи данных по оптическому волокну.

Оптоволоконная система передачи данных состоит из трех основных компонентов: источника света, носителя, по которому распространяется световой сигнал, и приемника сигнала, или детектора. Световой импульс принимают за единицу, а отсутствие импульса — за ноль. Свет распространяется в сверхтонком стеклянном волокне. При попадании на него света детектор генерирует электрический импульс. Присоединив к одному концу оптического волокна источник света, а к другому — детектор, мы получим однонаправленную систему передачи данных. Система принимает электрические сигналы и преобразует их в световые импульсы, передающиеся по волокну. На другой стороне происходит обратное преобразование в электрические сигналы.

Такая передающая система была бы бесполезна, если бы свет по дороге рассеивался и терял свою мощность. Однако в данном случае используется один интересный физический закон. Когда луч света переходит из одной среды в другую, например, из стекла (расплавленного и застывшего кварца) в воздух, луч отклоняется (эффект рефракции или преломления) на границе «стекло—воздух», как показано на рис. 2.4, *а*. Здесь мы видим, что луч света падает под углом α_1, выходя под углом β_1. Соотношение углов падения и отражения зависит от свойств смежных сред (в частности, от их коэффициентов преломления). Если угол падения превосходит некоторую критическую величину, луч света целиком отражается обратно в стекло, а в воздух ничего не проходит. Таким образом, луч света, падающий на границу сред под углом, превышающим критический, оказывается запертым внутри волокна, как показано на рис. 2.4, *б*, и может быть передан на большое расстояние почти без потерь.

На рис. 2.4, *б* показан только один пойманный луч света, однако поскольку любой луч света с углом падения, превышающим критический, будет отражаться от стенок волокна, то и множество лучей будет одновременно отражаться под различными углами. Про каждый луч говорят, что он обладает некоторой **модой**, а оптическое волокно, обладающее свойством передавать сразу несколько лучей, называется **многомодовым**. Однако если уменьшить диаметр волокна до нескольких длин волн света, то волокно начинает действовать подобно волноводу, и свет может двигаться только по прямой линии, без отражений от стенок волокна. Такое волокно называется **одномодовым**. Оно стоит дороже, но может использо-

ваться при передаче данных на большие расстояния. Сегодняшние одномодовые волоконные линии могут работать со скоростью 50 Гбит/с на расстоянии до 100 км. В лабораториях были достигнуты и более высокие скорости, правда, на меньших дистанциях.

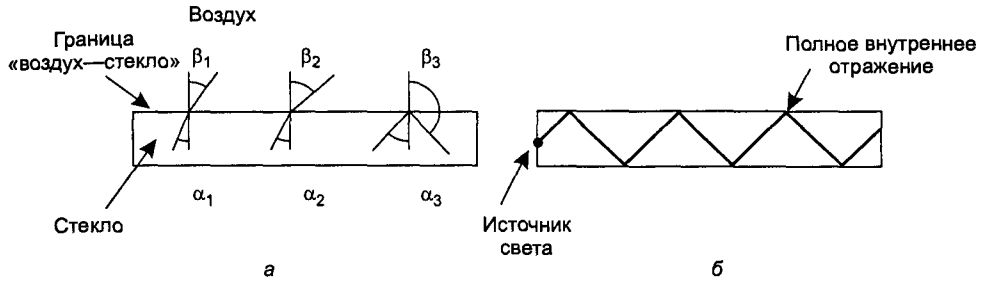


Рис. 2.4. Три примера преломления луча света, падающего под разными углами, на границе кварцевого волокна и воздуха (а); луч света, пойманный полным внутренним отражением (б)

Прохождение света по волокну

Оптическое волокно изготавливается из стекла, которое, в свою очередь, производится из песка — недорогого необработанного материала, доступного в неограниченных количествах. Изготовление стекла было известно уже в Древнем Египте, однако, чтобы свет мог проникнуть сквозь стекло, его толщина не должна превышать 1 мм, чего в то время было невозможно достичь. Стекло, достаточно прозрачное, чтобы его можно было использовать в окнах зданий, было изобретено в эпоху Возрождения. Для современных оптических кабелей применяется настолько прозрачное стекло, что если бы океаны вместо воды состояли из него, то дно океана было бы так же ясно видно, как поверхность суши с борта самолета в ясный день.

Ослабление силы света при прохождении через стекло зависит от длины волны. Для стекла, используемого в оптическом волокне, зависимость ослабления от длины волны в децибелах на километр длины волокна показана на рис. 2.5. Ослабление в децибелах вычисляется по формуле

$$\text{Ослабление в децибелах} = 10 \lg \frac{\text{передаваемая мощность}}{\text{принимаемая мощность}}.$$

Например, ослаблению мощности в два раза соответствует на графике $10 \lg 2 = 3$ дБ. На графике изображена ближняя инфракрасная часть спектра, используемая на практике. Видимый свет имеет несколько более короткие длины волн — от 0,4 до 0,7 мкм (1 мкм равен 10^{-6} м).

В системах связи используются три диапазона длин волн: 0,85, 1,30 и 1,55 мкм соответственно. Последние два обладают хорошими характеристиками ослабления (менее 5 % потерь на километр). Диапазон 0,85 мкм обладает более высоким ослаблением, однако его преимуществом является то, что для этой длины волны лазеры и электроника могут быть сделаны из одного и того же материала (арсе-

нида галлия). Все три диапазона обладают полосой пропускания от 25 000 до 30 000 ГГц.

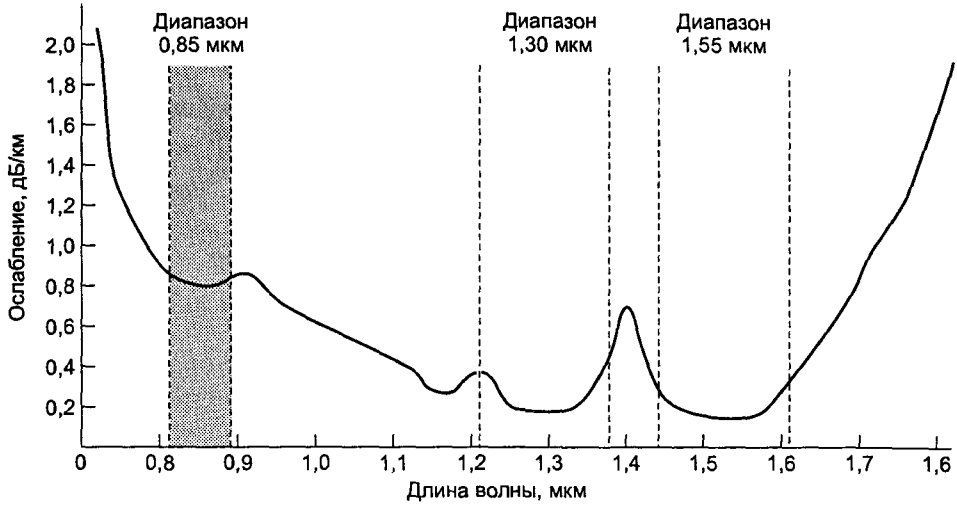


Рис. 2.5. Ослабление света в инфракрасной области спектра при прохождении через оптическое волокно

Световые импульсы удлиняются по мере их продвижения по волокну. Это удлинение называется **световой дисперсией**. Величина удлинения зависит от длины волны. Чтобы не допустить перекрывания соседних расширяющихся импульсов, можно увеличить расстояние между ними, однако при этом придется уменьшить скорость передачи. К счастью, было обнаружено, что эффект дисперсии можно предотвратить, если придавать импульсам специальную форму, а именно обратной величины от гиперболического косинуса. В этом случае будет возможно посылать импульсы на тысячи километров без искажения формы. Такие импульсы называются **уединенными волнами**. Значительная часть исследователей намерена перейти от лабораторных исследований уединенных волн к их промышленному использованию.

Оптоволоконные кабели

Структура оптоволоконного кабеля схожа с описанной ранее структурой коаксиального провода. Разница состоит лишь в том, что в первом нет экранирующей сетки. На рис. 2.6, *а* показана отдельная оптоволоконная жила. В центре ее располагается стеклянная сердцевина, по которой распространяется свет. В многомодовом оптоволокне диаметр сердечника составляет 50 мкм, что примерно равно толщине человеческого волоса. Сердечник в одномодовом волокне имеет диаметр от 8 до 10 мкм.

Сердечник покрыт слоем стекла с более низким, чем у сердечника, коэффициентом преломления. Он предназначен для более надежного предотвращения выхода света за пределы сердечника. Внешним слоем служит пластиковая оболочка,

защищающая остекление. Оптоволоконные жилы обычно группируются в пучки, защищенные внешней оболочкой. На рис. 2.6, *б* показан трехжильный кабель.

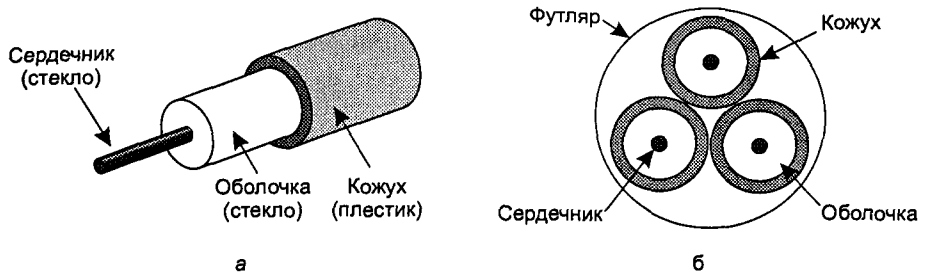


Рис. 2.6. Вид одиночного волокна сбоку (*а*); поперечное сечение трехжильного кабеля (*б*)

Обычно кабели кладутся в грунт на глубину около 1 м, где их могут случайно повредить грызуны или экскаватор. У побережья трансокеанические кабели укладываются в траншеи специальным механизмом. На большой глубине их обычно просто кладут на дно, где их могут зацепить рыболовные траулеры или перегрызть акулы.

Соединение отрезков кабеля может осуществляться тремя способами. Во-первых, на конец кабеля может прикрепляться специальный разъем, с помощью которого кабель вставляется в оптическую розетку. Подобное соединение приводит к потере 10–20 % силы света, зато оно позволяет легко изменить конфигурацию системы.

Во-вторых, они могут механически сращиваться — два аккуратно отрезанных конца кабеля укладываются рядом друг с другом и зажимаются специальной муфтой. Улучшение прохождения света достигается выравниванием концов кабеля. При этом через соединение пропускается свет, и задачей является добиться максимального соответствия мощности выходного сигнала мощности входного. Одно механическое сращивание кабелей занимает у опытного монтажника сетей около 5 минут и дает в результате потерю 10 % мощности света.

В-третьих, два куска кабеля могут быть сплавлены вместе. Сплавное соединение почти так же хорошо, как и сплошной кабель, но даже при таком методе происходит небольшое уменьшение мощности света.

Во всех трех типах соединений в точке соединения могут возникнуть отражения, и отраженный свет может интерферировать с сигналом.

Для передачи сигнала по оптоволоконному кабелю могут использоваться два типа источника света: светоизлучающие диоды (LED, Light Emitting Diode) и полупроводниковые лазеры. Они обладают различными свойствами, как показано в табл. 2.2. Их длина волны может быть настроена при помощи интерферометров Фабри—Перо (Fabry—Perot) или Маха—Цандера (Mach—Zehnder), устанавливаемых между источником и кабелем. Интерферометры Фабри—Перо представляют собой простые резонансные углубления, состоящие из двух параллельных зеркал. Свет падает перпендикулярно зеркалам, углубление отбирает те длины волн, которые укладываются в его размер целое число раз. Интерферометры Ма-

ха—Цандера разделяют свет на два луча, которые проходят различное расстояние и снова соединяются на выходе. Синфазными на выходе интерферометра окажутся лучи строго определенной длины.

Таблица 2.2. Сравнительные характеристики светодиодов и полупроводниковых лазеров

Характеристика	Светодиод	Полупроводниковый лазер
Скорость передачи данных	Низкая	Высокая
Тип волокна	Многомодовые	Многомодовые или одномодовые
Расстояние	Короткое	Дальнее
Срок службы	Долгий	Короткий
Чувствительность к температуре	Невысокая	Значительная
Цена	Низкая	Высокая

Приемный конец оптического кабеля представляет собой фотодиод, генерирующий электрический импульс, когда на него падает свет. Обычное время срабатывания фотодиода — около 1 нс, что ограничивает скорость передачи данных 1 Гбит/с. Термальный шум также имеет место, поэтому импульс света должен быть довольно мощным, чтобы его можно было обнаружить на фоне шума. При достаточной мощности импульса можно добиться пренебрежимо малой частоты ошибок.

Оптоволоконные сети

Волоконная оптика может использоваться как для междугородной связи, так и для локальных сетей, хотя ее установка значительно сложнее, чем подключение к Ethernet. Одним из вариантов соединений оптических кабелей в локальную сеть является кольцо, которое можно рассматривать как набор соединений «точка—точка», как показано на рис. 2.7. Интерфейс каждого компьютера пропускает свет далее по кольцу, а также служит Т-образным соединением, позволяющим данному компьютеру принимать и передавать сообщения.

Применяются два типа интерфейсов. Пассивный интерфейс состоит из двух ответвлений, вплавленных в основной кабель. На конце одного ответвления устанавливается светодиод или лазерный диод (для передачи), а на другом конце размещается принимающий фотодиод. Само разветвление является абсолютно пассивным элементом и поэтому в высшей степени надежным, поскольку поломка светодиода или фотодиода не приводит к разрыву кольца. Отрезанным от сети в этом случае окажется только один компьютер.

Другим типом интерфейса, показанным на рис. 2.7, является **активный повторитель**. Входящий световой импульс преобразуется в нем в электрический сигнал, усиливается при необходимости до требуемого уровня и снова пересылается в виде светового пучка. Интерфейс с компьютером представляет собой обыкновенный медный провод, соединяющий его с регенератором сигнала. Чисто оптические повторители сейчас тоже используются. Такие устройства не требуют преобразования света в электрический сигнал и обратно, поэтому они могут работать на очень больших скоростях.

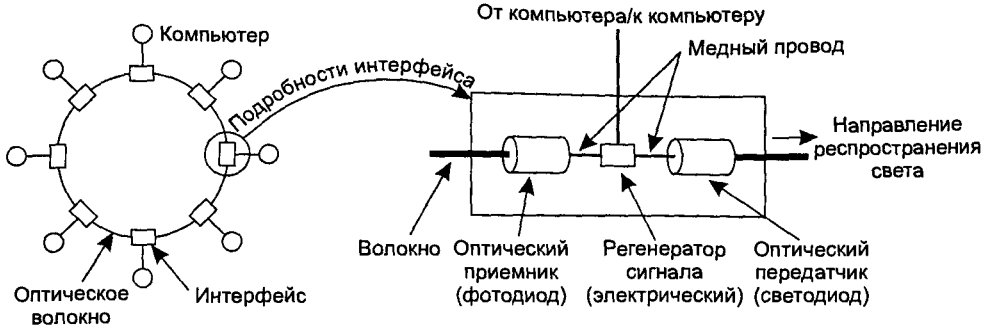


Рис. 2.7. Оптоволоконное кольцо с активными повторителями

При поломке активного повторителя кольцо разрывается и вся сеть перестает работать. С другой стороны, поскольку сигнал регенерируется каждым интерфейсом, соединения между компьютерами могут быть многокилометровой длины, что позволяет строить кольцо сколь угодно большой величины. Пассивный интерфейс ослабляет сигнал внутри каждого соединения, что сильно ограничивает количество компьютеров и размер кольца.

Кольцевая топология не является единственно возможной схемой построения локальной сети с использованием оптических кабелей. Построив сеть с топологией **пассивной звезды**, можно реализовать широковещание на основе оптоволоконных кабелей, как показано на рис. 2.8. В подобной конструкции каждый интерфейс состоит из оптического волокна, соединяющего передатчики с одним торцом стеклянного цилиндра, тогда как к другому торцу цилиндра присоединяются приемные оптические кабели. Таким образом, свет, испускаемый одним передатчиком, видят

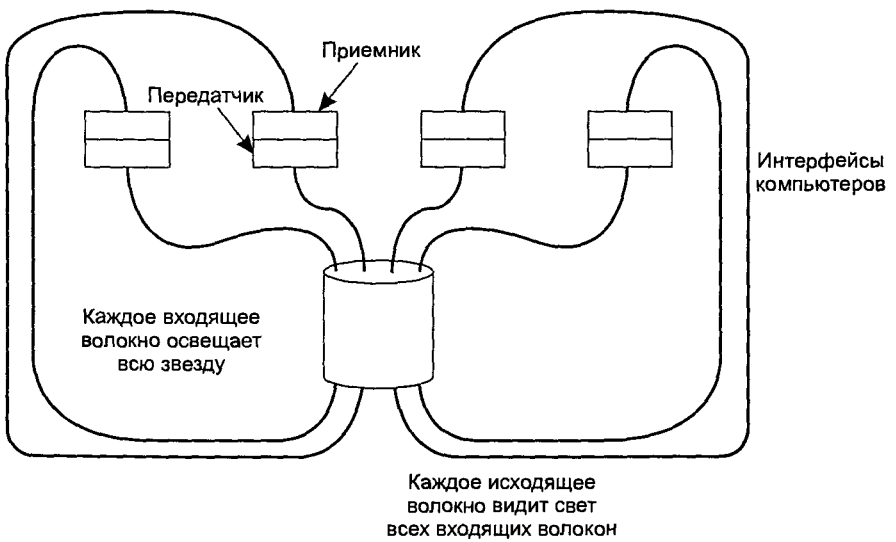


Рис. 2.8. Соединение типа «пассивная звезда» в оптоволоконных сетях

сразу все приемники. Именно так организуется широковещание. Поскольку энергия светового пучка разделяется в цилиндре между выходными линиями, количество узлов в подобной сети ограничивается чувствительностью фотодиодов.

Сравнение характеристик оптического волокна и медного провода

Сравнение характеристик оптического волокна и медного провода весьма поучительно. Оптическое волокно обладает рядом преимуществ. Во-первых, оно обеспечивает значительно более высокие скорости передачи, чем медный провод. Уже благодаря этому именно оптическое волокно должно применяться в высококачественных профессиональных сетях. Благодаря низкому коэффициенту ослабления, повторители для оптоволоконной связи требуются лишь через каждые 50 км, по сравнению с 5 км для медных проводов, что существенно снижает затраты для линий дальней связи. Преимуществом оптического волокна также является его толерантность по отношению к внешним электромагнитным возмущениям. Оно не подвержено коррозии, поскольку стекло является химически нейтральным. Это делает оптоволокно идеальным для применения на химических предприятиях.

Это может показаться странным, но телефонные компании любят оптическое волокно еще по одной причине: оно тонкое и легкое. Многие каналы для кабелей заполнены до отказа, так что новый кабель некуда положить. Если вынуть из такого канала все медные кабели и заменить их оптическими, то останется еще много свободного места, а медь можно очень выгодно продать скупщикам цветного металла. Кроме того, оптический кабель значительно легче медного. Тысяча медных витых пар длиной в 1 км весит около 8000 кг. Пара оптоволоконных кабелей весит всего 100 кг при гораздо большей пропускной способности, что значительно снижает затраты на дорогие механические системы. При прокладке новых маршрутов оптоволоконные кабели выигрывают у медных благодаря гораздо более низким затратам на их прокладку.

Наконец, оптоволоконные кабели не теряют свет, и к ним довольно сложно подключиться, что способствует их надежности и сохранности.

Отрицательной стороной оптоволоконной технологии является то, что для работы с ней требуются определенные навыки, которые имеются далеко не у всех инженеров. Кабель довольно хрупкий и ломается в местах сильных изгибов. Кроме того, поскольку оптическая передача данных является строго однонаправленной, для двухсторонней связи требуется либо два кабеля, либо две частотные полосы в одном кабеле. Наконец, оптический интерфейс стоит дороже электрического. Тем не менее очевидно, что будущее цифровой связи на расстояниях более нескольких метров — за волоконной оптикой. Подробнее обо всех аспектах оптоволоконных сетей см. (Hecht, 2001).

Беспроводная связь

В наше время появляется все большее количество информационных «наркоманов» — людей с потребностью постоянно находиться в подключенном режиме (on-line). Таким пользователям никакие кабельные соединения, будь то витая пара, коаксиальный кабель или оптическое волокно, не подходят. Им требуются по-

лучать данные непосредственно на переносные компьютеры, лэптопы, ноутбуки, электронные записные книжки, карманные компьютеры, палмтопы и компьютеры, встроенные в наручные часы. Короче говоря, они предпочитают пользоваться устройствами, не привязанными к наземным инфраструктурам. Для таких пользователей беспроводная связь является необходимостью. В данном разделе мы познакомимся с основами беспроводной связи, поскольку у нее есть ряд других важных применений, кроме предоставления доступа в Интернет желающим побродить по нему, лежа на пляже.

Существует мнение, что в будущем останется только два типа связи — оптоволоконная и беспроводная. Все стационарные (то есть не переносные) компьютеры, телефоны, факсы и т. д. будут соединяться оптоволоконными кабелями, а все переносные — с помощью беспроводной связи.

При некоторых обстоятельствах беспроводная связь может иметь свои преимущества и для стационарных устройств. Например, если прокладка оптоволоконного кабеля осложнена природными условиями (горы, джунгли, болота и т. д.), то беспроводная связь может оказаться предпочтительнее. Следует отметить, что современная беспроводная связь зародилась на Гавайских островах, где людей разделяли большие пространства Тихого океана и обычная телефонная система оказалась неприменима.

Электромагнитный спектр

Движение электронов порождает электромагнитные волны, которые могут распространяться в пространстве (даже в вакууме). Это явление было предсказано британским физиком Джеймсом Клерком Максвеллом (James Clerk Maxwell) в 1865 году. Первый эксперимент, при котором их можно было наблюдать, поставил немецкий физик Генрих Герц (Heinrich Hertz) в 1887 году. Число колебаний электромагнитных колебаний в секунду называется **частотой**, f, и измеряется в герцах (в честь Генриха Герца). Расстояние между двумя последовательными максимумами (или минимумами) называется **длиной волны**. Эта величина традиционно обозначается греческой буквой λ (лямбда).

Если в электрическую цепь включить антенну подходящего размера, то электромагнитные волны можно с успехом принимать приемником на некотором расстоянии. На этом принципе основаны все беспроводные системы связи.

В вакууме все электромагнитные волны распространяются с одной и той же скоростью, независимо от их частоты. Эта скорость называется **скоростью света**, c. Ее значение приблизительно равно $3 \cdot 10^8$ м/с, или около одного фута (30 см) за наносекунду. (Можно было бы переопределить, воспользовавшись таким совпадением, *фут*, постановив, что он равен расстоянию, которое проходит электромагнитная волна в вакууме за 1 нс. Это было бы логичнее, чем измерять длины размером сапога какого-то давно умершего короля.) В меди или стекле скорость света составляет примерно 2/3 от этой величины, кроме того, слегка зависит от частоты. Скорость света современная наука считает верхним пределом скоростей. Быстрее не может двигаться никакой объект или сигнал.

Величины f, λ и c (в вакууме) связаны фундаментальным соотношением

$$\lambda f = c. \qquad (2.2)$$

Поскольку c является константой, то, зная f, мы можем определить λ, и наоборот. Существует мнемоническое правило, которое гласит, что $\lambda f \approx 300$, если λ измеряется в метрах, а f — в мегагерцах. Например, волны с частотой 100 МГц имеют длину волны около 3 м, 1000 МГц соответствует 0,3 м, а длине волны 0,1 м соответствует частота 3000 МГц.

На рис. 2.9 изображен электромагнитный спектр. Радио, микроволновый, инфракрасный диапазоны, а также видимый свет могут быть использованы для передачи информации с помощью амплитудной, частотной или фазовой модуляции волн. Ультрафиолетовое, рентгеновское и гамма-излучения были бы даже лучше благодаря их высоким частотам, однако их сложно генерировать и модулировать, они плохо проходят сквозь здания и, кроме того, они опасны для всего живого. Диапазоны, перечисленные в нижней части рис. 2.9, представляют собой официальные названия ITU, основанные на длинах волн. Так, например, низкочастотный диапазон (LF, Low Frequency) охватывает длины волн от 1 км до 10 км (что приблизительно соответствует диапазону частот от 30 кГц до 300 кГц). Сокращения LF, MF и HF обозначают Low Frequency (низкая частота), Medium Frequency (средняя частота) и High Frequency (высокая частота) соответственно. Очевидно, при назначении диапазонам названий никто не предполагал, что будут использоваться частоты выше 10 МГц, поэтому более высокие диапазоны получили названия VHF (very high frequency — очень высокая частота), UHF (ultrahigh frequency — ультравысокая частота, УВЧ), SHF (superhigh frequency — сверхвысокая частота, СВЧ), EHF (Extremely High Frequency — чрезвычайно высокая частота) и THF (Tremendously High Frequency — ужасно высокая частота). Выше последнего диапазона имена пока не придуманы, но если следовать традиции, появятся диапазоны Невероятно (Incredibly), Поразительно (Astonishingly) и Чудовищно (Prodigiously) высоких частот (ITF, ATF и PTF).

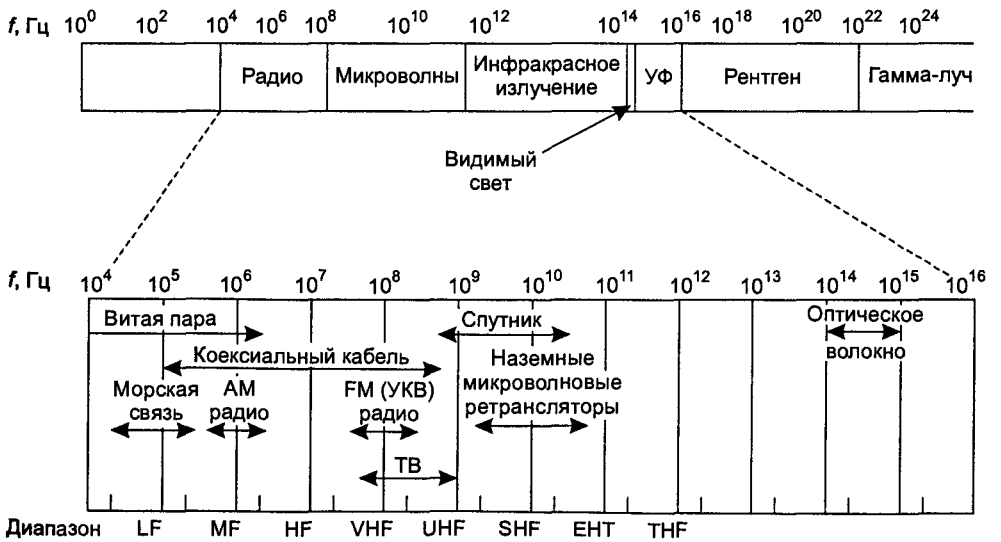


Рис. 2.9. Электромагнитный спектр и его применение в связи

Свойства радиоволн зависят от частоты. При работе на низких частотах радиоволны хорошо проходят сквозь препятствия, однако мощность сигнала в воздухе резко падает по мере удаления от передатчика. Соотношение мощности и удаленности от источника выражается примерно так: $1/r^2$. На высоких частотах радиоволны вообще имеют тенденцию распространяться исключительно по прямой линии и отражаться от препятствий. Кроме того, они поглощаются, например, дождем. Радиосигналы любых частот подвержены помехам со стороны двигателей с искрящими щетками и другого электрического оборудования.

Благодаря способности радиоволн распространяться на большие расстояния взаимные помехи, вызываемые одновременно работающими пользователями, представляют собой серьезную проблему. Поэтому все государства ведут очень строгий учет владельцев радиопередатчиков, за одним исключением (обсуждаемым далее).

В диапазонах VLF радиоволны LF и MF распространяются вдоль поверхности земли, как показано на рис. 2.10, *а*. Эти волны можно поймать радиоприемником на расстоянии около 1000 км, если используются низкие частоты, и на несколько меньших расстояниях, если частоты повыше. Радиовещание с амплитудной модуляцией (AM) использует диапазон средних волн (MF), по этой причине, например, передачи Бостонской средневолновой радиостанции не слышны в Нью-Йорке. Радиоволны этих диапазонов легко проникают сквозь здания, вследствие чего переносные радиоприемники работают и в помещениях. Основным препятствием для использования этих диапазонов для передачи данных является их относительно низкая пропускная способность (см. уравнение (2-3)).

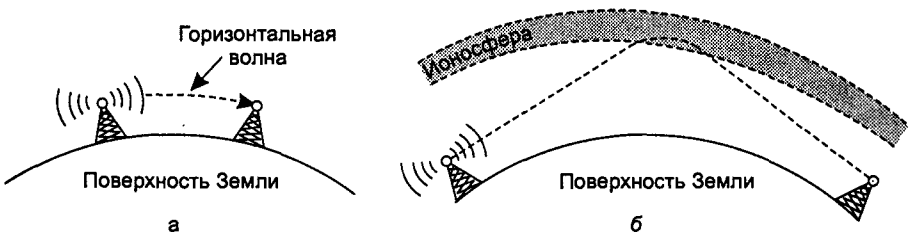


Рис. 2.10. Волны диапазонов VLF, LF и MF огибают неровности поверхности Земли (*а*); волны диапазона HF отражаются от ионосферы (*б*)

Радиоволны диапазонов HF и VHF поглощаются землей. Однако те из них, которые доходят до ионосферы, представляющей собой слой заряженных частиц, расположенный на высоте от 100 до 500 км, отражаются ею и посылаются обратно к поверхности Земли, как показано на рис. 2.10, *б*. При определенных атмосферных условиях сигнал может отразиться несколько раз. Радиолюбители используют такие диапазоны частот для дальней связи. Военные также осуществляют связь в диапазонах HF и VHF.

Связь в микроволновом диапазоне

На частотах выше 100 МГц радиоволны распространяются почти по прямой, поэтому могут быть сфокусированы в узкие пучки. Концентрация энергии в виде узкого пучка при помощи параболической антенны (вроде всем известной спут-

никовой телевизионной тарелки) приводит к улучшению соотношения сигнал/шум, однако для подобной связи передающая и принимающая антенны должны быть довольно точно направлены друг на друга. Кроме того, подобная направленность позволяет использовать несколько передатчиков, установленных в ряд, сигналы от которых принимаются также установленными в ряд приемными антеннами без взаимных помех. До изобретения оптоволоконной связи подобные микроволновые антенны в течение десятков лет составляли основу междугородной телефонной связи. На самом деле компания MCI, один из основных конкурентов AT&T, построила целую систему микроволновой связи с передачей сигнала от одной башни к другой. Расстояние между антеннами составляло десятки километров. Эта технология нашла отражение даже в названии компании: аббревиатура оператора междугородней связи MCI изначально расшифровывалась как Microwave Communications, Inc. С тех пор, впрочем, MCI уже успела перейти на оптоволоконные сети и объединилась с компанией WorldCom.

Микроволны распространяются строго по прямой, поэтому при слишком большом удалении антенн друг от друга на пути следования сигнала может оказаться земная поверхность (например, так случится, если поставить передатчик в Сан-Франциско, а приемник — в Амстердаме). Чем выше ретрансляционные башни, тем больше может быть расстояние между ними. Максимальное расстояние между повторителями можно очень грубо оценить как корень квадратный из их высоты. Так, при высоте ретрансляторов 100 м расстояние между ними может быть около 80 км.

В отличие от радиоволн с более низкими частотами, микроволны плохо проходят сквозь здания. Кроме того, даже при точной фокусировке луча на приемной антенне при прохождении сквозь пространство луч довольно значительно расширяется в диаметре. Часть волн может отражаться атмосферными слоями, благодаря чему на своем пути к приемной антенне отраженные волны пройдут большее расстояние, чем прямые. Это означает, что первые будут отличаться от последних по фазе, что может привести к подавлению сигнала. Такой эффект называется **многолучевым затуханием** и довольно часто представляет собой серьезную проблему. Наличие этого эффекта зависит от погоды и от частоты. Некоторые операторы связи держат около 10 % своих каналов свободными и временно переключаются на них в случае возникновения многолучевого затухания на какой-либо частоте.

Потребности во все большем диапазоне частот заставляют постоянно совершенствовать технологию, благодаря чему для связи используются все более высокие частоты. Диапазоны частот до 10 ГГц теперь применяются довольно широко, однако при частотах выше 4 ГГц появляется новая проблема: поглощение водой. Длина волн при такой частоте составляет всего несколько сантиметров, и такие волны сильно поглощаются дождем. Такой эффект может быть весьма полезен для тех, кто хочет соорудить огромную наружную микроволновую печь, чтобы жарить пролетающих мимо птичек, однако он представляет собой серьезную проблему в области радиосвязи. Пока что единственным решением является отключение линий связи, пересекаемых полосой дождя, и переключение на обходные пути.

Свойства радиоволн зависят от частоты. При работе на низких частотах радиоволны хорошо проходят сквозь препятствия, однако мощность сигнала в воздухе резко падает по мере удаления от передатчика. Соотношение мощности и удаленности от источника выражается примерно так: $1/r^2$. На высоких частотах радиоволны вообще имеют тенденцию распространяться исключительно по прямой линии и отражаться от препятствий. Кроме того, они поглощаются, например, дождем. Радиосигналы любых частот подвержены помехам со стороны двигателей с искрящими щетками и другого электрического оборудования.

Благодаря способности радиоволн распространяться на большие расстояния взаимные помехи, вызываемые одновременно работающими пользователями, представляют собой серьезную проблему. Поэтому все государства ведут очень строгий учет владельцев радиопередатчиков, за одним исключением (обсуждаемым далее).

В диапазонах VLF радиоволны LF и MF распространяются вдоль поверхности земли, как показано на рис. 2.10, *а*. Эти волны можно поймать радиоприемником на расстоянии около 1000 км, если используются низкие частоты, и на несколько меньших расстояниях, если частоты повыше. Радиовещание с амплитудной модуляцией (AM) использует диапазон средних волн (MF), по этой причине, например, передачи Бостонской средневолновой радиостанции не слышны в Нью-Йорке. Радиоволны этих диапазонов легко проникают сквозь здания, вследствие чего переносные радиоприемники работают и в помещениях. Основным препятствием для использования этих диапазонов для передачи данных является их относительно низкая пропускная способность (см. уравнение (2-3)).

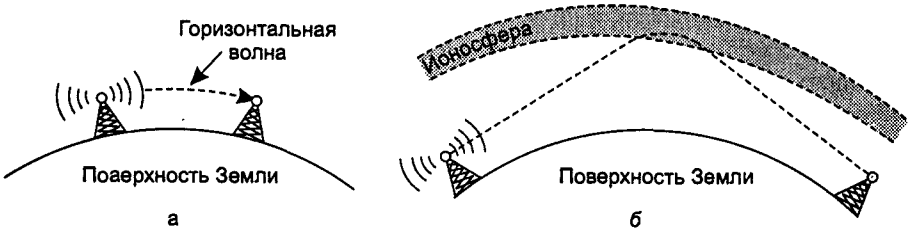


Рис. 2.10. Волны диапазонов VLF, LF и MF огибают неровности поверхности Земли (*а*); волны диапазона HF отражаются от ионосферы (*б*)

Радиоволны диапазонов HF и VHF поглощаются землей. Однако те из них, которые доходят до ионосферы, представляющей собой слой заряженных частиц, расположенный на высоте от 100 до 500 км, отражаются ею и посылаются обратно к поверхности Земли, как показано на рис. 2.10, *б*. При определенных атмосферных условиях сигнал может отразиться несколько раз. Радиолюбители используют такие диапазоны частот для дальней связи. Военные также осуществляют связь в диапазонах HF и VHF.

Связь в микроволновом диапазоне

На частотах выше 100 МГц радиоволны распространяются почти по прямой, поэтому могут быть сфокусированы в узкие пучки. Концентрация энергии в виде узкого пучка при помощи параболической антенны (вроде всем известной спут-

никовой телевизионной тарелки) приводит к улучшению соотношения сигнал/шум, однако для подобной связи передающая и принимающая антенны должны быть довольно точно направлены друг на друга. Кроме того, подобная направленность позволяет использовать несколько передатчиков, установленных в ряд, сигналы от которых принимаются также установленными в ряд приемными антеннами без взаимных помех. До изобретения оптоволоконной связи подобные микроволновые антенны в течение десятков лет составляли основу междугородной телефонной связи. На самом деле компания MCI, один из основных конкурентов AT&T, построила целую систему микроволновой связи с передачей сигнала от одной башни к другой. Расстояние между антеннами составляло десятки километров. Эта технология нашла отражение даже в названии компании: аббревиатура оператора междугородней связи MCI изначально расшифровывалась как Microwave Communications, Inc. С тех пор, впрочем, MCI уже успела перейти на оптоволоконные сети и объединилась с компанией WorldCom.

Микроволны распространяются строго по прямой, поэтому при слишком большом удалении антенн друг от друга на пути следования сигнала может оказаться земная поверхность (например, так случится, если поставить передатчик в Сан-Франциско, а приемник — в Амстердаме). Чем выше ретрансляционные башни, тем больше может быть расстояние между ними. Максимальное расстояние между повторителями можно очень грубо оценить как корень квадратный из их высоты. Так, при высоте ретрансляторов 100 м расстояние между ними может быть около 80 км.

В отличие от радиоволн с более низкими частотами, микроволны плохо проходят сквозь здания. Кроме того, даже при точной фокусировке луча на приемной антенне при прохождении сквозь пространство луч довольно значительно расширяется в диаметре. Часть волн может отражаться атмосферными слоями, благодаря чему на своем пути к приемной антенне отраженные волны пройдут большее расстояние, чем прямые. Это означает, что первые будут отличаться от последних по фазе, что может привести к подавлению сигнала. Такой эффект называется **многолучевым затуханием** и довольно часто представляет собой серьезную проблему. Наличие этого эффекта зависит от погоды и от частоты. Некоторые операторы связи держат около 10 % своих каналов свободными и временно переключаются на них в случае возникновения многолучевого затухания на какой-либо частоте.

Потребности во все большем диапазоне частот заставляют постоянно совершенствовать технологию, благодаря чему для связи используются все более высокие частоты. Диапазоны частот до 10 ГГц теперь применяются довольно широко, однако при частотах выше 4 ГГц появляется новая проблема: поглощение водой. Длина волн при такой частоте составляет всего несколько сантиметров, и такие волны сильно поглощаются дождем. Такой эффект может быть весьма полезен для тех, кто хочет соорудить огромную наружную микроволновую печь, чтобы жарить пролетающих мимо птичек, однако он представляет собой серьезную проблему в области радиосвязи. Пока что единственным решением является отключение линий связи, пересекаемых полосой дождя, и переключение на обходные пути.

Микроволновая радиосвязь стала настолько широко использоваться в междугородной телефонии, сотовых телефонах, телевещании и других областях, что начала сильно ощущаться нехватка ширины спектра. Данная связь имеет ряд преимуществ перед оптоволокном. Главное из них состоит в том, что не нужно прокладывать кабель, соответственно, не нужно платить за аренду земли на пути сигнала. Достаточно купить маленькие участки земли через каждые 50 км и установить на них ретрансляционные вышки, обойдя, таким образом, телефонные кабельные системы. Именно поэтому корпорации MCI удалось быстро внедриться в рынок междугородной связи. Компания Sprint пошла другим путем: она была образована Южной Тихоокеанской железной дорогой (South Pacific Railroad), которая уже владела правами на большой участок пути и просто закапывала кабель рядом с железнодорожным полотном.

Кроме того, микроволновая связь является относительно недорогой. Установка двух примитивных вышек (это могут быть просто большие столбы на четырех растяжках) с антеннами на каждой из них, скорее всего, обойдется дешевле, чем прокладка 50 км кабеля в перенаселенной городской местности или в горах. Это может быть также дешевле, чем аренда оптоволоконной линии у телефонной компании, особенно если телефонная компания еще не полностью расплатилась за медный кабель, который она уже сменила на оптоволоконный.

Политика распределения частот

Для предотвращения анархии при использовании частот существуют определенные национальные и международные соглашения, касающиеся политики их распределения. Понятно, что всем хочется сделать связь максимально быстрой, поэтому все хотели бы получить в свое распоряжение максимально широкий спектр. Национальные правительства распределяют частоты между AM- и FM-радиостанциями, телевидением, операторами сотовой связи, а также телефонными компаниями, полицией, морскими и аэронавигационными службами, военными, администрацией и еще многими другими потенциальными клиентами. Международное агентство ITU-R (WARC) пытается скоординировать действия различных структур, чтобы можно было производить устройства, способные работать в любой точке планеты. Тем не менее, рекомендации ITU-R не являются обязательными для исполнения. Так, например, Федеральная комиссия по связи, FCC (Federal Communication Commision), занимающаяся раздачей частотных диапазонов в США, иногда пренебрегает этими рекомендациями — чаще всего из-за соответствующей убедительной просьбы какой-нибудь влиятельной политической группировки, которой требуется конкретная часть спектра.

Даже если определенный диапазон выделен под конкретные цели (например, под сотовую связь), то встает новый вопрос: как распределять те или иные частоты внутри диапазона между операторами связи? В прошлом были популярны три алгоритма. Первый из них, часто называемый **конкурсом красоты**, подразумевал подробные объяснения претендентов, доказывающие, что именно предлагаемый ими сервис лучше всего отвечает интересам общественности. После этого «жюри» решает, чья история выглядит самой красивой. Такая направленность на угождение администрации, зачастую подкрепленная, что называется, рублем, ведет лишь к развитию взяточничества, коррупции, непотизма и т. д. Более того,

даже если какой-нибудь честный чиновник видит, что иностранная фирма может сделать для отечественного потребителя больше, чем иные национальные компании, то ему придется долго это доказывать, преодолевая сопротивление своих коллег.

Такие наблюдения в конце концов привели к созданию альтернативного алгоритма — обычной **лотереи** среди компаний, желающих получить свою долю спектрального пирога. Проблема здесь лишь в том, что в лотерее могут участвовать и фирмы, совершенно не заинтересованные ни в каких частотах. Например, если определенный частотный диапазон достается какому-нибудь крупному ресторану, он может очень выгодно продать его, ничем не рискуя.

Этот алгоритм очень бурно критиковался за то, что выиграть могут совершенно случайные лица. Результатом стало внедрение третьего алгоритма — **аукциона**. На аукционных торгах частоту выигрывал тот покупатель, который мог выложить наибольшую сумму. Когда в Англии в 2000 году проводился аукцион между операторами мобильной связи третьего поколения, ожидаемая сумма доходов составляла 4 миллиарда долларов. Она достигла 40 миллиардов, поскольку операторы просто впали в бешенство в борьбе за будущее своего бизнеса, предпочитая умереть, но не уйти с рынка мобильной связи. Эти события очень заинтересовали правительства других стран, которые тоже были не прочь получить такой доход, не прикладывая никак усилий. На аукционную систему перешли многие, и она работала, оставляя на своем пути обессиленные такой конкурентной борьбой компании, в один миг оказавшиеся на грани банкротства. В лучшем случае компаниям, выигравшим частоту, требуется несколько лет, чтобы выплатить все свои долги.

Совершенно другим подходом является следующий: вообще не распределять частоты. Пусть каждый работает на той частоте, которая ему больше нравится, но следит за мощностью своих передатчиков: она не должна быть такой, чтобы сигналы накладывались друг на друга. В соответствии с этим принципом, было решено выделить несколько частотных диапазонов, называемых **ISM** (Industrial, Scientific, Medical, то есть промышленные, научные, медицинские). Для работы в этих диапазонах не требуется специальной лицензии. Устройства, открывающие ворота гаража, домашние радиотелефоны, радиоуправляемые игрушки, беспроводные мыши и многие-многие другие устройства работают на ISM. Для уменьшения интерференции между независимыми устройствами, работающими в одном и том же диапазоне, комиссией FCC им предписывается использовать технологию расширенного спектра (см. ранее). Такие правила принимаются во всем мире.

Конкретные диапазоны ISM в разных странах свои. Например, в США устройства мощностью менее 1 Вт могут использовать диапазоны, показанные на рис. 2.11, без получения лицензии FCC. Диапазон 900 МГц — это лучшее, что есть во всем наборе ISM, но он уже перегружен и к тому же доступен не во всех странах. Диапазон 2,4 ГГц работает в большинстве стран, но подвержен помехам от микроволновых печей и радарных устройств. В этом же диапазоне работает система Bluetooth и некоторые ЛВС стандарта 802.11. Есть еще один, новый стандартный диапазон 5,7 ГГц, но он находится в начальной стадии своего развития, поэтому оборудование для него стоит очень дорого. Однако, поскольку многие сети 802.11 активно используют его, он становится все более популярным.

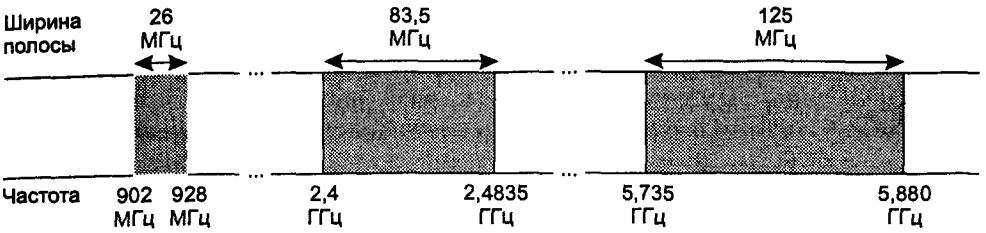


Рис. 2.11. Диапазоны ISM в США

Инфракрасные и миллиметровые волны

Инфракрасное и миллиметровое излучения без использования кабеля широко применяется для связи на небольших расстояниях. Дистанционные пульты управления для телевизоров, видеомагнитофонов и стереоаппаратуры используют инфракрасное излучение. Они относительно направленные, дешевые и легко устанавливаемые, но имеют один важный недостаток: инфракрасное излучение не проходит сквозь твердые объекты (попробуйте встать между телевизором и пультом). Мы начали с рассмотрения длинных радиоволн и постепенно продвигаемся к видимому свету, и уже инфракрасные волны мало напоминают радиоволны и ведут себя, как свет.

С другой стороны, тот факт, что инфракрасные волны не проходят сквозь стены, является также и положительным. Ведь это означает, что инфракрасная система в одной части здания не будет интерферировать с подобной системой в соседней комнате — вы, к счастью, не сможете управлять со своего пульта телевизором соседа. Кроме того, это повышает защищенность инфракрасной системы от прослушивания по сравнению с радиосистемой. По этой причине для использования инфракрасной системы связи не требуется государственная лицензия, в отличие от радиосвязи (кроме диапазонов ISM). Связь в инфракрасном диапазоне применяется в настольных вычислительных системах (например, для связи ноутбуков с принтерами), но все же не играет значимой роли в телекоммуникации.

Связь в видимом диапазоне

Ненаправленные оптические сигналы использовались в течение нескольких веков. Герой американской войны за независимость Пол Ревер (Paul Revere) в 1775 году в Бостоне использовал двоичные оптические сигналы, информируя с колокольни Старой Северной церкви (Old North Church) население о наступлении англичан. Более современным приложением является соединение локальных сетей в двух зданиях при помощи лазеров, установленных на крышах. Связь с помощью когерентных волн лазера является сугубо однонаправленной, поэтому для двусторонней связи необходимо на каждой крыше установить по лазеру и по фотодетектору. Такая технология позволяет организовать связь с очень высокой пропускной способностью при очень низкой цене. Кроме того, такая система довольно просто монтируется и, в отличие от микроволновой связи, не требует лицензии FCC (Федеральной комиссии связи США).

Узкий луч является сильной стороной лазера, однако он создает и некоторые проблемы. Чтобы попасть миллиметровым лучом в мишень диаметром 1 мм на расстоянии 500 м, требуется снайперское искусство высочайшей пробы. Обычно на лазеры устанавливаются линзы для небольшой расфокусировки луча.

Недостатком лазерного луча является также неспособность проходить сквозь дождь или густой туман, хотя в солнечные ясные дни он работает прекрасно. Тем не менее, автор однажды присутствовал на конференции в современной европейской гостинице, где организаторы заботливо предоставили комнату, полную терминалов, чтобы участники конференции могли читать свою электронную почту во время скучных презентаций. Поскольку местная телефонная станция не желала устанавливать большое количество телефонных линий всего на три дня, организаторы установили лазер на крыше и нацелили его на здание университетского компьютерного центра, который находится на расстоянии нескольких километров. В ночь перед конференцией они проверили связь — она работала прекрасно. В 9 часов следующего утра, в ясный солнечный день связь была полностью потеряна и отсутствовала весь день. Вечером организаторы опять тщательно проверили связь и снова убедились в ее прекрасной работе. На следующий день связи опять не было.

Когда конференция закончилась, организаторы обсудили эту проблему. Как выяснилось, в дневное время солнце нагревало крышу, горячий воздух от нее поднимался и отклонял лазерный луч, начинавший танцевать вокруг детектора (рис. 2.12). Этот эффект можно наблюдать невооруженным глазом в жаркий день на шоссе или над горячим радиатором автомобиля. Борясь с этим эффектом, астрономы располагают свои телескопы высоко в горах, подальше от атмосферы.

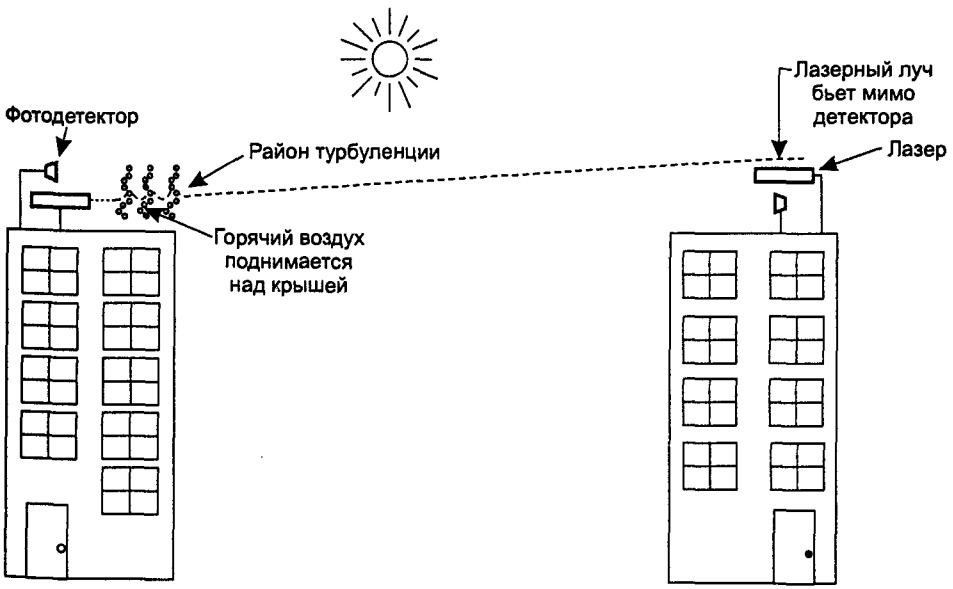


Рис. 2.12. Конвекционные потоки мешают работать лазерной системе. На рисунке изображена двунаправленная система с даумя лазерами

Спутники связи

В 1950-х и начале 60-х годов люди пытались организовать связь при помощи сигналов, отраженных от металлических метеозондов. К сожалению, мощность таких сигналов была слишком мала, и их практическое значение оказалось ничтожным. Затем ВМФ США обнаружил, что в небе постоянно висит некое подобие метеозонда — это была Луна. Была построена система для связи береговых служб с кораблями, в которой использовалось отражение сигналов от естественного спутника Земли.

Дальнейший прогресс в создании коммуникаций с помощью небесных тел на этом приостановился до запуска первого спутника связи. Ключевым отличием искусственной «луны» являлось то, что на спутнике было установлено оборудование, позволяющее усилить входящий сигнал перед отправкой его обратно на Землю. Это превратило космическую связь из забавного курьеза в мощную технологию.

Спутникам связи присущи определенные свойства, делающие их чрезвычайно привлекательными для самых разных областей применения. Проще всего представить себе спутник связи в виде своего рода огромного микроволнового повторителя, висящего в небе. Он включает в себя несколько **транспондеров**, каждый из которых настроен на определенную часть частотного спектра. Транспондеры усиливают сигналы и преобразуют их на новую частоту, чтобы при отправке на Землю отраженный сигнал не накладывался на прямой.

Нисходящий луч может быть как широким, покрывающим огромные пространства на Земле, так и узким, который можно принять в области, ограниченной лишь несколькими сотнями километров. Последний метод называется **трубой**.

В соответствии с законом Кеплера, период обращения спутника равен радиусу орбиты в степени 3/2. Таким образом, чем выше орбита, тем дольше период. Вблизи поверхности Земли период обращения вокруг нее составляет примерно 90 минут. Следовательно, спутники, расположенные на малой высоте, слишком быстро исчезают из вида приемно-передающих устройств, расположенных на Земле, поэтому необходимо организовывать непрерывные зоны покрытия. На высоте 35 800 км период составляет 24 часа. А на высоте 384 000 км спутник будет обходить Землю целый месяц, в чем может убедиться любой желающий, наблюдая за Луной.

Конечно, период обращения спутника очень важно иметь в виду, но это не единственный критерий, по которому определяют, где его разместить. Необходимо принимать во внимание так называемые пояса Ван Аллена (Van Allen belts) — области скопления частиц с большим зарядом, находящихся в зоне действия магнитного поля Земли. Любой спутник, попав в такой пояс, довольно быстро будет уничтожен этими частицами. В результате учета этих факторов были выделены три зоны, в которых можно безопасно размещать искусственные спутники. Они изображены на рис. 2.13. Из этого же рисунка можно узнать о некоторых из их свойств. Мы вкратце рассмотрим спутники, размещаемые в каждой из этих трех зон.



Рис. 2.13. Спутники связи и их свойства: высота орбиты, задержка, число спутников, необходимое для покрытия всей поверхности земного шара

Геостационарные спутники

В 1945 году писатель-фантаст Артур С. Кларк (Arthur S. Clarke) подсчитал, что спутник, расположенный на высоте 35 800 км на круговой экваториальной орбите, будет оставаться неподвижным относительно Земли. А значит, следить за ним будет гораздо проще (Clarke, 1945). Он развил свою мысль и описал целую коммуникационную систему, использующую такие (пилотируемые) **геостационарные спутники**. Он описал орбиты, солнечные батареи, радиочастоты и даже процедуры связи. К сожалению, в конце концов он пришел к неутешительному выводу о том, что такие спутники вряд ли будут иметь практическое значение, потому что на их борту невозможно разместить энергоемкие, хрупкие ламповые усилители. В связи с этим Кларк не стал больше развивать свою идею, хотя и написал несколько фантастических рассказов о подобных искусственных спутниках.

Положение вещей изменило изобретение транзистора, и вот в июле 1962 года производится запуск первого в мире спутника связи Telstar.

С тех пор спутники связи стали многомиллиардным бизнесом и единственным прибыльным делом, связанным с космическими технологиями. Про спутники, вращающиеся на большой высоте, говорят, что они расположены на **геостационарной орбите** (GEO, Geostationary Earth Orbit).

Современные технологии таковы, что расположение спутников чаще, чем через каждые 2° в 360-градусной экваториальной плоскости, является нерациональным. В противном случае возможна интерференция сигналов. Итак, если на каждые два градуса приходится 1 спутник, то всего их в экваториальной плоскости можно разместить 360/2 = 180. Сто восемьдесят спутников могут одновременно находиться в небе и вращаться в одной и той же плоскости на одной и той же высоте. Тем не менее у каждого транспондера есть возможность работы на разных частотах и с разной поляризацией, что позволяет увеличить максимальную пропускную способность всей системы.

Со временем возникла необходимость предотвращения беспорядочного использования околоземных орбит. Навести порядок в небе было поручено организации ITU. Процесс выделения орбит очень сильно связан с политикой, причем многие страны в борьбе за свой «кусок» неба напоминают далеких предков человека из каменного века. Это объясняется очень высокими потенциальными доходами, которые государство может извлечь, сдавая в аренду кусочки космоса. В то же время некоторые страны заявляют, что их государственные границы в высоту простираются до самой Луны и что использование орбит, проходящих над их территорией, иностранными государствами является нелегальным. Жаркие споры на эту тему подогревает еще и тот факт, что коммерческая связь — это далеко не единственное применение спутников связи, а значит, и их орбит. Ими пользуются операторы спутникового телевидения, правительственные структуры и военные.

Современные спутники могут быть довольно большими, весят до 4000 кг и потребляют до нескольких киловатт электроэнергии, вырабатываемой солнечными батареями. Эффекты гравитации, вызванные Солнцем, Луной и другими планетами, постепенно вызывают смещение с орбит и изменение ориентации. Приходится компенсировать это с помощью бортовых двигателей. Действия по сохранению параметров орбит спутников называются **позиционированием**. И все же приходит момент, когда топливо у бортовых двигателей заканчивается (такое случается примерно один раз в десять лет). Тогда спутник начинает беспомощно дрейфовать, постепенно сходя с орбиты. Понятно, что он перестает быть дееспособным и его нужно отключать. Обычно спутники связи заканчивают свою жизнь, постепенно входя в плотные слои атмосферы и сгорая там либо падая на землю.

Участки орбит — это не единственный предмет, за который борются страны и отдельные компании. Разумеется, распределению между всеми желающими подлежат и рабочие диапазоны частот, поскольку нисходящие сигналы спутников могут вызывать помехи в работе микроволновых устройств. Поэтому ITU были выделены частотные диапазоны, предназначенные исключительно для спутников связи. Самые важные из них показаны на табл. 2.3.

Таблица 2.3. Основные частотные диапазоны спутников связи

Диапазон	Нисходящие сигналы	Восходящие сигналы	Ширина полосы	Пробламы
L	1,5 ГГц	1,6 ГГц	15 МГц	Узкая полоса; переполнен
S	1,9 ГГц	2,2 ГГц	70 МГц	Узкая полоса; переполнен
C	4,0 ГГц	6,0 ГГц	500 МГц	Наземная интерференция
Ku	11 ГГц	14 ГГц	500 МГц	Дождь
Ka	20 ГГц	30 ГГц	3500 МГц	Дождь, стоимость оборудования

Диапазон C был первой полосой частот, предназначенной для трафика коммерческих спутников. Он разбивается на два поддиапазона. Один из них предназначен для сигналов с Земли (восходящих), другой — для сигналов со спутника (нисходящих). Таким образом, для двусторонней передачи требуется сразу два

канала. Они уже переполнены пользователями, поскольку на тех же частотах работают наземные микроволновые устройства связи. В 2000 году, в соответствии с международным соглашением, было добавлено два дополнительных диапазона: S и L. Тем не менее, они тоже весьма узки и уже заполнены.

Следующий высокочастотный диапазон коммерческой связи называется Ku (K under, то есть «под K»). Полоса пока еще не переполнена, и работающие на этих частотах спутники могут располагаться на угловом расстоянии 1° друг от друга. У диапазона Ku имеется еще одна проблема: волны этих частот глушатся дождем. Вода очень плохо пропускает микроволновый сигнал. К счастью, очень сильные ливни обычно бывают весьма узко локализованы, поэтому проблему удается решить с помощью нескольких наземных установок, расположенных довольно далеко друг от друга. Цена, которую приходится платить за «проблему дождя», весьма высока: это дополнительные антенны, кабели и электронные устройства для быстрого переключения станций. Наконец, самым высокочастотным диапазоном является Ka (K above, то есть «над K»). Основной проблемой является пока еще очень высокая стоимость оборудования для работы на этих частотах. Помимо коммерческих диапазонов, существует также множество военных и правительственных.

На современном спутнике имеется порядка 40 транспондеров, полоса каждого из которых составляет 80 МГц. Обычно каждый транспондер работает по принципу узкой трубы, однако недавно появились спутники, оснащенные бортовыми процессорами для обработки сигналов. В первых спутниках разделение транспондеров по каналам было статическим: весь доступный рабочий диапазон просто разделялся на несколько фиксированных полос. Теперь же сигнал транспондера разделяется на временные слоты, то есть каждому пользователю выделяется на передачу определенный промежуток времени. Далее в этой главе мы изучим оба принципа (частотное и временное мультиплексирование) более подробно.

Первые геостационарные спутники связи имели один луч, который охватывал примерно 1/3 земной поверхности и назывался **точечным лучом**. Однако по мере удешевления, уменьшения размеров и энергоемкости микроэлектронных элементов стали появляться более сложные стратегии. Стало возможно оборудовать каждый спутник несколькими антеннами и несколькими транспондерами. Каждый нисходящий луч сфокусировали на небольшой территории; таким образом смогли осуществить одновременную передачу нескольких сигналов. Обычно эти так называемые **пятна** имеют форму овала и могут иметь относительно малые размеры — порядка нескольких сотен километров. Американский спутник связи охватывает широким лучом 48 штатов, а также имеет два узких луча для Аляски и Гавайских островов.

Новым витком развития спутников связи стало создание недорогих миниатюрных апертурных терминалов — **VSAT** (Very Small Aperture Terminal) (Abramson, 2000). У этих небольших станций имеется антенна диаметром всего 1 м (сравните с 10-метровой антенной GEO), их выходная мощность составляет примерно 1 Вт. Скорость работы в направлении Земля – спутник обычно составляет 19,2 Кбит/с, зато связь спутник – Земля можно поддерживать на скорости 512 Кбит/с и выше. Спутниковое широковещательное телевидение использует эту технологию для односторонней передачи сигнала.

Многим микростанциям VSAT не хватает мощности для того, чтобы связываться друг с другом (через спутник, разумеется). Для решения этой проблемы устанавливаются специальные наземные концентраторы с большой мощной антенной. Концентратор (хаб) распределяет трафик между несколькими VSAT, как показано на рис. 2.14. В таком режиме либо приемник, либо передатчик обязательно имеет большую антенну и мощный усилитель. Недостатком такой системы является наличие задержек, достоинством — низкая цена за полноценную систему для конечного пользователя.

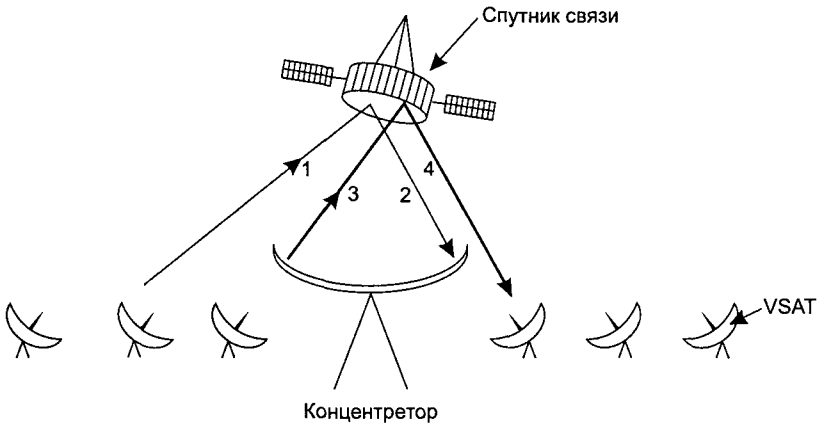


Рис. 2.14. Хаб распределяет трафик между несколькими VSAT

Системы VSAT имеют большие перспективы использования в сельской местности. Об этом как-то не очень часто вспоминают, но половина населения земного шара живет минимум в часе ходьбы от ближайшего телефона. Протянуть телефонные линии ко всем селам и деревням не по карману большинству стран так называемого третьего мира. Однако средств на установку тарелки VSAT, питающейся от солнечной батареи, может хватить не только у администрации региона, но и у частных лиц. Таким образом, VSAT — это технология, которая может позволить организовать связь в любой точке планеты.

Спутники связи обладают рядом свойств, которые радикально отличают их от любых наземных систем связи между абонентами. Во-первых, несмотря на предельно высокую скорость распространения сигнала (собственно, она практически равна скорости света — 300 000 км/с), расстояния между наземными приемно-передающими устройствами и спутниками таковы, что в технологии GEO задержки оказываются весьма значительными. В зависимости от взаимного расположения пользователя, наземной станции и спутника время передачи может составлять 250–300 мс. Обычно оно составляет 270 мс (соответственно, в два раза больше — 540 мс — в системах VSAT, работающих через хаб).

Для сравнения, сигнал в наземных микроволновых системах связи распространяется со скоростью примерно 3 мкс/км, а коаксиальный кабель и оптоволокно имеют задержку порядка 5 мкс/км. Разница задержек здесь объясняется тем, что в твердых телах сигнал распространяется медленнее, чем в воздухе.

Еще одним важным свойством спутников является то, что они являются исключительно широковещательным средством передачи данных. На отправку сообщения сотням абонентов, находящихся в зоне следа спутника, не затрачивается никаких дополнительных ресурсов по сравнению с отправкой сообщения одному из них. Для некоторых применений это свойство очень полезно. Например, можно представить себе кэширование на спутнике популярных веб-страниц, что резко повысит скорость их загрузки на сотни компьютеров, находящихся довольно далеко друг от друга. Конечно, широковещание симулируется обычными двухточечными сетями, однако спутниковое вещание в этом случае обходится значительно дешевле. С другой стороны, с точки зрения защиты информации и конфиденциальности данных, спутники — это прямо-таки беда: кто угодно может прослушивать абсолютно все. Здесь на защиту тех, кому важен ограниченный доступ к информации, встает криптография.

Спутники связи обладают еще одним замечательным свойством — независимостью стоимости передачи от расстояния между узлами. Звонок другу, живущему за океаном, стоит столько же, сколько звонок подружке, живущей в соседнем доме. Космические телекоммуникационные технологии, кроме того, обеспечивают очень высокую степень защиты от ошибок и могут быть развернуты на местности практически мгновенно, что очень важно для военных.

Средневысотные спутники

На гораздо более низких высотах, нежели геостационарные спутники, между двумя поясами Ван Аллена, располагаются **средневысотные спутники (MEO**, Medium-Earth Orbite Satellites). Если смотреть на них с Земли, то будет заметно их медленное дрейфование по небосводу. Средневысотные спутники делают полный оборот вокруг нашей планеты примерно за 6 часов. Соответственно, наземным приемопередатчикам необходимо следить за их перемещением. Поскольку эти спутники находятся гораздо ниже, чем геостационарные, то и «засвечиваемое» ими пятно на поверхности Земли имеет более скромные размеры. Зато для связи с ними требуются менее мощные передатчики. Спутники MEO не используются в телекоммуникациях[1], поэтому в дальнейшем мы не будем их рассматривать. Примерами средневысотных спутников являются 24 спутника системы **GPS** (Global Positioning System, глобальная система определения местонахождения), вращающихся вокруг Земли на высоте около 18 000 км.

Низкоорбитальные спутники

Снизим высоту еще больше и перейдем к рассмотрению **низкоорбитальных спутников (LEO**, Low-Earth Orbite Satellites). Для того чтобы создать целостную систему, охватывающую весь земной шар, нужно большое количество таких спутников. Причиной тому является, прежде всего, высокая скорость их движения по орбите. С другой стороны, благодаря относительно небольшому расстоя-

[1] В настоящее время средневысотные спутники находят все большее применение в телекоммуникациях, особенно в сотовой телефонной связи. — *Примеч. перев.*

нию между наземными передатчиками и спутниками не требуется особо мощных наземных передатчиков, а задержки составляют всего лишь несколько миллисекунд. В этом разделе мы рассмотрим три примера спутников LEO, два из которых относятся к голосовой связи, а один — к службам Интернета.

Iridium

Как уже было сказано ранее, в течение первых 30 лет существования спутников связи низкоорбитальные спутники использовались очень мало, поскольку они появлялись и исчезали из зоны видимости передатчика слишком быстро. В 1990 году фирма Motorola совершила большой прорыв в этой области, попросив FCC разрешить ей запустить 77 спутников связи для нового проекта **Iridium** (77-м элементом таблицы Менделеева является иридий). Впрочем, планы вскоре изменились, и было решено использовать только 66 спутников, поэтому проект следовало бы переименовать в Dysprosium[1], но это было бы менее благозвучно. Идея состояла в том, что на место исчезающего из вида спутника будет тотчас приходить следующий, этакая карусель. Предложение породило новую волну безумной конкуренции среди коммуникационных компаний. Каждая из них захотела «повесить» в небе свою цепочку низкоорбитальных спутников.

После семи лет притирки компаний друг к другу и решения вопросов финансирования в 1997 году совместными усилиями удалось, наконец, запустить спутники. Услуги связи начали предоставляться с ноября 1998 года. К сожалению, коммерческий спрос на большие и тяжелые телефоны спутниковой связи оказался незначительным, потому что за семь лет конкурентной борьбы, которые прошли до запуска проекта Iridium, сотовая связь шагнула очень далеко вперед. В результате Iridium практически не приносил прибыли, и в августе 1999 года его пришлось объявить банкротом — это было одно из самых эффектных корпоративных фиаско в истории. Спутники, как и другое имущество (стоимостью порядка $5 миллиардов), были проданы инвестору за $25 миллионов в качестве своего рода космического гаража. Проект Iridium был вновь запущен в марте 2001 года.

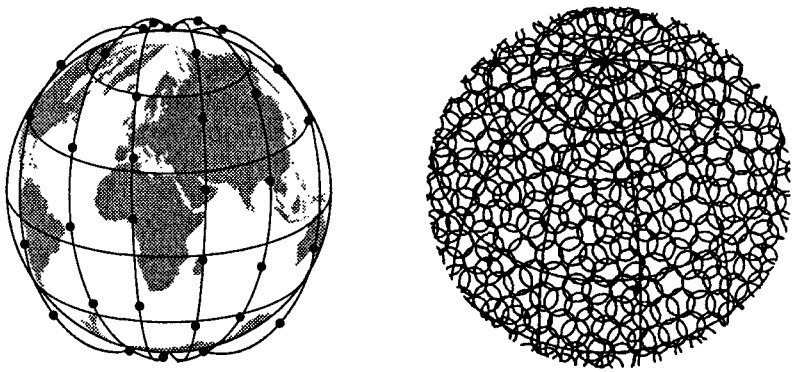
Эта система предоставляла (и предоставляет) связь с любой точкой земного шара при помощи ручных устройств, связывающихся напрямую со спутниками. Можно передавать речь, данные, факсы, информацию для пейджеров, а также навигационную информацию. И все это работает и на суше, и на море, и в воздухе! Основными клиентами Iridium являются судоходные, авиационные компании, фирмы, занимающиеся поиском нефти, а также частные лица, путешествующие в местах, где отсутствует телекоммуникационная инфраструктура (например, пустыни, джунгли, некоторые страны третьего мира).

Спутники Iridium вращаются по околоземной круговой полярной орбите на высоте 750 км. Они составляют ожерелье, ориентированное вдоль линий долготы (по одному спутнику на 32° долготы). Шесть таких ожерелий опоясывают Землю, как показано на рис. 2.15. Люди, которые не очень искушены в химической науке, могут представить себе всю эту систему в виде огромного атома диспрозия с Землей в качестве ядра и спутниками в качестве электронов.

[1] Диспрозий — 66-й элемент таблицы Менделеева. — *Примеч. перев.*

Каждый спутник имеет до 48 ячеек (пятен от лучей сигналов). Итого всю поверхность Земли, наподобие пчелиных сот, покрывают 1628 ячеек, как показано на рис. 2.15, *б*. На один спутник приходится 3840 каналов связи; соответственно, на все спутники — 253 440. Некоторые каналы используются пейджинговыми компаниями и для навигации, остальные — для передачи данных и речи.

Интересным свойством Iridium является то, что эта система обеспечивает пересылку данных между очень удаленными друг от друга абонентами путем передачи сигнала по цепочке от одного спутника к другому. Представьте себе двух человек, один из которых стоит на Северном полюсе, другой — на Южном. Они могут спокойно разговаривать друг с другом, при этом данные будут передаваться по «ожерелью» из спутников.



а *б*

Рис. 2.15. Шесть ожерелий Земли из спутников Iridium (*а*); 1628 движущихся ячеек охватываают всю Землю (*б*)

Globalstar

Альтернативой проекту Iridium является система **Globalstar**. Он построен на 48 низкоорбитальных спутниках, но имеет иную схему ретрансляции сигналов. Если в Iridium в качестве маршрутизаторов используются сами спутники, передающие по цепочке сигнал (что требует наличия на них довольно сложного оборудования), то в Globalstar применяется обычный принцип «узкой трубы». Допустим, звонок приходит на спутник с Северного полюса. Принятый сигнал отправляется обратно на Землю и захватывается крупной наземной приемно-передающей станцией рядом с домиком Санта-Клауса. Маршрутизация производится между такими станциями, разбросанными по всему миру. Наземная цель сигнала — ближайший ко второму абоненту наземный маршрутизатор. Через находящийся рядом с ним спутник вызов поступает к абоненту. Преимуществом такой схемы является то, что наиболее сложное оборудование устанавливается на поверхности Земли, а здесь работать с ним гораздо проще, чем на орбите. К тому же использование мощных наземных антенн позволяет принимать слабый сигнал со спутника; значит, можно уменьшить потребную мощность телефонов. В результате телефоны передают сигналы с мощностью всего несколько милливатт, и наземные антенны получают очень слабый сигнал даже после его усиления спутником. Тем не менее такой мощности хватает для нормальной работы.

Teledesic

Проект Iridium был рассчитан на абонентов, находящихся в различных нетривиальных местах. Наш следующий пример — проект Teledesic — предназначен для пользователей Интернета по всему миру, которым требуется высокая пропускная способность канала. Крестными отцами этой системы в 1990 году стали Крейг МакКоу (Craig McCaw), пионер мобильной связи, и Билл Гейтс (Bill Gates), всемирно известный основатель фирмы Microsoft, — он был очень недоволен улиточной скоростью, с которой телефонные компании предоставляли якобы высокую пропускную способность. Целью Teledesic было обеспечить миллионы пользователей Интернета спутниковым каналом связи со скоростью 100 Мбит/с и передачей данных в направлении спутник – Земля со скоростью до 720 Мбит/с. Для этого нужна небольшая стационарная антенна типа VSAT, полностью независимая от телефонной системы. Понятно, что телефонным операторам такая система невыгодна. При здоровой рыночной экономике это должно приводить к здоровой конкуренции.

Изначально система предполагала размещение на низковысотной орбите 288 спутников с малым следом на поверхности Земли, расположенных в 12 плоскостях прямо под нижним поясом Ван Аллена, на высоте 1350 км. Позднее было решено изменить схему, и стало 30 спутников с увеличенным следом на поверхности. Передача должна осуществляться в высокочастотном и еще не переполненном диапазоне с широкой полосой — Ka. Teledesic представляет собой космическую систему с коммутацией пакетов, при этом каждый спутник является маршрутизатором и может пересылать данные на соседние спутники. Когда пользователь запрашивает полосу для передачи данных, она предоставляется ему динамически на 50 мс. Систему предполагается запустить в 2005 году.

Спутники против оптоволокна

Такое сравнение не только уместно, но и поучительно. Всего лишь 20 лет назад люди смогли осознать, что будущее телекоммуникационных систем — за спутниками связи. В конце концов, телефонная система не особо менялась последние 100 лет; похоже, что не изменится и еще через 100 лет. Такая стабильность вызвана в том числе и мощной регулятивной средой, которая обязывала телефонные компании предоставлять качественный сервис за разумные деньги и взамен предлагала гарантированную прибыль за счет инвестиций. Для тех, кому требовалось передавать не только речь, но и данные, сделали модемы на 1200 бит/с. Собственно, это все, что долгое время предоставляла телефонная система.

В 1984 году в США и чуть позднее в Европе стала возникать конкурентная борьба в области связи, которая все поставила с ног на голову. Телефонные компании занялись прокладкой оптического волокна для междугородной телефонии и стали предоставлять услуги высокоскоростного доступа в Интернет, например, по ADSL (Asymmetric Digital Subscriber Line, асимметричная цифровая абонентская линия). Наконец-то стали снижаться искусственно завышенные тарифы на дальнюю связь, за счет которых долгое время удерживались низкие тарифы на местные переговоры.

Довольно неожиданно оптоволоконные кабели стали победителями среди средств связи. Тем не менее, у спутников имеются свои области применения, в которых оптоволокно, увы, бессильно. Рассмотрим некоторые из этих областей.

Во-первых, несмотря на то что у отдельно взятого оптического волокна пропускная способность выше, чем у всех спутников вместе взятых, большинству пользователей это мало что дает. Пока что оптоволоконные кабели используются в основном в телефонных сетях для обеспечения большого количества одновременных звонков, а частному сектору такие технологии малодоступны. При применении спутниковой системы достаточно установить антенну на крыше дома, и пользователь получит очень неплохую пропускную способность линии, никак не связанной с телефонной сетью. Эту идею использует, например, Teledesic.

Второй областью применения спутниковой связи является мобильная телефония. Очень многие люди испытывают потребность в том, чтобы быть постоянно на связи — во время путешествий, за рулем автомобиля, во время авиаперелетов и морских круизов. Оптоволокно в этих ситуациях протянуть невозможно, а вот спутниковая связь будет работать без проблем. Тем не менее, возможно, оптимальным вариантом является все-таки сочетание сотовой и оптоволоконной связи (за исключением случаев морских вояжей и авиаперелетов).

Третья область касается вопросов, в которых принципиально широковещание. Сообщение, отправленное через спутник, могут получить одновременно тысячи абонентов на Земле. Например, постоянно меняющуюся информацию о ситуации на мировых биржах проще и дешевле распространять среди огромного количества пользователей с помощью спутника, а не наземной эмуляции широковещания.

В-четвертых, нельзя забывать о местах, куда либо очень тяжело протянуть кабель, либо этого не позволяют сделать скудные средства местных бюджетов. В таких регионах обычно плохо развита наземная инфраструктура. Поэтому, например, Индонезия имеет спутник для внутреннего телефонного трафика. Приобрести его оказалось дешевле, чем проложить донный кабель между 13 667 островами.

В-пятых, спутниковая связь может быть использована там, где очень тяжело или необоснованно дорого обходится приобретение права на прокладку кабеля.

Шестой областью применения спутников является система, для которой критична скорость развертывания техники. Это, конечно, военная система.

В целом, основным средством телекоммуникаций на Земле, вероятно, будет комбинация оптоволокна и сотовой радиосвязи, но для некоторых специальных применений будет использоваться спутниковая система. Однако есть одно «но», которое может приостановить развитие всего этого: экономика. Хотя оптоволоконные кабели обладают очень высокой пропускной способностью, беспроводные системы, как наземные, так и спутниковые, будут вести очень жесткую политику ценовой конкуренции. Если будет продолжаться удешевление спутниковых систем (скажем, шаттлы скоро будут способны выводить на орбиту одновременно десятки спутников связи), а низкоорбитальные спутники постепенно будут все больше использоваться в телекоммуникациях, то не исключено, что оптоволоконные сети уйдут с ведущих ролей на большинстве рынков.

Коммутируемая телефонная сеть общего пользования

Когда между двумя компьютерами, принадлежащими одной компании и расположенными недалеко друг от друга, необходимо установить связь, часто проще

всего оказывается проложить между ними кабель. Подобным образом работают локальные сети. Однако когда расстояния велики, или компьютеров очень много, или кабель надо прокладывать поперек шоссе или еще какой-либо государственной магистрали, цена прямого кабельного соединения становится недоступно высокой. Кроме того, почти во всех странах мира законом запрещено протягивать частные линии связи над или под объектами государственной собственности. Поэтому проектировщики сетей должны рассчитывать на имеющиеся средства телекоммуникации.

Подобные средства связи, в частности, **коммутируемая телефонная сеть общего пользования** (PSTN, Public Switched Telephone Network), были созданы много лет назад с совершенно иной целью — передать человеческий голос в более или менее узнаваемом виде. Их применимость для соединения друг с другом компьютеров весьма незначительна, однако ситуация очень быстро меняется с внедрением оптоволоконной связи и цифровых технологий. В любом случае, телефонная система так тесно переплетена с компьютерными сетями (глобальными), что ее изучению стоит посвятить несколько разделов.

Чтобы понять, насколько значимой является телефонная сеть, проведем грубое, но показательное сравнение свойств типичного межкомпьютерного соединения при помощи кабеля и через телефонную линию. Итак, кабель, соединяющий два компьютера, может передавать данные со скоростью 10^9 бит/с, возможно, чуть больше. Сравним с соединением по телефонной линии с помощью модема, где скорость передачи ограничена 56 Кбит/с. Отличие — ни много ни мало — в 20 000 раз. С тем же успехом можно было бы сравнить какую-нибудь утку, лениво прогуливающуюся по лужайке, и ракету, летящую на Луну. Если вместо обычной телефонной линии установить ADSL, скорости все равно будут различаться в 1000–2000 раз.

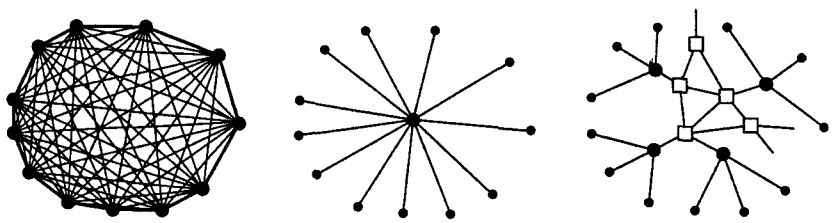
Беда в том, что разработчики компьютерных систем привыкли, как ни странно, иметь дело с компьютерными системами, поэтому когда они вдруг сталкиваются с иной системой, производительность которой (с их точки зрения) на 3–4 порядка ниже, то они посвящают массу времени и сил попыткам решить, как использовать эту систему более эффективно. В следующих разделах мы опишем телефонную систему и покажем, как она устроена и как работает. Дополнительные сведения о внутренностях телефонной системы см. (Bellamy, 2000).

Структура телефонной системы

Вскоре после того как Александр Грэхем Белл (Alexander Graham Bell) в 1876 году (всего на несколько часов раньше своего конкурента, Элиши Грея (Elisha Gray)) запатентовал телефон, на его изобретение появился огромный спрос. Вначале все было довольно забавно: этот рыночный сектор занимался торговлей телефонными аппаратами, которые продавались парами. Задача протягивания между ними единственного провода была возложена на покупателя. Вместо второго провода использовалась земля. Если владелец телефона хотел поговорить с n другими владельцами телефонов, ему приходилось протягивать отдельные провода ко всем n домам. За первый год существования такой телефонной сети города оказались опутанными настоящей сетью из проводов, тянущихся над домами и деревьями в

полнейшем беспорядке. Стало очевидно, что модель соединения телефонов «каждый с каждым» работать не будет (рис. 2.16, *а*).

К чести Белла, он заметил это и основал телефонную компанию Bell Telephone Company, открывшую свой первый офис в 1878 году в Нью-Хэйвене, штат Коннектикут. Компания прокладывала провод к каждому дому или офису пользователя. Чтобы позвонить, пользователь должен был покрутить ручку телефона, при этом в офисе телефонной компании звенел звонок, привлекающий внимание оператора, который вручную соединял звонившего с требуемым номером, втыкая разъем в нужное гнездо. Структура телефонной сети с одним коммутатором изображена на рис. 2.16, *б*.



а *б* *а*

Рис. 2.16. Сеть «каждый с каждым» (*а*); централизованный коммутатор (*б*); двухуроаневая иерархия (*а*)

Довольно скоро подобные офисы компании Bell System стали появляться повсюду, кроме того, возник спрос на междугородную связь, поэтому Bell System стала соединять свои офисы. Вскоре они столкнулись опять с той же проблемой: задача соединения каждого офиса с каждым очень быстро стала невыполнимой, поэтому были созданы офисы второго уровня (рис. 2.16, *в*). Через некоторое время количество офисов второго уровня также стало слишком большим. В конце концов иерархия разрослась до пяти уровней.

К 1890 году были созданы три основные части телефонной системы — коммутаторные телефонные станции, провода, соединяющие пользователей с ними (теперь уже изолированные витые пары, а не проволока с землей вместо второго провода), и линии междугородной связи, соединяющие отдельные телефонные станции. И хотя в каждой из этих областей с тех пор производились улучшения, основа модели Bell System осталась неизменной на протяжении более 100 лет. Краткую техническую историю телефонной системы см. (Hawley, 1991).

Вплоть до 1984 году, когда распалась корпорация AT&T, телефонная система представляла собой многоуровневую иерархическую структуру с высокой степенью избыточности. Приведенное далее описание сильно упрощено; тем не менее, оно передает суть дела. Каждый телефон соединен при помощи двух медных проводов с ближайшей **оконечной телефонной станцией**. Расстояние от телефона до ближайшего коммутатора обычно от 1 до 10 км, в городах меньше, чем в сельской местности.

В одних только Соединенных Штатах насчитывается около 22 000 оконечных телефонных станций. Двухпроводное соединение между телефоном каждого або-

нента и оконечной телефонной станцией называется **местной линией связи** (или локальным контуром). Если местные линии связи всего мира соединить последовательно в одну линию, то их можно будет протянуть до Луны и обратно 1000 раз.

Одно время 80 % капитала компании AT&T было вложено в медные провода местных линий связи. Таким образом, компания AT&T представляла собой самую большую в мире медную шахту. К счастью, этот факт не был широко известен в сообществе инвесторов. В противном случае какие-нибудь корпорации могли скупить AT&T, выкопать все провода и продать их меднообогатительным комбинатам с целью получения быстрой прибыли, тем самым прекратив всю телефонную связь в США.

Если абонент, подключенный к оконечному коммутатору, позвонит другому абоненту, подключенному к тому же коммутатору, то коммутирующий механизм установит между ними прямое электрическое соединение. Это соединение будет сохраняться в течение всего разговора.

Если же абоненты подключены к разным оконечным станциям, то должна использоваться другая процедура. У каждой оконечной станции имеется несколько линий к одному или нескольким коммутационным центрам, называемым **пригородно-междугородными станциями** (или, если они расположены в одной области, **транзитными станциями**). Соединяющие их линии называются **междугородными**. Если оба абонента подключены к одной и той же междугородной станции (что вполне вероятно, если они находятся недалеко друг от друга), то связь может быть установлена этой междугородной станцией. На рис. 2.16, *в* показана телефонная сеть, состоящая только из телефонных аппаратов (маленькие точки), оконечных коммутаторов (большие точки) и междугородных станций (квадраты).

Если у абонентов нет общей междугородной станции, то связь между ними будет установлена на более высоком иерархическом уровне. Междугородные станции объединены в сеть, состоящую из первичных, секционных и региональных коммутаторов. Все эти станции связываются друг с другом высокоскоростными **межстанционными линиями**. Число разного рода коммутационных центров, как и их топология (например, могут ли два секционных коммутатора связаться напрямую или они должны устанавливать связь через региональный коммутатор?), различается в разных странах и зависит от плотности расстановки телефонов на определенной территории. На рис. 2.17 изображено, как может быть установлен маршрут связи при среднем расстоянии между абонентами.

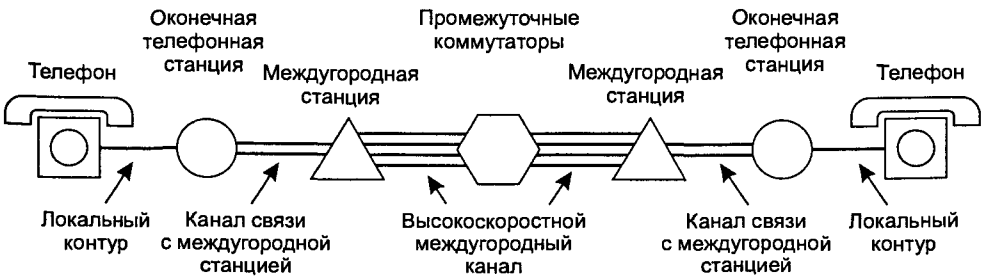


Рис. 2.17. Типичный маршрут связи при средней дистанции между абонентами

В телекоммуникациях применяется широкий спектр сред передачи данных. Местные линии связи строятся из витых пар категории 3, хотя на заре телефонии обычно использовались голые провода, располагавшиеся на расстоянии 25 см друг от друга. Для связи между коммутаторами широко используются коаксиальные кабели, микроволновая связь, все популярнее становятся оптоволоконные кабели.

В прошлом телефонная система была аналоговой, в виде электрического напряжения передавался сам голосовой сигнал. С появлением оптического волокна, цифровой электроники и компьютеров стала возможной цифровая передача сигнала на всех уровнях иерархии, кроме местных линий связи — последнего аналогового звена цепи. Цифровая связь предпочтительней потому, что нет необходимости в точности воспроизводить форму аналогового сигнала после того, как он проходит через множество коммутаторов и усилителей. Достаточно распознать лишь одно из двух состояний линии: 0 или 1. Это свойство делает цифровую передачу более надежной, чем аналоговая. К тому же она дешевле и проще в обслуживании.

В целом, телефонная система состоит из следующих трех компонентов.

1. Местные линии связи (аналоговые витые пары, подводящиеся в дома и офисы).
2. Магистральные каналы (цифровая связь на базе оптоволокна между коммутационными станциями).
3. Коммутационные станции (в них вызовы переадресуются с одних магистралей на другие).

После того как мы коснемся темы политики телефонии, мы вернемся к более подробному рассмотрению каждого из этих трех компонентов. Местные линии связи — это то, что соединяет каждого конкретного абонента со всей остальной системой, поэтому этот пункт чрезвычайно важен. Междугородным магистралям присуща фундаментальная проблема соединения множества вызовов в один и отправки его по единому кабелю. Это называется *уплотнением*, или мультиплексированием канала. Мы изучим три стратегии мультиплексирования. Наконец, есть два принципиально разных способа коммутации, которые мы также рассмотрим далее.

Политика телефонии

На протяжении десятков лет, вплоть до 1984 года, корпорация Bell System предоставляла услуги в области как локальной, так и междугородной связи на большей части Соединенных Штатов. В 1970-х годах правительство США пришло к мнению, что такая монополия является незаконной, и подала судебный иск на компанию AT&T. Правительство выиграло судебный процесс, и 1 января 1984 году компания AT&T была разделена на AT&T Long Lines, 23 местных телефонных компании **BOC** (Bell Operating Company) и еще несколько компаний. 23 компании BOC были объединены в 7 региональных RBOC для повышения их экономической жизнеспособности. По решению суда (это не было актом Конгресса США) за одну ночь вся структура телекоммуникаций в США была изменена.

Детали этой процедуры были описаны в документе под названием **MFJ**. (Modified Final Judgement — измененное окончательное судебное решение. Своего рода оксюморон: если решение могло быть изменено, значит, оно никак не окончательное.) Это событие привело к повышению конкуренции, снижению цен на дальнюю связь как для частных, так и для корпоративных абонентов. Однако одновременно с этим возросли цены на местную связь, поскольку был нарушен экономический баланс: раньше местные тарифы удерживались низкими именно за счет высоких междугородных тарифов. Во многих других странах сейчас рассматривается вопрос о конкуренции между аналогичными линиями.

Чтобы стало понятно, кто чем теперь должен был заниматься, Соединенные Штаты были разделены на 164 области, называемые **LATA** (Local Access and Transport Area — область локального доступа и транспорта). Области LATA приблизительно совпадали с областями, имеющими один и тот же телефонный код. В пределах LATA имелся один локальный оператор связи **LEC** (Local Exchange Carrier — местная телекоммуникационная компания), обладавший монополией на традиционные телефонные услуги в пределах области LATA. Наиболее важными операторами связи LEC являются компании BOC, хотя в некоторых областях LATA в роли LEC выступает одна из независимых телефонных компаний, общее число которых превышает 1500.

Связь между областями LATA поддерживалась оператором линии дальней связи **IXC** (IntereXchange Carrier). Изначально компания AT&T Long Lines была единственным крупным оператором линии дальней связи, однако сегодня корпорации WorldCom и Sprint составляют ей серьезную конкуренцию. Одной из главных проблем при разделении корпорации AT&T было гарантировать, что все владельцы линий дальней связи обеспечат одинаковые качество связи, тарифы и количество цифр в номерах телефонов. Общий вид структуры изображен на рис. 2.18. На нем изображено три примера областей LATA с несколькими оконечными телефонными станциями в каждой из них. Области LATA 2 и 3 включают в себя также небольшую иерархическую структуру с транзитными станциями (внутренними для LATA междугородными станциями).

Любой владелец линий дальней связи может создать свой коммутатор, называемый **POP** (Point of Presence — точка присутствия), в области LATA для обработки звонков, исходящих из этой области. Подразумевается, что местный оператор связи LEC соединит каждого владельца линий дальней связи с каждой оконечной станцией либо напрямую, как в случае LATA 1 и 3 (рис. 2.18), либо через транзитные коммутаторы, как в случае LATA 2. Кроме того, условия соединения, как технические, так и финансовые, должны быть идентичными для каждого владельца линий дальней связи. В этом случае абонент, скажем, области LATA 1 сможет выбирать себе владельца линий дальней связи для звонков, например, в область LATA 3.

Одним из требований документа MFJ было запрещение владельцам линий дальней связи предоставлять услуги в области локальной связи, а операторам связи — в области междугородной связи. Однако никаких других ограничений наложено не было, и все эти компании могли, например, торговать жареными цыплятами и т. п. В 1984 году такой закон звучал вполне недвусмысленно. Однако развивающиеся технологии сыграли довольно злую шутку с законом, в ре-

зультате чего он оказался устаревшим. Дело в том, что ни кабельное телевидение, ни сотовая телефонная связь не фигурировали в решении суда. По мере того как кабельное телевидение из одностороннего превращалось в двустороннее, а сотовые телефоны становились все популярнее, как локальные операторы связи, так и владельцы линий дальней связи начали сливаться с операторами кабельной и сотовой связи или скупать их.

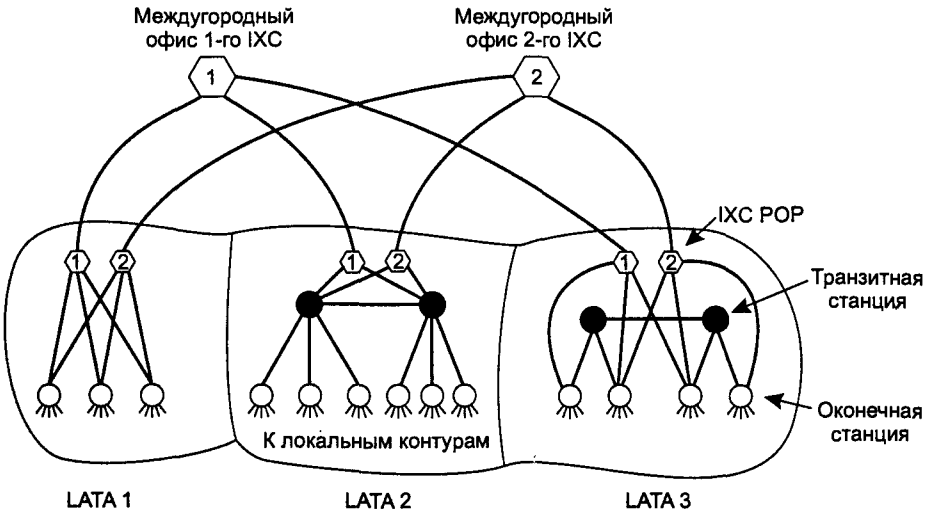


Рис. 2.18. Взаимоотношения между LATA, LEC и IXC. Кружками обозначены коммутаторы LEC. Шестиугольники принадлежат IXC, номер которых указан внутри

В 1995 году Конгресс США счел невозможным дальнейшие попытки поддерживать различия между компаниями различного типа и издал проект закона, разрешающий компаниям, занимающимся кабельным телевидением, местным телефонным компаниям, операторам дальней связи и сотовым операторам предоставлять любые услуги этих смежных областей связи. Идея заключалась в том, чтобы позволить компаниям предоставлять клиентам единый интегрированный пакет услуг, включающий в себя кабельное телевидение, телефон и информационные сервисы. Таким образом, компании смогли бы соревноваться в качестве и цене услуг. В феврале 1996 года законопроект стал законом, в результате чего многие BOC превратились в IXC, а компании, занимавшиеся, например, кабельным телевидением, также стали предоставлять услуги местной телефонной связи, соперничая с LEC.

Одно интересное предписание можно найти в законе 1996 года: операторы местной связи должны были реализовать переносимость локальных номеров телефона. Это означало, что абонент мог сменить оператора, не меняя своего номера телефона. Именно необходимость смены номера была причиной, по которой многие клиенты оставались верными своей телефонной компании. Новое требование было направлено на повышение качества услуг за счет роста конкуренции между операторами местной связи. В результате телекоммуникационный ландшафт США вновь подвергся значительной реструктуризации. Этому примеру снова последовали многие страны. Часто так и происходит: мировое сообщество

наблюдает за экспериментами, проводящимися в Соединенных Штатах, и если их результаты оказываются положительными, то многие страны перенимают опыт. Если же результаты оставляют желать лучшего, то каждая страна пробует сделать что-то свое.

Местные линии связи: модемы, ADSL, беспроводная связь

Пришло время вплотную заняться изучением принципов работы телефонной системы. Основные ее части изображены на рис. 2.19. Мы видим здесь местные линии, магистрали, а также междугородные и местные коммутационные станции, на каждой из которых установлено оборудование, осуществляющее маршрутизацию звонков. Оконечные телефонные станции в США и других крупных странах рассчитаны на 10 000 местных линий. И в самом деле, до недавних пор код региона вместе с кодом местной телефонной станции давал доступ к порядковому номеру абонента на этой станции. Например, (212) 601-xxxx дает доступ к телефонной станции 601, находящейся в регионе 212, к которой может быть подключено до 10 000 абонентов с номерами от 0000 до 9999. Однако по мере роста конкуренции в области локальной связи такая система становилась все менее приемлемой, поскольку слишком много компаний изъявляли желание получить собственный код оконечной станции. Чем больше становился код, тем сложнее становились схемы маршрутизации, которые должны были обеспечить нормальную работу системы.

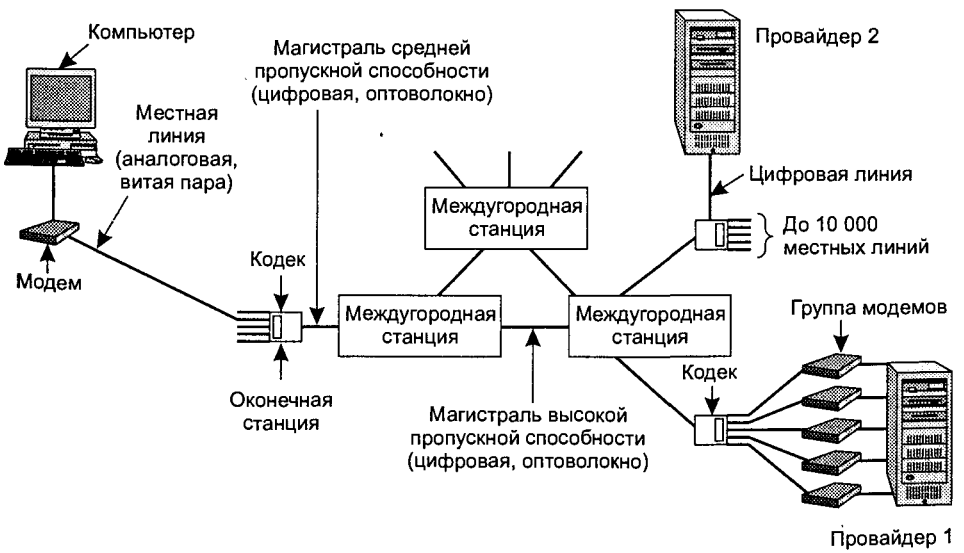


Рис. 2.19. Одновременное использование аналоговой и цифровой связи для соединения компьютеров. Преобразования осуществляются модемами и кодеками

Начнем с той части телефонной системы, с которой большинство людей знакомы очень хорошо. Итак, имеется двухпроводная линия, идущая от оконечной телефонной станции в дома и небольшие организации. Эта часть называется иног-

да **последней милей**, хотя длина местной линии на местности может составлять и несколько миль. На этом отрезке вот уже более 100 лет используется аналоговая связь, и похоже, что в ближайшие *n* лет ситуация не изменится (в основном из-за высокой стоимости перехода на цифровые линии). Тем не менее даже в этом последнем оплоте аналоговой связи мало-помалу происходят изменения. В данном разделе мы рассмотрим как традиционный подход к построению местных линий, так и новые тенденции, наблюдающиеся в этой области. Особое внимание уделим обмену данными при помощи домашних компьютеров.

Если компьютер желает отправить цифровые данные по аналоговой телефонной линии, то эти данные должны быть вначале преобразованы в аналоговые для передачи по местной линии. Преобразованием занимается устройство под названием **модем**, который мы вкратце изучим чуть позже. На оконечной станции телефонной компании данные вновь преобразуются в цифровые и отправляются по магистральному каналу.

Если на противоположном конце линии находится компьютер с модемом, то необходимо снова преобразовать сигнал из цифрового в аналоговый, чтобы он смог преодолеть «последнюю милю» и дойти до места назначения. Подобное решение изображено на рис. 2.19: у Провайдера 1 имеется набор модемов, каждый из которых подключен к своей местной линии связи. Этот провайдер может обрабатывать одновременно столько звонков, сколько модемов установлено на его сервере (предполагается, что вычислительных мощностей ему при этом хватает). Такой принцип работал до появления модемов со скоростью 56 Кбит/с. Скоро мы узнаем, что произошло дальше и почему.

Аналоговый сигнал представляет собой меняющееся во времени напряжение, с помощью которого и передается поток данных. Если бы среда передачи была идеальной, приемник получал бы сигнал, в точности повторяющий исходный. Но, к сожалению, таких сред в природе не существует, поэтому приходящий сигнал всегда несколько искажен относительно сигнала, передаваемого в линию отправителем. Если при этом речь идет о цифровых данных, то такие искажения могут привести к серьезным ошибкам.

Линии передачи всегда страдают от трех напастей — затухания, искажения из-за задержек, а также шума. Затухание, или **ослабление, сигнала** — это потеря энергии сигналом по мере его распространения по каналу. Затухание выражается в децибелах на километр. Степень ослабления сигнала зависит от его частоты. Чтобы понять степень этой частотной зависимости, представьте себе сигнал в виде суммы гармоник ряда Фурье. Каждая гармоника ослабляется в различной степени, в результате чего приемник получает искаженный спектр сигнала.

Более того, разные гармоники ряда Фурье распространяются в среде передачи данных с разными скоростями. Это ведет к **искажению сигнала**, получаемого приемником.

Еще одной проблемой является **шум**, то есть нежелательная энергия от посторонних источников, примешивающаяся к энергии передаваемого сигнала. Термальный (тепловой) шум в электрическом проводе присутствует всегда — он вызван случайным тепловым движением электронов, и от него избавиться невозможно. **Перекрестные помехи** вызваны индукцией, возникающей между двумя близко расположенными проводами. Иногда во время телефонного разговора мож-

но услышать какой-то чужой разговор. Это и есть перекрестная помеха. Наконец, существует **импульсный шум**, вызванный скачками напряжения и другими случайными причинами. В случае цифровых данных импульсный шум может повредить несколько бит передаваемых данных.

Модемы

Поскольку, как было отмечено ранее, ослабление и скорость распространения сигнала зависят от частоты, то было бы очень нежелательно иметь широкий спектр частот передаваемого сигнала. К сожалению, последовательности прямоугольных импульсов, соответствующие цифровому сигналу, имеют широкий спектр частот, следовательно, подвергаются значительному затуханию и искажению. Эти эффекты делают невозможной передачу в исходном диапазоне (при постоянном токе). Исключение может составлять только передача на малое расстояние при невысокой скорости.

Для решения этой проблемы вместо постоянного тока для передачи данных, особенно по телефонным линиям, применяется переменный ток. Непрерывный сигнал на частоте от 1000 до 2000 Гц называется **синусоидальной несущей частотой**. Амплитуда, частота и фаза несущей могут изменяться (модулироваться) для передачи информации. При **амплитудной модуляции** используются две различные амплитуды сигнала, соответствующие значениям нуля и единицы. При **частотной модуляции**, называемой также **частотной манипуляцией** (термин «манипуляция» широко используется в качестве синонима «модуляции»), для передачи цифрового сигнала используется несколько различных частот. При простейшей **фазовой модуляции** применяется сдвиг фазы несущей частоты на 180° через определенные интервалы времени. Улучшенным вариантом фазовой модуляции является сдвиг фазы на постоянный угол, например, на 45, 135, 225 или 315° для передачи 2 бит информации за один временной интервал. Можно также изменять фазу по окончании интервалов, что позволяет приемнику более четко распознать их границы.

На рис. 2.20 показаны три формы модуляции. Рисунок 2.20, *б* дает представление о том, как выглядит исходный сигнал при амплитудной модуляции. Амплитуда его либо ненулевая, либо равна нулю. На рис. 2.20, *в* показано, что тот же исходный сигнал кодируется двумя разными частотами. Наконец, из рис. 2.20, *г* видно, что два состояния кодируются наличием либо отсутствием фазового сдвига на границе каждого бита.

Устройство, принимающее последовательный поток битов и преобразующее его в выходной сигнал, модулируемый одним или несколькими из приведенных способов, а также выполняющий обратное преобразование, называется **модемом** (сокращение от «модулятор-демодулятор»). Модем устанавливается между (цифровым) компьютером и (аналоговой) телефонной линией.

Добиться увеличения скорости простым увеличением частоты дискретизации невозможно. Теорема Найквиста утверждает, что даже при наличии идеального канала с частотой 3000 Гц (каковым телефонная линия не является) невозможно передавать отсчеты сигнала чаще, чем с частотой 6000 Гц. На практике большинство модемов делают 2400 отсчетов в секунду и стремятся не к повышению этого значения, а к повышению числа бит на отсчет.

Число отсчетов (сэмплов) в секунду измеряется в **бодах**. За каждый бод передается один **символ**. Таким образом, линия, работающая со скоростью *n* бод, передает *n* символов в секунду. Например, линия со скоростью 2400 бод отправляет 1 символ за 416,667 мкс. Если символ состоит из двух состояний линии (к примеру, 0 В означает логический ноль, 1 В означает логическую единицу), то битовая скорость составляет 2400 бит/с. Если же используются четыре уровня напряжений (например, 0, 1, 2, 3), тогда каждый символ будет состоять уже из двух бит, поэтому та же самая линия на 2400 бод сможет передавать все те же 2400 символов в секунду, но уже с битовой скоростью 4800 бит/с. Аналогично, можно задать четыре степени фазового сдвига вместо двух, тогда и модулированный сигнал будет кодировать один символ двумя битами, а битовая скорость, опять же, будет в два раза выше. Такой метод применяется очень широко и называется **квадратурной фазовой манипуляцией**, **QPSK** (Quadrature Phase Shift Keying).

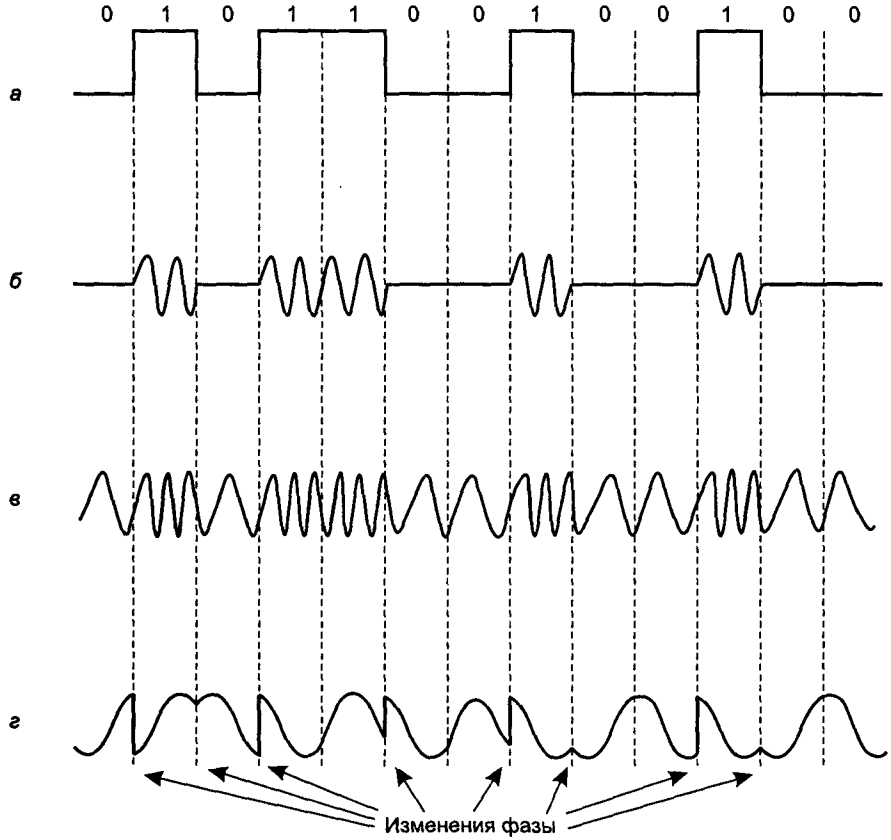


Рис. 2.20. Двоичный сигнал (*а*); амплитудная модуляция (*б*); частотная модуляция (*в*); фазовая модуляция (*г*)

Понятия полосы пропускания, бода, символа и битовой скорости часто путают, поэтому сейчас необходимо поставить все на свои места и четко определить.

Полосой пропускания среды называется диапазон частот, которые могут передаваться в этой среде с минимальным затуханием. Это физическое свойство материала. Полоса варьируется обычно от 0 до какого-то максимального значения и измеряется в герцах (Гц).

Скорость двоичной передачи, baud rate (в бодах) — это число отсчетов, совершаемых за одну секунду. Каждый отсчет передает единицу информации, то есть *символ*. Таким образом, скорость двоичной передачи равна скорости передачи символов. Метод модуляции (например, QPSK) определяет число бит, из которых состоит один символ.

Битовой скоростью называется объем информации, передаваемый по каналу за секунду. Битовая скорость равна произведению числа символов в секунду и числа бит на символ (символ/с × бит/символ).

Все хорошие модемы используют комбинированные методы модуляции сигналов для передачи максимального количества бит в одном боде. Зачастую, например, комбинируются амплитудная и фазовая модуляции. На рис. 2.21, *а* изображены точки, расположенные под углами 45, 135, 225 и 315°, с постоянным уровнем амплитуды (это видно по расстоянию до них от начала координат). Фаза этих точек равна углу, который линия, проведенная через точку и начало координат, составляет с положительным направлением горизонтальной оси. На рис. 2.21, *а* видно, что возможны четыре положения фазового сдвига, значит, можно передавать 2 бита на символ. Это метод QPSK.

На рис. 2.21, *б* изображен другой комбинированный метод модуляции, использующий 16 комбинаций амплитудных и фазовых сдвигов. С его помощью можно передавать уже 4 бита на символ. Такая схема называется **квадратурной амплитудной модуляцией**, или **QAM-16** (Quadrature Amplitude Modulation). Она может, например, использоваться для передачи 9600 бит/с по линии с пропускной способностью 2400 бод.

На рис. 2.21, *в* изображен еще один метод, комбинирующий амплитудную и фазовую модуляции. С помощью 64 комбинаций можно в один символ поместить 6 бит. Метод называется **QAM-64**). Существуют схемы QAM и более высоких порядков.

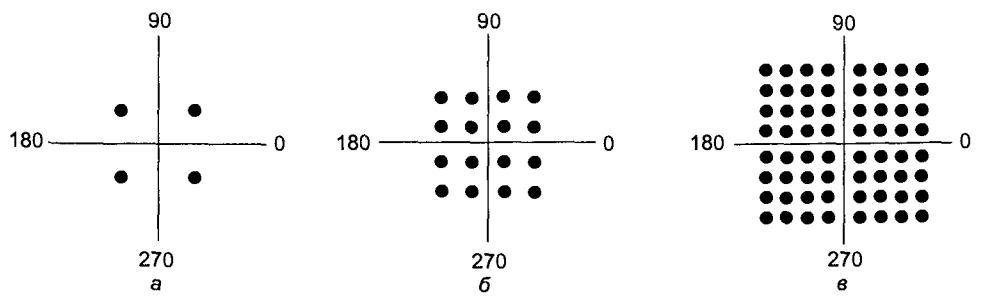


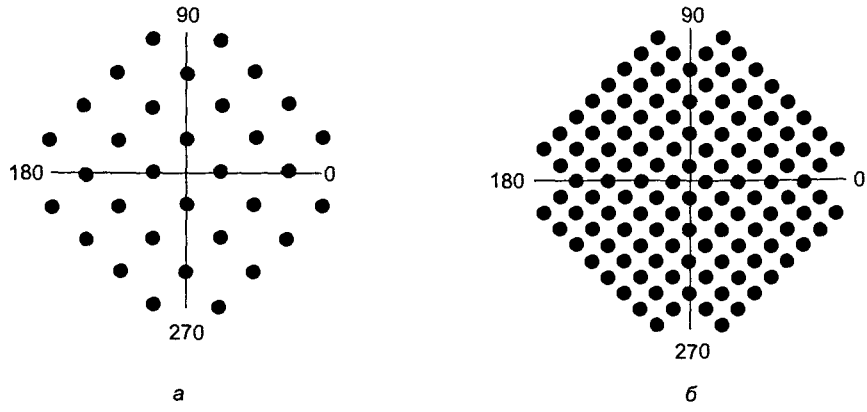
Рис. 2.21. QPSK (*а*); QAM-16 (*б*); QAM-64 (*в*)

Диаграммы, показывающие допустимые комбинации амплитуд и фаз (как на рис. 2.21), называются **амплитудно-фазовыми диаграммами** (диаграммами со-

звездий). У каждого стандарта высокоскоростных модемов есть своя диаграмма. Такой модем может общаться только с теми модемами, которые используют ту же диаграмму (хотя более быстрые модемы обычно могут эмулировать диаграммы всех более медленных модемов).

Чем больше точек находится на амплитудно-фазовой диаграмме, тем больше вероятность того, что даже слабый шум при детектировании амплитуды или фазы приведет к ошибке и порче битов. Для уменьшения этой вероятности были разработаны стандарты, подразумевающие включение в состав каждого отсчета нескольких дополнительных битов коррекции. Такие схемы называются **решетчатым кодированием**, или **TCM** (Trellis-Coded Modulation). Так, например, стандарт V.32 имеет 32 точки на диаграмме для передачи 4 бит/символ и 1 контрольный бит на линии 2400 бод, что позволяет достигнуть скорости 9600 бит/с с коррекцией ошибок. Амплитудно-фазовая диаграмма для V.32 показана на рис. 2.22, *а*. Идея «повернуть» диаграмму на 45° была вызвана некоторыми техническими соображениями; понятно, что информационная емкость «повернутых» и «не повернутых» диаграмм одна и та же.

Следующим шагом после скорости 9600 бит/с стала скорость 14 400 бит/с. Новый стандарт был назван **V.32 bis**. Такая скорость достигается передачей 6 бит данных и 1 контрольного бита на отсчет при частоте дискретизации 2400 бод. Амплитудно-фазовая диаграмма (рис. 2.22, *б*) состоит из 128 точек при использовании QAM-128. Эта скорость используется факс-модемами для передачи страниц, отсканированных в виде растровых изображений. QAM-256 не используется в обычных модемах для телефонных линий, зато применяется в кабельных сетях, как мы увидим далее.



а *б*

Рис. 2.22. V.32 для 9600 бит/с (*а*); V.32 для 14 400 бит/с (*б*)

Следующим телефонным стандартом после стандарта V.32 bis является **V.34** со скоростью 28 800 бит/с при частоте дискретизации 2400 бод и 12 битах данных на символ. Последние модемы этой серии были сделаны в соответствии со стандартом **V.34 bis** и использовали 14 бит/символ при 2400 бод, за счет чего была достигнута скорость 33 600 бит/с.

Для дальнейшего повышения скорости необходимо было прибегать к всяческим уловкам. Так, например, многие модемы используют предварительное сжатие данных, что позволяет превысить порог эффективной скорости передачи данных, составляющий 33 600 бит/с. С другой стороны, почти все модемы прослушивают линию перед началом передачи, и если обнаруживается, что ее качество недостаточно высоко, то они автоматически понижают скорость относительно своего максимума. Таким образом, *эффективная* скорость модема при его работе в реальных условиях может оказаться как ниже, так и выше официально заявленной.

Все современные модемы позволяют передавать данные одновременно в обоих направлениях (используя разные частотные диапазоны для каждого из направлений). Соединения, обладающие такой возможностью, называются **дуплексными**. Скажем, двухпутная железная дорога, по которой поезда могут перемещаться во встречных направлениях, является своеобразной дуплексной линии связи. Соединение с возможностью поочередной передачи данных в каждом из направлений называется **полудуплексным**. Примером полудуплексной линии является однопутная железная дорога. Наконец, соединение, при котором данные можно отправлять только в одном направлении, называется **симплексным**. Примером может служить дорога с односторонним движением. Еще один пример — оптоволокно с лазером на одном конце и фотоприемником на другом.

Ограничение скорости телефонных линий тридцатью пятью килобитами в секунду, обоснованное Шенноном, заставило разработчиков модемов остановиться на скорости 33 600 бит/с. Дальнейшее ускорение невозможно в силу законов термодинамики. Чтобы узнать, могут ли на самом деле модемы работать со скоростью 56 Кбит/с, следите внимательно за дальнейшими рассуждениями.

Чем объясняется теоретический предел в 35 Кбит/с? Это связано, прежде всего, со средней протяженностью местных линий связи и их качеством. Да, число 35 Кбит/с определяется длиной местных линий. На рис. 2.19 вызов, инициированный компьютером, находящимся слева, идет к Провайдеру 1 по двум местным линиям связи и имеет при этом форму аналогового сигнала. Первый раз это случается на стороне отправителя, второй раз — рядом с получателем. И там, и там на сигнал накладываются шумы. Если нам удастся избавиться каким-то образом от этих двух «последних миль», то максимальную скорость можно будет удвоить.

Именно эту идею использует Провайдер 2 (рис. 2.19). К нему от ближайшей оконечной телефонной станции протянута полностью цифровая линия. Сигналу, снимаемому с магистральной линии, не приходится иметь дело с кодеками, модемами, АЦП и ЦАП. Когда хотя бы на одном конце соединения отсутствует «последняя миля» (а большинство провайдеров сейчас уже пользуются только цифровыми каналами), максимальная скорость передачи повышается до 70 Кбит/с. Если же соединение устанавливается между двумя обычными пользователями, каждый из которых передает данные по аналоговой местной линии с помощью модема, максимальная скорость ограничена 33 600 бит/с.

Причина, по которой используются модемы со скоростью 56 Кбит/с, связана с теоремой Найквиста. Телефонная линия имеет полосу пропускания около

4000 Гц (включая защитные полосы). Таким образом, максимальное число независимых отсчетов в секунду — 8000. Число бит на отсчет, используемое в США, равно 8, причем 1 бит является контрольным, что позволяет передавать пользовательские данные с битовой скоростью 56 000 бит/с. В Европе все 8 бит являются информационными, поэтому, в принципе, максимальная скорость может достигать 64 000 бит/с, однако международным соглашением установлено ограничение в 56 000 бит/с.

Этот стандарт называется **V.90**. Он предоставляет пользователю возможность передачи данных в сторону провайдера со скоростью 33,6 Кбит/с, а в обратную сторону — со скоростью 56 Кбит/с. Причиной такого рассогласования скоростей является тот очевидный факт, что трафик от абонента обычно во много раз меньше трафика от провайдера (например, запрос веб-страницы — это всего лишь несколько байт, тогда как сама страница может иметь размеры, измеряемые мегабайтами). Теоретически, можно расширить канал от абонента и сделать его более скоростным, чем 33,6 Кбит/с, но многие линии слишком зашумлены и далеко не всегда выдерживают даже такую скорость. Поэтому было принято решение выделить основную часть полосы под поток данных от провайдера, давая ему тем самым шанс работать со скоростью 56 Кбит/с.

Следующим стандартом после V.90 стал **V.92**. Модемы V.92 могут отправлять данные со скоростью 48 Кбит/с, если на линии достаточно низкий уровень помех. Возможная скорость линии определяется в течение примерно половины обычного 30-секундного интервала, необходимого более старым модемам. Наконец, они имеют одну интересную функцию: при включенном режиме ожидания звонка входящий телефонный звонок разрывает интернет-сеанс.

Цифровые абонентские линии

Когда скорость связи по телефонным линиям достигла своего, по-видимому, конечного значения 56 Кбит/с, создалось впечатление, что развитие на этом остановится. Между тем каналы кабельного телевидения стали предлагать своим абонентам 10 Мбит/с при работе по общему кабелю, а спутниковые системы — до 50 Мбит/с. Доступ к Интернету стал неотъемлемой частью бизнеса таких операторов. И телефонные компании (прежде всего, уровня LEC) поняли, что необходимо двигаться дальше, создавая более конкурентоспособные системы. Для этого нужно было предоставить цифровые услуги конечным пользователям, используя местные линии. Системы, использующие каналы с расширенной пропускной способностью, иногда называют **широкополосными сетями**, хотя такой термин является скорее коммерческим, а не техническим.

Вскоре появилось множество предложений, носящих общее название **xDSL** (Digital Subscriber Line — цифровая абонентская линия), где вместо буквы *x* могли стоять другие буквы. Мы обсудим их, сделав основной упор на технологию, которая вскоре, похоже, станет самой популярной в этой области, — **ADSL** (Asymmetric DSL — асимметричная DSL). Поскольку окончательные стандарты для ADSL еще не утверждены, то некоторые детали, описываемые далее, со временем могут измениться, однако основной принцип останется, очевидно, неиз-

менным. Чтобы узнать больше про ADSL, читайте (Summers, 1999; Vetter и др., 2000).

Причиной, по которой модемы являются такими медленными устройствами, является то, что телефонные линии, по которым они традиционно работали, были изобретены для передачи человеческой речи, и вся система была направлена на оптимизацию именно в этом аспекте. Данные всегда оставались пасынками телефонной системы. В той точке, где заканчивается местная линия (оконечная телефонная станция), сигнал подвергается фильтрации, при которой вырезаются частоты ниже 300 Гц и выше 3400 Гц. Отсечка не является прямоугольной — оба края имеют уровень 3 дБ, поэтому полоса пропускания считается равной 4000 Гц, хотя на самом деле диапазон между двумя этими краями составляет только 3100 Гц. Таким образом, цифровым данным приходится пробираться по этому узкому каналу.

Хитрость, за счет которой работает xDSL, заключается в том, что ее абоненты подключаются к особому коммутатору, на котором отсутствует описанный ранее фильтр. Таким образом, на передачу данных отводится вся полоса пропускания местной линии. Лимитирующим фактором в этом случае становится сама физическая природа линии, а не искусственно вырезанный кусок диапазона в 3100 Гц.

К сожалению, емкость местных линий зависит от нескольких факторов, среди которых длина, толщина и собственно качество канала. График зависимости потенциально достижимой пропускной способности от длины линии приведен на рис. 2.23. Здесь предполагается, что все остальные факторы являются оптимальными (новые провода, аккуратные кабели и т. д.).

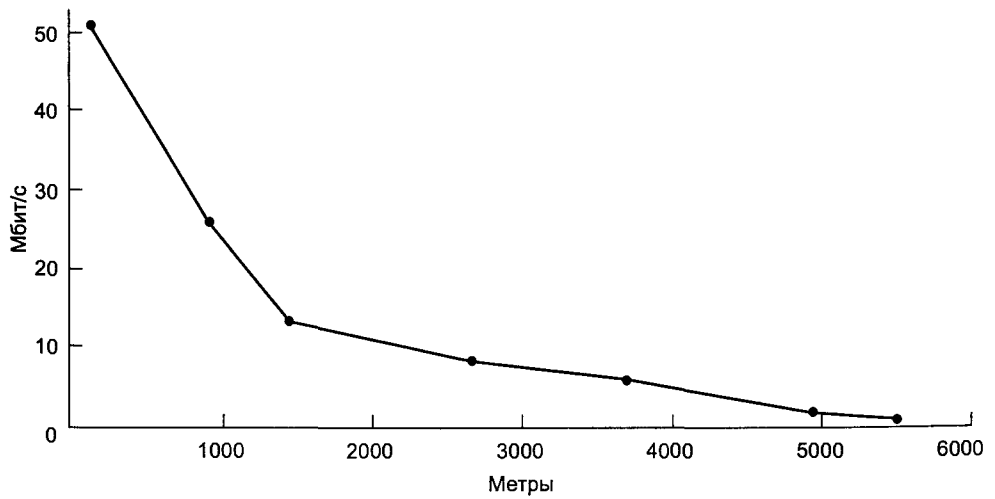


Рис. 2.23. Зависимость пропускной способности от расстояния для DSL по UTP категории 3

Реализация такой зависимости создает определенные проблемы для телефонной компании. Когда заявляется определенная скорость работы, автоматически ограничивается радиус, за пределами которого данное предложение не

может быть реализовано. Это означает, что когда приходит клиент, живущий достаточно далеко от телефонной станции, ему говорят: «Спасибо за проявленный интерес, но вы живете на 100 метров дальше от нас, чем нужно для того, чтобы стать нашим абонентом. Не могли бы вы переехать поближе?» Чем ниже предлагаемая скорость, тем больше радиус охвата и количество клиентов. Но, вместе с тем, чем ниже скорость, тем менее привлекательным выглядит предложение и тем меньше абонентов согласится выкладывать свои деньги. Это такой перекресток, на котором встречаются бизнес и технология. (Можно предложить, конечно, поставить мини-АТС в каждом микрорайоне, но это слишком дорогое решение.)

Службы xDSL разрабатывались с определенной целью. Во-первых, они должны были работать на существующих местных линиях, представляющих собой витые пары категории 3. Во-вторых, они не должны были влиять на работу аппаратуры абонента вроде телефона и факса. В-третьих, скорость работы должна была быть выше 56 Кбит/с. Наконец, в-четвертых, они должны были предоставлять постоянное подключение, и услуги при этом должны были оплачиваться только в виде фиксированной ежемесячной абонентской платы, но никак не поминутно.

Первое предложение ADSL исходило от компании AT&T и работало за счет разделения спектра местной линии, который составляет примерно 1,1 МГц, на три частотных диапазона. Вот они: диапазон обычной телефонной сети, **POTS** (Plain Old Telephone Service); исходящий диапазон (от абонента); входящий диапазон (от АТС). Технология, в которой для разных целей используются разные частоты, называется частотным уплотнением или частотным мультиплексированием. Далее мы изучим ее очень подробно. Другие операторы использовали несколько иной подход, и он нам кажется наиболее перспективным, поэтому сейчас мы его опишем.

Итак, альтернативный метод под названием **дискретная мультитональная модуляция, DMT** (Discrete MultiTone), иллюстрируется на рис. 2.24. Суть его состоит в разделении всего спектра местной линии шириной 1,1 МГц на 256 независимых каналов по 4312,5 Гц в каждом. Канал 0 — это POTS. Каналы с 1 по 5 не используются, чтобы голосовой сигнал не имел возможности интерферировать с информационным. Из оставшихся 250 каналов один занят контролем передачи в сторону провайдера, один — в сторону пользователя, а все прочие доступны для передачи пользовательских данных.

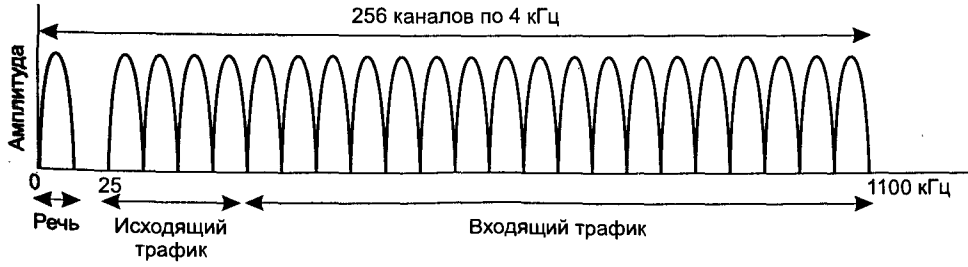


Рис. 2.24. Работа ADSL с использованием дискретной мультитональной модуляции

В принципе, каждый из свободных каналов может быть использован для полнодуплексной передачи, однако из-за помех, взаимной интерференции и т. д. практически это не реализуется. Провайдер может самостоятельно определять, сколько каналов использовать для входящего трафика, сколько для исходящего. Технически возможно осуществлять такое разделение в пропорции 50/50, но фактически большинство провайдеров предоставляет 80–90 % пропускной способности для передачи в сторону абонентов, исходя из их реальных потребностей. Обычно под исходящий трафик пользователю отводится 32 канала, по всем остальным информационным каналам он может принимать данные. В целях увеличения пропускной способности можно несколько последних каналов сделать дуплексными, однако это потребует введения в строй дополнительных схем, исключающих образование эха.

Стандарт ADSL (ANSI T1.413 и ITU G.992.1) позволяет принимать входящий трафик со скоростью 8 Мбит/с и отправлять исходящий со скоростью 1 Мбит/с. Тем не менее, лишь немногие провайдеры обеспечивают работу с такими параметрами. Стандартной услугой является 512 Кбит/с для входящего потока и 64 Кбит/с для исходящего. За дополнительную плату можно получить 1 Мбит/с и 256 Кбит/с в соответствующих направлениях.

В каждом канале используется схема модуляции, напоминающая V.34, хотя скорость отсчетов равна 4000 бод, а не 2400 бод, как в обычном телефонном стандарте. Качество линии отслеживается каждым каналом, и скорость передачи постоянно подстраивается под этот параметр, поэтому каналы могут иметь разную скорость. Сами данные передаются с помощью метода модуляции QAM, количество бит на бод достигает при этом 15, а амплитудно-фазовая диаграмма аналогична представленной на рис. 2.21, *б*. Пропускная способность входящего трафика при отведенных для него 224 каналах и 15 бит на отсчет на линии 4000 бод составляет 13,44 Мбит/с. На практике соотношение сигнал/шум никогда не бывает достаточным для достижения такой скорости, однако 8 Мбит/с на коротких дистанциях и качественных линиях — это реально. Данное обстоятельство стимулирует развитие и распространение стандарта.

Типичная организация ADSL-линии показана на рис. 2.25. На схеме видно, что телефонная компания установила в помещении у абонента специальное **устройство сопряжения с сетью**, **NID** (Network Interface Device). Эта маленькая пластмассовая коробочка маркирует окончание зоны владений телефонной компании и начало частной собственности абонента. Недалеко от этого устройства (а иногда вообще в одном блоке с ним) расположен **разветвитель**, который представляет собой аналоговый фильтр, отделяющий полосу POTS (0–4000 Гц) от каналов данных. Сигналы, проходящие по POTS, передаются на имеющийся телефон или факс, а все остальные отправляются на ADSL-модем. Последний является, на самом деле, цифровым сигнальным процессором, настроенным на работу в качестве двухсот пятидесяти QAM-модемов, работающих одновременно на разных частотах. Поскольку большинство модемов ADSL — внешние, то требуется организовать высокоскоростное соединение модема с системным блоком компьютера. Обычно это делается с помощью сетевой карты Ethernet со стороны компьютера. При этом организуется миниатюрная локальная сеть Ether-

net, состоящая из двух узлов: компьютера и модема. Изредка применяется USB-порт вместо Ethernet. В будущем, несомненно, появятся внутренние ADSL-модемы[1].

На противоположном конце кабеля, на оконечной коммутационной станции, также установлен разветвитель. Здесь голосовая составляющая сигнала отделяется от информационной и пересылается на обычный телефонный коммутатор. Сигнал, передающийся на частотах, превышающих 26 кГц, отправляется на устройство нового типа, которое называется **мультиплексором доступа к DSL, DSLAM** (Digital Subscriber Line Access Multiplexer), в состав которого в качестве ADSL-модема входит сигнальный процессор того же типа, что и у абонента. По мере восстановления по цифровому сигналу битовой последовательности формируются пакеты, отсылающиеся провайдеру.

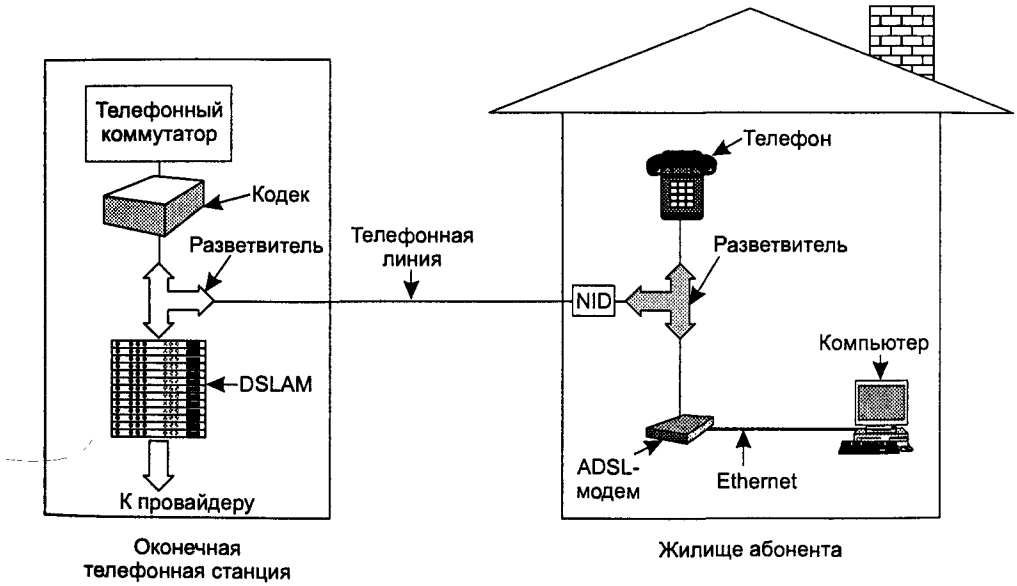


Рис. 2.25. Типичная конфигурация оборудования ADSL

Полное разделение голосовой связи и системы передачи данных позволило телефонным компаниям без особых проблем внедрить ADSL. Требуется всего лишь приобрести DSLAM и разветвитель и подсоединить абонентов ADSL к этому разветвителю. Прочие системы с высокой пропускной способностью (например, ISDN) требуют гораздо больших усилий для их внедрения и согласования с имеющимся коммутационным оборудованием.

Одним из недостатков представленной на рис. 2.25 системы является наличие NID и разветвителя в жилище пользователя. Установить это оборудование мо-

[1] Внутренние ADSL-модемы уже появились на российском рынке (например, семейство устройств CNAD-800, разработка CNet Technology). — *Примеч. перев.*

жет только технический специалист телефонной компании, что выражается, прежде всего, в высокой стоимости выездных услуг. По этой причине был стандартизован другой вариант комплектации — в нем отсутствует разветвитель. Его неофициально называют G.lite, а в соответствии со спецификацией ITU этот стандарт носит название **G.992.2**. Фактически, это та же самая схема, но без разветвителя. Имеющаяся телефонная линия используется как есть. Единственное, что пользователю необходимо сделать, это вставить в разъем каждого телефонного аппарата специальный микрофильтр, который в итоге должен оказаться в схеме между телефоном или ADSL-модемом и телефонной линией. Телефонный микрофильтр вырезает сигналы, частоты которых превышают 3400 Гц. Что же касается фильтра для ADSL-модема, то он, напротив, пропускает только высокие частоты, вырезая диапазон от 0 до 26 кГц. Однако система с разветвителем является более надежной, поэтому максимальная скорость работы G.lite — только 1,5 Мбит/с (против 8 Мбит/с в системах с разветвителем). При любом из вариантов на стороне телефонной станции разветвитель устанавливается, но это не требует выезда технического специалиста для подключения каждого абонента.

ADSL — это просто стандарт физического уровня. Что творится на более высоких уровнях, зависит от конкретной ситуации. Часто используется ATM благодаря способности этой технологии обеспечивать надлежащее качество обслуживания, а также благодаря широкому применению ATM в ядре телефонных сетей.

Беспроводные местные линии

Начиная с 1996 года, в США (и чуть позднее в других странах) компании получили законную возможность конкурировать с бывшими монополистами — местными телефонными компаниями (LEC). Компании, существующие со стародавних времен, получили даже специальное название — **ILEC** (Incumbent LEC — компания, занимающая должность LEC). Наиболее вероятными кандидатами на их место стали компании междугородной связи (IXC). Любая IXC, желающая заняться предоставлением местной связи, обязана проделать следующие шаги. Во-первых, необходимо приобрести или взять в аренду помещение под свою первую в городе оконечную станцию. Во-вторых, нужно наполнить это помещение оборудованием: коммутаторами, разветвителями и т. д. Все это можно запросто приобрести, поскольку производством подобной аппаратуры занимаются многие фирмы. В-третьих, необходимо протянуть оптическое волокно между своей коммутационной станцией и ближайшей местной телефонной станцией, чтобы абоненты имели доступ к национальной телефонной сети. В-четвертых, нужно привлечь к себе пользователей, предложив им более высокое качество обслуживания при более низких ценах по сравнению с ILEC.

После этого начинается самое сложное. Представьте себе, что пользователи действительно встрепенулись и побежали занимать очередь. Каким образом новорожденная местная телефонная компания, **CLEC** (Competitive LEC — LEC-конкурент), сможет обеспечить подключение абонентских телефонов и компьютеров к своей сияющей новизной коммутационной станции? Покупка права на прокладку кабеля и реализация этого права обошлись бы чрезвычайно дорого.

Многие CLEC нашли альтернативу традиционной витой паре в **беспроводной локальной линии, WLL** (Wireless Local Loop).

В некотором смысле стационарный телефон, использующий беспроводную локальную линию, больше напоминает мобильный телефон, однако есть три принципиальных технических отличия. Во-первых, пользователю такой системы требуется высокоскоростной доступ в Интернет. Причем, скорость должна быть по крайней мере равна скорости ADSL. Во-вторых, скорее всего, абонент не ожидает, что для установки системы ему потребуется технический специалист из фирмы, который поставит на крыше направленную антенну. В-третьих, пользователь не собирается никуда перемещаться вместе со своим телефоном и настольным компьютером, что исключает проблемы, связанные с мобильностью и организацией сотовой связи. Таким образом, появилась новая услуга — стационарная беспроводная связь (то есть телефонная связь и интернет-услуги, совмещенные в одной беспроводной системе).

Хотя первые WLL начали серьезно работать в 1998 году, нам нужно сперва вернуться в далекий 1969 год, чтобы узнать о происхождении этих систем. Именно тогда комиссия FCC разместила 2 образовательных телевизионных канала (по 6 МГц каждый) в диапазоне 2,1 ГГц. В последующие годы добавился еще 31 канал в диапазоне 2,5 ГГц, заняв в целом полосу шириной 198 МГц.

Образовательные каналы никуда не исчезали, но в 1998 году FCC отобрала частоты и разместила на них двухстороннюю радиосвязь. Тотчас же на базе этого диапазона стали развиваться беспроводные локальные линии. На таких частотах длина волны составляет 10–12 см и может распространяться на расстояние до 50 км, неплохо проникая сквозь любую растительность и дождевую завесу. И вот все 198 МГц вскоре оказались заняты беспроводными локальными линиями. Данный сервис получил название **многоканальной многоадресной распределительной службы**, **MMDS** (Multichannel Multipoint Distribution Service). Систему MMDS, как и обсуждаемую далее родственную ей LMDS, можно рассматривать в качестве региональной вычислительной сети (MAN).

Одним из достоинств данной услуги является то, что базовая технология уже хорошо развита, и оборудование, обеспечивающее ее работу, производится уже давно. Главный недостаток заключается в том, что полоса пропускания весьма ограничена и должна разделяться между большим количеством пользователей, расположенных на весьма немалой территории.

Узкая полоса пропускания MMDS привела к тому, что разработчики заинтересовались миллиметровыми волнами, которые могли бы стать альтернативой стандартному диапазону. В диапазонах 28–31 ГГц (в США) и 40 ГГц (в Европе) никто не работает, поскольку не так просто создать кремниевые интегральные схемы, работающие с подходящей скоростью. Проблема была решена, когда появились схемы, построенные на арсениде галлия. Появилась возможность использовать миллиметровые диапазоны в радиокоммуникациях. В ответ на это достижение комиссия FCC предоставила полосу шириной 1,3 ГГц новой службе беспроводной местной связи — **LMDS** (местная многоадресная распределительная служба). Это была беспрецедентная акция FCC, поскольку никогда под одно

применение не выделялась столь широкая полоса. В Европе для LMDS была выделена примерно такая же полоса, но на частотах начиная с 40 ГГц.

Работа LMDS изображена на рис. 2.26. Показана вышка с несколькими антеннами, направленными в разные стороны. Поскольку миллиметровые волны довольно-таки узконаправленные, каждая антенна захватывает свой сектор, не связанный с другими. Зона приема на используемых частотах ограничена 2–5 км. Это означает, что для того, чтобы услугами LMDS мог пользоваться большой город, необходимо установить множество таких башен.

Как и ADSL, LDMS использует асимметричное распределение полосы пропускания, отдавая предпочтение входящему трафику пользователя. В результате каждый сектор, разделяемый между всеми пользователями, может передавать данные со скоростями 36 Гбит/с и 1 Мбит/с (соответственно для каждого из направлений передачи). Если, скажем, каждый пользователь скачивает три 5-килобайтных веб-страницы в минуту, то на него уходит примерно 2000 бит/с. Это означает, что всего в секторе могут одновременно работать 18 000 таких пользователей. Тем не менее, чтобы уменьшить задержки, обычно ограничиваются 9 тысячами абонентов. Имея 4 сектора (см. рис. 2.26), получаем аудиторию, состоящую из 36 000 абонентов. Предполагая, что в час пик на линии находится каждый третий абонент, можно рассчитывать на 100 000 клиентов, пользующихся одной и той же башней и находящихся на расстоянии менее 5 км от нее. Примерно так представляют себе ситуацию многие компании CLEC. Исходя из этих подсчетов, они сделали вывод о том, что при весьма скромных вложениях в систему можно получать от нее высокий доход, конкурируя с телефонными компаниями и традиционными провайдерами. Предлагаемая скорость передачи при этом сравнима с кабельным телевидением, а цены гораздо ниже.

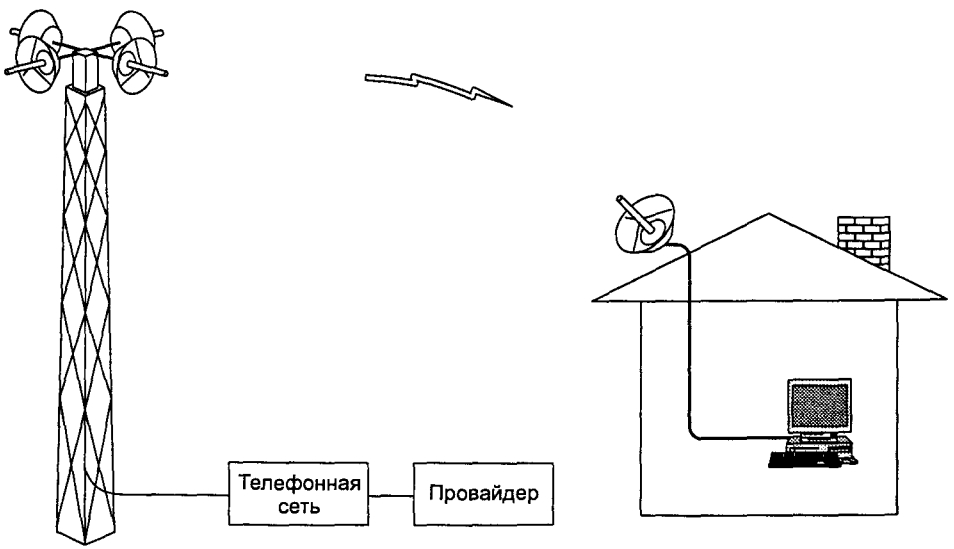


Рис. 2.26. Архитектуре системы LMDS

Тем не менее, есть определенные проблемы и в технологии LMDS. Во-первых, миллиметровые волны распространяются исключительно по прямой, поэтому антенны, установленные на башнях, должны находиться в зоне прямой видимости антенн, установленных на крышах домов. Неплохо вбирают в себя сигнал любопытные листья деревьев, поэтому башни должны быть достаточно высоки, чтобы на пути сигнала не оказалось растений. При этом декабрьская «чистая линия» сильно отличается от июльской, когда на деревьях много зелени. Дождь также вбирает в себя сигнал. В некоторой степени можно исключить возникающие из-за этого ошибки путем применения специальных кодов с коррекцией ошибок или повышения мощности передатчиков при плохих погодных условиях. И все же идеальным для LMDS является сухой климат. Скажем, в Санкт-Петербурге эта система будет работать значительно хуже, чем в Баку.

Беспроводные местные линии не войдут в моду до тех пор, пока не будут выработаны официальные стандарты, которые заставят производителей оборудования создавать такую продукцию, которую абонентам не пришлось бы менять при смене компании CLEC. Для создания этого стандарта институт IEEE выделил отдельный комитет 802.16. Результаты его работы были опубликованы в апреле 2002 года. IEEE называет стандарт 802.16 **стандартом беспроводных региональных вычислительных сетей** (Wireless MAN).

IEEE 802.16 регламентирует применение цифровой телефонии, доступа к сети Интернет, соединение удаленных ЛВС, теле- и радиовещание и др. Более детально мы рассмотрим этот стандарт в главе 4.

Магистрали и уплотнение

Экономия ресурсов играет важную роль в телефонной системе. Стоимость прокладки и обслуживания магистрали с высокой пропускной способностью и низкокачественной линии практически одна и та же (то есть львиная доля этой стоимости уходит на рытье траншей, а не на сам медный или оптоволоконный кабель). По этой причине телефонные компании совместно разработали несколько схем передачи нескольких разговоров по одному физическому кабелю. Схемы мультиплексирования (уплотнения) могут быть разделены на две основные категории: **FDM** (Frequency Division Multiplexing — частотное уплотнение) и **TDM** (Time Division Multiplexing — мультиплексирование с временным уплотнением). При частотном уплотнении частотный спектр делится между логическими каналами, при этом каждый пользователь получает в исключительное владение свой поддиапазон. При мультиплексировании с временным уплотнением пользователи по очереди (циклически) пользуются одним и тем же каналом, и каждому на короткий промежуток времени предоставляется вся пропускная способность канала.

Радиовещание с амплитудной модуляцией (AM) является иллюстрацией обеих идей мультиплексирования. Весь спектр имеет ширину около 1 МГц, примерно от 500 до 1500 кГц. Он поделен между логическими каналами (радиостанциями). Каждая станция работает лишь в отведенной ей части спектра, при этом разделение между каналами должно быть достаточно большим, чтобы предотвратить взаимные помехи каналов. Такая система является примером частотного мультиплексирования. Кроме того, в некоторых странах отдельные радиостан-

ции имеют два логических субканала, один из которых предназначен для музыки, другой — для рекламы. Они работают на одной и той же частоте, но в разное время, сменяя друг друга. Это — типичный пример системы с разделением времени (то есть временного уплотнения).

Далее мы рассмотрим частотное мультиплексирование. Затем обратимся к вопросу о том, как частотное мультиплексирование может быть реализовано в оптоволоконных каналах (мультиплексирование с использованием разных длин волн). Затем мы обсудим мультиплексирование с разделением времени, закончив данный раздел системой, используемой в оптоволоконной связи (SONET).

Частотное уплотнение

На рис. 2.27 показано, как три речевых телефонных канала могут объединяться в одну линию с использованием частотного уплотнения. Фильтры ограничивают используемую полосу частот примерно 3100 Гц на каждый речевой канал. При мультиплексировании нескольких каналов каждому из них выделяется полоса частот шириной 4000 Гц, что позволяет как следует разделить их. Для начала сигналы повышаются по частоте, причем для разных каналов величины сдвигов разные. После этого их можно суммировать, поскольку каждый канал теперь сдвинут в свою область спектра. Обратите внимание на то, что соседние каналы немного перекрываются, несмотря на разрыв между ними (последний называется **защитной полосой**). Происходит это из-за того, что частотная характеристика обрезанного фильтром сигнала вовсе не выглядит прямоугольной, а имеет некоторый наклон с обеих сторон. Это означает, что сильный всплеск в одном канале может ощущаться как нетермальный шум соседнем канале.

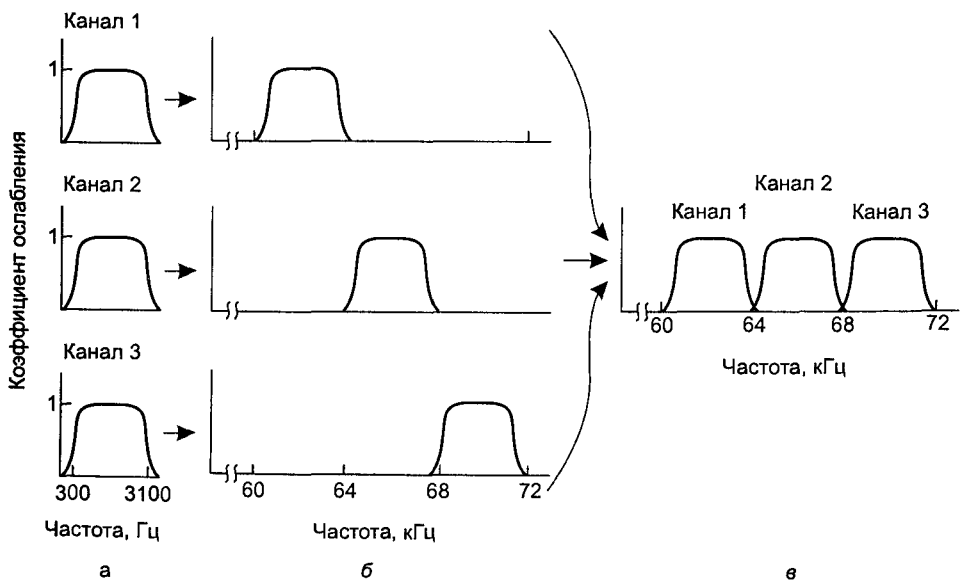


Рис. 2.27. Частотное уплотнение: исходные спектры сигналов (а); спектры, сдвинутые по частоте (*б*); уплотненный канал (*в*)

Используемые во всем мире схемы FDM в определенной степени стандартизированы. Широко распространенным стандартом является использование 12 речевых каналов с полосой канала 4000 Гц, уплотняемых в диапазоне от 60 до 108 кГц. Эти 12 каналов образуют **базовую группу**. Диапазон от 12 кГц до 60 кГц иногда используется для другой группы. Многие операторы связи предоставляют выделенные линии с пропускной способностью от 48 до 56 Кбит/с, основанные на группах. Пять групп (60 речевых каналов) могут быть уплотнены, тем самым они образуют **супергруппу**. Следующая единица в этой иерархии, состоящая из пяти (стандарт CCITT) или десяти (стандарт Bell System) супергрупп, называется **главной группой**. Кроме того, существуют и другие стандарты, объединяющие до 230 000 речевых каналов.

Спектральное уплотнение

В оптоволоконных каналах используется особый вариант частотного уплотнения. Он называется **спектральным уплотнением** (WDM, Wavelength-Division Multiplexing). Способ реализации частотного уплотнения в оптоволоконных линиях показан на рис. 2.28. Здесь четыре кабеля подходят к одному сумматору, и по каждому из них идет сигнал со своей энергией в своем частотном диапазоне. Четыре луча объединяются и дальше распространяются по одному волокну. На противоположном конце они расщепляются разветвителем. На каждом выходном кабеле имеется короткий специальный участок внутреннего слоя, который является фильтром, пропускающим сигнал только одной длины волны.

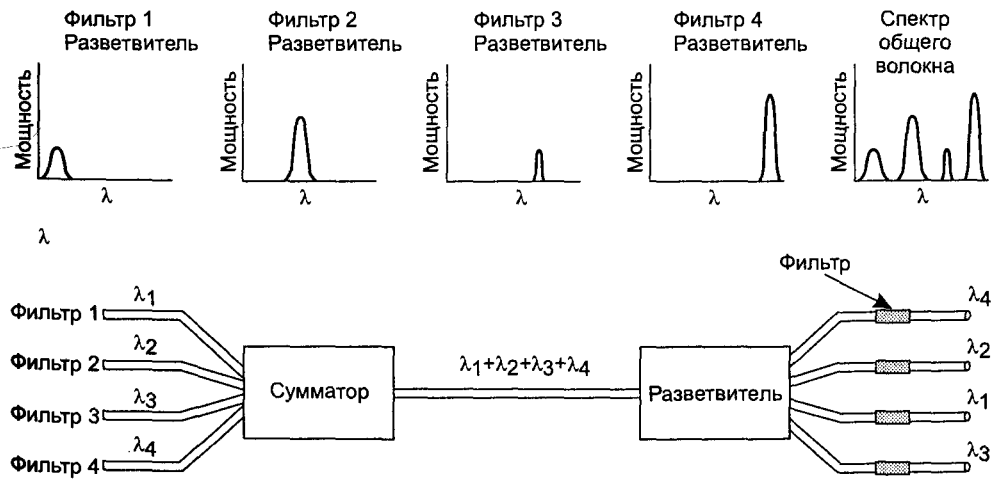


Рис. 2.28. Уплотнение с разделением длины волны

Данный метод не представляет собой ничего нового. Это просто частотное уплотнение на очень высоких частотах. Поскольку каждый сигнал передается в своем частотном диапазоне и эти диапазоны успешно разделяются, подобный вариант мультиплексирования может применяться для передачи на большие расстояния. Единственным отличием от электрического частотного уплотнения яв-

ляется в данном случае то, что система, использующаяся для уплотнения, то есть призма или дифракционная решетка, является абсолютно пассивным, а следовательно, чрезвычайно надежным элементом.

Технология WDM развивается с такой скоростью, что компьютерным технологиям остается только стыдиться перед ней своих темпов развития. Она была изобретена примерно в 1990 году. Первые коммерческие системы использовали 8 каналов по 2,5 Гбит/с на канал. К 1998 году на рынке появились уже 40-канальные системы с такой же пропускной способностью канала. В 2001 году была выпущена система из 96 каналов по 10 Гбит/с (то есть общая пропускная способность составила 960 Гбит/с). Такой емкости достаточно, чтобы передавать 30 полнометражных фильмов в секунду (в формате MPEG-2). В лабораториях уже работают версии, имеющие 200 каналов. При возрастании числа каналов длины волн различаются на очень малые величины (например, 0,1 нм). В этом случае системы называют **плотными WDM**, или **DWDM** (Dense WDM).

Следует отметить, что спектральное уплотнение является популярным. Один оптический кабель обычно работает на частоте не более нескольких гигагерц из-за невозможности более быстрого преобразования электрических сигналов в оптические и обратно. Однако возможности самого кабеля гораздо выше, поэтому, объединяя сигналы разных длин волн на одном кабеле, можно получить суммарную пропускную способность, линейно зависящую от числа каналов. Полоса пропускания одного волокна составляет 25 000 Гц (см. рис. 2.6), следовательно, даже при 1 бит/Гц можно разместить 2500 каналов по 10 Гбит/с (хотя соотношение бит/Гц можно увеличить).

Еще одной новой разработкой является оптический усилитель. Раньше необходимо было через каждые 100 км разбивать сигнал на каналы, преобразовывать оптические каналы в электрические и усиливать последние традиционным способом, после чего выполнять обратное преобразование и объединение. Теперь же любые оптические усилители могут регенерировать объединенный сигнал целиком через каждые 1000 км, при этом нет необходимости в оптико-электрических преобразованиях.

В примере на рис. 2.28, изображена система с постоянными длинами волн. Данные из входного кабеля 1 попадают на выходной кабель 3, а из кабеля 2 — в кабель 1 и т. д. Однако можно построить и коммутируемые WDM-системы. В таком устройстве выходные фильтры настраиваются с помощью интерферометров Фабри—Перо или Маха—Цандера. Чтобы узнать больше про спектральное уплотнение и его применения, читайте (Elmirghani and Mouftah, 2000; Hunter and Andonovic, 2000; Listani и др., 2001).

Мультиплексирование с разделением времени

Спектральное уплотнение — это прекрасно, однако в телефонных системах все еще много участков медного провода, поэтому нам необходимо снова вернуться к традиционным носителям. Частотное уплотнение все еще используется при передаче данных по медным проводникам или микроволновым каналам, однако это требует применения аналоговым схем, что не очень подходит для компьютерных технологий. Временное уплотнение, напротив, может быть реализовано исклю-

чительно цифровой электроникой, поэтому этот метод в последнее время находит все большее распространение. К сожалению, его можно применять только для работы с цифровыми данными. Поскольку по местным линиям передаются аналоговые сигналы, то необходимо выполнять аналого-цифровые преобразования на оконечных станциях, на которых все локальные линии соединяются в большие магистрали.

Сейчас мы рассмотрим, как несколько аналоговых речевых каналов оцифровываются и затем объединяются в один выходной канал. Компьютерные данные, посылаемые модемом, также передаются в аналоговом виде по местным телефонным линиям, поэтому нижеследующее описание касается их целиком и полностью. Аналоговые сигналы оцифровываются на оконечной телефонной станции устройством, называемым **кодек** (кодер-декодер), которое вырабатывает серии 8-битных чисел. Частота дискретизации кодека составляет 8000 отсчетов в секунду (125 мкс/отсчет). Это связано с теоремой Найквиста, в которой утверждается, что такой частоты достаточно для извлечения всей информации из телефонного канала с полосой частот в 4 кГц. При более низкой частоте часть информации была бы потеряна, а более высокая скорость отсчетов была бы излишней. Подобная технология называется **кодово-импульсной модуляцией**, **PCM** (pulse-code modulation). Кодово-импульсная модуляция составляет основу современной телефонной системы. В результате практически все временные интервалы, используемые в телефонной системе, кратны 125 мкс.

Когда технология цифровой передачи данных стала реальностью, CCITT так и не удалось достичь соглашения по поводу международного стандарта на кодово-импульсную модуляцию. И вот теперь в различных странах используется большое количество совершенно не совместимых друг с другом схем.

Метод мультиплексирования с разделением времени, используемый в Северной Америке и Японии, называется «носитель T1». Он изображен на рис. 2.29. (Строго говоря, формат называется DS1, а T1 — это название носителя, но мы не будем здесь проводить столь жесткого разграничения). Носитель T1 состоит из 24 объединенных речевых каналов. Обычно аналоговые каналы оцифровываются поочередно путем подачи на вход кодека результирующего аналогового потока. Это оптимальнее применения 24 кодеков с объединением их выходных цифровых сигналов. Каждый из 24 каналов по очереди превращается кодеком в 8-битную последовательность, вставляемую в выходной поток данных. Семь битов являются информационными, а восьмой используется для контроля. Таким образом, получается поток данных в $7 \cdot 8000 = 56\,000$ бит/с плюс $1 \cdot 8000 = 8000$ бит/с поток сигнальной информации для каждого канала.

Кадр состоит из $24 \cdot 8 = 192$ битов плюс еще один бит-ограничитель кадра, итого 193 бита каждые 125 мкс. В результате это дает огромную суммарную скорость передачи данных в 1,544 Мбит/с. 193-й бит используется для синхронизации кадров. Он представляет собой последовательность такого вида: 01010101.... Обычно приемник постоянно проверяет состояние этого бита, чтобы убедиться, не потерял ли он синхронизацию. Если это вдруг происходит, то приемник сканирует принятые данные, отыскивая кадровый бит и с его помощью восстанавливая синхронизацию. Аналоговые пользователи вообще не могут создавать битовые последовательности, поскольку они соответствуют синусоиде с частотой

4000 Гц, которую невозможно отфильтровать. Цифровые пользователи, разумеется, могут это делать, но вероятность создания именно такой последовательности довольно мала. Кроме того, когда система T1 используется только для передачи цифровых данных, то информационными являются только 23 канала. 24-й канал целиком выделяется под синхронизирующую последовательность, что позволяет намного быстрее восстановить синхронизацию.

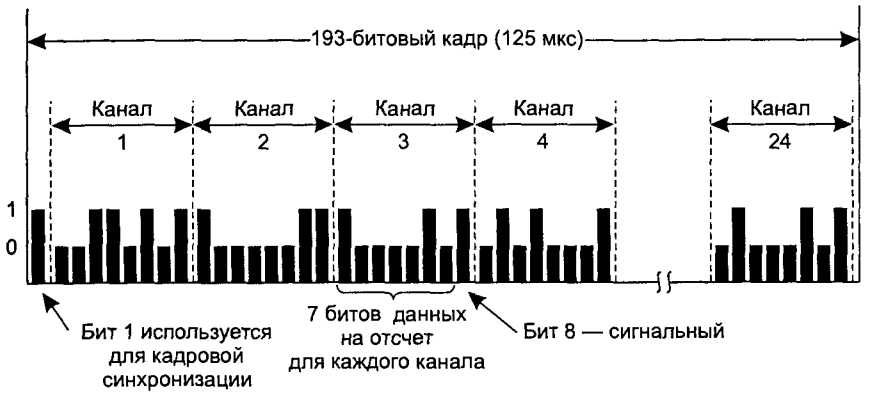


Рис. 2.29. Носитель T1 (1,544 Мбит/с)

Когда CCITT наконец достигло соглашения, было решено, что отводить 8000 бит/с на сигнальную информацию — это слишком много. В результате появился еще один стандарт канала со скоростью 1,544 Мбит/с, в котором аналоговый сигнал представлялся не семью, а восемью битами, то есть 256 дискретными уровнями вместо 128. Таким образом, существуют два несовместимых между собой варианта. В первом из них, **CCS** (common channel signaling — сигнализация по общему каналу), дополнительный бит, добавляемый не в начало 193-битового кадра, а в его конец, принимает последовательность значений 10101010... в нечетных кадрах, тогда как в четных используется для сигнальной информации, относящейся ко всем каналам.

Во втором варианте, **CAS** (Channel Associated Signaling — сигнализация, ассоциированная с каналом), у каждого канала имеется свой сигнальный подканал, на который выделяется один бит из 8 в каждом шестом кадре. Таким образом, пять из шести отсчетов являются 8-битовыми, а каждый шестой отсчет содержит 7 бит данных. Комитет CCITT также принял рекомендацию на применение носителя **E1** с кодово-импульсной модуляцией со скоростью 2,048 Мбит/с. Этот носитель содержит 32 8-битовых отсчета, упаковываемых в основной кадр — 125 мкс. 30 каналов используются для передачи данных, два являются сигнальными. Каждая группа из четырех кадров содержит 64 сигнальных бита, половина которых используется для сигнализации, ассоциированной с каналом, а другая половина используется для синхронизации кадров или резервируется для различных нужд в разных странах. Стандарт 2,048 Мбит/с используется за пределами Северной Америки и Японии вместо T1.

При оцифровывании речевого сигнала было бы соблазнительно попытаться с помощью статистических методов уменьшить количество бит, необходимых для передачи информации в каждом канале. Подобные приемы применимы не только для речевых, но также и для любых аналоговых сигналов. Все эти методы сжатия основываются на том принципе, что сигнал меняется относительно медленно по сравнению с частотой дискретизации, поэтому большая часть информации в 7- или 8-разрядном числе является избыточной.

Один из методов сжатия, называющийся **дифференциальной кодово-импульсной модуляцией**, заключается в выводе не амплитуды сигнала, а разности текущего и предыдущего значений амплитуды. Поскольку скачки более чем на ±16 уровней на шкале из 128 уровней маловероятны, то вместо 7 бит может оказаться достаточно 5. Если же сигнал неожиданно совершит большой скачок, то кодирующей системе понадобится несколько периодов дискретизации, чтобы догнать убежавший сигнал. При передаче речи такая ошибка, впрочем, вообще может быть проигнорирована.

Один из вариантов метода сжатия требует, чтобы значение сигнала в каждом отсчете отличалось от предыдущего на +1 или −1. При этом можно передавать всего лишь один бит, сообщающий, увеличился сигнал по отношению к предыдущему значению или уменьшился. Это называется **дельта-модуляцией** (рис. 2.30). Как и любой подобный метод сжатия, дельта-модуляция предполагает, что сигнал изменяется довольно медленно, то есть значения сигналов в соседних отсчетах различаются ненамного. Если сигнал изменяется быстрее, чем позволяет метод, информация теряется.

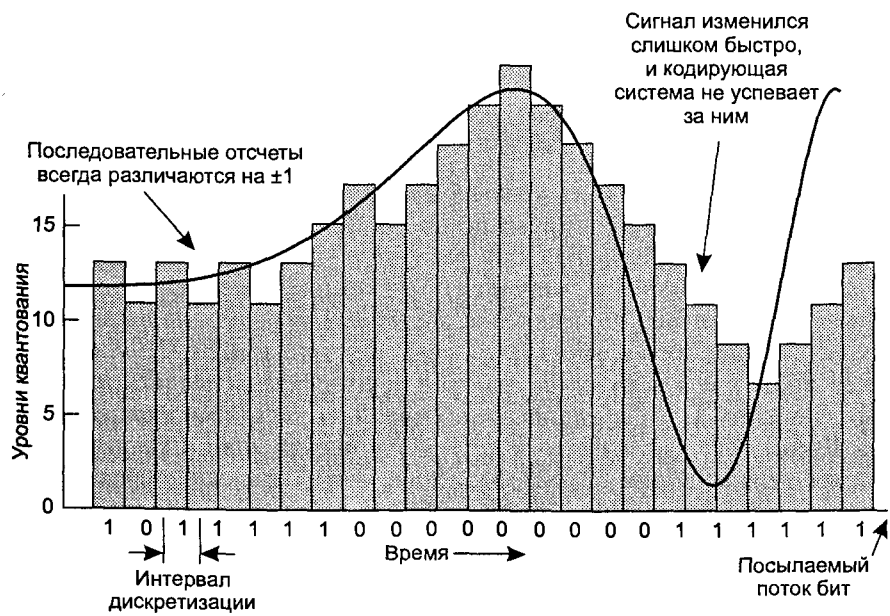


Рис. 2.30. Дельта-модуляция

Более сложным вариантом дифференциальной кодово-импульсной модуляции является предсказание следующего значения сигнала по нескольким предыдущим отсчетам. При этом кодируется только разница между реальным и предсказанным сигналом. Приемник и передатчик должны использовать, конечно же, один и тот же алгоритм предсказания. Серия подобных методов называется **кодированием с предсказанием**. Они полезны тем, что позволяют уменьшить размер чисел, которые нужно кодировать, а следовательно, количество передаваемых бит.

Носители T1 могут объединяться для передачи по каналам более высокого уровня при помощи мультиплексирования с разделением времени. На рис. 2.31 показано, как это может быть сделано. Слева мы видим четыре канала T1, объединяемых в один канал T2. Мультиплексирование в канале T2 и каналах более высоких порядков выполняется побитно, а не побайтно, причем 24 голосовых канала составляют один кадр T1. Четыре потока T1 по 1,544 Мбит/с образуют 6,176 Мбит/с, однако реально в канале T2 используется скорость передачи, равная 6,312 Мбит/с. Дополнительные биты используются для синхронизации кадров и восстановления в случае сбоя носителя. T1 и T3 активно используются рядовыми пользователями, тогда как T2 и T4 можно найти только внутри телефонной системы, поэтому они не столь известны.

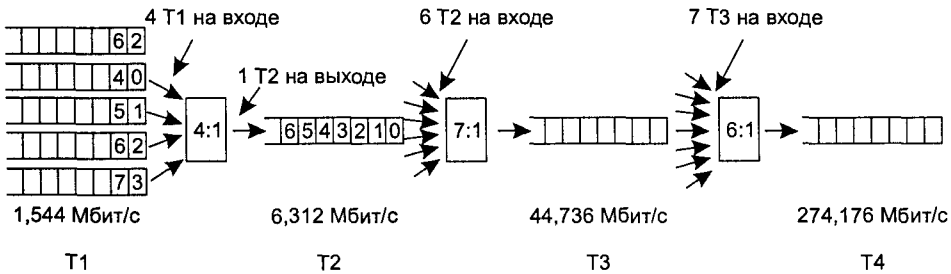


Рис. 2.31. Мультиплексирование потоков T1 на носителях высших порядков

На следующем уровне семь каналов T2 объединяются побитно в канал T3. Затем шесть потоков T3 формируют поток T4. На каждом этапе добавляется небольшое количество избыточной информации для синхронизации кадров.

Между США и остальным миром нет почти никаких договоренностей по поводу основного носителя, а также о том, каким образом они должны мультиплексироваться в каналы более высоких уровней. Используемый в США способ объединения по 4, 7 и 6 каналов не побудил другие страны сделать то же самое, поэтому стандарт CCITT предписывает объединять каналы по четыре на каждом уровне. Кроме того, различается и служебная информация. Стандарты CCITT описывают иерархию для 32, 128, 512, 2048 и 8192 каналов, передаваемых на скоростях 2,048, 8,848, 34,304, 139,264 и 565,148 Мбит/с.

SONET/SDH

Когда оптоволоконная связь только появилась, у каждой телефонной компании была своя собственная система мультиплексирования с разделением времени.

После раздела в 1984 года корпорации AT&T местным телефонным компаниям пришлось подключаться к разным междугородным линиям с различными оптическими системами TDM. Появилась очевидная потребность в стандартизации. В 1985 году исследовательское подразделение региональных телефонных компаний Bellcore начало разработку стандарта **SONET** (Synchronous Optical Network — синхронная оптическая сеть). Позднее к этой работе подключился комитет CCITT, что привело к созданию в 1989 году стандарта SONET, а также набора параллельных рекомендаций CCITT (G.707, G708 и G709). Рекомендации CCITT, получившие название SDH (Synchronous Digital Hierarchy — синхронная цифровая иерархия), отличаются от стандарта SONET небольшими деталями. Практически все телефонные линии дальней связи в США и во многих других странах сегодня используют SONET на физическом уровне. Дополнительную информацию см. в книгах (Bellamy, 2000; Goralsky, 2000; Shepard, 2001).

При разработке системы SONET ставились четыре главные цели. Во-первых, SONET должен был обеспечивать объединение сетей, построенных на различных носителях. Для достижения этой цели потребовалось определить общий стандарт, описывающий длины волн, синхронизацию, структуру кадра и другие вопросы.

Во-вторых, требовалось средство объединения цифровых систем США, Европы и Японии, построенных на 64-килобитных каналах с кодово-импульсной модуляцией, но использующих эти каналы различными (и не совместимыми друг с другом) способами.

В-третьих, SONET должен был предоставить способ объединения нескольких цифровых каналов. Во время разработки системы SONET наиболее быстрым широко используемым в США носителем был T3 со скоростью 44,736 Мбит/с. Стандарт T4 уже был описан, но мало использовался, а стандарта выше T4 определено не было. Одной из задач SONET было продолжить иерархию до скоростей, измеряющихся в гигабитах в секунду. Также нужен был стандартный способ объединения нескольких медленных каналов в один канал SONET.

В-четвертых, SONET должен был обеспечить поддержку операций, администрирования и обслуживания (OAM, Operation, Administration, Maintenance). Предыдущие системы справлялись с этой задачей не слишком хорошо.

Вначале было решено реализовать SONET на основе традиционной системы мультиплексирования с разделением времени, при этом вся пропускная способность оптоволоконного кабеля выделялась одному каналу, который разбивался на интервалы времени, выделяемые подканалам. SONET как таковая является синхронной системой. Интервалы между посылаемыми битами управляются таймером с точностью 10^{-9}. Биты отсылаются в линию также в строго определенные моменты времени, контролируемые главным таймером. Когда впоследствии была предложена коммутация ячеек в качестве основы ATM, тот факт, что такая система допускала произвольные интервалы между ячейками, сказался на названии асинхронного режима передачи (ATM, Asynchronous Transfer Mode), как бы в противоположность синхронному принципу функционирования сети SONET. В последней отправитель и получатель привязаны к общему таймеру, в ATM такого нет.

Обычный кадр SONET представляет собой блок из 810 байт, выдаваемых каждые 125 мкс. Поскольку SONET является синхронной системой, кадры выдаются независимо от наличия какой-либо полезной информации, которую необходимо переслать. Скорость 8000 кадров в секунду очень точно соответствует частоте дискретизации каналов PCM, используемых в телефонии.

Проще всего описать кадр SONET из 810 байт в виде прямоугольника из 9 рядов по 90 колонок. Тогда очевидно, что $8 \cdot 810 = 6480$ бит передаются 8000 раз в секунду, что дает скорость передачи 51,84 Мбит/с. Это основной канал SONET, называющийся **STS-1** (Synchronous Transport Signal — синхронный транспортный сигнал). Все магистрали SONET кратны STS-1.

Первые три колонки каждого кадра зарезервированы под системную управляющую информацию, как показано на рис. 2.32. Первые три ряда содержат заголовок раздела, следующие шесть рядов — заголовок линии. Заголовок секции генерируется и проверяется в начале и конце каждого раздела, тогда как заголовок линии генерируется и проверяется в начале и конце каждой линии.

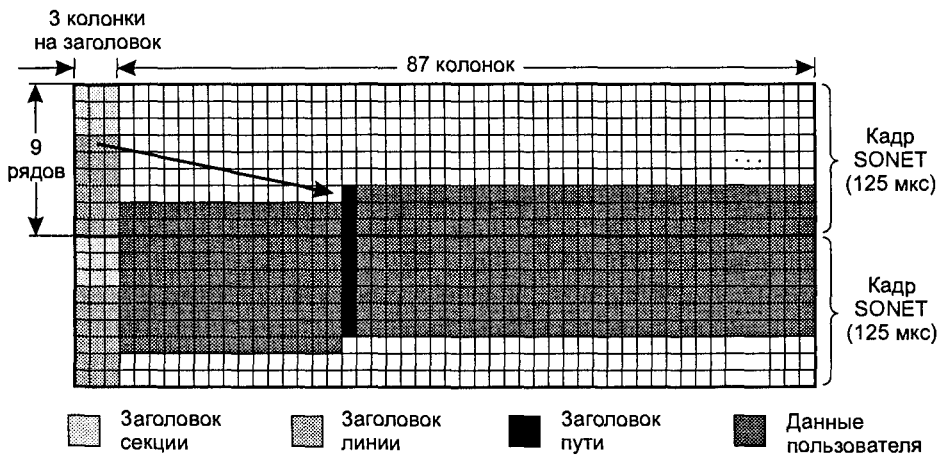


Рис. 2.32. Два соседних кадра системы SONET

Передатчик SONET посылает соседние кадры по 810 байт без межкадровых промежутков, даже если данных для передачи нет (в этом случае посылаются фиктивные байты). С точки зрения приемника это выглядит как бесконечный битовый поток. Как же он узнает, где находится граница каждого кадра? Дело в том, что первые два байта кадра содержат фиксированную последовательность, которую приемник и старается найти. Если в большом количестве последовательно принятых кадров обнаруживается одна и та же комбинация нулей и единиц, то логично предположить, что это граница кадра, и приемник считает себя синхронизированным с передатчиком. Теоретически, пользователь может регулярно вставлять служебную последовательность в поток, но на практике это не может сбить с толку приемник, так как данные, отправляемые несколькими пользователями, подвергаются уплотнению. Есть, впрочем, и другие причины.

В оставшихся 87 столбцах содержатся данные пользователя. Они передаются со скоростью 87 · 9 · 8 · 8000 = 50,112 Мбит/с. Однако на самом деле данные пользователя, называемые **синхронным полезным пакетом, SPE** (Synchronous Payload Envelope), не всегда начинаются с первой строки и четвертой колонки. SPE может начинаться где угодно в пределах кадра. А указатель на его первый байт хранится в первой строке заголовка линии. Первой колонкой SPE является заголовок пути (то есть заголовок для сквозного протокола подуровня).

Возможность начинать SPE в любом месте кадра SONET и даже занимать соседние два кадра, как показано на рис. 2.32, придает системе дополнительную гибкость. Например, если данные пользователя прибывают на источник, в то время как пустой кадр SONET уже передается, то они могут быть вставлены в текущий кадр, а не ждать начала следующего кадра.

Иерархическая система мультиплексирования SONET показана в табл. 2.4. Определены скорости для синхронных транспортных сигналов от STS-1 до STS-192. Оптический носитель (OC, Optical Carrier), соответствующий n-му синхронному транспортному сигналу (STS-n), называется OC-n и совпадает с STS-n с точностью до бита, с той разницей, что для синхронизации требуется некоторая перестановка битов. Названия SDH отличаются — они начинаются с OC-3, так как системы на основе рекомендаций CCITT не имеют стандартизованной скорости 51,84 Мбит/с. Носитель OC-9 присутствует в таблице, поскольку он довольно близко соответствует стандарту высокоскоростных магистралей передачи данных, принятому в Японии. OC-18 и OC-36 также применяются в Японии. В общую скорость потоков данных включены все управляющие сигналы. В скорость передачи полезной нагрузки SPE не входят заголовки линий и разделов. В скорость передачи данных пользователя включаются только 86 полезных колонок кадра.

Таблица 2.4. Скорости мультиплексирования SONET и SDH

SONET		SDH	Скорость передачи данных, Мбит/с		
Электрические	Оптические	Оптические	Общая	SPE	Пользователя
STS-1	OC-1		51,84	50,112	49,536
STS-3	OC-3	STM-1	155,52	150,336	148,608
STS-9	OC-9	STM-3	466,56	451,008	445,824
STS-12	OC-12	STM-4	622,08	601,344	594,432
STS-18	OC-18	STM-6	933,12	902,016	891,648
STS-24	OC-24	STM-8	1244,16	1202,688	1188,864
STS-36	OC-36	STM-12	1866,24	1804,032	1783,296
STS-48	OC-48	STM-16	2488,32	2405,376	2377,728
STS-192	OC-192	STM-64	9953,28	9621,504	9510,912

Если какой-нибудь из данных носителей не является мультиплексным, а переносит данные от одного источника, то к его названию добавляется латинская буква *c*, означающая *concatenated* (объединенный). Таким образом, OC-3 означает 155,52-мегабитный носитель, состоящий из трех отдельных носителей OC-1, а OC-3c означает передачу потока данных от одного источника со скоростью

155,52 Мбит/с. Три потока OC-1 в составе потока OC-3c разделяются одной колонкой. Первая колонка 1 отделяет поток 1, затем колонка 1 — поток 2, колонка 1 — поток 3, затем колонка 2 — поток 1, и так далее до конца кадра шириной 270 колонок и глубиной 9 строк.

Коммутация

С точки зрения среднего телефонного инженера, телефонная система состоит из двух частей: внешнего оборудования (местных телефонных линий и магистралей, вне коммутаторов) и внутреннего оборудования (коммутаторов), расположенного на телефонной станции. Мы рассмотрели внешнее оборудование. Теперь пора уделить внимание внутреннему.

В телефонных системах используются два различных приема: коммутации каналов и коммутации пакетов. Далее мы кратко познакомимся с каждым из них. Затем мы несколько подробнее обсудим коммутацию каналов, поскольку именно в соответствии с этой схемой функционирует современная телефонная система. Коммутацию пакетов мы изучим более детально в последующих главах.

Коммутация каналов

Когда вы (или ваш компьютер) снимаете телефонную трубку и набираете номер, коммутирующее оборудование телефонной системы отыскивает физический путь, состоящий из кабелей (медных или оптоволоконных; впрочем, это может быть и радиоканал) и ведущий от вашего телефона к телефону того, с кем вы связываетесь. Такая система, называемая **коммутацией каналов**, схематически изображена на рис. 2.33, *а*. Каждый из шести прямоугольников представляет собой коммутирующую станцию (оконечную или междугородную). В данном примере каждая станция имеет три входных и три выходных линии. Когда звонок проходит через коммутационную станцию, между входной и выходной линиями устанавливается физическое соединение, как показано пунктирными линиями (такова, по крайней мере, концепция).

На заре телефонии соединение устанавливалось вручную телефонным оператором, который замыкал две линии проводом с двумя штекерами на концах. С изобретением автоматического коммутатора связана довольно забавная история. Автоматический коммутатор изобрел в XIX веке владелец похоронного бюро Алмон Б. Строуджер (Almon B. Strowger) вскоре после изобретения телефона. Когда кто-либо умирал, родственник умершего звонил городскому телефонному оператору и говорил: «Соедините меня, пожалуйста, с похоронным бюро». К несчастью для мистера Строуджера, в его городе было два похоронных бюро, и жена владельца конкурирующей фирмы как раз работала телефонным оператором. Мистер Строуджер быстро понял, что либо он изобретет автоматический телефонный коммутатор, либо ему придется закрывать дело. Он выбрал первое. На протяжении почти 100 лет используемое во всем мире оборудование для коммутации каналов называлось искателем Строуджера. (История не упоминает, не устроилась ли жена его конкурента, уволенная с работы телефонного оператора, в телефонное справочное агентство, сообщая телефонный номер своего похоронного бюро всем желающим.)

Модель, изображенная на рис. 2.33, *а*, конечно, сильно упрощена, поскольку канал, соединяющий двух абонентов телефонной линии, на самом деле может быть не только медным проводом, но и, например, микроволновой или оптоволоконной магистралью, на которой объединены тысячи телефонных абонентов. Тем не менее, основная идея остается той же самой: когда один абонент звонит другому, устанавливается определенный путь, связывающий их, и этот путь остается неизменным до конца разговора.

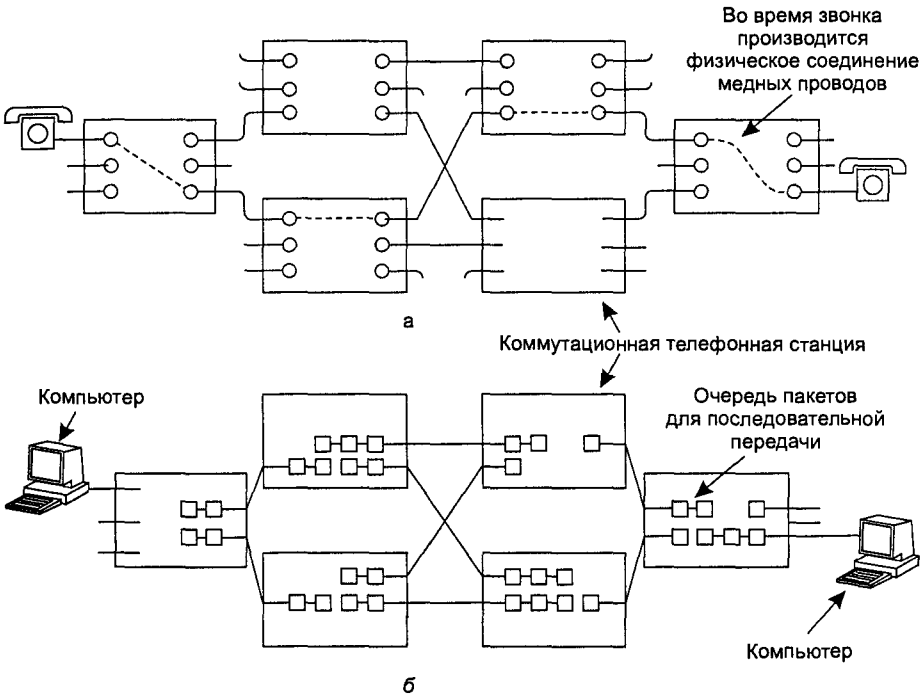


Рис. 2.33. Коммутация каналов (*а*); коммутация пакетов (*б*)

Альтернативным способом коммутации является **коммутация пакетов**, которая схематически изображена на рис. 2.33, *б*. Отдельные пакеты пересылаются как положено, однако заранее никакой путь между абонентами не устанавливается. Каждый пакет должен сам искать свой путь.

Важным свойством коммутации каналов является необходимость установления сквозного пути от одного абонента до другого *до* того, как будут посланы данные. Именно поэтому время от конца набора номера до начала разговора может занимать около 10 с и более для междугородных или международных звонков. В течение этого интервала времени телефонная система ищет путь, изображенный на рис. 2.33, *а*. Обратите внимание на то, что еще до начала передачи данных сигнал запроса на разговор должен пройти весь путь до пункта назначения и должен там быть распознан. Для многих компьютерных приложений (например, при проверке кредитной карточки клиента кассовым терминалом) длительное время установления связи является нежелательным.

В результате при установлении физического соединения между абонентами, как только этот путь установлен, единственной задержкой для распространения сигнала будет скорость распространения электромагнитного сигнала, то есть около 5 мс на каждые 1000 км. Еще одним свойством такой системы является то, что после начала разговора линия уже не может вдруг оказаться занятой, хотя она может быть занятой до установки соединения (например, благодаря отсутствию соответствующей возможности у коммутатора или магистрали).

Коммутация сообщений

Еще одной стратегией является **коммутация сообщений**, показанная на рис. 2.34, *б*. При использовании такой формы коммутации физический путь между абонентами заранее не устанавливается. Вместо этого, когда отправитель желает отослать данные, они сохраняются на ближайшей коммутационной станции (то есть на первом маршрутизаторе), а затем прыжками от одного маршрутизатора к другому передаются дальше. Каждый блок принимается целиком, анализируется на наличие ошибок, после чего пересылается дальше. В сетях такой прием называется **передачей с промежуточным хранением**, об этом уже рассказывалось в главе 1.

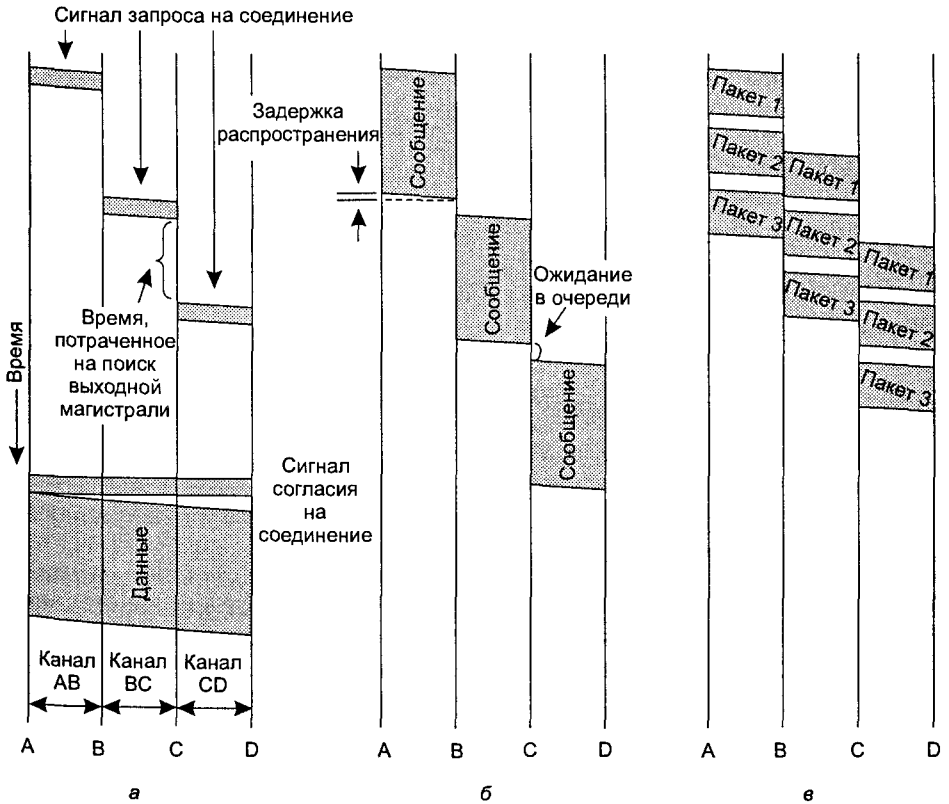


Рис. 2.34. Синхронизация событий при коммутации: каналов (*а*); сообщений (*б*); пакетов (*а*)

Коммутация сообщений использовалась в первой электромагнитной системе связи, а именно в телеграфе. Телеграмма печаталась на бумажной ленте в отключенном от линии режиме на станции-отправителе, а затем прочитывалась с ленты и передавалась по линии связи на следующую станцию, где она снова пробивалась на бумажной ленте. Оператор отрывал ленту от мотка и читал ее на одном из нескольких специальных устройств чтения лент, которых было столько же, сколько исходящих магистралей. Такой коммутатор назывался **конторой рваных лент** (torn tape office). Бумажные ленты уже далеко в прошлом, коммутация сообщений больше не используется, поэтому мы больше не будем затрагивать эту тему.

Коммутация пакетов

При коммутации сообщений размер блоков никак не ограничивается. Это означает, что маршрутизаторы (в современных системах) должны быть оснащены дисками для буферизации длинных блоков. Это также означает, что один блок может занять линию, связывающую два маршрутизатора, на несколько минут. В целом, система коммутации сообщений плохо подходит для интерактивного трафика. Данная проблема решилась, когда был изобретен метод **коммутации пакетов**, который мы уже упоминали в главе 1. В сетях с коммутацией пакетов на размер блоков накладывается жесткое ограничение, что позволяет буферизировать блоки не на диске, а в памяти маршрутизатора. Сети с коммутацией пакетов удовлетворяют требованиям интерактивного трафика, поскольку ни один пользователь не может монополизировать линию на длительное время (измеряемое миллисекундами). Еще одно преимущество коммутации пакетов перед коммутацией сообщений продемонстрировано на рис. 2.34, *б* и *в*: первый пакет многопакетного сообщения пересылается еще до того, как второй пакет полностью получен, что уменьшает задержку и повышает скорость передачи. По этим причинам компьютерные сети обычно используют систему коммутации пакетов, иногда — коммутацию каналов, но никогда не применяют коммутацию сообщений.

Коммутация каналов отличается от коммутации пакетов по ряду факторов. Во-первых, при коммутации каналов связь между двумя абонентами должна быть установлена до начала передачи данных. Пакетная коммутация такого требования не предъявляет. Первый пакет отсылается, как только он появляется.

Результатом предварительной установки соединения при коммутации каналов является резервирование всей пропускной способности для конкретного сеанса. Все пакеты следуют установленным путем. Кроме всего прочего, это означает, что пакеты не могут прийти в неправильной последовательности. При коммутации пакетов единого пути для них не существует, они могут приходить независимо друг от друга по пути, определяемому состоянием сети в момент передачи. Соответственно, не исключено, что пакеты придут в неправильном порядке.

Пакетная коммутация более устойчива к сбоям. На самом деле, именно это свойство стало причиной изобретения данного метода. Если, например, выходит из строя один из коммутаторов, то все линии, подключенные к нему, также выходят из строя. Но при коммутации пакетов данные могут быть отправлены в обход «умершего» коммутатора.

Предварительное установление соединения дает возможность предварительно зарезервировать полосу пропускания. В этом случае при получении пакета его можно мгновенно переслать дальше. При коммутации пакетов полоса пропускания не резервируется, поэтому могут возникать очереди из пакетов, ожидающих пересылки.

Если полоса пропускания резервируется заранее, на линии не может возникнуть затор (если только не пытаться послать больше пакетов, чем ожидается). С другой стороны, сама попытка установить предварительное соединение может оказаться неудачной из-за перегрузки канала. То есть ситуация перегрузки в разных методах может возникать в разное время: при коммутации пакетов такое может случиться при передаче данных, а при коммутации каналов — при установлении соединения.

Если определенная линия уже зарезервирована для какого-то соединения, но по ней ничего не передается, то она простаивает. Ею не могут воспользоваться другие абоненты. При коммутации пакетов линия не простаивает, а значит, такой метод является более эффективным с точки зрения всей системы. Понимание того, чем приходится жертвовать в каждом из этих методов, необходимо, чтобы ощутить разницу между пакетной коммутацией и коммутацией каналов. В любом случае ищется какой-то компромисс между гарантией качества обслуживания при больших затратах ресурсов и отсутствием гарантии качества при малых затратах ресурсов.

Коммутация пакетов использует передачу с промежуточным хранением. Пакет сохраняется в памяти маршрутизатора, после чего пересылается дальше. При коммутации каналов биты просто непрерывной чередой следуют по линии связи. Промежуточное сохранение пакетов вносит задержки в процесс передачи.

Еще одним отличием является то, что коммутация каналов абсолютно прозрачна для пользователей. Отправитель и получатель могут использовать любую скорость передачи, любой формат данных или метод формирования кадров. Транспортная часть системы ничего не знает об этих подробностях и даже не интересуется ими. При коммутации пакетов транспортная служба определяет основные параметры. Можно привести довольно грубое сравнение с различием между шоссе и железной дорогой. При движении по шоссе пользователь сам определяет скорость, размер и тип автомобиля. При перемещении по железной дороге подобные вопросы решаются специальной службой. Благодаря такой прозрачности системы передавать голос, данные и факсы в телефонной системе можно одновременно.

Наконец, еще одним различием между двумя способами коммутации является политика оплаты услуг. Системы с коммутацией каналов традиционно взимают плату за расстояние передачи и время на линии. В мобильных телефонах расстояние роли не играет (кроме международных звонков), а время играет не очень значительную роль (ну, например, тариф с 2000 бесплатных минут дороже, чем тариф с 1000 бесплатных минут, причем иногда звонки в ночное время и в выходные являются льготными). В случае коммутации пакетов время на линии вообще не принимается в расчет, однако иногда взимается плата за трафик. С обычных пользователей провайдеры иногда берут просто ежемесячную абонентскую

плату, поскольку это проще для обеих сторон, однако магистральные транспортные службы взимают с местных провайдеров плату за объем трафика. Все различия сведены в табл. 2.5.

Таблица 2.5. Сравнительные характеристики коммутации каналов и коммутации пакетов

Параметр	Коммутация каналов	Коммутация пакетов
Установка соединения	Требуется	Не требуется
Выделенный «медный» путь	Да	Нет
Каждый пакет перемещается по одному и тому же пути	Да	Нет
Пакеты приходят в правильном порядке	Да	Нет
Критичность выхода из строя коммутатора	Да	Нет
Доступная пропускная способность	Фиксированная	Динамическая
Возможность занятости линии	Во время установки соединения	Для каждого пакета
Возможность простоя линии	Да	Нет
Передача с промежуточным хранением	Нет	Да
Прозрачность	Да	Нет
Оплата	За время на линии	За трафик

Обе системы коммутации довольно важны, поэтому мы скоро вернемся к ним и подробнее опишем различные используемые технологии.

Мобильная телефонная система

Традиционная телефонная система (даже если она в один прекрасный день полностью перейдет на многогигабитный оптоволоконный кабель) никогда не сможет удовлетворить потребности огромной группы пользователей — людей, находящихся в пути. Люди хотят получить возможность быть на связи буквально везде: в автомобиле, самолете, бассейне и т. д. Через несколько лет не останется такой точки на Земле, откуда нельзя было бы позвонить по телефону, послать e-mail и выйти в Интернет. По крайней мере, люди ожидают именно это. Следовательно, интерес к беспроводной телефонии будет продолжать расти. В следующих разделах мы рассмотрим подробности, касающиеся этой темы.

Беспроводные телефоны бывают двух типов: домашние радиотелефоны и мобильные телефоны (иногда называемые **сотовыми телефонами**). **Радиотелефоны** представляют собой устройства, состоящие из базовой станции и одной или нескольких переносных трубок. Они предназначены для использования внутри жилья или в непосредственной близости от него. Их никогда не объединяют

в сети, поэтому в дальнейшем мы их рассматривать не будем. Вместо этого мы более подробно рассмотрим мобильные системы связи, которые используются как для передачи речи, так и для обмена данными.

На данный момент можно выделить уже три разных поколения **мобильных телефонов**, осуществляющих:

1) аналоговую голосовую связь;
2) цифровую голосовую связь;
3) цифровую голосовую связь и обмен данными (Интернет, электронная почта и т. д.).

Хотя большая часть нашего обсуждения будет посвящена техническому устройству этих систем, нельзя не отметить тот интересный факт, что огромное влияние на процесс развития технологий этого типа оказали политические и экономические решения. Первая мобильная система была предложена американской компанией AT&T, которая, с согласия комиссии FCC, установила мобильную связь на всей территории Соединенных Штатов. В результате целая страна обрела единую (аналоговую) систему связи, и мобильный телефон, купленный, например, в Калифорнии, успешно работал в Нью-Йорке. А в Европе все получилось наоборот: когда туда пришла мобильная связь, каждая страна бросилась разрабатывать собственные системы, в результате чего проиграли все.

Однако Европа чему-то научилась на своих ошибках, и с появлением цифровых систем государственные телефонные службы объединились, чтобы создать единый стандарт (GSM), по которому могли бы работать любые европейские мобильные телефоны. К тому времени в США государство вышло из бизнеса, связанного со стандартизацией, поэтому новые цифровые мобильные системы стали заботой коммерческих структур. Это привело к тому, что разные производители стали выпускать разнотипные мобильные телефоны, и в США появились две основные (и одна поменьше) несовместимые цифровые мобильные телефонные системы.

Несмотря на изначальное лидерство США, Европа сейчас обошла Штаты по популярности мобильной связи. Одной из причин является, конечно, единая европейская система. Еще одна причина довольно забавна. Она связана с телефонными номерами. В США не различаются номера мобильных и стационарных телефонов. Поэтому нет никакой возможности узнать, набирая номер, например, (212) 234-5678, попадете вы на городской телефон (дешевый или вообще бесплатный звонок) или на сотовый (дорогой звонок). Чтобы люди не нервничали каждый раз, гадая, куда они звонят, телефонные компании заставили абонентов сотовой связи платить за входящие звонки. Но многих такое решение отпугнуло — люди стали бояться потратить большую сумму денег на один только прием входящих звонков. В Европе у мобильных телефонов номер начинается с особого кода (обычно это число в районе 800–900), поэтому его сразу можно узнать. Значит, можно установить обычное правило, принятое в телефонии: платит звонящий (за исключением международных звонков, где платят оба).

Третий фактор, оказавший большое влияние на популярность мобильных систем, — это широкое распространение телефонов с предоплатой разговоров (до 75 % в некоторых регионах). Их можно купить во многих магазинах, и это не

представляет особой сложности. Они могут быть заряжены, например, на 20 или 50 евро, а при снижении баланса до нуля их можно «перезарядить» с помощью секретного PIN-кода. Теперь такие мобильные телефоны есть у любого подростка, и родители могут быть на связи со своим чадом, не опасаясь при этом, что он самостоятельно наговорит по телефону на кругленькую сумму. Если мобильный телефон используется лишь эпизодически, то это обходится практически бесплатно, поскольку почти всегда можно найти тариф, на котором отсутствует абонентская плата или плата за входящие звонки.

Мобильные телефоны первого поколения: аналоговая передача речи

Однако хватит о политике и бизнесе. Поговорим о технологиях. Начнем наше рассмотрение с самых первых из них. Мобильные радиотелефоны эпизодически применялись в морском судоходстве и военной связи в первые десятилетия XX века. В 1946 году в Сент-Луи была установлена первая система автомобильных телефонов. Она имела один большой передатчик, расположенный на крыше высокого здания, и единственный канал приема и передачи данных. Чтобы начать разговор, нужно было нажать на кнопку, которая включала передатчик и отключала приемник. Такие системы, известные как **тангентные**, существовали в некоторых городах еще в конце 50-х. CB-радио, системы, используемые в такси и полицейских машинах, часто используют эту же технологию.

В 1960-х годах появилась **усовершенствованная система мобильной телефонной связи, IMTS** (Improved Mobile Telephone System). Она также использовала мощный (200-ваттный) передатчик, установленный на вершине горы, но уже имела два частотных канала: один для отправки, другой — для приема данных. Поэтому микрофонная кнопка уже была не нужна. Благодаря разделению входящих и исходящих каналов пользователи мобильных телефонов не могли слышать чужие разговоры (в отличие от тангентных систем, используемых в такси).

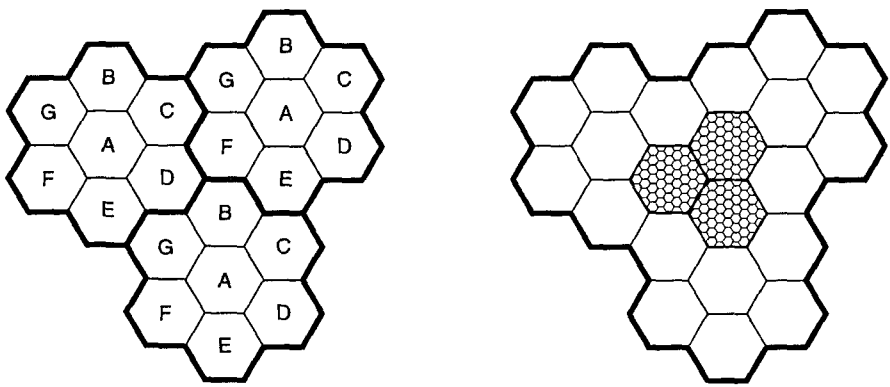
IMTS поддерживала 23 канала в диапазоне от 150 до 450 МГц. Из-за небольшого числа каналов пользователям часто подолгу приходилось ждать освобождения линии. К тому же из-за сильной мощности передатчика смежные системы должны были располагаться на расстоянии нескольких сотен километров друг от друга во избежание интерференции сигналов. В общем, из-за низкой емкости эта система была признана непрактичной.

Усовершенствованная мобильная телефонная связь (AMPS)

Все изменилось с появлением системы **усовершенствованной мобильной телефонной связи, AMPS** (Advanced Mobile Phone System), изобретенной компанией Bell Labs и впервые установленной в США в 1982 году. Она также использовалась в Англии, где называлась TACS, и в Японии — под именем MCS-L1. Несмотря на то что сейчас эта система уже ушла в прошлое, мы все же ее рассмотрим, поскольку многие ее фундаментальные свойства были напрямую унаследованы ее цифровым последователем, D-AMPS (для совместимости со старым оборудованием).

В любой мобильной телефонной системе географический регион охвата делится на **соты** (отсюда иногда применяемое название — «сотовые телефоны»). В AMPS размер сот составляет обычно от 10 до 20 км; в цифровых системах соты еще мельче. Каждая сота работает на своих частотах, не пересекающихся с соседними. Лежащая в основе телефонной системы AMPS идея разбиения территории на относительно небольшие ячейки и использования одних и тех же частот в различных (но не соседних) ячейках дает этой системе значительно большие возможности по сравнению с более ранними системами. В то время как в системе IMTS на территории диаметром 100 км для каждого звонка требовалась своя частота, система AMTS в той же области могла состоять из ста десятикилометровых сот и поддерживать от 5 до 10 звонков на одной и той же частоте в сильно удаленных друг от друга ячейках. Кроме того, небольшие размеры сот означают меньшую мощность, требующуюся для передатчиков, а значит, и меньшую стоимость устройств. Телефонные трубки имеют выходную мощность около 0,6 Вт, передатчики в автомобилях — около 3 Вт, что является максимальной мощностью, которую разрешает Федеральная комиссия связи США (Federal Communication Commission, FCC).

Идея повторного использования частоты проиллюстрирована на рис. 2.35, *а*. Соты имеют форму, близкую к окружности, однако на модели их легче представить в виде шестиугольников. В левой части рисунка все соты одного размера. Они объединены в группы по семь сот. Каждая буква соответствует определенному набору частот. Обратите внимание на то, что между ячейками с одинаковыми наборами частот располагается буфер примерно в две ячейки шириной, в котором данные частоты не используются — это обеспечивает хорошее разделение сигналов одинаковых частот и низкий уровень помех.



а *б*

Рис. 2.35. Повторное использование частоты (*а*); разбиение на микросоты (*б*)

Главная задача заключается в том, чтобы найти подходящие возвышенности для размещения антенн базовых станций. Для решения этой проблемы многие операторы связи заключили договоры с Римской католической церковью, поскольку последней принадлежит значительное количество высоких строений в различных странах. Удобно и то, что все они находятся под единым управлением.

Если в каком-нибудь регионе количество пользователей вырастает настолько, что система переполняется, то мощность передатчиков уменьшается, а переполненные соты разбиваются на соты меньшего размера (**микросоты**), как показано на рис. 2.35, *б*. Вопрос о размере ячеек обсуждается в (Hac, 1995).

В центре каждой ячейки располагается базовая станция (БС), с которой связываются все телефоны, находящиеся в ее зоне действия. Базовая станция состоит из компьютера и приемника/передатчика, соединенного с антенной. В небольших системах все базовые станции соединены с одним устройством, называемым **MTSO** (Mobile Telephone Switching Office — коммутатор мобильных телефонов) или **MSC** (Mobile Switching Center — мобильный коммутационный центр). Большой системе может потребоваться несколько коммутаторов, которые соединяются с коммутатором второго уровня, и т. д. Коммутаторы мобильных телефонов являются аналогами оконечных телефонных станций и в самом деле соединяются хотя бы с одним оконечным коммутатором обычной телефонной системы. Коммутаторы мобильных телефонов связываются с базовыми станциями, друг с другом и с коммутируемой телефонной сетью общего пользования (PSTN, Public Switched Telephone Network), построенной на основе коммутации пакетов.

В каждый момент времени мобильный телефон *логически* находится в зоне действия одной ячейки и управляется базовой станцией этой ячейки. Когда телефон физически покидает ячейку, его базовая станция замечает ослабление сигнала и опрашивает все окружающие станции, насколько хорошо они слышат сигнал этого телефона. Затем базовая станция передает управление данным телефоном ячейке, получающей от нее наиболее сильный сигнал, таким образом определяя ячейку, в которую переместился мобильный телефон. После этого телефон информируется о переходе в ведение новой БС, и если в этот момент ведется разговор, телефону будет предложено переключиться на новый канал (поскольку в соседних сотах одинаковые частотные каналы не используются). Подобный процесс называется **передачей** (handoff) и занимает около 300 мс. Назначение канала осуществляет коммутатор мобильных телефонов MTSO, являющийся центральным нервом системы. Базовые станции представляют собой всего лишь радиоретрансляторы.

Передача может осуществляться двумя способами. При **мягкой передаче** телефон переходит в ведение новой базовой станции еще до ухода со старой. При этом не происходит даже кратковременного пропадания связи. Недостатком такого метода является то, что в момент перехода с одной БС на другую телефон должен работать одновременно на двух частотах. Телефоны первого и второго поколения этого делать не умеют.

При **жесткой передаче** старая базовая станция обрывает связь с телефоном еще до того, как новая взяла его под свою опеку. Если последняя не может в течение какого-то времени наладить связь с телефоном (например, по причине отсутствия свободных частот), то разговор может оборваться. Пользователь, конечно, заметит это, но ничего не поделаешь — так иногда случается при использовании данной технологии.

Каналы

Система AMPS использует 832 дуплексных канала, каждый из которых состоит из пары симплексных каналов. 832 симплексных канала передачи располагаются

в диапазоне от 824 до 849 МГц, и еще 832 симплексных канала приема — от 869 до 894 МГц. Ширина каждого канала составляет 30 кГц. Таким образом, для разделения каналов в системе AMPS используется частотное уплотнение.

В диапазоне 800 МГц длина радиоволн составляет около 40 см. Такие радиоволны распространяются строго по прямой линии. Они поглощаются деревьями и отражаются от поверхности земли и зданий. Может случиться так, что сигнал с мобильного телефона достигнет базовой станции по прямому пути, но, кроме того, с небольшим запозданием попадет на ту же станцию, отразившись от земли или здания. Такой эффект может привести к появлению эха или искажению сигнала (многолучевое затухание). Иногда можно услышать отдаленный разговор, отразившийся несколько раз.

Все 832 канала можно разделить на четыре категории:

1. Управляющие каналы (от базы к мобильному телефону) для управления системой.
2. Пейджинговые каналы (от базы к мобильному телефону) для передачи сообщений мобильным пользователям.
3. Каналы доступа (двунаправленные) для установления соединения и назначения каналов.
4. Каналы данных (двунаправленные) для передачи голоса, факса или данных.

Для управления резервируется 21 канал. Они прошиваются в программируемом запоминающем устройстве (ППЗУ) каждого телефона. Поскольку одни и те же частоты не могут использоваться в соседних сотах, то число голосовых каналов, доступных в пределах одной ячейки, значительно меньше 832 — обычно около 45.

Управление вызовом

Каждый мобильный телефон в системе AMPS снабжается 32-разрядным порядковым номером и 10-значным телефонным номером, которые записываются в ППЗУ телефона. Телефонный номер состоит из 3-значного кода области, занимающего 10 бит, и 7-значного номера абонента, занимающего 24 бита. При включении телефон сканирует запрограммированный список из 21 управляющего канала, в котором он ищет наиболее сильный сигнал.

Затем телефон передает в эфир свой 32-разрядный порядковый номер и 34-разрядный телефонный номер. Как и вся управляющая информация в системе AMPS, этот пакет посылается в цифровой форме несколько раз с применением помехоустойчивого кодирования, хотя сами голосовые каналы являются аналоговыми.

Когда базовая станция слышит этот сигнал, она передает сообщение коммутатору MTSO, который фиксирует появление нового пользователя, а также информирует «домашний» коммутатор абонента о его новом местоположении. Обычно мобильный телефон регистрируется примерно каждые 15 минут.

Чтобы позвонить с мобильного телефона, его владелец включает телефон, вводит номер и нажимает клавишу SEND. При этом телефон посылает набранный телефонный номер вместе со своими идентификаторами по каналу доступа. Если при этом происходит коллизия, то телефон повторяет попытку позже. Когда базовая станция получает запрос, она информирует об этом коммутатор. Если звоня-

щий является клиентом оператора связи, которому принадлежит данный коммутатор (или одного из ее партнеров), тогда коммутатор ищет для него свободный канал. Если такой канал находится, то номер этого канала посылается обратно по управляющему каналу. Затем мобильный телефон автоматически переключается на выбранный голосовой канал и ждет, пока тот, кому звонят, ответит.

Входящие звонки обрабатываются иначе. Находящиеся в режиме ожидания телефоны постоянно прослушивают пейджинговый канал, ожидая адресованных им сообщений. Когда поступает звонок на мобильный телефон (с обычного или другого мобильного телефона), то пакет посылается на «домашний» коммутатор вызываемого, которому должно быть известно текущее местонахождение абонента. Этот пакет пересылается на базовую станцию в его текущей ячейке, которая посылает по пейджинговому каналу сообщение типа: «Элемент 14, вы здесь?». При этом телефон, которому звонят, по управляющему каналу отвечает: «Да». Тогда базовая станция ему сообщает: «Элемент 14, вам звонок по каналу 3». После этого сотовый телефон переключается на канал 3 и начинает издавать звуковые сигналы (или проигрывать мелодию, которую владельцу подарили на день рождения).

Второе поколение мобильных телефонов: цифровая передача голоса

Первое поколение сотовых телефонных систем было аналоговым. Второе поколение является цифровым. Как не было никаких четких стандартов в первом поколении мобильных телефонов, так не появились они и ко второму поколению. Сейчас используются четыре системы второго поколения: D-AMPS, GSM, CDMA и PDC. Далее мы обсудим первые три из них. PDC нашла применение только в Японии и является, на самом деле, модификацией D-AMPS, направленной на сохранение совместимости с японским аналоговым оборудованием первого поколения. Название **PCS** (Personal Communications Services — персональная служба связи) иногда используется в литературе по маркетингу и означает систему второго поколения (цифровую, разумеется). Изначально так назывался телефон, работающий в диапазоне 1900 МГц, впрочем, сейчас различия почти стерлись.

D-AMPS — цифровая усовершенствованная мобильная связь

Вторым поколением AMPS является полностью цифровая система D-AMPS. Она описывается международным стандартом IS-54 и его последователем — IS-136. Система D-AMPS была разработана таким образом, чтобы она могла успешно сосуществовать с AMPS и мобильные телефоны первого и второго поколения могли работать одновременно в одной и той же соте.

В частности, D-AMPS использует те же 30-герцевые каналы, что и AMPS. Они располагаются в том же диапазоне, то есть может получиться так, что какой-то канал будет аналоговым, а соседние с ним каналы — цифровыми. В зависимости от конкретного набора телефонов в данной ячейке ее коммутатор определяет, какие каналы цифровые, какие аналоговые, и может динамически менять их тип в зависимости от того, какие телефоны попадают или выходят из зоны действия

базовой станции ячейки. Когда D-AMPS была представлена как новая служба, для нее был выделен дополнительный диапазон, с расчетом на увеличение нагрузки. Исходящие каналы расположили на частотах 1850–1910 МГц, а соответствующие входящие каналы — на частотах 1930–1990 МГц. Как и в AMPS, каналы парные. В этой полосе длина волн составляет 16 см, поэтому стандартная антенна размером в четверть длины волны будет размером всего лишь 4 см, что дает возможность создать более компактные телефоны. Тем не менее многие телефоны D-AMPS могут использовать оба диапазона (как 850, так и 1900 МГц), что позволяет использовать увеличенный набор доступных каналов.

В мобильном телефоне системы D-AMPS голосовой сигнал захватывается микрофоном, оцифровывается и сжимается при помощи более сложной модели, чем дельта-модуляция и схема предсказания, которые мы изучали ранее. Метод компрессии в данном случае принимает в расчет особенности человеческого голоса, сжимая речь со стандартных 56 Кбит/с (PCM-кодирование) до 8 Кбит/с и даже меньше. Сжатие производится специальной схемой, называемой **вокодером** (Bellamy, 2000), прямо в телефоне, а не на базовой или коммутационной станции. Это уменьшает размеры информации, которую необходимо передать в эфир. При использовании стационарной телефонии нет никакого смысла в сжатии данных в самом телефонном аппарате, поскольку уменьшение трафика в локальной линии никак не влияет на общую емкость системы.

Когда же речь идет о мобильной связи, то в оцифровке и сжатии данных в самой трубке есть значительная выгода: достаточно сказать, что три абонента D-AMPS могут одновременно использовать одну и ту же пару частотных каналов за счет мультиплексирования с разделением времени. Каждая пара частот поддерживает скорость 25 кадров/с (40 мс на кадр). Кадры состоят из шести временных интервалов по 6,67 мс, как показано на рис. 2.36 для самой низкочастотной канальной пары.

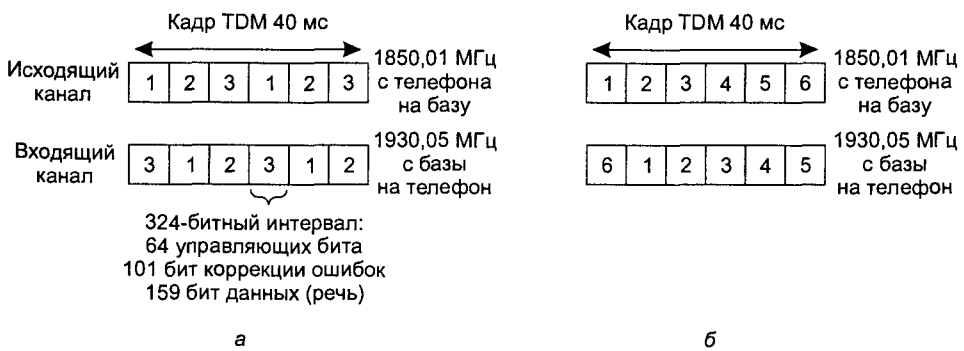


Рис. 2.36. Канал D-AMPS с тремя абонентами (а); канал D-AMPS с шестью абонентами (*б*)

Каждый кадр обслуживает трех пользователей, которые поочередно занимают исходящий и входящий каналы. Во время первого кадрового интервала (рис. 2.36, *а*), например, пользователь 1 может передавать данные на базовую станцию, а в это время пользователь 3 может принимать данные. Кадровый интервал состоит из 324 бит, из них 64 используются для организации защитного интерва-

ла, синхронизации и функций управления. Таким образом, пользователю предоставляется 260 бит. Из них 101 используется для исправления ошибок при передаче по зашумленному эфиру, поэтому в чистом виде для полезных данных остается лишь 159 бит. При скорости 50 интервалов в секунду пропускная способность, доступная для передачи сжатой речевой информации, составляет около 8 Кбит/с, то есть 1/7 стандартной пропускной способности PCM.

Использование улучшенных алгоритмов сжатия может позволить уложить речь в 4 Кбит/с, в этом случае один кадр может использоваться одновременно шестью абонентами, как показано на рис. 2.36, *б*. С точки зрения операторов мобильной связи, возможность сжатия данных в 3–6 раз относительно AMPS — это большая победа. Этим объясняется популярность «персональных служб связи». Конечно, качество звука при 4 Кбит/с не сравнить с 56 Кбит/с, однако некоторые операторы, тем не менее, рекламируют высококачественный звук, который можно якобы сравнить со звуком Hi-Fi аппаратуры. Но должно быть очевидно, что канал 8 Кбит/с никогда не даст даже качества древнего модема на 9600 бит/с.

Структура управления D-AMPS довольно сложна. Не вдаваясь в подробности, можно сказать, что группы из 16 кадров формируют суперкадр, и некоторая часть служебной информации появляется ограниченное количество раз в суперкадре. Используются шесть основных управляющих каналов: конфигурация системы, управление в реальном и модельном (не реальном) времени, пейджинговые функции, ответы на запросы доступа и короткие сообщения. Но концептуально работа D-AMPS не отличается от работы AMPS. Когда телефон включен, он находится в контакте с базовой станцией, сообщая о себе и прослушивая управляющий канал на предмет входящих звонков. Обнаружив новый телефон, коммутатор информирует домашнюю базу абонента о его местонахождении, благодаря чему звонки могут быть корректно маршрутизированы.

Системы AMPS и D-AMPS различаются методом передачи сигнала телефона с одной базовой станции на другую. В AMPS этим занимается коммутатор, не привлекая никакие мобильные устройства. Как видно из рис. 2.36, в D-AMPS треть времени мобильный телефон занимается не передачей и не приемом информации. Он использует пустые кадровые интервалы для измерения качества линии. Когда он обнаруживает, что сигнал пропадает, то жалуется на это коммутатору, который разрывает соединение с текущей базовой станцией. В это время телефон может попытаться найти станцию с более сильным сигналом. Как и в AMPS, на передачу уходит около 300 мс. Метод, используемый в D-AMPS, называется **передачей с помощью телефона, MAHO** (Mobile Assisted HandOff).

GSM — глобальная система мобильной связи

Система D-AMPS широко распространена в США и (в несколько измененной форме) в Японии. Практически весь остальной мир использует систему под названием **GSM** (Global System For Mobile Communications — глобальная система мобильной связи). Впрочем, GSM начинает проникать и в США. В первом приближении, система GSM подобна D-AMPS. И та, и другая — сотовые системы. И там, и там применяется частотное уплотнение. Каждый телефон передает данные на одной частоте, а получает — на другой (последняя выше первой: 80 МГц

в D-AMPS и 55 МГц в GSM). В обеих системах пара частотных каналов разбивается с помощью временного уплотнения на кадровые интервалы, используемые несколькими абонентами. Однако каналы GSM значительно шире каналов AMPS (200 кГц против 30 кГц) и обслуживают относительно мало дополнительных пользователей (8 против 3), в результате чего в GSM скорость передачи данных одним пользователем оказывается гораздо выше, чем в D-AMPS.

Далее мы рассмотрим лишь основные свойства GSM. А печатный вариант стандарта GSM занимает свыше 5000 (sic!) страниц. Основная часть текста относится к описанию инженерных аспектов системы, в частности, устройства приемников, обрабатывающих многолучевое распространение сигналов, синхронизации приемников и передатчиков. Ни о том, ни о другом мы не сможем рассказать в этой книге.

Итак, каждая полоса частот имеет ширину 200 кГц. Система GSM имеет 124 пары симплексных каналов, как показано на рис. 2.37. Ширина пропускания каждого симплексного канала составляет 200 кГц. Канал поддерживает 8 отдельных соединений при помощи временного уплотнения. Каждой активной в данный момент базовой станции назначен один кадровый интервал на пару каналов. Теоретически, каждая сота может иметь до 992 каналов, однако многие из них сознательно делают недоступными во избежание конфликтов с соседними сотами. На рис. 2.36 восемь заштрихованных кадровых интервалов принадлежат одному и тому же соединению, по четыре в каждом направлении. Прием и передача происходят в разных интервалах, поскольку аппаратура GSM не может работать одновременно в двух режимах, и на перестройку требуется некоторое время. Если мобильной станции присвоен диапазон 980,4/935,4 МГц и кадровый интервал 2 хочет осуществить передачу на базовую станцию, он воспользуется нижним набором заштрихованных интервалов (а также последующими), размещая в каждом из них порцию данных. Так будет продолжаться до тех пор, пока не будут посланы все данные.

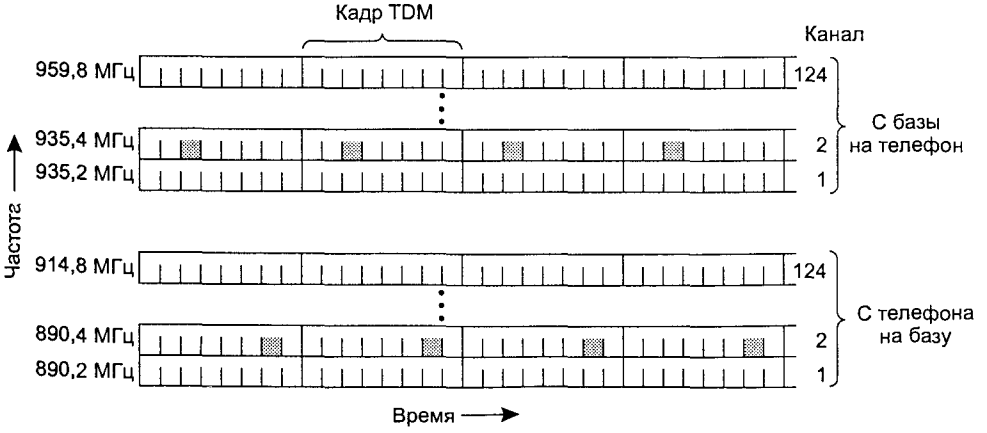


Рис. 2.37. GSM имеет 124 частотных канала, в каждом из них 8-интервальная система с разделением времени

Интервалы TDM, изображенные на рис. 2.37, являются частью сложной иерархической системы кадров. Каждый интервал имеет специфическую структуру, как и их группы. Упрощенная иерархия изображена на рис. 2.38. Мы видим здесь, что интервал TDM состоит из 148-битного кадра данных, который занимает канал на 577 мкс (включая защитный интервал длиной 30 мкс). Кадры данных начинаются и заканчивается тремя нулями, это делается для их разграничения. В них также входят 57-битные информационные (*Information*) поля, в каждом из которых присутствует контрольный бит проверки содержимого (голос/данные). Между информационными полями имеется 26-битное поле синхронизации (*Sync*), которое используется приемником для синхронизации с границей кадра передатчика.

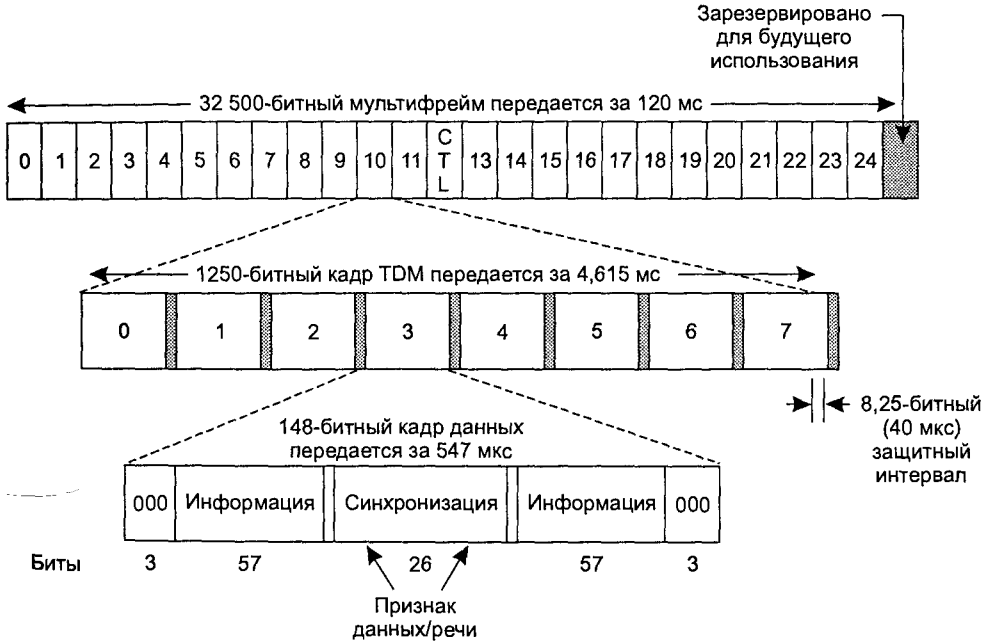


Рис. 2.38. Часть иерархической структуры кадров GSM

Кадр данных передается за 547 мкс, но передатчику разрешается посылать данные только через каждые 4,615 мс, поскольку он делит канал с семью другими станциями. Общая скорость каждого канала составляет 270 883 бит/с. Она делится между 8 пользователями. Итого получается 33,854 Кбит/с, что более чем в два раза превышает 16,2 Кбит/с D-AMPS (324 бита 50 раз в секунду). Тем не менее, как и в AMPS, на накладные расходы тратится большая часть пропускной способности, и в итоге на одного пользователя приходится 24,7 Кбит/с (перед началом исправления ошибок). После исправления ошибок остается 13 Кбит/с, с помощью которых нужно передать голос. Это уже почти в два раза лучше, чем D-AMPS (за счет использования соответственно увеличенной пропускной способности).

Как видно на рис. 2.38, 8 кадров данных образуют один кадр TDM, а 26 кадров TDM образуют 120-миллисекундный мультикадр. В мультифрейме двенадцатый интервал используется для служебных целей, а двадцать пятый зарезервирован для будущего использования, поэтому для пользовательского трафика остается только 24 интервала.

Тем не менее, в дополнение к 26-интервальному мультифрейму, показанному на рис. 2.38, используется еще и 51-интервальный мультифрейм (не показан на рисунке). Некоторые интервалы нужны для управляющих каналов. **Широковещательный управляющий канал** представляет собой непрерывный поток, исходящий от базовой станции, в котором содержатся ее идентификационная информация и статус канала. Все мобильные устройства производят мониторинг мощности сигнала, по которому они определяют моменты перехода в ведение новой БС.

Выделенный управляющий канал используется для поиска мобильного телефона, обновления информации о нем, регистрации и установки соединения. В частности, каждая БС содержит базу данных телефонов, находящихся в текущий момент под ее юрисдикцией. Информация, необходимая для обновления этой базы, передается по выделенному управляющему каналу.

Наконец, есть еще **общий управляющий канал**, разделяемый на три логических подканала. Первый из них — **пейджинговый канал**, с помощью которого базовая станция сообщает о входящих звонках. Каждый мобильный телефон постоянно прослушивает его в ожидании звонка, на который он должен ответить. Второй — **канал случайного доступа**, позволяющий пользователям запросить интервал в выделенном управляющем канале. Если два запроса сталкиваются (коллизия), они искажаются, и им приходится впоследствии осуществлять повторные попытки. Используя выделенный управляющий канал, мобильный телефон может инициировать исходящий звонок. Присвоенный интервал объявляется при помощи третьего подканала — **канала предоставления доступа**.

CDMA — множественный доступ с кодовым разделением каналов

D-AMPS и GSM — это довольно традиционные системы. Они используют частотное и временное уплотнение для разделения спектра на каналы и разделения каналов на интервалы. Однако есть еще одна система из этой серии под названием **CDMA** (Code Division Multiple Access — множественный доступ с кодовым разделением каналов), которая работает совершенно по-другому. Когда CDMA была впервые предложена, реакция представителей соответствующей промышленности напоминала реакцию королевы Изабеллы, когда к ней пришел Колумб и сказал, что он достиг Индии, поплыв в направлении, противоположном нужному. Так или иначе, благодаря упорству единственной компании, Qualcomm, CDMA теперь признается не только полноценной системой мобильной связи, но и лучшей из существующих систем третьего поколения. Она также используется в США при работе с оборудованием второго поколения, конкурируя с D-AMPS. Например, персональная служба связи Sprint использует CDMA, а AT&T Wireless — D-AMPS. CDMA описывается международным стандартом IS-95, и иногда на эту

систему ссылаются именно таким образом. Также используется название торговой марки — **cdmaOne**.

Да, CDMA полностью отличается от AMPS, D-AMPS и GSM. Вместо разделения доступного частотного диапазона на сотни узких каналов в CDMA каждая станция может при передаче все время пользоваться полным спектром частот. Одновременный множественный доступ обеспечивается за счет применения теории кодирования. CDMA также отдыхает от мысли о том, что одновременно пришедшие кадры должны портиться. Вместо этого предполагается, что сигналы добавляются линейно.

Прежде чем разбирать алгоритм работы, рассмотрим следующую аналогию. Представьте себе зал ожидания в аэропорту. Множество пар оживленно беседуют. Временное уплотнение можно сравнить с ситуацией, когда все люди находятся в центре зала и говорят по очереди. Частотное уплотнение мы сравним с ситуацией, при которой люди находятся в разных углах и ведут свои разговоры, которые не слышны другим. Это происходит одновременно, но независимо. Для CDMA лучше всего подходит сравнение с ситуацией, когда все в центре зала, однако каждая пара говорящих использует свой язык общения. Франкоговорящие промывают косточки всем остальным, воспринимая чужие разговоры как шум. Таким образом, ключевой идеей CDMA является выделение полезного сигнала при игнорировании всего остального. Далее следует слегка упрощенное описание технологии CDMA.

В CDMA каждый битовый интервал разбивается на m коротких периодов, называемых **элементарными сигналами**, или чипами (chip). Обычно в битовом интервале помещаются 64 или 128 элементарных сигналов. В нашем примере мы будем допускать, что битовый интервал содержит только 8 элементарных сигналов на бит, и это надо воспринимать лишь как упрощение.

Каждой станции соответствует уникальный m-битный код, называющийся **элементарной последовательностью**. Чтобы передать 1 бит, станция посылает свою элементарную последовательность. Чтобы передать бит со значением 0, нужно отправить вместо элементарной последовательности ее дополнение (все единицы последовательности меняются на нули, а все нули — на единицы). Никакие другие комбинации передавать не разрешается. Таким образом, если $m = 8$ и станции присвоена 8-битная элементарная последовательность 00011011, то бит со значением «1» передается кодом 00011011 (что соответствует элементарной последовательности), а бит со значением «0» передается кодом 11100100 (дополнение элементарной последовательности).

Оправдать возросшее в m раз количество информации, которое необходимо передавать (чтобы скорость составила b бит/с, нужно отправлять mb элементарных сигналов в секунду), можно только за счет увеличения в m раз пропускной способности. Таким образом, CDMA является одной из форм связи с расширенным спектром (предполагается, что никаких изменений в методах модуляции и кодирования не производилось). Если имеется полоса шириной 1 МГц, на которой работают 100 станций, то при частотном уплотнении каждая из них получила бы свои 10 кГц и работала бы со скоростью 10 Кбит/с (предположим, используется 1 бит/Гц). При CDMA все станции используют всю ширину диапазона

(1 МГц), так что скорость передачи элементарных сигналов составляет 1 Мчип/с. При кодировании одного бита элементарными последовательностями, число которых менее 100, эффективная пропускная способность CDMA выше, чем FDM, причем проблема размещения каналов решена.

Из педагогических соображений удобнее использовать биполярную запись и двоичный 0 обозначать −1, а двоичную 1 обозначать +1. В скобках будем показывать элементарные последовательности. Так, единичный бит для станции *A* будет выглядеть как (−1 −1 −1 +1 +1 −1 +1 +1). На рис. 2.39, *а* мы покажем элементарные последовательности четырех станций. На рис. 2.39, *б* изображены они же, но в биполярной нотации.

A: 0 0 0 1 1 0 1 1	A: (−1 −1 −1 +1 +1 −1 +1 +1)
B: 0 0 1 0 1 1 1 0	B: (−1 −1 +1 −1 +1 +1 +1 −1)
C: 0 1 0 1 1 1 0 0	C: (−1 +1 −1 +1 +1 +1 −1 −1)
D: 0 1 0 0 0 0 1 0	D: (−1 +1 −1 −1 −1 −1 +1 −1)
а	*б*

Шесть примеров:

− − 1 −	C	S_1 = (−1 +1 −1 +1 +1 +1 −1 −1)
− 1 1 −	B + $\overline{\text{C}}$	S_2 = (−2 0 0 0 +2 +2 0 −2)
1 0 − −	A + $\overline{\text{B}}$	S_3 = (0 0 −2 +2 0 −2 0 +2)
1 0 1 −	A + B + C	S_4 = (−1 +1 −3 +3 +1 −1 −1 +1)
1 1 1 1	A + B + C + D	S_5 = (−4 0 −2 0 +2 0 +2 −2)
1 1 0 1	A + B + $\overline{\text{C}}$ + D	S_6 = (−2 −2 0 −2 0 −2 +4 0)

в

S_1 C = (1 +1 +1 +1 +1 +1 +1 +1) /8 = 1
S_2 C = (2 +0 +0 +0 +2 +2 +0 +2) /8 = 1
S_3 C = (0 +0 +2 +2 +0 −2 +0 −2) /8 = 0
S_4 C = (1 +1 +3 +3 +1 −1 +1 −1) /8 = 1
S_5 C = (4 +0 +2 +0 +2 +0 −2 +2) /8 = 1
S_6 C = (2 −2 +0 −2 +0 −2 −4 +0) /8 = −1

г

Рис. 2.39. Двоичные элементарные последовательности для четырех станций (*а*); биполярные элементарные двоичные последовательности (*б*); шесть примеров передачи (*в*); восстановление сигнала станции C (*г*)

Каждая станция имеет собственную уникальную элементарную последовательность. Обозначим символом S вектор длины m для станции S, а символом $\overline{S}$ — дополнение S. Все элементарные последовательности попарно **ортогональны**. Мы имеем в виду, что нормированное скалярное произведение двух различных элементарных последовательностей S и T (пишется $S \bullet T$) равно 0. Известно, как генерировать такие последовательности с помощью метода, известного как **коды Уолша**. Используя математическую запись, можно выразить сказанное ранее таким образом:

$$S \bullet T \equiv \frac{1}{m}\sum_{i=1}^{m} S_i T_i = 0 \qquad (2.4)$$

Попросту говоря, сколько одинаковых пар, столько и разных. Это свойство ортогональности мы строго докажем чуть позже. Обратите внимание: если $S \bullet T = 0$, то и $S \bullet \overline{T}$ также равно 0. Нормированное скалярное произведение любой элементарной последовательности на саму себя равно 1:

$$S \bullet S = \frac{1}{m}\sum_{i=1}^{m} S_i S_i = \frac{1}{m}\sum_{i=1}^{m} S_i^2 = \frac{1}{m}\sum_{i=1}^{m} (\pm 1)^2 = 1.$$

Это действительно так, поскольку каждое из m слагаемых суммы равно 1, поэтому вся сумма равна m. Обратите также внимание на то, что $S \bullet \overline{S} = -1$.

В течение каждого битового интервала станция может либо передавать 1, посылая свою элементарную последовательность, либо передавать 0, посылая дополнение к последовательности, либо может молчать и ничего не передавать. Предположим, что все станции синхронизировались во времени, то есть все последовательности начали передаваться в один и тот же момент.

Когда две или более станции пытаются осуществить одновременную передачу, их биполярные сигналы линейно складываются. Например, если при передаче одного элементарного сигнала три станции послали +1, а одна послала −1, то в результате получится +2. Можно рассматривать это как сложение напряжений: три станции имеют на выходе +1 В, а одна имеет на выходе −1 В. В результате сложения получаем +2 В.

На рис. 2.39, *в* изображено шесть примеров передачи, в которой одновременно принимают участие одна или несколько станций. В первом примере *C* передает единичный бит, поэтому мы просто получаем элементарную последовательность этой станции. Во втором примере и *B*, и *C* передают единичные биты, в результате чего мы получаем сумму их биполярных последовательностей, а именно:

$$(-1\ -1\ +1\ -1\ +1\ +1\ +1\ -1) + (-1\ +1\ -1\ +1\ +1\ +1\ -1\ -1) = (-2\ 0\ 0\ 0\ +2\ +2\ 0\ -2).$$

В третьем примере станция *A* посылает 1, а станция *B* посылает 0. Прочие молчат. В четвертом примере *A* и *C* посылают 1, тогда как *B* посылает 0. В пятом примере все четыре станции посылают 1. Наконец, в последнем случае *A*, *B* и *D* посылают единичный бит, а *C* посылает нулевой. Обратите внимание на то, что каждой из шести последовательностей (от S_1 до S_6), представленных на рис. 2.39, *в*, соответствует один битовый интервал.

Чтобы восстановить исходный битовый поток каждой из станций, приемник должен заранее знать элементарные последовательности всех передатчиков, с которыми он работает. Восстановление осуществляется путем вычисления нормированного скалярного произведения принятой последовательности (то есть линейной суммы сигналов всех станций) и элементарной последовательности той станции, чей исходный сигнал восстанавливается. Если принята элементарная последовательность *S* и приемник пытается понять, что передала станция с элементарной последовательностью *C*, то производится вычисление нормированного скалярного произведения $S \bullet C$.

Чтобы понять, как это все работает, давайте представим себе эти две станции, *A* и *C*. Пусть обе передают единичный бит, в то время как станция *B* передает ну-

левой бит. Приемник получает сумму сигналов, которая равна $S = A + \overline{B} + C$ и вычисляет произведение:

$$S \bullet C = (A + \overline{B} + C) \bullet C = A \bullet C + \overline{B} \bullet C + C \bullet C = 0 + 0 + 1 = 1.$$

Первые два слагаемые равны нулю, потому что все пары элементарных последовательностей тщательно подбирались такими, чтобы они были ортогональными, см. выражение (2.4). Теперь должно быть понятно, почему это условие должно быть наложено на элементарные последовательности.

Можно представить себе эту задачу и по-другому. Допустим, приемник получил вместо суммы сигналов отдельные сигналы. В этом случае приемник будет вычислять скалярные произведения каждого из них по отдельности, а результаты складывать. Благодаря свойству ортогональности, все скалярные произведения, кроме $C \bullet C$, равны 0. Сложение с последующим вычислением скалярного произведения равносильно суммированию скалярных произведений.

Обратимся снова к шести примерам, показанным на рис. 2.39, *в*. Конкретный результат декодирования этих последовательностей представлен на рис. 2.39, *г*. Допустим, приемник заинтересован в извлечении потока битов, посланного станцией C, из всех шести последовательностей S_1–S_6. Для этого он вычисляет каждый бит путем суммирования парных произведений принятой последовательности (S) и вектора C (см. рис. 2.39, *б*), затем деления результата на 8 (так как $m = 8$ в данном случае). Как видите, каждый раз находится верный бит. Это так же просто, как говорить по-французски!

В идеальной система CDMA без шума емкость (то есть допустимое количество станций) может быть сколь угодно большим, как и емкость идеального бесшумного канала Найквиста может увеличиваться за счет повышения количества бит на отсчет. На практике, конечно же, физические ограничения очень сильно уменьшают емкость системы. Во-первых, мы предполагали, что все последовательности синхронизированы по времени. На самом же деле точную синхронизацию обеспечить невозможно. Единственное что можно сделать, — это организовать форсирование приемником отправки со стороны передатчика достаточно длинной элементарной последовательности, по которой приемник мог бы осуществить синхронизацию. Все остальные (несинхронизированные) посылки при этом рассматриваются как случайный шум. Если их не очень много, базовый алгоритм декодирования работает неплохо. С наложением элементарных последовательностей на шумовой фон связана довольно обширная теория (см. Pickholtz и др., 1982). Как нетрудно догадаться, чем длиннее элементарная последовательность, тем выше вероятность ее корректного детектирования на фоне шума. Для повышения надежности битовая последовательность может использовать код с коррекцией ошибок. Элементарные последовательности никогда не используют коррекцию ошибок.

Еще одним очевидным допущением, которым мы пользовались в наших рассуждениях, является предположение о том, что мощности всех станций такие же, как воспринимаемые приемником. Система CDMA обычно используется в беспроводной связи, где всегда присутствует базовая стационарная станция и множество мобильных станций, расположенных на разных расстояниях от нее. Уровни мощности, воспринимаемые приемником, конечно, зависят от того, на-

сколько далеко находятся передатчики. Хорошим эвристическим правилом является правило компенсации мощностей: чем слабее сигнал, принимаемый мобильным телефоном от базовой станции, тем мощнее должен быть его исходящий сигнал. Другими словами, мобильная станция, получающая слабый сигнал от базовой станции, будет посылать более мощный сигнал, чем станция, получающая мощный сигнал с БС. Мощности могут также контролироваться базовыми станциями, выдающими команды мобильным станциям увеличить или уменьшить свою мощность.

Еще мы предполагали, что приемник знает, кто отправляет ему данные. В принципе, имея достаточно мощные вычислительные возможности, базовая станция может слушать одновременно всех отправителей и исполнять алгоритм декодирования параллельно для всех передатчиков. Но об этом проще говорить, чем реализовывать. В CDMA есть еще много сложных вещей, которые мы опустили в нашем кратком рассказе. Тем не менее, это хорошо продуманная схема, которая все шире применяется в беспроводной связи. Стандартной полосой CDMA является 1,25 МГц (против 30 кГц в D-AMPS и 200 кГц в GSM), и в этой полосе система может обслуживать гораздо больше пользователей, чем любая другая система. При этом каждому пользователю предоставляется пропускная способность, которая, по крайней мере, не хуже, чем в GSM, а зачастую даже лучше.

Инженеры, которые хотят получить более полное представление о CDMA, могут обратиться к книге (Lee and Miller, 1998). Альтернативная схема, в которой распределение осуществляется не по частотам, а по времени, описано в (Crespo и др., 1995). Еще одну схему можно найти в (Sari и др., 2000). Для чтения всех этих книг необходимо обладать знаниями теории связи.

Мобильные телефоны третьего поколения: цифровая речь и данные

Каким будет будущее мобильной телефонии? Давайте попробуем разобраться. Развитием этой отрасли движет большое количество факторов. Во-первых, объем передаваемых данных уже превышает объем передаваемой речи в стационарных сетях, и первый показатель растет экспоненциально, тогда как последний растет довольно вяло. Многие эксперты предрекают такое же будущее и мобильным сетям: трафик данных превысит голосовой трафик. Во-вторых, компьютерная индустрия и индустрии телефонии и развлечений уже стали полностью цифровыми и быстро объединяются. Многие восхищаются компактностью и малым весом портативного устройства, которое выступает в качестве телефона, проигрывателя компакт-дисков, DVD-проигрывателя, терминала для электронной почты, обладает веб-интерфейсом, возможностями текстового редактора, включает в себя электронные игры и многое другое. С его помощью можно без всяких проводов в любой точке мира получить высокоскоростной доступ в Интернет. Все это называется третьим поколением мобильной телефонии. Дополнительную информацию см. (Huber и др., 2000; Sarikaya, 2000).

Еще в 1992 году международный союз телекоммуникаций, ITU, сделал попытку конкретизировать и реализовать эти мечты и выпустил проект под назва-

нием **IMT-2000**, где IMT означало «международная мобильная связь» (International Mobile Telecommunications). Что касается числа 2000, то оно нужно было для трех вещей: во-первых, оно указывало на год, в котором задумывалось ввести в строй этот проект; во-вторых, именно на такой частоте (в мегагерцах) должна была работать система; в-третьих, предполагалось установить такую ширину полосы (в килогерцах).

Ни один из трех пунктов осуществлен не был. В 2000 году система реализована не была. ITU рекомендовал правительствам всех стран зарезервировать частоту 2000 МГц (2 ГГц) для международного роуминга. Рекомендации последовал только Китай. Наконец, в какой-то момент осознали, что невозможно выделить каждому пользователю пропускную способность в 2 Мбит/с, особенно учитывая *особую* мобильность многих из них (просто нереально с достаточно высокой скоростью осуществлять передачу с одной базовой станции на другую). Более реалистично выглядит выделение 2 Мбит/с стационарным абонентам, которые сидят дома (в этом случае такая система будет серьезным конкурентом ADSL), 384 Кбит/с для людей, которые не спеша прогуливаются по парку, и 144 Кбит/с — для связи с абонентами, движущимися в автомобилях. Тем не менее, вокруг **3G**, как называют третье поколение мобильной связи, кипит бурная деятельность. Третье поколение еще не оправдало в полной мере тех надежд, которые с ним связывали, однако вскоре несомненно оправдает.

Вот основные сервисы, для предоставления которых задумывалась сеть IMT-2000:

1. Высококачественная передача речи.
2. Обмен сообщениями (замена e-mail, факса, SMS, чата и т. д.).
3. Мультимедиа (проигрывание музыки, видео, фильмов, телевидения и т. д.).
4. Доступ в Интернет (включая просмотр страниц с аудио- и видеоинформацией).

В качестве дополнительных услуг могут быть видеоконференции, телепрезентации, групповые электронные игры, мобильная коммерция (использование мобильного телефона для оплаты покупок). Более того, все эти сервисы должны быть доступны по всему миру (с автоматическим соединением через спутник в тех местах, где станционарная сеть отсутствует) на основе постоянного подключения и с гарантированным качеством обслуживания.

ITU задумывал IMT-2000 как единую технологию, чтобы производители могли выпустить универсальное устройство, которое можно было бы продавать по всему миру (как компьютеры и проигрыватели компакт-дисков и не в пример мобильным телефонам и телевизорам). Одна стандартная технология сильно упрощает жизнь операторам связи и привлекает клиентов. Война форматов (так получилось с Betamax и VHS в мире видеозаписи), которая вначале воспринималась как вид конкуренции, оказалась неблагоприятной для бизнеса.

Было выдвинуто несколько технических предложений, впоследствии некоторые отсеялись и остались две основные технологии. Первая из них называется широкополосным CDMA (W-CDMA, Wideband CDMA) и была предложена фирмой Ericsson. Система использует расширение спектра с применением кода прямой последовательности, такой метод мы уже описывали ранее. Полоса пропускания

составляет 5 МГц и предназначена для межсетевого обмена с сетями стандарта GSM, хотя система не имеет обратной совместимости с GSM. Зато она обладает свойством, которое позволяет пользователю при выходе из соты W-CDMA и входе в ячейку GSM не прерывать звонок. Эта система была продвинута Европейским Союзом, который назвал ее **UMTS** (Universal Mobile Telecommunications System — универсальная система мобильной связи).

Вторым претендентом стала система **CDMA2000**, предложенная Qualcomm. В ней также используется принцип расширения спектра с применением кода прямой последовательности, да и вообще ее можно рассматривать как расширение IS-95 (между прочим, имеется обратная совместимость с этим стандартом). Полоса пропускания имеет ширину 5 МГц, однако CDMA2000 не предназначена для межсетевого взаимодействия с GSM, и передача соединения при переходе в ячейку GSM (или D-AMPS) не осуществляется. Среди других технических отличий от W-CDMA стоит отметить иную скорость следования элементарных посылок, иные кадровый интервал, используемый спектр и способ синхронизации.

Если бы инженеров из Ericsson и Qualcomm посадили за стол переговоров и поставили бы задачу выработать единую систему, они, наверное, справились бы с этим. В конце концов, базовый принцип обеих систем — это CDMA на канале с полосой 5 МГц. Вроде бы никто не собирается драться на дуэли из-за скорости элементарных посылок. Беда в том, что настоящей проблемой, как всегда, является отнюдь не инженерное решение, а политика. Европе требовалась система, умеющая работать с GSM; Соединенным Штатам нужна была система, совместимая с одной из уже существующих там систем (IS-95). Каждая сторона поддерживала свою компанию (Ericsson находится в Швеции, Qualcomm — в Калифорнии). В конце концов, обе компании оказались вовлечены во множественные тяжбы, связанные с патентами на технологию CDMA.

В марте 1999 года судебные разбирательства закончились тем, что Ericsson согласилась приобрести инфраструктуру Qualcomm. Компании также согласились на единый стандарт 3G, однако с множеством несовместимых функций, которые, впрочем, в большой степени связаны с документацией, а не с техническими различиями. Несмотря на все разногласия, в скором времени появятся службы и устройства 3G.

О системах 3G написано много, причем отзывы в основном восторженные. Большинство пишет о третьем поколении мобильной связи в том духе, что это самое большое достижение со времен изобретения хлеборезки. Вот библиографические ссылки: (Collins and Smith, 2001; De Vriendt и др., 2002; Harte и др., 2002; Lu, 2002; Sarikaya, 2000). Тем не менее, есть авторы, которые полагают, что отрасль мобильной телефонии идет в неверном направлении (Garber, 2002; Goodman, 2000).

Пока бояре борются в попытках прийти к соглашению по системам 3G, некоторые операторы связи уже делают первые робкие шаги в направлении 3G, предлагая, что называется, **2,5G**, хотя более точно было бы назвать это 2,1G. Одна такая система называется **EDGE** (Enhaced Data rates for GSM Evolution — повышенные скорости передачи для развития GSM) и представляет собой обычный GSM с увеличенным числом бит на бод. Проблема состоит в том, что чем больше би-

тов используется, тем больше вероятность ошибок. Поэтому в EDGE применяются девять различных схем модуляции и коррекции ошибок. Отличаются они друг от друга процентом пропускной способности, выделяемым на исправление ошибок, возникающих вследствие повышенной скорости.

Еще одной системой «второго с половиной поколения» является **GPRS** (General Packet Radio Service — общие услуги пакетной радиосвязи) — пакетная сеть на базе D-AMPS или GSM. Она позволяет обмениваться IP-пакетами по голосовым каналам сотовой связи. При работе GPRS некоторые временные интервалы на некоторых частотах резервируются под пакетный трафик. Число и расположение этих интервалов могут динамически изменяться базовой станцией в зависимости от соотношения голосового и информационного трафика в ячейке.

Доступные временные интервалы делятся на несколько логических каналов, используемых для разных целей. Базовая станция определяет, в каких интервалах располагаются эти каналы. Один канал предназначен для передачи пакетов с базовой на мобильную станцию, причем каждый пакет имеет поле индикации места своего назначения. Чтобы послать IP-пакет, мобильная станция запрашивает один или несколько временных интервалов, посылая на БС соответствующий запрос. Если запрос приходит неповрежденным, обратно отсылается информация о частоте и интервале, в котором можно передавать IP-пакет. Как только на базовую станцию прибывает пакет, она по обычному кабельному соединению пересылает его в Интернет. Поскольку система GPRS работает лишь как надстройка над существующей голосовой системой, ее можно рассматривать в лучшем случае как временную затычку, которая не понадобится, когда будет введена в строй 3G.

Даже несмотря на то, что 3G до сих пор не реализован в полном объеме, многие исследователи рассматривают его появление как уже свершившийся факт и поэтому не очень интересуются проблемами его изучения. Эти люди уже работают над созданием систем четвертого поколения (Berezdivin и др., 2002; Guo and Chaskar, 2002; Huang and Zhuang, 2002; Kellerer и др., 2002; Misra и др., 2002). 4G будет характеризоваться высокой пропускной способностью, повсеместной применимостью, полной интеграцией с кабельными сетями, особенно IP, адаптивным управлением ресурсами и частотным спектром, программным радио и высоким качеством обслуживания в области мультимедиа.

С другой стороны, повсеместно устанавливается такое большое количество точек доступа к беспроводным ЛВС стандарта 802.11, что многие рассматривают 3G не как свершившийся факт, а как мертворожденное поколение систем. По мнению многих, людям не составит труда оставаться на связи, просто перемещаясь от одной такой точки доступа к другой. Сказать, что данная отрасль находится в стадии бурных изменений — значит не сказать ничего. Посмотрим, что будет лет через пять. Скорее всего, изменится очень многое.

Кабельное телевидение

Мы уже изучили более или менее подробно стационарные и беспроводные телефонные системы. Они, безусловно, будут играть важную роль в сетевых технологиях будущего. Тем не менее, все популярнее становится альтернативная стацио-

нарная сетевая система, а именно кабельное телевидение. Многие уже получают доступ в Интернет и телефонные услуги по кабельным сетям, и их операторы стремятся расширить потребительский рынок. В следующих разделах мы будем обсуждать кабельное телевидение как сетевую структуру и как альтернативу телефонной системе, которую мы только что изучили. Дополнительную информацию по этой теме можно получить в изданиях (Laubach и др., 2001; Louis, 2002; Ovadia, 2002; Smith, 2002).

Абонентское телевидение

Кабельное телевидение впервые появилось в конце 1940-х годов и было способом улучшить прием сигнала в отдаленных поселках и горной местности. Система изначально состояла из большой антенны, расположенной на вершине холма и улавливающей телевизионный сигнал, усилителя, называющегося **распределительным устройством**, и коаксиального кабеля, по которому сигнал доставлялся непосредственно к абонентам, как показано на рис. 2.40.

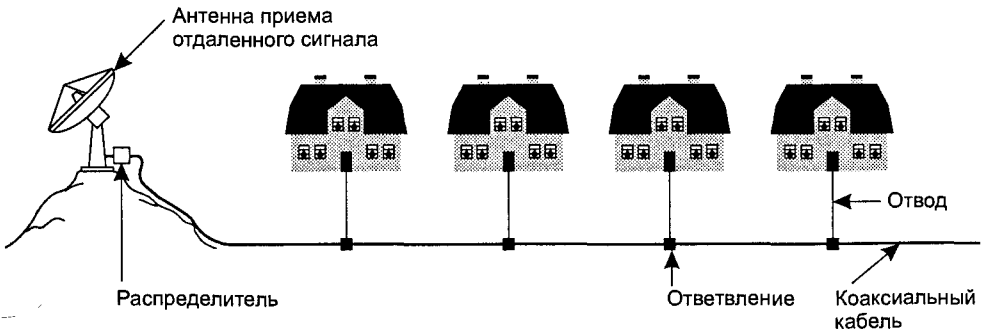


Рис. 2.40. Первая система кабельного телевидения

Вначале такая система называлась **абонентским телевидением**, или телевидением с коллективной антенной. Ее могло содержать даже какое-нибудь маленькое частное семейное предприятие. Любой предприниматель, немножко знакомый с электроникой, мог установить у себя в населенном пункте оборудование, и ему оставалось только найти клиентов, готовых оплачивать услуги. По мере роста числа абонентов необходимо было добавлять кабели и усилители. Передача была исключительно односторонней: от распределителя к пользователям. К 1970 году появились тысячи независимых систем.

В 1974 году корпорация Time основала новый канал под названием «Домашняя билетная касса», который представлял собой кабельное кино. Затем появились другие подобные тематические каналы: спортивный, кулинарный, новостной и т. д. Это привело к двум изменениям в данной отрасли. Во-первых, крупные корпорации стали скупать существующие кабельные системы и прокладывать свои кабели для привлечения новых клиентов. Во-вторых, со временем появилась необходимость в объединении систем, зачастую расположенных в различ-

ных городах, с целью основания новых кабельных каналов. Различные кабельные компании стали объединять свои сети, организуя единые региональные и национальные сети. Примерно то же самое происходило восемьюдесятью годами ранее с телефонными сетями. Изолированные друг от друга телефонные станции стали объединяться, что позволило организовывать междугородные звонки.

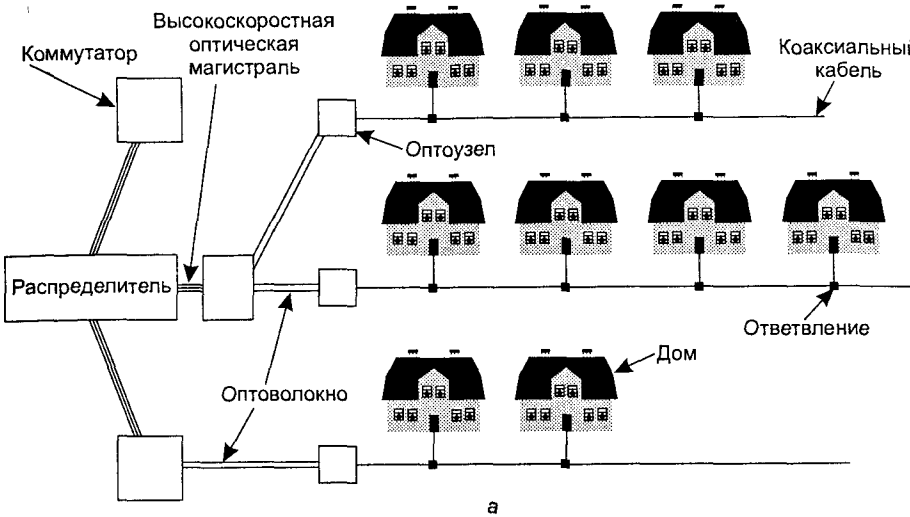
Кабельный Интернет

В течение долгих лет кабельная система расширялась, и обычные кабели между городами стали заменяться оптоволоконными с высокой пропускной способностью. Примерно то же самое стало происходить в телефонной сети. Система, использующая оптическое волокно на длинных магистралях и коаксиальный кабель для подвода сигнала к домам, получила название **HFC** (Hybrid Fiber Coax — комбинированная оптокоаксиальная кабельная система). Электрооптические преобразователи, реализующие интерфейс между оптической и электрической частями сети, называются **оптоузлами**. Поскольку пропускная способность оптических кабелей гораздо выше, чем коаксиальных, один оптоузел может обслуживать несколько низкоскоростных линий. Часть современной системы HFC показана на рис. 2.41, *а*.

В последнее время многие операторы кабельных сетей решили, что пора начать проникновение в бизнес предоставления доступа в Интернет. Некоторые, впрочем, захотели заняться также кабельной телефонией. Технические различия кабельного телевидения и телефонии определили инженерные задачи, которые предстояло решить. Прежде всего необходимо было заменить все односторонние усилители двухсторонними.

Между тем есть еще одно существенное различие между HFC (рис. 2.41, *а*) и телефонной системой (рис. 2.41, *б*), которое устранить гораздо сложнее. Кабель может быть один на несколько домов, а телефонный провод местной линии в каждую квартиру подводится свой. Когда речь идет о широковещательном телевидении, особой разницы нет. Все телепрограммы распространяются по кабелю, и не важно, 10 или 10 000 абонентов будут подключены к нему. Но когда один и тот же кабель используется для доступа в Интернет, то один клиент, скачивающий очень большой файл, потенциально может тем самым отнимать существенную часть пропускной способности у всех остальных. Чем больше пользователей, тем жестче конкуренция между ними в этом смысле. В телефонной системе такого нет: передача большого файла по каналу ADSL никак не влияет пропускную способность соседнего канала. С другой стороны, пропускная способность коаксиального кабеля много выше, чем витой пары.

Как же была решена эта проблема? Довольно просто: длинные кабели были разделены на короткие участки, напрямую подключаемые к оптоузлу. Доступная полоса пропускания на участке от распределителя до каждого оптоузла очень велика, и, поскольку в одном сегменте кабеля обычно не бывает большого числа абонентов, трафик вполне управляем. Обычный кабельный сегмент охватывает 500–2000 домов, однако все больше людей подключается к кабельному Интернету, поэтому иногда требуется более мелкое разбиение, что приводит к появлению дополнительных оптоузлов.



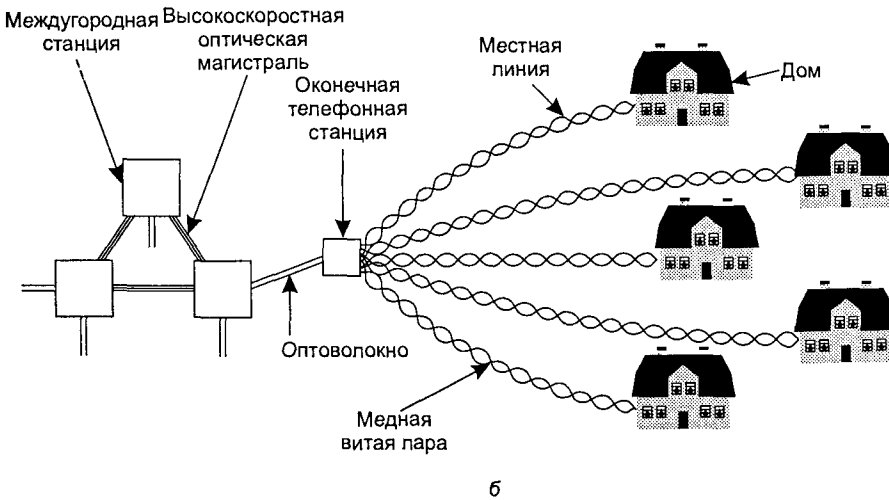


Рис. 2.41. Кабельное телевидение (а); стационарная телефонная система (б)

Распределение спектра

Если выкинуть все телевизионные каналы и использовать кабельную инфраструктуру исключительно для доступа в Интернет, это приведет к появлению большого числа недовольных пользователей, поэтому никто так не делает. Более того, в большинстве городов существуют определенные ограничения, не позволяющие так сделать, даже если какая-нибудь компания и захочет. Значит, нужно было найти какой-то способ совместного существования телевизионного сигнала и цифровых данных на одном кабеле.

Кабельное телевидение в Северной Америке традиционно занимает частоты с 54 до 550 МГц (за исключением диапазона с 88 до 108 МГц, отданного FM-радио). Ширина полосы каждого канала составляет 6 МГц, включая защитные полосы. В Европе нижний предел обычно ограничен 65 МГц, а каналы имеют ширину полосы 6–8 МГц, что позволяет увеличить разрешение, требуемое системам PAL и SECAM, однако это не очень принципиально. Нижняя часть спектра не используется. Современные кабели хорошо работают на частотах свыше 550 МГц, часто до 750 МГц и выше. Было принято решение выделить под исходящие каналы частоты 5–42 МГц (чуть выше в Европе), а высокие частоты использовать для входящих каналов. Распределение спектра в кабельных системах показано на рис. 2.42.



Рис. 2.42. Распределение частот в типичной системе кабельного телевидения, используемой для доступа в Интернет

Обратите внимание: поскольку телевизионный сигнал целиком идет только в одном направлении (входящем), можно использовать исходящие усилители, работающие только в диапазоне 5–42 МГц, а входящие — в диапазоне 54 МГц и выше, как показано на рисунке. Итак, входящий и исходящий спектры имеют сильный дисбаланс благодаря тому, что основная доля трафика приходится на телевидение и входящие интернет-каналы. И операторы кабельного телевидения, и компании, занимающиеся кабельным Интернетом, остались довольны таким распределением. Как мы уже говорили, телефонные компании часто предлагают асимметричный DSL-сервис, хотя у них нет особых технических оснований, чтобы так делать.

Длинные коаксиальные кабели не лучше местных телефонных линий, когда речь идет о передаче цифровых данных. Поэтому здесь также нужна аналоговая модуляция. Обычно применяется схема, при которой берутся каждые 6 или 8 МГц входящего канала и модулируются с помощью QAM-64 или (если кабель отменного качества) QAM-256. При канале шириной 6 МГц и методе QAM-64 мы получаем скорость около 36 Мбит/с. Если вычесть накладные расходы, чистая скорость передачи данных составит 27 Мбит/с. При использовании QAM-256 чистая скорость повышается до 39 Мбит/с. Европейские значения на треть выше.

Для исходящих потоков даже QAM-64 не очень подходит. Слишком много на соответствующих частотах помех от микроволновых устройств, CB-радиостанций и других источников. Исходя из этого применяют более медленную, но надежную схему — QPSK. Этот метод (см. рис. 2.21) использует всего лишь два бита на бод вместо 6 или 8, которые используются методами QAM на входящем потоке. Таким образом, асимметрия между пропускной способностью входящих и исходящих каналов даже выше, чем можно предположить исходя из рис. 2.42.

Кроме обновления усилителей, оператору нужно обновить и распределительное устройство на входе системы. Вместо латентного усилителя нужно поставить интеллектуальное цифровое вычислительное устройство с высокоскоростным оптоволоконным интерфейсом к провайдеру. Иногда обновляется даже имя этого устройства: вместо распределителя его называют **CMTS** (Cable Modem Termination System — оконечное устройство кабельного модема). Далее мы воздержимся от столь значительного обновления и будем по-прежнему называть распределитель распределителем.

Кабельные модемы

Для доступа в Интернет нужен кабельный модем — устройство, имеющее два интерфейса: один к компьютеру, второй — к кабельной сети. В первые годы существования кабельного Интернета у оператора связи были свои модемы, которые устанавливались у абонента специалистом службы технической поддержки. Однако затем стало понятно, что открытый стандарт может позволить создать рынок конкурентоспособных кабельных модемов, снизить цены на них и тем самым привлечь клиентов. Более того, возможность купить кабельный модем в обычном магазине и установить его самостоятельно (как пользователи всегда устанавливали телефонные модемы стандарта V.9x) позволит избежать ужасных расходов на оплату выезда специалиста.

В результате многие операторы кабельных сетей объединились с фирмой CableLabs с целью выработки стандарта на кабельные модемы и тестирования продукции на совместимость. Модемы появившегося стандарта **DOCSIS** (Data Over Cable Service Interface Specification — спецификация передачи данных по кабельному интерфейсу) сейчас только начинают заменять собственные модемы операторов. Европейская версия стандарта называется **EuroDOCSIS**. Однако не всем операторам нравится идея свободной продажи стандартных кабельных модемов — слишком уж хорошие деньги они получают за сдачу в аренду модемов своим захваченным в плен клиентам. Открытый стандарт, породивший десятки фирм — производителей кабельных модемов, продающих их в магазинах, ведет к концу подобной практики.

Интерфейс между модемом и компьютером довольно традиционен. Обычно это Ethernet со скоростью 10 Мбит/с (иногда USB). Кабельные модемы скоро будут напоминать обычные внутренние модемы и размерами, и способом установки.

Второй интерфейс более сложный. Немалая часть стандарта посвящена радиоинженерным решениям, но обсуждение этого вопроса выходит за рамки данной

книги. Единственное, что необходимо отметить, это то, что, как и ADSL-модемы, кабельные модемы находятся на постоянном подключении. Они устанавливают соединение сразу же после подачи питания и постоянно поддерживают его, поскольку операторы кабельных сетей не взимают плату за время на линии.

Чтобы лучше понять, как происходит работа кабельного модема, рассмотрим, что происходит при его включении. Модем начинает прослушивать входящий канал в поисках специального пакета, время от времени посылаемого распределителем. В нем сообщаются системные параметры для модемов, только что включившихся в работу. После обнаружения данного пакета новый модем объявляет о своем появлении по одному из исходящих каналов. Распределитель отвечает, присваивая модему входящий и исходящий каналы. Впрочем, исходное распределение каналов может быть динамически изменено распределителем, если он решит, что необходимо сбалансировать нагрузку.

Затем модем определяет, на каком расстоянии от распределителя он находится. Для этого посылается специальный пакет и высчитывается время, через которое приходит ответ. Этот процесс называется **измерением дальности**. Модему необходимо знать эти данные, чтобы настроить работу исходящих каналов и правильно синхронизироваться. Время работы делится на **мини-интервалы**. Каждый исходящий пакет должен умещаться в один или несколько соседних мини-интервалов. Распределитель анонсирует каждое начало цикла мини-интервалов, однако этот «стартовый выстрел» модемы слышат не одновременно, поскольку они находятся на разных расстояниях. Зная свое удаление от распределителя, модем может вычислить, когда на самом деле был послан принятый им сигнал начала мини-интервала. Длина мини-интервала зависит от сети. Обычно объем полезной информации в нем равен 8 байт.

Во время инициализации распределитель также присваивает модему мини-интервал для запроса пропускной способности исходящего канала. Как правило, одному и тому же мини-интервалу запроса соответствует несколько модемов, что приводит к конкуренции между ними. Когда компьютер хочет отослать пакет данных, он передает его модему, который запрашивает необходимое количество мини-интервалов для него. Если запрос принят, то распределитель посылает подтверждение по входящему каналу. В подтверждении модему сообщается, какие мини-интервалы зарезервированы для него. После этого пакет отправляется, начиная с первого «своего» мини-интервала. Используя специальное поле заголовка, можно сообщить о необходимости передать дополнительные пакеты.

Если же один и тот же мини-интервал хотят получить несколько станций одновременно, то никакого подтверждения не высылается, а эти станции могут повторить попытку только через случайный промежуток времени. Если при повторной попытке снова возникла коллизия, то случайный промежуток удваивается. (Для читателей, уже немного знакомых с сетевыми технологиями: это интервальный метод ALOHA с экспоненциальной двоичной отсрочкой передачи. Ethernet не может использоваться в качестве кабельного интерфейса, поскольку станции не могут прослушивать линию. Мы вернемся к этим вопросам в главе 4.)

Входящие каналы управляются не так, как исходящие. Во-первых, отправитель в этом случае только один — распределитель, поэтому не возникает никакой борьбы за линию и нет необходимости в мини-интервалах, которые, на самом деле, являются разновидностью статистического временного уплотнения. Во-вторых, трафик входящего канала обычно гораздо выше, чем исходящего, поэтому используются пакеты фиксированного размера — 204 байта. Часть пакета — код коррекции ошибок Рида—Соломона плюс еще некоторая служебная информация. Собственно данные занимают в пакете 184 байта. Эти числа были выбраны из соображений совместимости с цифровым телевидением, использующим MPEG-2, так что телевизионный и входящий информационный каналы имеют один и тот же формат. Логическая структура соединения показана на рис. 2.43.

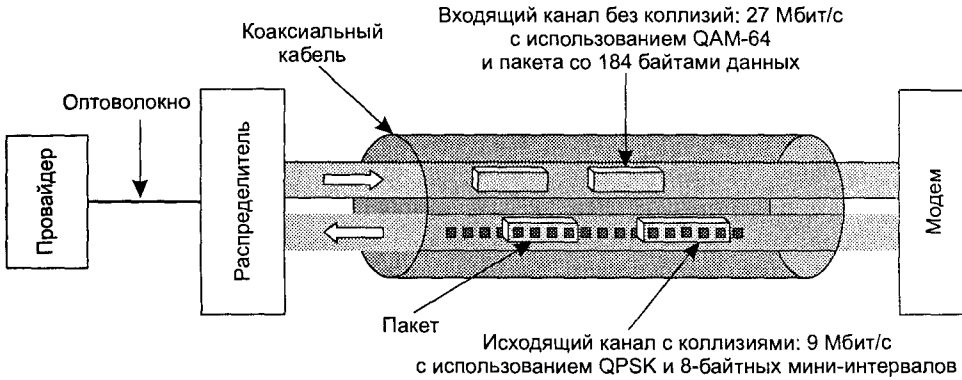


Рис. 2.43. Типичная схема входящего и исходящего каналов, принятая в США

Вернемся к инициализации модема. Когда он закончил измерение дальности и получил исходящий канал, входящий канал и мини-интервалы, он может начать передавать пакеты. Первый пакет, который он посылает, адресован провайдеру и содержит запрос на получение IP-адреса, который присваивается динамически с использованием протокола DHCP (мы изучим его в главе 5). Также у распределителя запрашивается точное время суток.

Следующий шаг связан с защитой данных. Поскольку кабель — это совместно используемый ресурс, каждый желающий может прочитывать трафик, проходящий мимо него. Чтобы предотвратить нежелательный доступ к информации соседа (буквально), все данные передаются в зашифрованной форме в обоих направлениях. Часть процедуры инициализации включает в себя обмен ключами шифра. На первый взгляд кажется невозможной задача незаметной передачи ключа при свете дня и огромном скоплении народа. На самом деле, задача вполне осуществима, но чтобы понять, как это делается, придется подождать до главы 8 (вкратце: используется алгоритм Дифи—Хеллмана).

Наконец, модему нужно идентифицировать себя по защищенному каналу. После этого инициализация считается завершенной. Пользователь может соединяться с провайдером и начинать работу.

Конечно, о кабельных модемах можно было бы говорить еще долго. Вот некоторые ссылки (Adams and Dulchinos, 2001; Donaldson and Jones, 2001; Dutta-Roy, 2001).

ADSL или кабель?

Что лучше, ADSL или кабельная сеть? С тем же успехом можно спорить о том, какая операционная система лучше. Или какой язык. Или какая религия. Ответ зависит от того, кого вы спрашиваете. Давайте сравним ADSL и кабельные сети по нескольким параметрам. И та, и другая система в качестве магистрального носителя использует оптическое волокно, однако на его концах используются разные типы носителей. В кабельных сетях это коаксиал, в ADSL — витая пара. Теоретически, пропускная способность коаксиального провода в сотни раз выше, чем у витой пары. Тем не менее, полная пропускная способность все равно недоступна пользователям кабельных систем, потому что большая часть полосы пропускания занята совершенно бесполезными вещами — телевизионными программами. На практике довольно трудно говорить о реальной эффективной емкости каналов. Провайдеры ADSL заявляют некоторую пропускную способность (например, 1 Мбит/с по входящему каналу, 256 Кбит/с по исходящему) и обычно достигают примерно 80 % от нее. Провайдеры кабельных сетей не делают никаких заявлений относительно скорости, поскольку все зависит от того, сколько пользователей находится в данном сегменте кабеля. Иногда скорость будет выше, чем в ADSL, иногда — ниже. Раздражает в данном случае, на самом деле, непредсказуемость. Если сейчас все «летает», это не означает, что через минуту будет так же, потому что не исключено, что именно сейчас самый большой свинтус в районе, занимающий своим трафиком всю пропускную способность сегмента, включает свой компьютер.

По мере привлечения в ADSL все большего числа пользователей качество обслуживания практически не снижается, поскольку каждый абонент имеет выделенное соединение. В кабельной системе каждый новый пользователь сегмента снижает качество обслуживания в целом. Единственный выход из данной ситуации — разбивать загруженные участки на более мелкие и подсоединять их напрямую к оптическому кабелю. Это стоит довольно дорого, поэтому операторы всячески стараются избежать таких ситуаций.

Мы, между прочим, уже изучили одну систему с совместно используемым каналом — мобильную телефонную систему. Там тоже имеются группы пользователей, находящихся в одной ячейке, каждый из которых занимает какую-то часть пропускной способности. Обычно существует жесткое разделение используемых ресурсов, для этого применяется частотное или временное уплотнение, потому что речевой трафик обычно довольно ровный. Однако применять жесткое разделение ресурсов при передаче данных оказывается крайне неэффективным, потому что зачастую канал вообще простаивает, тогда зарезервированные ресурсы тратятся просто так. Несмотря на все это, в этом смысле кабельная система гораздо ближе к мобильной телефонии, чем к стационарным системам.

Доступность — это параметр, по которому ADSL и кабельные сети отличаются друг от друга. У каждого есть телефон, но не каждый живет достаточно близко к оконечной ADSL-станции, чтобы установить соответствующую систему. С другой стороны, не у всех есть кабель в доме или в районе, но если уж он есть, то удаленность от оптоузла или распределительного устройства большой роли не играет. Стоит также отметить, что, поскольку кабельные системы начались с кабельного телевидения, корпоративных клиентов у них очень мало.

Будучи двухточечной системой, ADSL является более защищенной, чем кабельная сеть. Любой абонент последней может запросто считывать все пакеты, проходящие мимо него. По этой причине любой приличный оператор кабельной сети предлагает шифрование трафика обоих направлений. И все же, даже если пакет перехвачен в зашифрованном виде, это менее безопасно, чем полное отсутствие возможности перехвата.

Телефонная система, вообще говоря, надежнее кабеля. Например, существует система резервного питания, которая позволяет телефонной сети работать даже при временных отключениях электричества. Если же отключится питание какого-либо усилителя кабельной сети, все пользователи, находящиеся в его ведении, сразу потеряют соединение.

Наконец, существует большой выбор провайдеров ADSL. Иногда это даже форсируется специальными законодательными актами. Этого не скажешь про операторов кабельных сетей. Далеко не всегда есть какой-либо выбор.

Вывод такой: ADSL и кабельные сети имеют много общего и отличаются друг от друга не так уж сильно. Они предлагают сравнимое качество обслуживания и, по мере роста конкуренции между ними, по-видимому, будут предлагать сравнимые цены.

Резюме

Физический уровень является базовым для сетей любого типа. Природа носителей информации накладывает два фундаментальных ограничения на все каналы, и это определяет их допустимую пропускную способность. Первое ограничение носит имя Найквиста и имеет отношение к идеальным бесшумным каналам. Второе ограничение, ограничение Шеннона, говорит о каналах с шумом.

Среда передачи данных может быть управляемой или неуправляемой. Основными управляемыми носителями являются витая пара, коаксиальный кабель и оптоволоконный кабель. Среди неуправляемых носителей следует выделить радио, микроволны, инфракрасные и лазерные волны, распространяющиеся по воздуху. Быстро развивающейся областью является спутниковая связь, особенно системы LEO (низкоорбитальные спутники).

Ключевым элементом большинства глобальных сетей является телефонная система. Ее главные компоненты — это местные линии, магистрали и коммутаторы. Местные линии — это аналоговые каналы на основе витой пары, которым для передачи цифровых данных требуются модемы. ADSL имеет скорость до 50 Мбит/с, достигая ее путем разделения местной линии на множество виртуальных кана-

лов и отдельной модуляции каждого из них. Беспроводные локальные линии — это еще одно новое и интересное направление, особенно примечательна система LMDS.

Магистрали всегда являются цифровыми, в них применяются различные способы уплотнения, включая частотное (FDM), временное (TDM) и спектральное (WDM). Важны технологии как коммутации каналов, так и коммутации пакетов.

Для мобильных применений обычная стационарная телефонная сеть не подходит. Мобильные телефоны сейчас очень широко распространены в качестве средства передачи речи, однако вскоре не меньшее распространение эти устройства получат в качестве полноценного средства передачи данных. Первое поколение мобильных телефонных систем было аналоговым, доминировала система AMPS. Второе поколение было цифровым, и здесь главенствующие роли играли стандарты D-AMPS, GSM и CDMA. Третье поколение будет цифровым и будет базироваться на широкополосном CDMA.

Альтернативной сетевой системой является кабельное телевидение, которое сильно видоизменилось с тех времен, когда оно было телевизионной системой с общей антенной, и до сегодняшнего дня, когда это гибридная оптокоаксиальная сеть. В принципе, данная система обладает высокой пропускной способностью, но реальное качество обслуживания сильно зависит от числа и деятельности активных пользователей.

Вопросы

1. Сосчитайте коэффициенты Фурье для функции $f(t) = t$ $(0 \le t \le 1)$.
2. По бесшумному каналу с полосой пропускания 4 кГц каждую 1 мс передаются отсчеты сигнала. Какова будет максимальная скорость передачи данных?
3. Ширина телевизионных каналов составляет 6 МГц. Сколько бит в секунду может быть передано по такому каналу при использовании четырехуровневых цифровых сигналов? Предполагается, что шума в канале нет.
4. Какова максимально допустимая скорость передачи данных при передаче двоичного сигнала по каналу с полосой пропускания 3 кГц и отношением сигнал/шум 20 дБ?
5. Какое отношение сигнал/шум требуется для использования линии с полосой пропускания 50 кГц в качестве носителя T1?
6. В чем отличие пассивной звезды от активного повторителя в оптоволоконной сети?
7. Какова пропускная способность полосы спектра 0,1 мкм для длины волны 1 мкм?
8. Требуется переслать последовательность компьютерных экранных изображений по оптоволоконному кабелю. Размеры экрана 480×640 пикселов, каждый пиксел по 24 бита. Требуется передавать 60 экранов в секунду. Какая необхо-

дима для этого пропускная способность, а также какая часть спектра (в микрометрах) будет использована при условии, что передача осуществляется на длине волны 1,30 мкм?

9. Верна ли теорема Найквиста для оптоволоконного кабеля, или только для медного провода?
10. На рис. 2.5 самый левый диапазон длин волн уже, чем остальные. Почему?
11. Радиоантенны обычно лучше всего работают при размере антенны, равном длине волны радиосигнала. Диаметр антенны варьируется в пределах от 1 см до 5 м. Какому диапазону частот это соответствует?
12. Ослабление сигнала вследствие интерференции достигает максимального уровня, когда 2 луча прибывают со сдвигом фазы в 180°. Какова должна быть разница в пути прохождения сигнала, чтобы ослабление сигнала в микроволновой линии связи было максимальным (длина канала 50 км, частота 1 ГГц)?
13. Лазерный луч диаметром 1 мм нацелен на детектор диаметром 1 мм, установленный на крыше здания на расстоянии 100 м. На какой угол должен отклониться лазерный луч, чтобы он промахнулся мимо детектора?
14. 66 низкоорбитальных спутников проекта Iridium образуют шесть ожерелий вокруг Земли. Период их обращения составляет 90 минут. Каков средний интервал, необходимый наземному передатчику для осуществления передачи (hand off)?
15. Пусть имеется спутник, вращающийся на высоте геостационарных спутников, но имеющий отклонение орбитальной плоскости от экваториальной плоскости, равное углу ϕ. Кажется ли этот спутник неподвижным пользователю, находящемуся на поверхности Земли на ϕ-м градусе северной широты? Если нет, опишите движение спутника.
16. Сколько кодов оконечных телефонных станций существовало до 1984 года, когда доступ к каждой из станций осуществлялся через трехзначный код региона и первые три цифры местного номера? Коды регионов начинались с одной из цифр из диапазона 2–9, вторая цифра всегда была 1 или 0, а третья цифра могла быть любой. Первые две цифры местного номера были из диапазона 2–9; третья цифра могла быть любой.
17. Используя только информацию, данную в тексте, подсчитайте максимальное число телефонов, которое может обслуживать существующая в США телефонная система без изменения системы номеров и добавления оборудования. Может ли быть реально достигнуто это число? При подсчетах считать факс или модем эквивалентным телефону, предполагая, что на одной абонентской линии установлен только один телефон.
18. Простая телефонная система состоит из двух оконечных коммутаторов и одного междугородного коммутатора, с которым оконечные коммутаторы соединены дуплексным кабелем с полосой пропускания 1 МГц. За восьмичасовой рабочий день с одного телефона производится в среднем 4 звонка. Средняя продолжительность одного разговора составляет 6 минут. 10 % звонков явля-

ются междугородными (то есть проходят через междугородный коммутатор). Каково максимальное количество телефонов, которое может поддерживать оконечный коммутатор? (Предполагается 4 кГц на канал.)

19. У местной телефонной компании 10 млн абонентов. Все телефоны подключены к центральному коммутатору медными витыми парами. Средняя длина витых пар составляет 10 км. Сколько стоит медь местных телефонных линий? Предполагается, что провода круглые в сечении, диаметром 1 мм. Плотность меди равна 9,0 г/см^3, а цена меди — 3 доллара за килограмм.
20. Какой системой является нефтепровод — симплексной, полудуплексной, дуплексной или вообще не вписывается в эту классификацию?
21. Стоимость мощных микропроцессоров упала настолько, что теперь возможна их установка в каждый модем. Как это отразилось на обработке ошибок в телефонной линии?
22. Амплитудно-фазовая диаграмма модема того же типа, что изображен на рис. 2.17, состоит из точек с координатами: (1, 1), (1, −1), (−1, 1) и (−1, −1). Сколько бит в секунду сможет передавать такой модем на скорости 1200 бод?
23. Амплитудно-фазовая диаграмма модема того же типа, что изображена на рис. 2.21, состоит из точек с координатами: (0, 1) и (0, 2). Какого типа модуляция используется данным модемом: амплитудная или фазовая?
24. Амплитудно-фазовая диаграмма состоит из точек, расположенных на окружности с центром в начале координат. Какой тип модуляции применяется в данном случае?
25. Сколько частот использует полнодуплексный модем с модуляцией QAM-64?
26. Система ADSL, использующая DMT (цифровую мультиканальную тональную модуляцию), резервирует $^3/_4$ доступных каналов данных под входящее соединение. На каждом канале используется модуляция типа QAM-64. Какова емкость входящего соединения?
27. В примере с четырьмя секторами, изображенном на рис. 2.26, каждому сектору соответствует свой канал с пропускной способностью 36 Мбит/с. Теория очередей говорит о том, что если канал загружен на 50 %, то время формирования очереди будет равно времени передачи информации. Сколько времени потребуется при этих условиях на то, чтобы скачать веб-страничку размером 5 Кбайт? Сколько времени потребуется на скачивание страницы размером более 1 Мбайт по линии ADSL? С помощью модема на 56 Кбит/с?
28. Десять сигналов, каждому из которых требуется полоса 4000 Гц, мультиплексируются в один канал с использованием частотного уплотнения (FDM). Какова должна быть минимальная полоса уплотненного канала? Ширину защитных интервалов считать равной 400 Гц.
29. Почему период дискретизации кодово-импульсной модуляции был выбран равным 125 мкс?
30. Каков процент накладных расходов в носителе T1, то есть какой процент от пропускной способности 1,544 Мбит/с недоступен для конечного потребителя?

31. Сравните максимальную пропускную способность бесшумных каналов с полосой пропускания 4 кГц, использующих:
    1) аналоговое кодирование с двумя битами на отсчет;
    2) систему T1 с кодово-импульсной модуляцией.
32. При потере синхронизации система T1 пытается восстановить синхронизацию при помощи первого бита каждого кадра. Сколько кадров должно быть исследовано для восстановления синхронизации с вероятностью ошибки 0,001?
33. В чем отличие, если оно есть, между демодуляторной частью модема и кодирующей частью кодека? (Оба преобразуют аналоговый сигнал в цифровой.)
34. Сигнал передается в цифровом виде с периодом дискретизации 125 мкс по бесшумному каналу с полосой пропускания 4 кГц. Сколько бит в секунду в действительности передается каждым из следующих методов кодирования:
    1) стандарт CCITT 2,048 Мбит/с;
    2) дифференциальная импульсно-кодовая модуляция с 4-битовыми относительными значениями сигнала;
    3) дельта-модуляция?
35. Чистый синусоидальный сигнал амплитуды A кодируется при помощи дельта-модуляции с частотой x отсчетов в секунду. Выходное значение +1 соответствует изменению амплитуды на $+A/8$, выходное значение −1 соответствует $-A/8$. Какова максимальная частота сигнала, который может быть передан таким образом без накапливающейся ошибки?
36. В сети SONET точность системных часов составляет 10^{-9}. Сколько времени понадобится, чтобы дрейф часов составил 1 бит? Что следует из этих расчетов?
37. В табл. 2.4 скорость пользователя для канала OC-3 составляет 148,608 Мбит/с. Покажите, как это число может быть получено из параметров канала OC-3 системы SONET.
38. Для работы со скоростями передачи данных ниже STS-1 SONET имеет систему виртуального согласования (VT). Это часть полезной нагрузки, которая вставляется в кадр STS-1 и комбинируется с другими частями полезной нагрузки, заполняя весь кадр данных. Например, VT1.5 использует три колонки, VT2 использует 4 колонки, VT3 — 6 колонок, VT6 — 12 колонок кадра STS-1. Какие типы VT могут помочь согласовать скорость со следующими системами:
    1) служба DS-1 (1,544 Мбит/с);
    2) европейская служба CEPT-1 (2,048 Мбит/с);
    3) служба DS-2 (6,312 Мбит/с)?
39. В чем заключается основное отличие коммутации сообщений от коммутации пакетов?
40. Какова доступная для пользователя полоса пропускания в канале OC-12c?

41. Три сети с коммутацией пакетов состоят из n узлов каждая. Топология первой сети представляет собой звезду с центральным коммутатором, вторая является двунаправленным кольцом, а в третьей все узлы соединены друг с другом. Какими будут наименьшее, среднее и наибольшее расстояния между узлами каждой сети в прыжках?
42. Сравните задержку при передаче сообщения длиной в x бит по пути из k прыжков в сети с коммутацией каналов и в (слабо загруженной) сети с коммутацией пакетов. Время установки канала составляет s секунд, задержка распространения сигнала равна d секунд на прыжок, размер пакета равен p бит, а скорость передачи данных составляет b бит/с. При каких условиях сеть с коммутацией пакетов будет обладать меньшим временем задержки?
43. Предположим, что x бит данных пользователя передается по пути в k прыжков в сети с коммутацией пакетов в виде серии пакетов, каждый из которых состоит из p бит данных и h бит заголовка, причем $x \gg p + h$. Скорость передачи линий составляет b бит/с, а задержкой распространения сигнала можно пренебречь. Каким должно быть значение p, чтобы значение суммарной задержки было минимальным?
44. В обычной сотовой телефонной системе с шестигранными ячейками запрещено использовать одинаковые частотные диапазоны в соседних ячейках. Если доступно 840 частот, сколько из них может быть использовано в одной ячейке?
45. Реальная форма набора ячеек редко бывает такой правильной, как показано на рис. 2.35. Даже отдельные ячейки почти всегда имеют неправильную форму. Выскажите свои предположения относительно причин этого явления.
46. Сколько микроячеек системы PCS диаметром 100 м потребуется, чтобы покрыть ими Сан-Франциско (120 км^2)?
47. Когда пользователь сотовой телефонной системы пересекает границу между сотами, в некоторых случаях разговор прерывается несмотря на то, что все приемники и передатчики функционируют нормально. Почему?
48. D-AMPS обладает значительно более плохим качеством звука, чем GSM. Это связано с требованием обратной совместимости с AMPS (GSM не имеет подобных ограничений)? Если нет, то какова настоящая причина?
49. Подсчитайте максимальное число пользователей, которые одновременно могут работать в одной ячейке D-AMPS. Проделайте те же вычисления для GSM. Объясните разницу.
50. Пусть A, B и C одновременно передают нулевые биты, используя систему CDMA и элементарные последовательности, показанные на рис. 2.39, *б*. Как будет выглядеть результирующая элементарная последовательность?
51. При обсуждении ортогональности элементарных последовательностей CDMA утверждалось, что если $S \bullet T = 0$, то и $S \bullet \overline{T} = 0$. Докажите это.
52. Рассмотрим еще один подход к вопросу свойства ортогональности элементарных последовательностей CDMA. Каждый бит в паре последовательностей может совпадать или не совпадать с другим. Выразите свойство ортогональности в терминах совпадений и несовпадений парных битов.

53. Приемник CDMA получает элементарную последовательность: (−1 + 1 − 3 + 1 − 1 − 3 + 1 + 1). Предполагая, что исходные последовательности такие, как показано на рис. 2.39, *б*, какие станции посылали сигналы и какие именно?

54. Топология телефонной системы в части, включающей оконечный коммутатор, соединенный с телефонами абонентов, представляет собой звезду. Кабельное телевидение, напротив, состоит из единого длинного кабеля, объединяющего все дома в одной местности. Предположим, что в кабельном телевидении будущего вместо медного кабеля будет применяться оптоволоконный с пропускной способностью 10 Гбит/с. Сможет ли подобная линия воспроизвести работу телефонной линии и обеспечить каждому абоненту отдельную линию до оконечного коммутатора? Если да, то сколько телефонов может быть подключено к одному кабелю?

55. Система кабельного телевидения состоит из 100 коммерческих каналов, в которых программы время от времени прерываются рекламой. На что это больше похоже — на временное или частотное уплотнение?

56. Оператор кабельной сети предоставляет доступ в Интернет в районе, состоящем из 5000 домов. Компания использует коаксиальный кабель и распределяет спектр таким образом, что пропускная способность входящего потока для каждого кабеля составляет 100 Мбит/с. Чтобы привлечь клиентов, компания объявила, что каждому дому будет предоставлено 2 Мбит/с для входящего трафика в любое время. Опишите, что нужно компании, чтобы сдержать слово.

57. Используя распределение спектра, показанное на рис. 2.42, а также данную в тексте информацию, подсчитайте, сколько мегабит в секунду отводится в кабельной системе на входящий и исходящий каналы.

58. С какой скоростью пользователь кабельной сети может принимать данные, если все остальные пользователи пассивны?

59. Мультиплексирование потоков данных STS-1 играет важную роль в технологии SONET. Мультиплексор 3:1 уплотняет три входных потока STS-1 в один выходной поток STS-3. Уплотнение производится побайтно, то есть первые три выходных байта соответствуют первым байтам входных потоков 1, 2 и 3 соответственно. Следующие три байта — вторым байтам потоков 1, 2 и 3, и т. д. Напишите программу, симулирующую работу мультиплексора 3:1. В программе должно быть пять процессов. Главный создает четыре других процесса (для трех входных потоков и мультиплексора). Каждый процесс входного потока считывает в кадр STS-1 данные из файла в виде последовательности из 810 байт. Затем кадры побайтно отсылаются процессу мультиплексора. Мультиплексор принимает потоки и выводит результирующий кадр STS-3 (снова побайтно), записывая его на стандартное устройство вывода. Для взаимодействия между процессами используйте метод «труб».

лава 3
ровень передачи данных

Ключевые аспекты организации уровня передачи данных
Обнаружение и исправление ошибок
Элементарные протоколы передачи данных
Протоколы скользящего окна
Верификация протоколов
Примеры протоколов передачи данных
Резюме
Вопросы

этой главе мы рассмотрим принципы построения уровня 2 — уровня передачи анных (иногда его называют также канальным уровнем). Мы обсудим алгоритмы, беспечивающие надежную эффективную связь между двумя компьютерами. Мы удем рассматривать две машины, физически связанные каналом связи, действу-ющим подобно проводу (например, коаксиальным кабелем или телефонной ли-ией). Основное свойство канала, которое делает его подобным проводу, заклю-ается в том, что биты принимаются точно в том же порядке, в каком передаются.

На первый взгляд может показаться, что данная проблема настолько проста, то и изучать тут нечего, — машина *A* просто посылает биты в канал, а машина *B* их оттуда извлекает. К сожалению, в каналах связи иногда случаются ошибки при передаче данных. Кроме того, скорость передачи данных ограничена, а время распространения сигнала отлично от нуля. Все эти ограничения оказывают серь-зное влияние на эффективность передачи данных. Использующиеся для связи протоколы должны учитывать все эти факторы. Данным протоколам и посвяще-на эта глава.

После знакомства с ключевыми аспектами устройства уровня передачи дан-ых мы изучим его протоколы, рассмотрев природу ошибок, их причины, мето-

ды их обнаружения и исправления. Затем мы обсудим ряд протоколов, начиная с простых и далее рассматривая все более сложные протоколы. Каждый следующий протокол будет решать все более сложные проблемы уровня передачи данных. Наконец, мы рассмотрим вопросы моделирования и верификации протоколов и приведем несколько примеров протоколов передачи данных.

Ключевые аспекты организации уровня передачи данных

Уровень передачи данных должен выполнять ряд специфических функций. К ним относятся:

- обеспечение строго очерченного служебного интерфейса для сетевого уровня;
- обработка ошибок передачи данных;
- управление потоком данных, исключающее затопление медленных приемников быстрыми передатчиками.

Для этих целей канальный уровень берет пакеты, полученные с сетевого уровня, и вставляет их в специальные **кадры** для передачи. В каждом кадре содержится заголовок, поле данных и концевик. Структура кадра показана на рис. 3.1. Управление кадрами — это основа деятельности уровня передачи данных. В следующих разделах мы более подробно изучим обозначенные выше вопросы.

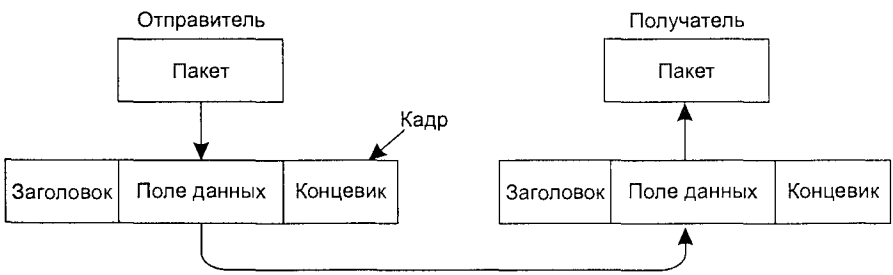


Рис. 3.1. Взаимодействие между пакетами и кадрами

Хотя эта глава и посвящена детальному рассмотрению уровня передачи данных и соответствующих протоколов, многие вопросы, обсуждаемые здесь, такие как контроль ошибок и контроль потока, относятся также к транспортным и другим протоколам. На самом деле, во многих сетях эти функции являются прерогативой верхних уровней и вообще не относятся к уровню передачи данных. С другой стороны, не так уж это важно, потому что основные принципы все равно остаются неизменными. Аргументом в пользу рассмотрения их именно в свете уровня передачи данных является то, что здесь они предстают в наиболее простой форме и их легко показать в деталях.

Глава 3
Уровень передачи данных

- Ключевые аспекты организации уровня передачи данных
- Обнаружение и исправление ошибок
- Элементарные протоколы передачи данных
- Протоколы скользящего окна
- Верификация протоколов
- Примеры протоколов передачи данных
- Резюме
- Вопросы

В этой главе мы рассмотрим принципы построения уровня 2 — уровня передачи данных (иногда его называют также канальным уровнем). Мы обсудим алгоритмы, обеспечивающие надежную эффективную связь между двумя компьютерами. Мы будем рассматривать две машины, физически связанные каналом связи, действующим подобно проводу (например, коаксиальным кабелем или телефонной линией). Основное свойство канала, которое делает его подобным проводу, заключается в том, что биты принимаются точно в том же порядке, в каком передаются.

На первый взгляд может показаться, что данная проблема настолько проста, что и изучать тут нечего, — машина *A* просто посылает биты в канал, а машина *B* их оттуда извлекает. К сожалению, в каналах связи иногда случаются ошибки при передаче данных. Кроме того, скорость передачи данных ограничена, а время распространения сигнала отлично от нуля. Все эти ограничения оказывают серьезное влияние на эффективность передачи данных. Использующиеся для связи протоколы должны учитывать все эти факторы. Данным протоколам и посвящена эта глава.

После знакомства с ключевыми аспектами устройства уровня передачи данных мы изучим его протоколы, рассмотрев природу ошибок, их причины, мето-

ды их обнаружения и исправления. Затем мы обсудим ряд протоколов, начиная с простых и далее рассматривая все более сложные протоколы. Каждый следующий протокол будет решать все более сложные проблемы уровня передачи данных. Наконец, мы рассмотрим вопросы моделирования и верификации протоколов и приведем несколько примеров протоколов передачи данных.

Ключевые аспекты организации уровня передачи данных

Уровень передачи данных должен выполнять ряд специфических функций. К ним относятся:

- обеспечение строго очерченного служебного интерфейса для сетевого уровня;
- обработка ошибок передачи данных;
- управление потоком данных, исключающее затопление медленных приемников быстрыми передатчиками.

Для этих целей канальный уровень берет пакеты, полученные с сетевого уровня, и вставляет их в специальные **кадры** для передачи. В каждом кадре содержится заголовок, поле данных и концевик. Структура кадра показана на рис. 3.1. Управление кадрами — это основа деятельности уровня передачи данных. В следующих разделах мы более подробно изучим обозначенные выше вопросы.

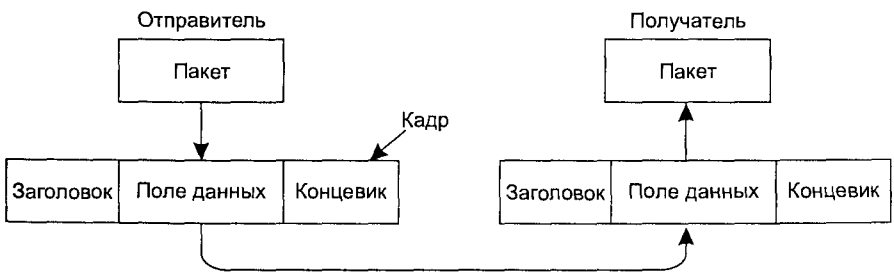


Рис. 3.1. Взаимодействие между пакетами и кадрами

Хотя эта глава и посвящена детальному рассмотрению уровня передачи данных и соответствующих протоколов, многие вопросы, обсуждаемые здесь, такие как контроль ошибок и контроль потока, относятся также к транспортным и другим протоколам. На самом деле, во многих сетях эти функции являются прерогативой верхних уровней и вообще не относятся к уровню передачи данных. С другой стороны, не так уж это важно, потому что основные принципы все равно остаются неизменными. Аргументом в пользу рассмотрения их именно в свете уровня передачи данных является то, что здесь они предстают в наиболее простой форме и их легко показать в деталях.

Сервисы, предоставляемые сетевому уровню

Задача уровня передачи данных заключается в предоставлении сервисов сетевому уровню. Основным сервисом является передача данных от сетевого уровня передающей машины сетевому уровню принимающей машины. На передающей машине работает некая сущность, или процесс, который передает биты с сетевого уровня на уровень передачи данных для передачи их по назначению. Работа уровня передачи данных заключается в передаче этих битов на принимающую машину так, чтобы они могли быть переданы сетевому уровню принимающей машины, как показано на рис. 3.2, *а*. В действительности данные передаются по пути, показанному на рис. 3.2, *б*, однако проще представлять себе два уровня передачи данных, связывающихся друг с другом при помощи протокола передачи данных. По этой причине на протяжении этой главы будет использоваться модель, изображенная на рис. 3.2, *а*.

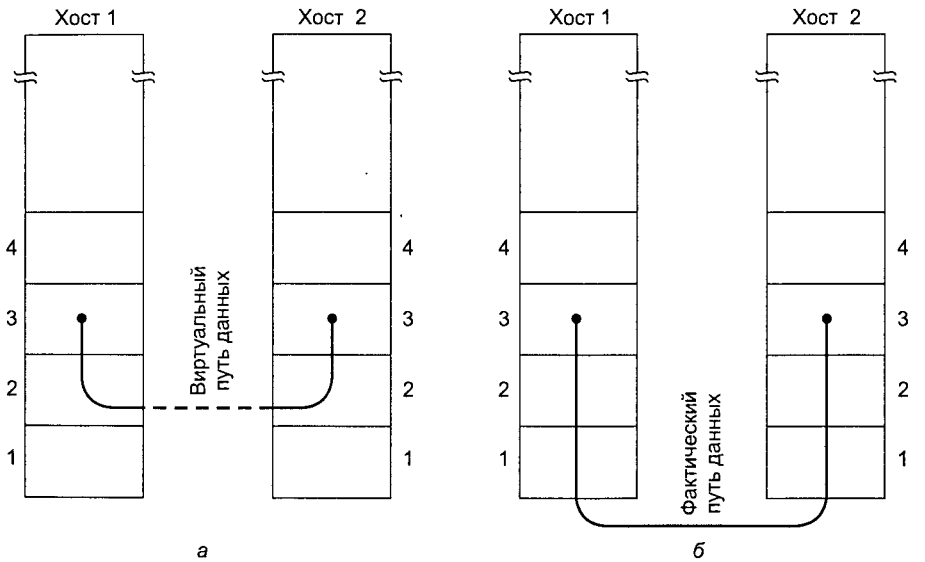


Рис. 3.2. Виртуальное соединение (*а*); реальное соединение (*б*)

Уровень передачи данных может предоставлять различные сервисы. Их набор может быть разным в разных системах. Обычно возможны следующие варианты.

1. Сервис без подтверждений, без установки соединения.
2. Сервис с подтверждениями, без установки соединения.
3. Сервис с подтверждениями, ориентированная на соединение.

Рассмотрим эти варианты по очереди.

Сервис без подтверждений и без установки соединения заключается в том, что передающая машина посылает независимые кадры принимающей машине,

а принимающая машина не посылает подтверждений о приеме кадров. Никакие соединения заранее не устанавливаются и не разрываются после передачи кадров. Если какой-либо кадр теряется из-за шума в линии, то на уровне передачи данных не предпринимается никаких попыток восстановить его. Данный класс сервисов приемлем при очень низком уровне ошибок. В этом случае вопросы, связанные с восстановлением потерянных при передаче данных, могут быть оставлены верхним уровням. Он также применяется в линиях связи реального времени, таких как передача речи, в которых лучше получить искаженные данные, чем получить их с большой задержкой. Сервис без подтверждений и без установки соединения используется в уровне передачи данных в большинстве локальных сетей.

Следующим шагом в сторону повышения надежности является сервис с подтверждениями, без установки соединения. При его использовании соединение также не устанавливается, но получение каждого кадра подтверждается. Таким образом, отправитель знает, дошел ли кадр до пункта назначения в целости. Если в течение установленного интервала времени подтверждения не поступает, кадр посылается снова. Такая служба полезна в случае использования каналов с большой вероятностью ошибок, например, в беспроводных системах.

Вероятно, следует отметить, что предоставление подтверждений является скорее оптимизацией, чем требованием. Сетевой уровень всегда может послать пакет и ожидать подтверждения его доставки. Если за установленный период времени подтверждение не будет получено отправителем, сообщение может быть выслано еще раз. Проблема при использовании данной стратегии заключается в том, что кадры обычно имеют жесткое ограничение максимальной длины, связанное с аппаратными требованиями. Пакеты сетевого уровня таких ограничений не имеют. Таким образом, если среднее сообщение разбивается на 10 кадров и 20 % из них теряется по дороге, то передача сообщения таким методом может занять очень много времени. Если подтверждать получение отдельных кадров и в случае ошибки посылать их повторно, передача всего сообщения займет гораздо меньше времени. В таких надежных каналах, как, например, оптоволоконный кабель, накладные расходы на подтверждения на уровне передачи данных только снизят пропускную способность канала, однако для беспроводной связи такие расходы окупятся и уменьшат время передачи длинных сообщений.

Наиболее сложным сервисом, который может предоставлять уровень передачи данных, является ориентированная на соединение служба с подтверждениями. При использовании данного метода источник и приемник, прежде чем передать друг другу данные, устанавливают соединение. Каждый посылаемый кадр нумеруется, а канальный уровень гарантирует, что каждый посланный кадр действительно принят на другой стороне канала связи. Кроме того, гарантируется, что каждый кадр был принят всего один раз и что все кадры были получены в правильном порядке. В службе без установления соединения, напротив, возможно, что при потере подтверждения один и тот же кадр будет послан несколько раз и, следовательно, несколько раз получен. Ориентированный на соединение сервис предоставляет процессам сетевого уровня эквивалент надежного потока битов.

При использовании ориентированного на соединение сервиса передача данных состоит из трех различных фаз. В первой фазе устанавливается соединение, при этом обе стороны инициализируют переменные и счетчики, необходимые для слежения за тем, какие кадры уже приняты, а какие — еще нет. Во второй фазе передаются кадры данных. Наконец, в третьей фазе соединение разрывается и при этом освобождаются все переменные, буферы и прочие ресурсы, использовавшиеся во время соединения.

Рассмотрим типичный пример: глобальная сеть, состоящая из маршрутизаторов, соединенных от узла к узлу выделенными телефонными линиями. Когда кадр прибывает на маршрутизатор, аппаратура проверяет его на наличие ошибок (с помощью метода, который мы изучим чуть позднее) и передает кадр программному обеспечению уровня передачи данных (которое может быть внедрено в микросхему сетевой карты). Программа уровня передачи данных проверяет, тот ли это кадр, который ожидался, и если да, то передает пакет, хранящийся в поле полезной нагрузки кадра, программе маршрутизации. Программа маршрутизации выбирает нужную выходящую линию и передает пакет обратно программе уровня передачи данных, который передает его дальше по сети. Прохождение сообщения через два маршрутизатора показано на рис. 3.3.

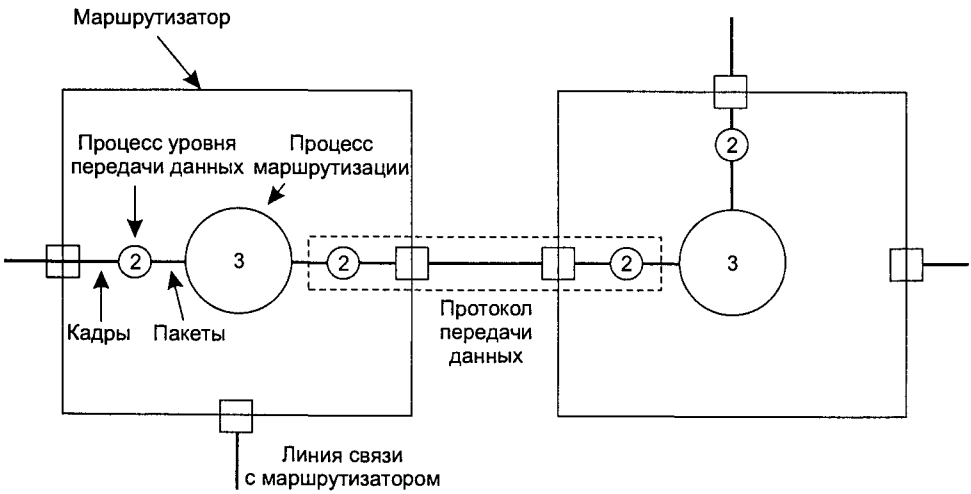


Рис. 3.3. Протокол передачи данных

Программы маршрутизации часто требуют правильного выполнения работы, то есть им нужно надежное соединение с упорядоченными пакетами на всех линиях, соединяющих маршрутизаторы. Такие программы обычно не любят, если приходится слишком часто беспокоиться о потерянных пакетах. Сделать ненадежные линии надежными или хотя бы довольно хорошими — задача уровня передачи данных, показанного на рисунке пунктирным прямоугольником. Заметим, что хотя на рисунке показаны несколько копий программы уровня передачи данных, на самом деле все линии связи обслуживаются одной копией программы с различными таблицами и структурами данных для каждой линии.

Формирование кадра

Для предоставления сервиса сетевому уровню уровень передачи данных должен использовать сервисы, предоставляемые ему физическим уровнем. Физический уровень принимает необработанный поток битов и пытается передать его по назначению. Этот поток не застрахован от ошибок. Количество принятых бит может быть меньше, равно или больше числа переданных бит; кроме того, значения принятых битов могут отличаться от значений переданных. Уровень передачи данных должен обнаружить ошибки и, если нужно, исправить их.

Обычно уровень передачи данных разбивает поток битов на отдельные кадры и считает для каждого кадра контрольную сумму. (Алгоритмы подсчета контрольных сумм будут обсуждаться далее в этой главе.) Когда кадр прибывает в пункт назначения, его контрольная сумма подсчитывается снова. Если она отличается от содержащейся в кадре, то уровень передачи данных понимает, что при передаче кадра произошла ошибка, и принимает меры (например, игнорирует испорченный кадр и посылает передающей машине сообщение об ошибке).

Разбиение потока битов на отдельные кадры представляет собой более сложную задачу, чем это может показаться на первый взгляд. Один из способов разбиения на кадры заключается во вставке временных интервалов между кадрами, подобно тому, как вставляются пробелы между словами в тексте. Однако сети редко предоставляют гарантии сохранения временных параметров при передаче данных, поэтому возможно, что эти интервалы при передаче исчезнут или, наоборот, будут добавлены новые интервалы.

Поскольку для отметки начала и конца кадра полагаться на временные параметры слишком рискованно, были разработаны другие методы. В данном разделе мы рассмотрим четыре метода маркировки границ кадров.

1. Подсчет количества символов.
2. Использование сигнальных байтов с символьным заполнением.
3. Использование стартовых и стоповых битов с битовым заполнением.
4. Использование запрещенных сигналов физического уровня.

Первый метод формирования кадров использует поле в заголовке для указания количества символов в кадре. Когда уровень передачи данных на принимающем компьютере видит это поле, он узнает, сколько символов последует, и таким образом определяет, где находится конец кадра. Этот прием проиллюстрирован на рис. 3.4, *а* для четырех кадров размером 5, 5, 8 и 8 символов соответственно.

Недостаток такой системы в том, что при передаче может быть искажен сам счетчик. Например, если размер второго кадра из числа 5 станет из-за ошибки в канале числом 7, как показано на рис. 3.4, *б*, то принимающая машина потеряет синхронизацию и не сможет обнаружить начало следующего кадра. Даже если контрольная сумма не совпадет (скорее всего) и принимающий компьютер поймет, что кадр принят неверно, то он все равно не сможет определить, где начало следующего кадра. Запрашивать повторную передачу кадра также бесполезно, поскольку принимающий компьютер не знает, сколько символов нужно пропустить до начала повторной передачи. По этой причине метод подсчета символов теперь практически не применяется.



Рис. 3.4. Поток символов: без ошибок (*а*); с одной ошибкой (*б*)

Второй метод формирования кадров решает проблему восстановления синхронизации после сбоя при помощи маркировки начала и конца каждого кадра специальными байтами. В прошлом стартовые и стоповые байты отличались друг от друга, но в последнее время большинство протоколов перешло на использование в обоих случаях одного и того же байта, называемого **флаговым**. Это показано на рис. 3.5, *а* как FLAG. Таким образом, если приемник теряет синхронизацию, ему необходимо просто найти флаговый байт, с помощью которого он распознает конец текущего кадра. Два соседних флаговых байта говорят о том, что кончился один кадр и начался другой.

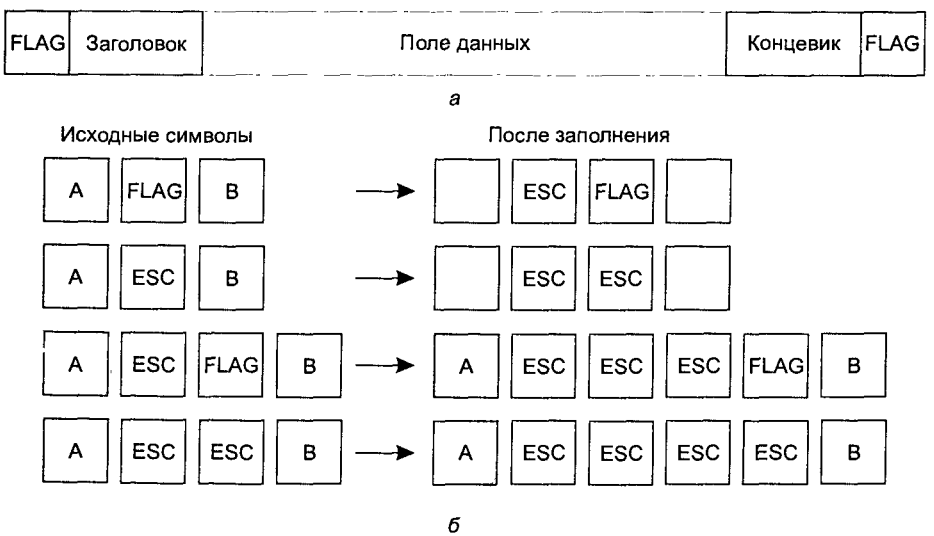


Рис. 3.5. Кадр, ограниченный флаговыми байтами (а); четыре примера байтовых последовательностей до и после символьного заполнения (*б*)

Этот метод иногда приводит к серьезным проблемам при передачи бинарных данных, таких как объектные коды программ или числа с плавающей запятой. В передаваемых данных запросто может встретиться последовательность, используемая в качестве флагового байта. Возникновение такой ситуации, скорее всего, собьет синхронизацию. Одним из способов решения проблемы является добавление специального escape-символа (знака переключения кода, ESC) непосредственно перед случайно совпавшим с флаговым байтом внутри кадра. Уровень передачи данных получателя вначале убирает эти escape-символы, затем передает кадр на сетевой уровень. Такой метод называется символьным заполнением. Таким образом, настоящий флаг можно отличить от «подложного» по наличию или отсутствию перед ним ESC.

Следующий логичный вопрос: а что, если и символ ESC случайно окажется среди прочих данных? Решение такое же: вставить перед этим фиктивным escape-символом настоящий. Тогда любой одиночный ESC будет частью escape-последовательности, а двойной будет указывать на то, что служебный байт случайно оказался в потоке данных. Некоторые примеры показаны на рис. 3.6, *б*. В любом случае, байтовая последовательности после ее очищения от вставных символов в точности совпадает с исходной.

Схема символьного заполнения, показанная на рис. 3.5, — это немного упрощенная модель протокола PPP, с помощью которого большинство домашних компьютеров соединяется с интернет-провайдером. Мы изучим PPP в этой главе.

Главный недостаток этого метода заключается в том, что он тесно связан с 8-битными символами. Между тем не во всех кодировках один символ соответствует 8 битам. Например, UNICODE использует 16-битное кодирование. По мере развития сетевых технологий недостатки использования длины символьного кода в механизме формирования кадров становились все очевиднее. Поэтому потребовалось создание новой техники, допускающей использование символов произвольного размера.

Новый метод позволяет использовать кадры и наборы символов, состоящие из любого количества битов. Вот как это реализуется. Каждый кадр начинается и завершается специальной последовательностью битов, 01111110 (на самом деле это все тот же флаговый байт). Если в битовом потоке передаваемых данных встретится пять идущих подряд единиц, уровень передачи данных автоматически вставит в выходной поток нулевой бит. **Битовое заполнение** аналогично символьному, при котором в кадр перед случайно встретившимся среди данных флагом вставляется escape-символ.

Когда принимающая сторона встречает пять единиц подряд, за которыми следует ноль, она автоматически удаляет этот ноль. Битовое заполнение, как и символьное, является абсолютно прозрачным для сетевого уровня обоих компьютеров. Если флаговая последовательность битов (01111110) встречается в данных пользователя, она передается в виде 011111010, но в памяти принимающего компьютера сохраняется опять в исходном виде: 01111110. На рис. 3.6 приведен пример битового заполнения.

Благодаря битовому заполнению границы между двумя кадрами могут быть безошибочно распознаны с помощью флаговой последовательности. Таким обра-

зом, если приемная сторона потеряет границы кадров, ей нужно всего лишь отыскать в полученном потоке битов флаговый байт, поскольку он встречается только на границах кадров и никогда — в данных пользователя.

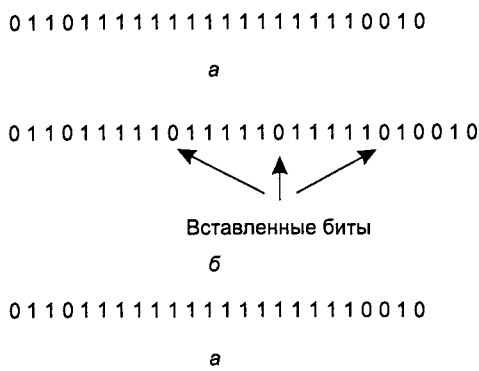


Рис. 3.6. Битовое заполнение: исходные данные (*а*); данные на линии (*б*); данные, сохраненные в памяти после удаления вставных битов (*а*)

Наконец, последний метод формирования кадров приемлем только в сетях, в которых физический носитель обладает некоторой избыточностью. Например, некоторые локальные сети кодируют один бит данных двумя физическими битами. Обычно бит 1 кодируется парой высокого и низкого уровней сигналов (отрицательный перепад), а бит 0 — наоборот, парой низкого и высокого уровней (положительный перепад). В такой схеме каждый передаваемый бит данных содержит в середине переход, благодаря чему упрощается распознавание границ битов. Комбинации уровней сигналов (низкий—низкий и высокий—высокий) не используются для передачи данных, но используются в качестве ограничителей кадров в некоторых протоколах.

В заключение разговора о кадрировании заметим, что многие протоколы передачи данных для повышения надежности применяют комбинацию счетчика символов с другим методом формирования кадра. Когда прибывает кадр, для обнаружения его конца используется счетчик. Кадр воспринимается как правильный только в том случае, если соответствующий разделитель присутствует в нужной позиции и совпадает контрольная сумма. В противном случае сканируется входной поток, в котором ищется следующий разделитель.

Обработка ошибок

Решив проблему маркировки начала и конца кадра, мы сталкиваемся с новой проблемой: как гарантировать доставку сетевому уровню принимающей машины всех кадров и при этом расположить их в правильном порядке. Предположим, что отправитель просто посылает кадры, не заботясь о том, дошли ли они до адресата. Этого было бы достаточно для сервиса без подтверждений и без установления соединения, но не для ориентированного на соединение сервиса с подтверждениями.

Обычно для гарантирования надежной доставки поставщику посылается информация о том, что происходит на другом конце линии. Протокол требует от получателя посылать обратно специальные управляющие кадры, содержащие позитивные или негативные сообщения о полученных кадрах. Получив позитивное сообщение, отправитель узнает, что посланный им кадр успешно получен на том конце линии. Негативное сообщение, напротив, означает, что с кадром что-то случилось и его нужно передать снова.

Кроме того, посланный кадр может из-за неисправности оборудования или какой-нибудь помехи (например, шума) пропасть полностью. В этом случае принимающая сторона его просто не получит и, соответственно, никак не прореагирует, а отправитель может вечно ожидать положительного или отрицательного ответа и так ничего и не получить.

Чтобы избежать зависаний сети в случае полной потери кадров, используются таймеры уровня передачи данных. После посылки кадра включается таймер и отсчитывает интервал времени, достаточный для получения принимающим компьютером этого кадра, его обработки и посылки обратно подтверждения. В нормальной ситуации кадр правильно принимается, а подтверждение посылается назад и вручается отправителю, прежде чем истечет установленный интервал времени, и только после этого таймер отключается.

Однако если либо кадр либо подтверждение теряется по пути, установленный интервал времени истечет, и отправитель получит сообщение о возможной проблеме. Самым простым решением для отправителя будет послать кадр еще раз. Однако при этом возникает опасность получения одного и того же кадра несколько раз уровнем передачи данных принимающего компьютера и повторной передачи его сетевому уровню. Чтобы этого не случилось, необходимо последовательно пронумеровать отсылаемые кадры, так чтобы получатель мог отличить повторно переданные кадры от оригиналов.

Вопрос управления таймерами и порядковыми номерами, гарантирующими, что каждый кадр доставлен сетевому уровню принимающего компьютера ровно один раз, не больше и не меньше, является очень важной частью задачи, решаемой уровнем передачи данных. Далее в этой главе мы подробно изучим методы управления на серии постепенно усложняющихся примеров.

Управление потоком

Еще один важный аспект разработки уровня передачи данных (а также более высоких уровней) связан с вопросом о том, что делать с отправителем, который постоянно желает передавать кадры быстрее, чем получатель способен их получать. Такая ситуация может возникнуть, если у передающей стороны оказывается более мощный (или менее загруженный) компьютер, чем у принимающей. Отправитель продолжает посылать кадры на высокой скорости до тех пор, пока получатель не окажется полностью ими завален. Даже при идеально работающей линии связи в определенный момент получатель просто не сможет продолжать обработку все прибывающих кадров и начнет их терять. Очевидно, что для предотвращения подобной ситуации следует что-то предпринять.

В настоящее время применяются два подхода. При первом, называющемся **управлением потоком с обратной связью**, получатель отсылает отправителю информацию, разрешающую последнему продолжить передачу или, по крайней мере, сообщающую о том, как идут дела у получателя. При втором подходе, **управлении потоком с ограничением**, в протокол встраивается механизм, ограничивающий скорость, с которой передатчики могут передавать данные. Обратная связь с получателем отсутствует. В этой главе мы рассмотрим только подход с обратной связью, поскольку второй подход никогда не применяется на уровне передачи данных (впрочем, мы вернемся к нему в главе 5).

Известны различные схемы контроля потока с обратной связью, но большинство из них используют один и тот же принцип. Протокол содержит четко определенные правила, определяющие, когда отправитель может посылать следующий кадр. Эти правила часто запрещают пересылку кадра до тех пор, пока получатель не даст разрешения, либо явно, либо неявно. Например, при установке соединения получатель может сказать: «Вы можете послать мне сейчас n кадров, но не посылайте следующие кадры, пока я не попрошу вас продолжать». В данной главе мы рассмотрим различные механизмы, основанные на этом принципе.

Обнаружение и исправление ошибок

Как было показано в главе 2, телефонная система состоит из трех частей — коммутаторов, межкоммутаторных магистралей и местных телефонных линий. Первые две части являются сегодня почти полностью цифровыми. Местные телефонные линии до сих пор представляют собой аналоговые медные витые пары и останутся такими еще в течение нескольких лет, поскольку их замена стоит очень дорого. И хотя в цифровой части телефонной системы ошибки случаются редко, в местных телефонных линиях вероятность ошибки очень велика. Кроме того, в последнее время все большее распространение получает беспроводная связь, в которой уровень ошибок на несколько порядков выше, чем в соединяющих телефонные станции магистралях. Отсюда следует вывод: ошибки при передаче данных останутся важным фактором еще на долгие годы. Сейчас мы приступаем к изучению методов их обнаружения и устранения.

Вследствие особенностей физических процессов, порождающих их, ошибки в некоторых типах носителей (например, радио) чаще бывают не единичными, а групповыми. В этом есть как положительные, так и отрицательные стороны. Положительные связаны с тем, компьютеры всегда посылают данные битовыми блоками. Представьте себе блок размером в 1000 бит при вероятности ошибки 0,001 на бит. Если бы ошибки были независимыми, то очень большой процент блоков содержал бы ошибки. Однако если ошибки приходят пакетами, искажая по 100 бит подряд, то из 100 блоков будут испорчены в среднем только один или два. Неудобством групп ошибок является то, что их значительно труднее исправить, чем изолированные ошибки.

Корректирующее кодирование

Разработчики сетей создали две основные стратегии для борьбы с ошибками. Каждый метод основывается на добавлении к передаваемым данным некоторой избыточной информации. В одном случае этой информации должно быть достаточно чтобы выявить, какие данные должны были прийти. В другом случае избыточной информации должно быть достаточно только для того, чтобы получатель понял что произошла ошибка (без указания ее типа) и запросил повторную передачу Первая стратегия использует коды, называющиеся корректирующими, или кодами с исправлением ошибок. Вторая — **код с обнаружением ошибок**. Использование кода с обнаружением ошибок часто называют **прямым исправлением ошибок**.

Каждая стратегия занимает свою, так сказать, экологическую нишу. В высоконадежных каналах, таких как оптоволокно, дешевле использовать код с обнаружением ошибок и просто заново передавать случайные поврежденные блоки. Однако, скажем, беспроводные соединения, в которых может возникать множество ошибок, чаще используют коды с избыточностью, достаточной для того, чтобы приемник мог определить, какие данные должны были прийти. Это надежнее, чем полагаться на повторную передачу, которая тоже, возможно, не сможет пройти без ошибок.

Чтобы понять, как могут обнаруживаться и исправляться ошибки, необходимо рассмотреть подробнее, что же представляет собой ошибка. Обычно кадр состоит из m битов данных (то есть информационных битов) и r избыточных или контрольных битов. Пусть полная длина кадра равна n (то есть $n = m + r$). Набор из n бит, содержащий информационные и контрольные биты, часто называют n-битовым **кодовым словом** или кодовой комбинацией.

Если рассмотреть два кодовых слова, например 10001001 и 10110001, можно определить число различающихся в них соответствующих разрядов. В данном примере различаются 3 бита. Для нахождения этого числа нужно сложить два кодовых слова по модулю 2 (операция «исключающее или») и сосчитать количество единиц в результате, например:

$$\begin{array}{r} 10001001 \\ 10110001 \\ \hline 00111000 \end{array}$$

Количество битов, которыми различаются два кодовых слова, называется **кодовым расстоянием**, или расстоянием между кодовыми комбинациями в смысле Хэмминга (Hamming, 1950). Смысл этого числа состоит в том, что если два кодовых слова находятся на кодовом расстоянии d, то для преобразования одного кодового слова в другое понадобится d ошибок в одиночных битах.

В большинстве приложений передачи данных все 2^m возможных сообщений являются допустимыми, однако благодаря использованию контрольных битов не все 2^n возможных кодовых слов используются. Зная алгоритм формирования контрольных разрядов, можно построить полный список всех допустимых кодовых слов и в этом списке найти такую пару кодовых слов, кодовое расстояние

между которыми будет минимальным. Это расстояние называется **минимальным кодовым расстоянием** кода, или расстоянием всего кода в смысле Хэмминга.

Способности кода по обнаружению и исправлению ошибок зависят от его минимального кодового расстояния. Для обнаружения d ошибок в одном кодовом слове необходим код с минимальным кодовым расстоянием, равным $d + 1$, поскольку d однобитовых ошибок не смогут изменить одну допустимую комбинацию так, чтобы получилась другая допустимая комбинация. Когда приемник встречает запрещенную кодовую комбинацию, он понимает, что при передаче произошла ошибка. Аналогично, для исправления d ошибок в одном кодовом слове требуется код с минимальным кодовым расстоянием, равным $2d + 1$, так как в данном случае даже при d однобитовых ошибках результат окажется ближе к исходному кодовому слову, чем к любому другому, и, следовательно, его можно будет однозначно восстановить.

В качестве простейшего примера кода с обнаружением ошибок рассмотрим код, в котором к данным добавляется один **бит четности**. Бит четности выбирается таким образом, чтобы количество единиц во всем кодовом слове было четным (или нечетным). Например, при посылке числа 10110101 с добавлением бита четности в конце оно становится равным 101101011, тогда как 10110001 преобразуется в 101100010. Код с единственным битом четности имеет кодовое расстояние, равное 2, так как любая однократная ошибка в любом разряде образует кодовое слово с неверной четностью. Такой код может использоваться для обнаружения однократных ошибок.

В качестве простейшего примера корректирующего кода рассмотрим код, у которого есть всего четыре допустимые кодовые комбинации:

0000000000, 0000011111, 1111100000 и 1111111111

Этот код имеет расстояние, равное 5, что означает, что он может исправлять двойные ошибки. Если приемник получит кодовое слово 0000000111, он поймет, что оригинал должен быть равен 0000011111. Однако если тройная ошибка изменит 0000000000 на 0000000111, ошибка будет исправлена неверно.

Попробуем создать код, состоящий из m информационных и r контрольных битов, способный исправлять одиночные ошибки. Каждому из 2^m допустимых сообщений будет соответствовать n недопустимых кодовых слов, отстоящих от сообщения на расстояние 1. Их можно получить инвертированием каждого из n битов n-битового кодового слова. Таким образом, каждому из 2^m допустимых сообщений должны соответствовать $n + 1$ кодовых комбинаций. Поскольку общее количество возможных кодовых комбинаций равно 2^n, получается, что $(n + 1)2^m \leq 2^n$. Так как $n = m + r$, это требование может быть преобразовано к виду $(m + r + 1) \leq 2^r$. При заданном m данная формула описывает нижний предел требуемого количества контрольных битов для возможности исправления одиночных ошибок.

Этот теоретический нижний предел может быть достигнут на практике с помощью метода Хэмминга (1950). Биты кодового слова нумеруются последовательно слева направо, начиная с 1. Биты с номерами, равными степеням 2 (1, 2, 4, 8, 16 и т. д.), являются контрольными. Остальные биты (3, 5, 6, 7, 9, 10 и т. д.) заполняются m битами данных. Каждый контрольный бит обеспечивает четность

(или нечетность) некоторой группы битов, включая себя самого. Один бит мо жет входить в несколько различных групп битов, четность которых вычисляется Чтобы определить, в какие группы контрольных сумм будет входить бит данны в k-й позиции, следует разложить k по степеням числа 2. Например, $11 = 8 + 2 + 1$ а $29 = 16 + 8 + 4 + 1$. Каждый бит проверяется только теми контрольными бита ми, номера которых входят в этот ряд разложения (например, 11-й бит проверя ется битами 1, 2 и 8).

Когда прибывает кодовое слово, приемник обнуляет счетчик. Затем он прове ряет каждый контрольный бит k ($k = 1, 2, 4, 8, ...$) на четность. Если сумма оказы вается нечетной, он добавляет число k к счетчику. Если после всех провер счетчик равен нулю, значит, все проверки были пройдены успешно. В противно случае он содержит номер неверного бита. Например, если ошибку дают провер ки битов 1, 2 и 8, это означает, что инвертирован бит 11, так как он являетс единственным битом, контролируемым битами 1, 2 и 8. На рис. 3.7 изображен некоторые ASCII-символы, кодированные 11-битовым кодом Хэмминга. Напо минаем, что данные хранятся в битах 3, 5, 6, 7, 9, 10 и 11.

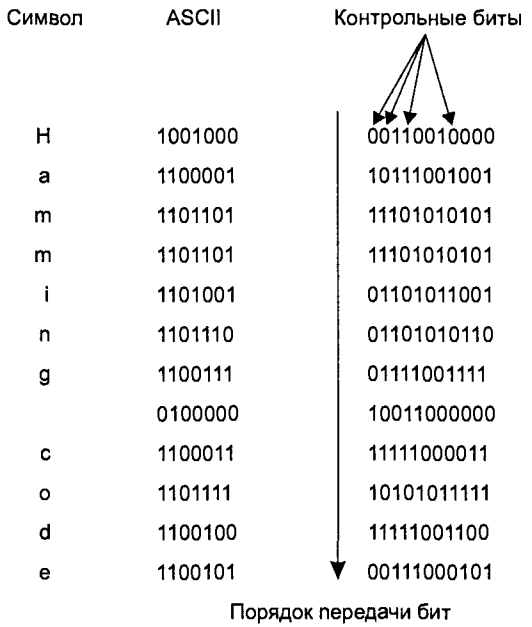


Рис. 3.7. Корректирующий код Хэмминга

Коды Хэмминга позволяют исправлять только одиночные ошибки. Однак один не слишком хитрый трюк позволяет исправлять при помощи этого код и наборы ошибок. Для этого последовательность k кодовых слов организуется виде матрицы, по одному кодовому слову в ряду. Обычно данные передаются п кодовым словам, слева направо. Но чтобы иметь возможность исправлять набо ры ошибок, данные из этой таблицы следует передавать по столбцу за один пр

между которыми будет минимальным. Это расстояние называется **минимальным кодовым расстоянием** кода, или расстоянием всего кода в смысле Хэмминга.

Способности кода по обнаружению и исправлению ошибок зависят от его минимального кодового расстояния. Для обнаружения d ошибок в одном кодовом слове необходим код с минимальным кодовым расстоянием, равным $d + 1$, поскольку d однобитовых ошибок не смогут изменить одну допустимую комбинацию так, чтобы получилась другая допустимая комбинация. Когда приемник встречает запрещенную кодовую комбинацию, он понимает, что при передаче произошла ошибка. Аналогично, для исправления d ошибок в одном кодовом слове требуется код с минимальным кодовым расстоянием, равным $2d + 1$, так как в данном случае даже при d однобитовых ошибках результат окажется ближе к исходному кодовому слову, чем к любому другому, и, следовательно, его можно будет однозначно восстановить.

В качестве простейшего примера кода с обнаружением ошибок рассмотрим код, в котором к данным добавляется один **бит четности**. Бит четности выбирается таким образом, чтобы количество единиц во всем кодовом слове было четным (или нечетным). Например, при посылке числа 10110101 с добавлением бита четности в конце оно становится равным 101101011, тогда как 10110001 преобразуется в 101100010. Код с единственным битом четности имеет кодовое расстояние, равное 2, так как любая однократная ошибка в любом разряде образует кодовое слово с неверной четностью. Такой код может использоваться для обнаружения однократных ошибок.

В качестве простейшего примера корректирующего кода рассмотрим код, у которого есть всего четыре допустимые кодовые комбинации:

0000000000, 0000011111, 1111100000 и 1111111111

Этот код имеет расстояние, равное 5, что означает, что он может исправлять двойные ошибки. Если приемник получит кодовое слово 0000000111, он поймет, что оригинал должен быть равен 0000011111. Однако если тройная ошибка изменит 0000000000 на 0000000111, ошибка будет исправлена неверно.

Попробуем создать код, состоящий из m информационных и r контрольных битов, способный исправлять одиночные ошибки. Каждому из 2^m допустимых сообщений будет соответствовать n недопустимых кодовых слов, отстоящих от сообщения на расстояние 1. Их можно получить инвертированием каждого из n битов n-битового кодового слова. Таким образом, каждому из 2^m допустимых сообщений должны соответствовать $n + 1$ кодовых комбинаций. Поскольку общее количество возможных кодовых комбинаций равно 2^n, получается, что $(n + 1)2^m \leq 2^n$. Так как $n = m + r$, это требование может быть преобразовано к виду $(m + r + 1) \leq 2^r$. При заданном m данная формула описывает нижний предел требуемого количества контрольных битов для возможности исправления одиночных ошибок.

Этот теоретический нижний предел может быть достигнут на практике с помощью метода Хэмминга (1950). Биты кодового слова нумеруются последовательно слева направо, начиная с 1. Биты с номерами, равными степеням 2 (1, 2, 4, 8, 16 и т. д.), являются контрольными. Остальные биты (3, 5, 6, 7, 9, 10 и т. д.) заполняются m битами данных. Каждый контрольный бит обеспечивает четность

(или нечетность) некоторой группы битов, включая себя самого. Один бит может входить в несколько различных групп битов, четность которых вычисляется. Чтобы определить, в какие группы контрольных сумм будет входить бит данных в k-й позиции, следует разложить k по степеням числа 2. Например, $11 = 8 + 2 + 1$, а $29 = 16 + 8 + 4 + 1$. Каждый бит проверяется только теми контрольными битами, номера которых входят в этот ряд разложения (например, 11-й бит проверяется битами 1, 2 и 8).

Когда прибывает кодовое слово, приемник обнуляет счетчик. Затем он проверяет каждый контрольный бит k ($k = 1, 2, 4, 8, ...$) на четность. Если сумма оказывается нечетной, он добавляет число k к счетчику. Если после всех проверок счетчик равен нулю, значит, все проверки были пройдены успешно. В противном случае он содержит номер неверного бита. Например, если ошибку дают проверки битов 1, 2 и 8, это означает, что инвертирован бит 11, так как он является единственным битом, контролируемым битами 1, 2 и 8. На рис. 3.7 изображены некоторые ASCII-символы, кодированные 11-битовым кодом Хэмминга. Напоминаем, что данные хранятся в битах 3, 5, 6, 7, 9, 10 и 11.

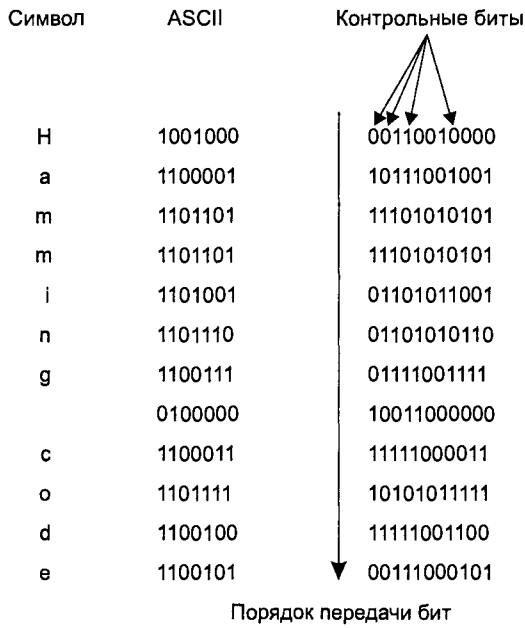


Рис. 3.7. Корректирующий код Хэмминга

Коды Хэмминга позволяют исправлять только одиночные ошибки. Однако один не слишком хитрый трюк позволяет исправлять при помощи этого кода и наборы ошибок. Для этого последовательность k кодовых слов организуется в виде матрицы, по одному кодовому слову в ряду. Обычно данные передаются по кодовым словам, слева направо. Но чтобы иметь возможность исправлять наборы ошибок, данные из этой таблицы следует передавать по столбцу за один при-

ем слева направо. То есть сначала передается первая слева колонка. После передачи всех k битов отправляется вторая слева колонка, и т. д., как показано на рис. 3.7. Когда кадр приходит к получателю, матрица восстанавливается также столбец за столбцом. Если на блок данных наложится пакет ошибок, инвертирующий k соседних битов, она затронет не более 1 бита в каждом кодовом слове. А поскольку код Хэмминга может исправлять одиночные ошибки, то можно будет восстановить весь блок. Данный метод использует kr проверочных битов, благодаря которым блок из km битов данных может выдержать один пакет ошибок длиной не более k бит.

Коды с обнаружением ошибок

Коды с исправлением ошибок широко применяются в беспроводных системах связи, которые славятся зашумленностью среды передачи и, следовательно, большим количеством ошибок по сравнению с системами с медным или оптоволоконным кабелем. Передать что-либо, не используя исправление ошибок, было бы практически невозможно. Однако в системах, где информация передается по проводу, уровень ошибок гораздо ниже, поэтому обнаружение ошибок и повторная передача являются более подходящим методом.

В качестве примера рассмотрим канал с изолированными ошибками, возникающими с вероятностью 10^{-6} на бит. Пусть блок данных состоит из 1000 бит. Для создания кода, исправляющего однократные ошибки, потребуется 10 дополнительных битов на блок. Для мегабита данных это составит 10 000 проверочных битов. Чтобы просто обнаруживать одиночную 1-битовую ошибку, достаточно одного бита четности на блок. На каждые 1000 блоков придется переслать повторно в среднем еще один блок (1001 бит). Таким образом, суммарные накладные расходы на обнаружение ошибки и повторную передачу составят всего 2001 бит на мегабит данных против 10 000 битов, необходимых для кода Хэмминга.

Если к блоку добавлять всего один бит четности, то в случае возникновения пакета ошибок вероятность обнаружения ошибки будет всего лишь 0,5, что абсолютно неприемлемо. Этот недостаток может быть исправлен, если рассматривать каждый посылаемый блок как прямоугольную матрицу n бит шириной и k бит высотой (принцип ее построения был описан ранее). Бит четности должен вычисляться отдельно для каждого столбца и добавляться к матрице в качестве последней строки. Затем матрица передается построчно. Когда блок прибывает, приемник проверяет все биты четности. Если хотя бы один из них неверен, он запрашивает повторную отсылку всего блока. Этот процесс может циклически продолжаться до тех пор, пока весь блок не будет принят без ошибок в битах четности.

Такой метод позволяет обнаружить одиночный пакет ошибок длиной не более n, так как в этом случае будет изменено не более 1 бита в каждом столбце. Однако пакет ошибок длиной $n + 1$ не будет обнаружен, если будут инвертированы первый и последний биты, а все остальные биты останутся неизменными. (Пакет ошибок не предполагает, что все биты неверны, предполагается только,

что по меньшей мере первый и последний биты инвертированы.) Если в блоке при передаче возникнет длинный пакет ошибок или несколько одиночных ошибок, вероятность того, что четность любого из n столбцов будет верной (или неверной), равна 0,5, поэтому вероятность необнаружения ошибки будет равна 2^{-n}.

Хотя приведенная схема в некоторых случаях может быть приемлемой, на практике широко используется другой метод — **полиномиальный код**, также известный как **CRC** (Cyclic Redundancy Check — циклический избыточный код). В основе полиномиальных кодов лежит представление битовых строк в виде многочленов с коэффициентами, равными только 0 или 1. Кадр из k бит рассматривается как список коэффициентов многочлена степени $k - 1$, состоящего из k членов от x^{k-1} до x^0. Старший (самый левый) бит кадра соответствует коэффициенту при x^{k-1}, следующий бит — коэффициенту при x^{k-2}, и т. д. Например, число 110001 состоит из 6 бит и, следовательно, представляется в виде многочлена пятой степени с коэффициентами 1, 1, 0, 0, 0 и 1: $x^5 + x^4 + x^0$.

С данными многочленами осуществляются арифметические действия по модулю 2 в соответствии с алгебраической теорией поля. При этом перенос при сложении и заем при вычитании не производится. И сложение, и вычитание эквивалентны исключающему ИЛИ (XOR):

10011011	00110011	11110000	01010101
+ 11001010	+ 11001101	− 10100110	− 10101111
01010001	11111110	01010110	11111010

Деление чисел осуществляется так же, как и деление обычных двоичных чисел, с той разницей, что вычитание производится по модулю 2, как показано выше.

При использовании циклического кода отправитель и получатель должны сначала договориться насчет **образующего многочлена**, $G(x)$. Старший и младший биты образующего многочлена должны быть равны 1. Для вычисления **контрольной суммы** для некоторого кадра из m бит, соответствующего полиному $M(x)$, необходимо, чтобы этот кадр был длиннее образующего многочлена. Идея состоит в добавлении контрольной суммы в конец кадра таким образом, чтобы получившийся многочлен делился на образующийся многочлен $G(x)$ без остатка. Получатель, приняв кадр, содержащий контрольную сумму, пытается разделить его на $G(x)$. Ненулевой остаток от деления означает ошибку.

Алгоритм вычисления контрольной суммы при этом может быть следующим:

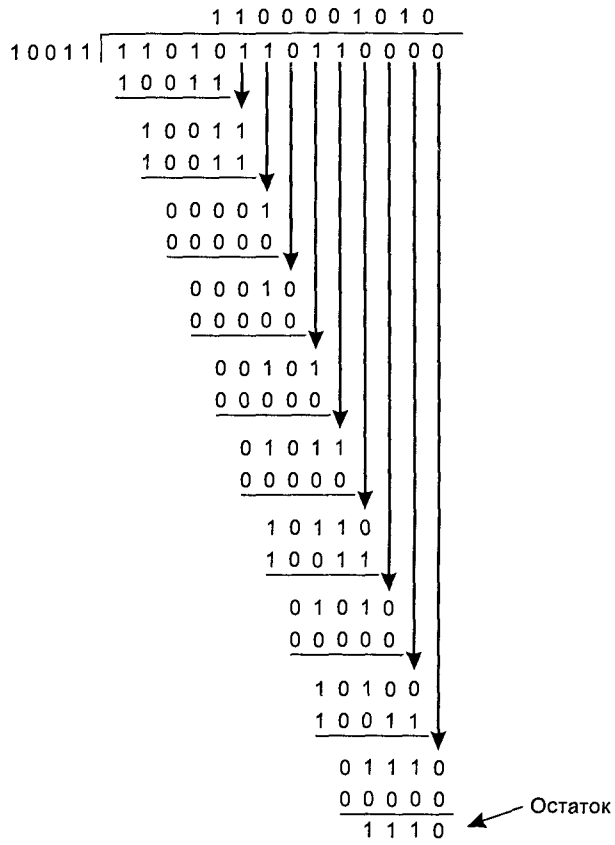
1. Пусть r — степень многочлена $G(x)$. Добавим r нулевых битов в конец кадра так, чтобы он содержал $m + r$ бит и соответствовал многочлену $x^rM(x)$.
2. Разделим по модулю 2 битовую строку, соответствующую многочлену $x^rM(x)$, на битовую строку, соответствующую образующему многочлену $G(x)$.
3. Вычтем по модулю 2 остаток от деления (он должен быть не более r бит) из битовой строки, соответствующей многочлену $x^rM(x)$. Результат и будет передаваемым кадром, который мы будем называть многочленом $T(x)$.

На рис. 3.8 показаны вычисления для кадра 1101011011 и образующего многочлена $G(x) = x^4 + x + 1$.

Кадр: 1 1 0 1 0 1 1 0 1 1

Образующий многочлен: 1 0 0 1 1

Сообщение после добавления 4 нулевых битов: 1 1 0 1 0 1 1 0 1 1 0 0 0 0



Передаваемый кадр: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Рис. 3.8. Вычисление контрольной суммы циклического кода

Должно быть очевидно, что многочлен $T(x)$ делится (по модулю 2) на $G(x)$ без остатка. В любом случае, если вы уменьшите делимое на остаток, то результат должен делиться без остатка. Например, в десятичной системе счисления, если разделить 210 278 на 10 941, то остаток от деления будет равен 2399. Если вычесть 2399 из 210 278, то результат (207 879) будет делиться на 10 941 без остатка.

Теперь проанализируем возможности этого метода. Какие ошибки сможет он обнаруживать? Представим, что произошла ошибка при передаче кадра, так что вместо многочлена $T(x)$ получатель принял $T(x) + E(x)$. Каждый единичный бит многочлена $E(x)$ соответствует инвертированному биту в пакете. Если в много-

члене $E(x)$ k бит равны 1, это означает, что произошло k единичных ошибок. Единичный пакет ошибок характеризуется первой единицей, смесью нулей и единиц и конечной единицей, а остальные биты равны 0.

Получатель делит принятый кадр с контрольной суммой на образующий многочлен $G(x)$, то есть он вычисляет $[T(x) + E(x)]/G(x)$. $T(x)/G(x)$ равно 0, поэтому результат вычислений просто равен $E(x)/G(x)$. Ошибки, которые случайно окажутся кратными образующему многочлену $G(x)$, останутся незамеченными, остальные ошибки будут обнаружены.

Если происходит единичная ошибка, то $E(x) = x^i$, где i означает номер ошибочного бита. Если образующий многочлен $G(x)$ содержит два или более членов, то $E(x)$ никогда не будет делиться на него без остатка, поэтому будут обнаружены все единичные ошибки.

В случае двух изолированных однобитовых ошибок $E(x) = x^i + x^j$, где $i > j$, это можно также записать в виде $E(x) = x^j(x^{i-j} + 1)$. Если предположить, что образующий многочлен $G(x)$ не делится на x, то достаточным условием обнаружения всех двойных ошибок будет неделимость на $G(x)$ многочлена $x^k + 1$ для любого k от 1 до максимального значения $i - j$, то есть до максимальной длины кадра. Известны простые многочлены с небольшими степенями, обеспечивающие защиту для длинных кадров. Например, многочлен $x^{15} + x^{14} + 1$ не является делителем для $x^k + 1$ для любого k от 1 до 32 768.

Если ошибка затронет нечетное количество битов в кадре, многочлен $E(x)$ будет содержать нечетное число членов (например, $x^5 + x^2 + 1$, но не $x^2 + 1$). Интересно, что в системе арифметических операций по модулю 2 многочлены с нечетным числом членов не делятся на $x + 1$. Если в качестве образующего выбрать многочлен, делящийся на $x + 1$, то с его помощью можно обнаружить все ошибки, состоящие из нечетного количества инвертированных битов.

Чтобы показать, что никакой многочлен с нечетным количеством членов не делится на $x + 1$, предположим, что многочлен $E(x)$ содержит нечетное количество членов и делится на $x + 1$. Таким образом, $E(x)$ может быть представлен в виде произведения $E(x) = (x + 1)Q(x)$. Вычислим значение данного многочлена для $x = 1$: $E(1) = (1 + 1)Q(1)$. Поскольку $1 + 1 = 0$ (по модулю 2), то $E(1)$ должен быть равен 0. Однако если многочлен $E(x)$ содержит нечетное количество членов, то замена всех x на 1 будет всегда давать в результате 1. Следовательно, не существует многочлена с нечетным количеством членов, делящегося на $x + 1$.

И наконец, что наиболее важно, полиномиальный код с r контрольными битами будет обнаруживать все пакеты ошибок длиной $\leq r$. Пакет ошибок длиной k может быть представлен в виде многочлена $x^i(x^{k-1} + \ldots + 1)$, где i определяет, насколько далеко от правого конца кадра располагается пакет ошибок. Если образующий многочлен $G(x)$ содержит член x^0, то x^i не будет его множителем, поэтому если степень выражения в скобках меньше степени $G(x)$, то остаток от деления никогда не будет нулевым.

Если длина пакета ошибок равна $r + 1$, то остаток от деления будет нулевым тогда и только тогда, когда пакет ошибок будет идентичен $G(x)$. По определению пакета ошибок, его первый и последний биты должны быть равны 1, поэтому бу-

дет ли он совпадать с образующим многочленом, будет зависеть от $r - 1$ промежуточных битов. Если все комбинации считать равновероятными, то вероятность такой нераспознаваемой ошибки будет равна $(1/2)^{r-1}$.

Также можно показать, что при возникновении пакета ошибок длиннее $r + 1$ битов или нескольких коротких пакетов вероятность пропуска ошибки составляет $(1/2)^r$ при условии, что все комбинации битов равновероятны.

Некоторые образующие многочлены стали международными стандартами. Вот, например, полином, использующийся в IEEE 802:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1.$$

Среди других его полезных свойств имеется и такое: этот многочлен позволяет определяться любые пакеты ошибок длиной не более 32 бит и пакеты, дающие нечетное число бит.

Хотя алгоритм вычисления контрольной суммы может показаться сложным, Питерсон (Peterson) и Браун (Brown) в 1961 году показали, что может быть создана простая схема для аппаратной проверки и подсчета контрольных сумм на основе сдвигового регистра. Эта схема до сих пор применяется почти во всей аппаратуре ЛВС, а иногда и двухабонентских систем.

В течение десятилетий предполагалось, что кадры, контрольные суммы которых вычисляются, содержат случайные биты. Все исследования алгоритмов подсчета контрольных сумм основывались на этом предположении. Недавние исследования реальных данных показали, что эти предположения были совершенно неверными. Как следствие, при некоторых обстоятельствах вероятность прохождения необнаруженных ошибок оказалась значительно выше, чем считалось ранее (Partridge и др., 1995).

Элементарные протоколы передачи данных

Знакомство с протоколами мы начнем с рассмотрения трех протоколов возрастающей сложности. Симулятор этих и следующих протоколов заинтересованные читатели могут найти на веб-сайте (см. предисловие). Прежде чем приступить к изучению протоколов, полезно высказать некоторые допущения, лежащие в основе данной модели связи. Для начала мы предполагаем, что на физическом уровне, уровне передачи данных и сетевом уровне находятся независимые процессы, общающиеся с помощью передачи друг другу сообщений. Во многих случаях процессы физического уровня и уровня передачи данных будут работать на процессоре внутри специальной сетевой микросхемы ввода-вывода, а процесс сетевого уровня — на центральном процессоре. Однако другие варианты реализации также возможны (например, три процесса внутри одной микросхемы ввода-вывода; физический уровень и уровень передачи данных, вызываемые как процедуры процессом сетевого уровня, и т. д.). В любом случае, представление трех уровней в виде отдельных процессов будет служить поддержанию концептуальной чистоты обсуждения, а также подчеркнет независимость уровней.

Другим ключевым допущением будет то, что машина *A* хочет послать на машину *B* длинный поток данных, используя надежный ориентированный на соединение сервис. Позднее мы рассмотрим случай, при котором одновременно машина *B* также хочет послать данные на машину *A*. Предполагается, что у машины *A* имеется бесконечный источник данных, готовых к отправке, и что ей никогда не требуется ждать готовности данных. Когда уровень передачи данных машины *A* запрашивает данные, сетевой уровень всегда готов их ему предоставить. (Это ограничение также будет потом отброшено.)

Также предполагается, что компьютеры не выходят из строя. При передаче могут возникать ошибки, но не проблемы, связанные с поломкой оборудования или случайным сбросом.

При рассмотрении уровня передачи данных пакет, передаваемый ему по интерфейсу сетевым уровнем, рассматривается как чистые данные, каждый бит которых должен быть доставлен сетевому уровню принимающей машины. Тот факт, что сетевой уровень принимающей машины может интерпретировать часть этого пакета как заголовок, не касается уровня передачи данных.

Получив пакет, уровень передачи данных формирует из пакета кадры, добавляя заголовок и концевик пакета передачи данных (см. рис. 3.1). Таким образом, кадр состоит из внедренного пакета, некоторой служебной информации (в заголовке) и контрольной суммы (в концевике). Затем кадр передается уровню передачи данных принимающей машины. Мы будем предполагать наличие соответствующих библиотечных процедур, например, `to_physical_layer` для отправки кадра и `from_physical_layer` для получения кадра. Передающая аппаратура вычисляет и добавляет контрольную сумму (создавая концевик), так что уровень передачи данных может не беспокоиться об этом. Например, может использоваться алгоритм циклических кодов, обсуждавшийся в предыдущем разделе.

Вначале получатель ничего не должен делать. Он просто сидит без дела, ожидая, что что-то произойдет. В приводимых в данной главе примерах протоколов ожидание событий уровнем передачи данных обозначается вызовом процедуры `wait_for_event(&event)`. Эта процедура возвращает управление, только когда что-то происходит (например, прибывает кадр). При этом переменная `event` сообщает, что именно случилось. Наборы возможных событий отличаются в разных протоколах и поэтому будут описываться для каждого протокола отдельно. Следует заметить, что в действительности уровень передачи данных не находится в холостом цикле ожидания событий, как мы предположили, а получает прерывание, когда это событие происходит. При этом он приостанавливает свои текущие процессы и обрабатывает пришедший кадр. Тем не менее, для простоты мы проигнорируем эти детали и предположим, что уровень передачи данных все свое время посвящает работе с одним каналом.

Когда приемная машина получает кадр, аппаратура вычисляет его контрольную сумму. Если она неверна (то есть при передаче возникли ошибки), то уровень передачи данных получает соответствующую информацию (*event* = cksum_err). Если кадр прибывает в целости, уровню передачи данных об этом также сообщается (*event*=frame_arrival), после чего он может получить этот кадр у физического уровня с помощью процедуры `from_physical_layer`. Получив неповрежденный

кадр, уровень передачи данных проверяет управляющую информацию, находящуюся в заголовке кадра, и если все в порядке, часть этого кадра передается сетевому уровню. Заголовок кадра не передается сетевому уровню ни при каких обстоятельствах.

Для запрета передачи сетевому уровню любой части заголовка кадра есть веская причина: поддержание полного разделения сетевого уровня и уровня передачи данных. До тех пор, пока сетевой уровень ничего не знает о формате кадра и протоколе уровня передачи данных, изменения формата и протокола не потребуют изменений программного обеспечения сетевого уровня. Поддержание строгого интерфейса между сетевым уровнем и уровнем передачи данных значительно упрощает разработку программ, так как протоколы различных уровней могут развиваться независимо.

В листинге 3.1 показаны некоторые объявления (на языке C), общие для многих протоколов, обсуждаемых далее. Определены пять структур данных: `boolean`, `seq_nr`, `packet`, `frame_kind` и `frame`. Тип `boolean` представляет собой перечисляемый тип, переменные которого могут принимать значения `true` или `false`. Тип `seq_nr` является целым без знака, используемым для нумерации кадров. Эти последовательные номера могут принимать значения от 0 до числа *MAX_SEQ* включительно, которое определяется в каждом протоколе, использующем его. Тип `packet` является единицей информации, используемой при обмене информацией между сетевым уровнем и уровнем передачи данных одной машины или двумя равноранговыми сетевыми уровнями. В нашей модели пакет всегда состоит из *MAX_PKT* байт, хотя на практике он обычно имеет переменную длину.

Структура `frame` состоит из четырех полей: `kind`, `seq`, `ack` и `info`, — первые три из которых содержат управляющую информацию, а последнее может содержать данные, которые необходимо передать. Эти три управляющих поля вместе называются **заголовком кадра**.

Поле `kind` сообщает о наличии данных в кадре, так как некоторые протоколы отличают кадры, содержащие только управляющую информацию, от кадров, содержащих также и данные. Поля `seq` и `ack` используются соответственно для хранения последовательного номера кадра и подтверждения. Подробнее их использование будет описано далее. Поле данных кадра, `info`, содержит один пакет. В управляющем кадре поле `info` не используется. В реальной жизни используется поле `info` переменной длины, полностью отсутствующее в управляющих кадрах.

Важно понимать взаимоотношения между пакетом и кадром. Сетевой уровень создает пакет, принимая сообщение от транспортного уровня и добавляя к нему заголовок сетевого уровня. Этот пакет передается уровню передачи данных, который включает его в поле `info` исходящего кадра. Когда кадр прибывает на пункт назначения, уровень передачи данных извлекает пакет из кадра и передает его сетевому уровню. Таким образом, сетевой уровень может действовать так, как будто машины обмениваются пакетами напрямую.

В листинге 3.1 также перечислен ряд процедур. Это библиотечные процедуры, детали которых зависят от конкретной реализации, и их внутреннее устройство мы рассматривать не будем. Как уже упоминалось ранее, процедура `wait_for_event` представляет собой холостой цикл ожидания какого-нибудь события. Процедуры `to_network_layer` и `from_network_layer` используются уровнем передачи

данных для передачи пакетов сетевому уровню и для получения пакетов от сетевого уровня соответственно. Обратите внимание: процедуры from_physical_layer и to_physical_layer используются для обмена кадрами между уровнем передачи данных и физическим уровнем, тогда как процедуры to_network_layer и from_network_layer применяются для передачи пакетов между уровнем передачи данных и сетевым уровнем. Другими словами, процедуры to_network_layer и from_network_layer относятся к интерфейсу между уровнями 2 и 3, тогда как процедуры from_physical_layer и to_physical_layer относятся к интерфейсу между уровнями 1 и 2.

В большинстве протоколов предполагается использование ненадежного канала, который может случайно потерять целый кадр. Для обработки подобных ситуаций передающий уровень передачи данных, посылая кадр, запускает таймер. Если за установленный интервал времени ответ не получен, таймер воспринимает это как тайм-аут, и уровень передачи данных получает сигнал прерывания.

В наших примерах протоколов это реализовано в виде значения event=timeout, возвращаемого процедурой wait_for_event. Для запуска и остановки таймера используются процедуры start_timer и stop_timer соответственно. Событие timeout может произойти, только если запущен таймер. Процедуру start_timer разрешается запускать во время работающего таймера. Такой вызов просто переинициализирует часы, чтобы можно было начать отсчет заново (до нового тайм-аута, если таковой будет иметь место).

Процедуры start_ack_timer и stop_ack_timer используются для управления вспомогательными таймерами при формировании подтверждений в особых обстоятельствах.

Процедуры enable_network_layer и disable_network_layer используются в более сложных протоколах, где уже не предполагается, что у сетевого уровня всегда есть пакеты для отправки. Когда уровень передачи данных разрешает работу сетевого уровня, последний получает также разрешение прерывать работу первого, когда ему нужно послать пакет. Такое событие мы будем обозначать как event=network_layer_ready. Когда сетевой уровень отключен, он не может инициировать такие события. Тщательно следя за включением и выключением сетевого уровня, уровень передачи данных не допускает ситуации, в которой сетевой уровень заваливает его пакетами, для которых не осталось места в буфере.

Последовательные номера кадров всегда находятся в пределах от 0 до *MAX_SEQ* (включительно). Число *MAX_SEQ* различно в разных протоколах. Для увеличения последовательного номера кадров на 1 циклически (то есть с обнулением при достижении числа *MAX_SEQ*) используется макрос inc. Он определен в виде макроса, поскольку используется прямо в строке в тех местах программы, где быстродействие является критичным. Как мы увидим позднее в этой книге, производительность сети часто ограничена быстродействием протоколов. Определение простых операций в виде макросов не снижает удобочитаемости программы, увеличивая при этом ее быстродействие. К тому же, поскольку *MAX_SEQ* в разных протоколах будет иметь разные значения, то, определив инкремент в виде макроса, можно один и тот же код без проблем использовать в нескольких протоколах. Такая возможность крайне полезна для симулятора.

Листинг 3.1. Общие объявления для последующих протоколов. Объявления располагаются в файле protocol.h

```
#define MAX_PKT 1024   /* определяет размер пакета в байтах */

typedef enum {false, true} boolean;      /* тип boolean */
typedef unsigned int seq_nr;             /* порядковые номера кадров или подтверждений
*/
typedef struct {unsigned char data[MAX_PKT];} packet;  /* определение пакета */
typedef enum {data, ack, nak} frame_kind;  /* отпределение типа пакета */

typedef struct {                  /* данный уровень занимается транспортировкой кадров
*/
  frame_kind kind;                /* тип кадра */
  seq_nr seq;                     /* порядковый номер */
  seq_nr ack;                     /* номер подтверждения */
  packet info;                    /* пакет сетевого уровня */
} frame;

/* ожидать события и вернуть тип события в переменной event */
void wait_for_event(event_type *event);

/* получить пакет у сетевого уровня для передачи по каналу */
void from_network_layer(packet *p);

/* передать информацию из полученного пакета сетевому уровню */
void to_network_layer(packet *p);

/* получить пришедший пакет у физического уровня и скопировать его в r */
void from_physical_layer(frame *r);

/* передать кадр физическому уровню для передачи */
void to_physical_layer(frame *s);

/* запустить таймер и разрешить событие timeout */
void start_timer(seq_nr k);

/* остановить таймер и запретить событие timeout */
void stop_timer(seq_nr k);

/* запустить вспомогательный таймер и разрешить событие ack_timeout */
void start_ack_timer(void);

/* остановить вспомогательный таймер и запретить событие ack_timeout */
void stop_ack_timer(void);

/* разрешить сетевому уровню инициировать событие network_layer_ready */
void enable_network_layer(void);

/* запретить сетевому уровню инициировать событие network_layer_ready */
void disable_network_layer(void);

/* макрос inc развертывается прямо в строке: Циклически увеличить переменную k */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

Объявления в листинге 3.1 являются частью всех последующих протоколов. Для экономии места и удобства ссылок они были извлечены и собраны вместе, но, по идее, они должны быть объединены с протоколами. В языке C такое объединение производится с помощью директивы препроцессора `#include` с указанием ссылки на файл *protocol.h*, в котором помещаются данные определения.

Неограниченный симплексный протокол

В качестве первого примера мы рассмотрим самый простой протокол. Данные передаются только в одном направлении. Сетевой уровень на передающей и приемной сторонах находится в состоянии постоянной готовности. Временем обработки можно пренебречь. Размер буфера неограничен. И что лучше всего, канал связи между уровнями передачи данных никогда не теряет и не искажает кадры. Этот совершенно нереальный протокол, который мы назовем «утопия», показан в листинге 3.2.

Протокол состоит из двух процедур, `sender` (отправитель) и `receiver` (получатель). Процедура `sender` работает на уровне передачи данных посылающей машины, а процедура `receiver` — на уровне передачи данных принимающей машины. Ни последовательные номера, ни подтверждения не используются, поэтому *MAX_SEQ* не требуется. Единственным возможным событием является `frame_arrival` (то есть прибытие неповрежденного кадра).

Процедура `sender` представляет собой бесконечный цикл, начинающийся с оператора `while`, посылающий данные на линию с максимально возможной скоростью. Тело цикла состоит из трех действий: получения пакета (всегда обязательное) с сетевого уровня, формирования исходящего пакета с помощью переменной `s` и отсылки пакета адресату. Из служебных полей кадра данный протокол использует только поле `info`, поскольку другие поля относятся к обработке ошибок и управлению потоком, которые в данном протоколе не применяются.

Процедура принимающей стороны ничуть не сложнее. Вначале она ожидает, пока что-нибудь произойдет, причем единственно возможным событием в данном протоколе может быть получение неповрежденного пакета. Когда пакет появляется, процедура `wait_for_event` возвращает управление, при этом переменной `event` присваивается значение `frame_arrival` (которое все равно игнорируется). Обращение к процедуре `from_physical_layer` удаляет вновь прибывший кадр из аппаратного буфера и помещает его в переменную `r`. Наконец, порция данных передается сетевому уровню, а уровень передачи данных отправляется ждать следующий кадр.

Листинг 3.2. Неограниченный симплексный протокол

```
/* Протокол 1 (утопия) обеспечивает только одностороннюю передачу данных – от
отправителя к получателю. Предполагается, что в канале связи нет ошибок и что
получатель способен мгновенно обрабатывать получаемые данные. Соответственно,
отправитель в цикле передает данные на линию с максимально доступной для него
скоростью. */

typedef enum {frame_arrival} event_type;
```

```
#include "protocol.h"

void sender1(void)
{
  frame s;                              /* буфер для исходящего кадра */
  packet buffer;                        /* буфер для исходящего пакета */

  while (true) {
      from_network_layer(&buffer);      /* получить у сетевого уровня пакет для передачи
*/
      s.info = buffer;                  /* скопировать его в кадр s для передачи */
      to_physical_layer(&s);            /* послать кадр s */
  }                                     /* Мы дни за днями шепчем «Завтра, завтра».
                                           Так тихими шагами жизнь ползет
                                           К последней недописанной странице.
                                               - Макбет, V, v */
}

void receiver1(void)
{
  frame r;
  event_type event; /* заполняется процедурой ожидания событий, но не используется
здесь */

  while (true) {
      wait_for_event(&event); /* единственное возможное событие – прибытие кадра,
событие frame_arrival */
      from_physical_layer(&r);          /* получить прибывший кадр */
      to_network_layer(&r.info);        /* передать данные сетевому уровню */
  }
}
```

Симплексный протокол с ожиданием

Теперь мы отбросим самое нереальное предположение, использованное в протоколе 1, — способность получающего сетевого уровня мгновенно обрабатывать приходящие данные (или, что то же самое, наличие у получающего уровня передачи данных неограниченного буферного пространства, в которое он помещает все приходящие кадры). Сохраняется предположение о том, что в канале связи нет ошибок. Линия связи остается симплексной.

Основная проблема, которую нам предстоит решить, — как предотвратить ситуацию, когда отправитель посылает данные быстрее, чем получатель может их обработать. То есть если получателю требуется время Δt, чтобы выполнить процедуры `from_physical_layer` и `to_network_layer`, то отправитель должен передавать со средней скоростью меньшей, чем один кадр за интервал времени Δt. Более того, если мы предполагаем, что в принимающей аппаратуре не производится автоматической буферизации, то отправитель не должен посылать новый кадр до тех пор, пока старый кадр не будет считан процедурой `from_physical_layer`. В противном случае новый кадр окажется записанным поверх старого.

При некоторых обстоятельствах (например, при синхронной передаче, когда уровень передачи данных принимающей машины обрабатывает всего одну входную линию) может быть достаточно всего лишь вставить задержку в передающую программу протокола 1, снизив скорость его работы настолько, чтобы уберечь принимающую сторону от забивания данными. Однако в реальных условиях обычно каждый уровень передачи данных должен одновременно обрабатывать несколько линий, и время, необходимое на обработку одного кадра, может меняться в довольно значительных пределах. Если разработчик сети может рассчитать наихудшую ситуацию для приемника, он может запрограммировать настолько медленную работу отправителя, что даже при самой медленной обработке поступающих кадров получатель будет успевать их обрабатывать. Недостаток такого подхода в его консерватизме. В данном случае использование пропускной способности будет намного ниже оптимального уровня. Исключением может быть только один случай — когда время реакции принимающего уровня передачи данных изменяется очень незначительно.

Лучшим решением данной проблемы является обратная связь со стороны получателя. Передав пакет сетевому уровню, получатель посылает небольшой управляющий пакет отправителю, разрешая тем самым посылать следующий кадр. Отправитель, отослав кадр, должен ждать разрешения на отправку следующего кадра.

Протоколы, в которых отправитель посылает один кадр, после чего ожидает подтверждения, называются **протоколами с ожиданием**. В листинге 3.3 приведен пример симплексного протокола с ожиданием.

Листинг 3.3. Симплексный протокол с ожиданием

```
/* Протокол 2 (с ожиданием) также обеспечивает только одностороннюю передачу данных,
от отправителя к получателю. Снова предполагается, что в канале связи нет ошибок.
Однако на этот раз емкость буфера получателя ограничена, и, кроме того, ограничена
скорость обработки данных получателя. Поэтому протокол должен не допускать отправления
данных быстрее, чем получатель способен их обработать. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;           /* буфер для исходящего кадра */
  packet buffer;     /* буфер для исходящего пакета */
  event_type event;  /* единственное возможное событие – прибытие кадра (событие
frame_arrival)*/

  while (true) {
      from_network_layer(&buffer);  /* получить у сетевого уровня пакет для передачи
*/
      s.info = buffer;              /* скопировать его в кадр s для передачи */
      to_physical_layer(&s);        /* до свидания, кадрик, до свидания */
      wait_for_event(&event);       /* не продолжать, пока на это не будет получено
разрешения */
  }
}
```

```
void receiver2(void)
{
  frame r, s;                    /* буферы для кадров */
  event_type event;              /* frame_arrival является единственным возможным
событием */
  while (true) {
      wait_for_event(&event);        /* единственное возможное событие – прибытие кадра
(событие frame_arrival)*/
      from_physical_layer(&r);       /* получить прибывший кадр */
      to_network_layer(&r.info);     /* передать данные сетевому уровню */
      to_physical_layer(&s);         /* передать пустой кадр, чтобы разбудить
отправителя */
  }
}
```

Как и в протоколе 1, отправитель в начале цикла работы получает пакет от сетевого уровня, формирует из него кадр и отправляет кадр по линии связи. Однако теперь, в отличие от протокола 1, отправитель должен ждать прибытия кадра с подтверждением, прежде чем он пойдет на следующую итерацию цикла и обратится к сетевому уровню за следующим пакетом. В данной модели уровень передачи данных отправителя даже не должен просматривать полученный по линии кадр: его содержимое не имеет значения, поскольку сам кадр означает только одно: подтверждение.

Единственное отличие процедуры `receiver2` от `receiver1` заключается в том, что после передачи пакета сетевому уровню `receiver2` посылает кадр подтверждения обратно отправителю, после чего идет на следующую итерацию цикла. Поскольку для отправителя важно только прибытие ответного кадра, а не его содержание, то получателю не нужно заполнять кадр специальной информацией.

Симплексный протокол для зашумленных каналов

Теперь рассмотрим реальную ситуацию: канал связи, в котором могут быть ошибки. Кадры могут либо портиться, либо теряться. Однако мы будем предполагать, что если кадр будет поврежден при передаче, то приемная аппаратура определит это при подсчете контрольной суммы. Если кадр будет поврежден таким образом, что контрольная сумма совпадет, что очень маловероятно, то этот протокол (и любой другой протокол) передаст неверный пакет сетевому уровню.

На первый взгляд может показаться, что нужно лишь добавить таймер к протоколу 2. Получатель будет возвращать подтверждение только в случае получения правильных данных. Неверные пакеты будут просто игнорироваться. Через некоторое время у отправителя истечет интервал времени, и он отправит кадр еще раз. Этот процесс будет повторяться до тех пор, пока кадр, наконец, не прибудет в целости.

В приведенной выше схеме имеется один критический недостаток. Прежде чем читать дальше, попытайтесь понять, что же неверно в данном алгоритме.

Чтобы осознать, чем плох данный вариант протокола, вспомните, что задача уровня передачи данных заключается в предоставлении безошибочной прозрачной связи между двумя процессами сетевого уровня. Сетевой уровень машины *A* передает серию пакетов своему уровню передачи данных, который должен гарантировать доставку идентичной серии пакетов сетевому уровню машины *B* ее уровнем передачи данных. В частности, сетевой уровень машины *B* не может распознать недостачу пакета или дублирование пакета, поэтому уровень передачи данных должен гарантировать, что дублирования пакетов не произойдет ни при каких обстоятельствах.

Рассмотрим следующий сценарий.

1. Сетевой уровень машины *A* передает пакет 1 своему уровню передачи данных. Пакет доставляется в целости на машину *B* и передается ее сетевому уровню. Машина *B* посылает кадр подтверждения назад на машину *A*.
2. Кадр подтверждения полностью теряется в канале связи. Он просто не попадает на машину *A*. Все было бы намного проще, если бы терялись только информационные — но не управляющие — кадры, однако канал связи, к сожалению, не делает между ними большой разницы.
3. У уровня передачи данных машины *A* внезапно истекает отведенный интервал времени. Не получив подтверждения, он предполагает, что посланный им кадр с данными был поврежден или потерян, и посылает этот кадр еще раз.
4. Дубликат кадра прибывает на уровень передачи данных машины *B* и передается на сетевой уровень. Если машина *A* посылала на машину *B* файл, то часть этого файла продублировалась, таким образом, копия файла на машине *B* будет неверной. Другими словами, протокол допустил ошибку.
5. Понятно, что необходим некий механизм, с помощью которого получатель смог бы отличать новый кадр от переданного повторно. Наиболее очевидным способом решения данной проблемы является помещение отправителем порядкового номера кадра в заголовке кадра. Тогда по номеру кадра получатель сможет понять, новый это кадр или дубликат.

Поскольку отводить в кадре много места под заголовок нежелательно, возникает вопрос: каково минимальное количество бит, достаточное для порядкового номера кадра? Единственная неопределенность в данном протоколе может возникнуть между кадром m и следующим за ним кадром $m + 1$. Если кадр m потерян или поврежден, получатель не подтвердит его и отправитель пошлет его еще раз. Когда он будет успешно принят, получатель пошлет отправителю подтверждение. Именно здесь находится источник потенциальной проблемы. В зависимости от того, будет получено подтверждение или нет, отправитель может послать кадр m или кадр $m + 1$.

Отправитель может начать посылать кадр $m + 2$ только когда получит подтверждение получения кадра $m + 1$. Но это означает, что кадр m уже был отправлен и подтверждение его получения было отправлено и получено. Следовательно, неопределенность может возникнуть только между двумя соседними кадрами.

Таким образом, должно быть достаточно всего одного бита информации (со значением 0 или 1). В каждый момент времени получатель будет ожидать при-

бытия кадра с определенным порядковым номером. Кадр с неверным номером будет отбрасываться как дубликат. Кадр с верным номером принимается, передается сетевому уровню, после чего номер следующего ожидаемого кадра увеличивается по модулю 2 (то есть 0 становится 1, а 1 — 0).

Пример подобного протокола приведен в листинге 3.4. Протоколы, в которых отправитель ожидает положительного подтверждения, прежде чем перейти к пересылке следующего кадра, часто называются **PAR** (Positive Acknowledgement with Retransmission — положительное подтверждение с повторной передачей) или **ARQ** (Automatic Repeat reQuest — автоматический запрос повторной передачи). Подобно протоколу 2, он также передает данные только в одном направлении.

Листинг 3.4. Протокол с положительным подтверждением и повторной передачей

```
/* Протокол 3 (PAR) обеспечивает симплексную передачу данных по ненадежному каналу. */

#define MAX_SEQ 1                       /* в протоколе 3 должно быть равно 1 */
typedef enum  {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;            /* порядковый номер следующего исходящего кадра
*/
  frame s;                              /* временная переменная */
  packet buffer;                        /* буфер для исходящего пакета */
  event_type event;

  next_frame_to_send = 0;               /* инициализация исходящих последовательных
номеров */
  from_network_layer(&buffer);          /* получить первый пакет у сетевого уровня */
  while (true) {
      s.info = buffer;                /* сформировать кадр для передачи */
      s.seq = next_frame_to_send; /* вставить порядковый номер в кадр */
      to_physical_layer(&s);          /* послать кадр по каналу */
      start_timer(s.seq);             /* запустить таймер ожидания подтверждения */
      wait_for_event(&event); /* ждать события frame_arrival, cksum_err или timeout */
      if (event == frame_arrival) {
          from_physical_layer(&s); /* получить подтверждение */
          if (s.ack == next_frame_to_send) {
              from_network_layer(&buffer); /* получить следующий пакет у сетевого
уровня */
              inc(next_frame_to_send);  /* инвертировать значение переменной next_
frame_to_send */
          }
      }
  }
}

void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
```

```
  event_type event;

  frame_expected = 0;
  while (true) {
      wait_for_event(&event); /* ожидание возможных событий: frame_arrival, cksum_err
*/
      if (event == frame_arrival) {
          /* прибыл неповрежденный кадр */
          from_physical_layer(&r);          /* получить прибывший кадр */
          if (r.seq == frame_expected) {    /* именно этот кадр и ожидался */
              to_network_layer(&r.info);      /* передать данные сетевому уровню */
              inc(frame_expected);            /* в следующий раз ожидать кадр с другим
порядковым номером */
          }
          s.ack = 1 - frame_expected;       /* номер кадра, для которого посылается
подтверждение */
          to_physical_layer(&s);            /* ни одно из полей не используется */
      }
  }
}
```

Протокол 3 отличается от своих предшественников тем, что и отправитель, и получатель запоминают номера кадров. Отправитель запоминает номер следующего кадра в переменной `next_frame_to_send`, а получатель запоминает порядковый номер следующего ожидаемого кадра в переменной `frame_expected`. Перед бесконечным циклом в каждой процедуре размещена короткая фаза инициализации.

Передав кадр, отправитель запускает таймер. Если он уже был запущен, он настраивается на отсчет нового полного интервала времени. Период выбирается достаточно большим, чтобы даже в худшей ситуации кадр успел дойти до получателя, получатель успел его обработать и подтверждение успело вернуться к отправителю. Только по истечении отведенного времени можно утверждать, что потерялся кадр или его подтверждение, а значит, необходимо послать дубликат. Если время, после которого наступает тайм-аут, сделать слишком коротким, то передающая машина будет повторно посылать слишком много кадров, в которых нет необходимости. Хотя лишние кадры в данном случае не повлияют на правильность приема данных, они повлияют на производительность системы.

После передачи кадра отправитель запускает таймер и ждет какого-либо события. Возможны три ситуации: либо придет неповрежденный кадр подтверждения, либо будет получен поврежденный кадр подтверждения, либо просто истечет интервал времени. В первом случае отправитель возьмет у сетевого уровня следующий пакет и положит его в буфер поверх старого пакета. Кроме того, он увеличит порядковый номер кадра. Если же прибудет поврежденный кадр подтверждения или подтверждение не придет вовсе, то ни буфер, ни номер не будут изменены и будет послан дубликат кадра.

Когда неповрежденный кадр прибывает к получателю, проверяется его номер. Если это не дубликат, то кадр принимается и передается сетевому уровню, после чего формируется подтверждение. Дубликаты и поврежденные кадры на сетевой уровень не передаются.

Протоколы скользящего окна

В предыдущих протоколах информационные кадры передавались только в одну сторону. В большинстве практических ситуаций требуется передача данных в обоих направлениях. Один из способов получения дуплексной передачи данных заключается в создании двух отдельных симплексных каналов связи, по которым данные передаются в противоположных направлениях. При этом получается два отдельных физических канала, каждый из которых имеет прямой канал для данных и обратный канал для подтверждений. В обоих случаях пропускная способность обратных каналов почти не используется. В результате пользователь платит за два канала, но использует емкость одного.

Более прогрессивной идеей представляется использование одного канала для передачи данных в обоих направлениях. В конце концов, ведь в протоколах 2 и 3 кадры уже передавались по каналу в двух направлениях, а обратный канал обладает той же пропускной способностью, что и прямой. В такой модели кадры с данными от машины *A* для машины *B* перемешиваются с кадрами подтверждений от *A* к *B*. Получатель может отличить кадр с данными от кадра с подтверждением по специальному полю `kind` заголовка кадра.

Помимо чередования кадров с подтверждениями и информационных кадров, возможно и другое улучшение протокола. Приняв кадр с данными, получатель может не посылать сразу кадр с подтверждением, а подождать, пока сетевой уровень даст ему следующий пакет. Подтверждение добавляется к исходящему информационному кадру с помощью поля `ack` заголовка кадра. В результате для передачи подтверждения почти не будет затрачено ресурсов. Подобная техника называется **piggybacking** (комбинированная, или ярусная, перевозка).

Основное преимущество совмещения передачи прямых и обратных пакетов заключается в улучшенном использовании пропускной способности канала. Поле `ack` в заголовке кадра занимает всего несколько бит, тогда как отдельный кадр потребует заголовка и контрольной суммы. Кроме того, чем меньше количество прибывающих кадров, тем меньше прерываний работы программы, связанных с этим событием, и, возможно, меньше буферов у получателя (в зависимости от способа организации программного обеспечения получателя). В следующем рассматриваемом нами протоколе расходы на совмещение передачи прямых и обратных пакетов составляют всего 1 бит заголовка кадра. Эти расходы редко превышают несколько бит.

Однако при совмещении передачи прямых и обратных пакетов в протоколе появляются новые проблемы. Как долго должен уровень передачи данных ждать пакета, с которым следует переслать подтверждение? Если уровень передачи данных будет ждать дольше, чем отправитель, то последний пошлет кадр повторно, что неприемлемо. Если бы уровень передачи данных мог предсказывать будущее, он бы знал, ждать ему пакета или отправлять подтверждение отдельным кадром. Это, конечно, невозможно, поэтому следует установить еще один интервал ожидания (меньший, чем интервал ожидания отправителя), по истечении которого подтверждение отправляется отдельным кадром. Если же сетевой уровень

успеет передать уровню передачи данных пакет, то подтверждение будет отослано вместе с этим пакетом в одном кадре.

Следующие три протокола являются двунаправленными и принадлежат к классу протоколов **скользящего окна** (sliding window). Как будет показано далее, они отличаются друг от друга эффективностью, сложностью и требованиями к размерам буфера. Во всех протоколах скользящего окна каждый исходящий кадр содержит порядковый номер (варьирующийся от 0 до некоего максимума). Поскольку на этот номер обычно отводится поле размером n бит, максимальное значение номера составляет $2^n - 1$. В протоколах скользящего окна с ожиданием обычно на это поле отводится всего один бит, что ограничивает порядковый номер значениями 0 и 1, однако в более сложных версиях может использоваться произвольное значение n.

Сущность всех протоколов скользящего окна заключается в том, что в любой момент времени отправитель работает с определенным набором порядковых номеров, соответствующих кадрам, которые ему разрешено посылать. Про такие кадры говорят, что они попадают в **посылающее окно**. Аналогично получатель работает с **принимающим окном**, соответствующим набору кадров, которые ему позволяется принять. Окно получателя и окно отправителя не обязаны иметь одинаковые нижний и верхний пределы и даже не обязаны быть одного размера. Размеры одних протоколов фиксируются, а размеры других могут увеличиваться или уменьшаться по мере передачи или приема кадров.

Хотя данные протоколы предоставляют уровню передачи данных большую свободу в вопросе, касающемся порядка передачи и приема кадров, требование доставки пакетов сетевому уровню принимающей машины в том же прядке, в котором они были получены от сетевого уровня передающей машины, сохраняется. Также сохраняется аналогичное требование к физическому уровню, касающееся сохранения порядка доставки кадров. То есть физический уровень должен функционировать подобно проводу, доставляя все кадры в том порядке, в котором они были посланы.

Порядковые номера в окне отправителя соответствуют кадрам, которые уже отправлены, но на которые еще не получены подтверждения. Получаемому от сетевого уровня пакету дается наибольший порядковый номер, и верхняя граница окна увеличивается на единицу. Когда поступает подтверждение, на единицу возрастает нижняя граница окна. Таким образом, окно постоянно содержит список неподтвержденных кадров.

Так как кадры, находящиеся в окне отправителя, могут быть потеряны или повреждены во время передачи, отправитель должен хранить их в памяти на случай возможной повторной передачи. Таким образом, если максимальный размер кадра равен n, то отправителю потребуется n буферов для хранения неподтвержденных кадров. Если окно достигает максимального размера, уровень передачи данных должен отключить сетевой уровень до тех пор, пока не освободится буфер.

Окно принимающего уровня передачи данных соответствует кадрам, которые он может принять. Любой кадр, не попадающий в окно, игнорируется без каких-либо комментариев. Когда прибывает кадр с порядковым номером, соответствующим нижнему краю окна, он передается на сетевой уровень, формируется под-

тверждение и окно сдвигается на одну позицию. В отличие от окна отправителя, окно получателя всегда сохраняет свой изначальный размер. Окно единичного размера говорит о том, что уровень передачи данных может принимать кадры только в установленном порядке, однако при больших размерах окна это не так. Сетевому уровню, напротив, данные всегда предоставляются в строгом порядке, независимо от размера окна уровня передачи данных.

На рис. 3.9 показан пример для окна с максимальным размером 1. Вначале кадров в окне нет, поэтому само окно пустое и его верхний и нижний края совпадают.

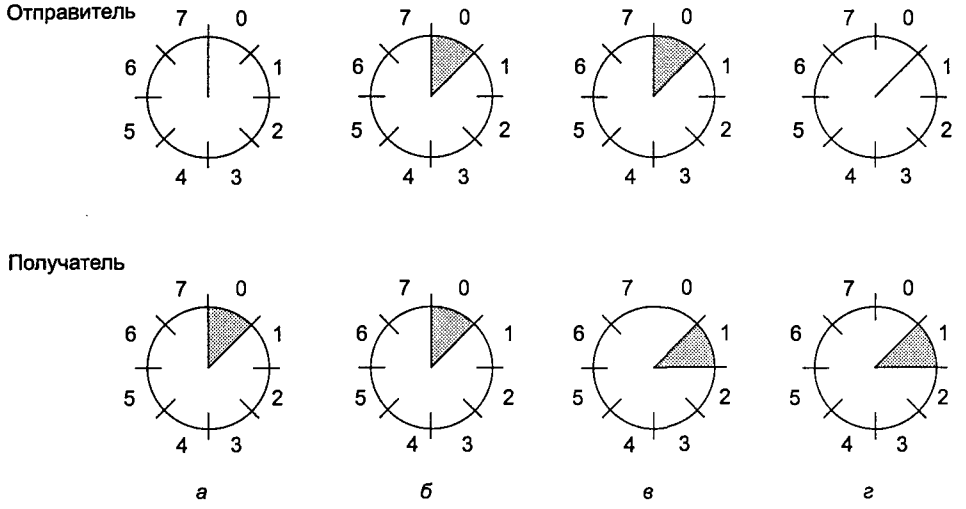


Рис. 3.9. Скользящее окно размера 1 с 3-битовым порядковым номером: начальная ситуация (*а*); после отправки первого кадра (*б*); после приема первого кадра (*в*); после приема первого подтверждения (*г*)

Протокол однобитового скользящего окна

Прежде чем рассматривать общий случай, изучим протокол скользящего окна с максимальным размером окна, равным 1. Такой протокол использует метод ожидания, поскольку, отослав кадр, отправитель, прежде чем послать следующий кадр, должен дождаться подтверждения.

Данный протокол приведен в листинге 3.5. Как и другие протоколы, он начинается с определения некоторых переменных. Переменная `next_frame_to_send` содержит номер кадра, который отправитель пытается послать. Аналогично переменная `frame_expected` хранит номер кадра, ожидаемого получателем. В обоих случаях возможными значениями могут быть только 0 и 1.

В нормальной ситуации только один уровень передачи данных может начинать передачу. Другими словами, только одна из программ должна содержать обращения к процедурам `to_physical_layer` и `start_timer` вне основного цикла. Если оба уровня передачи данных начинают одновременно передавать, возникает не-

простая ситуация. Начинающая машина получает первый пакет от своего сетевого уровня, создает из него кадр и посылает его. Когда этот (или другой) кадр прибывает, получающий уровень передачи данных проверяет, не является ли этот кадр дубликатом, аналогично протоколу 3. Если это тот кадр, который ожидался, он передается сетевому уровню и окно получателя сдвигается вверх.

Листинг 3.5. 1-битовый протокол скользящего окна

```
/* Протокол 4 (скользящее окно) является дуплексным и более надежным, чем протокол 3.
*/

#define MAX_SEQ 1                    /* в протоколе 4 должно быть равно 1 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void protocol4 (void)
{
  seq_nr next_frame_to_send;         /* только 0 или 1 */
  seq_nr frame_expected;             /* только 0 или 1 */
  frame r, s;                        /* временная переменная */
  packet buffer;                     /* текущий посланный пакет */
  event_type event;

  next_frame_to_send = 0;            /* номер следующего кадра в исходящем потоке */
  frame_expected = 0;                /* номер ожидаемого кадра */
  from_network_layer(&buffer);       /* получить первый пакет у сетевого уровня */
  s.info = buffer;                   /* подготовить первый кадр для передачи */
  s.seq = next_frame_to_send;        /* вставить порядковый номер в кадр */
  s.ack = 1 - frame_expected;        /* подтверждение, посылаемое «верхом» на кадре
данных */
  to_physical_layer(&s);             /* послать кадр по каналу */
  start_timer(s.seq);                /* запустить таймер ожидания подтверждения */

  while (true) {
      wait_for_event(&event);        /* ждать возможного события: frame_arrival, cksum_
err или timeout */
      if (event == frame_arrival) { /* кадр прибыл в целости */
          from_physical_layer(&r);  /* получить кадр */

          if (r.seq == frame_expected) {
              /* обработать входящий поток кадров */
              to_network_layer(&r.info);  /* передать пакет сетевому уровню */
              inc(frame_expected);  /* инвертировать порядковый номер кадра,
ожидаемого в следующий раз */
          }

          if (r.ack == next_frame_to_send) { /* обработать исходящий поток кадров */
              from_network_layer(&buffer);  /* получить следующий пакет у сетевого
уровня */
              inc(next_frame_to_send);  /* инвертировать порядковый номер посылаемого
кадра */
          }
      }
```

```
        s.info = buffer;                /* подготовить кадр для передачи */
        s.seq = next_frame_to_send;     /* вставить порядковый номер в кадр */
        s.ack = 1 - frame_expected;     /* порядковый номер последнего полученного кадра
*/
        to_physical_layer(&s);          /* передать кадр */
        start_timer(s.seq);             /* запустить таймер ожидания подтверждения */
    }
}
```

Поле подтверждения содержит номер последнего полученного без ошибок кадра. Если этот номер совпадает с номером кадра, который пытается передать отправитель, последний понимает, что этот кадр успешно принят получателем, и что он может пересылать следующий кадр. В противном случае он должен продолжать попытки переслать тот же кадр.

Теперь давайте изучим протокол 4 и посмотрим, насколько он устойчив к нестандартным ситуациям. Предположим, что машина *A* пытается послать кадр 0 машине *B*, а машина *B* пытается послать кадр 0 машине *A*. Предположим также, что на машине *A* установлен слишком короткий период ожидания подтверждения. Соответственно, машина *A* посылает серию одинаковых кадров со значениями полей `seq=0` и `ack=1`.

Когда первый неповрежденный кадр прибудет на машину *B*, он будет принят, и значение переменной `frame_expected` будет установлено равным 1. Все последующие кадры будут проигнорированы, поскольку машина *B* будет теперь ожидать кадра с порядковым номером 1, а не 0. Более того, поскольку у всех кадров дубликатов значение поля `ack=1`, а машина *B* продолжает ожидать подтверждения для кадра 0, то *B* и не станет запрашивать новый пакет у своего сетевого уровня.

В ответ на каждый отвергнутый дубликат, присылаемый машиной *A*, машина *B* посылает кадр, содержащий поля `seq=0` и `ack=0`. Наконец, один из этих кадров принимается машиной *A*, в результате чего машина *A* переходит к передаче

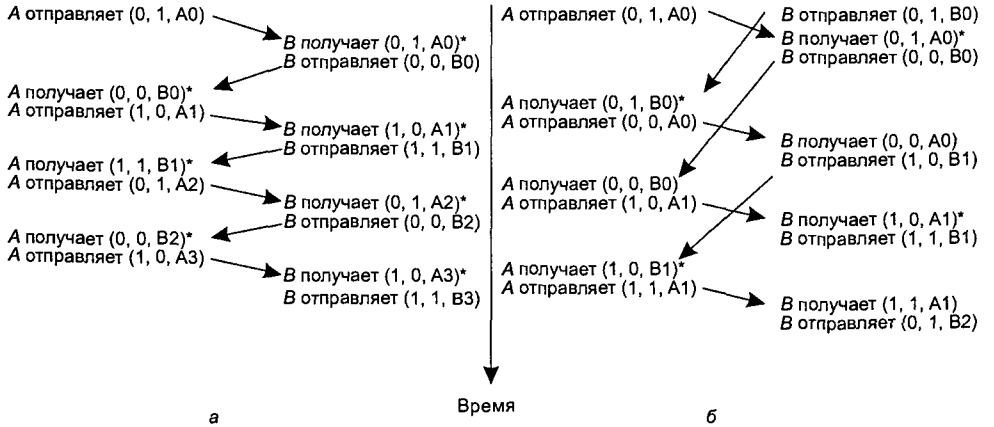


Рис. 3.10. Два сценария для протокола 4: нормальная ситуация (*а*); нештатная ситуация (*б*). Обозначения: (seq, ack, номер пакета). Звездочка означает, что сетевой уровень принял пакет

следующего пакета. Никакая комбинация потерянных кадров или преждевременно истекших интервалов ожидания не может заставить этот протокол ни выдать сетевому уровню дубликат пакета, ни пропустить пакет, ни зависнуть.

Однако, если обе стороны одновременно вышлют друг другу начальный пакет, возникает запутанная ситуация, проиллюстрированная на рис. 3.10. В левой части рисунка показано нормальное функционирование протокола. Правая часть рисунка демонстрирует аномальную ситуацию. Если машина *B* ожидает первого кадра от машины *A*, прежде чем послать свой первый кадр, то последовательность будет такой, как показана в левой части рисунка. При этом принимается каждый кадр. Однако если машины *A* и *B* одновременно начнут передачу, их первые кадры пересекутся и уровни передачи данных попадут в ситуацию *б*. В ситуации *а* с каждым кадром прибывал новый пакет и дубликатов нет. В ситуации *б* половина кадров содержит дубликаты несмотря на то, что ошибок в канале связи не было. Подобные ситуации могут возникнуть в результате преждевременного истечения периода ожидания, даже если одна сторона начнет диалог первой. В самом деле, если время ожидания истечет слишком быстро, кадр может быть послан три и более раз.

Протокол с возвратом на *n*

До сих пор мы по умолчанию подразумевали, что время, необходимое на передачу кадра от отправителя к получателю, и время, необходимое на передачу подтверждения от получателя к отправителю, пренебрежимо малы. Иногда это предположение является совершенно неверным. В таких ситуациях большое время прохождения кадров по сети может значительно снижать эффективность использования пропускной способности канала. В качестве примера рассмотрим спутниковый канал связи с пропускной способностью 50 Кбит/с и временем, требуемым для прохождения сигнала в оба конца, равным 500 мс. Попытаемся использовать протокол 4 для пересылки кадров размером в 1000 бит через спутник. В момент времени $t = 0$ отправитель начинает посылать первый кадр. В момент времени $t = 20$ мс кадр полностью послан. В момент времени $t = 270$ мс получатель принял кадр полностью и отправил обратно подтверждение. В итоге в лучшем случае только через 520 мс после начала передачи кадра подтверждение будет получено отправителем. В данном случае еще предполагается, что приемник не тратит времени на обработку принятого кадра и подтверждение такое короткое, что временем его передачи и приема можно пренебречь. Это означает, что передающая машина была заблокирована в течение 500/520, или 96 % времени. Другими словами, использовалось только 4 % доступной пропускной способности. Очевидно, что сочетание большого времени прохождения сигнала, высокой пропускной способности и коротких кадров совершенно неприемлемо с точки зрения эффективности.

Описанная проблема является следствием правила, заставлявшего отправителя дожидаться подтверждения, прежде чем посылать следующий кадр. Смягчив это требование, можно значительно повысить эффективность. Решение проблемы заключается в разрешении отправителю послать не один кадр, а несколько, например w, прежде чем остановиться и перейти в режим ожидания подтвержде-

ний. Можно подобрать число w так, чтобы отправитель мог безостановочно посылать кадры. В приведенном ранее примере w должно быть равно, по меньшей мере, 26. Отправитель начинает, как и ранее, с передачи кадра 0. К тому моменту, когда он закончит отправку 26 кадров (в момент времени t = 520 мс), как раз прибудет подтверждение кадра 0. Затем подтверждения станут прибывать каждые 20 мс. Таким образом, отправитель будет получать разрешения на передачу следующего кадра как раз вовремя. В любой момент времени у отправителя будет 25 или 26 неподтвержденных кадров и, следовательно, достаточно будет окна размером 26.

Такая техника называется **конвейерной обработкой**. Если пропускная способность канала равна b бит/с, размер кадра равен l бит, то время передачи одного кадра составит l/b с. Пусть время прохождения сигнала по каналу в оба конца равно R с. После отправки последнего бита информационного кадра в течение $R/2$ с происходит его пересылка получателю и в течение еще по крайней мере $R/2$ с — прием подтверждения отправителем. Поэтому общая задержка составляет R с. В протоколе с ожиданием линия занята в течение l/b с и свободна в течение R с, что дает коэффициент эффективности использования линии равный $l/(l + bR)$. При $l < bR$ эффективность использования линии будет менее 50 %. Конвейерный режим может использоваться для загрузки избежания простаивания линии, если время прохождения сигнала значительно по сравнению со временем самой передачи. Если же задержка распространения сигнала мала, дополнительное усложнение протокола не является оправданным.

При конвейерном режиме передачи кадров по ненадежному каналу возникает ряд серьезных проблем. Во-первых, что произойдет, если повредится или потеряется кадр в середине длинного потока? Большое количество последующих кадров прибудет к получателю прежде, чем отправитель обнаружит, что произошла ошибка. Когда поврежденный кадр приходит получателю, он, конечно, должен быть отвергнут, однако что должен делать получатель со всеми правильными последующими кадрами? Как уже говорилось, получающий уровень передачи данных обязан передавать пакеты сетевому уровню, соблюдая строгий порядок. На рис. 3.11 изображен эффект, оказываемый конвейерной обработкой на исправление ошибок. Сейчас мы рассмотрим этот вопрос более подробно.

Есть два способа решения проблемы возникновения ошибок при конвейеризации кадров. Первый способ называется **возвратом на *n*** и заключается просто в игнорировании всех кадров, следующих за ошибочным. Для таких кадров подтверждения не посылаются. Эта стратегия соответствует приемному окну размера 1. Другими словами, уровень передачи данных отказывается принимать какой-либо кадр, кроме кадра со следующим номером, который он должен передать сетевому уровню. Если окно отправителя заполнится раньше, чем истечет время ожидания, конвейер начнет простаивать. Наконец, лимит времени у отправителя истечет, и он начнет передавать повторно сразу все кадры, не получившие подтверждения, начиная с поврежденного или потерянного кадра. Такой подход при высоком уровне ошибок может привести к потере большой доли пропускной способности канала.

На рис. 3.11, *а* изображен возврат на *n* для случая большого окна приемника. Кадры 0 и 1 корректно принимаются, и высылается подтверждение этого факта. Однако кадр 2 потерялся или был испорчен. Ничего не подозревающий отправитель продолжает посылать кадры, пока не выйдет время ожидания кадра 2. Только после этого он возвращается к месту сбоя и заново передает все кадры, начиная с кадра 2 (отправляя 2, 3, 4 и т. д.)

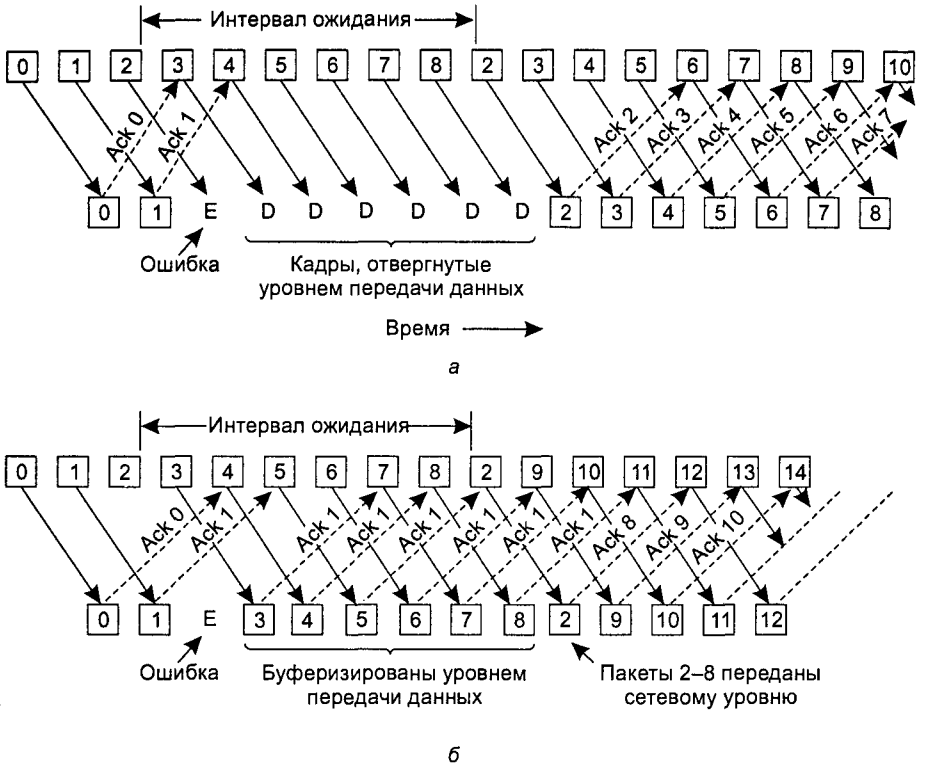


Рис. 3.11. Конвейеризация и коррекция ошибок: эффект при размере окна 1 (*а*); эффект при размере окна > 1 (*б*)

Другая общая стратегия обработки ошибок при конвейерной передаче кадров, называемая **выборочным повтором**, заключается в том, что получатель хранит в буфере все правильные кадры, принятые им после неверного или потерянного кадра. При этом неверный кадр отбрасывается. Когда отправитель наконец замечает, что что-то случилось (то есть заканчивается время ожидания подтверждения), он просто отправляет еще раз только потерянный или испорченный кадр, не повторяя отправку всех последующих, как это продемонстрировано на рис. 3.11, *б*. Если вторая попытка будет успешной, получающий уровень передачи данных сможет быстро передать накопившиеся пакеты сетевому уровню, после чего выслать подтверждение получения кадра с наибольшим номером. Выборочный повтор часто комбинируется с отправкой получателем «отрицательного

подтверждения» (NAK — Negative Acknowledgement) при обнаружении ошибки, например, при неверной контрольной сумме или при измененном порядке следования кадров. NAK стимулируют повторную отправку еще до того, как закончится время ожидания подтверждения от отправителя. Таким образом, эффективность работы несколько повышается.

На рис. 3.11, *б* кадры 0 и 1 снова принимаются корректно, а кадр 2 теряется. После получения кадра 3 уровень передачи данных приемника замечает, что один кадр выпал из последовательности. Для кадра 2 отправителю посылается NAK, однако кадр 3 сохраняется в специальном буфере. Далее прибывают кадры 4 и 5, они также буферизируются уровнем передачи данных вместо передачи на сетевой уровень. NAK 2 приходит к отправителю, заставляя его переслать кадр 2. Когда последний оказывается у получателя, у уровня передачи данных уже имеются кадры 2, 3, 4 и 5, которые сразу же в нужном порядке отдаются сетевому уровню. Теперь уже можно выслать подтверждение получения всех кадров, включая пятый, что и показано на рисунке. Если NAK вдруг потеряется, то отправитель по окончании времени ожидания 2 сам отправит кадр 2 (и только его!), однако это может произойти значительно позже, чем при помощи NAK. Короче говоря, NAK ускоряет процесс повторной пересылки испорченного кадра.

Метод выборочного повтора соответствует приемному окну размером более 1. Любой кадр в пределах окна может быть принят, после чего он будет храниться в буфере до тех пор, пока все предыдущие кадры не будут переданы сетевому уровню. При большом окне данный способ может потребовать значительного размера буфера.

Листинг 3.6. Протокол скользящего окна с возвратом на *n*

```
/* Протокол 5 (конвейерный) допускает наличие нескольких неподтвержденных кадров.
Отправитель может передать до MAX_SEQ кадров, не ожидая подтверждения. Кроме того, в
отличие от предыдущих протоколов, не предполагается, что у сетевого уровня всегда есть
новые пакеты. При появлении нового пакета сетевой уровень инициирует событие network_
layer_ready */

#define MAX_SEQ 7                          /* должно быть 2^n-1 */
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* возвращает true, если (a <=b < c циклично; иначе false */
  if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
      return(true);
  else
      return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[])
{
/* подготовить и послать информационный кадр */
  frame s;                                 /* временная переменная */
```

```
  s.info = buffer[frame_nr];                  /* вставить пакет в кадр */
  s.seq = frame_nr;                           /* вставить порядковый номер в кадр */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);  /* подтверждение, посылаемое
«верхом» на кадре данных */
  to_physical_layer(&s);                      /* послать кадр по каналу */
  start_timer(frame_nr);                      /* запустить таймер ожидания подтверждения */
}

void protocol5(void)
{
  seq_nr next_frame_to_send;       /* MAX_SEQ > 1; используется для исходящего потока
*/
  seq_nr ack_expected;             /* самый старый неподтвержденный кадр */
  seq_nr frame_expected;           /* следующий кадр, ожидаемый во входящем потоке */
  frame r;                         /* временная переменная */
  packet buffer[MAX_SEQ+1];        /* буферы для исходящего потока */
  seq_nr nbuffered;     /* количество использующихся в данный момент выходных буферов
*/
  seq_nr i;                        /* индекс массива буферов */
  event_type event;

  enable_network_layer();      /* разрешить события network_layer_ready */
  ack_expected = 0;            /* номер следующего ожидаемого входящего подтверждения
*/
  next_frame_to_send = 0;      /* номер следующего посылаемого кадра */
  frame_expected = 0;          /* номер следующего ожидаемого входящего кадра */
  nbuffered = 0;               /* вначале буфер пуст */

  while (true) {
      wait_for_event(&event);  /* четыре возможных события: см. event_type выше */

      switch(event) {
          case network_layer_ready:  /* у сетевого уровня есть пакет для передачи */
              /* получить, сохранить и передать новый кадр */
              from_network_layer(&buffer[next_frame_to_send]); /* получить новый пакет
у сетевого уровня */
              nbuffered = nbuffered + 1;  /* увеличить окно отправителя */
              send_data(next_frame_to_send, frame_expected, buffer);  /* передать кадр
*/
              inc(next_frame_to_send);  /* увеличить верхний край окна отправителя */
              break;

          case frame_arrival:           /* прибыл кадр с данными или с подтверждением
*/
              from_physical_layer(&r);  /* получить пришедший кадр у физического
уровня */

              if (r.seq == frame_expected) {
                  /* кадры принимаются только по порядку номеров */
                  to_network_layer(&r.info);  /* передать пакет сетевому уровню */
                  inc(frame_expected);     /* передвинуть нижний край окна получателя
*/
              }

              /* подтверждение для кадра n подразумевает также кадры n - 1, n - 2
и т. д. */
```

```
            while (between(ack_expected, r.ack, next_frame_to_send)) {
                /* отправить подтверждение вместе с информационным кадром */
                nbuffered = nbuffered - 1;  /* в буфере на один кадр меньше */
                stop_timer(ack_expected);   /* кадр прибыл в целости; остановить
таймер */
                inc(ack_expected);          /* уменьшить окно отправителя */
            }
            break;

        case cksum_err: ;              /* плохие кадры просто игнорируются */
            break;

        case timeout:  /* время истекло; передать повторно все неподтвержденные
кадры */
            next_frame_to_send = ack_expected;  /* номер первого посылаемого
повторно кадра */
            for (i = 1; i <= nbuffered; i++) {
                send_data(next_frame_to_send, frame_expected, buffer);  /* переслать
повторно 1 кадр */
                inc(next_frame_to_send);  /* приготовиться к пересылке следующего
кадра */
            }
      }

      if (nbuffered < MAX_SEQ)
          enable_network_layer();
      else
          disable_network_layer();
  }
}
```

Выбор одной из двух приведенных ранее стратегий является вопросом компромисса между пропускной способностью и размером буфера уровня передачи данных. В зависимости от того, что в конкретной ситуации является более критичным, может использоваться тот или иной метод. В листинге 3.6 показан конвейерный протокол, в котором принимающий уровень передачи данных принимает кадры по порядку. Все кадры, следующие за ошибочным, игнорируются. В данном протоколе мы впервые отказались от предположения, что у сетевого уровня всегда есть неограниченное количество пакетов для отсылки. Когда у сетевого уровня появляется пакет, который он хочет отправить, уровень инициирует событие `network_layer_ready`. Однако чтобы ограничить количество неподтвержденных пакетов числом *MAX_SEQ*, уровень передачи данных должен иметь возможность отключать на время сетевой уровень. Для этой цели служит пара библиотечных процедур — `enable_network_layer` и `disable_network_layer`.

Обратите внимание на то, что в любой момент времени может быть *MAX_SEQ*, а не *MAX_SEQ* + 1 неподтвержденных пакетов, хотя различаются *MAX_SEQ* + 1 порядковых номеров: от 0 до *MAX_SEQ*. Чтобы понять, почему необходимо такое ограничение, рассмотрим сценарий с *MAX_SEQ* = 7.

1. Отправитель посылает кадры с 0 по 7.
2. Подтверждение для кадра 7 наконец прибывает к отправителю.

3. Отправитель посылает следующие восемь кадров, снова с номерами с 0 по 7.
4. Еще одно подтверждение для кадра 7 прибывает к отправителю.

Но вот вопрос: все восемь кадров, входящих во второй набор, благополучно дошли до адресата или все они потерялись (включая игнорированные кадры после ошибочного)? В обоих случаях получатель отправит кадр 7 в качестве подтверждения. У отправителя нет способа отличить один случай от другого. По этой причине максимальное количество неподтвержденных кадров должно быть ограничено числом *MAX_SEQ*.

Хотя в протоколе 5 кадры, поступившие после ошибки, не буферизируются получателем, тем не менее, отправитель должен хранить отправленные кадры в своем буфере, пока не получит на них подтверждение. Если поступает подтверждение на кадр n, кадры $n - 1$, $n - 2$ (то есть все предыдущие кадры) автоматически считаются подтвержденными. Эта особенность наиболее важна в случае потери или повреждения какого-либо кадра с подтверждением. Получив подтверждение, уровень передачи данных проверяет, не освободился ли у него какой-нибудь буфер. Если буфер освобождается, то заблокированному ранее сетевому уровню можно снова разрешить инициировать события `network_layer_ready`.

Для этого протокола предполагается, что всегда есть обратный трафик, по которому можно отправлять подтверждение. Если же это условие не выполняется, то никакие подтверждения отосланы не будут. Протокол 4 не нуждается в подобном допущении, поскольку за каждым принятым кадром следует обратный, даже если только что уже был отправлен какой-то кадр в сторону отправителя. В следующем протоколе проблема отсутствия обратного трафика будет решена гораздо элегантней.

Поскольку протокол 5 хранит несколько неподтвержденных кадров, ему требуется несколько таймеров, по одному на кадр. Для каждого кадра время считается независимо от других. Все таймеры могут симулироваться программно, используя единственные аппаратные часы, вызывающие периодические прерывания. Данные таймеров могут храниться в программе в виде связанного списка, каждый узел которого будет хранить количество временных интервалов системных часов, оставшихся до истечения срока ожидания, номер кадра и указатель на следующий узел списка.

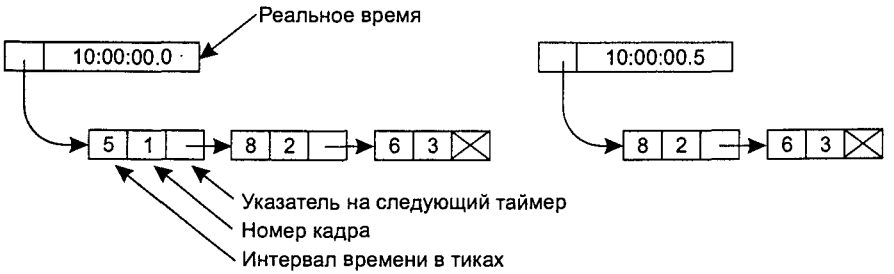


Рис. 3.12. Программная симуляция работы нескольких таймеров

В качестве иллюстрации приводится рис. 3.12, на котором показана программная реализация нескольких таймеров. Предположим, что часы изменяют свое состояние каждые 100 мс. Пусть начальное значение реального времени будет 10:00:00.0, и имеются три таймера тайм-аутов, установленные на время 10:00:00.5, 10:00:01.3 и 10:00:01.9. Каждый раз, когда аппаратные часы изменяют свое значение, реальное время обновляется и счетчик этих изменений в голове списка уменьшается на единицу. Когда значение счетчика становится равным нулю, инициируется тайм-аут, а узел удаляется из списка, как показано на рис. 3.12, *б*. Такая организация таймеров не требует выполнения большой работы при каждом прерывании от системных часов, хотя при работе процедур `start_timer` и `stop_timer` требуется сканирование списка. В протоколе у данных процедур имеется входной параметр, означающий номер кадра, таймер которого нужно запустить или остановить.

Протокол с выборочным повтором

Протокол 5 хорошо работает, если ошибки встречаются нечасто, однако при плохой линии он тратит впустую много времени, передавая кадры по два раза. В качестве альтернативы можно использовать протокол, в котором получатель буферизирует кадры, принятые после потерянного или испорченного кадра. Такой протокол не отвергает кадры только лишь потому, что предыдущий кадр был поврежден или потерян.

В этом протоколе и отправитель, и получатель работают с окнами допустимых номеров кадров. Размер окна отправителя начинается с нуля и растет до некоторого определенного уровня, *MAX_SEQ*. Размер окна получателя, напротив, всегда фиксированного размера, равного *MAX_SEQ*. Получатель должен иметь буфер для каждого кадра, номер которого находится в пределах окна. С каждым буфером связан бит, показывающий, занят буфер или свободен. Когда прибывает кадр, функция `between` проверяет, попал ли его порядковый номер в окно. Если да, то кадр принимается и хранится в буфере. Это действие производится независимо от того, является ли данный кадр следующим кадром, ожидаемым сетевым уровнем. Он должен храниться на уровне передачи данных до тех пор, пока все предыдущие кадры не будут переданы сетевому уровню в правильном порядке. Алгоритм протокола показан в листинге 3.7.

Листинг 3.7. Протокол скользящего окна с выборочным повтором

```
/* Протокол 6 (выборочный повтор) принимает кадры в любом порядке, но передает их
сетевому уровню, соблюдая порядок. С каждым неподтвержденным кадром связан таймер. При
срабатывании таймера передается повторно только этот кадр, а не все неподтвержденные
кадры, как в протоколе 5. */

#define MAX_SEQ 7                          /* должно быть 2^n-1 */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout}
event_type;
#include "protocol.h"
```

```
boolean no_nak = true;          /* отрицательное подтверждение (nak) еще не посылалось
*/
seq_nr oldest_frame = MAX_SEQ+1;          /* начальное значение для симулятора */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* то же, что и в протоколе 5, но короче и понятнее */
  return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet
buffer[])
{
/* сформировать и послать данные, а также положительное или отрицательное
подтверждение */
  frame s;                                  /* временная переменная */

  s.kind = fk;                              /* kind == data, ack, или nak */
  if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
  s.seq = frame_nr;                 /* имеет значение только для информационных кадров
*/
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
  if (fk == nak) no_nak = false;    /* один nak на кадр, пожалуйста */
  to_physical_layer(&s);            /* переслать кадр */
  if (fk == data) start_timer(frame_nr % NR_BUFS);
  stop_ack_timer();                 /* отдельный кадр с подтверждением не нужен */
}

void protocol6(void)
{
  seq_nr ack_expected;                      /* нижний край окна отправителя */
  seq_nr next_frame_to_send;                /* верхний край окна отправителя + 1 */
  seq_nr frame_expected;                    /* нижний край окна получателя */
  seq_nr too_far;                           /* верхний край окна получателя + 1 */
  int i;                                    /* индекс массива буферов */
  frame r;                                  /* временная переменная */
  packet out_buf[NR_BUFS];                  /* буферы для исходящего потока */
  packet in_buf[NR_BUFS];                   /* буферы для входящего потока */
  boolean arrived[NR_BUFS];                 /* входящая битовая карта */
  seq_nr nbuffered;     /* количество использующихся в данный момент выходных буферов
*/
  event_type event;

  enable_network_layer();           /* инициализация */
  ack_expected = 0;                 /* следующий номер подтверждения во входном потоке
*/
  next_frame_to_send = 0;           /* номер следующего посылаемого кадра */
  frame_expected = 0;               /* номер следующего ожидаемого входящего кадра */
  too_far = NR_BUFS;                /* верхний край окна получателя + 1 */
  nbuffered = 0;                    /* вначале буфер пуст */

  for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
  while (true) {
```

```
     wait_for_event(&event);          /* пять возможных событий: см. event_type выше
*/
     switch(event) {
         case network_layer_ready:      /* получить, сохранить и передать новый кадр
*/
             nbuffered = nbuffered + 1;  /* увеличить окно отправителя */
             from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /* получить
новый пакет у сетевого уровня */
             send_frame(data, next_frame_to_send, frame_expected, out_buf);  /*
передать кадр */
             inc(next_frame_to_send);  /* увеличить верхний край окна отправителя */
             break;

         case frame_arrival:  /* прибыл кадр с данными или с подтверждением */
             from_physical_layer(&r);  /* получить пришедший кадр у физического
уровня */
             if (r.kind == data) {
                 /* кадр прибыл в целости */
                 if ((r.seq != frame_expected) && no_nak)
                     send_frame(nak, 0, frame_expected, out_buf); else start_ack_
timer();
                 if (between(frame_expected, r.seq, too_far) && (arrived[r.seq%NR_
BUFS] == false)) {
                     /* кадры могут приниматься в любом порядке */
                     arrived[r.seq % NR_BUFS] = true;  /* пометить буфер как занятый
*/
                     in_buf[r.seq % NR_BUFS] = r.info;  /* поместить данные в буфер
*/
                     while (arrived[frame_expected % NR_BUFS]) {
                         /* передать пакет сетевому уровню и сдвинуть окно */
                         to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                         no_nak = true;
                         arrived[frame_expected % NR_BUFS] = false;
                         inc(frame_expected);  /* передвинуть нижний край окна
получателя */
                         inc(too_far);  /* передвинуть верхний край окна получателя
*/
                         start_ack_timer();  /* запустить вспомогательный таймер на
случай, если потребуется пересылка подтверждения отдельным кадром */
                     }
                 }
             }
             if((r.kind==nak) && between(ack_expected,(r.ack+1)%(MAX_SEQ+1),next_
frame_to_send))
                 send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected, out_
buf);

             while (between(ack_expected, r.ack, next_frame_to_send)) {
                 nbuffered = nbuffered - 1;  /* отправить подтверждение вместе с
информационным кадром */
                 stop_timer(ack_expected % NR_BUFS);  /* кадр прибыл в целости */
                 inc(ack_expected);  /* передвинуть нижний край окна отправителя */
             }
             break;
```

```
          case cksum_err: if (no_nak) send_frame(nak, 0, frame_expected, out_buf);
break;  /* поврежденный кадр */
          case timeout: send_frame(data, oldest_frame, frame_expected, out_buf);
break;  /* время истекло */
          case ack_timeout: send_frame(ack,0,frame_expected, out_buf);  /* истек
период ожидания «попутки» для подтверждения: послать подтверждение */
      }

      if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
  }
}
```

Способность протокола принимать кадры в произвольном порядке вызывает некоторые проблемы, отсутствовавшие в предыдущих протоколах, в которых все пакеты принимались строго по порядку номеров. Проще всего проиллюстрировать это на примере. Предположим, что порядковый номер кадра состоит из 3 бит, так что отправитель может посылать до семи кадров, прежде чем перейти в режим ожидания подтверждения. Начальное состояние окон отправителя и получателя изображено на рис. 3.13, *а*. Отправитель передает кадры с 0 по 6. Окно получателя позволяет ему принимать любые кадры с номерами от 0 по 6 включительно. Все семь кадров прибывают успешно, поэтому получатель подтверждает их прием и передвигает окно для приема кадров с номерами 7, 0, 1, 2, 3, 4 и 5, как показано на рис. 3.12, *б*. Все семь буферов помечаются как свободные.

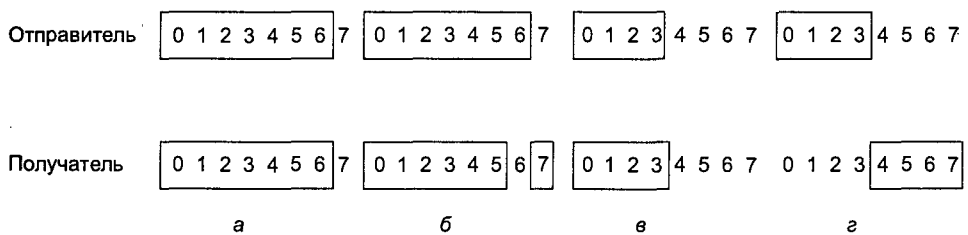


Рис. 3.13. Начальная ситуация при размере окна 7 (*а*); 7 кадров были посланы и приняты, но не подтверждены (*б*); начальная ситуация при размере окна 4 (*в*); ситуация после того, как 4 кадра были отправлены и получены, но не подтверждены (*г*)

Именно в этот момент происходит какое-нибудь бедствие — например, молния ударяет в телефонный столб и стирает все подтверждения. Отправитель, не дождавшись подтверждений, посылает повторно кадр 0. К сожалению, кадр 0 попадает в новое окно и поэтому принимается получателем (рис. 3.13, *б*). Получатель снова отправляет подтверждение для кадра 6, поскольку были приняты все кадры с 0 по 6.

Отправитель с радостью узнает, что все переданные им кадры успешно достигли адресата, поэтому он тут же передает кадры 7, 0, 1, 2, 3, 4 и 5. Кадр 7 принимается получателем, и содержащийся в нем пакет передается сетевому уровню. Сразу после этого принимающий уровень передачи данных проверяет наличие кадра 0, обнаруживает его и передает внедренный пакет сетевому уровню. Таким образом, сетевой уровень получает неверный пакет; это означает, что протокол со своей задачей не справился.

Причина неудачи в том, что при сдвиге приемного окна новый интервал допустимых номеров кадров перекрыл старый интервал. Соответственно, присылаемый набор кадров может содержать как новые кадры (если все подтверждения были получены), так и повторно высланные старые кадры (если подтверждения были потеряны). У принимающей стороны нет возможности отличить одну ситуацию от другой.

Решением данной проблемы является предоставление гарантии того, что в сдвинутом положении окно не перекроет исходное окно. Для этого размер окна не должен превышать половины от количества порядковых номеров (это показано на рис. 3.13, *в* и *г*). Например, если для порядковых номеров используются 4 бита, они должны изменяться в пределах от 0 до 15. В таком случае в любой момент времени только восемь кадров могут быть неподтвержденными. Таким образом, если будут получены кадры с 0 по 7 и будет передвинуто окно для приема кадров с 8 по 15, получатель сможет безошибочно отличить повторную передачу (кадры с 0 по 7) от новых кадров (с 8 по 15). Поэтому в протоколе 6 применяется окно размером (*MAX_SEQ* + 1)/2.

Возникает новый вопрос: сколько буферов должно быть у получателя? Ни при каких условиях он не должен принимать кадры, номера которых не попадают в окно. Поэтому количество необходимых буферов равно размеру окна, а не диапазону порядковых номеров. В приведенном выше примере для 4-битовых порядковых номеров требуется восемь буферов с номерами от 0 до 7. Когда прибывает кадр i, он помещается в буфер i mod 8. Обратите внимание на то, что хотя i и (i + 8), взятые по модулю 8, «соревнуются» за один и тот же буфер, они никогда не оказываются в одном окне одновременно, потому что это привело бы к увеличению размера окна по крайней мере до 9.

По этой же причине количество необходимых таймеров также равно числу буферов, а не диапазону порядковых номеров. Таким образом, удобно связать каждый таймер со своим буфером. Когда интервал времени истекает, содержимое буфера высылается повторно.

В протоколе 5 предполагалось, что загрузка канала довольно высока. Когда прибывал кадр, он не подтверждался сразу. Вместо этого подтверждение «ехало верхом» на встречном информационном кадре. Если обратный поток информации был невелик, подтверждения задерживались на довольно большой период времени. Если в одном направлении посылалось много информации, а во встречном — вообще ничего, то высылалось только *MAX_SEQ* пакетов, после чего протокол останавливался.

В протоколе 6 эта проблема решена. По прибытии кадра с данными процедура `start_ack_timer` запускает вспомогательный таймер. Если таймер сработает раньше, чем появится кадр с данными для передачи, то будет послан отдельный кадр с подтверждением. Прерывание от вспомогательного таймера называется событием `ack_timeout`. При такой организации возможен однонаправленный поток данных, так как отсутствие встречных информационных кадров, на которых можно было бы отправлять подтверждения, больше не является препятствием. Для этого требуется всего один таймер. При вызове процедуры *start_ack_timer*,

если таймер уже был запущен, его параметры переустанавливаются на отсчет полного интервала времени.

Важно, что период времени вспомогательного таймера должен быть существенно короче интервала ожидания подтверждения. При этом условии подтверждение в ответ на полученный правильный кадр должно приходить прежде, чем у отправителя истечет период ожидания и он пошлет этот кадр второй раз.

Протокол 6 использует более эффективную стратегию обработки ошибок, чем протокол 5. Как только у получателя появляются подозрения, что произошла ошибка, он высылает отправителю отрицательное подтверждение (NAK). Получатель может делать это в двух случаях: если он получил поврежденный кадр или если прибыл кадр с номером, отличным от ожидаемого (возможность потери кадра). Чтобы избежать передачи нескольких запросов на повторную передачу одного и того же кадра, получатель должен запоминать, был ли уже послан NAK для данного кадра. В протоколе 6 для этой цели применяется переменная `no_nak`, принимающая значение `true`, если NAK для ожидаемого кадра (с номером `frame_expected`) еще не был послан. Если NAK повреждается или теряется по дороге, в этом нет большой беды, так как у отправителя в конце концов истечет период ожидания положительного подтверждения и он, так или иначе, вышлет недостающий кадр еще раз. Если после того как NAK будет выслан и потерян прибудет не тот кадр, переменной `no_nak` опять будет присвоено `true` и будет запущен вспомогательный таймер. Когда время истечет, будет послано положительное подтверждение (ACK) для восстановления синхронизации текущих состояний отправителя и получателя.

В некоторых ситуациях время, необходимое для прохождения кадра по каналу, его обработки и отсылки обратно подтверждения, остается практически неизменным. В таких ситуациях отправитель может поставить время ожидания несколько больше ожидаемого интервала между отправкой кадра и получением подтверждения. Однако если это время меняется в широких пределах, перед отправителем возникает непростой выбор значения времени ожидания. Если выбрать слишком короткий интервал, то увеличится риск ненужных повторных передач, следовательно, снизится производительность канала. При выборе слишком большого значения протокол будет тратить много времени на ожидания, что также снизит пропускную способность.

В любом случае пропускная способность на что-то тратится. Если встречный поток данных нерегулярен, то время прихода подтверждений также будет непостоянным, уменьшаясь при наличии встречного потока и увеличиваясь при его отсутствии. Переменное время обработки кадров получателем также может вызвать ряд проблем. Итак, если среднеквадратичное отклонение интервала ожидания подтверждения невелико по сравнению с самим интервалом, то таймер может быть установлен довольно «туго» и польза от отрицательных подтверждений (NAK) не очень высока. В противном случае таймер может быть установлен довольно «свободно», и отрицательные подтверждения могут существенно ускорить повторную передачу потерянных или поврежденных кадров.

С вопросом тайм-аутов и отрицательных подтверждений тесно связана проблема определения кадра, вызвавшего тайм-аут. В протоколе 5 это всегда кадр

с номером ack_expected, поскольку он является старшим. В протоколе 6 нет столь простого способа определить кадр, интервал ожидания которого истек. Предположим, были переданы кадры с 0 по 4, то есть список неподтвержденных кадров выглядит так: 01234 (от первого к последнему). Теперь допустим, что у кадра 0 истекает интервал ожидания и он передается повторно, затем посылается кадр 5 (новый), потом интервал ожидания истекает у кадров 1 и 2, и посылается кадр 6 (также новый). В результате список неподтвержденных кадров принимает вид 3405126, начиная с самого старого и заканчивая самым новым. Если весь встречный поток данных потеряется, интервалы ожидания этих семи кадров истекут именно в таком порядке. Чтобы не усложнять и без того непростой пример протокола, мы не показываем подробностей управления таймером. Вместо этого предполагается, что переменной oldest_frame при наступлении тайм-аута присваивается номер кадра, интервал времени которого истек.

Верификация протоколов

Реальные протоколы и реализующие их программы обычно весьма сложны. Поэтому множество исследователей посвятили свои труды попыткам применить формальные математические методы для задания и проверки (верификации) протоколов. В следующих разделах мы обсудим некоторые модели и методы. Хотя мы будем рассматривать их в контексте уровня передачи данных, они применимы и к другим уровням.

Модели конечных автоматов

Ключевой концепцией, используемой в моделях протоколов, является модель конечных автоматов. Данный метод рассматривает состояния, в которых находится протокольная машина (то есть отправитель или получатель) в каждый момент времени. Состояние протокольной машины включает в себя значения всех ее переменных, в том числе и программные счетчики.

В большинстве случаев многие состояния можно объединить для анализа в группы. Например, рассматривая получателя в протоколе 3, мы можем выделить из всех его возможных состояний два самых важных: ожидание кадра 0 и ожидание кадра 1. Все остальные состояния могут рассматриваться как временные, промежуточные между этими двумя. Обычно выбираются состояния, соответствующие ожидаемым событиям (это соответствует вызову процедуры wait(event) в наших примерах протоколов). Состояние протокольной машины полностью определяется состоянием ее переменных. Количество состояний равно 2^n, где n — количество битов, необходимых для описания всех переменных, вместе взятых.

Состояние всей системы представляет собой комбинацию всех состояний двух протокольных машин и состояний канала. Состояние канала определяется его содержимым. При использовании протокола 3 у канала может быть четыре возможных состояния: кадр 0 или кадр 1, перемещающийся от отправителя к получателю, кадр подтверждения, двигающийся в противоположном направлении, или

пустой канал. Если отправитель и получатель могут находиться в двух различных состояниях, то вся система будет иметь 16 различных состояний.

Следует сказать несколько слов о состоянии канала. Концепция кадра, находящегося в канале, конечно, является абстракцией. Имеется в виду, что кадр частично передан, частично принят, но еще не обработан получателем. Кадр остается «в канале» до тех пор, пока протокольная машина не выполнит процедуру `from_physical_layer` и не обработает его.

Каждое состояние может быть соединено с другими состояниями несколькими (0 или больше) **переходами**. Переходы всегда соответствуют каким-либо событиям. Для протокольной машины переход происходит, когда прибывает новый кадр, истекает время ожидания какого-либо таймера, происходит прерывание и т. д. Для канала обычными событиями являются следующие: помещение протокольной машиной в канал нового кадра, доставка кадра протокольной машине или потеря кадра вследствие всплеска шума. Имея полное описание протокольной машины и характеристик канала, можно нарисовать направленный граф, изображающий все состояния в виде узлов, а переходы — в виде направленных дуг.

Одно особое состояние обозначается как **начальное**. Оно соответствует состоянию системы в момент ее запуска или состоянию, близкому к нему. Из начального состояния можно попасть во многие (вероятно, даже во все) другие состояния с помощью серии переходов. Применяя хорошо известные методы теории графов (например, вычисляя транзитивное замыкание графа), можно определить, какие из состояний достижимы, а какие нет. Такой метод называется **анализом достижимости** (Lin и др., 1987). Данный анализ может быть полезен для определения правильности протокола.

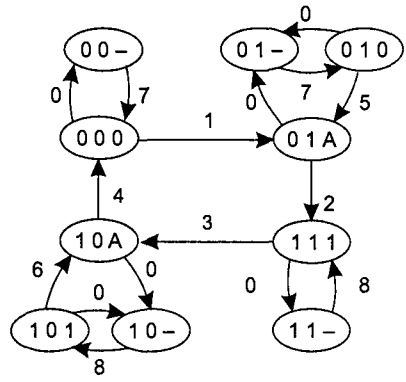
Формально модель протокола машины конечных состояний можно рассматривать как набор их четырех множеств (S, M, I, T), где S — множество состояний, в которых могут находиться процессы и канал; M — множество кадров, передающихся по каналу; I — множество начальных состояний процессов; T — множество переходов между состояниями.

В начальный момент времени все процессы находятся в исходном состоянии. Затем начинают происходить события — например, появляются кадры, которые нужно передать по каналу, или срабатывает таймер. Каждое событие может переключить один из процессов или канал в новое состояние. Тщательно нумеруя все возможные предшественники каждого состояния, можно построить дерево достижимости и проанализировать протокол.

Анализ достижимости может применяться для обнаружения разнообразных ошибок спецификации протокола. Например, если какой-либо кадр может оказаться в таком состоянии, в котором конечный автомат не сможет определить, какое действие следует предпринять, это означает, что спецификация ошибочна (неполнота протокола). Если имеется ряд состояний, из которых нет выхода (то есть никакие кадры больше не смогут передаваться), это также означает ошибку спецификации (тупик). Менее серьезной ошибкой спецификации протокола является описание обработки события в состоянии, в котором это событие не может произойти (лишний переход). Могут быть обнаружены и другие ошибки.

В качестве примера конечного автомата рассмотрим рис. 3.14, *а*. Этот граф соответствует протоколу 3, описанному ранее, — у каждой протокольной машины может быть два состояния, а у канала — четыре состояния. Итого возможны 16 состояний, некоторые из которых недоступны из начального состояния. Недостижимые состояния не показаны на рисунке. Для упрощения также опущены ситуации с ошибками контрольных сумм.

Каждое состояние помечено тремя символами: *SRC*, где *S* является 0 или 1 и соответствует номеру кадра, который пытается послать отправитель; *R* также может быть равно 0 или 1 и означает номер кадра, ожидаемого получателем, а *C* может принимать значения 0, 1, *A* или пусто (-), в соответствии с состоянием канала. В данном примере было выбрано исходное состояние (000). То есть отправитель только что послал кадр 0, получатель ожидает кадра 0, и кадр 0 находится в канале.



а

Переход	Кто управляет?	Кадр принят	Кадр передан	Сетевому уровню
0	–	(кадр потерян)		–
1	Получатель	0	Подтверждение	Да
2	Отправитель	Подтверждение	1	–
3	Получатель	1	Подтверждение	Да
4	Отправитель	Подтверждение	0	–
5	Получатель	0	Подтверждение	Нет
6	Получатель	1	Подтверждение	Нет
7	Отправитель	(тайм-аут)	0	–
8	Отправитель	(тайм-аут)	1	–

б

Рис. 3.14. Диаграмма состояний для протокола 3 (*а*); переходы (*б*)

На рис. 3.14 показаны девять типов переходов. Переход 0 заключается в потере каналом содержимого. Переход 1 состоит в том, что канал успешно доставляет пакет 0 получателю, при этом получатель передает пакет 0 своему се-

тевому уровню, изменяет свое состояние на ожидание кадра 1 и пересылает подтверждение. Остальные переходы приведены на рис. 3.14, *б*. Прибытие кадра с неверной контрольной суммой не показано, так как оно не изменяет состояния в протоколе 3.

При нормальной работе протокола переходы 1, 2, 3 и 4 повторяются друг за другом снова и снова. На каждом цикле доставляются два пакета, возвращая отправителя в исходное состояние, в котором он пытается переслать новый кадр с последовательным номером 0. Если канал теряет кадр 0, происходит переход из состояния (000) в состояние (00–). Наконец у отправителя срабатывает таймер (переход 7), и система возвращается в состояние (000). Потеря подтверждения является более сложной ситуацией, так как требует для восстановления два перехода — 7 и 5 (или 8 и 6 в симметричной ситуации).

Одно из свойств, которыми должен обладать протокол с 1-битовым порядковым номером, заключается в том, что ни при каких обстоятельствах получатель не должен передавать своему сетевому уровню два нечетных пакета подряд, не передав между ними четного пакета, и наоборот. Для изображенного на рис. 3.14 графа это требование может быть перефразировано более формально так: «не должно быть пути, начинающегося в исходном состоянии, при котором переход 1 осуществляется два раза подряд без перехода 3 между ними, и наоборот». Из рисунка видно, что данное требование удовлетворяется.

Другим сходным требованием является отсутствие пути, при котором отправитель изменяет свое состояние дважды (например, с 0 на 1 и обратно на 0) при неизменном состоянии получателя. Существование такого пути означало бы, что два кадра подряд были безвозвратно потеряны, причем так, что получатель этого не заметил. При этом в последовательности доставляемых сетевому уровню пакетов образовалась бы незамеченная брешь в два пакета.

Еще одним важным свойством протоколов является отсутствие тупиков. **Тупиком** называется ситуация, в которой протокол не может продвинуться дальше (то есть доставлять пакеты сетевому уровню) ни при каких обстоятельствах. В терминах графической модели наличие тупика характеризуется существованием подмножества состояний, достижимого из исходного состояния и обладающего двумя следующими свойствами:

1. Из этого подмножества состояний нет перехода.
2. В подмножестве нет переходов, обеспечивающих дальнейшую работу протокола.

Попав в тупиковую ситуацию, протокол остается в ней навсегда. По графу видно, что в протоколе 3 нет тупиков.

Сети Петри

Конечный автомат является не единственным методом формального описания протокола. В данном разделе будет описана другая методика — сеть Петри (Danthine, 1980). Сетевая модель Петри состоит из четырех основных элементов: позиций, переходов, дуг и маркеров (или фишек). **Позицией** называется состояние, в котором может находиться система или ее часть. На рис. 3.15 показана сеть

Петри, состоящая из двух позиций, *A* и *B*, изображенных в виде кружков. В данный момент система находится в состоянии *A*, отмеченном **маркером** (жирной точкой) в позиции *A*. **Переход** обозначается вертикальной или горизонтальной чертой. У каждого перехода может быть ноль или несколько **входящих дуг**, идущих от своих исходных позиций, а также ноль или несколько **исходящих дуг**, направляющихся к выходным позициям.

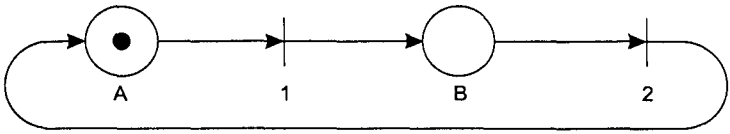


Рис. 3.15. Сеть Петри из двух позиций и двух переходов

Переход называется **разрешенным**, если имеется хотя бы один маркер на его входных позициях. Любой разрешенный переход может возбуждаться по желанию, удаляя маркер со всех исходных позиций и размещая их на всех выходных позициях. Количество маркеров на исходных и выходных позициях не обязано совпадать. Если разрешены два или более переходов, произойти может любой из них. Благодаря недетерминированности выбора перехода, который произойдет, сети Петри могут использоваться для моделирования протоколов. Сеть Петри, изображенная на рис. 3.15, детерминирована и может применяться для моделирования любого двухфазного процесса (например, поведения ребенка: поел, поспал, опять поел, опять поспал, и т. д.). Как и при любом моделировании, лишние детали игнорируются.

На рис. 3.16 показана сетевая модель Петри, соответствующая графу, описанному в листинге 3.4. В отличие от конечного автомата, здесь нет составных состояний. Состояния отправителя, получателя и канала изображаются отдельно. Переходы 1 и 2 соответствуют обычной и повторной (по тайм-ауту) передаче кадра 0. Переходы 3 и 4 означают то же самое, но для кадра 1. Переходы 5, 6 и 7 соответствуют потере кадра 0, подтверждения и кадра 1. Переходы 8 и 9 означают приход к получателю кадра с неверным номером. Переходы 10 и 11 обозначают получение принимающей машиной следующего кадра и передачу его сетевому уровню.

Сети Петри, так же как и конечные автоматы, могут применяться для обнаружения ошибок в протоколах. Например, если какая-нибудь последовательность переходов будет включать переход 10 дважды без перехода 11 между ними, это будет означать, что протокол неверен. Концепция тупиков в сети Петри также мало отличается от своего аналога в модели машины конечных состояний.

Сети Петри могут представляться в виде набора алгебраических формул, напоминающих грамматические правила. Каждому переходу соответствует одно правило грамматики. Каждое такое правило описывает входные и выходные позиции перехода. Поскольку на рис. 3.16 изображено 11 переходов, то и грамматика имеет 11 правил. Пронумеруем правила таким образом, чтобы каждое из них соответствовало переходу с тем же номером. Грамматика сети Петри, изображенной на рис. 3.16, представлена ниже:

1: BD → AC
2: A → A
3: AD → BE
4: B → B
5: C →
6: D →
7: E →
8: CF → DF
9: EG → DG
10: CG → DF
11: EF → DG

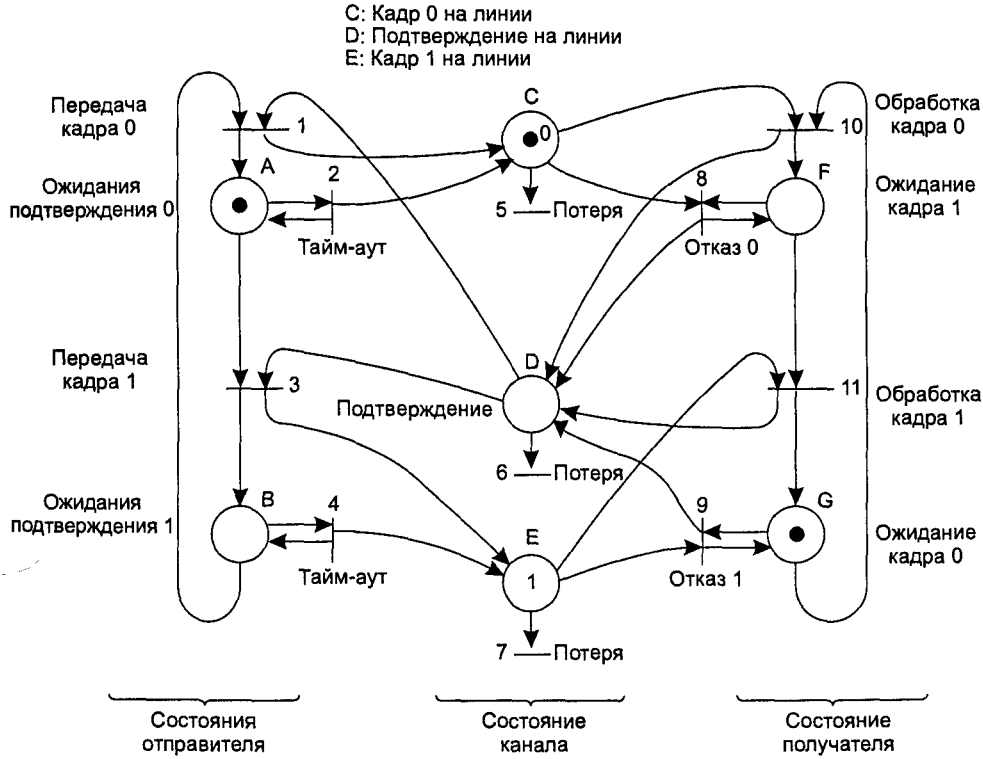


Рис. 3.16. Сетевая модель Петри для протокола 3

Интересно, что нам удалось компактно описать довольно сложный протокол набором из 11 элементарных правил грамматики, легко реализуемых компьютерной программой.

Текущее состояние сети Петри представляется неупорядоченным набором позиций, каждая из которых появляется в наборе столько раз, сколько фишек в ней имеется. Любое правило из грамматики, имеющее левую часть, может активироваться, удаляя свои левые позиции из текущего состояния сети и добавляя свои правые (выходные) позиции к текущему состоянию. Текущее состояние (маркировка) сети, изображенной на рис. 3.16, — *ACG*, поэтому, например, правило 10

(*CG* → *DF*) может быть применено, а правило 3 (*AD* → *BE*) — нет, потому что *D* не имеет маркера.

Примеры протоколов передачи данных

В следующих разделах мы рассмотрим некоторые широко используемые протоколы передачи данных. Первый из них, классический бит-ориентированный протокол HDLC, часто употреблялся во многих сетях. Второй, PPP, — это протокол уровня передачи данных, используемый при подключении к Интернету домашних компьютеров.

HDLC — высокоуровневый протокол управления каналом

В данном разделе мы рассмотрим группу тесно связанных друг с другом протоколов, немного устаревших, но все еще широко применяемых в сетях. Все они произошли от одного протокола передачи данных, применявшегося в разработанных компанией IBM мейнфреймах, — этот протокол называется **SDLC** (Synchronous Data Link Control — синхронное управление каналом). После разработки протокола SDLC корпорация IBM представила его на рассмотрение институтов ANSI и ISO для утверждения в качестве стандарта США и международного стандарта соответственно. ANSI модифицировал протокол в **ADCCP** (Advanced Data Communication Control Procedure — усовершенствованная процедура управления информационным обменом), а ISO переделала его в **HDLC** (High-level Data Link Control — высокоуровневый протокол управления каналом). После этого протокол был принят комитетом CCITT, который адаптировал HDLC для своего протокола доступа к каналу **LAP** (Link Access Procedure — процедура доступа к каналу), являющегося частью стандарта сетевого интерфейса X.25, однако затем снова изменил его на **LAPB**, повысив его совместимость с более поздней версией HDLC. Что хорошо в стандартах, так это то, что у вас всегда есть из чего выбрать. Если же вас не устраивает ни один из имеющихся стандартов, вы можете просто подождать появления новой модели в новом году.

В основе всех этих протоколов лежат одни и те же принципы. Все они являются бит-ориентированными, и во всех применяется битовое заполнение, обеспечивающее прозрачность данных. Они различаются только в незначительных, но, тем не менее, вызывающих раздражение деталях. Последующее обсуждение бит-ориентированных протоколов нужно рассматривать как общее введение. Специфические детали протоколов приводятся в соответствующих официальных описаниях.

Во всех бит-ориентированных протоколах используется формат кадра, показанный на рис. 3.17. Поле *Address* (адрес) чрезвычайно важно для линий с несколькими терминалами, где оно используется для идентификации одного из терминалов. В двухточечных сетях это поле иногда используется, чтобы отличать команды от ответов.

Поле *Control* (управляющей информации) используется для хранения порядковых номеров, подтверждений и других служебных данных, как будет показано далее.

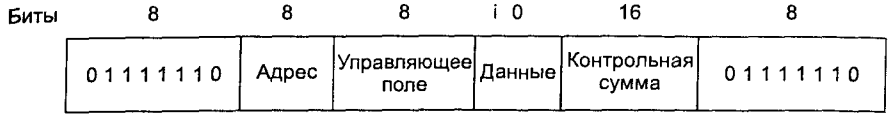


Рис. 3.17. Формат кадра бит-ориентированных протоколов

Поле *Data* (данные) может содержать произвольную информацию. Оно может быть любой длины, хотя эффективность контрольной суммы снижается с увеличением длины кадра из-за увеличения вероятности многочисленных пакетов ошибок.

Поле *Checksum* (контрольная сумма) является разновидностью циклического избыточного кода, который мы рассматривали в разделе «Коды с обнаружением ошибок».

В качестве заголовка и концевика кадра используется флаговый байт (01111110). В линиях «точка — точка», которые в текущий момент времени простаивают, флаговые последовательности передаются постоянно. Кадр минимального размера состоит из трех полей, занимающих в общей сложности 32 бита, не считая флаги в начале и в конце.

Все кадры можно разделить на три категории: **информационные**, **супервизорные** и **ненумерованные**. Содержимое поля *Control* для этих трех типов кадров показано на рис. 3.18. Протокол использует скользящее окно с 3-битовым порядковым номером. В каждый момент времени в сети может находиться не более семи неподтвержденных кадров. Поле *Seq* на рис. 3.18, *а* содержит порядковый номер кадра. Поле *Next* является пересылаемым вместе с кадром подтверждением. Однако все протоколы придерживаются соглашения о том, что вместо номера последнего правильно принятого кадра в поле *Next* пересылается номер первого не принятого кадра (то есть следующего ожидаемого кадра). Впрочем, номер кадра, используемого для подтверждения, не принципиален. Важно лишь, чтобы все участники придерживались одного и того же соглашения.

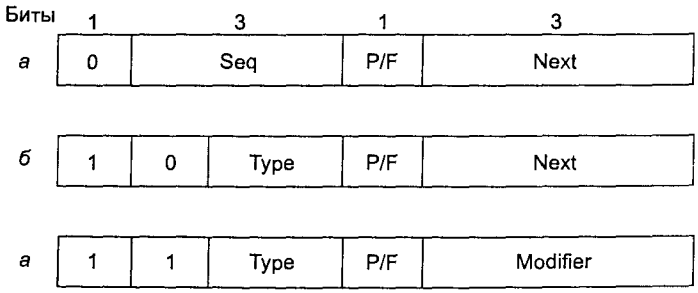


Рис. 3.18. Управляющее поле: информационного кадра (а); супервизорного кадра (*б*); ненумерованного кадра (*в*)

Бит *P/F* означает *Poll/Final* (Опрос/Финальный). Он используется, когда компьютер (или концентратор) опрашивает группу терминалов. В случае значения *P* компьютер предлагает терминалу посылать данные. Во всех кадрах, кроме последнего, посылаемых терминалом, бит *P/F* устанавливается в *P*. В последнем кадре этот бит устанавливается в *F*.

Некоторые протоколы используют бит *P/F*, чтобы заставить другую машину послать супервизорный кадр немедленно, не ожидая попутного потока данных. Этот бит также изредка используется в ненумерованных кадрах.

Тип супервизорного кадра указывается с помощью значения поля *Type*. Если *Type* = 0, значит, данный кадр является подтверждением. Он официально называется RECEIVE READY (к приему готов). Такой кадр сообщает номер следующего ожидаемого кадра и применяется при отсутствии попутного потока данных для передачи подтверждения.

Type = 1 является признаком отрицательного подтверждения, официально называющегося REJECT (отказ). Он применяется для сообщения об обнаружении ошибки передачи. Поле *Next* в этом случае содержит номер первого неверно полученного кадра (то есть первого кадра, который следует переслать повторно). Отправитель должен переслать повторно все неподтвержденные кадры, начиная с кадра с номером *Next*. Эта стратегия больше напоминает протокол 5, нежели протокол 6.

Type = 2 означает RECEIVE NOT READY (к приему не готов). При этом, как и в случае RECEIVE READY, подтверждается прием всех кадров вплоть до *Next*–1, однако отправителю сообщается, что передачу следует приостановить. Сигнал неготовности к приему предназначен не для использования в качестве альтернативы схеме скользящих окон, а для обозначения наличия у получателя каких-либо временных проблем, например отсутствия свободной памяти в буферах. Когда получатель сможет продолжить работу, он пошлет сигнал готовности, отказа или другой управляющий кадр.

Type = 3 означает SELECTIVE REJECT (выборочный отказ). Такой байт представляет собой запрос повторной передачи только указанных кадров. В этом он больше похож на протокол 6, чем на протокол 5, и поэтому наиболее полезен, когда размер окна отправителя не превышает половины количества используемых порядковых номеров. Таким образом, если получатель хочет сохранить в буфере несвоевременные кадры для последующего использования, он может запросить повторную передачу любого кадра с помощью SELECTIVE REJECT. Протоколы HDLC и ADCCP поддерживают этот тип кадра, а протоколы SDLC и LAPB — нет, то есть в этих протоколах нет команды выборочного отказа, а кадры типа 3 не используются.

Третий класс кадров составляют ненумерованные кадры. Иногда они применяются для служебных целей, но могут переносить и данные, когда требуется ненадежный, не требующий соединения сервис. В отличие от предыдущих двух классов, в которых различные бит-ориентированные протоколы были почти идентичными, в вопросе использования ненумерованных кадров они очень сильно различаются. Для обозначения типа кадра зарезервировано 5 бит, однако используются значительно меньше, чем 32 возможных комбинации.

Все протоколы поддерживают команду DISC (DISConnect — прервать связь), позволяющую предупредить, что машина скоро будет выключена (например, для профилактического обслуживания). Также имеется команда, позволяющая машине, только что вернувшейся в подключенный режим (on-line), заявить о своем присутствии и принудительно обнулить все порядковые номера. Эта команда называется SNRM (Set Normal Response Mode — установить нормальный режим ответа). К сожалению, этот «нормальный режим» является чем угодно, но не нормой. Это несбалансированный (то есть асимметричный) режим, при котором один конец линии является ведущим (master), а другой — ведомым (slave). Команда SNRM появилась еще в те времена, когда обмен данными означал общение примитивного терминала с компьютером, которое, конечно, было асимметричным. Чтобы лучше учитывать ситуацию равноправных партнеров, в протоколы HDLC и LAPB была добавлена команда SABM (Set Asynchronous Balanced Mode — установить асинхронный сбалансированный режим), которая инициализирует линию и объявляет равенство сторон. Кроме того, в этих протоколах имеются дополнительные команды SABME и SNRME, которые отличаются от SABM и SNRM только тем, что вводят расширенный формат кадров с 7-битовым порядковым номером вместо 3-битового.

Третьей командой, поддерживаемой всеми этими протоколами, является FRMR (FRaMe Reject — отклонить кадр), применяющаяся, когда приходит кадр с верной контрольной суммой, но недопустимой семантикой. Например, супервизорный кадр типа 3 в протоколе LAPB, кадр длиной менее 32 бит, недопустимый управляющий кадр или подтверждение кадра, находящегося вне пределов окна и т. д. Данные включают управляющее поле неправильного кадра, параметры окна и набор битов, указывающих тип ошибки.

Управляющие кадры могут быть повреждены или потеряны так же, как и информационные кадры, поэтому им также нужны подтверждения. Для этой цели предназначен специальный служебный кадр, называемый UA (Unnumbered Acknowledgement — ненумерованное подтверждение). Поскольку неподтвержденным может быть только один управляющий кадр, то не возникает вопроса о том, какой именно служебный кадр подтверждается.

Остальные управляющие кадры занимаются инициализацией, опросом и сообщением состояния. Есть также управляющий кадр, который может содержать произвольную информацию, UI (Unnumbered Information). Эта информация не передается на сетевой уровень, но получается и обрабатывается самим уровнем передачи данных.

Несмотря на широкое распространение, протокол HDLC имеет большое количество недостатков. Обсуждение ряда проблем, связанных с этим протоколом, см. в (Fiorini и др., 1994).

Уровень передачи данных в Интернете

Интернет состоит из отдельных машин (хостов и маршрутизаторов) и связывающей их коммуникационной инфраструктуры. В пределах одного здания для соединения широко применяются локальные сети, но на больших территориях ин-

фраструктура строится на основе выделенных линий, соединяющих отдельные машины по принципу «точка — точка». Локальные сети будут рассматриваться в главе 4, здесь же мы обсудим протоколы передачи данных, используемые для линий «точка — точка» в Интернете.

На практике соединение «точка — точка» используется прежде всего в двух ситуациях. Во-первых, у тысяч организаций есть по одной или по несколько локальных сетей, в каждой из которых есть несколько хостов (персональных компьютеров, рабочих станций пользователя, серверов и т. д.) наряду с маршрутизаторами (или функционально близких к ним мостов). Маршрутизаторы часто соединяются магистральной локальной сетью. Обычно вся связь с внешним миром осуществляется через один или два маршрутизатора, связанных выделенными линиями «точка—точка» с удаленными маршрутизаторами. Именно эти маршрутизаторы вместе с выделенными линиями образуют подсети, из которых состоит Интернет.

Еще одна важная роль, которую соединения «точка — точка» играют в Интернете, заключается в том, что они связывают миллионы индивидуальных пользователей с помощью модемов и телефонных линий. Обычно пользователь дозванивается со своего домашнего компьютера до **поставщика услуг Интернета** или, как его еще называют, **провайдера**, и работает как полноценный интернет-хост. Этот метод отличается от использования выделенной линии между персональным компьютером и маршрутизатором только лишь тем, что, когда пользователь заканчивает сеанс связи, соединение прерывается. Домашний персональный компьютер, звонящий поставщику услуг Интернета, изображен на рис. 3.19. Модем показан в данном случае как внешнее устройство, однако современные компьютеры могут быть укомплектованы и внутренними модемами.

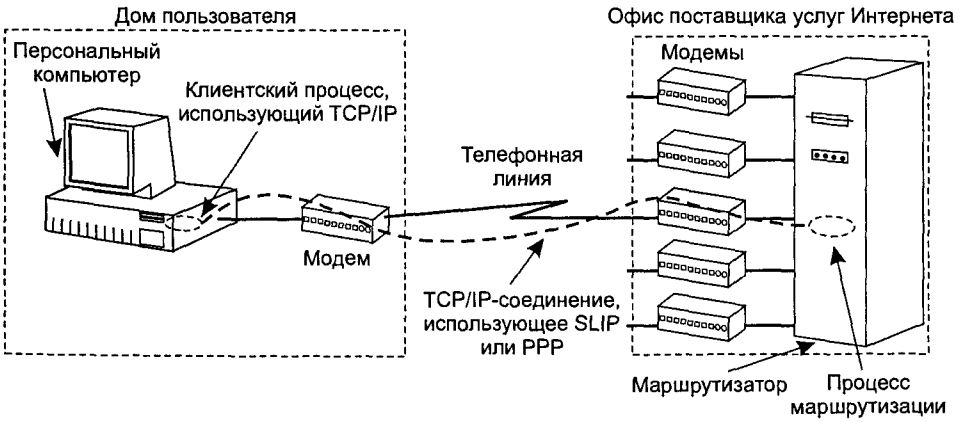


Рис. 3.19. Домашний персональный компьютер, действующий как хост Интернета

Как для соединения двух маршрутизаторов по выделенной линии, так и для соединения маршрутизатора с хостом требуется протокол, который бы занимался формированием кадров, обработкой ошибок и другими функциями уровня передачи данных, рассматривавшимися в данной главе. Одним из таких про-

токолов, широко распространенным в Интернете, является PPP. Рассмотрим его.

PPP — протокол двухточечного соединения

В Интернете двухточечные протоколы применяются очень часто в самых разных случаях, включая обеспечение соединения между маршрутизаторами, между пользователями и провайдерами. Обсуждаемый далее протокол называется **PPP** (Point-to-Point Protocol — протокол передачи от точки к точке), описан в RFC 1661 и доработан в некоторых более поздних документах RFC (например, RFC 1662 и 1663). Протокол PPP выполняет обнаружение ошибок, поддерживает несколько протоколов, позволяет динамически изменять IP-адреса во время соединения, разрешает аутентификацию, а также имеет ряд других свойств.

Протокол PPP обеспечивает следующий набор методов:

1. Метод формирования кадров, однозначно обозначающий конец одного кадра и начало следующего. Формат кадров также обеспечивает обнаружение ошибок.
2. Протокол управления каналом, позволяющий устанавливать каналы связи, тестировать их, договариваться о параметрах их использования и снова отключать их, когда они не нужны. Этот протокол называется **LCP** (Link Control Protocol). Он поддерживает синхронные и асинхронные линии, бит- и байт-ориентированное кодирование.
3. Способ договориться о параметрах сетевого уровня, который не зависит от используемого протокола сетевого уровня. Для каждого поддерживаемого сетевого уровня этот метод должен иметь свой сетевой протокол управления (**NCP**, Network Control Protocol).

Чтобы посмотреть, как все это работает вместе, рассмотрим типичный сценарий, когда домашний пользователь звонит поставщику услуг Интернета, чтобы превратить тем самым свой домашний компьютер во временный хост. Сначала персональный компьютер звонит через модем на маршрутизатор провайдера. После того, как модем маршрутизатора ответит на звонок и установит физическое соединение, персональный компьютер посылает маршрутизатору серию LCP-пакетов в поле данных пользователя одного или нескольких PPP-кадров. Эти пакеты и ответы на них определяют параметры протокола PPP.

После того как обе стороны согласовывают параметры, посылается серия NCP-пакетов для настройки сетевого уровня. Обычно персональный компьютер желает запустить стек протоколов TCP/IP, для чего ему нужен IP-адрес. На всех пользователей IP-адресов не хватает, поэтому обычно у каждого поставщика услуг Интернета имеется целый набор таких адресов, и он динамически назначает их каждому присоединившемуся персональному компьютеру на время сеанса связи. Если у провайдера есть n IP-адресов, он может одновременно подключить к Интернету до n машин, однако общее количество его клиентов может быть во много раз больше. Для назначения IP-адреса используется протокол NCP для IP.

После этого персональный компьютер фактически становится хостом Интернета и может посылать и принимать IP-пакеты так же, как и постоянные хосты.

Когда пользователь заканчивает сеанс связи, NCP используется, чтобы разорвать соединение сетевого уровня и освободить IP-адрес. Затем LCP используется для разрыва соединения уровня передачи данных. Наконец, компьютер дает модему команду повесить трубку, чем освобождает линию на физическом уровне.

Чтобы не изобретать велосипед, был выбран формат кадра PPP, близкий к формату кадра HDLC. В отличие от бит-ориентированного протокола HDLC, PPP является байт-ориентированным. В частности, в PPP применяется символьное заполнение на модемных телефонных линиях, поэтому все кадры состоят из целого числа байтов. С помощью протокола PPP невозможно послать кадр, состоящий из 30,25 байт, как это можно было сделать в протоколе HDLC. Кадры PPP могут посылаться не только по телефонным линиям, но и по сети SONET или по настоящим бит-ориентированным HDLC-линиям (например, по линиям, соединяющим маршрутизаторы). Формат кадра PPP показан на рис. 3.20.

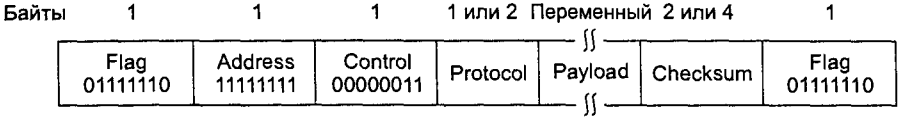


Рис. 3.20. Полный формат кадра PPP для работы в ненумерованном режиме

Все PPP-кадры начинаются со стандартного флагового байта протокола HDLC (01111110). Если такой байт встречается в поле данных, то применяется символьное заполнение. Следом за ним идет поле *Address* (адрес), которому всегда присваивается двоичное значение 11111111, что означает, что все станции должны принимать этот кадр. Использование такого адреса позволяет избежать необходимости назначения адресов передачи данных.

За полем адреса следует поле *Control*, его значение по умолчанию равно 00000011. Это число означает ненумерованный кадр. Другими словами, PPP по умолчанию не обеспечивает надежной передачи с использованием порядковых номеров и подтверждений. В зашумленных каналах, например при беспроводной связи, может применяться надежная передача с порядковыми номерами. Детали этого описаны в RFC 1663, но на практике такой способ применяется редко.

Так как в конфигурации по умолчанию поля *Address* и *Control* являются константами, протокол LCP предоставляет возможность двум сторонам договориться о возможности пропускать оба поля и сэкономить, таким образом, по 2 байта на кадр.

Четвертое поле кадра PPP — *Protocol* (протокол). Оно определяет тип пакета, содержащегося в поле данных (*Payload*). Определены коды для протоколов LCP, NCP, IP, IPX, AppleTalk и др. Номера протоколов сетевого уровня, например, IP, IPX, OSI CLNP, XNS, начинаются с бита 0. С бита 1 начинаются коды, используемые для переговоров об использовании других протоколов. К ним относятся LCP, а также различные протоколы NCP для каждого поддерживаемого протокола сетевого уровня. Размер поля *Protocol* по умолчанию составляет 2 байта, однако путем переговоров с помощью LCP этот размер может быть уменьшен до 1 байта.

Поле *Payload* (поле полезной нагрузки, или поле данных) может быть переменной длины, вплоть до некоего оговоренного максимального значения. Если размер не оговорен во время установки соединения при помощи LCP, то по умолчанию он может составлять до 1500 байт. При необходимости данные пользователя могут дополняться специальными символами.

Следом за полем *Payload* располагается поле *Checksum* (контрольная сумма), которое в обычном состоянии занимает 2 байта, но в случае необходимости по договоренности может занимать 4.

Итак, PPP является механизмом формирования кадров, поддерживающим различные протоколы, которым можно пользоваться при модемных соединениях, в последовательных по битам линиях HDLC, сетях SONET и других физических средах. PPP поддерживает обнаружение ошибок, переговоры о параметрах, сжатие заголовков, а также, по желанию, надежное соединение с использованием кадров HDLC.

Рассмотрим теперь способы установления и разрыва соединения. Упрощенная диаграмма на рис. 3.21 показывает фазы, через которые проходит линия связи при ее установлении, использовании и разъединении. Эта последовательность применима как к соединению с помощью модемов, так и к соединениям между маршрутизаторами.

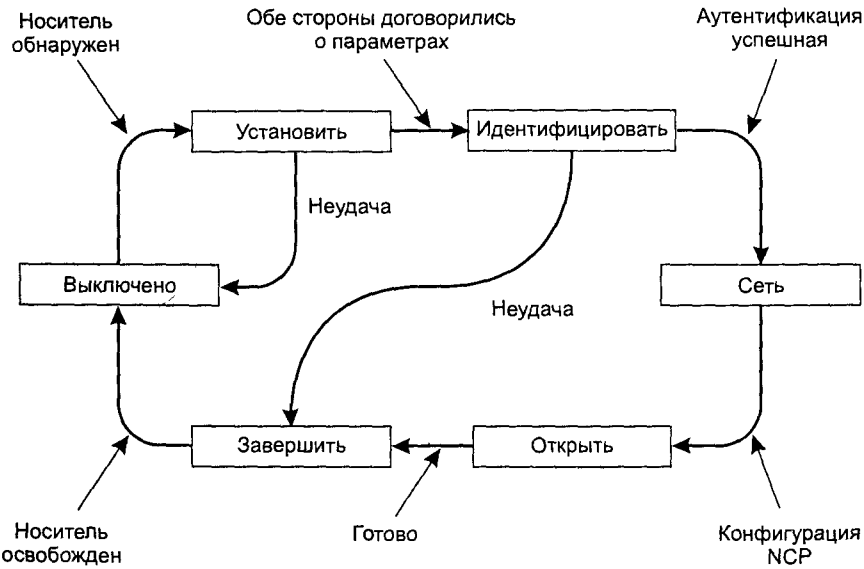


Рис. 3.21. Упрощенная фазовая диаграмма установки и разрыва соединения

Начальное состояние протокола таково: линия отключена (*DEAD*), физический носитель отсутствует, соединения на физическом уровне не существует. После того как физическое соединение установлено, линия переходит в состояние *ESTABLISH* (установка). В этот момент начинаются переговоры о параметрах с помощью протокола LCP. При успешном результате переговоров линия переходит в фазу

AUTHENTICATE (идентифицировать). Теперь обе стороны по желанию могут проверить, кем является собеседник. При переходе к фазе *NETWORK* (сеть) включается соответствующий протокол NCP для настройки сетевого уровня. Если настройка проходит успешно, линия переходит в фазу *OPEN* (открытая), при этом может осуществляться передача данных. Когда передача данных закончена, линия переходит к фазе *TERMINATE* (завершение), а затем снова в состояние *DEAD* (отключена), когда физическое соединение разрывается.

Протокол LCP используется для переговоров об используемых параметрах уровня передачи данных во время установочной фазы (*ESTABLISH*). Причем в его ведении находятся отнюдь не сами обсуждаемые параметры, а механизм переговоров. Он предоставляет способ инициировать процесс подачи предложения и поддерживает ответный процесс принятия или отказа от поданного предложения целиком или частично. Кроме того, он предоставляет методы проверки качества линии, чтобы договаривающиеся процессы могли решить, стоит ли вообще устанавливать соединение на этой линии. Наконец, протокол LCP позволяет отключить линию, если она больше не используется.

В RFC 1661 определены одиннадцать типов LCP-пакетов. Они приведены в табл. 3.1. Четыре первых типа, имена которых содержат слово *Configure-* (конфигурировать), позволяют инициатору переговоров (I) предложить значения параметров, а ответчику (R) принять или отказаться от них. В последнем случае ответчик может выдвинуть встречное заявление о том, что он вообще не хочет вести переговоры. Обсуждаемые параметры и их значения являются частью LCP-пакета.

Коды, начинающиеся со слова *Terminate* (завершить), используются для отключения линии, когда она перестает использоваться. Коды *Code-reject* (код отвергнут) и *Protocol-reject* (протокол отвергнут) применяются ответчиком для сообщения, что он получил что-то непонятное. Это может означать, что произошла ошибка при передаче пакета или инициатор и ответчик используют разные версии протокола LCP. Пакеты, названия которых начинаются с *Echo* (эхо), применяются для проверки качества линии. И наконец, *Discard-request* используется для отладки. Если ему удается пройти по линии, он просто игнорируется ответчиком, чтобы не мешать человеку, осуществляющему отладку.

Таблица 3.1. Типы LCP-пакетов

Имя	Направление	Описание
Configure-request	I→R	Предложение о параметрах и их значениях
Configure-ack	I←R	Все предложенные параметры приняты
Configure-nak	I←R	Некоторые параметры не приняты
Configure-reject	I←R	Некоторые параметры не обсуждаются
Terminate-request	I→R	Запрос на отключение линии
Terminate-ack	I←R	Согласие на отключение линии
Code-reject	I←R	Получен неизвестный запрос
Protocol-reject	I←R	Запрошен неизвестный протокол

Имя	Направление	Описание
Echo-request	I→R	Запрос на обратную пересылку кадра
Echo-reply	I←R	Согласие на обратную пересылку кадра
Discard-request	I→R	Предложение проигнорировать этот кадр (для тестирования)

Обсуждаемые параметры могут включать установку максимального размера поля полезной нагрузки информационных кадров, разрешение аутентификации, выбор используемого протокола, разрешение на проверку качества линии во время нормальной работы, а также выбор различных параметров сжатия заголовков.

О протоколах NCP можно рассказать не так уж много. Каждый из них используется со своим сетевым протоколом и позволяет осуществлять переговоры о параметрах, специфичных для своего протокола. Например, для протокола IP важнейшим свойством является возможность назначения динамического IP-адреса.

Резюме

Задачей уровня передачи данных является преобразование необработанного потока битов, поступающего с физического уровня, в поток кадров, которые может использовать сетевой уровень. Используются различные методы кадрирования, включая подсчет символов, символьное и битовое заполнение. Протоколы уровня передачи данных могут обладать возможностями контроля ошибок, который осуществляется для повторной передачи потерянных или испорченных кадров. Во избежание опережения медленного приемника быстрым отправителем применяется контроль потока. Механизм скользящих окон широко используется для объединения контроля ошибок и управления потоком.

Протоколы скользящего окна можно классифицировать по размеру окна приемника и отправителя. Для протокола с ожиданием оба окна имеют единичный размер. Когда размер окна отправителя больше 1 — например, так делают для уменьшения затрат времени на ожидание подтверждения при большой задержке распространения сигнала по линии, — получатель может либо отвергать все кадры, кроме ожидаемого, либо накапливать поступающие кадры в буфере до тех пор, пока они не понадобятся.

Мы рассмотрели в этой главе ряд примеров протоколов. Протокол 1 предназначен для идеальной среды передачи, в которой отсутствуют ошибки, и для идеального приемника, который может обработать входящий поток любого размера. В протоколе 2 все еще предполагается, что среда не порождает ошибки при передаче, но в данном протоколе уже присутствует контроль потока. Протокол 3 обрабатывает ошибки при помощи порядковых номеров кадров и алгоритма с ожиданием. Протокол 4 предоставляет возможность двунаправленной передачи и использует концепцию комбинированных пакетов. В протоколе 5 используются скользящие окна с возвратом на n. Наконец, протокол 6 отличается применением метода выборочного повтора и отрицательных подтверждений (NAK).

Протоколы можно моделировать разными способами, позволяющими продемонстрировать их корректность (или ее отсутствие). Модели конечных автоматов и сетей Петри используются именно для этих целей.

Многие сети построены на бит-ориентированных протоколах уровня передачи данных — SDLC, HDLC, ADCCP или LAPB. Все они используют флаговые байты, ограничивающие кадры, и битовое заполнение, предотвращающее случайное появление флаговых байтов среди данных пользователя. Также все они используют метод скользящего окна для контроля потока. В Интернете в качестве основного протокола линий «точка — точка» используется PPP.

Вопросы

1. Сообщение верхнего уровня разбито на 10 кадров, у каждого из которых шанс дойти до назначения без повреждений составляет 80 %. Если уровень передачи данных не обеспечивает проверки ошибок, сколько раз в среднем потребуется пересылать все сообщение?
2. В протоколе уровня передачи данных используется следующее кодирование символов:

   A: 01000111; B: 11100011; FLAG: 01111110; ESC: 11100000.

   Как в двоичных кодах будет выглядеть кадр, состоящий из четырех символов — A, B, ESC, FLAG, при использовании каждого из следующих методов кадрирования:

   - подсчет символов;
   - флаговые байты с символьным заполнением;
   - начальные и конечные флаговые байты с битовым заполнением.
3. В потоке данных, для которого применяется алгоритм символьного заполнения, встречается следующий фрагмент данных:

   A B ESC C ESC FLAG FLAG D.

   Каким будет выходной поток после заполнения символами?
4. Один из ваших однокурсников, большой скряга, предположил, что использовать после концевого флагового байта кадра начальный флаговый байт следующего кадра — слишком расточительно, вполне можно обойтись флаговым байтом. Таким образом можно сэкономить передачу одного байта. Вы согласитесь с ним?
5. Каким будет на выходе после применения битового заполнения на уровне передачи данных следующий поток бит: 0111101111101111110?
6. Может ли потеря, вставка или изменение одного бита при использовании битового заполнения вызвать не обнаруживаемую контрольной суммой ошибку? Если нет, почему? Если да, то каким образом? Влияет ли на результат длина используемой контрольной суммы?
7. При каких обстоятельствах протокол без обратной связи (например, с кодом Хэмминга) может быть предпочтительнее протоколов с обратной связью, обсуждаемых в данной главе?

8. Для обеспечения большей надежности, нежели та, которую предоставляет единственный бит четности, в некотором методе обнаружения ошибок один бит четности суммирует все четные биты, а другой — все нечетные. Каково будет в этом случае расстояние кода по Хэммингу?
9. При помощи кода Хэмминга передаются 16-битные сообщения. Сколько контрольных битов потребуется для того, чтобы приемник гарантированно мог обнаруживать и исправлять одиночные битовые ошибки? Как будет выглядеть код для передачи следующего сообщения: 1101001100110101? Предполагается, что код Хэмминга использует проверку четных битов.
10. С помощью кода Хэмминга с проверкой на четность необходимо закодировать байт, двоичное значение которого равно 10101111. Как будет выглядеть эта последовательность после кодирования?
11. Приемник получает 12-битную последовательность в коде Хэмминга, ее шестнадцатеричное значение равно 0xE4F. Как (в шестнадцатеричном виде) выглядела исходная последовательность? Предполагается, что ошибочным может быть только 1 бит.
12. Один из способов обнаружения ошибок заключается в передаче данных в виде блока из n рядов по k бит с добавлением битов четности к каждому ряду и каждой строке. Будет ли такая схема обнаруживать все одиночные ошибки? Двойные ошибки? Тройные ошибки?
13. В блоке битов из n рядов и k строк для обнаружения ошибок используются горизонтальные и вертикальные биты четности. Какова вероятность того, что инверсия 4 битов не будет обнаружена?
14. Чему равен остаток от деления $x^7 + x^5 + 1$ на образующий многочлен $x^3 + 1$?
15. Поток бит 10011101 передается с использованием стандартного метода циклического избыточного кода (CRC), описанного в тексте. Образующий многочлен равен $x^3 + 1$. Какая битовая последовательность будет реально передаваться? Предполагается, что третий бит слева при передаче инвертировался. Докажите, что эта ошибка будет обнаружена приемником.
16. В протоколах передачи данных CRC почти всегда помещается в конце кадра, а не в заголовке. Почему?
17. Скорость передачи данных в канале составляет 4 Кбит/с, а время распространения сигнала — 20 мс. При каком размере кадров эффективность протокола с ожиданиями составит по меньшей мере 50 %?
18. Кабель T1 длиной 3000 км используется для передачи 64-байтовых кадров при помощи протокола 5. Если задержка распространения сигнала составляет 6 мкс/км, сколько бит следует отвести на порядковые номера кадров?
19. Может ли отправитель при использовании протокола 3 начать передачу, когда таймер уже запущен? Если да, то как такое может произойти? Если нет, почему это невозможно?

20. Представьте себе протокол скользящего окна, в котором используется так много битов на порядковые номера кадров, что номера никогда не используются дважды. Какое соотношение должно связывать четыре границы окна и размер окна?

21. Предположим, что в процедуре between протокола 5 вместо условия $a \le b < c$ проверяется условие $a \le b \le c$. Как это повлияет на правильность протокола и его эффективность? Поясните свой ответ.

22. Когда прибывает информационный кадр, протокол 6 проверяет, отличается ли номер кадра от ожидаемого и равна ли переменная no_nak значению true. При выполнении обоих условий посылается NAK. В противном случае запускается вспомогательный таймер. Предположим, что в тексте программы пропущен оператор else. Повлияет ли это на правильность работы протокола?

23. Предположим, что из конца текста программы протокола 6 удалены три строки цикла while. Повлияет ли это на правильность работы протокола или же только на его быстродействие? Поясните свой ответ.

24. Предположим, что оператор case, обрабатывающий случай ошибки контрольной суммы, убран из блока switch в протоколе 6. Как это повлияет на работу протокола?

25. В протоколе 6 в программе, обрабатывающей событие прихода кадра frame_arrival, есть раздел, используемый для отрицательных подтверждений (NAK). Этому участку программы передается управление, когда получаемый кадр является NAK, а также при выполнении другого условия. Приведите пример сценария, в котором наличие этого условия является важным.

26. Представьте, что вы разрабатываете программное обеспечение уровня передачи данных для линии, по которой данные поступают к вам, но не от вас. Другая сторона использует протокол HDLC с 3-битным порядковым номером и размером окна в 7 кадров. Вы хотите для увеличения эффективности буферизировать как можно большее число кадров, однако изменение программы передающей стороны не допускается. Можно ли использовать окно получателя размером более 1 и, тем не менее, гарантировать правильность работы протокола в любых ситуациях? Если да, то какой максимальный размер окна может быть использован?

27. Протокол 6 применяется на безошибочной линии со скоростью 1 Мбит/с. Максимальный размер кадра 1000 бит. Новые пакеты формируются примерно раз в секунду. Интервал тайм-аута установлен на период 10 мс. Если отключить специальный таймер подтверждений, то будут происходить лишние тайм-ауты. Сколько раз в среднем будет передаваться одно сообщение?

28. В протоколе 6 значение $MAX_SEQ = 2^n - 1$. Хотя это условие, очевидно, желательно для эффективного использования битов заголовка, важность его не была показана. Будет ли протокол корректно работать, например, при $MAX_SEQ = 4$?

29. Кадры длиной 1000 бит посылаются по спутниковому каналу с пропускной способностью 1 Мбит/с. Подтверждения всегда посылаются в информационных кадрах. Заголовки кадров очень короткие. Используются 3-битовые порядковые номера. Какой будет максимальная эффективность использования канала при применении:
    1) протокола с ожиданием;
    2) протокола 5;
    3) протокола 6.
30. Какая часть пропускной способности канала теряется на заголовки и повторные передачи при использовании протокола 6 на сильно загруженном спутниковом канале с пропускной способностью 50 Кбит/с. Кадры данных состоят из 40-битовых заголовков и 3960 бит данных. Время распространения сигнала от Земли до спутника составляет 270 мс. Кадры ACK никогда не посылаются. Размер кадров NAK равен 40 бит. Вероятность ошибки для кадра данных составляет 1 %, а для кадра NAK она пренебрежимо мала. Порядковые номера занимают 8 бит.
31. Предположим, что безошибочный спутниковый канал с пропускной способностью 64 Кбит/с используется для пересылки 512-байтных кадров данных в одном направлении с очень короткими подтверждениями, идущими в обратном направлении. Какова будет максимальная скорость передачи данных при размере окна, равном 1, 7, 15 и 127? Время распространения сигнала от Земли до спутника — 270 мс.
32. Кабель длиной в 100 км работает на скорости T1. Скорость распространения сигнала равна 2/3 от скорости света в вакууме. Сколько бит помещается в кабеле?
33. Протокол 4 моделируется при помощи модели конечных автоматов. В скольких состояниях может находиться каждая машина? Канал связи? Вся система (две машины и канал)? Ошибки контрольных сумм игнорировать.
34. Покажите последовательность переходов на сетевой модели Петри, изображенной на рис. 3.16, соответствующую последовательности состояний (000), (01A), (01–), (010), (01A) на рис. 3.14. Объясните, что при этом происходит.
35. Постройте сетевую модель Петри по следующим правилам переходов: $AC \rightarrow B$, $B \rightarrow AC$, $CD \rightarrow E$ и $E \rightarrow CD$. По сетевой модели Петри постройте модель конечного автомата с состояниями, достижимыми из начального состояния ACD. Какие хорошо известные принципы кибернетики моделирует эти правила перехода?
36. Основные идеи протоколов PPP и HDLC очень близки. Чтобы случайно встретившийся в поле данных флаговый байт не вызвал ошибки синхронизации кадров, в протоколе HDLC используется битовое заполнение. Назовите хотя бы одну причину, по которой в протоколе PPP вместо этого применяется символьное заполнение.
37. Каковы минимальные накладные расходы при пересылке IP-пакета по протоколу PPP? Учитывайте только накладные расходы самого протокола PPP, а не заголовки протокола IP.

38. Целью данного упражнения является реализация механизма обнаружения ошибок с помощью стандартного алгоритма циклического избыточного кода (CRC), описанного в тексте. Напишите две программы: генератор (`generator`) и верификатор (`verifier`). Программа-генератор считывает со стандартного устройства ввода n-битное сообщение из нулей и единиц, представленных в виде строки ASCII-текста. Вторая строка является k-битным многочленом (также в ASCII). На устройстве вывода печатается текст из $n + k$ нулей и единиц, представляющий собой сообщение, подлежащее пересылке. Затем печатается многочлен в том же виде, в каком он был считан. Программа-верификатор считывает результат работы генератора и выводит сообщение, в котором сообщается, корректен ли данный результат. Наконец, напишите программу (`alter`), вносящую сбой, а именно инвертирующую только один бит первой строки в зависимости от аргумента (например, порядкового номера бита, предполагая, что слева располагается бит с номером 1). Все остальные данные передаются без изменений. Набрав в командной строке `generator <file | verifier` пользователь должен увидеть сообщение о том, что данные переданы корректно. Набрав `generator <file | alter arg | verifier` пользователь должен получить сообщение об ошибке при передаче.
39. Напишите программу для моделирования поведения сети Петри. Программа должна считывать правила переходов и список состояний, соответствующих сетевому уровню, передающему или принимающему новый пакет. Из начального состояния, также считываемого программой, программа должна случайно выбирать переходы, проверяя возможность ситуации, когда один хост примет два сообщения, и при этом другой хост между этими событиями не выпустит нового сообщения.

Глава 4
Подуровень управления доступом к среде

- Проблема распределения канала
- Протоколы коллективного доступа
- Сеть Ethernet
- Беспроводные локальные сети
- Широкополосные беспроводные сети
- Bluetooth
- Коммутация на уровне передачи данных
- Резюме
- Вопросы

Как уже указывалось в главе 1, все сетевые технологии могут быть разделены на две категории: использующие соединения от узла к узлу и сети с применением широковещания. Эта глава посвящена широковещательным сетям и их протоколам.

Главной проблемой любых широковещательных сетей является вопрос о том, как определить, кому предоставить канал, если пользоваться им одновременно хотят несколько компьютеров. Для примера представьте себе конференцию, в которой принимают участие шесть человек, причем каждый использует свой телефон. Все они соединены таким образом, что каждый может слышать всех остальных. Весьма вероятно, что когда один из них закончит свою речь, сразу двое или трое начнут говорить одновременно, тем самым создав неловкую ситуацию. При личной встрече подобные проблемы предотвращаются внешними средствами, например поднятием руки для получения разрешения говорить. Когда доступен лишь один канал, определить, кто может говорить следующим, значительно труднее. Для решения этой проблемы разработано много протоколов, которые и будут обсуждаться в данной главе. В литературе широковещательные каналы иногда

называют **каналами с множественным доступом**, или **каналами с произвольным доступом**.

Протоколы, применяющиеся для определения того, кто будет говорить следующим, относятся к подуровню уровня передачи данных, называемому **MAC** (Medium Access Control — управление доступом к среде). Подуровень MAC особенно важен в локальных сетях, так как почти все они используют канал множественного доступа. В глобальных сетях, напротив, применяются двухточечные соединения. Исключением являются только спутниковые сети. Поскольку каналы множественного доступа тесно связаны с локальными сетями, в данной главе в основном будут описываться локальные сети, а также некоторые вопросы, напрямую не связанные с темой подуровня MAC.

Технически подуровень управления доступом к среде является нижней частью уровня передачи данных, поэтому логичнее было бы изучить сначала его, а затем протоколы «точка—точка», рассмотренные в главе 3. Тем не менее, большинству людей понять протоколы, включающие многих участников, легче после того, как хорошо изучены протоколы с двумя участниками. По этой причине при рассмотрении уровней мы слегка отклонились от строгого следования снизу вверх по иерархической лестнице.

Проблема распределения канала

Центральной проблемой, обсуждаемой в этой главе, является распределение одного широковещательного канала между многочисленными пользователями, претендующими на него. Сначала мы в общих чертах рассмотрим статические и динамические схемы распределения канала. Затем обсудим несколько конкретных алгоритмов.

Статическое распределение канала в локальных и региональных сетях

Традиционным способом распределения одного канала — например, телефонного кабеля — между многочисленными конкурирующими пользователями является FDM (Frequency Division Multiplexing — частотное уплотнение). При наличии N пользователей полоса пропускания делится на N диапазонов одинаковой ширины (см. рис. 2.27), и каждому пользователю предоставляется один из них. Поскольку при такой схеме у каждого оказывается свой личный частотный диапазон, то конфликта между пользователями не возникает. При небольшом количестве абонентов, каждому из которых требуется постоянная линия связи (например, коммутаторы операторов связи), частотное уплотнение предоставляет простой и эффективный механизм распределения.

Однако при большом и постоянно меняющемся количестве отправителей данных или пульсирующем трафике частотное уплотнение не может обеспечить достаточно эффективное распределение канала. Если количество пользователей в какой-либо момент времени меньше числа диапазонов, на которые разделен

спектр частот, то большая часть спектра не используется и тратится попусту. Если, наоборот, количество пользователей окажется больше числа доступных диапазонов, то некоторым придется отказать в доступе к каналу, даже если абоненты, уже захватившие его, почти не будут использовать пропускную способность.

Однако даже если предположить, что количество пользователей можно каким-то способом удерживать на постоянном уровне, то разделение канала на статические подканалы все равно является неэффективным. Основная проблема здесь состоит в том, что если какая-то часть пользователей не пользуется каналом, то эта часть спектра просто пропадает. Они сами при этом занимают линию, не передавая ничего, и другим не дают передать данные. Кроме того, в большинстве компьютерных систем трафик является чрезвычайно неравномерным (вполне обычным является отношение пикового трафика к среднему как 1000:1). Следовательно, большую часть времени большая часть каналов не будет использоваться.

То, что характеристики статического частотного уплотнения оказываются неудачными, можно легко продемонстрировать на примере простых вычислений теории массового обслуживания. Для начала сосчитаем среднее время задержки T для канала с пропускной способностью C бит/с, по которому прибывают λ кадров в секунду. Длина кадров является случайной величиной с экспоненциально распределенной плотностью вероятности, среднее значение которой равно $1/\mu$ бита на кадр. При таких параметрах скорость прибытия составляет λ кадров в секунду, а скорость обслуживания — μC кадров в секунду. Теория массового обслуживания говорит о том, что пуассоновское время прибытия и обслуживания равно

$$T = \frac{1}{\mu C - \lambda}.$$

Пусть, например, пропускная способность C равна 100 Мбит/с, средняя длина кадра $1/\mu$ = 10 000 бит, скорость прибытия кадров λ = 5000 кадров в секунду. Тогда T = 200 мкс. Обратите внимание: если бы мы не учли задержки при формировании очереди и просто посчитали, сколько времени нужно на передачу кадра длиной 10 000 бит по сети с пропускной способностью 100 Мбит/с, то получили бы неправильный ответ: 100 мкс. Это число приемлемо лишь при отсутствии борьбы за канал.

Теперь давайте разделим канал на N независимых подканалов, у каждого из которых будет пропускная способность C/N бит/с. Средняя входная скорость в каждом подканале теперь будет равна λ/N кадров в секунду. Сосчитав новое значение средней задержки T, получим:

$$T_{\text{FDM}} = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT. \qquad (4.1)$$

Это означает, что при использовании частотного уплотнения значение средней задержки стало в N раз хуже значения, которое было бы в канале, если бы все кадры были каким-то волшебным образом организованы в одну общую очередь.

Те же самые аргументы применимы и к временному уплотнению (TDM, Time Division Multiplexing — мультиплексная передача с временным разделением). Каждому пользователю в данном случае статически выделяется N-й интервал времени. Если интервал не используется абонентом, то он просто пропадает. С тем же успехом можно разделить сети физически. Если взять 100-мегабитную сеть и сделать из нее десять 10-мегабитных, статически распределив по ним пользователей, то в результате средняя задержка возрастет с 200 мкс до 2 мс.

Таким образом, ни один статический метод распределения каналов не годится для пульсирующего трафика, поэтому далее мы рассмотрим динамические методы.

Динамическое распределение каналов в локальных и региональных сетях

Прежде чем приступить к рассмотрению многочисленных методов распределения каналов, следует тщательно сформулировать решаемую проблему. В основе всех разработок в данной области лежат следующие пять допущений.

1. **Станционная модель.** Модель состоит из N независимых **станций** (компьютеров, телефонов, персональных средств связи и т. д.), в каждой из которых программа пользователя формирует кадры для передачи. Станции иногда называют **терминалами**. Вероятность формирования кадра в интервале времени Δt равна $\Delta\lambda t$, где λ является константой (скорость прибытия новых кадров). Как только кадр сформирован, станция блокируется и ничего не делает, пока кадр не будет успешно передан.
2. **Предположение о едином канале.** Единый канал доступен для всех. Все станции могут передавать и принимать данные по нему. С точки зрения аппаратуры все станции считаются равными, хотя программно протокол может устанавливать для них различные приоритеты.
3. **Допущение о коллизиях.** Если два кадра передаются одновременно, они перекрываются по времени, в результате сигнал искажается. Такое событие называется конфликтом, или коллизией. Все станции могут обнаруживать конфликты. Искаженный вследствие конфликта кадр должен быть передан повторно. Других ошибок, кроме тех, которые вызваны конфликтами, нет.

4*а*. **Непрерывное время.** Передача кадров может начаться в любой момент времени. Не существует никаких синхронизирущих импульсов, которые делили бы время на дискретные интервалы.

4*б*. **Дискретное время.** Время разделено на дискретные интервалы (такты). Передача кадра может начаться только с началом такта. Один временной интервал может содержать 0, 1 или более кадров, что соответствует свободному интервалу, успешной передаче кадра или коллизии.

5*а*. **Контроль несущей.** Станции могут определить, свободна или занята линия, до ее использования. Если канал занят, станции не будут пытаться передавать кадры по нему, пока он не освободится.

5*б*. **Отсутствие контроля несущей.** Станции не могут определить, свободна или занята линия, пока не попытаются ее использовать. Они просто начинают передачу. Только потом они могут определить, была ли передача успешной.

О приведенных ранее допущениях следует сказать несколько слов. Первое допущение утверждает, что станции независимы и работают с постоянной скоростью. Также неявно предполагается, что у каждой станции имеется только один пользователь или программа, поэтому пока станция заблокирована, она не производит никакой работы. Более сложные модели рассматривают многопрограммные станции, которые могут работать в заблокированном состоянии, однако и анализ подобных станций намного сложнее.

Допущение о едином канале является, на самом деле, центральным в данной модели. Никаких внешних каналов связи нет. Станции не могут тянуть руки, привлекая к себе внимание и убеждая учителя спросить их.

Допущение о коллизиях также является основным, хотя в некоторых системах (особенно в системах с расширенным спектром) данное допущение звучит не столь строго, что приводит к неожиданным результатам. Кроме того, в некоторых локальных сетях, например в сети *token ring* (маркерное кольцо), применяется механизм разрешения коллизий, реализуемый за счет специальных пакетов — маркеров, передающихся от станции к станции. Помещать в канал кадр может только тот, у кого в данный момент находится маркер. Далее мы обсудим модель моноканала с конкуренцией и коллизиями.

Для времени существует два альтернативных допущения. В одних системах время может быть непрерывным (4*а*), в других — дискретным, поэтому мы рассмотрим оба варианта.

Аналогично этому, контроль несущей также реализован не во всех системах. Станции локальных сетей обычно знают, когда линия занята, однако в беспроводных сетях контроля несущей нет, потому что отдельно взятая станция не может «слышать» все остальные из-за разницы частотных диапазонов. Станции проводных сетей с контролем несущей могут приостанавливать собственную передачу, обнаружив коллизию. Обратите внимание на слово «несущая». В данном случае оно означает электрический сигнал, распространяющийся по каналу[1].

Протоколы коллективного доступа

Известно много алгоритмов коллективного доступа. В следующих разделах будут рассмотрены наиболее интересные алгоритмы и даны примеры их применения.

Система ALOHA

В 70-х годах Норман Абрамсон (Norman Abramson) и его коллеги из Гавайского университета разработали новый элегантный метод решения проблемы распределения каналов. Их труды впоследствии стали основой многих исследований

[1] Слово «carrier» («несущая») в английском языке означает также «перевозчик». — *Примеч. перев.*

(Abramson, 1985). Хотя в работе Абрамсона, получившей название ALOHA, использовалась широковещательная радиосвязь со стационарными передатчиками, основная идея применима к любой системе, в которой независимые пользователи соревнуются за право использования одного общего канала.

В данной главе мы рассмотрим две версии системы ALOHA: чистую и дискретную. Они отличаются тем, делится ли время на дискретные интервалы, в течение которых передаются кадры, или нет. В чистой системе ALOHA не требуется общая синхронизация времени, а в дискретной требуется.

Чистая система ALOHA

В основе системы ALOHA лежит простая идея: разрешить пользователям передачу, как только у них появляются данные для отсылки. Конечно, при этом будут столкновения, и столкнувшиеся кадры будут разрушены. Однако благодаря свойству обратной связи широковещательной системы отправитель всегда может установить, дошел ли его кадр до получателя или был разрушен. Для этого ему нужно просто прослушивать канал, как это делают все остальные пользователи. В локальных сетях обратная связь мгновенная, а в спутниковых системах существует задержка в 270 мс, и только после этого отправитель может узнать, насколько успешной была передача. Если кадр был уничтожен, отправитель просто выжидает некоторое случайное время и пытается переслать этот кадр снова. Время ожидания должно быть случайным. В противном случае при равных фиксированных интервалах времени ожидания коллизии будут повторяться снова и снова. Системы, в которых несколько пользователей используют один общий канал таким способом, что время от времени возникают конфликты, называются **системами с конкуренцией**.

На рис. 4.1 показан пример формирования кадров в системе ALOHA. Все кадры на нашем рисунке имеют один размер, так как при этом пропускная способность системы сделана максимальной именно за счет фиксированного размера кадров.

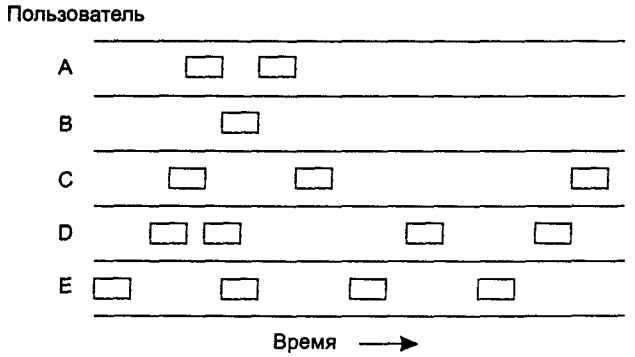


Рис. 4.1. В чистой системе ALOHA кадры передаются в абсолютно произвольное время

Когда два кадра одновременно пытаются занять канал, они сталкиваются и уничтожаются. Даже если только один первый бит второго кадра перекрывается последним битом первого кадра, оба кадра уничтожаются полностью. При этом оба

кадра будут переданы позднее повторно. Контрольная сумма не может (и не должна) отличать полную потерю информации от частичной. Потеря есть потеря.

Самым интересным в данной ситуации является вопрос об эффективности канала системы ALOHA. Другими словами, какая часть всех передаваемых кадров способна избежать коллизий при любых обстоятельствах? Сначала рассмотрим бесконечное множество интерактивных пользователей, сидящих за своими компьютерами (станциями). Пользователь всегда находится в одном из двух состояний: ввод с клавиатуры и ожидание. Вначале все пользователи находятся в состоянии ввода. Закончив набор строки, пользователь перестает вводить текст, ожидая ответа. В это время станция передает кадр, содержащий набранную строку, и опрашивает канал, проверяя успешность передачи кадра. Если кадр передан успешно, пользователь видит ответ и продолжает набор. В противном случае пользователь ждет, пока кадр не будет передан.

Пусть «время кадра» означает интервал времени, требующийся для передачи стандартного кадра фиксированной длины (то есть длину кадра, деленную на скорость передачи данных). На данный момент мы предполагаем, что бесконечное количество пользователей порождает новые кадры, распределенные по Пуассону со следующим средним значением: N кадров за время кадра. (Допущение о бесконечном количестве пользователей необходимо для того, чтобы гарантировать, что величина N не станет уменьшаться по мере блокирования пользователей.) Если $N > 1$, это означает, что сообщество пользователей формирует кадры с большей скоростью, чем может быть передано по каналу, и почти каждый кадр будет страдать от столкновений. Мы будем предполагать, что $0 < N < 1$.

Помимо новых кадров, станции формируют повторные передачи кадров, пострадавших от столкновений. Допустим также, что вероятность k попыток передачи старых и новых кадров за время кадра также имеет пуассоновское распределение со средним значением G за время кадра. Очевидно, что $G \geq N$. При малой загрузке канала (то есть при $N \approx 0$) столкновений будет мало, поэтому мало будет и повторных передач, то есть $G \approx N$. При большой загрузке канала столкновений будет много, а следовательно, $G > N$. Какая бы ни была нагрузка, производительность канала S будет равна предлагаемой загрузке G, умноженной на вероятность успешной передачи, то есть $S = GP_0$, где P_0 — вероятность того, что кадр не пострадает в результате коллизии.

Кадр не пострадает от коллизии в том случае, если в течение интервала времени его передачи не будет послано больше ни одного кадра, как показано на рис. 4.2. При каких условиях затененный кадр будет передан без повреждений? Пусть t — это время, требуемое для передачи кадра. Если какой-либо пользователь сформирует кадр в интервале времени между t_0 и $t_0 + t$, то конец этого кадра столкнется с началом затененного кадра. При этом судьба затененного кадра предрешена еще до того, как будет послан его первый бит, однако, поскольку в чистой системе ALOHA станции не прослушивают канал до начала передачи, у них нет способа узнать, что канал занят и по нему уже передается кадр. Аналогичным образом любой другой кадр, передача которого начнется в интервале от $t_0 + t$ до $t_0 + 2t$, столкнется с концом затененного кадра.

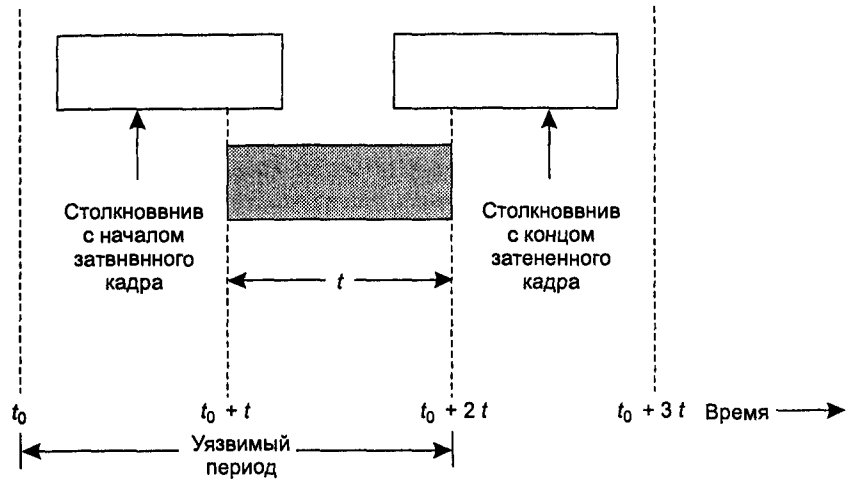


Рис. 4.2. Опасный интервал времени для затененного кадра

Вероятность того, что в течение времени кадра будет сформировано k кадров, можно вычислить по формуле распределения Пуассона:

$$\Pr[k] = \frac{G^k e^{-G}}{k!}. \tag{4.2}$$

Таким образом, вероятность формирования нуля кадров в течение этого интервала времени равна e^{-G}. Среднее количество кадров, сформированных за интервал времени длиной в два кадра, равно $2G$. Вероятность того, что никто не начнет передачу в течение всего опасного периода, равна $P_0 = e^{-2G}$. Учитывая, что $S = GP_0$, получаем:

$$S = Ge^{-2G}.$$

Зависимость производительности канала от предлагаемого трафика показана на рис. 4.3. Максимальная производительность достигает значения $S = 1/2e$, что приблизительно равно 0,184 при $G = 0{,}5$. Другими словами, лучшее, на что мы можем надеяться, — это использовать канал на 18 %. Этот результат несколько разочаровывает, однако в случае, когда каждый передает, когда хочет, трудно ожидать стопроцентного успеха.

Дискретная система ALOHA

В 1972 г. Робертс (Roberts) опубликовал описание метода, позволяющего удвоить производительность систем ALOHA (Roberts, 1972). Его предложение заключалось в разделении времени на дискретные интервалы, соответствующие времени одного кадра. При таком подходе пользователи должны согласиться с определенными временными ограничениями. Одним из способов достижения синхронизации является установка специальной станции, испускающей синхронизирующий сигнал в начале каждого интервала.

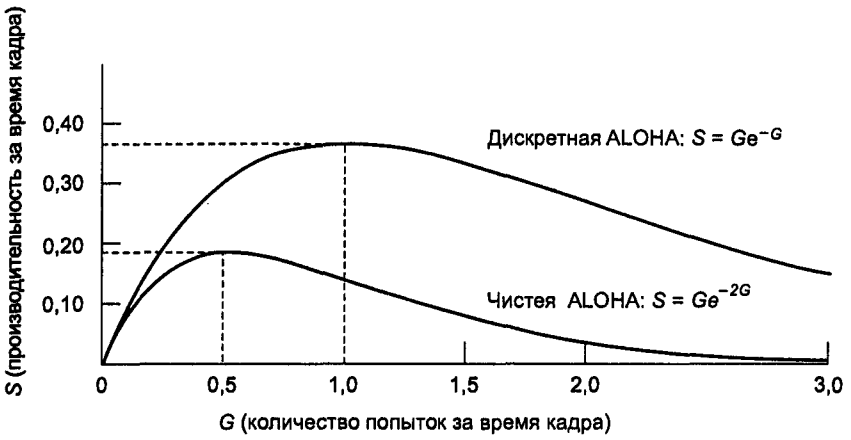


Рис. 4.3. Зависимость производительности канала от предлагаемого трафика для систем ALOHA

В системе Робертса, известной под названием **дискретная ALOHA**, в отличие от **чистой системы ALOHA** Абрамсона, компьютер не может начинать передачу сразу после нажатия пользователем клавиши Enter. Вместо этого он должен дождаться начала нового такта. Таким образом, непрерывная чистая система ALOHA превращается в дискретную. Поскольку опасный временной интервал теперь становится в два раза короче, вероятность отсутствия передачи по каналу за тот же интервал времени, в течение которого передается тестовый кадр, равна e^{-G}. В результате получаем:

$$S = Ge^{-G}. \tag{4.3}$$

Как видно из рис. 4.3, дискретная система ALOHA имеет пик при $G = 1$. При этом производительность канала составляет $S = 1/e$, что приблизительно равно 0,368, то есть в два раза больше, чем в чистой системе ALOHA. Для дискретной системы ALOHA в оптимальной ситуации 37 % интервалов будут пустыми, 37 % — с успешно переданными кадрами и 26 % — со столкнувшимися кадрами. При увеличении количества попыток передачи в единицу времени G количество пустых интервалов уменьшается, но увеличивается количество конфликтных интервалов. Чтобы увидеть, насколько быстро растет количество конфликтных интервалов, рассмотрим передачу тестового кадра. Вероятность того, что он избежит столкновения, равна e^{-G}. Фактически, это вероятность того, что все остальные пользователи будут молчать в течение данного тактового интервала. Таким образом, вероятность столкновения равна $1 - e^{-G}$. Вероятность передачи кадра ровно за k *попыток (то есть после* $k - 1$ *столкновения, за которыми последует успешная передача), равна*

$$P_k = e^{-G}(1 - e^{-G})^{k-1}.$$

Ожидаемое число попыток передачи для одного кадра равно

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1 - e^{-G})^{k-1} = e^{G}.$$

Поскольку число попыток передачи для одного кадра E экспоненциально зависит от количества попыток передачи в единицу времени G, небольшое увеличение нагрузки в канале может сильно снизить его производительность.

Дискретная система ALOHA чрезвычайно важна по одной причине, которая на первый взгляд не кажется очевидной. Она появилась в 1970-х годах, применялась в некоторых экспериментальных системах, затем была почти забыта. Когда был изобретен метод доступа в Интернет по кабельным сетям, вновь возникла проблема распределения единственного канала между большим числом конкурирующих абонентов. Тогда с полок достали запыленные описания дискретной ALOHA. Не раз уже было так, что вполне работоспособные протоколы и методы оказывались невостребованными по политическим причинам (например, когда какая-нибудь крупная компания выражала желание, чтобы все на свете использовали исключительно ее продукцию), однако по прошествии многих лет какой-нибудь мудрый человек вспоминал о существовании одного древнего метода, способного решить современную проблему. По этой причине мы изучим в этой главе ряд элегантных протоколов, которые сейчас широко не используются, но запросто могут оказаться востребованными в будущем — если, конечно, об их существовании будет знать достаточное количество разработчиков сетей. Разумеется, мы изучим и используемые в настоящее время протоколы.

Протоколы множественного доступа с контролем несущей

В дискретной системе ALOHA максимальный коэффициент использования канала, который может быть достигнут, равен $1/e$. Такой скромный результат неудивителен, поскольку станции передают данные, когда хотят, не считаясь с тем, что делают остальные станции. В такой системе неизбежно возникает большое количество коллизий. Однако в локальных сетях можно организовать процесс таким образом, что станции будут учитывать поведение друг друга. За счет этого можно достичь значения коэффициента использования канала значительно большего, чем $1/e$. В данном разделе мы рассмотрим некоторые протоколы, позволяющие улучшить производительность канала.

Протоколы, в которых станции прослушивают среду передачи данных и действуют в соответствии с этим, называются **протоколами с контролем несущей**. Было разработано много таких протоколов. Кляйнрок (Kleinrock) и Тобаги (Tobagi) в 1975 году детально исследовали несколько таких протоколов. Далее мы рассмотрим несколько версий протоколов с контролем несущей.

Настойчивый и ненастойчивый CSMA

Первый протокол с опросом несущей, который мы рассмотрим, называется **1-настойчивый протокол CSMA** (Carrier Sense Multiple Access — множественный доступ с контролем несущей). Когда у станции появляются данные для передачи, она сначала прослушивает канал, проверяя, свободен он или занят. Если канал занят, то есть по нему передает какая-либо другая станция, станция ждет, пока он освободится. Когда канал освобождается, станция передает кадр. Если происхо-

дит столкновение, станция ждет в течение случайного интервала времени, затем снова прослушивает канал и, если он свободен, пытается передать кадр еще раз. Такой протокол называется протоколом CSMA с настойчивостью 1, так как станция передает кадр с вероятностью 1, как только обнаружит, что канал свободен.

Задержка распространения сигнала оказывает сильное влияние на производительность данного протокола. Существует небольшая вероятность того, что как только станция начнет передачу, другая станция также окажется готовой к передаче и опросит канал. Если сигнал от первой станции еще не успел достичь второй станции, вторая станция решит, что канал свободен, и также начнет передачу, результатом чего будет коллизия. Чем больше время распространения сигнала, тем выше вероятность столкновений и ниже производительность протокола.

Даже при нулевой задержке распространения сигнала все равно будут столкновения. Если две станции придут в состояние готовности в то время, когда передает какая-то третья станция, обе будут ждать, пока она не закончит передачу, после чего сами одновременно станут передавать, и в результате произойдет столкновение. Если бы они не были столь нетерпеливы, количество столкновений было бы меньшим. Однако даже такая система значительно лучше чистой системы ALOHA, так как обе станции воздерживаются от передачи, пока передает третья станция. Очевидно, что благодаря этому производительность системы с опросом несущей должна быть выше даже чем у дискретной системы ALOHA.

Вторым протоколом с опросом несущей является **ненастойчивый протокол CSMA**. В данном протоколе предпринята попытка сдержать стремление станций начинать передачу, как только освобождается канал. Прежде чем начать передачу, станция опрашивает канал. Если никто не передает в данный момент по каналу, станция начинает передачу сама. Однако если канал занят, станция не ждет освобождения канала, постоянно прослушивая его и пытаясь захватить сразу, как только он освободится, как в предыдущем протоколе. Вместо этого станция ждет в течение случайного интервала времени, а затем снова прослушивает линию. Очевидно, данный алгоритм должен привести к лучшему использованию канала и к большим интервалам ожидания, чем протокол CSMA с настойчивостью 1.

Наконец, третий протокол, который мы рассмотрим, это **протокол CSMA с настойчивостью p**. Он применяется в дискретных каналах и работает следующим образом. Когда станция готова передавать, она опрашивает канал. Если канал свободен, она с вероятностью p начинает передачу. С вероятностью $q = 1 - p$ она отказывается от передачи и ждет начала следующего такта. Этот процесс повторяется до тех пор, пока кадр не будет передан или какая-либо другая станция не начнет передачу. В последнем случае станция ведет себя так же, как в случае столкновения. Она ждет в течение случайного интервала времени, после чего начинает все снова. Если при первом прослушивании канала он оказывается занят, станция ждет следующего интервала времени, после чего применяется тот же алгоритм. На рис. 4.4 показана расчетная зависимость производительности канала от предлагаемого потока кадров для всех трех протоколов, а также для чистой и дискретной систем ALOHA.

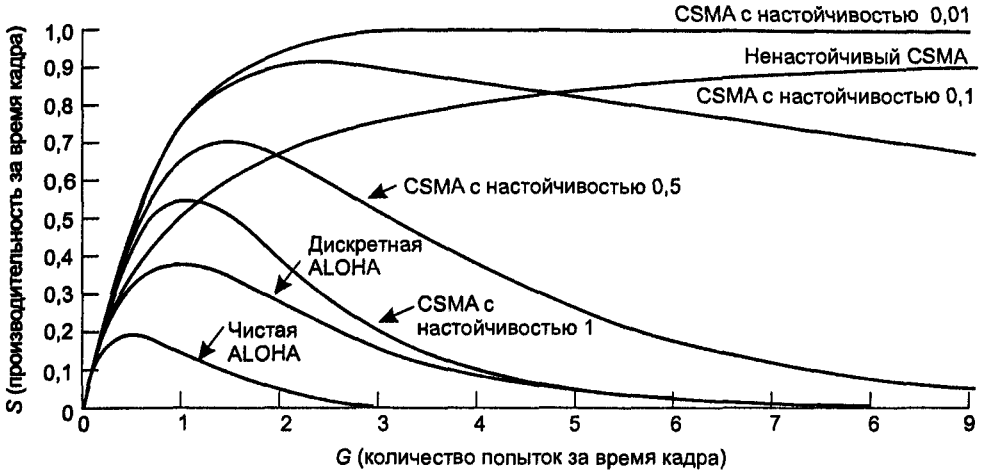


Рис. 4.4. Сравнение использования канала в зависимости от его загрузки для различных протоколов коллективного доступа

Протокол CSMA с обнаружением конфликтов

Настойчивый и ненастойчивый протоколы CSMA, несомненно, являются улучшениями системы ALOHA, поскольку они гарантируют, что никакая станция не начнет передачу, если она определит, что канал уже занят. Еще одним шагом вперед является прекращение станцией передачи, если выясняется, что произошел конфликт. Другими словами, если две станции, обнаружив, что канал свободен, одновременно начали передачу, они практически немедленно обнаруживают столкновение. Вместо того чтобы пытаться продолжать передачу своих кадров, которые все равно уже не могут быть приняты получателями, им следует прекратить передачу. Таким образом экономится время и улучшается производительность канала. Такой протокол, называемый CSMA/CD (Carrier-Sense Multiple Access with Collision Detection — множественный доступ с контролем несущей и обнаружением конфликтов), широко применяется в локальных сетях в подуровне MAC. В частности, он является основой чрезвычайно популярных ЛВС Ethernet, поэтому мы уделим некоторое время более или менее подробному рассмотрению CSMA/CD.

В протоколе CSMA/CD, так же как и во многих других протоколах локальных сетей, применяется концептуальная модель, показанная на рис. 4.5. В момент времени t_0 одна из станций закончила передачу кадра. Все остальные станции, готовые к передаче, теперь могут попытаться передать свои кадры. Если две или более станций одновременно начнут передачу, то произойдет столкновение. Столкновения могут быть обнаружены по мощности или длительности импульса принимаемого сигнала в сравнении с передаваемым сигналом.

Обнаружив коллизию, станция прекращает передачу, ждет случайный период времени, после чего пытается снова при условии, что к этому моменту не начала передачу другая станция. Таким образом, наша модель протокола CSMA/CD бу-

дет состоять из чередования периодов конкуренции и передачи, а также периодов простоя канала (когда все станции молчат).

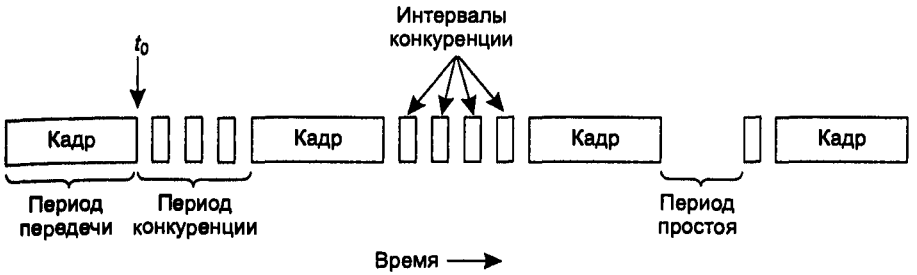


Рис. 4.5. Протокол CSMA/CD может находиться в одном из трех состояний: конкуренции, передвчи и простоя

Рассмотрим более подробно алгоритм борьбы за право передачи по каналу. Предположим, две станции одновременно начали передачу в момент времени t_0. Сколько понадобится времени на то, чтобы они поняли, что произошло столкновение? От ответа на этот вопрос зависит длина периода конкуренции, а следовательно, величина задержки и производительность канала. Минимальное время обнаружения конфликта равно времени распространения сигнала от одной станции до другой.

Исходя из этих рассуждений, можно предположить, что станция, которая не слышит столкновения в течение времени, требующегося для прохождения сигнала по всему кабелю, может быть уверена, что ей удалось захватить кабель. Под термином «захватить» имеется в виду, что все остальные станции знают, что эта станция передает, и не будут сами пытаться передавать. Однако такое заключение неверно. Рассмотрим следующий сценарий. Пусть время, необходимое для прохождения сигнала между двумя самыми дальними станциями, равно τ. В момент времени t_0 одна из станций начинает передачу. Через интервал времени $\tau - \varepsilon$, за мгновение до того, как сигнал достигнет самой дальней станции, та станция также начинает передавать. Конечно, почти мгновенно она обнаруживает столкновение и останавливается, но всплеск шума, вызванный столкновением, достигает передающей станции только через интервал времени $2\tau - \varepsilon$ с момента начала передачи. Другими словами, станция не может быть уверена в том, что захватила канал, до тех пор, пока не пройдет интервал времени 2τ с момента начала передачи. По этой причине для моделирования интервала конкуренции мы будем использовать дискретную систему ALOHA с шириной интервала 2τ. В коаксиальном кабеле длиной 1 км $\tau \approx 5$ мкс. Для простоты мы будем предполагать, что каждый интервал времени 2τ содержит всего 1 бит. Как только канал захвачен, станция может передавать с любой скоростью, не обязательно 1 бит за 2τ с.

Следует отметить, что обнаружение столкновения является *аналоговым* процессом. Аппаратура станции должна прослушивать кабель во время передачи. При этом, если то, что она слышит, отличается от того, что она передает, станция понимает, что произошло столкновение. Способ кодирования сигнала должен

позволять определять столкновения (например, столкновение двух сигналов в 0 В в явном виде не так просто обнаружить). По этой причине используется специальное кодирование.

Передающая станция должна постоянно прослушивать канал, выявляя всплески шума, которые могут означать столкновение. По этой причине CSMA/CD с моноканалом считается полудуплексной системой. Станция не может одновременно передавать и принимать кадры, поскольку задействован механизм обратной связи для определения столкновений.

Во избежание неправильного понимания вопроса следует также отметить, что ни один протокол подуровня MAC не может гарантировать надежную доставку. Даже при отсутствии столкновений получатель может не получить правильную копию кадра по различным причинам (например, из-за нехватки места в буфере или пропущенного прерывания).

Протоколы без столкновений

Хотя в протоколе CSMA/CD столкновения не могут происходить после того, как станция захватывает канал, они могут случаться в период конкуренции. Эти столкновения снижают производительность системы, особенно при большой длине кабеля (то есть при больших τ) и коротких кадрах. Метод CSMA/CD оказывается не универсальным. В данном разделе мы рассмотрим протоколы, которые решают проблему борьбы за право занять канал, причем делают это даже без периода конкуренции.

В описываемых далее протоколах предполагается наличие N станций, у каждой из которых есть постоянный уникальный адрес в пределах от 0 до $N-1$. То, что некоторые станции могут часть времени оставаться пассивными, роли не играет. Также предполагается, что задержка распространения сигнала пренебрежимо мала. Главный вопрос сохраняется: какой станции будет предоставлен канал после передачи данного кадра? Мы будем по-прежнему использовать модель, изображенную на рис. 4.5, с ее дискретными интервалами конкуренции.

Протокол битовой карты

В первом протоколе без столкновений, который мы рассмотрим, называющемся **основным методом битовой карты**, каждый период конкуренции состоит ровно из N временных интервалов. Если у станции 0 есть кадр для передачи, она передает единичный бит во время 0-го интервала. Другим станциям не разрешается передача в это время. Во время интервала 1 станция 1 также сообщает, есть ли у нее кадр для передачи, передавая бит 1 или 0. В результате к окончанию интервала N все N станций знают, кто хочет передавать. В этот момент они начинают передачу в соответствии со своим порядком номеров (рис. 4.6).

Поскольку все знают, чья очередь передавать, столкновений нет. После того как последняя станция передает свой кадр, что все станции отслеживают, прослушивая линию, начинается новый период подачи заявок из N интервалов. Если станция переходит в состояние готовности (получает кадр для передачи) сразу после того, как она отказалась от передачи, это значит, что ей не повезло и она

должна ждать следующего цикла. Протоколы, в которых намерение передавать объявляется всем перед самой передачей, называются **протоколами с резервированием.**



Рис. 4.6. Базовый протокол битовой карты

Оценим производительность такого протокола. Для удобства будем измерять время в однобитовых интервалах периода подачи заявок, при этом кадр данных состоит из d единиц времени. При слабой загрузке канала бит-карта просто будет повторяться снова и снова, изредка перемежаясь кадрами.

Рассмотрим эту ситуацию с точки зрения станции с небольшим номером, например, 0 или 1. Обычно в тот момент, когда у нее возникает потребность в передаче, текущий интервал времени уже находится где-то в середине бит-карты. В среднем станция будет ждать $N/2$ интервалов до окончания текущего периода резервирования и еще N интервалов следующего (своего) периода резервирования, не считая кадров, передаваемых между двумя этими периодами, прежде чем она сможет начать передачу.

Перспективы станций с большими номерами более радужны. В среднем время ожидания передачи составит половину цикла ($N/2$ однобитовых интервалов). Станциям с большими номерами редко приходится ждать следующего цикла. Поскольку станциям с небольшими номерами приходится ждать в среднем $1{,}5N$ интервалов, а станциям с большими номерами — $N/2$ интервалов, среднее время ожидания для всех станций составляет N интервалов. При низкой загрузке канала его производительность легко сосчитать. Накладные расходы на кадр составляют N бит, и при длине кадра в d бит эффективность равна $d/(N + d)$.

При сильной загруженности канала, когда все станции хотят что-то передать, период подачи заявок из N бит чередуется с N кадрами. При этом накладные расходы на передачу одного кадра составляют всего один бит, а эффективность равна $d/(d + 1)$. Среднее время задержки для кадра будет равно сумме времени ожидания в очереди внутри своей станции и дополнительных $N(d + 1)/2$ однобитовых интервалов, когда он попадет в начало своей внутренней очереди.

Двоичный обратный отсчет

Недостатком базового протокола бит-карты являются накладные расходы в 1 бит на станцию. Используя двоичный адрес станции, можно улучшить эффективность канала. Станция, желающая занять канал, объявляет свой адрес в виде битовой строки, начиная со старшего бита. Предполагается, что все адреса станций имеют одинаковую длину. Биты адреса в каждой позиции логически складываются (логическое ИЛИ). Мы будем называть этот протокол **протоколом с двоич-**

ным обратным отсчетом. Он используется в сети Datakit (Fraser, 1987). Неявно предполагается, что задержки распространения сигнала пренебрежимо малы, поэтому станции слышат утверждаемые номера практически мгновенно.

Во избежание конфликтов следует применить правило арбитража: как только станция с 0 в старшем бите адреса видит, что в суммарном адресе этот 0 заменился единицей, она сдается и ждет следующего цикла. Например, если станции 0010, 0100, 1001 и 1010 конкурируют за канал, то в первом битовом интервале они передают биты 0, 0, 1 и 1 соответственно. В этом случае суммарный первый бит адреса будет равен 1. Следовательно, станции с номерами 0010 и 0100 считаются проигравшими, а станции 1001 и 1010 продолжают борьбу.

Следующий бит у обеих оставшихся станций равен 0 — таким образом, обе продолжают. Третий бит равен 1, поэтому станция 1001 сдается. Победителем оказывается станция 1010, так как ее адрес наибольший. Выиграв торги, она может начать передачу кадра, после чего начнется новый цикл торгов. Схема протокола показана на рис. 4.7. Данный метод предполагает, что приоритет станции напрямую зависит от ее номера. В некоторых случаях такое жесткое правило может играть положительную, в некоторых — отрицательную роль.



Рис. 4.7. Протокол с двоичным обратным отсчетом. Прочерк означает молчание

Эффективность использования канала при этом методе составляет $d/(d + \log_2 N)$. Однако можно так хитро выбрать формат кадра, что его первое поле будет содержать адрес отправителя, тогда даже эти $\log_2 N$ бит не пропадут зря и эффективность составит 100 %.

Мок (Mok) и Уард (Ward) в 1979 году описали вариант протокола с обратным отсчетом, в котором использовался параллельный, а не последовательный интерфейс. Они также предложили использовать виртуальные номера станций. После каждой передачи станции, которая успешно послала кадр, присваивается виртуальный номер 0, тем самым дается возможность захвата канала станциями, которые молчат слишком долго. Например, если станции *C*, *H*, *D*, *A*, *G*, *B*, *E*, *F* имели приоритеты 7, 6, 5, 4, 3, 2, 1 и 0 соответственно, тогда при успешной передаче станции *D* она помещается в конец списка, получая номер 0. Приоритеты старших

станций C и H остаются неизменными (7 и 6), а приоритеты остальных станций увеличиваются на 1 (например, приоритет A был 4, а стал 5). Таким образом, на следующем цикле формируется такой список: C, H, A, G, B, E, F, D. Теперь станция D сможет получить доступ к каналу, только если он больше никому не нужен.

Двоичный обратный отсчет является примером простого, элегантного и эффективного протокола, который еще предстоит открыть заново разработчикам будущих сетей. Хочется надеяться, что когда-нибудь он займет свою нишу в сетевых технологиях.

Протоколы с ограниченной конкуренцией

Итак, мы рассмотрели две основные стратегии предоставления доступа к каналу в кабельных сетях: соревнование, как в CSMA, и бесконфликтные методы. Каждую стратегию можно оценить по двум важным параметрам: времени задержки при низкой загрузке канала и эффективности канала при большой загрузке. В условиях низкой загрузки конфликты (то есть чистая или дискретная системы ALOHA) предпочтительнее, так как время задержки в таких системах меньше. По мере роста загруженности канала системы со столкновениями становятся все менее привлекательными, поскольку возрастают накладные расходы, связанные с конфликтами. Для бесконфликтных протоколов справедливо обратное. При низкой нагрузке у них довольно высокое время задержки, но по мере увеличения нагрузки эффективность использования канала не уменьшается, как у конфликтных протоколов, а наоборот, возрастает.

Очевидно, было бы неплохо объединить лучшие свойства обеих стратегий и получить протокол, использующий разные стратегии при разной загруженности канала. Такие протоколы мы будем называть **протоколами с ограниченной конкуренцией**. Они в самом деле существуют, и их рассмотрением мы завершим изучение сетей с опросом носителя.

До сих пор мы рассматривали только симметричные протоколы коллективного доступа, в которых каждая станция пытается получить доступ к каналу с равной вероятностью p. Интересно, что производительность всей системы может быть улучшена при использовании асимметричного протокола, в котором станциям назначаются различные вероятности.

Прежде чем приступить к рассмотрению асимметричных протоколов, давайте кратко рассмотрим производительность в симметричном случае. Предположим, что k станций борются за доступ к каналу. Вероятность передачи каждой станции в каждый интервал времени равна p. Вероятность того, что какая-то станция успешно получит доступ к каналу на данный интервал времени, равна $kp(1-p)^{k-1}$. Чтобы найти оптимальное значение вероятности p, продифференцируем данное выражение по p, приравняем результат к нулю и решим полученное уравнение относительно p. В результате мы получим, что наилучшее значение p равно $1/k$. Заменяя в формуле p на $1/k$, получаем вероятность успеха при оптимальном значении p:

$$P[\textit{успех при оптимальной вероятности } p] = \left(\frac{k-1}{k}\right)^{k-1}. \qquad (4.4)$$

Зависимость этой вероятности от количества готовых станций графически показана на рис. 4.8. Для небольшого числа станций значение вероятности успеха является неплохим, однако как только количество станций достигает хотя бы пяти, вероятность снижается почти до асимптотической величины, равной $1/e$.



Рис. 4.8. Вероятность получения доступа к каналу в симметричном протоколе

Из рисунка очевидно, что вероятность получения доступа к каналу для какой-либо станции можно увеличить, только снизив конкуренцию за канал. Этим занимаются протоколы с ограниченной конкуренцией. Они сначала делят все станции на группы (необязательно непересекающиеся). Состязаться за интервал 0 разрешается только членам группы 0. Если кто-то из них выигрывает, он получает канал и передает по нему кадр. Если никто из них не хочет передавать или происходит столкновение, члены группы 1 состязаются за интервал 1, и т. д. При соответствующем разбиении на группы конкуренция за каждый интервал времени уменьшается, что увеличивает вероятность его успешного использования (см. левую часть графика).

Вопрос в том, как разбивать станции на группы. Прежде чем обсуждать общий случай, рассмотрим несколько частных случаев. В одном из крайних случаев в каждой группе будет по одной станции. Такое разбиение гарантирует полное отсутствие конфликтов, так как на каждый интервал времени будет претендовать только одна станция. Подобные протоколы уже рассматривались ранее (например, протокол с двоичным обратным отсчетом). Еще одним особым случаем является разбиение на группы, состоящие из двух станций. Вероятность того, что обе станции одновременно начнут передачу в течение одного интервала, равна p^2, и при малых значениях p этим значением можно пренебречь. По мере увеличения количества станций в группах вероятность столкновений будет возрастать, однако длина бит-карты, необходимой чтобы перенумеровать все группы, будет уменьшаться. Другим предельным случаем будет одна группа, в которую войдут все станции (то есть дискретная система ALOHA). Нам требуется механизм динамического разбиения станций на группы с небольшим количеством крупных групп при слабой загруженности канала и большом количестве мелких групп (может быть, даже состоящих из одной станции каждая), когда загруженность канала высока.

Протокол адаптивного прохода по дереву

Одним из простых способов динамического разбиения на группы является алгоритм, разработанный во время Второй мировой войны в армии США для проверки солдат на сифилис (Dorfman, 1943). Брался анализ крови у N солдат. Часть каждого образца помещалась в одну общую пробирку. Этот смешанный образец проверялся на наличие антител. Если антитела не обнаруживались, все солдаты в данной группе объявлялись здоровыми. В противном же случае группа делилась пополам, и каждая половина группы проверялась отдельно. Подобный процесс продолжался до тех пор, пока размер группы не уменьшался до одного солдата.

В компьютерной версии данного алгоритма (Capetanakis, 1979) станции рассматриваются в виде листьев двоичного дерева, как показано на рис. 4.9. В первом временном интервале состязания за право передачи участвуют все станции. Если кому-нибудь это удается, то на этом работа алгоритма заканчивается. Если же происходит столкновение, то ко второму этапу состязаний допускается только половина станций, а именно станции, относящиеся к узлу 2 дерева. Если одна из станций успешно захватывает канал, то следующее состязание устраивается для второй половины станций (относящихся к узлу 3 дерева). Если снова происходит конфликт, то к следующему интервалу времени среди состязающихся остается уже четверть станций, относящихся к узлу 4.

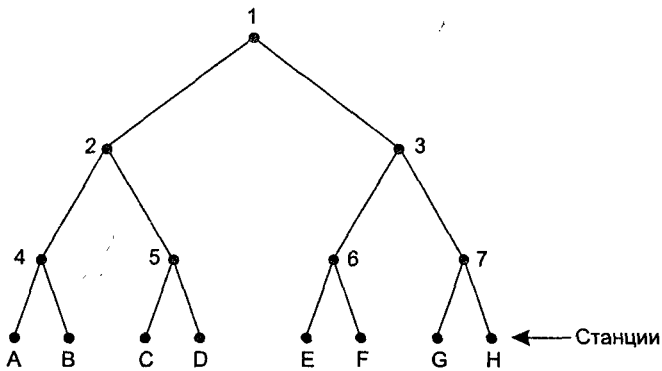


Рис. 4.9. Дерево из восьми станций

Таким образом, если столкновение происходит во время интервала 0, то все дерево сканируется на единичную глубину для поиска готовых станций. Каждый однобитовый слот ассоциируется с определенным узлом дерева. Если происходит столкновение, поиск продолжается для левого и правого дочерних узлов. Если количество станций, претендующих на передачу, равно нулю или единице, поиск в данном узле дерева прекращается.

При сильной загруженности канала вряд ли стоит начинать поиск готовой станции с узла 1, поскольку шансов, что всего одна станция из всех будет претендовать на канал, мало. По той же причине могут быть пропущены также узлы 2 и 3. А на каком уровне дерева следует начинать опрос в общем случае? Очевидно, что чем сильнее загруженность канала, тем на более низком уровне

дерева должен начинаться поиск готовых станций. Будем предполагать, что каждая станция довольно точно может оценить q (количество готовых на данный момент станций), отслеживая недавний трафик.

Пронумеруем уровни дерева на рис. 4.9 — узел 1 на уровне 0, узлы 2 и 3 на уровне 1 и т. д. Обратите внимание на то, что каждый узел на уровне i включает 2^{-i} часть от всех станций. Если q готовых станций распределены равномерно, то ожидаемое их число ниже узла на уровне i равно $2^{-i}q$. Интуитивно ясно, что оптимальным уровнем для начала поиска будет тот, на котором среднее число конкурирующих в интервале станций равно 1, то есть уровень, на котором $2^{-i}q = 1$. Отсюда $i = \log_2 q$.

Были разработаны многочисленные усовершенствования базового алгоритма — в частности, некоторые детали обсуждаются у Бертсекаса (Bertsekas) и Галлагера (Gallager) в издании 1992 года. Например, рассмотрим случай, при котором передавать хотят только станции G и H. На узле 1 произойдет конфликт, поэтому будет проверен узел 2. Он окажется пустым. Узел 3 проверять нет смысла, так как там гарантированно будет столкновение. (Нам известно, что под узлом 1 находятся 2 или более станций, а так как под узлом 2 нет ни одной станции, то все они должны быть под узлом 3.) Поэтому проверку узла 3 можно пропустить и сразу проверить узел 6. Поскольку под узлом 6 ничего не оказалось, то проверку узла 7 также можно пропустить и проверить уэел G.

Протоколы множественного доступа со спектральным разделением

Еще один метод распределения канала заключается в его разбиении на подканалы с помощью частотного, временного или смешанного разделения и динамического распределения подканалов по мере необходимости. Подобные схемы часто применяются в оптоволоконных локальных сетях, при этом возможна одновременная передача по каналу на разных длинах волн (то есть частотах). В данном разделе мы рассмотрим один такой протокол (Humblet и др., 1992).

Проще всего построить целиком оптическую локальную сеть с помощью коммутатора «пассивная звезда» (см. рис. 2.8). Смысл состоит в том, что два оптических волокна от каждой станции вплавляются в стеклянный цилиндр. По одному волокну свет попадает в цилиндр, а по другому он из цилиндра попадает в другое волокно. Свет, выходящий в цилиндр из одного волокна, освещает весь цилиндр и попадает во все выходящие из него волокна. Пассивные звезды могут объединять до нескольких сотен станций.

Для реализации одновременной передачи спектр делится на каналы (диапазоны длин волн), как показано на рис. 2.27. В протоколе **WDMA** (Wavelength Division Multiple Access — множественный доступ со спектральным разделением) каждой станции выделяется два канала. Узкий канал обеспечивает получение станцией управляющей информации, а широкий канал — передачу данных станцией.

Каждый канал делится на группы временных интервалов, как показано на рис. 4.10. Пусть количество интервалов в управляющем канале равно m, а количество интервалов в канале данных — $n + 1$, из которых n интервалов исполь-

зуются для данных, а последний интервал — для сообщения станцией о своем состоянии (главным образом о том, какие интервалы в обоих каналах свободны). В обоих каналах последовательность интервалов повторяется бесконечно, при этом интервал 0 помечается специальным образом, чтобы все могли его распознать. Все каналы синхронизируются одними глобальными часами.

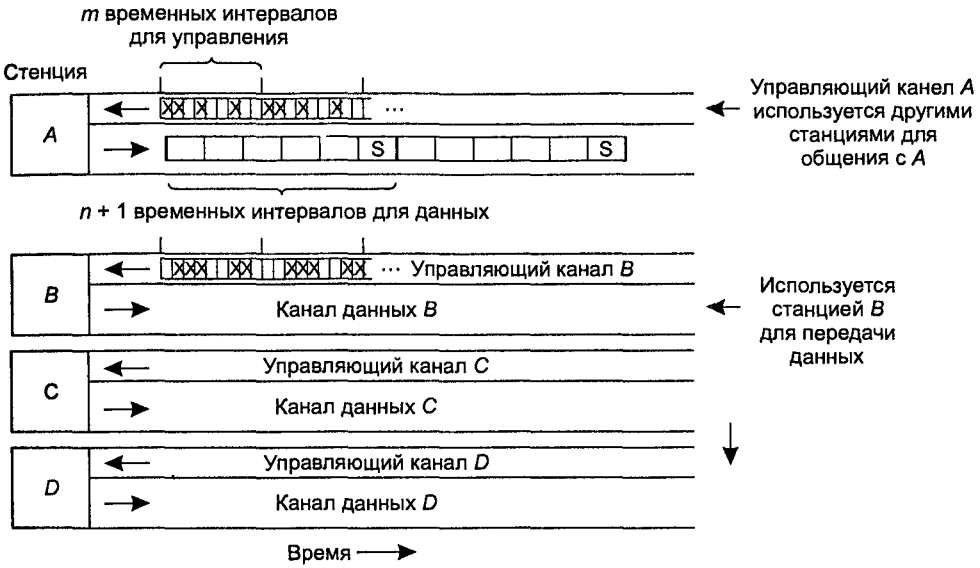


Рис. 4.10. Множественный доступ со спектральным разделением

Протоколом поддерживаются три класса трафика: 1) ориентированный на соединение с постоянной скоростью передачи данных, например, несжатое видео; 2) ориентированный на соединение с переменной скоростью передачи данных, например, передача файлов; 3) дейтаграммный трафик, как, например, UDP-пакеты. Для двух ориентированных на соединение протоколов идея заключается в том, что если станция *A* хочет связаться со станцией *B*, она сначала должна вставить кадр CONNECTION REQUEST (запрос на соединение) в свободный интервал управляющего канала *B*. Если станция *B* согласна, связь происходит по каналу данных *A*.

У каждой станции есть два передатчика и два приемника.

1. Приемник с фиксированной длиной волны для прослушивания своего управляющего канала.
2. Передатчик с настраиваемой длиной волны для передачи по управляющему каналу другой станции.
3. Передатчик с фиксированной длиной волны для передачи кадров данных.
4. Приемник с настраиваемой длиной волны для приема кадров данных.

Другими словами, каждая станция прослушивает свой управляющий канал для получения запросов, но должна настраиваться на длину волны передатчика

для получения данных. Настройка длины волны осуществляется с помощью интерферометра Фабри—Перо или Маха—Цандера, отсекающего все длины волн, которые не входят в требуемый диапазон.

Рассмотрим теперь, как станция *A* устанавливает канал связи класса 2 со станцией *B*, например, для переноса файлов. Сначала станция *A* настраивает свой приемник данных на канал данных станции *B* и ждет интервала состояния. В этом интервале сообщается, какие управляющие интервалы свободны в данный момент, а какие заняты. Так, на рис. 4.10 мы видим, что из восьми управляющих интервалов станции *B* свободны только интервалы 0, 4 и 5. Остальные интервалы заняты (помечены крестиками).

Станция *A* выбирает один из свободных интервалов, например, интервал 4, и вставляет в него свое сообщение с запросом на связь (CONNECTION REQUEST). Поскольку станция *B* постоянно прослушивает свой управляющий канал, она видит запрос и разрешает установление связи, назначая интервал 4 станции *A*. Это назначение объявляется в интервале состояния управляющего канала. Когда станция *A* видит это назначение, она понимает, что односторонняя связь установлена. Если станция *A* запрашивала двустороннюю связь, тот же алгоритм повторяется с той разницей, что теперь станция *B* запрашивает связь у станции *A*.

Может случиться так, что станции *A* и *C* одновременно будут пытаться захватить интервал 4 станции *B*. В этом случае интервал не достанется никому. Обе станции заметят это, отслеживая интервал состояния в управляющем канале станции *B*. После этого станции *A* и *C* подождут в течение случайного интервала времени и попытаются еще раз.

Таким образом, у каждой станции есть бесконфликтный способ послать другой станции короткое управляющее сообщение. Чтобы переслать файл, станция *A* посылает станции *B* управляющее сообщение примерно такого содержания: «Пожалуйста, просмотрите мои выходные данные в интервале 3. Там для вас есть кадр данных». Получив управляющее сообщение, станция *B* настраивает свой приемник на выходной канал станции *A*, чтобы прочитать информационный кадр. В зависимости от протокола более высокого уровня, станция *B* может использовать тот же самый механизм, чтобы при необходимости послать обратно подтверждение.

Проблема может возникнуть, если станции *A* и *C* одновременно соединены со станцией *B* и при этом обе вдруг предложат станции *B* посмотреть в интервал 3. Станции *B* придется выбрать случайным способом одну из станций, при этом другая передача будет потеряна.

Для трафика с постоянной скоростью передачи данных используется одна из разновидностей этого протокола. Когда станция *A* просит о предоставлении связи, она одновременно с запросом говорит нечто такое: «Можно я буду посылать вам кадр в интервале 3?» Если станция *B* может принять его (то есть интервал 3 свободен), устанавливается соединение с гарантированной пропускной способностью. В противном случае станция *A* может попытаться изменить свое предложение, выбрав другой свободный интервал.

Для трафика третьего типа (дейтаграмма) используется еще один вариант протокола. Вместо того чтобы вставлять запрос CONNECTION REQUEST в только что

найденный управляющий интервал (4), станция пишет сообщение DATA FOR YOU IN SLOT 3 («данные для вас в интервале 3»). Если станция *B* свободна во время следующего интервала 3, передача будет успешной. В противном случае кадр с данными теряется. При такой связи соединение не требуется.

Возможно несколько вариантов реализации всего протокола. Например, вместо того чтобы предоставлять каждой станции свой управляющий канал, можно распределить один управляющий канал между всеми станциями. Каждой станции предоставляется блок интервалов в каждой группе; таким образом, несколько виртуальных каналов эффективно мультиплексируются в одном физическом.

Кроме того, для каждой станции можно использовать только один настраиваемый приемник и один настраиваемый передатчик. Для этого каждый канал станции делится на m управляющих интервалов, за которыми будет следовать $n + 1$ информационный интервал. Недостатком такого подхода является то, что отправителям придется дольше ждать управляющий интервал, и последующие кадры данных тоже сдвинутся дальше, так как между ними располагаются управляющие кадры.

Разработано и реализовано множество WDMA-протоколов, различающихся в деталях. У некоторых один канал управления, у других — несколько. В некоторых задержка распространения учитывается, в некоторых — нет. Одни учитывают время настройки, другие — нет. Протоколы также различаются сложностью обработки, пропускной способностью и масштабируемостью. Благодаря большому числу используемых частот систему иногда называют **DWDM** (Dense Wavelength Division Multiplexing — мультиплексирование по длине волны высокой плотности). Дополнительную информацию см. в (Bogineni и др., 1993; Chen, 1994; Goralski, 2001; Levine and Akyildiz, 1995).

Протоколы беспроводных локальных сетей

С ростом доли переносных компьютеров и средств связи растет и потребность в их соединении с внешним миром. Даже самые первые мобильные телефоны могли связываться с другими телефонами. У первых портативных компьютеров такой способности не было, однако вскоре широкое распространение получили модемы. Чтобы перейти в подключенный режим (on-line), компьютер следовало присоединить к телефонной розетке. Необходимость подключения к фиксированной сети означала, что компьютеры были переносными, но не мобильными.

Чтобы называться мобильными, портативные компьютеры должны использовать для связи радио (или инфракрасные волны). В этом случае их владельцы смогут читать и посылать письма по электронной почте, находясь в автомобиле или на катере. Систему, состоящую из портативных компьютеров, общающихся по радио, можно рассматривать как беспроводную локальную сеть — мы уже обсуждали это в разделе «Беспроводные ЛВС: 802.11». Свойства таких сетей отличаются от свойств обычных локальных сетей, таким сетям требуются специальные протоколы управления доступом к носителю (MAC). В данном разделе мы познакомимся с некоторыми из этих протоколов. Подробнее о беспроводных локальных сетях можно прочитать в (Geier, 2002; O'Hara and Petrick, 1999).

Обычная конфигурация беспроводных локальных сетей подразумевает наличие офисного здания с заранее размещенными в нем базовыми станциями (называемыми также точками доступа). Все базовые станции соединены друг с другом медным проводом или оптоволоконным кабелем. Если мощность передатчиков базовых станций и переносных компьютеров настроена так, что диапазон приема составляет около 3–4 м, то каждая комната становится сотой, а все здание превращается в большую сотовую систему, подобную традиционной сотовой телефонной системе, описанной в главе 2. В отличие от обычной сотовой системы, у каждой соты в данном случае всего один канал, покрывающий весь доступный частотный диапазон и работающий со всеми станциями, находящимися в нем. Обычно пропускная способность такого канала составляет от 1 до 2 Мбит/с.

В дальнейших рассуждениях для простоты мы допустим, что все передатчики работают в неком фиксированном диапазоне. Когда приемник попадает в зону приема двух активных передатчиков, результирующий сигнал искажается и становится бесполезен, поэтому здесь мы больше не будем рассматривать системы типа CDMA. Важно понимать, что в некоторых беспроводных ЛВС не все станции находятся в пределах досягаемости друг друга, что приводит к возникновению разного рода сложностей. Кроме того, при установке беспроводных сетей в помещении присутствие стен между станциями может оказать сильнейшее влияние на эффективный диапазон каждой станции.

Можно наивно попытаться применить в локальных беспроводных сетях протокол CSMA (Carrier-Sense Multiple Access — множественный доступ с опросом несущей) — просто прослушивать эфир и осуществлять передачу только тогда, когда он никем не занят. Однако проблема заключается в том, что в действительности имеет значение интерференция на приемнике, а не на передатчике, поэтому этот протокол здесь не годится. Чтобы наглядно увидеть суть проблемы, рассмотрим рис. 4.11, где показаны четыре беспроводные станции. Для нашей проблемы не имеет значения, какая из них является базовой, а какая — переносной. Мощность передатчиков такова, что интерферировать могут только соседние станции, то есть *A* с *B*, *C* с *B* и *D*, но не с *A*.

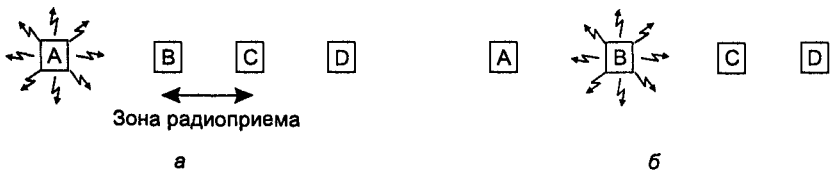


Рис. 4.11. Беспроводная локальная сеть: *A* передает (*а*); *B* передает (*б*)

Сначала рассмотрим, что происходит, когда станция *A* передает станции *B*, как изображено на рис. 4.11, *а*. Если станция *C* опрашивает канал, то она не будет слышать станцию *A*, поскольку та расположена слишком далеко, и может прийти к неверному выводу о том, что канал свободен и что можно посылать данные. Если станция *C* начнет передавать, она будет интерферировать со станцией *B* и исказит кадр, передаваемый станцией *A*. Проблема, заключающаяся

в том, что одна станция не может слышать возможного конкурента, поскольку конкурент расположен слишком далеко от нее, иногда называется **проблемой скрытой станции**.

Теперь рассмотрим обратную ситуацию: станция *B* передает станции *A*, как показано на рис. 4.11, *б*. Станция *C* при опросе канала слышит выполняемую передачу и может ошибочно предположить, что она не может передавать данные станции *D*, когда в действительности такая передача создала бы помехи только в зоне от станции *B* до станции *C*, где в данный момент не ведется прием. Такая ситуация иногда называется **проблемой засвеченной станции**.

Проблема заключается в том, что перед тем как начать передачу, станции необходимо знать, есть ли какая-нибудь активность вблизи приемника. Протокол CSMA же всего лишь может сообщить об активности вокруг станции, опрашивающей канал. В случае передачи по проводу все сигналы достигают всех станций, поэтому во всей системе одновременно только одна станция может вести передачу. В системе с использованием радиосвязи, радиус передачи и приема которой ограничен небольшими зонами, одновременно могут передавать несколько станций, если только они передают различным принимающим станциям, находящимся достаточно далеко друг от друга.

Можно представить себе этот вопрос и по-другому. Допустим, в офисном здании у каждого работника имеется беспроводной портативный компьютер. И вот, например, Линда хочет отправить сообщение Марку. Компьютер Линды контролирует то, что происходит вокруг него, и, не обнаружив никакой активности, начинает передачу. В комнате, где находится Марк, при этом может возникнуть коллизия из-за того, что кто-то третий передает ему данные одновременно с Линдой. Ее компьютер, естественно, этого обнаружить не может.

Протоколы MACA и MACAW

Одним из первых протоколов, разработанных для беспроводных локальных сетей, является **MACA** (Multiple Access with Collision Avoidance — множественный доступ с предотвращением столкновений) (Karn, 1990). Идея, лежащая в основе этого протокола, заключается в том, что отправитель заставляет получателя передать короткий кадр, чтобы окружающие станции могли услышать эту передачу и воздержаться от действий на время, требуемое для приема большого информационного кадра. Протокол MACA проиллюстрирован на рис. 4.12.

Рассмотрим ситуацию, в которой станция *A* передает станции *B*. Станция *A* начинает с того, что посылает станции *B* кадр RTS (Request To Send — запрос на передачу), как показано на рис. 4.12, *а*. Этот короткий кадр (30 байт) содержит длину кадра данных, который последует за ним. Затем станция *B* отвечает кадром CTS (Clear To Send — разрешение передачи), как показано на рис. 4.12, *б*. Кадр CTS также содержит длину информационного кадра (скопированную из кадра RTS). Приняв кадр CTS, станция *A* начинает передачу.

Теперь посмотрим, как реагируют станции, которые слышат передачу одного из этих кадров. Любая станция, которая слышит кадр RTS, находится близко к станции *A* и поэтому должна хранить молчание, пока кадр CTS не будет принят станцией *A*. Станции, слышащие кадр CTS, находятся вблизи от станции *B*, сле-

довательно, должны воздержаться от передачи, пока станция *B* не получит кадр данных, длину которого они могут узнать из кадра CTS.

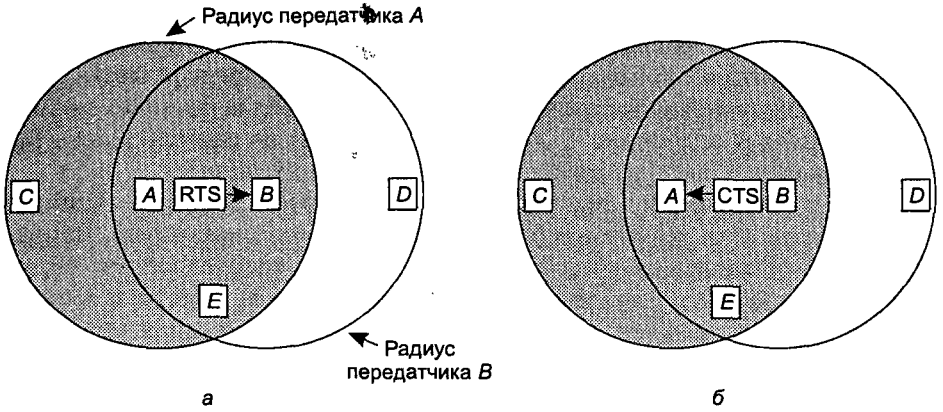


Рис. 4.12. Протокол MACA: станция *A* посылает кадр RTS станции *B* (*а*); станция *B* отвечает кадром CTS станции *A* (*б*)

На рис. 4.12 станция *C* находится в зоне станции *A*, но не входит в зону станции *B*. Поэтому она слышит кадр RTS, передаваемый станцией *A*, но не слышит кадр CTS, которым отвечает станция *B*. Поскольку она не интерферирует с кадром CTS, она не обязана воздерживаться от передачи в то время, пока пересылается информационный кадр. Станция *D*, напротив, находится близко от станции *B*, но далеко от станции *A*. Она не слышит кадра RTS, но слышит кадр CTS, а это означает, что она находится вблизи станции, собирающейся принять кадр с данными. Поэтому ей нельзя вести передачу, пока этот кадр не будет передан. Станция *E* слышит оба управляющих сообщения и так же, как и станция *D*, должна хранить молчание, пока не будет завершена передача информационного кадра.

Несмотря на все меры предосторожности, конфликты все равно могут произойти. Например, станции *B* и *C* могут одновременно послать кадры RTS станции *A*. При этом кадры столкнутся и не будут приняты. В этом случае передатчики, не услышав кадр CTS в установленный срок, ждут случайное время и после этого повторяют попытку. Алгоритм выдержки времени, использующийся в случае конфликта, называется двоичным экспоненциальным откатом, и мы изучим его, когда будем рассматривать сеть Ethernet.

Основываясь на изучении модели протокола MACA, Бхаргаван (Bharghavan) со товарищи в 1994 году осуществили тонкую настройку протокола MACA, чтобы улучшить его производительность. Новый протокол был назван **MACAW** (MACA for Wireless — MACA для беспроводных сетей). Для начала исследователи заметили, что без подтверждений на уровне передачи данных потерянные кадры не передавались повторно, пока их нехватку не обнаруживал транспортный уровень. Для решения этой проблемы был введен кадр подтверждения (ACK), которым получатель отвечал на каждый успешно принятый кадр дан-

ных. Кроме того, было использовано свойство протокола CSMA — станции научились прослушивать эфир и воздерживаться от передачи кадра RTS, если рядом уже кто-то передавал такой же кадр той же станции. Также было решено связать алгоритм выдержки времени (в случае конфликта) не с отдельной станцией, а с потоком данных, то есть с парой станций «источник — приемник». Это изменение протокола очень улучшило его. Наконец, был добавлен механизм обмена между станциями информацией о перегрузке. Кроме того, алгоритм выдержки времени в случае конфликта был несколько смягчен, что улучшило производительность системы.

Сеть Ethernet

Итак, мы в целом закончили обсуждение общих вопросов, касающихся протоколов распределения канала. Пришло время перейти к практическим приложениям, в частности, к локальным сетям. Как уже было сказано в разделе «Ethernet» (глава 1), IEEE в свое время разработал серию стандартов IEEE 802, описывающих локальные и региональные сети. Некоторые стандарты выжили, некоторые — нет (см. табл. 1.4). Люди, верящие в реинкарнацию, считают, что одним из членов Ассоциации стандартов IEEE является вновь родившийся Чарльз Дарвин, отбраковывающий слабые технологии. В общем-то, действительно выжили сильнейшие. Наиболее важны стандарты 802.3 (Ethernet) и 802.11 (беспроводные ЛВС). О 802.15 (Bluetooth) и 802.16 (беспроводные региональные сети) говорить всерьез пока не приходится. Впрочем, в пятом издании этой книги, вероятно, можно будет найти соответствующий анализ. В стандартах 802.3 и 802.11 физические уровни и уровни управления доступом к среде (MAC) различаются. Однако уже подуровни управления логическим соединением (LLC, определен стандартом 802.2) схожи, что позволяет организовать единое сопряжение с сетевым уровнем.

Мы уже представили в общих чертах Ethernet в разделе «Ethernet» (глава 1) и больше не будем повторять этот материал. Вместо этого мы сразу обратимся к рассмотрению таких технических деталей построения сетей Ethernet, как протоколы, а также новые технологии высокоскоростной (гигабитной) сети Ethernet. Так как Ethernet и IEEE 802.3 — это одно и то же (за исключением двух небольших деталей, которые мы вкратце обсудим), то многие используют оба названия. Мы тоже будем говорить то «Ethernet», то «IEEE 802.3». Дополнительную информацию, касающуюся Ethernet, можно найти в книгах (Breyer and Riley, 1999; Seifert, 1998; Spurgeon, 2000).

Кабели Ethernet

Поскольку само слово Ethernet связано с кабелем (ether — эфир, среда распространения сигнала), то именно с этого мы и начнем обсуждение. В сетях Ethernet обычно используются четыре типа кабеля, показанные в табл. 4.1.

Исторически сложилось так, что кабель **10Base5 («толстый Ethernet»)** стал первым носителем данных в сетях 802.3. Он внешне напоминал желтый садовый

шланг для поливки растений, и через каждые 2,5 м имелась маркировка мест подсоединения отводов. (Стандарт 802.3 не *требует*, чтобы цвет кабеля был именно желтым, но *рекомендует* это.) Соединения обычно делаются на основе **ответвителей «зуб вампира»** . Зуб ответвителя *чрезвычайно* аккуратно вводится на половину толщины внутренней жилы кабеля. Обозначение 10Base5 говорит о следующем: скорость работы — 10 Мбит/с, сигнал немодулированный (BASEband signaling), максимальная длина сегмента — 500 м. Итак, первая цифра названия — это скорость в мегабитах в секунду. Затем следует слово Base (иногда его пишут заглавными буквами — BASE), указывающее на то, что сигнал передается на базовой частоте, то есть без модуляции. Когда-то был разработан широкополосный вариант 10Broad36, но он так и не появился на мировом рынке и практически исчез. Наконец, если речь идет о коаксиальном кабеле, то после слова Base следует округленная до 100-метровых единиц максимальная длина сетевого сегмента.

Таблица 4.1. Наиболее распространенные типы кабелей Ethernet

Название	Тип	Максимальная длина сегмента	Узлов на сегмент	Преимущества
10Base5	Толстый коаксиальный	500 м	100	Первый кабель; ныне устарел
10Base2	Тонкий коаксиальный	185 м	30	Не нужны концентраторы
10Base-T	Витая пара	100 м	1024	Низкая цена
10Base-F	Оптоволокно	2000 м	1024	Лучший вариант при прокладке между зданиями

На смену толстому Ethernet пришел кабель типа **10Base2** (**«тонкий Ethernet»**), который, в отличие от шлангоподобного 10Base5, замечательно сгибается. Для отводов вместо зубастых ответвителей используются стандартные BNC-коннекторы, с помощью которых легко образуются Т-образные соединения. BNC-коннекторы проще в использовании и надежнее. Кроме того, они гораздо дешевле, и их удобнее монтировать. Недостатком является меньшая, чем у 10Base5, максимальная длина сегмента — 185 м, то есть на сегмент можно «посадить» не более 30 машин.

Обнаружение обрывов кабеля, чрезмерной длины сегментов, выхода из строя ответвителей и соединителей является основной проблемой обоих типов кабелей. Были разработаны специальные методики, позволяющие решить указанные задачи. Основная идея такова: по каналу передается импульс определенной формы. Если он встречает на своем пути какую-либо преграду или конец кабеля, образуется эхо, которое приходит обратно к отправителю. Тщательно измерив временной интервал между отправкой импульса и приходом эха, можно локализовать неисправность. Такой метод называется **измерением отраженного сигнала**.

Задачи поиска обрывов кабеля привели к созданию систем с измененной схемой подключения, в которой от каждой станции кабель идет к центральному **концентратору** (хабу), где станции соединяются друг с другом электроникой.

Обычно при этом используются традиционные для телефонии витые пары, главным образом потому, что большинство офисных помещений уже оборудовано соответствующей проводкой с большим запасом. Такая схема называется 10Base-T. Концентраторы не буферизуют входящий трафик. Далее мы обсудим улучшенные системы с использованием коммутаторов (switch), сохраняющих данные в собственном буфере.

Все три схемы подключения представлены на рис. 4.13. В 10Base5 **приемопередатчик** (трансивер) снаружи обжимает кабель так, чтобы его контактная игла соприкасалась с внутренней жилой. Он содержит электронные компоненты, позволяющие обнаруживать несущую и коллизии. При этом, обнаружив коллизию, приемопередатчик рассылает по всему кабелю специальный пакет, сообщающий о сбое. Таким образом гарантируется, что все остальные приемопередатчики тоже сообразят, что произошло столкновение.

В схеме 10Base5 ответвительный кабель соединяет приемопередатчик с интерфейсной платой (сетевой картой) компьютера. Длина этого кабеля может достигать 50 м. Он состоит из пяти независимых изолированных витых пар. Две витые пары используются для передачи данных от компьютера и к компьютеру. Еще по двум витым парам передаются управляющие сигналы. Пятая, не всегда используемая пара позволяет компьютеру управлять питанием приемопередатчика. Некоторые приемопередатчики могут обслуживать до восьми компьютеров, что уменьшает требуемое количество приемопередатчиков.

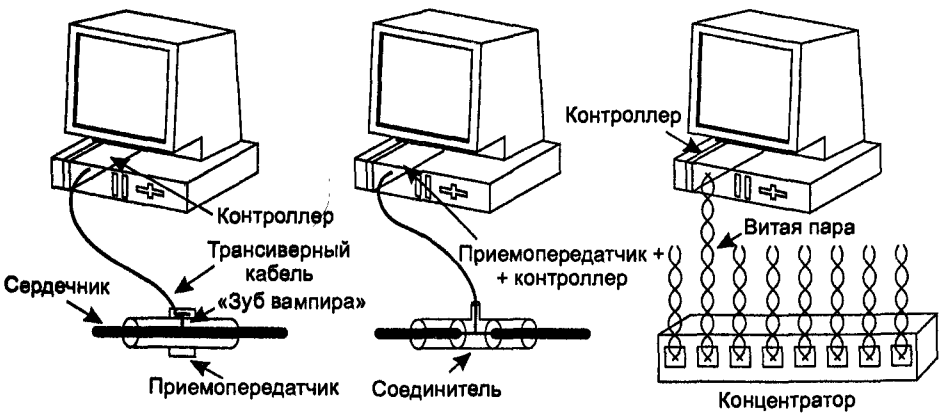


Рис. 4.13. Три типа кабельных соединений стандарта 802.3: 10Base5 (*а*); 10Base2 (*б*); 10Base-T (*в*)

Трансиверный кабель заканчивается на сетевой плате, установленной внутри компьютера. Сетевая карта содержит микросхему контроллера, посылающую кадры приемопередатчику и принимающую кадры у приемопередатчика. Контроллер отвечает за правильный формат сборки данных в кадры, а также за подсчет контрольных сумм исходящих кадров и проверку контрольных сумм входящих кадров. Некоторые контроллеры также управляют пулом буферов для приходящих кадров и очередью буферов передаваемых кадров, прямым доступом к памяти компьютера и другими вопросами, связанными с сетью.

В системе 10Base2 соединение с кабелем представляет собой обычный Т-образный пассивный BNC-коннектор. Электроника приемопередатчика располагается на плате контроллера, и у каждой станции обычно имеется свой приемопередатчик.

В системе 10Base-T никакого общего кабеля нет, есть только концентратор (ящик, набитый электроникой), к которому каждая машина подсоединена при помощи своего собственного кабеля. В такой конфигурации добавление и удаление станции осуществляются проще, а обрыв кабеля обнаруживается довольно легко. Недостатком системы 10Base-T является ограничение максимальной длины кабеля длиной 100 м, в лучшем случае 200 м, если используются высококачественные (категории 5) витые пары. Тем не менее системы 10Base-T быстро стали доминировать в сетях Ethernet благодаря легкости их установки и возможности использования уже существующей стандартной телефонной проводки. Более быстрая версия системы 10Base-T (100Base-T) будет обсуждаться далее.

Четвертый возможный вариант кабеля для сетей Ethernet называется **10Base-F** и построен на основе оптоволоконного кабеля. Такой кабель довольно дорог вследствие высокой цены соединителей и терминаторов, однако он обладает отличным отношением сигнал/шум и к тому же позволяет соединять сильно удаленные друг от друга концентраторы.

На рис. 4.14 изображены четыре способа прокладки кабелей в здании. На рис. 4.14, *а* единый кабель прокладывается от комнаты к комнате, и к нему подсоединяются все станции. На рис. 4.14, *б* показана магистраль, проходящая сквозь здание от фундамента до крыши, к которой на каждом этаже через специальные усилители (повторители) присоединены горизонтальные кабели. В некоторых зданиях в качестве горизонтальных кабелей устанавливаются тонкие 10Base2, а магистраль создается на основе толстого кабеля 10Base5. Наиболее распространенной топологией является дерево, показанное на рис. 4.14, *в*, поскольку при наличии нескольких путей между парами станций в сети может возникнуть интерференция сигналов.

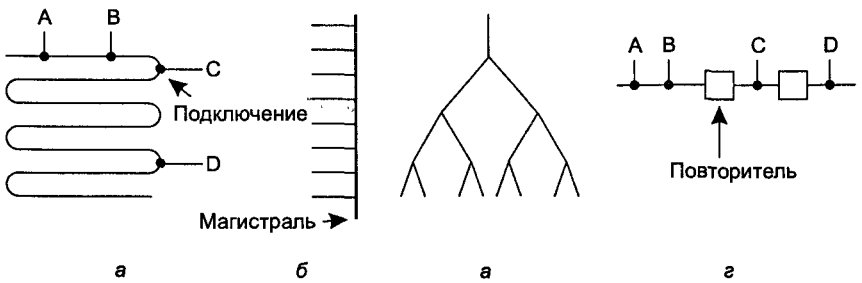


Рис. 4.14. Кабельная топология: линейная (а); магистраль (*б*); дерево (*в*); сегментированная (*г*)

Все версии стандарта 802.3 имеют ограничения по длине кабелей. Для построения сетей больших размеров несколько кабелей соединяются **повторителями**, как показано на рис. 4.14, *г*. Повторитель — это устройство физического

уровня. Он принимает, усиливает (регенерирует) и передает сигналы в обоих направлениях. С точки зрения программного обеспечения, ряд кабелей, соединенных повторителями, не отличается от сплошного кабеля (отличие заключается только во временной задержке, связанной с повторителями). Система может состоять из большого количества сегментов кабеля и повторителей, однако два приемопередатчика должны располагаться на расстоянии не более 2,5 км, и между ними должно быть не более четырех повторителей.

Манчестерский код

Ни в одной из версий Ethernet не применяется прямое двоичное кодирование бита 0 напряжением 0 В и бита 1 — напряжением 5В, так как такой способ приводит к неоднозначности. Если одна станция посылает битовую строку 00010000, то другая может интерпретировать ее как 10000000 или 01000000, так как они не смогут отличить отсутствие сигнала (0 В) от бита 0 (0 В). Можно, конечно, кодировать единицу положительным напряжением +1 В, а ноль — отрицательным напряжением −1 В. Но при этом все равно возникает проблема, связанная с синхронизацией передатчика и приемника. Разные частоты работы их системных часов могу привести к рассинхронизации и неверной интерпретации данных. В результате приемник может потерять границу битового интервала. Особенно велика вероятность этого в случае длинной последовательности нулей или единиц.

Таким образом, принимающей машине нужен способ однозначного определения начала, конца и середины каждого бита без помощи внешнего таймера. Это реализуется с помощью двух методов: **манчестерского кодирования** и **разностного манчестерского кодирования**. В манчестерском коде каждый временной интервал передачи одного бита делится на два равных периода. Бит со значением 1 кодируется высоким уровнем напряжения в первой половине интервала и низким — во второй половине, а нулевой бит кодируется обратной последовательностью — сначала низкое напряжение, затем высокое. Такая схема гарантирует смену напряжения в середине периода битов, что позволяет приемнику синхронизироваться с передатчиком. Недостатком манчестерского кодирования является то, что оно требует двойной пропускной способности линии по отношению к прямому двоичному кодированию, так как импульсы имеют половинную ширину. Например, для того чтобы отправлять данные со скоростью 10 Мбит/с, необходимо изменять сигнал 20 миллионов раз в секунду. Манчестерское кодирование показано на рис. 4.15, *б*.

Разностное манчестерское кодирование, показанное на рис. 4.15, *в*, является вариантом основного манчестерского кодирования. В нем бит 0 кодируется изменением состояния в начале интервала, а бит 1 — сохранением предыдущего уровня. В обоих случаях в середине интервала обязательно присутствует переход. Разностная схема требует более сложного оборудования, зато обладает хорошей защищенностью от шума. Во всех сетях Ethernet используется манчестерское кодирование благодаря его простоте. Высокий сигнал кодируется напряжением в +0,85 В, а низкий сигнал — −0,85 В, в результате чего постоянная составляющая напряжения равна 0 В. Разностное манчестерское кодирование

в Ethernet не используется, но используется в других ЛВС (например, стандарт 802.5, маркерное кольцо).

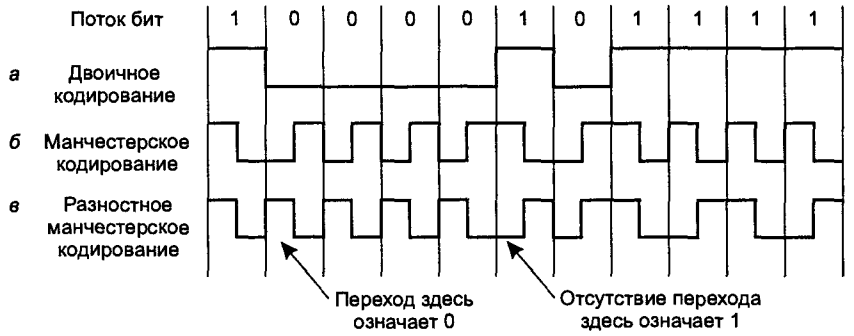


Рис. 4.15. Двоичное кодирование (*а*); манчестерское кодирование (*б*); разностное манчестерское кодирование (*в*)

Протокол подуровня управления доступом к среде в Ethernet

Исходная структура кадра, предложенная в свое время DIX (DEC, Intel, Xerox), показана на рис. 4.16. Каждый кадр начинается с поля *Preamble* (преамбула, заголовок) длиной 8 байт которое содержит последовательность 10101010. Манчестерское кодирование такой последовательности битов дает в результате меандр с частотой 10 МГц и длительностью 6,4 мкс, что позволяет получателю синхронизировать свои часы с часами отправителя. Далее до конца кадра они должны сохранять синхронизированное состояние за счет манчестерского кода, хранящего отметки границ битов.

Кадр содержит два адреса: получателя и отправителя. По стандарту разрешаются 2-байтовые и 6-байтовые адреса, однако параметры немодулированной передачи со скоростью 10 Мбит/с предусматривают только 6-байтовые адреса. Старший бит адреса получателя содержит 0 для обычных адресов и 1 для групповых получателей. Групповые адреса позволяют нескольким станциям принимать информацию от одного отправителя. Кадр, отправляемый групповому адресату, может быть получен всеми станциями, входящими в эту группу. Такой механизм называется **групповой рассылкой**. Если адрес состоит только из единиц, то кадр могут принять абсолютно все станции сети. Таким способом осуществляется **широковещание**. Разница между групповой рассылкой и широковещанием весьма существенна, поэтому еще раз повторим: кадр, предназначенный для групповой рассылки, посылается некоторой группе станций Ethernet; широковещательный же кадр получают абсолютно все станции сети. Групповая рассылка более избирательна, но требует некоторых усилий при управлении группами. Широковещание — это более грубая технология, но зато не требует никакой настройки групп.

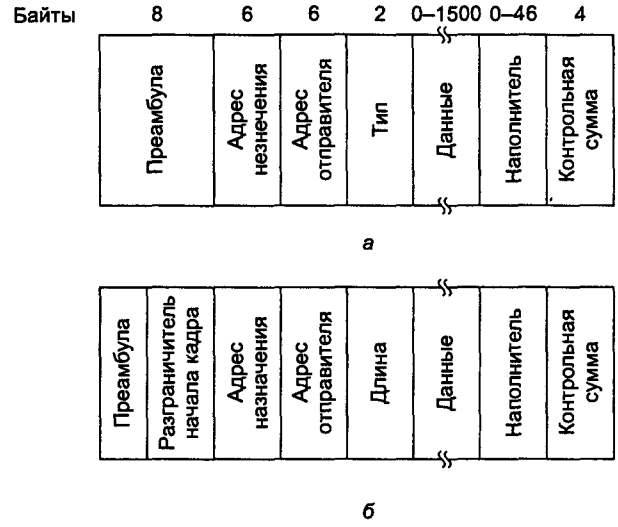


Рис. 4.16. Форматы кадров: DIX Ethernet (*а*); IEEE 802.3 (*б*)

Еще одной интересной особенностью адресации является использование 46-го бита (соседнего со старшим битом), позволяющего отличать локальные адреса от глобальных. Локальные адреса назначаются администратором каждой сети и не имеют смысла за ее пределами. Глобальные адреса, напротив, назначаются IEEE, и это гарантирует, что один и тот же глобальный адрес не используется двумя станциями. При 48 – 2 = 46 доступных битах может быть назначено около $7 \cdot 10^{13}$ глобальных адресов. Идея заключается в том, что каждая станция может быть однозначно идентифицирована по ее 48-битовому номеру. Найти по этому номеру саму станцию — задача сетевого уровня.

Затем следует поле *Type*, которое показывает приемнику, что делать с кадром. Дело в том, что одновременно на одной и той же машине могут работать несколько протоколов сетевого уровня, поэтому когда приходит кадр Ethernet, ядро должно понимать, какому протоколу его передать. Поле *Type* определяет процесс, который должен взять себе кадр.

Наконец, за полем *Type* следует поле данных, размер которого ограничен 1500 байтами. Такое ограничение было выбрано, в общем-то, произвольно в те времена, когда официально был закреплен стандарт DIX. При выборе ссылались на то, что приемопередатчику нужно довольно много оперативной памяти для того, чтобы хранить весь кадр. А память в том далеком 1978 году была еще очень дорогой. Соответственно, увеличение верхней границы размера поля данных привело бы к необходимости установки большего объема памяти, а значит, к удорожанию всего приемопередатчика.

Между тем, кроме верхней границы размера поля данных очень важна и нижняя граница. Поле данных, содержащее 0 байт, вызывает определенные проблемы. Дело в том, что когда приемопередатчик обнаруживает столкновение, он обрезает текущий кадр, а это означает, что отдельные куски кадров постоянно

блуждают по кабелю. Чтобы было легче отличить нормальные кадры от мусора, сети Ethernet требуется кадр размером не менее 64 байт (от поля адреса получателя до поля контрольной суммы включительно). Если в кадре содержится меньше 46 байт данных, в него вставляется специальное поле *Pad*, с помощью которого размер кадра доводится до необходимого минимума.

Другой (и даже более важной) целью установки ограничения размера кадра снизу является предотвращение ситуации, когда станция успевает передать короткий кадр раньше, чем его первый бит дойдет до самого дальнего конца кабеля, где он может столкнуться с другим кадром. Эта ситуация показана на рис. 4.17. В момент времени 0 станция *A* на одном конце сети посылает кадр. Пусть время прохождения кадра по кабелю равно τ. За мгновение до того, как кадр достигнет конца кабеля (то есть в момент времени $\tau - e$), самая дальняя станция *B* начинает передачу. Когда станция *B* замечает, что получает большую мощность, нежели передает сама, она понимает, что произошло столкновение. Тогда она прекращает передачу и выдает 48-битный шумовой сигнал, предупреждающий остальные станции. Примерно в момент времени 2τ отправитель замечает шумовой сигнал и также прекращает передачу. Затем он выжидает случайное время и пытается возобновить передачу.

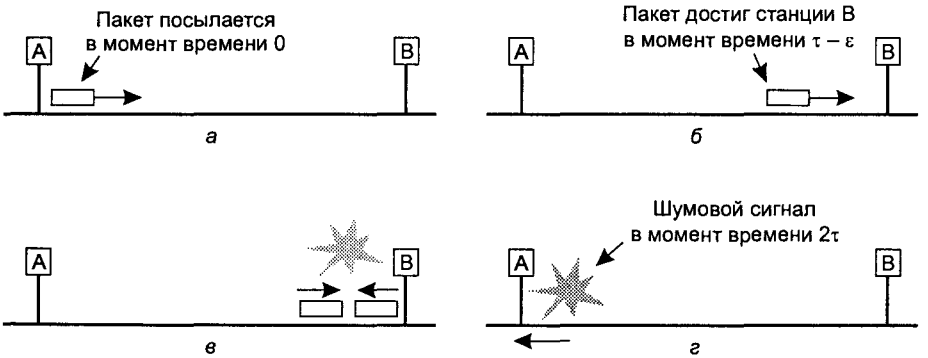


Рис. 4.17. Обнаружение столкновения может занять 2τ

Если размер кадра будет слишком маленьким, отправитель закончит передачу прежде, чем получит шумовой сигнал. В этом случае он не сможет понять, произошло это столкновение с его кадром или с каким-то другим, и, следовательно, может предположить, что его кадр был успешно принят. Для предотвращения такой ситуации все кадры должны иметь такую длину, чтобы время их передачи было больше 2τ. Для локальной сети со скоростью передачи 10 Мбит/с при максимальной длине кабеля в 2500 м и наличии четырех повторителей (требование спецификации 802.3) минимальное время передачи одного кадра должно составлять в худшем случае примерно 50 мкс, включая время на прохождение через повторитель, которое, разумеется, отлично от нуля. Следовательно, длина кадра должна быть такой, чтобы время передачи было по крайней мере не меньше этого минимума. При скорости 10 Мбит/с на передачу одного бита тратится 1000 нс, значит, минимальный размер кадра должен быть равен 500 бит. При

этом можно гарантировать, что система сможет обнаружить коллизии в любом месте кабеля. Из соображений большей надежности это число было увеличено до 512 бит или 64 байт. Кадры меньшего размера с помощью поля *Pad* искусственно дополняются до 64 байт.

По мере роста скоростей передачи данных в сети минимальный размер кадра должен увеличиваться, или должна пропорционально уменьшаться максимальная длина кабеля. Для 2500-метровой локальной сети, работающей на скорости 1 Гбит/с, минимальный размер кадра должен составлять 6400 байт. Или же можно использовать кадр размером 640 байт, но тогда надо сократить максимальное расстояние между станциями сети до 250 м. По мере приближения к гигабитным скоростям подобные ограничения становятся все более суровыми.

Последнее поле кадра стандарта Ethernet содержит контрольную сумму. По сути дела, это 32-битный хэш-код данных. Если какие-либо биты приняты неправильно (в результате шума в канале), контрольная сумма практически наверняка будет неправильной, и ошибка, таким образом, будет замечена. Алгоритм вычисления контрольной суммы основан на циклическом избыточном коде (CRC), который мы уже обсуждали в главе 3.

Когда институт IEEE принимал стандарт Ethernet, в формат кадра было внесено два изменения, как показано на рис. 4.16, *б*. Во-первых, преамбула была уменьшена до 7 байт, а последний байт был объявлен ограничителем кадра (*Start of Frame*) для совместимости со стандартами 802.4 и 802.5. Во-вторых, поле *Type* было преобразовано в *Length*. Конечно, приемник при этом потерял возможность определения действия над пришедшим кадром, но эта проблема была решена добавлением небольшого заголовка поля данных, предназначенного именно для подобной информации. Мы отдельно обсудим формат поля данных, когда будем рассматривать управление логическим соединением.

К сожалению, ко времени опубликования 802.3 по всему миру распространилось уже немало программного обеспечения и оборудования, соответствующих стандарту DIX Ethernet, поэтому изменение формата кадра было воспринято производителями и пользователями без энтузиазма. В 1997 году в IEEE поняли, что бороться бесполезно и бессмысленно и объявили оба стандарта приемлемыми. К счастью, все поля *Type*, использовавшиеся до 1997 года, имели значения больше 1500. Соответственно, любые номера, меньшие или равные 1500, можно было без сомнений интерпретировать как *Length*, а превышающие 1500 — как *Type*. Теперь IEEE может говорить, что все используют предложенный им стандарт, и при этом все пользователи и производители могут без зазрения совести продолжать работать точно так же, как и раньше.

Алгоритм двоичного экспоненциального отката

Рассмотрим, как осуществляется рандомизация периода ожидания после столкновения. Модель представлена на рис. 4.5. После возникновения коллизии время делится на дискретные интервалы, длительность которых равна максимальному времени кругового обращения сигнала (то есть его прохождения по кабелю в прямом и обратном направлениях), 2τ. Для удовлетворения потребностей Ethernet

при максимальном размере сети необходимо, чтобы один интервал составлял 512 битовых интервалов, или 51,2 мкс.

После первого столкновения каждая станция ждет или 0 или 1 интервал, прежде чем попытаться передавать опять. Если две станции столкнутся и выберут одно и то же псевдослучайное число, то они столкнутся снова. После второго столкновения каждая станция выбирает случайным образом 0, 1, 2 или 3 интервала из набора и ждет опять. При третьем столкновении (вероятность такого события после двойного столкновения равна $^1/_4$) интервалы будут выбираться в диапазоне от 0 до $2^3 - 1$.

В общем случае после i столкновений случайный номер выбирается в диапазоне от 0 до $2^i - 1$, и это количество интервалов станция пропускает. Однако после 10 столкновений подряд интервал рандомизации фиксируется на отметке 1023. После 16 столкновений подряд контроллер признает свое поражение и возвращает компьютеру ошибку. Дальнейшим восстановлением занимаются более высокие уровни.

Этот алгоритм, называемый **двоичным экспоненциальным алгоритмом отката**, был выбран для динамического учета количества станций, пытающихся осуществить передачу. Если выбрать интервал рандомизации равным 1023, то вероятность повторного столкновения будет пренебрежимо мала, однако среднее время ожидания составит сотни тактов, в результате чего среднее время задержки будет слишком велико. С другой стороны, если каждая станция будет выбирать время ожидания всего из двух вариантов, 0 и 1, то в случае столкновения сотни станций они будут продолжать сталкиваться снова и снова до тех пор, пока 99 из них не выберут 1, а одна станция — 0. Такого события можно будет ждать годами. Экспоненциально увеличивая интервал рандомизации по мере возникновения повторных столкновений, алгоритм обеспечивает небольшое время задержки при столкновении небольшого количества станций и одновременно гарантирует, что при столкновении большого числа станций конфликт будет разрешен за разумное время.

Как следует из приведенного описания, в системе CSMA/CD нет подтверждений. Поскольку простое отсутствие столкновений еще не гарантирует, что биты не были искажены всплесками шума в кабеле, для надежной связи необходимо проверять контрольную сумму и, если она правильная, посылать отправителю кадр подтверждения. С точки зрения протокола это будет еще один обычный кадр, которому так же придется бороться за канал, как и информационному кадру. Однако несложная модификация алгоритма борьбы за канал позволит ускорить пересылку подтверждения успешного приема кадра (Tokoro and Tamaru, 1977). Все, что для этого требуется, — зарезервировать первый временной интервал после успешной передачи кадра за получившей этот кадр станцией. К сожалению, стандарт не предусматривает такой возможности.

Производительность сети стандарта 802.3

Оценим производительность Ethernet в условиях большой постоянной загрузки, то есть когда k станций постоянно готовы к передаче. Строгий анализ алгоритма двоичного экспоненциального отката довольно сложен. Вместо этого мы после-

дуем за рассуждениями Меткалфа (Metcalfe) и Боггса (Boggs) (1976) и предположим, что вероятность повторной передачи в каждом интервале времени постоянна. Если каждая станция передает в течение одного интервала времени с вероятностью p, то вероятность того, что какой-либо станции удастся завладеть каналом, равна

$$A = kp(1-p)^{k-1}. \tag{4.5}$$

Значение A будет максимальным, когда $p = 1/k$. При k, стремящемся к бесконечности, A будет стремиться к $1/e$. Вероятность того, что период соревнования за канал будет состоять ровно из j интервалов, будет равна c, следовательно, среднее число интервалов борьбы за канал будет равно

$$\sum_{j=0}^{\infty} jA(1-A)^{j-1} = \frac{1}{A}.$$

Так как длительность каждого интервала времени равна 2τ, средняя продолжительность периода борьбы будет составлять $w = 2\tau/A$. При оптимальном значении вероятности p среднее количество интервалов за период борьбы никогда не будет превосходить e, таким образом, средняя продолжительность периода борьбы будет равна $2\tau e \approx 5{,}4\tau$.

Если среднее время передачи кадра составляет P секунд, то эффективность канала при его сильной загруженности будет равна

$$\text{Эффективность канала} = \frac{P}{P + 2\tau/A}. \tag{4.6}$$

В этой формуле мы видим, как максимальная длина кабеля влияет на производительность, и становится очевидным недостаток топологии сети, показанной на рис. 4.14, *а*. Чем длиннее кабель, тем более долгим становится период борьбы за канал. Из этих рассуждений становится понятно, почему стандарт Ethernet накладывает ограничение на максимальное расстояние между станциями.

Полезно переформулировать уравнение (4.6) в терминах длины кадра F, пропускной способности сети B, длины кабеля L и скорости распространения сигнала c для оптимального случая: e интервалов столкновений на кадр. При $P = F/B$ уравнение (4.6) примет вид

$$\text{Эффективность канала} = \frac{1}{1 + 2BLe/cF}. \tag{4.7}$$

Если второе слагаемое делителя велико, эффективность сети будет низкой. В частности, увеличение пропускной способности или размеров сети (произведение BL) уменьшит эффективность при заданном размере кадра. К сожалению, основные исследования в области сетевого оборудования нацелены именно на увеличение этого произведения. Пользователи хотят большой скорости при больших расстояниях (что обеспечивают, например, оптоволоконные региональные сети), следовательно, для данных приложений стандарт Ethernet будет не лучшим решением.

На рис. 4.18 показана зависимость эффективности канала от числа готовых станций для $2\tau = 51{,}2$ мкс и скорости передачи данных, равной 10 Мбит/с. Для

расчетов используется уравнение (4.7). При 64-байтном временном интервале 64-байтные кадры оказываются неэффективными, и это неудивительно. С другой стороны, если использовать кадры длиной 1024 байта, то при асимптотическом значении e периода состязания за канал, равном 64-байтовому интервалу, то есть 174 байтам, эффективность канала составит 85 %.

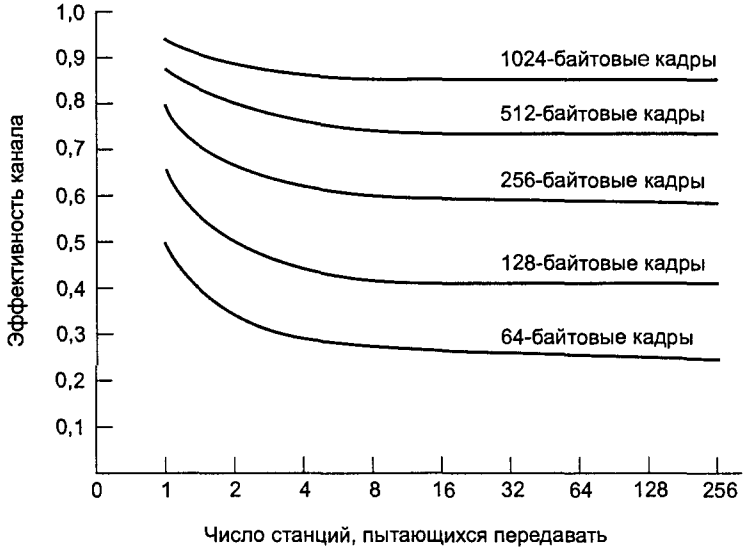


Рис. 4.18. Эффективность сетей стандарта 802.3 на скорости 10 Мбит/с с 512-битовыми интервалами времени

Чтобы определить среднее количество станций, готовых к передаче в условиях сильной загрузки, можно воспользоваться следующей грубой моделью. Каждый передаваемый кадр занимает канал на период состязания и на время передачи кадра, что составляет в сумме $P + w$ секунд. Таким образом, за секунду по каналу передается $1/(P + w)$ кадров. Если каждая станция формирует кадры со средней скоростью λ кадров в секунду, то при нахождении системы в состоянии k суммарная входная скорость k незаблокированных станций составит $k\lambda$ кадров в секунду. Поскольку в состоянии равновесия входная скорость должна быть равна выходной, мы можем приравнять эти две скорости и решить уравнение относительно k. (Обратите внимание: w является функцией от k.) Более подробный анализ см. в (Bertsekas and Gallager, 1992).

Следует отметить, что теоретическому анализу производительности сетей Ethernet (и других сетей) было посвящено много работ. Практически во всех этих исследованиях предполагается, что трафик подчиняется пуассоновскому распределению. Когда же исследователи рассмотрели реальные потоки данных, то обнаружилось, что сетевой трафик редко распределен по Пуассону, а чаще бывает автомодельным (Paxson and Floyd, 1994; Willinger и др., 1995). Это означает, что при увеличении периода усреднения трафик не сглаживается. Дисперсия среднего количества пакетов в каждую минуту часа не меньше дисперсии среднего

количества пакетов в каждую секунду минуты. Следствием этого открытия является то, что большинство моделей сетевого трафика не соответствуют реальной работе сетей и поэтому должны восприниматься весьма критически.

Коммутируемые сети Ethernet

При добавлении станций к Ethernet трафик сначала будет расти. Наконец, локальная сеть насытится. Одним из решений в данном случае является увеличение скорости передачи данных — например, переход с 10 Мбит/с на 100 Мбит/с. Однако доля мультимедийных данных в общем потоке становится все заметнее, и даже 100-мегабитные и гигабитные версии Ethernet могут перестать справляться со своей задачей.

К счастью, возможно не столь радикальное решение, а именно, коммутированная локальная сеть Ethernet, показанная на рис. 4.19. Сердцем системы является **коммутатор**, содержащий высокоскоростную плату, в слоты которой обычно вставляются от 4 до 32 контроллеров линий, в каждом из которых от одного до восьми разъемов. Чаще всего к разъему подключается витая пара 10Base-T, соединяющая коммутатор с единственным хостом.

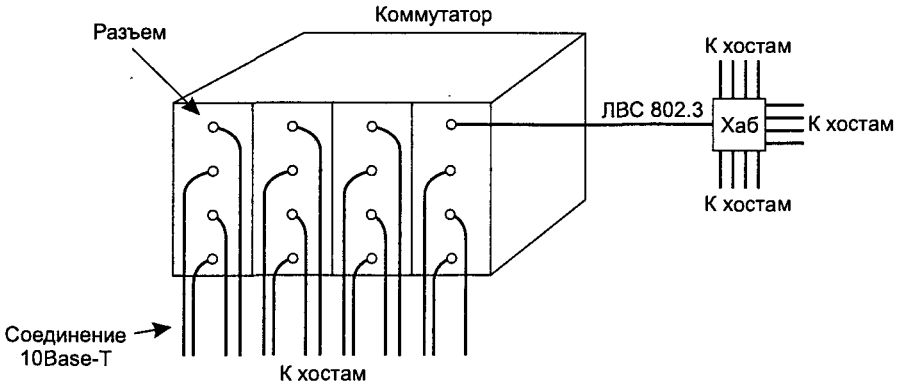


Рис. 4.19. Простой пример коммутируемой сети Ethernet

Когда станция хочет передать кадр Ethernet, она посылает стандартный кадр в коммутатор. Плата в коммутаторе, получив кадр, проверяет, не адресован ли этот кадр станции, подсоединенной к той же плате. Если да, то кадр пересылается ей. В противном случае кадр пересылается по объединительной плате карте, к которой подключена станция-получатель. Объединительная плата обычно работает на скорости в несколько гигабит в секунду с использованием собственного протокола.

Что произойдет, если две машины, присоединенные к одной и той же карте коммутатора, одновременно станут передавать кадры? Результат зависит от конструкции карты. Одним из вариантов может быть объединение всех портов карты вместе с образованием на карте небольшой локальной сети. Столкновения в такой сети обнаруживаются и обрабатываются так же, как и в любой другой сети

CSMA/CD — при помощи повторных передач кадров с использованием алгоритма двоичного экспоненциального отката. При использовании такого типа карт в каждый момент времени возможна передача только одной станции из подключенных к карте, но все карты могут передавать или принимать данные параллельно. При такой схеме коммутатора каждая карта образует свое **пространство столкновений**, независимое от других. Наличие только одной станции в пространстве столкновений исключает собственно столкновения и повышает производительность.

Возможна также и другая разновидность карт — с буферизацией данных, приходящих на каждый вход, в оперативной памяти карты. При этом все входные порты могут передавать и принимать кадры одновременно в дуплексном режиме, что далеко не всегда удается реализовать в моноканале с применением CSMA/CD. После приема кадра карта может проверить, кому он предназначается. Если адресатом является какой-то из портов текущей карты, то кадр сразу же туда и направляется. Если же нужно передать данные на порт другой карты, то это делается с помощью объединительной платы. При этом каждый порт обладает отдельным пространством коллизий, поэтому столкновения не возникают. Общая производительность системы может быть повышена на порядок по сравнению с 10Base5, поскольку в последней используется единое пространство столкновений.

Так как коммутатор ожидает на каждом входном порту кадры Ethernet, можно использовать некоторые из этих портов в качестве концентраторов. На рис. 4.19 порт в правом верхнем углу соединен не с одной станцией, а с 12-портовым концентратором. Прибывая в концентратор, кадры состязаются самым обычным образом, включая столкновения и двоичный откат. Удачливые кадры попадают в коммутатор и подвергаются там той же процедуре, что и все остальные кадры, то есть перенаправляются на нужные выходные линии через высокоскоростную объединяющую плату. Концентраторы дешевле коммутаторов, однако их быстрое удешевление означает лишь намечающуюся тенденцию к устареванию. Тем не менее, все еще существуют действующие концентраторы.

Быстрый Ethernet

Когда-то казалось, что 10 Мбит/с — это просто фантастически высокая скорость. Примерно так же воспринимали пользователи акустических модемов на 300 бит/с появление модемов со скоростью 1200 бит/с. Однако мир меняется очень быстро. В качестве одного из следствий закона Паркинсона («Работа занимает все отведенное на нее время») можно привести следующее правило: «Данные занимают всю предоставленную пропускную способность канала». Шутки шутками, но постоянно ощущалась и продолжает ощущаться нехватка скорости и ширины канала. Для решения этих проблем различными компаниями было разработано множество оптоволоконных кольцевых ЛВС. Одна из таких систем называется **FDDI** (Fiber Distributed Data Interface — распределенный интерфейс передачи данных по волоконно-оптическим каналам), а другая — **волоконный канал** (Fibre Channel). В двух словах их судьбу можно описать так: они обе использовались

в магистральных сетях, но ни одна из них так и не стала доступна непосредственно конечному пользователю. В обоих случаях управление станциями осуществлялось очень сложными методами, что привело к необходимости создания дорогостоящих, сложных микросхем. Урок из этой истории следовало бы извлечь только такой: KISS (Keep It Simple, Stupid! — «Не будь глупцом и упрощай», — одна из заповедей разработчика).

Так или иначе, отрицательный опыт — это тоже опыт, и после неудачной попытки создания волоконно-оптических локальных сетей возникло множество Ethernet-сетей, работающих со скоростями свыше 10 Мбит/с. Многим приложениям требовалась высокая пропускная способность, и поэтому появились 10-мегабитные ЛВС, связанные лабиринтами кабелей, повторителей, мостов, маршрутизаторов и шлюзов. Сетевым администраторам иногда казалось, что система держится еле-еле и может развалиться от любого прикосновения.

Вот при таких обстоятельствах в 1992 году институт IEEE начал пересмотр стандартов и дал заказ комитету 802.3 выработать спецификацию более быстрых сетей. Одно из предложений состояло в том, чтобы сохранить 802.3 без изменений и просто увеличить скорость работы. Другое заключалось в том, чтобы полностью его переделать, снабдить новым набором функций — например, обеспечить возможность передачи данных реального времени, оцифрованной речи. При этом предлагалось сохранить старое название стандарта (такой коммерческий прием). После некоторых колебаний комитет решил все-таки изменить лишь скорость работы 802.3, а все остальные параметры оставить прежними. Сторонники отвергнутого предложения поступили так, как в этой ситуации поступил бы любой человек, связанный с компьютерной индустрией: они хлопнули дверью, организовали собственный комитет и разработали свой стандарт (собственно, 802.12), который, впрочем, с треском провалился.

Комитет 802.3 решил продолжить линию старого доброго Ethernet по следующим трем соображениям.

1. Необходимость обратной совместимости с существующими ЛВС Ethernet.
2. Боязнь того, что в новом протоколе могут вскрыться неожиданные проблемы.
3. Желание успеть переделать стандарт до того, как изменится технология в целом.

Работа шла довольно быстро (по меркам комитета стандартизации), и уже в июне 1995 года официально объявили о создании стандарта **802.3u**. С технической точки зрения, в нем нет ничего нового по сравнению с предыдущей версией. Честнее было бы назвать это не новым стандартом, а расширением 802.3 (чтобы еще больше подчеркнуть обратную совместимость с ним). Поскольку жаргонное название «быстрый Ethernet» используется уже практически всеми, то и мы будем следовать этой моде.

Основная идея быстрого Ethernet довольно проста: оставить без изменений все старые форматы кадров, интерфейсы, процедуры и лишь уменьшить битовый интервал со 100 нс до 10 нс. Как это технически осуществить? Можно скопировать принцип, применяемый с 10Base-5 или 10Base-2, но в 10 раз уменьшить максимальную длину сегмента. Однако преимущества проводки 10Base-T

были столь неоспоримы, что практически все системы типа «быстрый Ethernet» в результате были построены именно на этом типе кабеля. Таким образом, в быстром Ethernet используются исключительно концентраторы (хабы) и коммутаторы; никаких моноканалов с ответвителями типа «зуб вампира» или с BNC-коннекторами здесь нет.

Однако некоторые технические решения все же необходимо было принять. Самый важный вопрос заключался в том, какие типы кабелей поддерживать. Одним из претендентов была витая пара категории 3. Основным аргументом в его пользу было то, что практически все западные офисы уже были оборудованы по крайней мере четырьмя витыми парами категории 3 (а то и лучше): они использовались в телефонных линиях, и их длина (до ближайшего телефонного щита) составляла не более 100 м. Иногда можно было встретить два таких кабеля. Таким образом, можно было установить в организациях быстрый Ethernet, и для этого не требовалось перекладывать кабель во всем здании. Это было очень существенно для многих.

Было во всем этом лишь одно неудобство: витые пары третьей категории не способны передавать сигналы, изменяющиеся со скоростью 200 мегабод (100 Мбит/с с манчестерским кодированием) на 100 м (именно таково максимальное расстояние между компьютером и концентратором, установленное стандартом для 10Base-T, см. табл. 4.1). Витые пары категории 5 с такой задачей справились бы без всяких проблем, а для оптоволокна это и вовсе смешная цифра. Надо было найти какой-то компромисс. Не мудрствуя лукаво, комитет 802.3 разрешил применять все три типа кабелей, как показано в табл. 4.2, с условием, что решения на основе витой пары третьей категории будут чуть живее и смогут обеспечить необходимую емкость канала.

Таблица 4.2. Основные типы кабелей для быстрых сетей Ethernet

Название	Тип	Длина сегмента	Преимущества
100Base-T4	Витая пара	100 м	Использование неэкранированной витой пары категории 3
100Base-TX	Витая пара	100 м	Полный дуплекс при 100 Мбит/с (витая пара категории 5)
100Base-FX	Оптоволокно	2000 м	Полный дуплекс при 100 Мбит/с; большая длина сегмента

В схеме **100Base-4T**, использующей витую пару категории 3, сигнальная скорость составляет 25 МГц, что лишь на 25 % больше, чем 20 МГц стандарта Ethernet (помните, что в манчестерском кодировании, показанном на рис. 4.15, требуется удвоенная частота). Чтобы достичь требуемой пропускной способности, в схеме 100Base-4T применяются четыре витые пары. Так как стандартные телефонные кабели в течение десятилетий как раз и состояли из четырех витых пар, большинство организаций могут запросто ими воспользоваться в новых целях. Для этого, правда, придется оставить весь офис без телефонов, но это, согласитесь, не самая высокая цена за возможность быстрее получать электронную почту.

Из четырех витых пар одна всегда направляется на концентратор, одна — от концентратора, а две оставшиеся переключаются в зависимости от текущего направления передачи данных. Для достижения скорости 100 Мбит/с от манчестерского кодирования пришлось отказаться, однако, учитывая сегодняшние тактовые частоты и небольшие расстояния между станциями ЛВС, без него вполне можно обойтись. Кроме того, по линии посылаются троичные сигналы, то есть 0, 1 или 2. При использовании трех витых пар в направлении передачи данных это означало передачу 1 из 27 возможных символов за один такт, то есть 4 бита плюс некоторая избыточность, что при тактовой частоте в 25 МГц как раз и составляет требуемые 100 Мбит/с. Кроме того, есть еще обратный канал, работающий на скорости 33,3 Мбит/с по оставшейся витой паре. Такая схема, известная как **8B/6T** (8 битов в виде 6 троичных цифр), вряд ли получит приз за элегантность, но зато она работает на уже имеющихся в каждом офисе кабелях.

Устройство системы **100Base-TX**, использующей витые пары категории 5, проще, так как кабели этого типа могут работать с сигналами на частоте 125 МГц. Поэтому для каждой станции используются только две витые пары: одна к концентратору, другая от него. Прямое битовое кодирование не используется. Вместо него имеется специальная система кодирования, называемая **4B/5B**. Она является последователем FDDI и совместима с ней. Каждая группа из четырех тактовых интервалов, каждый из которых содержит один из двух сигнальных значений, образует 32 комбинации. 16 из них используются для передачи четырехбитных групп 0000, 0001, 0010, ..., 1111. Оставшиеся 16 используются для служебных целей — например, для маркировки границ кадров. Используемые комбинации тщательно подбирались с целью обеспечения достаточного количества передач для поддержки синхронизации с тактогенератором. Система 100Base-TX является полнодуплексной, станции могут передавать на скорости 100 Мбит/с и одновременно принимать на той же скорости. Зачастую кабели 100Base-TX и 100Base-T4 называют просто **100Base-T**.

Последний вариант, **100Base-FX**, использует два оптических многомодовых кабеля, по одному для передачи в каждом направлении, то есть также полный дуплекс на скорости 100 Мбит/с в каждом направлении. Кроме того, расстояние между станциями при этом может достигать 2 км.

В 1997 году, пойдя навстречу потребностям пользователей, комитет 802 ввел в стандарт новый тип кабеля, 100Base-T2, позволяющий организовать работу быстрой сети Ethernet на основе двух витых пар третьей категории. Однако для отработки довольно сложного алгоритма кодирования потребовался соответствующий цифровой сигнальный процессор, что сделало данное расширение довольно дорогим удовольствием. Пока что вследствие своей чрезвычайной сложности, цены и перехода многих организаций на витые пары пятой категории 100Base-T2 используется мало.

В системах 100Base-T могут применяться два типа концентраторов, как показано на рис. 4.19. В концентраторе все входящие линии (по крайней мере, все линии, подключенные к одной плате) логически соединены с образованием единой области столкновений. Применяются все стандартные алгоритмы, включая двоичный откат; таким образом, система работает аналогично старой доброй сети

Ethernet. В частности, в каждый момент времени может передавать только одна станция. Другими словами, концентраторы работают с полудуплексными соединениями.

В коммутаторе все приходящие кадры буферируются на плате, обслуживающей линию. Хотя при таком подходе карты и сам концентратор становятся дороже, все станции могут передавать (и принимать) кадры одновременно, что значительно повышает суммарную производительность системы, часто на порядок и более. Хранящиеся в буфере кадры обмениваются между картой источника и картой приемника по высокоскоростной объединительной плате. Объединительная плата никак не стандартизована. В этом нет необходимости, так как она спрятана глубоко внутри коммутатора. Кроме того, это предоставляет свободу производителям коммутаторов, которые будут состязаться, пытаясь создать все более быструю плату. Поскольку оптоволоконные кабели системы 100Base-FX слишком длинны для алгоритма столкновений, стандартного для сети Ethernet, их следует присоединять к буферированным коммутируемым концентраторам, так чтобы каждый кабель представлял собой отдельную область столкновений.

В заключение разговора о быстром Ethernet стоит отметить, что практически все коммутаторы могут поддерживать и 10-, и 100-мегабитные станции. Это сделано для пущего удобства тех, кто постепенно обновляет аппаратуру. По мере появления новых 100-мегабитных станций нужно просто покупать необходимое количество плат обслуживания линий и вставлять их в коммутатор. В стандарт включена возможность автоматического выбора станциями оптимальной скорости (10 или 100) и полудуплексного или полнодуплексного режима. Большая часть аппаратуры быстрого Ethernet использует эту функцию для самонастройки.

Гигабитная сеть Ethernet

Не успело еще, как говорится, обсохнуть молоко на губах только что родившегося стандарта быстрого Ethernet, как комитет 802 приступил к работе над новой версией (1995). Ее почти сразу окрестили **гигабитной сетью Ethernet**, а в 1998 году новый стандарт был уже ратифицирован IEEE под официальным названием 802.3z. Тем самым разработчики подчеркнули, что это последняя разработка в линейке 802.3 (если только кто-нибудь в срочном порядке не придумает называть стандарты, скажем, 802.3ы. Между прочим, Бернард Шоу предлагал расширить английский алфавит и включить в него, в частности, букву «ы». — *Примеч. перев.*). Далее мы обсудим некоторые ключевые свойства гигабитного Ethernet. Более подробную информацию можно найти в (Seifert, 1998).

Главные предпосылки создания 802.3z были те же самые, что и при создании 802.3u, — повысить в 10 раз скорость, сохранив обратную совместимость со старыми сетями Ethernet. В частности, гигабитный Ethernet должен был обеспечить дейтаграммный сервис без подтверждений как при односторонней, так и при групповой передаче. При этом необходимо было сохранить неизменными 48-битную схему адресации и формат кадра, включая нижние и верхние ограничения его размера. Новый стандарт удовлетворил всем этим требованиям.

Гигабитные сети Ethernet строятся по принципу «точка — точка», в них не применяется моноканал, как в исходном 10-мегабитном Ethernet, который теперь, кстати, величают **классическим Ethernet**. Простейшая гигабитная сеть, показанная на рис. 4.20, *а*, состоит из двух компьютеров, напрямую соединенных друг с другом. В более общем случае, однако, имеется коммутатор или концентратор, к которому подсоединяется множество компьютеров, возможна также установка дополнительных коммутаторов или концентраторов (рис. 4.20, *б*). Но в любом случае к одному кабелю гигабитного Ethernet всегда присоединяются два устройства, ни больше, ни меньше.

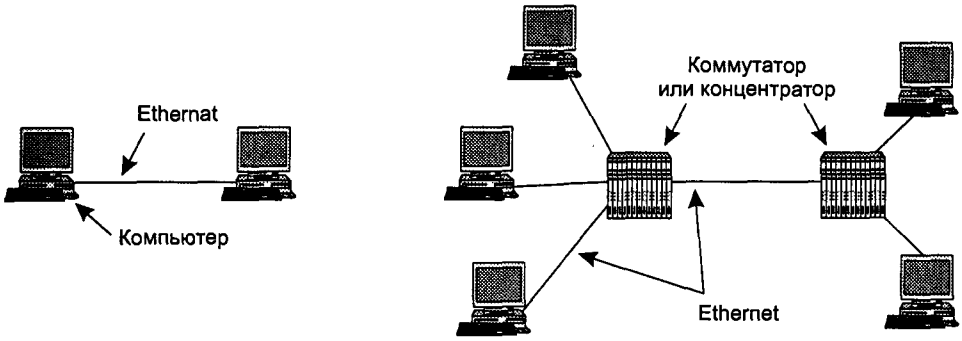


Рис. 4.20. Сеть Ethernet, состоящая из двух станций (*а*); сеть Ethernet, состоящая из множества стенций (*б*)

Гигабитный Ethernet может работать в двух режимах: полнодуплексном и полудуплексном. «Нормальным» считается полнодуплексный, при этом трафик может идти одновременно в обоих направлениях. Этот режим используется, когда имеется центральный коммутатор, соединенный с периферийными компьютерами или коммутаторами. В такой конфигурации сигналы всех линий буферизируются, поэтому абоненты могут отправлять данные, когда им вздумается. Отправитель не прослушивает канал, потому что ему не с кем конкурировать. На линии между компьютером и коммутатором компьютер — это единственный потенциальный отправитель; передача произойдет успешно даже в том случае, если одновременно с ней ведется передача со стороны коммутатора (линия полнодуплексная). Так как конкуренции в данном случае нет, протокол CSMA/CD не применяется, поэтому максимальная длина кабеля определяется исключительно мощностью сигнала, а вопросы времени распространения шумового всплеска здесь не встают. Коммутаторы могут работать на смешанных скоростях; более того, они автоматически выбирают оптимальную скорость. Самонастройка поддерживается так же, как и в быстром Ethernet.

Полудуплексный режим работы используется тогда, когда компьютеры соединены не с коммутатором, а с концентратором. Хаб не буферизирует входящие кадры. Вместо этого он электрически соединяет все линии, симулируя моноканал обычного Ethernet. В этом режиме возможны коллизии, поэтому применяет-

ся CSMA/CD. Поскольку кадр минимального размера (то есть 64-байтный) может передаваться в 100 раз быстрее, чем в классической сети Ethernet, максимальная длина сегмента должна быть соответственно уменьшена в 100 раз. Она составляет 25 м — именно при таком расстоянии между станциями шумовой всплеск гарантированно достигнет отправителя до окончания его передачи. Если бы кабель имел длину 2500 м, то отправитель 64-байтного кадра при 1 Гбит/с успел бы много чего наделать даже за то время, пока его кадр прошел только десятую часть пути в одну сторону, не говоря уже о том, что сигнал должен еще и вернуться обратно.

Комитет разработчиков стандарта 802.3z совершенно справедливо заметил, что 25 м — это неприемлемо малая длина, и ввел два новых свойства, позволивших расширить радиус сегментов. Первое называется **расширением носителя**. Заключается это расширение всего-навсего в том, что аппаратура вставляет собственное поле заполнения, растягивающее нормальный кадр до 512 байт. Поскольку это поле добавляется отправителем и изымается получателем, то программному обеспечению нет до него никакого дела. Конечно, тратить 512 байт на передачу 46 байт — это несколько расточительно с точки зрения эффективности использования пропускной способности. Эффективность такой передачи составляет всего 9 %.

Второе свойство, позволяющее увеличить допустимую длину сегмента, — это **пакетная передача кадров**. Это означает, что отправитель может посылать не единичный кадр, а пакет, объединяющий в себе сразу много кадров. Если полная длина пакета оказывается менее 512 байт, то, как в предыдущем случае, производится аппаратное заполнение фиктивными данными. Если же кадров, ждущих передачу, хватает на то, чтобы заполнить такой большой пакет, то работа системы оказывается очень эффективной. Такая схема, разумеется, предпочтительнее расширения носителя. Эти методы позволили увеличить максимальную длину сегмента до 200 м, что, наверное, для организаций уже вполне приемлемо.

Трудно представить себе организацию, которая потратила бы немало усилий и средств на установку плат для высокопроизводительной гигабитной сети Ethernet, а потом соединила бы компьютеры концентраторами, симулирующими работу классического Ethernet со всеми его коллизиями и прочими проблемами. Концентраторы, конечно, дешевле коммутаторов, но интерфейсные платы гигабитного Ethernet все равно относительно дороги, поэтому экономия на покупке концентратора вместо коммутатора себя не оправдывает. Кроме того, это резко снижает производительность, и становится вообще непонятно, зачем было тратить деньги на гигабитные платы. Однако обратная совместимость — это нечто священное в компьютерной индустрии, поэтому, несмотря ни на что, в 802.3z подобная возможность предусматривается.

Гигабитный Ethernet поддерживает как медные, так и волоконно-оптические кабели, что отражено в табл. 4.3. Работа на скорости 1 Гбит/с означает, что источник света должен включаться и выключаться примерно раз в наносекунду. Светодиоды просто не могут работать так быстро, поэтому здесь необходимо применять лазеры. Стандартом предусматриваются две операционных длины

волны: 0,85 мкм (короткие волны) и 1,3 мкм (длинные). Лазеры, рассчитанные на 0,85 мкм, дешевле, но не работают с одномодовыми кабелями.

Таблица 4.3. Кабели гигабитного Ethernet

Название	Тип	Длина сегмента	Преимущества
1000Base-SX	Оптоволокно	550 м	Многомодовое волокно (50, 62,5 мкм)
1000Base-LX	Оптоволокно	5000 м	Одномодовое (10 мкм) или многомодовое (50, 62,5 мкм) волокно
1000Base-CX	2 экранированные витые пары	25 м	Экранированная витая пара
1000Base-T	4 неэкранированные витые пары	100 м	Стандартная витая пара 5-й категории

Официально допускается использование трех диаметров волокна: 10, 50 и 62,5 мкм. Первое предназначено для одномодовой передачи, два других — для многомодовой. Не все из шести комбинаций являются разрешенными, а максимальная длина сегмента зависит как раз от выбранной комбинации. Числа, приведенные в табл. 4.3, — это наилучший случай. В частности, пятикилометровый кабель можно использовать только с лазером, рассчитанным на длину волны 1,3 мкм и работающим с 10-микрометровым одномодовым волокном. Такой вариант, видимо, является наилучшим для магистралей разного рода кампусов и производственных территорий. Ожидается, что он будет наиболее популярным несмотря на то, что он самый дорогой.

1000Base-CX использует короткий экранированный медный кабель. Проблема в том, что его поджимают конкуренты как сверху (1000Base-LX), так и снизу (1000Base-T). В результате сомнительно, что он завоюет широкое общественное признание.

Наконец, еще один вариант кабеля — это пучок из четырех неэкранированных витых пар. Поскольку такая проводка существует почти повсеместно, то, похоже, это будет «гигабитный Ethernet для бедных».

Новый стандарт использует новые правила кодирования сигналов, передающихся по оптоволокну. Манчестерский код при скорости передачи данных 1 Гбит/с потребовал бы скорости изменения сигнала в 2 Гбод. Это слишком сложно и занимает слишком большую долю пропускной способности. Вместо манчестерского кодирования применяется схема, называющаяся **8B/10B**. Как нетрудно догадаться по названию, каждый байт, состоящий из 8 бит, кодируется для передачи по волокну десятью битами. Поскольку возможны 1024 результирующих кодовых слова для каждого входящего байта, данный метод дает некоторую свободу выбора кодовых слов. При этом принимаются в расчет следующие правила:

- ни одно кодовое слово не должно иметь более четырех одинаковых битов подряд;
- ни в одном кодовом слове не должно быть более шести нулей или шести единиц.

Почему именно такие правила? Во-первых, они обеспечивают достаточное количество изменений состояния в потоке данных, необходимое для того, чтобы приемник оставался синхронизированным с передатчиком. Во-вторых, количество нулей и единиц стараются примерно выровнять. К тому же многие входящие байты имеют два возможных кодовых слова, ассоциированных с ними. Когда кодирующее устройство имеет возможность выбора кодовых слов, оно, вероятно, выберет из них то, которое сравняет число нулей и единиц.

Ссбалансированному количеству нулей и единиц потому придается такое значение, что необходимо держать постоянную составляющую сигнала на как можно более низком уровне. Тогда она сможет пройти через преобразователи без изменений. Люди, занимающиеся computer science, не в восторге от того, что преобразовательные устройства диктуют те или иные правила кодирования сигналов, но жизнь есть жизнь.

Гигабитный Ethernet, построенный на 1000Base-T, использует иную схему кодирования, поскольку изменять состояние сигнала в течение 1 нс для медного кабеля затруднительно. Здесь применяются 4 витые пары категории 5, что дает возможность параллельно передавать 4 символа. Каждый символ кодируется одним из пяти уровней напряжения. Таким образом, один сигнал может означать 00, 01, 10 или 11. Есть еще специальное, служебное значение напряжения. На одну витую пару приходится 2 бита данных, соответственно, за один временной интервал система передает 8 бит по 4 витым парам. Тактовая частота равна 125 МГц, что позволяет работать со скоростью 1 Гбит/с. Пятый уровень напряжения был добавлен для специальных целей — кадрирования и управления.

1 Гбит/с — это довольно много. Например, если приемник отвлечется на какое-то дело в течение 1 мс и при этом забудет/не успеет освободить буфер, это означает, что он «проспит» примерно 1953 кадра. Может быть и другая ситуация: один компьютер выдает данные по гигабитной сети, а другой принимает их по классическому Ethernet. Вероятно, первый быстро завалит данными второго. В первую очередь переполнится буфер обмена. Исходя из этого было принято решение о внедрении в систему контроля потока (так было и в быстром Ethernet, хотя эти системы довольно сильно различаются).

Для реализации контроля потока одна из сторон посылает служебный кадр, сообщающий о том, что второй стороне необходимо приостановиться на некоторое время. Служебные кадры — это, на самом деле, обычные кадры Ethernet, в поле *Type* которых записано 0x8808. Первые два байта поля данных — командные, а последующие, по необходимости, содержат параметры команды. Для контроля потока используются кадры типа PAUSE, причем в качестве параметра указывается продолжительность паузы в единицах времени передачи минимального кадра. Для гигабитного Ethernet такая единица равна 512 нс, а паузы могут длиться до 33,6 мс.

Гигабитный Ethernet был стандартизован, и комитет 802 заскучал. Тогда IEEE предложил ему начать работу над 10-гигабитным Ethernet. Начались долгие попытки найти в английском алфавите какую-нибудь букву после z. Когда стало очевидно, что такой буквы нет в природе, от старого подхода решено было отказаться и перейти к двухбуквенным индексам. Так в 2002 году появился стандарт 802.3ae. Судя по всему, появление 100-гигабитного Ethernet уже тоже не за горами.

Стандарт IEEE 802.2: протокол LLC

Теперь, пожалуй, нужно вернуться назад и сравнить то, что мы изучили в этой главе, с материалом предыдущей главы. Из главы 3 мы узнали, как между двумя машинами по ненадежной линии с помощью различных протоколов передачи данных могла быть установлена надежная связь. Эти протоколы обеспечивали защиту от ошибок (с помощью подтверждений) и управление потоком (при помощи протокола скользящего окна).

В этой главе, напротив, о надежной связи не было сказано ни слова. Единственное, что предлагает набор стандартов 802, — это дейтаграммный сервис. В некоторых случаях этого оказывается вполне достаточно. Например, при передаче IP-пакетов не требуется и даже не ожидается никакой гарантии. IP-пакет может быть просто помещен в информационное поле кадра стандарта 802 и послан в нужном направлении. Если он потеряется, ничего не поделаешь.

Тем не менее, существуют системы, в которых требуется протокол передачи данных, обеспечивающий защиту от ошибок и управление потоком. Комитет IEEE разработал такой протокол, способный работать поверх Ethernet и других протоколов стандарта 802. Этот протокол, получивший название **LLC** (Logical Link Control — управление логическим соединением), скрывает различия между различными типами сетей 802.x, предоставляя сетевому уровню единый формат и интерфейс. Формат, интерфейс и протокол основаны на протоколе HDLC, который мы уже рассматривали в главе 3. Подуровень LLC образует верхнюю половину уровня передачи данных, а подуровень MAC — нижнюю, как показано на рис. 4.21.

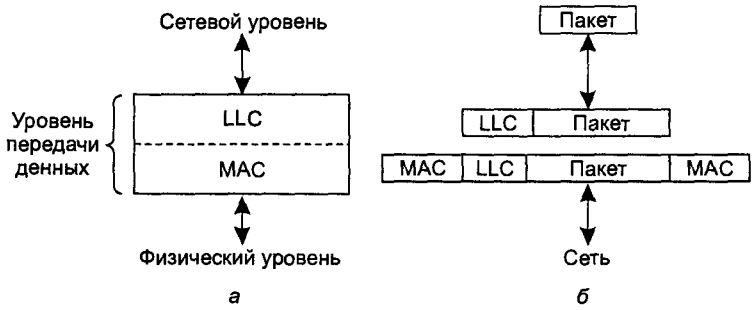


Рис. 4.21. Расположение подуровня LLC (*а*); форматы протокола (*б*)

Обычно подуровень LLC используется следующим образом. Сетевой уровень передает пакет для LLC с помощью примитивов доступа к подуровню. Затем LLC добавляет к нему свой заголовок, содержащий порядковый номер и номер подтверждения. Затем получившаяся в результате структура помещается в поле данных кадра 802.x, который передается по каналу. На приемной станции происходит обратный процесс.

Подуровень LLC предоставляет три следующих варианта сервисов: ненадежный дейтаграммный сервис, дейтаграммный сервис с подтверждениями и надеж-

ный ориентированный на соединение сервис. В заголовке LLC имеются три поля: целевая точка доступа, исходная точка доступа и контрольное поле. Точки доступа определяют, с какого процесса пришел кадр и какому процессу его нужно доставить. Это аналог поля *Type* кадра DIX. Контрольное поле содержит порядковые номера и номера подтверждений, что также очень сильно напоминает стиль HDLC (см. рис. 3.17), однако форматы все-таки немного различаются. Эти поля используются в основном тогда, когда требуется надежное соединение на уровне передачи данных. В данном случае протоколы идентичны обсуждавшимся в главе 3. Для передачи через Интернет IP-пакетов подтверждения на уровне LLC не требуются.

Ретроспектива Ethernet

Ethernet существует вот уже 20 лет, и никаких серьезных конкурентов за это время не появилось. Похоже, и в ближайшее время не появятся. Очень немногие микропроцессорные архитектуры, операционные системы и языки программирования могут похвастаться таким долгим и уверенным лидерством. Вероятно, Ethernet чем-то очень выгодно отличается от всех остальных систем. Чем же?

Возможно, основной причиной столько длительного успеха является простота и гибкость системы. Простота в данном случае означает, прежде всего, надежность, невысокую цену и легкость обслуживания. С тех пор, как на смену ответвителям типа «зуб вампира» пришли BNC-коннекторы, чисто технические поломки стали чрезвычайно редки. Человек так устроен, что он с трудом может отказаться от чего-либо, что хорошо работает, в пользу чего-то другого. Нужно принять во внимание и тот факт, что огромное количество кое-как собранной компьютерной аппаратуры работает не слишком надежно. Именно по этой причине так называемые апгрейды часто дают результат, прямо противоположный ожидаемому. Бывает так, что системы после них работают не лучше, а даже хуже.

Вторая причина популярности Ethernet — это низкая цена. В самом деле, тонкий Ethernet и витая пара — это сравнительно недорогие среды передачи информации. Сетевые карты тоже весьма дешевы. Единственное, что требует каких-то существенных затрат, — это концентраторы и коммутаторы. Но к тому времени, когда эти устройства появились, сети Ethernet уже прочно вошли в жизнь многих предприятий и организаций.

Сети Ethernet не доставляют большой головной боли системным администраторам — они обслуживаются без особых проблем. Не нужно устанавливать никакое программное обеспечение (кроме драйверов), и нет никаких конфигурационных таблиц (в которых так просто ошибиться). Новые узлы добавляются очень просто.

Еще одно достоинство Ethernet заключается в хорошем взаимодействии с TCP/IP — доминирующим протоколом сети Интернет. IP — это протокол без установления соединения, поэтому он без проблем внедряется в локальных сетях Ethernet, которые также используют протоколы без соединения. IP имеет довольно плохую совместимость с сетями ATM, ориентированными на установку соединения. Этот факт крайне негативно сказывается на популярности ATM.

Разработчикам Ethernet удалось добиться хороших показателей по самым главным направлениям. Скорости выросли на несколько порядков, в систему были внедрены коммутаторы и хабы, но эти изменения никак не коснулись программного обеспечения. Если продавец скажет: «Вот отличная новая сетевая система! Она работает просто фантастически быстро и надежно! Вам необходимо только выкинуть весь ваш старый железный хлам и стереть все старые программы», — у него возникнут проблемы с объемами продаж. FDDI, волоконно-оптический канал и ATM были, конечно, быстры, ничего не скажешь, особенно по сравнению с тогдашним Ethernet, но именно из-за того, что они не были совместимы с Ethernet и были гораздо сложнее и тяжелее в обслуживании, про них вскоре все забыли. А потом оказалось, что Ethernet догнал и перегнал их по скорости работы, сумев не растерять при этом все свои старые достоинства. Все соперники Ethernet давно почили в бозе, кроме ATM, — этакой телефонной крысы, живущей в самом ядре телефонии.

Беспроводные локальные сети

Несмотря на то, что Ethernet чрезвычайно популярен, ему все же есть с кем соперничать. Беспроводные локальные вычислительные сети становятся все более популярными, все больше и больше офисных зданий, аэровокзалов и других общественных мест оборудуются соответствующей аппаратурой. Беспроводные ЛВС могут существовать в двух конфигурациях: с базовой станцией и без нее. Стандарт 802.11 принимает это во внимание и поддерживает оба варианта. Далее мы вкратце рассмотрим данный стандарт.

Собственно говоря, мы уже касались вопроса беспроводных ЛВС в главе 1, разделе «Ethernet». Теперь более пристальный взгляд обратим на технологическую сторону стандарта 802.11. В последующих разделах речь пойдет о стеке протоколов, методах радиопередачи (на физическом уровне), протоколе подуровня MAC, структуре кадра и сервисах. Дополнительную информацию о стандарте 802.11 можно найти в следующих изданиях (Crow и др., 1997; Geier, 2002; Heegar и др., 2001; Kapp, 2002; O'Hara и Petrick, 1999; Serevance, 1999). Чтобы получить информацию из первых рук, обратитесь к официальному техническому описанию стандарта.

Стандарт 802.11: стек протоколов

Все протоколы, используемые семейством стандартов 802.x, схожи по структуре. Часть стека протоколов изображена на рис. 4.22. Физический уровень практически соответствует физическому уровню в модели OSI, а вот уровень передачи данных во всех протоколах 802.x разбит на два или более подуровня. Что касается 802.11, то подуровень MAC (подуровень управления доступом к среде) отвечает за распределение канала, то есть за то, какая станция будет передавать следующей. Над MAC в иерархии находится подуровень LLC (управления логическим соединением), задача которого состоит в том, чтобы сделать различия стандартов

802.x невидимыми для сетевого уровня. Мы уже рассматривали LLC ранее, когда речь шла об Ethernet, поэтому сейчас мы не будем возвращаться к этому материалу.



Рис. 4.22. Часть стека протоколов 802.11

Стандарт 802.11 1997 года определяет три метода передачи, реализуемые на физическом уровне. Метод инфракрасной передачи сильно напоминает метод, используемый в системах дистанционного управления бытовой техникой. В двух других методах применяется радиосвязь небольшого радиуса действия (при этом работают методы FHSS и DSSS). Они оба используют не подлежащую лицензированию часть спектра (диапазон ISM 2,4 ГГц). В этой же части спектра осуществляют передачу сигнала радиобрелоки открывания ворот гаражей, так что не удивляйтесь, если ваш ноутбук не сможет в какой-то момент поделить эфир с вашим же гаражом. Кроме того, в этом же диапазоне работают домашние радиотелефоны и СВЧ-печи. Вне зависимости от метода скорость работы составляет 1–2 Мбит/с, и сигнал используется относительно маломощный, что позволяет уменьшить количество конфликтов между передатчиками. С целью увеличения пропускной способности в 1999 году были разработаны два дополнительных метода: OFDM и HR-DSSS. Они работают со скоростями 54 Мбит/с и 11 Мбит/с соответственно. В 2001 году была представлена новая модификация OFDM, работающая в другом частотном диапазоне. Сейчас мы изучим вкратце все эти методы. Они относятся к физическому уровню, которому была посвящена, вообще говоря, вся глава 2. Но из-за того, что они так тесно связаны с ЛВС вообще и с подуровнем MAC стандарта 802.11 в частности, мы обращаемся к ним здесь.

Стандарт 802.11: физический уровень

Все пять рассматриваемых далее методов передачи данных позволяют передать кадр подуровня MAC с одной станции на другую. Различаются они используемыми технологиями и достижимыми скоростями. Детальное рассмотрение этих

методов выходит за рамки данной книги, мы лишь дадим краткое описание, которое, возможно, заинтересует читателей и снабдит их необходимыми терминами для поиска более подробной информации в Интернете или где-то еще.

При передаче в инфракрасном диапазоне (вне диапазона видимого света) используются длины волн 0,85 или 0,95 мкм. Возможны две скорости передачи: 1 и 2 Мбит/с. При 1 Мбит/с используется схема кодирования с группировкой четырех бит в 16-битное кодовое слово, содержащее 15 нулей и 1 единицу. Это так называемый **код Грея**. Одно из его свойств заключается в том, что небольшая ошибка в синхронизации может привести в худшем случае к ошибке в одном бите выходной последовательности. При скорости передачи 2 Мбит/с уже 2 бита кодируются в 4-битное кодовое слово, также имеющее всего одну единицу: 0001, 0010, 0100 или 1000. Сигналы инфракрасного диапазона не проникают сквозь стены, поэтому соты, расположенные в разных комнатах, очень хорошо изолированы друг от друга. Однако из-за довольно низкой пропускной способности (а также потому, что солнечный свет может искажать инфракрасные сигналы) этот метод не слишком популярен.

В методе **FHSS** (Frequency Hopping Spread Spectrum — передача широкополосных сигналов по методу частотных скачков) используются 79 каналов шириной 1 МГц каждый. Диапазон, в котором работает этот метод, начинается с 2,4 ГГц (это нелицензируемый [ISM] диапазон). Для определения последовательностей скачков частот используется генератор псевдослучайных чисел. Поскольку при этом для всех станций используется один и тот же генератор, они синхронизированы во времени и одновременно осуществляют одинаковые частотные скачки. Период времени, в течение которого станция работает на определенной частоте, называется **временем пребывания**. Это настраиваемая величина, но она должна быть не более 400 мс. Рандомизация, осуществляемая в методе FHSS, является простым способом распределения неуправляемого ISM-диапазона. Кроме того, постоянная смена частот — это неплохой (хотя, конечно, недостаточный) способ защиты информации от несанкционированного прослушивания, поскольку незваный слушатель, не зная последовательности частотных переходов и времени пребывания, не сможет подслушать передаваемые данные. При связи на более длинных дистанциях может возникать проблема многолучевого затухания, и FHSS может оказаться хорошим подспорьем в борьбе с ней. Этот метод также относительно слабо чувствителен к интерференции с радиосигналом, что делает его популярным при связи между зданиями. Главный недостаток FHSS — это низкая пропускная способность.

Третий метод модуляции называется **DSSS** (Direct Sequence Spread Spectrum — передача широкополосного сигнала по методу прямой последовательности). Скорости передачи ограничены 1 или 2 Мбит/с. DSSS немного напоминает уже обсуждавшуюся в разделе «Второе поколение мобильных телефонов: цифровая передача голоса» систему CDMA, однако имеет и некоторые отличия. Каждый бит передается в виде 11 элементарных сигналов, которые называются **последовательностью Баркера**. Для этого используется модуляция с фазовым сдвигом со скоростью 1 Мбод (1 бит на бод при работе на 1 Мбит/с и 2 бита на бод при работе на 2 Мбит/с). В течение нескольких лет комиссия FCC требова-

ла, чтобы все беспроводное оборудование в США работало в нелицензируемых диапазонах, однако в мае 2002 года это требование было снято, поскольку появились новые технологии.

Первая высокоскоростная беспроводная ЛВС, **802.11a**, использовала метод **OFDM** (Orthogonal Frequency Division Multiplexing — ортогональное частотное уплотнение) для передачи сигнала со скоростью до 54 Мбит/с в расширенном нелицензируемом диапазоне 5 ГГц. Как и полагается при частотном уплотнении, здесь используются разные частоты. Всего их 52, из них 48 частот предназначены для данных, 4 — для синхронизации (почти как в ADSL). Одновременная передача сигналов на разных частотах позволяет говорить о расширенном спектре, хотя этот метод существенно отличается от CDMA и FHSS. Разделение сигнала на много узких диапазонов имеет преимущества перед передачей в одном широком диапазоне — в частности, более низкую чувствительность к узкополосной интерференции и возможность использования независимых диапазонов. Система кодирования довольно сложна. Она основана на модуляции с фазовым сдвигом для скоростей до 18 Мбит/с и на QAM при более высоких скоростях. При 54 Мбит/с 216 бит данных кодируются 288-битными кодовыми словами. Одним из преимуществ OFDM является совместимость с европейской системой HiperLAN/2 (Doufexi и др., 2002). Метод имеет хорошую спектральную эффективность в терминах соотношения бит/герц и хороший иммунитет против многолучевого затухания.

Наконец, мы подошли к методу **HR-DSSS** (High Rate Direct Sequence Spread Spectrum — высокоскоростная передача широкополосного сигнала по методу прямой последовательности). Это еще один широкополосный способ, который для достижения скорости 11 Мбит/с кодирует биты со скоростью 11 миллионов элементарных сигналов в секунду. Стандарт называется **802.11b**, но он не является последователем 802.11a. На самом деле 802.11b был признан и попал на рынок даже раньше, чем 802.11a. Скорости передачи данных, поддерживаемые этим стандартом, равны 1, 2, 5,5 и 11 Мбит/с. Две низкие скорости требуют 1 Мбод при 1 и 2 битах на бод соответственно. Используется модуляция с фазовым сдвигом (для совместимости с DSSS). Две высокие скорости требуют кодирования со скоростью 1,375 Мбод при 4 и 8 битах на бод соответственно. Применяется **код Уолша — Адамара**. Скорость передачи может быть динамически изменена во время работы для достижения оптимальных результатов в зависимости от условий нагрузки и зашумленности линии. На практике скорость работы стандарта 802.11b почти всегда равна 11 Мбит/с. Хотя 802.11b медленнее, чем 802.11a, диапазон первого почти в 7 раз шире, что бывает очень важно во многих ситуациях.

Улучшенная версия 802.11b называется **802.11g**. Этот стандарт был принят IEEE в ноябре 2001 года после долгих обсуждений того, чья же патентованная технология будет применяться. В итоге в 802.11g применяется метод модуляции OFDM, взятый из 802.11a, однако рабочий диапазон совпадает с 802.11b (узкий нелицензированный диапазон 2,4 ГГц). Теоретически максимальная скорость 802.11g равна 54 Мбит/с. До сих пор не очень понятно, может ли быть достигнута такая скорость на практике. Зато, пока суть да дело, комитет 802.11 может гордо заявить, что он разработал три высокоскоростных стандарта беспроводных

ЛВС: 802.11a, 802.11b и 802.11g (не говоря уж о трех низкоскоростных беспроводных ЛВС). Можно вполне обоснованно удивляться тому, что же в этом хорошего. Ну, видимо, три — это просто счастливое число для комитета 802.11.

Стандарт 802.11: протокол подуровня управления доступом к среде

Однако вернемся из области электротехники в область computer science. Протокол подуровня MAC (напомним, MAC расшифровывается как Medium Access Control — управление доступом к среде) в стандарте 802.11 довольно сильно отличается от аналогичного протокола Ethernet благодаря присущей беспроводным сетям сложности по сравнению с проводными сетями. В Ethernet станция просто ожидает, пока в канале настанет тишина, и тогда начинает передачу. Если шумовой всплеск не приходит обратно в течение времени, необходимого на пересылку 64 байт, то можно утверждать, что кадр почти наверняка доставлен корректно. В беспроводных сетях такой фокус не проходит.

Во-первых, существует проблема скрытой станции — мы уже упоминали о ней ранее, а сейчас приводим еще и иллюстрацию (рис. 4.23, *а*). Поскольку не все станции могут слышать друг друга, передача, идущая в одной части соты, может быть просто не воспринята станцией, находящейся в другой ее части. В приведенном на рисунке примере станция *C* передает данные станции *B*. Если станция *A* прослушает канал, она не обнаружит ничего подозрительного и сделает ложный вывод о том, что она имеет право начать передачу станции *B*. Кроме того, есть и обратная проблема, показанная на рис. 4.23, *б*. Здесь *B* хочет отправить данные для станции *C* и прослушивает канал. Услышав, что в нем уже осуществляется какая-то передача, станция *B* делает опять-таки ложный вывод о том, что передача для *C* сейчас невозможна. Между тем станция *A* — источник сигнала, который смутил станцию *B*, — может на самом деле осуществлять передачу для станции *D* (на рисунке не показана). Ситуация усугубляется еще и тем, что большинство радиосистем являются полудуплексными, то есть не могут одновременно и на одной и той же частоте посылать сигналы и воспринимать всплески шума на линии. В итоге 802.11 не может использовать, как Ethernet, метод CSMA/CD.

Как бороться с этой проблемой? Стандарт 802.11 поддерживает два режима работы. Первый называется **DCF** (Distributed Coordination Function — распределенная координация) и не имеет никаких средств централизованного управления (в этом смысле напоминая Ethernet). Второй режим, **PCF** (Point Coordination Function — сосредоточенная координация), подразумевает, что базовая станция берет на себя функцию управления активностью всех станций данной соты. Все реализации стандарта должны поддерживать DCF, тогда как PCF является дополнительной возможностью. Сейчас мы перейдем к рассмотрению этих режимов.

В режиме DCF 802.11 использует протокол, называемый CSMA/CA (CSMA with Collision Avoidance — CSMA с предотвращением коллизий). Здесь ведется прослушивание как физического, так и виртуального канала. Протокол CSMA/CA может работать в двух режимах. В первом режиме станция перед передачей про-

слушивает канал. Если он свободен, начинается пересылка данных. Во время пересылки канал не прослушивается, и станция передает кадр целиком, причем он может быть разрушен на стороне приемника из-за интерференции сигналов. Если канал занят, отправитель дожидается его освобождения и затем начинает передачу. Если возникает коллизия, станции, не поделившие между собой канал, выжидают в течение случайных интервалов времени (используется двоичный экспоненциальный откат такой же, как в Ethernet) и затем снова пытаются отправить кадр.

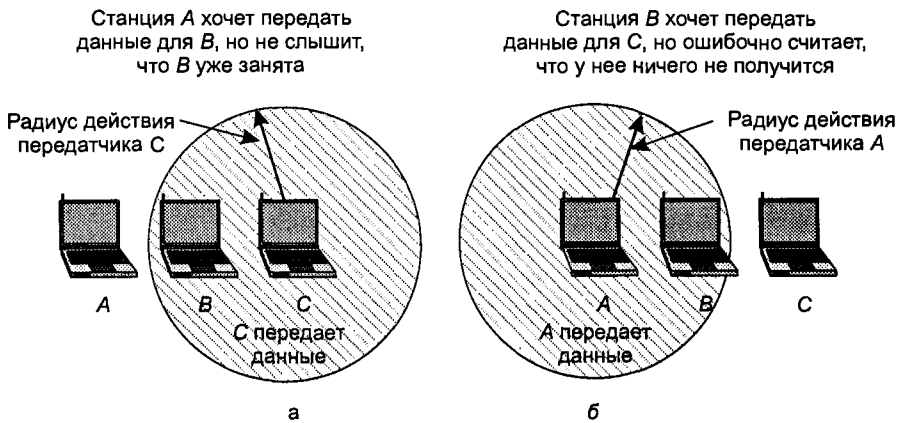


Рис. 4.23. Проблема скрытой станции (*а*); проблема засвеченной станции (*б*)

Другой режим CSMA/CA основан на протоколе MACAW и использует контроль виртуального канала, как показано на рис. 4.24. В этом примере станция *A* хочет передать данные станции *B*. Станция *C* находится в зоне действия (то есть слышит) *A*, а также, возможно, в зоне действия *B*, но это не имеет значения. Станция *D* входит в зону действия *B*, но не входит в зону действия *A*.

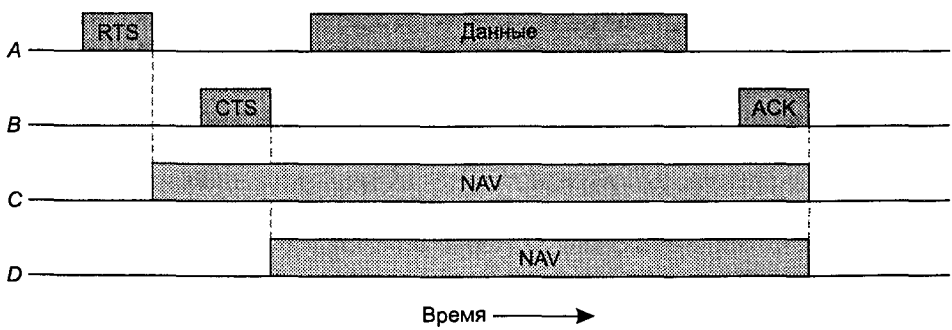


Рис. 4.24. Использование прослушивания виртуального канала в протоколе CSMA/CA

Протокол начинает работать тогда, когда *A* решает, что ей необходимо послать данные *B*. *A* посылает станции *B* кадр RTS, запрашивая разрешение на пе-

редачу. Если *B* может принять данные, она отсылает обратно положительное подтверждение, кадр CTS. После приема CTS *A* запускает таймер ACK и начинает передачу данных. В случае корректного приема *B* генерирует кадр ACK, сообщающий станции *A* о конце передачи. Если интервал времени таймера на станции *A* истекает прежде, чем получен ACK, весь алгоритм работы протокола повторяется с самого начала.

Теперь рассмотрим этот же процесс с точки зрения станций *C* и *D*. *C* находится в зоне действия *A*, поэтому она также принимает кадр RTS и понимает, что скоро по каналу будут передаваться какие-то данные и лучше при этом помолчать и подождать окончания активности соседних станций. Исходя из информации, содержащейся в RTS, станция *C* может предположить, сколько времени займет передача последовательности, включая конечный ACK. В течение этого промежутка *C* считает, что ее виртуальный канал занят и она может отдыхать. Индикацией такого состояния является последовательность **NAV** (Network Allocation Vector — вектор выделенной сети), показанная на рис. 4.24. Станция *D* не слышит RTS, посылаемый *A*, зато слышит CTS, посланный станцией *B*, и также выставляет NAV. Обратите внимание: сигналы NAV не передаются, а являются лишь внутренними напоминаниями станций о том, что нужно хранить молчание в течение определенного промежутка времени.

В противоположность проводным каналам, беспроводные шумны и ненадежны, в какой-то степени из-за СВЧ-печей, работающих в том же диапазоне. В результате вероятность корректной передачи кадра уменьшается пропорционально увеличению длины кадра. Если вероятность ошибки в одном бите равна p, то вероятность того, что n-битный кадр будет принят корректно, равна $(1-p)^n$. Например, для $p = 10^{-4}$ вероятность корректной передачи полного Ethernet-кадра длиной 12 144 бит составляет менее 30 %. Если $p = 10^{-5}$, примерно один кадр из 9 будет испорчен. Даже при $p = 10^{-6}$ более 1 % кадров будет испорчено, то есть за 1 секунду ошибки в кадрах будут возникать примерно 12 раз. Длинные кадры вообще имеют очень мало шансов дойти до получателя неповрежденными, и их нужно посылать заново.

Для решения проблемы зашумленных каналов беспроводных сетей применяется разбиение кадров на небольшие отрезки, каждый из которых содержит собственную контрольную сумму. Фрагменты нумеруются и подтверждаются индивидуально с использованием протокола с ожиданием (то есть отправитель не может передать фрагмент с номером $k + 1$, пока не получит подтверждения о доставке фрагмента с номером k). Захватив канал с помощью диалога, состоящего из RTS и CTS, отправитель может передать несколько кадров подряд, как показано на рис. 4.25. Последовательность фрагментов называется **пачкой фрагментов**.

Фрагментация повышает производительность путем принудительной повторной пересылки коротких отрезков кадров, в которых произошла ошибка, а не кадров целиком. Размер фрагмента не закрепляется стандартом, а является настраиваемым параметром каждой ячейки беспроводной сети и может оптимизироваться базовой станцией. Механизм выставления NAV удерживает станции от передачи только до прихода первого подтверждения о доставке. Но есть и другой

механизм (он описан далее), позволяющий получателю принять всю пачку фрагментов без интерференции с сигналами сторонних станций.

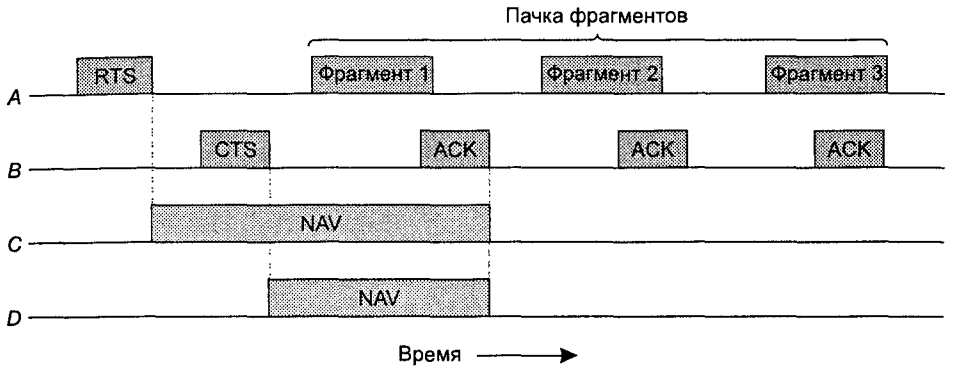


Рис. 4.25. Пачка фрагментов

Итак, все описанное ранее относится к режиму DCF (режим распределенной координации) стандарта 802.11. В данном режиме отсутствует централизованный контроль, и станции самостоятельно борются за эфирное время примерно так же, как в Ethernet. Но беспроводные сети могут работать и в другом режиме, который называется PCF (режим сосредоточенной координации). Здесь базовая станция опрашивает все подчиненные ей станции, выявляя те из них, которые требуют предоставить им канал. Порядок «выступлений» полностью и централизованно координируется базовой станцией, поэтому коллизии в режиме PCF исключены. Стандарт лишь предписывает осуществлять такую координацию, но не дает конкретных указаний, касающихся частоты, порядка опросов или наличия либо отсутствия каких-либо приоритетов у отдельных станций.

Механизм основан на том, что базовая станция широковещательным способом периодически (10–100 раз в секунду) передает **сигнальный кадр**. В нем содержатся такие системные параметры, как последовательности смены частот и периоды пребывания на частотах (для FHSS), данные для синхронизации и т. д. Он также является приглашением для новых станций, которые желают войти в список опрашиваемых станций. Попав в этот список, станция получает гарантированную долю пропускной способности (при определенных параметрах скорости), то есть ей гарантируется качество обслуживания.

Аккумуляторы имеют свойство разряжаться, поэтому в беспроводных сетях экономия электроэнергии — это очень животрепещущий вопрос. В частности, базовая станция может приказать станции идти спать (то есть перейти в режим пониженного потребления) до тех пор, пока ее не разбудят (базовая станция или пользователь). Однако даже в таком «сонном» режиме станция должна иметь возможность сохранять в буфере приходящие кадры для последующей их обработки.

Режимы PCF и DCF могут сосуществовать даже внутри одной соты сети. Поначалу это может показаться нереальным: как это так — сосредоточенный и рас-

пределенный контроль одновременно? Однако стандарт 802.11 действительно предлагает такую возможность. Это делается путем очень аккуратного определения межкадрового интервала. После отправки кадра необходимо какое-то время простоя, прежде чем какая-либо станция получит разрешение послать кадр. Всего определено четыре интервала, каждый из которых имеет собственное предназначение. Они изображены на рис. 4.26.

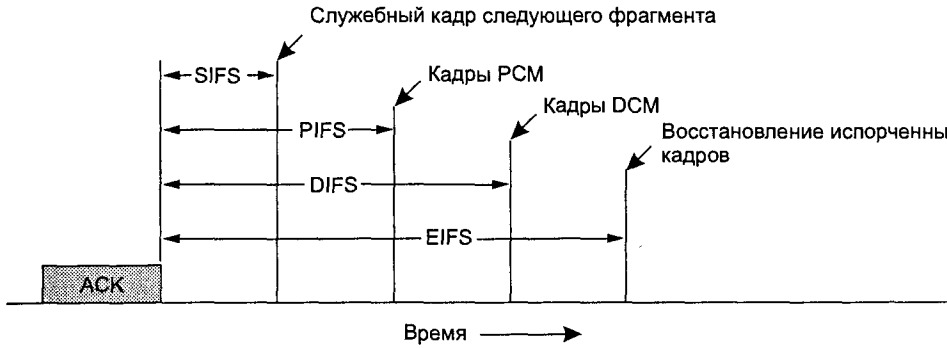


Рис. 4.26. Межкадровые интервалы в стандарте 802.11

Самый короткий интервал — это **SIFS** (Short InterFrame Interval — короткий межкадровый интервал). Он используется для того, чтобы одна из сторон, ведущих диалог с помощью управляющих кадров, могла получить шанс начать первой. Здесь может быть CTS, посылаемый приемником в ответ на запрос RTS; ACK, посылаемый им же после окончания приема фрагмента или целого кадра; очередная часть пакета фрагментов, посылаемая отправителем (то есть он не посылает RTS после каждого фрагмента).

После интервала SIFS ответить может всегда только одна станция. Если она упускает свой шанс и время **PIFS** (PCF InterFrame Spacing — межкадровый интервал PCF) истекает, то базовая станция может послать сигнальный кадр или кадр опроса. Этот механизм позволяет станции, посылающей кадр данных или последовательность фрагментов, закончить свою передачу без какого-либо вмешательства со стороны соседей, но дает и базовой станции возможность после окончания передачи станцией захватить канал, не борясь за него с другими желающими.

Если базовой станции нечего сказать и интервал **DIFS** (DCF InterFrame Spacing — межкадровый интервал DCF) истекает, то любая станция может попытаться захватить канал. Применяются при этом обычные правила борьбы, включая двоичный экспоненциальный откат в случае коллизии.

Последний временной интервал называется **EIFS** (Extended InterFrame Spacing — расширенный межкадровый интервал). Он используется только той станцией, которая только что получила испорченный или неопознанный кадр и хочет сообщить об этом факте. Почему наиболее низкий приоритет отдан именно этому событию? Дело в том, что приемник может сразу не сообразить, что про-

исходит, и ему нужно выждать в течение какого-то интервала, чтобы не прервать своим возмущенным возгласом идущий в это время диалог между станциями.

Стандарт 802.11: структура кадра

Стандарт 802.11 определяет три класса кадров, передаваемых по каналу: информационные, служебные и управляющие. Все они имеют заголовки с множеством полей, используемых подуровнем MAC. Кроме того, есть поля, используемые физическим уровнем, но они в основном относятся к методам модуляции, поэтому здесь мы их рассматривать не будем.

Формат информационного кадра показан на рис. 4.27. Вначале идет поле *Управление кадром* (Frame Control). Оно содержит 11 вложенных полей. Первое из них — *Версия протокола*, именно оно позволяет двум протоколам работать одновременно в одной ячейке сети. Затем следуют поля *Тип* (информационный, служебный или управляющий) и *Подтип* (например, RTS или CTS). Биты *К DS* и *От DS* говорят о направлении движения кадра: к межсотовой системе распределения (например, Ethernet) или от нее. Бит *MF* говорит о том, что далее следует еще один фрагмент. Бит *Повтор* маркирует повторно посылаемый фрагмент. Бит *Управление питанием* используется базовой станцией для переключения станции в режим пониженного потребления или выхода из этого режима. Бит *Продолжение* говорит о том, что вообще-то у отправителя имеются еще кадры для пересылки. Бит *W* является индикатором использования шифрования в теле кадра по алгоритму **WEP** (Wired Equivalent Protocol — протокол обеспечения конфиденциальности). Наконец, *О* говорит приемнику о том, что кадры с этим битом должны обрабатываться строго по порядку.

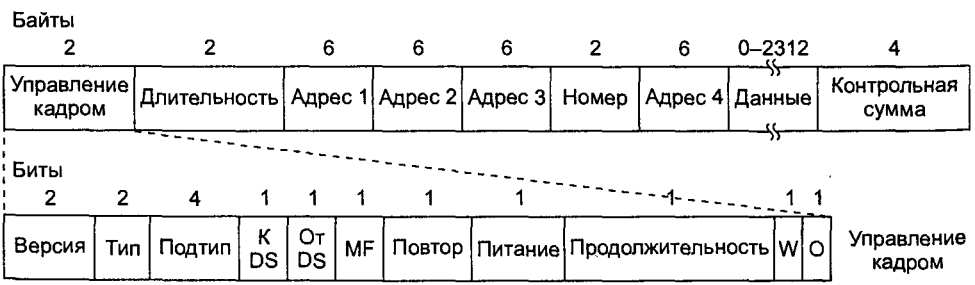


Рис. 4.27. Информационный кадр стандарта 802.11

Второе основное поле информационного кадра — это поле *Длительность*. В нем задается время, которое будет потрачено на передачу кадра и подтверждения. Это поле можно найти и в служебных кадрах, и именно в соответствии с ним станции выставляют признаки NAV. Заголовок кадра содержит также четыре адреса в формате, соответствующем стандарту IEEE 802. Понятно, что нужны адреса отправителя и получателя, но что же содержится в двух оставшихся? Дело в том, что кадры могут входить в ячейку или покидать ее через базовую станцию. Два адреса как раз и хранят адреса исходной и целевой ячеек при передаче трафика между ячейками.

Поле *Номер* позволяет нумеровать фрагменты. Из 16 доступных бит 12 идентифицируют кадр, а 4 — фрагмент. Поле *Данные* содержит передаваемую по каналу информацию, его длина может достигать 2312 байт. В конце, как обычно, расположено поле *Контрольная сумма*.

Управляющие кадры имеют формат, сходный с форматом информационных кадров, за одним исключением: в управляющем кадре отсутствуют поля базовых станций, поскольку таким кадрам незачем выходить за пределы соты. Служебные кадры гораздо короче: в них содержится один или два адреса, отсутствуют поля *Данные* и *Номер*. Ключевой здесь является информация, содержащаяся в поле *Подтип*. Значениями обычно являются RTS, CTS или ACK.

Сервисы

Стандарт 802.11 утверждает, что все совместимые беспроводные ЛВС должны предоставлять девять типов сервисов (услуг). Их можно разделить на две категории: сервисы распределения (к ним относятся пять из девяти) и станционные (соответственно, четыре сервиса). Сервисы распределения связаны с управлением станциями, находящимися в данной соте, и взаимодействием с внешними станциями. Станционные сервисы, наоборот, имеют отношение к управлению активностью внутри одной соты.

Пять сервисов распределения предоставляются базовой станцией и имеют дело с мобильностью станций при их входе в соту или выходе из нее. При этом станции устанавливают либо разрывают взаимодействие с базовой станцией. Ниже перечислены сервисы распределения.

1. **Ассоциация.** Этот сервис используется мобильными станциями для подключения к базовым станциям (БС). Обычно он применяется сразу же после вхождения в зону действия БС. Мобильная станция передает идентификационную информацию и сообщает о своих возможностях (поддерживаемой скорости передачи данных, необходимости PCF-услуг, или опроса) и требованиях по управлению электропитанием. Базовая станция может принять или отвергнуть мобильную станцию. Если последняя принята, она должна пройти идентификацию.
2. **Дизассоциация.** По инициативе мобильной или базовой станции может быть произведена дизассоциация, то есть разрыв отношений. Это требуется при выключении станции или ее уходе из зоны действия БС. Впрочем, базовая станция также может быть инициатором дизассоциации, если, например, она временно выключается для проведения технического обслуживания.
3. **Реассоциация.** С помощью этого сервиса станция может сменить БС. Очевидно, данная услуга используется при перемещении станции из одной соты в другую. Если она проходит корректно и без сбоев, то при переходе никакие данные не теряются. (Однако, как и в сети Ethernet, в стандарте 802.11 все услуги предоставляются лишь с обязательством приложения максимальных усилий к их исполнению, но не с гарантией.)
4. **Распределение.** С помощью этой услуги определяется маршрутизация кадров, посылаемых базовой станции. Если адрес назначения является локаль-

ным с точки зрения БС, то кадры следуют просто напрямую (передаются в эфире). В противном случае их необходимо пересылать по проводной сети.

5. **Интеграция.** Если кадру нужно пройти через сеть, не подчиняющуюся стандарту 802.11 и использующую другую схему адресации и/или формат кадра, то на помощь приходит данный сервис. Он реализует трансляцию форматов.

Оставшиеся четыре сервиса — это внутренние услуги соты. Они предоставляются после прохождения *ассоциации*, описанной выше. Ниже перечислены станционные сервисы:

1. **Идентификация.** Поскольку беспроводные коммуникации подразумевают очень легкое подключение к сети и возможность приема/отправки данных любыми станциями, попавшими в зону действия БС, то возникает необходимость идентификации. Только после идентификации станции разрешается обмен данными. После принятия мобильной станции в ряды текущих абонентов соты базовая станция посылает специальный кадр запроса, позволяющий понять, знает ли станция присвоенный ей секретный ключ (пароль). Подтверждение осуществляется путем шифрования кадра запроса и отсылки его назад базовой станции. Если шифрование выполнено корректно, мобильная станция получает нормальные права доступа к сети. Изначально стандарт не требует, чтобы базовая станция отправляла свои идентификационные данные мобильной станции, но работа над исправлением этого упущения уже ведется.
2. **Деидентификация.** Если станция, работавшая в сети, покидает ее, она должна произвести деидентификацию. После выполнения данного сервиса она больше не сможет использовать ячейку.
3. **Конфиденциальность.** Чтобы сохранить передаваемые по сети данные в тайне от посторонних «ушей», их необходимо шифровать. Данный сервис осуществляет операции по шифрации и дешифрации информации. Применяется алгоритм RC4, изобретенный Рональдом Ривестом (Ronald Rivest) из M.I.T.
4. **Доставка данных.** Собственно говоря, именно этот сервис является ключевым во всей работе сети. Ведь сеть 802.11 существует для обмена данными. Поскольку стандарт 802.11 основан на стандарте Ethernet, а в последнем доставка данных не является гарантированной на 100 %, то для беспроводных сетей это тем более верно. Обнаруживать и исправлять ошибки поручено верхним уровням.

У ячейки 802.11 имеются некоторые параметры, которые проверяются и иногда случаях настраиваются. Они относятся к шифрованию данных, интервалам тайм-аутов, скоростям передачи данных, частотам сигналов и т. д.

Беспроводные сети на основе стандарта 802.11 все шире применяются в учреждениях, аэропортах, гостиницах, ресторанах, а также на производственных территориях и в университетских кампусах по всему миру. Быстрый рост этих сетей уже практически неизбежен. Рекомендую изучить книгу (Hills, 2001) — в ней описано, как сеть 802.11 развертывалась в университете Карнеги — Мелона.

Широкополосные беспроводные сети

Что-то мы засиделись в помещении. Выйдем на улицу и посмотрим — может быть, там тоже есть какие-нибудь интересные сети? А там действительно что-то есть, и в основном мы увидим то, что называется «последней милей». Во многих странах телефонная система в какой-то момент перестала быть жестко управляемой, и появилось множество фирм, предлагающих локальные услуги голосовой связи и интернет-услуги.

Предложений действительно много. Проблема только в том, что прокладка волоконно-оптического кабеля, коаксиала или даже витой пары пятой категории к миллионам абонентов обходится очень дорого. Что же делать?

Ответ одновременно и прост, и сложен. Нужны широкополосные беспроводные сети. Установить одну большую антенну на горке где-нибудь рядом с населенным пунктом и расставить на крышах домов абонентов приемные антенны гораздо проще и дешевле, чем рыть траншеи и протягивать кабель. Таким образом, конкурирующие операторы связи оказались крайне заинтересованы в развитии многомегабитных беспроводных систем связи, реализующих услуги голосовой коммуникации, доступа к Интернету, видео по заказу и т. д. Как было продемонстрировано на рис. 2.27, очень неплохо с задачей справляется система LMDS. Однако еще до недавних пор каждый оператор стремился использовать свою собственную систему. Отсутствие стандартов означало, что аппаратное и программное обеспечение не могли быть запущены в массовое производство. Это, в свою очередь, приводило к установке неоправданно высоких цен при низкой популярности.

Наконец светлые головы, работающие в данной отрасли, поняли, что именно стандартизация широкополосных беспроводных сетей является главным камнем преткновения. По традиции, сформировать комитет для разработки стандарта было поручено IEEE. В комитет **802.16** вошли представители крупнейших фирм и научных организаций. Работа была начата в июле 1999 года, а официальный результат был представлен в апреле 2002 года. Название стандарта таково: «Эфирный интерфейс стационарных широкополосных беспроводных систем доступа». Тем не менее, большинство предпочитает использовать более короткое название: **беспроводные региональные сети**, или **беспроводные местные линии**. Мы считаем, что все эти названия равноценны и взаимозаменяемы.

Подобно некоторым другим стандартам из серии 802, стандарт 802.16 построен с использованием идей модели OSI. Здесь можно найти и уровни, и подуровни, используется схожая терминология, сервисные примитивы и т. п. К сожалению, как и OSI, стандарт 802.16 страдает громоздкостью. В последующих разделах мы приведем краткое описание основных его свойств, но такое исследование является далеко не полным, в нем опущены многие детали. Дополнительную общую информацию о широкополосных беспроводных сетях вы найдете в (Bolcskei и др., 2001; Webb, 2001). Информацию о конкретном стандарте 802.16 можно найти в (Eklund и др., 2002).

Сравнение стандартов 802.11 и 802.16

Казалось бы, зачем провозглашать новый стандарт? Чем так плохи сети 802.11, что их нельзя уж и на улицу вынести? Объяснений появлению нового стандарта можно найти несколько. Во-первых, 802.11 и 802.16 решают совершенно разные проблемы. Прежде чем говорить о каких-либо технических подробностях, стоит, наверное, сказать несколько слов в защиту создателей именно нового стандарта.

Сети 802.11 и 802.16 похожи друг на друга прежде всего в том, что и те, и другие предназначены для обеспечения беспроводной связи с высокой пропускной способностью. Однако дальше начинаются различия. Во-первых, 802.16 предоставляет услуги связи для целых зданий, а здания, как известно, отличаются крайне малой мобильностью. Они не часто мигрируют из соты в соту. Между тем сети 802.11 во многом ориентированы на мобильность абонентов. В здании может быть расположено несколько компьютеров, и этот вариант несколько сложнее, чем мобильная станция с единственным ноутбуком. Владельцы зданий обычно готовы платить за хорошую связь больше, чем владельцы ноутбуков, поэтому появляется возможность установки качественных радиоприемников и передатчиков. В частности, этот аспект влияет на то, что сеть 802.16 может использовать полнодуплексную связь, тогда как экономия денег пользователями сети 802.11 не дает такой возможности.

Сети 802.16 могут охватывать целые районы городов, и расстояния при этом исчисляются километрами. Следовательно, получаемый станциями сигнал может быть разной мощности в зависимости от их удаленности от передатчика. Эти отклонения влияют на соотношение сигнал/шум, что, в свою очередь, приводит к использованию нескольких схем модуляции. Этот вид телекоммуникаций является открытым, эфирный сигнал проходит над целыми городами, поэтому вопросы защиты информации здесь крайне важны.

В каждой ячейке широкополосной региональной сети может быть намного больше пользователей, чем в обычной ячейке 802.11, и при этом каждому пользователю предоставляется гораздо более высокая пропускная способность, чем пользователю беспроводной ЛВС. Кроме всего прочего, трудно представить себе, чтобы начальство фирмы собрало в одной комнате офисного здания 50 сотрудников, для того чтобы посмотреть, что будет с сетью, если на каждую из 50 мобильных станций загружать свой видеофильм. В среде пользователей 802.16 такая ситуация может легко возникнуть сама собой, только не в пределах одной комнаты, а в пределах одной ячейки региональной сети. Нелицензированной (ISM) полосы частот явно недостаточно для такой нагрузки, поэтому сети 802.16 работают в высокочастотном диапазоне 10–66 ГГц. На момент разработки стандарта это была единственная подходящая незанятая часть спектра.

Но ведь миллиметровые волны такого диапазона обладают совсем иными физическими свойствами, чем более длинные волны ISM-диапазона. Значит, необходимо разработать специальный физический уровень для 802.16. Миллиметровые волны, увы, очень эффективно поглощаются водой (особенно дождем, снегом, градом, а иногда даже туманом). Значит, нужно как-то это учитывать при обработке ошибок. Данный тип волн является узконаправленным (волны,

применяемые в стандарте 802.11, распространяются по всем направлениям), поэтому вопрос о многолучевом распространении здесь становится больше теоретическим, чем практическим.

Еще одна проблема — качество обслуживания. Сети 802.11 поддерживают передачу информации в реальном времени (в режиме PCF), но вообще-то они не предназначены для телефонной связи и большого мультимедийного трафика. Стандарт 802.16, напротив, ориентирован на поддержку подобных приложений, поскольку он предназначен как для обывателей, так и для деловых людей.

В двух словах можно так обрисовать различие между описываемыми сетями: стандарт 802.11 — это мобильный аналог Ethernet, а 802.16 — беспроводной, но стационарный аналог кабельного телевидения. Разница оказалась столь большой, что стандарты получились совсем не похожие. Это и понятно: они служат разным целям.

Можно попробовать провести некую аналогию с сотовой телефонной системой. Говоря о мобильных телефонах, мы подразумеваем голосовую связь, осуществляемую в довольно узком частотном диапазоне, с невысокой мощностью и волнами средней длины. Никто (пока что) не смотрит на экранах мобильных телефонов двухчасовые фильмы с высоким разрешением. Даже UMTS, похоже, не изменит ситуацию. Короче говоря, запросы у пользователей беспроводных региональных сетей гораздо выше, чем у владельцев мобильных аппаратов. Следовательно, нужна совсем другая система. Вопрос о возможном использовании стандарта 802.16 для мобильных приложений остается открытым. Он не подразумевает этого, но такая возможность, в принципе, есть. На сегодняшний день ведется беспроводное обслуживание исключительно стационарных абонентов.

Стандарт 802.16: стек протоколов

Набор протоколов, используемый стандартом 802.16, показан на рис. 4.28. Общая структура подобна другим стандартам серии 802, однако здесь мы видим больше подуровней. Нижний подуровень занимается физической передачей данных. Используется обычная узкополосная радиосистема с обыкновенными схемами модуляции сигнала. Над физическим уровнем находится подуровень сведения (с ударением на второй слог), скрывающий от уровня передачи данных различия технологий. На самом деле, в стандарте 802.11 можно найти нечто подобное, просто комитет не потрудился дать этому название в стиле модели OSI.

Хотя мы и не показали этого на рисунке, уже разрабатываются два новых протокола физического уровня. Стандарт 802.16a будет поддерживать OFDM в диапазоне 2–11 ГГц. 802.16b сможет работать в 5-гигагерцевом ISM-диапазоне. Обе вариации на тему 802.16 представляют собой попытки приблизиться к 802.11.

Уровень передачи данных состоит из трех подуровней. Нижний из них относится к защите информации, что очень критично для сетей, в которых передача данных осуществляется в эфире, физически никак не защищенном от прослушивания. В сетях 802.11 в прямом смысле слова «стены помогают» скрыть информацию от несанкционированного доступа. На этом подуровне производится шифрация, дешифрация данных, а также управления ключами доступа.

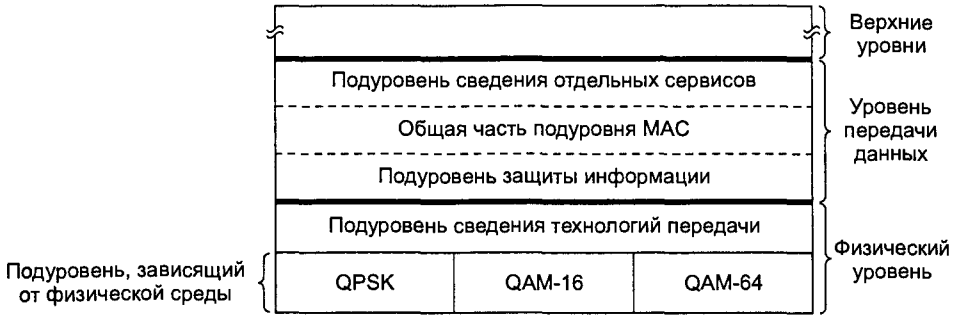


Рис. 4.28. Стек протоколов 802.16

Затем следует общая часть подуровня MAC. Именно на этом уровне иерархии располагаются основные протоколы — в частности, протоколы управления каналом. Идея состоит в том, что базовая станция контролируют всю систему. Она очень эффективно распределяет очередность передачи входящего трафика абонентам, немалую роль играет и в управлении исходящим трафиком (от абонента к базовой станции). От всех остальных стандартов 802.x MAC-подуровень стандарта 802.16 отличается тем, что он полностью ориентирован на установку соединения. Таким образом можно гарантировать определенное качество обслуживания при предоставлении услуг телефонной связи и при передаче мультимедиа.

Подуровень сведения отдельных сервисов играет роль подуровня логической связи в других протоколах 802.x. Его функция заключается в организации интерфейса для сетевого уровня. Сложность здесь состоит в том, что стандарт 802.16, по идее разработчиков, должен быть полностью совместимым как с дейтаграммными протоколами (например, PPP, IP и Ethernet), так и с ATM. Возникает проблема: пакетные протоколы — это протоколы без установления соединения, а ATM ориентирован на установление соединения. Это означает, что любое соединение с использованием ATM должно отображаться на соединение 802.16. Это осуществляется без проблем. Но куда прикажете отображать пришедший IP-пакет? Решением этой проблемы занимается подуровень сведения сервисов.

Стандарт 802.16: физический уровень

Как было сказано ранее, широкополосным беспроводным сетям необходим широкий частотный спектр, который можно найти только в диапазоне от 10 до 66 ГГц. Миллиметровые волны обладают одним интересным свойством, которое отсутствует у более длинных микроволн: они распространяются не во всех направлениях (как звук), а по прямым линиям (как свет). Следовательно, на базовой станции должно быть установлено множество антенн, покрывающих различные секторы окружающей территории, как показано на рис. 4.29. В каждом секторе будут свои пользователи. Секторы не зависят друг от друга, чего не скажешь о сотовой радиосвязи, в которой сигналы распространяются сразу по всем направлениям.

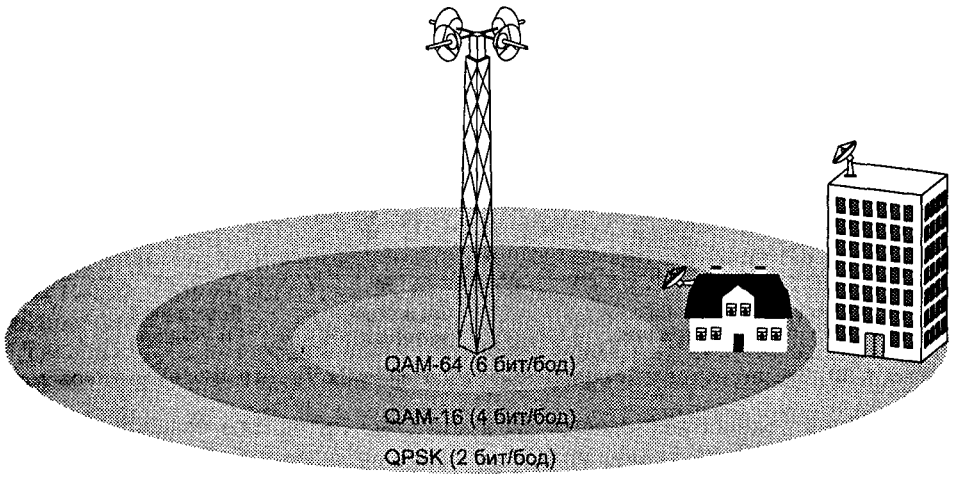


Рис. 4.29. Оперативная среда передачи данных сетей 802.16

Поскольку мощность сигнала передаваемых миллиметровых волн сильно уменьшается с увеличением расстояния от передатчика (то есть базовой станции), то и соотношение сигнал/шум также понижается. По этой причине 802.16 использует три различных схемы модуляции в зависимости от удаления абонентской станции. Если абонент расположен недалеко от БС, то применяется QAM-64 с шестью битами на отсчет. На среднем удалении используется QAM-16 и 4 бита/бод. Наконец, если абонент расположен далеко, то работает схема QPSK с двумя битами на отсчет. Например, при типичной полосе спектра 25 МГц QAM-64 дает скорость 150 Мбит/с, QAM-16 — 100 Мбит/с, а QPSK — 50 Мбит/с. Другими словами, чем дальше находится абонент от базовой станции, тем ниже скорость передачи данных (то же самое мы наблюдали в ADSL, см. рис. 2.23). Фазовые диаграммы всех трех методов были показаны на рис. 2.21.

Перед разработчиками сетей 802.16 стояла трудная задача: необходимо было создать широкополосную систему с учетом приведенных ранее физических ограничений. Для этого следовало продумать наиболее эффективный способ использования доступного спектра. Схемы работы стандартов GSM и DAMPS были отвергнуты сразу: и там, и там для входящего и исходящего трафика используются хоть и разные, но эквивалентные по ширине полосы частот. Для голосовой связи это действительно логично, но при работе в Интернете предоставление широкой полосы для исходящего трафика является непозволительной роскошью. Стандарт 802.16 обеспечивает гибкость распределения полосы пропускания. Применяются две схемы модуляции: **FDD** (Frequency Division Duplexing — дуплексная связь с частотным разделением) и **TDD** (Time Division Duplexing — дуплексная связь с временным разделением). Последний метод показан на рис. 4.30. Что здесь происходит? Базовая станция периодически передает кадры, разделенные на временные интервалы. Первая часть временных интервалов отводится под входящий трафик. Затем следует защитный интервал (разделитель), позволяю-

щий станциям переключать режимы приема и передачи, а за ним — интервалы исходящего трафика. Число отводимых тактов может динамически меняться, что позволяет подстроить пропускную способность под трафик каждого из направлений.

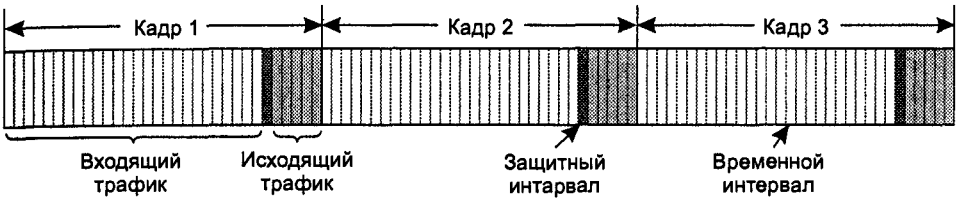


Рис. 4.30. Дуплексная связь с временным разделением: кадры и временные интервалы

Входящий трафик разбивается на временные интервалы базовой станцией. Она полностью контролирует это направление передачи. Исходящий трафик от абонентов управляется более сложным образом и зависит от требуемого качества обслуживания. Мы еще вернемся к распределению временных интервалов, когда будем обсуждать подуровень MAC.

Еще одним интересным свойством физического уровня является его способность упаковывать несколько соседних кадров MAC в одну физическую передачу. Это дает возможность повысить эффективность распределения спектра путем уменьшения числа различных преамбул и заголовков, столь любимых физическим уровнем.

Необходимо также отметить, что для непосредственного исправления ошибок на физическом уровне используется код Хэмминга. Почти все сетевые технологии просто полагаются на контрольные суммы и обнаруживают ошибки с их помощью, запрашивая повторную передачу испорченных фрагментов. Но при широкополосной беспроводной связи на больших расстояниях возникает столько ошибок, что их обработкой приходится заниматься физическому уровню, хотя на более высоких уровнях и применяется метод контрольных сумм. Основная задача коррекции ошибок на физическом уровне состоит в том, чтобы заставить канал выглядеть лучше, чем он есть на самом деле (точно так же компакт-диски кажутся столь надежными носителями только лишь благодаря тому, что больше половины суммарного числа бит отводится под исправление ошибок на физическом уровне).

Стандарт 802.16: протокол подуровня MAC

Итак, уровень передачи данных разделен на три подуровня, как показано на рис. 2.28. Поскольку мы не будем вплоть до восьмой главы касаться принципов криптографии, то сейчас нет смысла пояснять работу подуровня защиты информации. Достаточно сказать, что для сокрытия передаваемых данных применяется шифрация, причем шифруются только сами данные; а заголовки не шифруются. Это означает, что нехороший дядя может узнать, кто с кем разговаривает, но не может подслушать содержание разговора.

Если вы уже знакомы с криптографией, то далее приводится один абзац, из которого вы поймете, какие именно принципы применяются подуровнем защиты информации. В противном случае в следующем абзаце вы найдете мало знакомых слов. Лучше перечитать его после ознакомления с главой 8.

Когда абонент соединяется с базовой станцией, выполняется взаимная идентификация с использованием алгоритма RSA с открытым ключом (сертификат X.509). Сама передаваемая информация шифруется с помощью симметричного криптографического ключа: либо DES с цепочкой цифровых блоков, либо тройной DES с двумя ключами. Вскоре, возможно, будет добавлен AES (Rijndael). Целостность данных проверяется алгоритмом SHA-1. Ну что, не очень страшный абзац получился?

Теперь перейдем к общей части подуровня MAC. Кадры MAC всегда занимают целое число временных интервалов физического уровня. Каждый кадр разбит на части, первые две из которых содержат карту распределения интервалов между входящим и исходящим трафиком. Там находится информация о том, что передается в каждом такте, а также о том, какие такты свободны. Карта распределения входящего потока содержит также разнообразные системные параметры, которые важны для станций, только что подключившихся к эфиру.

Канал входящего трафика устроен довольно просто, поскольку есть базовая станция, которая определяет, что разместить в каждой части кадра. С исходящим каналом все несколько сложнее, поскольку имеются конкурирующие между собой станции, желающие получить доступ к нему. Его распределение тесно связано с вопросом качества обслуживания. Определены четыре класса сервисов:

1. Сервис с постоянной битовой скоростью.
2. Сервис реального времени с переменной битовой скоростью.
3. Сервис, работающий не в реальном масштабе времени, с переменной битовой скоростью.
4. Сервис с обязательством приложения максимальных усилий по предоставлению услуг.

Все предоставляемые стандартом 802.16 сервисы ориентированы на соединение, и каждое соединение получает доступ к одному из приведенных ранее классов сервиса. Это определяется при установке связи. Такое решение сильно отличается как от 802.11, так и от Ethernet, где отсутствовали какие-либо намеки на установление соединения на подуровне MAC.

Сервис с постоянной битовой скоростью предназначен для передачи несжатой речи, такой, какая передается по каналу T1. Здесь требуется передавать предопределенный объем данных в предопределенные временные интервалы. Это реализуется путем назначения каждому соединению такого типа своих интервалов. После того как канал оказывается распределенным, доступ к временным интервалам осуществляется автоматически, и нет необходимости запрашивать каждый из них по отдельности.

Сервис реального масштаба времени с переменной битовой скоростью применяется при передаче сжатых мультимедийных данных и других программных

приложений реального времени. Необходимая в каждый момент времени пропускная способность может меняться. Та или иная полоса выделяется базовой станцией, которая опрашивает через определенные промежутки времени абонента с целью выявления необходимой на текущий момент ширины канала.

Сервис, работающий не в реальном масштабе времени, с переменной битовой скоростью предназначен для интенсивного трафика — например, для передачи файлов большого объема. Здесь базовая станция тоже опрашивает абонентов довольно часто, но не в строго установленные моменты времени. Абонент, работающий с постоянной битовой скоростью, может установить в единицу один из специальных битов своего кадра, тем самым предлагая базовой станции опросить его (это означает, что у абонента появились данные, которые нужно передать с новой битовой скоростью).

Если станция не отвечает на k опросов подряд, базовая станция включает ее в широковещательную группу и прекращает персональные опросы. Теперь если станции потребуется передать данные, то во время широковещательного опроса она должна ответить базовой станции, запрашивая тем самым сервис. Таким образом, станции с малым трафиком не отнимают у базовой станции ценное время на персональные опросы.

Наконец, сервис с обязательством приложения максимальных усилий используется для всех остальных типов передачи. Никаких опросов здесь нет, а станции, желающие захватить канал, должны соперничать с другими станциями, которым требуется тот же класс сервиса. Запрос пропускной способности осуществляется во временных интервалах, помеченных в карте распределения исходящего потока как доступные для конкуренции. Если запрос прошел удачно, это будет отмечено в следующей карте распределения входящего потока. В противном случае абоненты-неудачники должны продолжать борьбу. Для минимизации числа коллизий используется взятый из Ethernet алгоритм двоичного экспоненциального отката.

Стандартом определены две формы распределения пропускной способности: для станции и для соединения. В первом случае абонентская станция собирает вместе все требования своих абонентов (например, компьютеров, принадлежащих жильцам здания) и осуществляет коллективный запрос. Получив полосу, она распределяет ее между пользователями по своему усмотрению. В последнем случае базовая станция работает с каждым соединением отдельно.

Стандарт 802.16: структура кадра

Все кадры подуровня управления доступом к среде (MAC) начинаются с одного и того же заголовка. За ним следует (или не следует) поле данных, и кончается кадр также необязательным полем контрольной суммы (CRC). Это показано на рис. 4.31. Поле данных отсутствует в служебных кадрах, которые предназначены, например, для запроса временных интервалов. Контрольная сумма (как ни странно) тоже является необязательной благодаря тому, что исправление ошибок производится на физическом уровне и никогда не бывает попыток повторно переслать кадры информации, передающейся в реальном масштабе времени. Так если

все равно нет повторных передач, зачем же беспокоить аппаратуру вычислением и проверкой контрольных сумм?



Рис. 4.31. Обычный кадр (*а*); кадр запроса канала (*б*)

Давайте кратко рассмотрим поля заголовка (рис. 4.31, *а*). Бит *EC* говорит о том, шифруется ли поле данных. Поле *Тип* указывает тип кадра (в частности, сообщает о том, пакуется ли кадр и есть ли фрагментация). Поле *CI* указывает на наличие либо отсутствие поля финальной контрольной суммы. Поле *EK* сообщает, какой из ключей шифрования используется (если он вообще используется). В поле *Длина* содержится информация о полной длине кадра, включая заголовок. *Идентификатор соединения* сообщает, какому из соединений принадлежит кадр. В конце заголовка имеется поле *Контрольная сумма заголовка*, значение которого вычисляется с помощью полинома $x^8 + x^2 + x + 1$.

На рис. 4.31, *б* показан другой тип кадра. Это кадр запроса канала. Он начинается с единичного, а не нулевого бита и в целом напоминает заголовок обычного кадра, за исключением второго и третьего байтов, которые составляют 16-битный номер, говорящий о требуемой полосе для передачи соответствующего числа байтов. В кадре запроса канала отсутствует поле данных, нет и контрольной суммы всего кадра.

Можно долго говорить о стандарте 802.16, но все-таки не здесь. За дополнительной информацией обращайтесь, пожалуйста, к официальному описанию стандарта.

Bluetooth

В 1994 году компания Л. М. Эриксона (L. M. Ericsson) заинтересовалась вопросом беспроводной связи между мобильными телефонами и другими устройствами (например, PDA). Совместно с четырьмя другими небезызвестными компаниями (IBM, Intel, Nokia и Toshiba) была сформирована специальная группа (то есть консорциум), которая занялась развитием стандарта беспроводной связи

вычислительных устройств и устройств связи, а также созданием аксессуаров, использующих недорогие маломощные радиоустройства небольшого радиуса действия. Проект был назван **Bluetooth** («Синий зуб») в честь великого короля викингов по имени Гаральд Синий Зуб II (940–981), который объединил (читай, завоевал) Данию и Норвегию. Ну да, он тоже сделал это без помощи проводов.

Изначально идея состояла в том, чтобы избавиться от надоевших кабелей между устройствами, однако затем системы стали расширяться и вторгаться во владения беспроводных ЛВС. Поскольку такой поворот событий действительно делает стандарт более полезным, его даже стали прочить в конкуренты стандарту 802.11. Чтобы еще больше усугубить конкуренцию, системы сделали близкими не только на уровне идей, но и на уровне электрических сигналов. Нелишним также будет отметить и тот факт, что компания Hewlett-Packard несколько лет назад представила миру инфракрасную сеть для объединения компьютерной периферии без использования проводов. Однако эта идея не получила широкого распространения.

Неустрашимая специальная группа Bluetooth несмотря на все это выпустила в июле 1999 года 1500-страничную спецификацию V1.0. Вскоре после этого группа стандартизации IEEE взяла данный документ за основу стандарта 802.15 (персональные сети) и начала кромсать его, подгоняя под свои нужды. Может поначалу показаться странной идея разработки стандарта того, что уже стандартизовано, причем очень детально специфицировано, особенно если не видно никаких несовместимых систем, которые надо было бы совмещать. Однако опыт показывает, что открытые стандарты, созданные такими нейтральными организациями, как IEEE, зачастую способствуют популяризации и развитию новых технологий. Необходимо сделать одно уточнение: спецификация Bluetooth охватывает целую систему, от физического до прикладного уровня. Комитет IEEE 802.15 занимается стандартизацией только лишь физического уровня и уровня передачи данных; часть стека протоколов, касающаяся других уровней, не входит в его компетенцию.

Несмотря на то, что в 2002 году IEEE утвердил первый стандарт персональных сетей, 802.15.1, специальная группа Bluetooth по-прежнему активно работает и занимается улучшениями системы. Хотя на данный момент Bluetooth и стандарт IEEE не совсем идентичны, ожидается, что скоро произойдет их объединение.

Архитектура Bluetooth

Начнем изучение системы Bluetooth с краткого обзора того, из чего она состоит и для чего предназначена. Основу Bluetooth составляет **пикосеть** (piconet), состоящая из одного главного узла и нескольких (до семи) подчиненных узлов, расположенных в радиусе 10 м. В одной и той же комнате, если она достаточно большая, могут располагаться несколько пикосетей. Более того, они могут даже связываться друг с другом посредством моста (специального узла), как показано на рис. 4.32. Несколько объединенных вместе пикосетей составляют **рассеянную сеть** (scatternet).

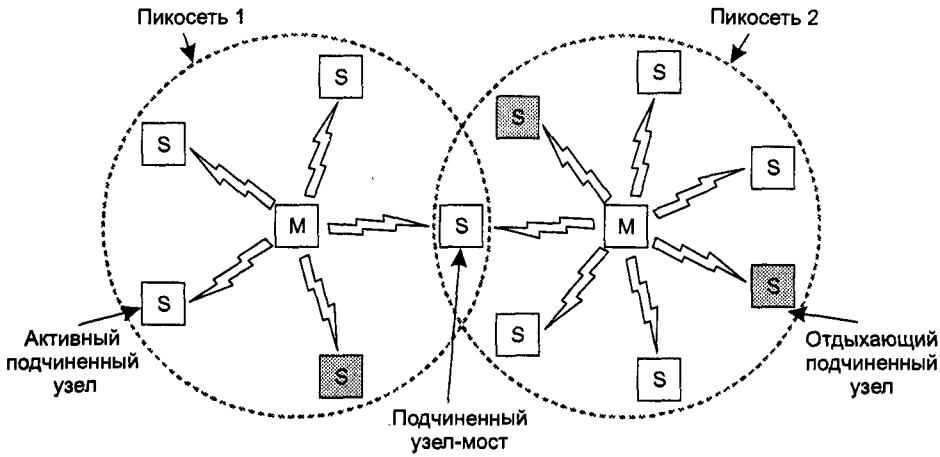


Рис. 4.32. Две пикосети могут, соединившись, сформировать рассеянную сеть

Помимо семи активных подчиненных узлов, один главный узел может поддерживать до 255 так называемых отдыхающих узлов. Это устройства, которые главный узел перевел в режим пониженного энергопотребления — за счет этого продлевается ресурс их источников питания. В таком режиме узел может только отвечать на запросы активации или на сигнальные последовательности от главного узла. Существуют еще два промежуточных режима энергопотребления — приостановленный и анализирующий, но мы их сейчас рассматривать не будем.

Такое решение с главным и подчиненным узлами оказалось очень простым и дешевым в реализации (вся микросхема Bluetooth стоит менее $5). Поскольку этого и добивались разработчики, такой вариант и был принят. Последствием этого является то, что подчиненные узлы получились очень неразговорчивыми — они лишь выполняют то, что им прикажет главный узел. В основе пикосетей лежит принцип централизованной системы с временным уплотнением. Главный узел контролирует временные интервалы и распределяет очередность передачи данных каждым из подчиненных узлов. Связь существует только между подчиненным и главным узлами. Прямой связи между подчиненными узлами нет.

Приложения Bluetooth

Большинство сетевых протоколов просто предоставляют каналы связи между коммуникационными единицами и оставляют прикладное использование этих каналов на усмотрение разработчиков. Например, в стандарте 802.11 ничего не говорится о том, что пользователи должны использовать свои ноутбуки для чтения электронной почты, работы в Интернете и т. п. В противоположность этому спецификация Bluetooth V1.1 называет 13 конкретных поддерживаемых приложений и для каждого из них предоставляет свой набор протоколов. К сожалению, это приводит к сильному усложнению системы, поэтому многие детали в нашем описании мы вынуждены опустить. Тринадцать приложений, называемые про-

филями, перечислены в табл. 4.4. Из этой таблицы становятся понятны цели, которые ставили перед собой разработчики системы.

Таблица 4.4. Профили Bluetooth

Название	Описание
Общий доступ	Процедуры управления связью
Определение сервиса	Протокол для определения предлагаемых сервисов
Последовательный порт	Альтернатива кабелю последовательного порта
Общий объектный обмен	Определяет клиент-серверные взаимоотношения при передаче объектов
Доступ к ЛВС	Протокол связи между мобильным компьютером и стационарной ЛВС
Удаленный доступ	Позволяет ноутбуку получать удаленный доступ посредством мобильного телефона
Факс	Позволяет мобильному факсу связываться с мобильным телефоном
Беспроводная телефония	Связывает трубку с локальной базовой станцией
Intercom	Цифровые переносные рации
Гарнитура	Обеспечивает связь устройств hands-free с телефонами
Передача объектов	Обеспечивает обмен простыми объектами
Передача файлов	Предоставляет легкий способ пересылки файлов
Синхронизация	Позволяет PDA синхронизироваться с другим компьютером

Профиль группового доступа на самом деле не является приложением. Это скорее та основа, на которой строятся реальные приложения. Его главная задача состоит в обеспечении установки и поддержки защищенной от несанкционированного доступа связи (создании канала) между главным и подчиненным узлами. Также довольно общим является профиль определения сервиса, используемый для определения того, какие сервисы могут быть предоставлены другими устройствами. Вся аппаратура системы Bluetooth должна поддерживать два описанных ранее профиля. Все прочие являются необязательными.

Профиль последовательного порта — это транспортный протокол, который используется большинством других профилей. Он эмулирует последовательный канал и особенно полезен при работе с приложениями, которым требуется этот канал.

Профиль общего объектного обмена определяет клиент-серверные взаимоотношения, возникающие при обмене данными. Клиенты инициируют операции, но подчиненная станция может выступать либо в роли клиента, либо в роли сервера. Как и профиль последовательного порта, это один из кирпичиков, из которых строятся другие профили.

Следующая группа, состоящая из трех профилей, имеет отношение к сетям. Профиль доступа к ЛВС позволяет устройству Bluetooth подсоединиться к стационарной вычислительной сети. Этот профиль является самым настоящим конкурентом стандарта 802.11. Профиль удаленного доступа был, собственно говоря,

тем, ради чего изначально был задуман весь проект. Он позволяет ноутбуку соединяться с мобильным телефоном, имеющим встроенный модем, без использования проводов. Профиль «Факс» похож на предыдущий с той разницей, что он позволяет беспроводным факс-машинам отсылать и получать факсы при помощи мобильного телефона. Опять же, никакие провода для связи между устройствами не требуются.

Следующие три профиля относятся к телефонии. Профиль беспроводной телефонии обеспечивает связь телефонной трубки с базой. Сейчас домашний телефон не может использоваться в качестве мобильного, даже если он не имеет совсем никаких проводов, однако в будущем, надо полагать, эти два устройства будут объединены. Профиль Intercom позволяет двум телефонам соединяться друг с другом наподобие раций. Наконец, последний профиль этой группы представляет собой приложение, позволяющее устройствам hands-free держать связь с базой (телефоном). Это удобно, например, при управлении автомобилем.

Оставшиеся три профиля предназначены для организации обмена объектами (данными) между беспроводными устройствами. Объекты могут представлять собой электронные визитные карточки, изображения или файлы с данными. В частности, профиль синхронизации предназначен для загрузки данных в PDA или ноутбук, когда владелец этих устройств, например, выходит из дома. После возвращения данные можно извлечь.

Неужели действительно необходимо было так подробно описывать в стандарте все приложения и предоставлять наборы протоколов для каждого из них? Может быть, и нет, но было создано довольно много рабочих групп, занимавшихся различными аспектами применения системы. Каждая рабочая группа разработала свой профиль. Считайте это демонстрацией закона Конвея в действии. (В апреле 1968 года в журнале *Datamation* была опубликована статья Мелвина Конвея (Melvin Conway), в которой утверждалось, что если поручить написание компилятора n программистам, то получится n-проходный компилятор. В более общем виде эта мысль звучит так: структура программного обеспечения отражает структуру группы разработчиков.) Наверное, можно было обойтись не тринадцатью, а двумя наборами протоколов (один для передачи файлов и один для передачи данных в реальном масштабе времени).

Bluetooth: набор протоколов

Стандарт Bluetooth включает в себя множество протоколов, довольно свободно разбитых на уровни. Структура стека протоколов не следует ни модели OSI, ни TCP/IP, ни 802, ни какой-либо другой известной модели. Тем не менее IEEE работает над тем, чтобы как-то вписать Bluetooth в модель 802. Базовая архитектура протоколов в модифицированном комитетом 802 виде представлена на рис. 4.33.

В самом низу находится физический (радиотехнический) уровень, который вполне соответствует представлениям моделей OSI и 802 о том, каким должен быть этот уровень. На нем описываются радиосвязь и применяемые методы мо-

дуляции. Многое здесь направлено на то, чтобы сделать систему как можно дешевле и доступнее массовому покупателю.

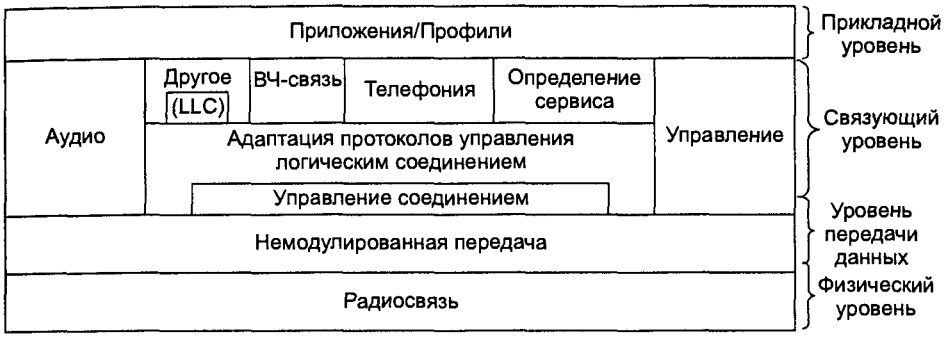


Рис. 4.33. Архитектура протоколов Bluetooth: версия 802.15

Уровень немодулированной передачи чем-то напоминает подуровень MAC, но включает в себя и некоторые элементы физического уровня. Здесь описывается то, как главный узел управляет временными интервалами и как эти интервалы группируются в кадры.

Далее следует уровень, содержащий группу связанных между собой протоколов. Протокол управления соединением устанавливает логические каналы между устройствами, управляет режимами энергопотребления, идентификацией, а также качеством предоставляемых услуг. Подуровень адаптации протоколов управления логическими соединениями (часто называемый L2CAP) скрывает от верхних уровней технические детали нижних уровней. Его можно считать аналогом обычного подуровня LLC из семейства стандартов 802. Однако устроен он по-другому. Аудиопротоколы и протоколы управления, как следует из их названия, занимаются соответственно передачей звука и управлением. Приложения могут обращаться к ним напрямую, минуя протокол L2CAP.

Следующий уровень называется связующим, он содержит множество разнообразных протоколов. Для совместимости с сетями 802.x IEEE поместил сюда, например, LLC. Впрочем, протоколы высокочастотной (ВЧ) связи, телефонии и определения сервисов всегда принадлежали Bluetooth. Протокол ВЧ-связи эмулирует работу стандартного последовательного порта ПК, к которому обычно подключаются клавиатура, мышь, модем и другие устройства. Он позволяет последователям этих традиционных устройств нетрадиционно обходиться без проводов. Протокол телефонии является протоколом, работающим в реальном масштабе времени. Он используется тремя соответствующими профилями, ориентированными на передачу речи. В его задачи входят установка и разрыв телефонного соединения. Наконец, протокол определения сервисов используется для поиска доступных в сети сервисов.

На самом верхнем уровне находятся приложения и профили. Они работают с протоколами нижних уровней, которые обеспечивают их функционирование. Каждому приложению сопоставлен свой набор протоколов. Специфические уст-

ройства типа гарнитур используют только те протоколы, которые необходимы для их работы.

В следующих разделах мы рассмотрим три нижних уровня стека протоколов Bluetooth, поскольку они, пусть грубо, но все-таки соответствуют физическому уровню и подуровню MAC.

Bluetooth: уровень радиосвязи

Уровень радиосвязи переносит информацию бит за битом от главного узла к подчиненным и обратно. В реальности это маломощная приемопередающая система с радиусом действия порядка 10 м. Она работает в нелицензируемом диапазоне 2,4 ГГц. Диапазон разделен на 79 каналов по 1 МГц в каждом. В качестве метода модуляции применяется частотная манипуляция с 1 битом на герц, что дает суммарную скорость 1 Мбит/с. Однако большая часть спектра занята служебной информацией. Для распределения каналов применяется расширенный спектр со скачкообразной перестройкой частоты (1600 скачков частоты в секунду, время пребывания — 625 мкс). Все узлы пикосетей перестраивают частоты одновременно; последовательность частот генерируется главным узлом.

Поскольку сети 802.11 и Bluetooth работают в одном и том же нелицензируемом диапазоне 2,4 ГГц на одних и тех же 79 каналах, то, естественно, они интерферируют друг с другом. Однако Bluetooth развивается гораздо стремительнее 802.11, и похоже, что именно Bluetooth будет хоронить 802.11, а не наоборот. Но и 802.11, и 802.15 являются стандартами IEEE, и конечно, IEEE пытается найти какой-нибудь выход из этой пикантной ситуации. Но это не такая простая задача, как кажется: дело в том, что обе системы используют данный диапазон по той причине, что он не подлежит лицензированию. Стандарт 802.11a работает в другом нелицензируемом диапазоне (5 ГГц), но он гораздо уже, благодаря физическим свойствам радиоволн. Так что 802.11a оказывается далеко не панацеей от всех случаев. Некоторые компании решили проблему радикально: они просто предали Bluetooth анафеме. Решение вполне в духе рыночных отношений: более сильная (в политическом и экономическом, а не техническом смысле) технология требует от более слабой изменить свой стандарт таким образом, чтобы она не мешалась под ногами. Некоторые мысли по этому поводу можно также найти у (Lansford и др., 2001).

Bluetooth: уровень немодулированной передачи

Уровень немодулированной передачи — это наиболее близкий к MAC-подуровню элемент иерархии Bluetooth. Он трансформирует простой поток бит в кадры и определяет некоторые ключевые форматы. В простейшем случае главный узел каждой пикосети выдает последовательности временных интервалов по 625 мкс, причем передача данных со стороны главного узла начинается в четных тактах, а со стороны подчиненных узлов — в нечетных. Это, по сути дела, традиционное временное уплотнение, в котором главная сторона получает одну половину времен-

ных интервалов, а подчиненные делят между собой вторую. Кадры могут быть длиной 1, 3 или 5 тактов.

Тактирование со скачкообразным изменением частоты отводит на успокоение системы 250–260 мкс при каждом скачке. Время успокоения можно и уменьшить, но цена за это довольно велика. В кадре, состоящем из одного такта, после успокоения системы остается время на передачу 366 из 625 бит. Из них 126 бит уходит на код доступа и заголовок. Остается 240 бит полезной информации на кадр. Если совместить пять временных интервалов в один кадр, понадобится всего один период успокоения системы, поэтому из $5 \cdot 625 = 3125$ бит доступны 2781 бит на уровне немодулированной передачи. Получается, что длинные кадры эффективнее однотактовых.

Все кадры передаются между главным и подчиненными узлами по логическому каналу, называемому **соединением**. Существует два типа соединений. Первый называется **ACL** (Asynchronous Connectionless — асинхронный без установления связи), он используется для коммутации пакетов данных, которые могут появиться в произвольный момент времени. Такие данные появляются с уровня L2CAP на передающей стороне и доставляются на тот же уровень на принимающей стороне. Трафик ACL доставляется по принципу максимальных прилагаемых усилий для обеспечения сервиса. Никаких гарантий не дается. Кадры могут теряться и пересылаться повторно. У подчиненного узла может быть только одно ACL-соединение со своим главным узлом.

Второй вид соединения называется **SCO** (Synchronous Connection Oriented — синхронный с установлением связи). Он предназначен для передачи данных в реальном масштабе времени — это требуется, например, при телефонных разговорах. Такой тип канала получает фиксированный временной интервал для передачи в каждом из направлений. Из-за критичной ко времени передачи природы SCO кадры, переданные по данному типу канала, никогда не пересылаются заново. Вместо этого может быть использована прямая коррекция ошибок, обеспечивающая более надежное соединение. У подчиненного узла может быть до трех соединений типа SCO с главным узлом, каждое из которых представляет собой аудиоканал PCM с пропускной способностью 64 000 бит/с.

Bluetooth: уровень L2CAP

Уровень L2CAP выполняет три основные функции. Во-первых, он принимает пакеты размером до 64 Кбайт с верхних уровней и разбивает их на кадры для передачи по физическому каналу. На противоположном конце этот же уровень протоколов используется для обратного действия — объединения кадров в пакеты.

Во-вторых, L2CAP занимается мультиплексированием и демультиплексированием множества источников пакетов. После сборки пакета он определяет, куда следует направить пакет (например, на протоколу ВЧ-связи или телефонии).

В-третьих, L2CAP отвечает за качество обслуживания как во время передачи, так и во время ожидания. Во время установления соединения он договаривается о максимально разрешенном количестве передаваемой полезной информации, что позволяет избежать заваливания данными устройства, которому приходится

работать с маленькими пакетами. Это действие необходимо потому, что далеко не все участники сети могут работать с 64-килобайтными пакетами.

Bluetooth: структура кадра

Существует несколько форматов кадров, наиболее важный из которых показан на рис. 4.34. В начале кадра указывается код доступа, который обычно служит идентификатором главного узла. Это позволяет двум главным узлам, которые расположены достаточно близко, чтобы «слышать» друг друга, различать, кому из них предназначаются данные. Затем следует 54-битный заголовок, в котором содержатся поля, характерные для кадра подуровня MAC. Далее расположено поле данных, размер которого ограничен 2744 битами (для передачи из пяти тактов). Если кадр имеет длину, соответствующую одному тактовому интервалу, то формат остается таким же, с той разницей, что поле данных в этом случае составляет 240 бит.

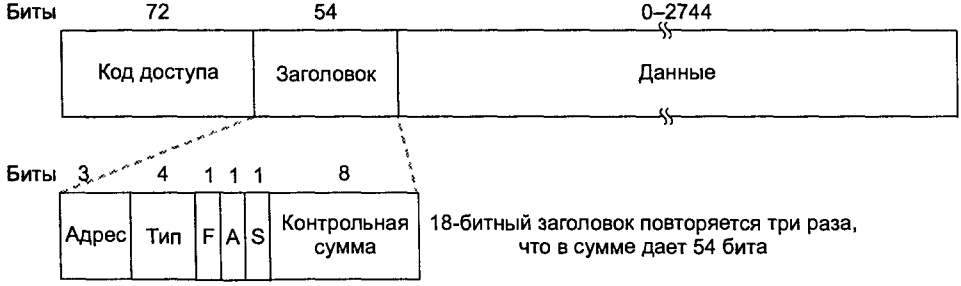


Рис. 4.34. Типичный информационный кадр Bluetooth

Рассмотрим, из чего состоит заголовок кадра. Поле *Адрес* идентифицирует одно из восьми устройств, которому предназначена информация. Поле *Тип* определяет тип передаваемого кадра (ACL, SCO, опрос или пустой кадр), метод коррекции ошибок и количество временных интервалов, из которых состоит кадр. Бит *F* (Flow — поток) выставляется подчиненным узлом и сообщает о том, что его буфер заполнен. Это такая примитивная форма управления потоком. Бит *A* (Acknowledgement — подтверждение) представляет собой подтверждение (ACK), отсылаемое заодно с кадром. Бит *S* (Sequence — последовательность) используется для нумерации кадров, что позволяет обнаруживать повторные передачи. Это протокол с ожиданием, поэтому 1 бита действительно оказывается достаточно. Далее следует 8-битная контрольная сумма заголовка. Весь 18-битный заголовок кадра повторяется трижды, что в итоге составляет 54 бита, как показано на рис. 4.34. На принимающей стороне несложная схема анализирует все три копии каждого бита. Если они совпадают, бит принимается таким, какой он есть. В противном случае все решает большинство. Как видите, на передачу 10 бит тратится в данном случае 54 бита. Причина очень проста: за все нужно платить. За обеспечение передачи данных с помощью дешевых маломощных устройств

(2,5 мВт) с невысокими вычислительными способностями приходится платить большой избыточностью.

В кадрах ACL применяются разные форматы поля данных. Самый простой формат — в кадрах SCO: длина поля данных всегда равна 240 бит. Возможны три варианта: 80, 160 или 240 бит полезной информации. При этом оставшиеся биты поля данных используются для исправления ошибок. Самая надежная версия (80 бит полезной информации) устроена очень просто: одно и то же содержимое повторяется три раза (что и составляет 240 бит). То есть метод здесь применяется тот же, что и в заголовке кадра.

Поскольку подчиненные узлы могут использовать только нечетные временные интервалы, им достается 800 интервалов в секунду. Столько же получает и главный узел. При 80 битах полезных данных, передающихся в одном кадре, емкость канала подчиненного узла равна 64 000 бит/с. Этому же значению равна и емкость канала главного узла. Этого как раз хватает для организации полнодуплексного PCM-канала голосовой связи (именно поэтому 1600 скачков в секунду было выбрано в качестве скорости перестройки частот). Все эти цифры говорят о том, что полнодуплексный канал со скоростью 64 000 бит/с в каждую сторону при самом надежном способе передачи информации вполне устраивает пикосеть, невзирая на то, что суммарная скорость передачи данных на физическом уровне равна 1 Мбит/с. При самом ненадежном варианте (240 бит информации на кадр) можно организовать три полнодуплексных голосовых канала одновременно. Именно по этой причине для подчиненного узла максимальное количество соединений типа SCO равно трем.

О системах Bluetooth можно было бы рассказать еще много интересного, но объем книги все-таки ограничен, и нам придется остановиться на этом. Дополнительную информацию можно узнать в следующих изданиях (Bhagwat, 2001; Bisdikian, 2001; Bray and Sturman, 2002; Haartsen, 2000; Johansson и др., 2001; Miller and Bisdikian, 2001; Sairam и др., 2002).

Коммутация на уровне передачи данных

У многих организаций имеется по несколько локальных сетей, которые необходимо объединять между собой. Локальные сети могут быть объединены с помощью специальных устройств, называемых **мостами**, которые работают на уровне передачи данных. Они анализируют адреса, содержащиеся в кадрах этого уровня, и в соответствии с ними осуществляют маршрутизацию. Поскольку мосты не исследуют сами данные, передающиеся в кадрах, то они одинаково хорошо справляются с пакетами IPv4 (используемыми сейчас в Интернете), IPv6 (которые будут использоваться в Интернете позже), AppleTalk, ATM, OSI и многими другими. В отличие от мостов, маршрутизаторы анализируют адреса *пакетов* и работают, основываясь на этой информации. Хотя кажется, что можно провести очень четкое разделение между мостами и маршрутизаторами, некоторые современные разработки, такие как коммутируемый Ethernet, несколько замутили воду, как мы увидим далее. В следующих разделах мы рассмотрим мосты и коммутаторы, особое внимание обратив на те из них, которые связывают различные типы ЛВС се-

рии 802. Подробную информацию о мостах, коммутаторах и т. п. можно найти у (Perlman, 2000).

Прежде чем перейти к обсуждению мостов, стоит рассмотреть несколько часто встречающихся ситуаций, в которых они используются. Перечислим шесть причин, по которым в организации может появиться несколько локальных сетей.

Во-первых, у многих университетов и отделов корпораций есть свои локальные сети, соединяющие персональные компьютеры, рабочие станции и серверы. Поскольку цели различных факультетов или отделов различны, то и объединение в локальные сети часто происходит по факультетам и отделам, которым не очень интересно, как построена сеть у соседей. Однако рано или поздно им требуется взаимодействие, поэтому появляется необходимость в мостах. В данном примере несколько отдельных локальных сетей образовалось вследствие автономности их владельцев.

Во-вторых, организации могут размещаться в нескольких зданиях, значительно удаленных друг от друга. Может оказаться дешевле создать несколько отдельных локальных сетей и затем соединить их с помощью мостов и лазерных линий связи, вместо того чтобы протягивать коаксиальный кабель по всей территории.

В-третьих, иногда бывает необходимо логически разделить одну локальную сеть на несколько, чтобы снизить нагрузку. Так, например, во многих университетах в сети объединены тысячи рабочих станций, на которых работают студенты и сотрудники. Файлы обычно хранятся на файл-серверах и загружаются на машины пользователей по мере необходимости. Огромные масштабы не позволяют объединить все эти рабочие станции в одну локальную сеть, так как для этого потребовалась бы слишком высокая суммарная пропускная способность. Вместо этого формируются несколько локальных сетей, соединенных мостами, как показано на рис. 4.35. В каждую локальную сеть входят несколько рабочих станций и свой файловый сервер, поэтому основной трафик ограничивается рамками локальной сети и не нагружает магистраль.

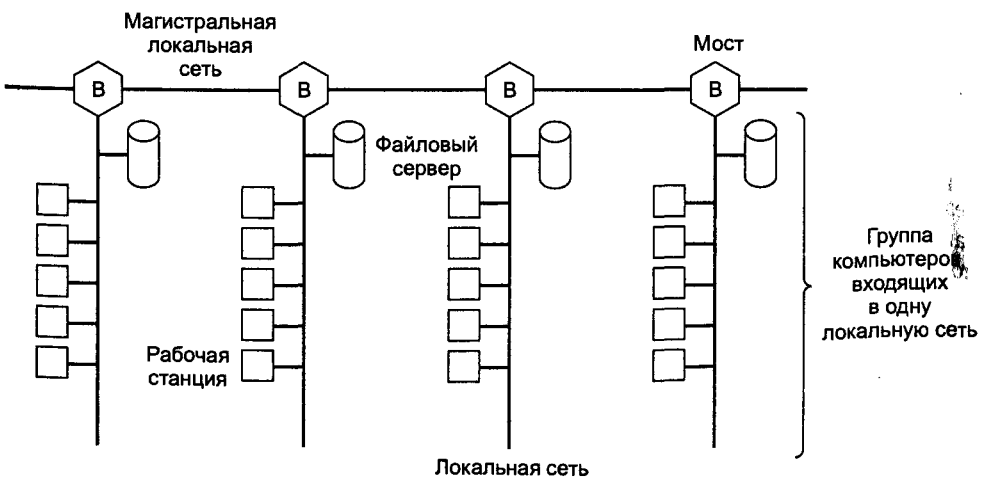


Рис. 4.35. Система из нескольких локальных сетей, соединенных магистралью, справляется с нагрузкой, сильно превышающей возможности отдельной ЛВС

Надо отметить, что несмотря на изображение ЛВС в форме моноканалов (рис. 4.35, классический вид ЛВС), в наше время они чаще реализуются с применением концентраторов или специальных коммутаторов. Тем не менее, длинный моноканал с большим количеством машин, присоединенных к нему, и концентратор, соединенный с примерно тем же числом машин, функционально идентичны. В обоих случаях все компьютеры образуют одну область коллизий, и все используют протокол CSMA/CD для отправки кадров. А вот коммутируемые локальные сети отличаются от данных типов сетей. Об этих отличиях пойдет речь далее.

В-четвертых, в некоторых ситуациях единая локальная сеть может оказаться адекватной в терминах нагрузки, но нереализуемой по причине большого расстояния между отдельными машинами (более 2,5 км для 802.3). Даже если прокладка кабеля не вызывает особых проблем, сеть работать не будет из-за слишком большой задержки двойного оборота. В данном случае единственным решением является создание нескольких отдельных сетей, соединенных мостами. Таким образом, общая дистанция, покрываемая сетью, может быть увеличена.

В-пятых, система из нескольких локальных сетей надежнее, чем единая локальная сеть, в которой один вышедший из строя узел, передающий в линию нескончаемый поток мусора, может вывести из строя всю сеть. С помощью мостов, поставленных в особо критичных местах, можно разбить единую локальную сеть на независимые в терминах отказоустойчивости отсеки. В этом случае мосты работают пожарными дверями, которые разделяют части здания. В данном случае они исключают ситуацию, при которой выход из строя одного узла привел бы к неработоспособности всей сети. В отличие от повторителя, мосты не передают из одной сети в другую всю информацию подряд, а делают это выборочно, причем можно программно настроить мост, указав ему, что передавать, а что нет.

Наконец, в-шестых, мосты увеличивают закрытость и безопасность отдельных локальных сетей. Большинство локальных сетей могут работать в так называемом **беспорядочном режиме**, в котором компьютеру передаются *все* кадры, а не только адресованные ему. Шпионы, хакеры и другие плохие люди любят это свойство локальных сетей. Если установить мосты и настроить их соответствующим образом, можно защититься от утечки информации за пределы локальной сети и от вторжения извне.

В идеале мосты должны быть полностью прозрачны. Это означает, что любые передвижения машины из одного сегмента сети в другой должны происходить без каких-либо изменений аппаратуры, программного обеспечения или конфигурационных таблиц. Кроме того, машины любого сегмента должны иметь возможность общаться с машинами других сегментов независимо от используемых в сегментах и между ними типов ЛВС. Этой цели удается достичь , но лишь изредка.

Мосты между 802.x и 802.y

Рассмотрев причины, по которым могут понадобиться мосты, рассмотрим принципы их работы. На рис. 4.36 показана работа простого двухпортового моста. Хост *A* беспроводной локальной сети (802.11) хочет передать пакет на стационар-

ный хост *B* сети Ethernet (802.3), с которой сеть 802.11 объединена при помощи моста. Этот пакет спускается на подуровень управления логическим соединением (LLC) и получает заголовок этого подуровня (показан на рисунке черным прямоугольником). После этого он передается подуровню управления доступом к носителю (MAC), и там к нему добавляется заголовок стандарта 802.11 (а также концевик, не показанный на рисунке). Затем этот кадр передается в эфире и улавливается базовой станцией, которая видит, что его нужно передать стационарной сети Ethernet. Кадр достигает моста, соединяющего сети 802.11 и 802.3, на физическом уровне. На подуровне MAC моста от кадра отрезается заголовок стандарта 802.11. В таком обезглавленном виде (только с заголовком LLC-уровня) он передается подуровню LLC моста. В данном примере пакет предназначается узлу сети 802.3, поэтому он спускается по уровням Ethernet-стороны моста. После этого пакет уходит в сеть Ethernet. Обратите внимание на то, что у моста, соединяющего *k* разнотипных сетей, будет *k* подуровней MAC и столько же физических уровней.

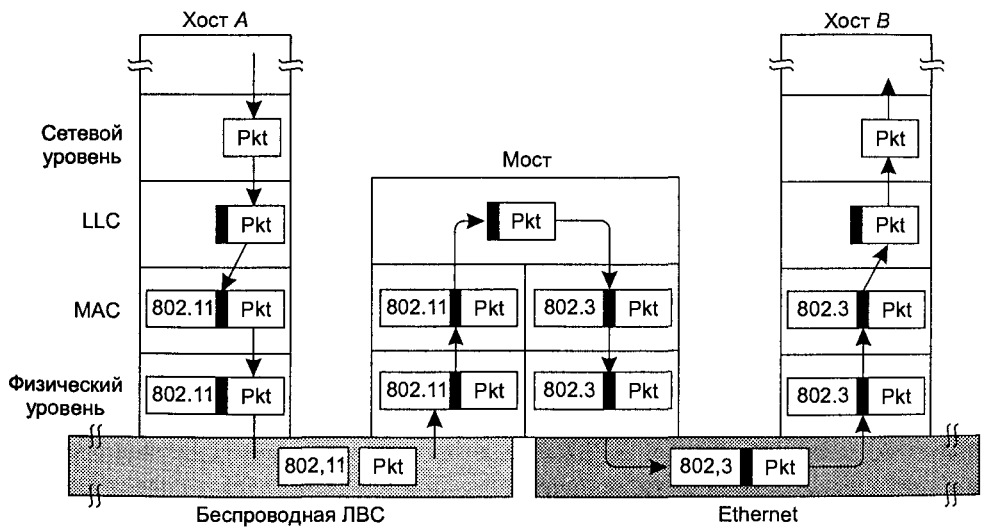


Рис. 4.36. Передача пакета по мосту из локальной сети 802.11 в сеть 802.3

Кажется, что передача кадра из одной локальной сети в другую происходит очень просто. К сожалению, это не так. В оставшихся разделах этой главы мы обсудим те трудности, которые могут встретиться при создании моста, соединяющего различные локальные (и региональные) сети стандарта 802. Особое внимание мы сосредоточим на сетях 802.3, 802.11 и 802.16, хотя существуют и другие мосты. С каждой из комбинаций соединения сетей связаны свои проблемы.

Во-первых, все объединяемые локальные сети используют различные форматы кадров (рис. 4.37). Не в пример различиям между Ethernet, маркерной шиной и маркерным кольцом, которые образовались по историческим и политическим причинам, в данном случае в этих различиях есть определенный смысл. Напри-

мер, поле *Длительность* в кадре стандарта 802.11 оказалось там благодаря применению протокола MACAW; включать его в Ethernet-кадр было бы бессмысленно. В результате любые переносы из сети в сеть требуют переформатирования, на что, в свою очередь, требуется процессорное время, к тому же необходимо заново подсчитать контрольную сумму. Испорченные биты, находящиеся в памяти моста, могут привести к тому, что ошибки не будут распознаны.



Рис. 4.37. Форматы кадров стандартов IEEE 802. Масштаб не выдержан

Во-вторых, скорости передачи данных объединяемых сетей не обязательно одинаковы. При пересылке длинной серии кадров из быстрой сети в медленную мост не сможет отправлять кадры дальше с той же скоростью, с которой они приходят. Например, если гигабитный Ethernet посылает поток битов в локальную сеть 802.11b, имеющую 11 Мбит/с, с нормальной для себя скоростью, мосту придется буферизировать их в надежде на то, что ему хватит при этом памяти. Мосты, объединяющие три и более локальных сетей, сталкиваются с аналогичной проблемой, когда несколько сетей пытаются одновременно передавать кадры в какую-либо одну сеть, даже если все они работают с одинаковыми скоростями.

В-третьих, возможно, самая серьезная проблема заключается в том, что разные локальные сети стандарта 802.x имеют различную максимальную длину кадра. Очевидно, что если пытаться передать длинный кадр в сеть, не способную его обработать, возникнет неловкая ситуация. Разбиение кадра на отдельные части не входит в компетенцию протокола данного уровня. Все протоколы подразумевают, что кадр либо пришел, либо нет. Они не могут принимать кадры частями. Вовсе не утверждается, что таких протоколов не может быть в принципе. Они должны существовать, и они существуют. Дело в том, что никакие протоколы уровня передачи данных не поддерживает такой возможности, поэтому мосты должны держаться подальше от анализа полезной информации; содержащейся в кадрах. Таким образом, решения данной проблемы в рамках стандарта 802 нет. Слишком длинные кадры должны игнорироваться. Такова плата за прозрачность мостов.

Еще один вопрос связан с защитой информации. И 802.11, и 802.16 поддерживают шифрование на уровне передачи данных. А Ethernet не обладает такой возможностью. Это означает, что самые сложные и интересные сервисы, связан-

ные с криптографической защитой, предоставляемые в беспроводных сетях, становятся недоступны, если трафик проходит через Ethernet. Более того, если беспроводная станция шифрует данные на уровне передачи данных, то не будет никакой возможности расшифровать их при попадании в сеть 802.3. Если же такое шифрование не производилось, весь трафик в совершенно не защищенном виде попадет в эфир. Оба варианта оказываются чреватыми проблемами.

Одним из решением проблемы защиты информации является шифрование на более высоких уровнях, но в этом случае станция 802.11 должна знать, с кем она говорит — с другой станцией 802.11 (что означает использование шифрования на уровне передачи данных) или нет (то есть шифрование на уровне передачи данных отсутствует). Но форсирование совершения такого выбора нарушает прозрачность.

Наконец, есть что сказать и о качестве обслуживания в связи с мостами. И 802.11, и 802.16 обеспечивают его по-разному. В первом случае для этого существует режим PCF, в последнем — соединения с постоянной битовой скоростью. В Ethernet отсутствует определенная концепция качества обслуживания, поэтому трафик сети любого из этих двух стандартов при прохождении через Ethernet потеряет гарантию качества.

Локальное межсетевое взаимодействие

В предыдущем разделе обсуждались проблемы, возникающие при объединении различных локальных сетей стандарта IEEE 802.x с помощью одного моста. Однако в больших организациях, в которых имеется множество локальных сетей, их объединение связано с большим количеством проблем, даже если все ЛВС одного типа (например, Ethernet). В идеале было бы здорово купить в ближайшем магазине побольше мостов, предназначенных для стандарта IEEE, вставить в них нужные штекеры и сразу же получить прекрасно работающую сеть. И чтобы не надо было менять аппаратно-программное обеспечение, адресацию, конфигурационные таблицы... вообще ничего. Просто вставить кабель и начать работу. Кроме того, хотелось бы, чтобы появление мостов в системе не повлияло на работу уже существующих объединяемых сетей. Другими словами, мосты должны быть абсолютно прозрачными (невидимыми). Как ни странно, это не сон и не сказка. Это почти так и есть. Давайте разберемся, в чем тут секрет.

В простейшем случае прозрачный мост работает в беспорядочном режиме, принимая все кадры, передаваемые во всех локальных сетях, к которым он присоединен. В качестве примера рассмотрим конфигурацию, изображенную на рис. 4.38. Мост B1 соединен с сетями 1 и 2, а мост B2 соединен с сетями 2, 3 и 4. Кадр, появившийся на мосту B1 со стороны сети 1, предназначающийся станции *A*, может быть проигнорирован мостом, так как он уже находится в нужной сети, но кадр, появившийся на мосту B1 со стороны сети 1, предназначающийся станции *C* или *F*, должен быть переправлен на другую сторону.

При появлении кадра мост должен решить, игнорировать его или переправить, и если переправить, то в какую сеть. Выбор производится на основе адреса

получателя, который сравнивается с адресами, хранящимися в большой таблице моста. Таблица содержит адреса всех возможных получателей и номера сетей, в которых они содержатся. Например, в таблице моста B2 станция *A* числится принадлежащей к сети LAN 2, поскольку все, что требуется знать мосту B2, — это в какую сеть переслать кадр для станции *A*. То, что на самом деле этому кадру предстоит перейти еще через один мост, моста B2 не касается.

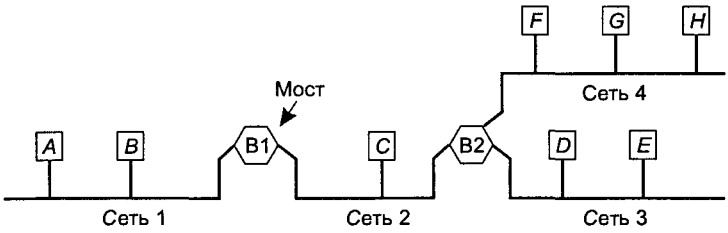


Рис. 4.38. Структура из четырех сетей и двух мостов

Когда мосты включаются первый раз, все их хэш-таблицы пусты. Ни один мост не знает, где находятся адресаты, поэтому они используют алгоритм заливки: каждый приходящий кадр с неизвестным адресом переправляется сразу по всем направлениям, кроме того, откуда он пришел. Со временем мосты узнают расположение адресатов. Кадры, расположение получателей которых известно, направляются только в одну нужную сеть.

Для обучения прозрачных мостов используется алгоритм так называемого **противоточного обучения**. Как уже упоминалось ранее, мосты работают в беспорядочном режиме, поэтому они видят все кадры, посылаемые во всех их сетях. Просматривая адреса отправителей, они могут определить, в какой сети находится станция, отправившая кадр. Например, если мост B1 на рис. 4.38 видит кадр, приходящий к нему по сети 2 от станции *C*, то он понимает, что станция *C* находится в сети 2, и делает соответствующую запись в своей таблице. Поэтому любой последующий кадр, адресованный станции *C* и приходящий по сети 1, будет переправляться в сеть 2, а приходящий по сети 2 — игнорироваться.

Топология сети может меняться по мере того как отдельные станции и мосты будут включаться, выключаться, а также перемещаться. Для поддержки динамической топологии в таблице помимо номера станции и номера сети указывается также время прибытия кадра от данной станции. При получении новых кадров это время обновляется. Таким образом, для каждой станции известно время последнего полученного от нее кадра.

Время от времени программа сканирует хэш-таблицу и удаляет все записи, сделанные ранее нескольких минут тому назад. Таким образом, если какой-либо компьютер был выключен, перенесен в новое место и включен снова, уже через несколько минут он сможет нормально работать, и для этого не потребуется никаких специальных действий. Обратная сторона такого алгоритма заключается в том, что кадры, направляемые какой-либо станции, молчавшей в течение нескольких минут, должны будут снова посылаться во все концы методом заливки.

Процедура обработки входящего кадра зависит от того, по какой сети он прибыл и в какую сеть направляется.

1. Если сеть отправителя и сеть получателя совпадают, кадр игнорируется.
2. Если сеть отправителя и сеть получателя различаются, кадр переправляется.
3. Если сеть получателя неизвестна, используется алгоритм заливки.

Алгоритм применяется для каждого входящего кадра. Специальные СБИС отслеживают и обновляют записи в таблице каждые несколько микросекунд.

Мосты связующего дерева

Для повышения надежности иногда используются два и более параллельных моста между парами локальных сетей, как показано на рис. 4.39. Такое решение, впрочем, создает некоторые дополнительные проблемы, поскольку в топологии образуются кольца.

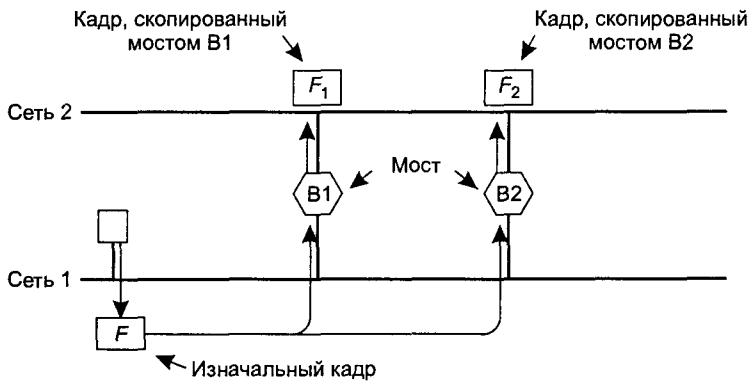


Рис. 4.39. Два параллельных прозрачных моста

В качестве примера, иллюстрирующего указанные проблемы, рассмотрим кадр F с неизвестным адресом назначения (рис. 4.39) Каждый мост, действуя по обычным правилам обработки кадров с неизвестным получателем, использует метод заливки, что в данном примере означает копирование кадра в сеть 2. Вскоре после этого мост 1 видит кадр F_2 с неизвестным получателем, который он копирует в сеть 1, формируя кадр F_3 (не показанный на рисунке). Аналогично этому, мост 2 копирует кадр F_1 в сеть 1, в результате создавая кадр F_4 (также не показан). Потом мост 1 копирует кадр F_4, а мост 2 копирует кадр F_3. Этот цикл продолжается вечно.

Решение данной проблемы заключается в установлении связи между мостами и наложении связующего дерева, покрывающего все сети, на действующую топологию. В результате некоторые возможные соединения между сетями игнорируются с целью создания фиктивной бескольцевой топологии. Например, на рис. 4.40, *а* показаны девять сетей, соединенные десятью мостами. Эта система может быть представлена в виде графа с сетями в качестве узлов. Дуга графа со-

единяет две сети, если эти сети соединены мостом. Такой граф можно редуцировать до связующего дерева, удалив из него дуги, изображенные пунктирными линиями на рис. 4.40, *б*. В получившемся связном дереве между каждыми двумя сетями существует только один путь. После того как мосты договорятся друг с другом о топологии связующего дерева, все межсетевые коммуникации осуществляются только по ветвям этого дерева. Поскольку путь от отправителя к получателю единственный, зацикливание невозможно.

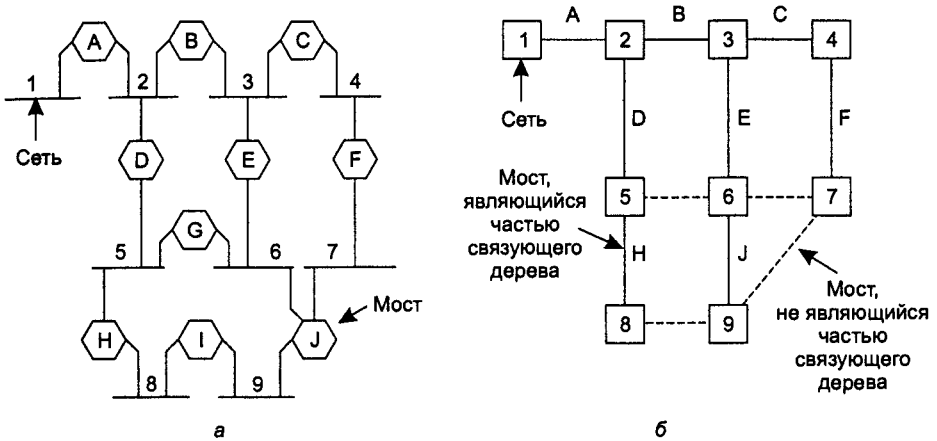


Рис. 4.40. Объединенные сети (*а*); связующее дерево, покрывающее сети (*б*)

Чтобы построить связующее дерево, мосты должны выбрать, кто из них будет корнем дерева. Для этого каждый мост рассылает кадры, содержащие серийный номер моста, установленный производителем. Эти номера гарантированно являются уникальными. Корнем дерева становится мост с наименьшим серийным номером. Затем строится дерево кратчайших путей от корня к каждому мосту и к каждой сети. Это дерево и будет связующим. Если какой-либо мост или сеть не функционируют, рассчитывается другое дерево.

В результате от каждой сети к корню дерева устанавливается уникальный путь, и, следовательно, между каждыми двумя сетями пути также будут уникальными. Хотя дерево покрывает все сети, в нем не обязательно присутствуют все мосты (во избежание петель). Алгоритм построения дерева продолжает работать постоянно, обнаруживая изменения в топологии и обновляя структуру дерева. Распределенный алгоритм построения связующего дерева был изобретен Радием Перлманом (Radia Perlman) и подробно описан в книге (Perlman, 2000). Он стандартизован и имеет идентификатор IEEE 802.1D.

Удаленные мосты

Обычно мосты применяются для соединения двух и более удаленных локальных сетей. Например, у какой-нибудь компании может быть несколько заводов в различных городах, каждый со своей локальной сетью. В идеале все эти сети должны быть соединены, чтобы вся система действовала как одна большая локальная сеть.

Для достижения этой цели можно установить на каждой сети по мосту и соединить эти мосты линиями «точка — точка» (например, арендованными у телефонной компании линиями). Простая система из трех локальных сетей показана на рис. 4.41. К данной конструкции применим обычный алгоритм маршрутизации. Проще всего рассматривать три двухточечные линии как локальную сеть без хостов. Нигде до сих пор не утверждалось, что в локальной сети обязательно должны быть хосты.

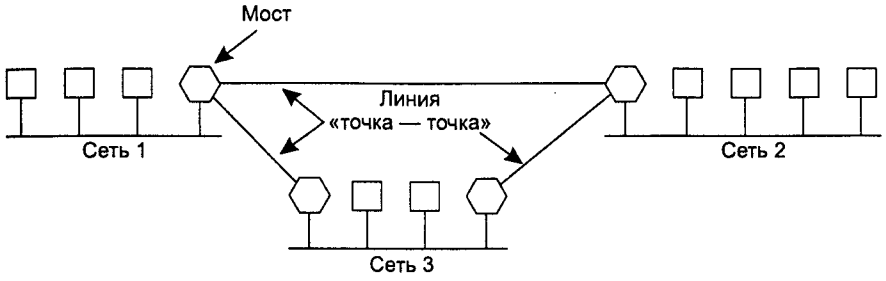


Рис. 4.41. Для соединения дальних сетей можно использовать удаленные хосты

На двухточечных линиях могут использоваться различные протоколы. Одним из вариантов может быть выбор какого-либо стандартного двухточечного протокола передачи данных с помещением кадров уровня доступа к носителю (MAC) целиком в поле данных. Такая стратегия наиболее оправданна в том случае, когда сети идентичны, при этом единственной проблемой остается доставка кадров в правильную сеть. Другой вариант заключается в том, что у кадров уровня доступа к носителю на первом мосту удаляются заголовок и концевик и в поле полезной нагрузки протокола «точка — точка» помещается то, что осталось. На втором мосту заголовок и концевик уровня доступа к носителю создаются заново. Недостаток такого метода состоит в том, что контрольная сумма по дороге несколько раз пересчитывается заново, что увеличивает вероятность появления не обнаруженных ошибок.

Повторители, концентраторы, мосты, коммутаторы, маршрутизаторы и шлюзы

Мы уже успели в нашей книге рассмотреть множество способов доставки кадров и пакетов из одного сегмента кабеля в другой. Мы упоминали повторители, мосты, концентраторы, маршрутизаторы и шлюзы. Все эти устройства используются очень широко, однако в чем-то они различаются едва уловимо, а в чем-то весьма существенно. Число их весьма велико, поэтому лучше рассмотреть их все в совокупности, отмечая сходства и различия.

Надо начать с того, что эти устройства работают на разных уровнях, как показано на рис. 4.42, *а*. Это имеет значение, поскольку от этого зависит, какую часть информации устройство использует для маршрутизации. Типичный сценарий таков: у пользователя появляются какие-то данные, которые необходимо отпра-

вить на удаленную машину. Они передаются на транспортный уровень, который добавляет к ним свой заголовок (например, заголовок TCP) и передает результирующую единицу информации на сетевой уровень. Тот, в свою очередь, тоже добавляет свой заголовок, в результате чего формируется *пакет* сетевого уровня (например, IP-пакет). На рис. 4.42, *б* IP-пакет выделен серым цветом. Пакет отправляется на уровень передачи данных (канальный уровень), где обрастает еще одним заголовком и контрольной суммой (CRC). Наконец формируется кадр, который спускается на физический уровень и может быть передан, например, по ЛВС.

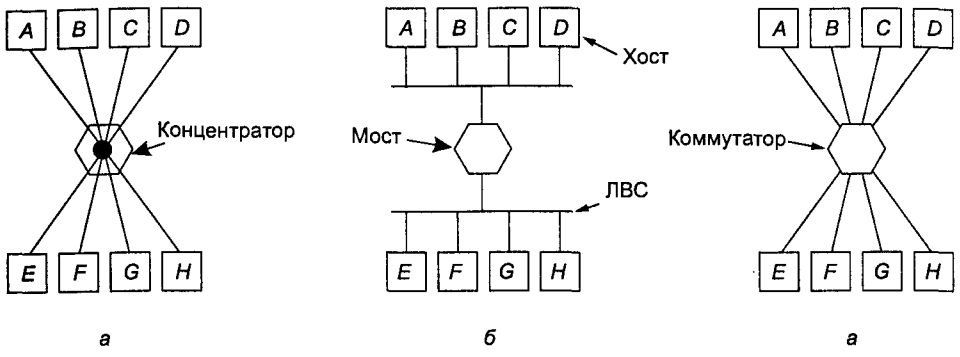


Рис. 4.42. Соответствие устройств уровням (*а*); кадры, пакеты и заголовки (*б*)

Приступим к рассмотрению коммутирующих устройств и взглянем на то, как они соотносятся с пакетами и кадрами. На самом нижнем, физическом уровне работают повторители. Это аналоговые устройства, к которым подсоединяются концы двух сегментов кабеля. Сигнал, появляющийся на одном из них, усиливается повторителем и выдается на второй. Повторители не знают слов «пакет», «кадр» или «заголовок». Они знают слово «напряжение». В классическом Ethernet допускается установка четырех повторителей, что позволяет расширять максимальную длину кабеля с 500 до 2500 м.

Теперь обратимся к концентраторам. Концентратор (хаб) имеет несколько входов, объединяемых электрически. Кадры, прибывающие на какой-либо вход, передаются на все остальные линии. Если одновременно по разным линиям придут два кадра, они столкнутся, как в коаксиальном кабеле. То есть концентратор представляет собой одну область столкновений. Все линии, подсоединяемые к нему, должны работать с одинаковыми скоростями. Концентраторы отличаются от повторителей тем, что они обычно не усиливают входные сигналы, поскольку предназначены не для этого. Их задача — обеспечивать согласованную работу нескольких плат с несколькими входами, к которым подключаются линии с по-

хожими параметрами. Впрочем, во всем остальном хабы не очень отличаются от повторителей. Ни те, ни другие не анализируют и не используют адреса стандарта 802. Принцип работы концентратора показан на рис. 4.43, *а*.



а *б* *а*

Рис. 4.43. Концентратор (*а*); мост (*б*); коммутатор (*в*)

Перейдем теперь на уровень передачи данных. Здесь мы обнаружим мосты и коммутаторы. Только что мы как раз более или менее подробно обсуждали мосты, поэтому знаем, что мост соединяет две или более ЛВС, как показано на рис. 4.43, *б*. Когда прибывает кадр, мост программно извлекает из заголовка и анализирует адрес назначения, сопоставляя его с таблицей и определяя, куда этот кадр должен быть передан. В Ethernet это 48-битный адрес, показанный на рис. 4.16. Как и в концентраторах, в современных мостах имеются вставные сетевые платы, обычно рассчитанные на 4 или 8 входов определенного типа. Плата Ethernet, например, не может обрабатывать кадры сетей типа маркерное кольцо, поскольку она не знает, в какой части заголовка искать адрес назначения. Тем не менее, мост может иметь несколько плат, благодаря чему может работать с сетями разных типов. Каждая линия, подключенная к мосту, является областью столкновений, в отличие от линий концентратора.

Коммутаторы похожи на мосты в том, что для маршрутизации используют адреса кадров. На самом деле многие употребляют эти понятия как синонимы. Различаются они тем, что коммутаторы чаще всего используются для соединения отдельных компьютеров (рис. 4.43, *в*), а не сетей. Следовательно, если хост *A* (рис. 4.43, *б*) хочет отправить кадр на хост *B*, мост получит этот кадр, но отвергнет его. Вместе с тем, коммутатор на рис. 4.43, *в* должен самым активным образом способствовать передаче кадра от хоста *A* к хосту *B*, поскольку для кадра это единственная возможность. Так как каждый порт коммутатора обычно соединен с одним компьютером, в коммутаторах должно быть гораздо больше разъемов для сетевых плат, чем в мостах, поскольку последние соединяют целые сети. Каждая плата содержит буфер для хранения пришедших кадров. Поскольку каждый порт является областью столкновений, то кадры из-за коллизий теряться не могут. Однако если скорость передачи данных по каналу превысит максимальную скорость их обработки, буфер может переполниться и продолжающие приходить кадры будут отвергаться.

Несколько уменьшить эту проблему помогают современные коммутаторы, которые начинают пересылать кадры, едва получив их заголовки и не дожидаясь полной их докачки (конечно, для этого должна быть свободна выходная линия). Такие коммутаторы не используют протоколы с ожиданием. Иногда их называют **сквозными коммутаторами**. Этот метод чаще всего реализуется аппаратно, тогда как в мостах традиционно присутствует процессор, программно реализующий маршрутизацию с ожиданием. Но поскольку все современные мосты и коммутаторы содержат специальные интегральные схемы коммутации, техническая разница между ними практически стирается, и остается лишь разница в том, что вы слышите о них в рекламе.

Итак, мы вкратце рассмотрели повторители и концентраторы, которые весьма сходны друг с другом, а также коммутаторы и мосты, которые также не сильно различаются. Теперь же мы перейдем к маршрутизаторам, которые резко отличаются от всего рассмотренного ранее. Когда пакет прибывает на маршрутизатор, отрезаются заголовки и концевики кадров и остаются только поля данных (выделены серым на рис. 4.42), которые и передаются программному обеспечению маршрутизатора. Далее анализируется заголовок пакета, и в соответствии с ним выбирается его дальнейший путь. Если это IP-пакет, то в заголовке будет содержаться 32-битный (IPv4) или 128-битный (IPv6), а не 48-битный (стандарт 802) адрес. Программное обеспечение маршрутизатора не интересуется адресами кадров и даже не знает, откуда эти кадры взялись (то ли с ЛВС, то ли с двухточечной линии). Более подробно мы изучим маршрутизаторы и принципы маршрутизации в главе 5.

Поднявшись еще на уровень выше, мы обнаружим транспортные шлюзы. Они служат для соединения компьютеров, использующих различные транспортные протоколы, ориентированные на установление соединения. Например, такая ситуация возникает, когда компьютеру, использующему TCP/IP, необходимо передать данные компьютеру, использующему ATM. Транспортный шлюз может копировать пакеты, одновременно приводя их к нужному формату.

Наконец, шлюзы приложений уже работают с форматами и содержимым пакетов, занимаясь переформатированием на более высоком уровне. Например, шлюз e-mail может переводить электронные письма в формат SMS-сообщений для мобильных телефонов.

Виртуальные локальные сети

На заре развития технологий локальных сетей толстые желтые провода опутали огромное количество офисов. Можно было подключить к сети каждый компьютер, мимо которого шел такой провод. Зачастую кабели объединялись в центральную магистраль (как показано на рис. 4.35) или шли к центральному концентратору. Никто не задумывался над тем, к какой ЛВС подключить компьютер. Все соседние компьютеры были станциями одной сети независимо от того, подходили они друг другу или нет. Логическое соединение определялось исключительно расположением в пространстве.

С развитием в 1990-е годы систем на основе 10Base-T и концентраторов все изменилось. Из офисных зданий стали исчезать эти желтые провода, напоминающие садовые шланги, и им на смену пришли витые пары, которые шли к щитам, висящим по концам коридоров и напичканным проводами (см. рис. 4.44). Если чиновник, ответственный за прокладку кабелей в здании, был способен смотреть в будущее, то устанавливались витые пары категории 5; если же он был крохобором, то использовались существующие (телефонные) витые пары категории 3, которые были заменены только с приходом сетей типа «быстрый Ethernet».

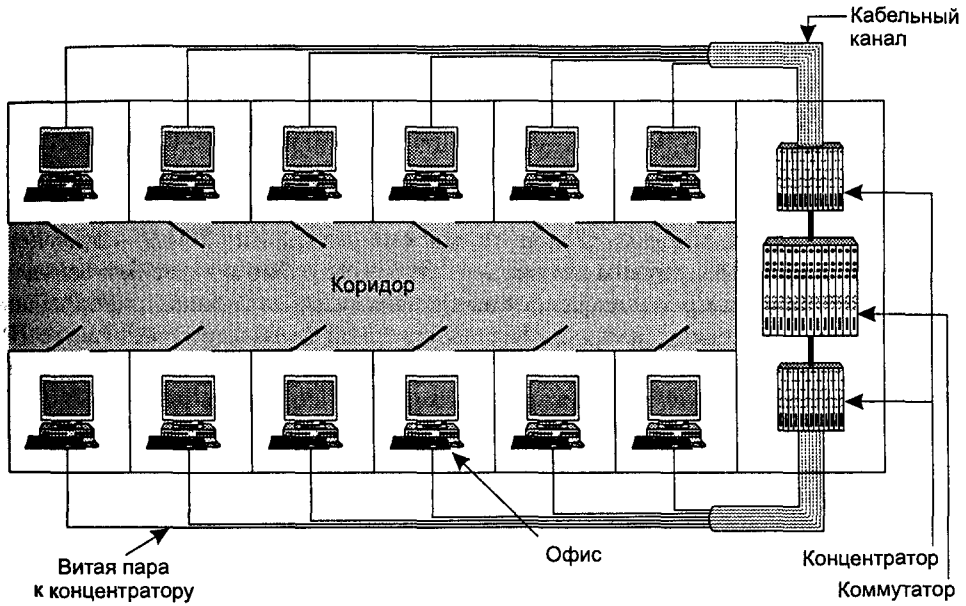


Рис. 4.44. Здание с централизованной проводкой

В Ethernet с концентраторами (а позднее — с коммутаторами) появилась возможность настройки локальных сетей не физически, а логически. Если компании требовалось k ЛВС, она приобретала k концентраторов. Аккуратно собрав сеть (то есть вставив нужные соединители в нужные разъемы), можно было определить ее пользователей таким образом, чтобы это имело некий реальный организационный смысл и не зависело от расположения станций внутри здания. Конечно, если два человека работают в одном отделе, но в разных зданиях, они, скорее всего, будут пользоваться разными концентраторами, а значит, и разными сетями. Тем не менее, это намного лучше, чем основывать членство в той или иной ЛВС исключительно на принципе территориального расположения.

А вообще, разве важно, кто подключен к какой ЛВС? Ведь все равно почти во всех организациях сети объединены между собой. На самом же деле это действительно зачастую бывает важно. Сетевые администраторы по целому ряду причин любят группировать пользователей ЛВС в соответствии со структурой организации, а не по принципу физического расположения компьютеров. Одной из таких

причин является защита информации. Любой сетевой интерфейс может быть переведен в беспорядочный режим, когда обрабатывается весь трафик, приходящий по каналу. Многие отделы, такие как отдел исследований, патентный, бухгалтерия, владеют информацией, которую они не хотели бы выносить за пределы своего подразделения. В этой ситуации наилучшим выходом является объединение компьютеров всех работников отдела в одну ЛВС и запрет на передачу каких-либо данных наружу. Руководство обычно не устраивает заявление о том, что такое решение невозможно реализовать до тех пор, пока все работники отдела не будут размещены в соседних кабинетах.

Второй причиной является нагрузка на сеть. Сеть может оказаться настолько загруженной, что ее будет лучше разделить на несколько. Например, если народ из отдела исследований решит провести какой-нибудь хитрый эксперимент, то бухгалтерия будет не очень счастлива от сознания, что он проводится за счет емкостей их общей сети.

Еще одна причина — широковещание. Большинство сетей и многие протоколы верхних уровней поддерживают широковещание. Например, если пользователь хочет отправить пакет на IP-адрес *x*, как ему узнать, какой MAC-адрес подставлять в кадры? Мы изучим этот вопрос более подробно в главе 5, а сейчас в двух словах обрисуем решение проблемы: данная станция должна широковещательным методом послать запрос: «Кто знает, какой MAC-адрес работает с IP-адресом *x*?» Затем она дожидается ответа. Есть много других примеров с использованием широковещательной передачи. По мере объединения различных ЛВС число широковещательных пакетов, проходящих через каждую машину, растет линейно пропорционально общему числу машин.

С широковещанием связана еще одна проблема: время от времени сетевой интерфейс ломается и начинает генерировать бесконечный поток кадров, получаемый всеми станциями. Это **широковещательный штурм**, который состоит в следующем: 1) вся пропускная способность сети занята этими бессмысленными кадрами; 2) все машины объединенных сетей вынуждены заниматься исключительно обработкой и отвержением этого мусора.

На первый взгляд кажется, что широковещательные штурмы можно ограничить установкой своего рода дамб — разделяя сети мостами или коммутаторами на несколько частей. Однако если речь идет о прозрачности (то есть о предоставлении машинам возможности перенесения в другие ЛВС без каких-либо изменений конфигурации), то широковещательные кадры должны обрабатываться и пересылаться мостами.

Рассмотрев причины того, что компаниям требуются многочисленные локальные сети ограниченных размеров, давайте вернемся к проблеме разделения логической и физической топологий. Допустим, пользователь переезжает в другое помещение, не меняя принадлежности к тому или иному отделу, или наоборот — меняет отдел, не переезжая никуда из своего офиса. Если сеть в данной организации оборудована концентраторами, то действия сетевого администратора в указанной ситуации сводятся к тому, что он доходит до щита с проводами и вставляет соединитель, идущий от пользователя, в разъем другого концентратора.

Во многих компаниях организационные перестановки происходят постоянно, то есть сетевой администратор постоянно занимается манипуляциями с соединителями и разъемами. А в некоторых случаях такие перестановки оказываются невозможными из-за того, что провод пользовательского компьютера проходит слишком далеко и просто не дотягивается до нужного щита (вследствие непродуманной прокладки кабелей).

Пользователи стали требовать у производителей создания более гибких сетей, и те ответили им созданием **виртуальных ЛВС**, для внедрения которых необходимо было изменить только программное обеспечение. Виртуальные сети даже в какой-то момент были стандартизованы комитетом 802. Сейчас они применяются все шире и шире. Далее мы вкратце обсудим виртуальные сети. Дополнительную информацию можно найти в (Breyer and Riley, 1999; Seifert, 2000).

Виртуальные сети построены на основе специально разработанных коммутаторов, хотя в их состав могут входить и обычные концентраторы (см. рис. 4.44). Для создания системы, построенной на виртуальных ЛВС, сетевому администратору прежде всего нужно решить, сколько всего будет виртуальных сетей, какие компьютеры будут в них входить и как они будут называться. Зачастую ВЛВС (неформально) называют в соответствии с цветами радуги, поскольку тогда сразу становятся понятнее и нагляднее цветные диаграммы, показывающие принадлежность пользователей виртуальным сетям. Скажем, пользователи «красной» сети будут изображены красным цветом, «синей» — синим, и т. д. Таким образом, на одном рисунке можно отобразить физическую и логическую структуры одновременно.

В качестве примера рассмотрим четыре ЛВС, изображенные на рис. 4.45, *а*. Здесь восемь машин входят в виртуальную сеть *С* (Серая), а еще семь машин — в виртуальную сеть *Б* (белая). Четыре физических сети объединены двумя мостами, *B1* и *B2*. Если используется центральная проводка на основе витой пары, можно также установить четыре концентратора (не показаны на рисунке), однако логически моноканал и сеть с концентратором — это одно и то же. Чтобы не загромождать рисунок, мы не стали изображать на нем концентраторы. Термин «мост» сейчас применяется в основном тогда, когда к каждому порту подсоединено несколько машин; в противном случае слова «мост» и «коммутатор» можно считать синонимами. На рис. 4.45, *б* показаны те же самые станции тех же самых виртуальных сетей, только здесь используются коммутаторы, к каждому порту которых подключено по одному компьютеру.

Чтобы виртуальные сети функционировали корректно, необходимо наличие конфигурационных таблиц в мостах или коммутаторах. Эти таблицы сообщают о том, через какие порты (каналы) производится доступ к тем или иным виртуальным сетям. Когда кадр прибывает, например, из серой ВЛВС, его нужно разослать на все порты, помеченные буквой *С*. Это правило справедливо как для ординарных (то есть однонаправленных) передач, так и для групповых и широковещательных.

Имейте в виду, что порт может быть помечен сразу несколькими буквами, то есть он может обслуживать несколько ВЛВС. Это видно и на рис. 4.45, *а*. Пусть у машины *А* имеется кадр для широковещательной передачи. Мост *B1* принимает

его и замечает, что он пришел с машины с пометкой *C*. Поэтому его необходимо ретранслировать на все порты (кроме того, с которого пришел кадр), принадлежащих «серой» виртуальной сети. У *B1* есть только два порта, кроме входящего, и оба помечены как *C*, поэтому кадр передается на оба из них.

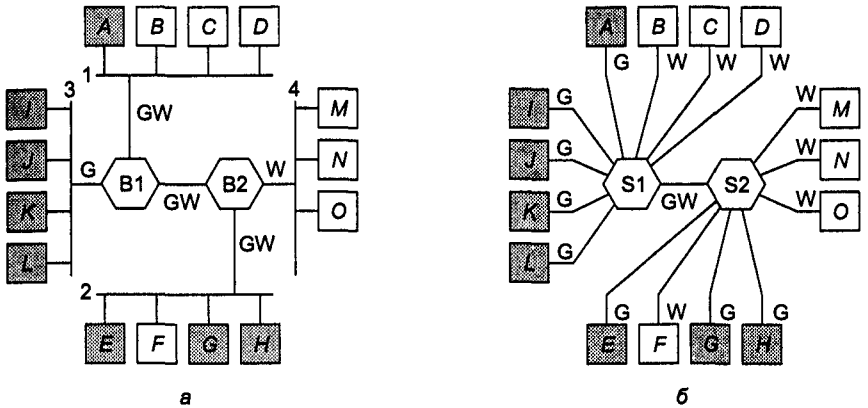


Рис. 4.45. Четыре физические ЛВС, объединенные в две виртуальные сети, серую и белую, двумя мостами (*а*); те же 15 машин, объединенных в две виртуальные сети коммутаторами (*б*)

С мостом *B2* история несколько другая. Здесь мост знает, что в ЛВС 4 нет машин типа *C*, поэтому туда кадры не передаются вообще. Ретрансляция производится только в ЛВС 2. Если какой-нибудь пользователь ЛВС 4 перейдет на работу в другой отдел, а там будет сеть *C*, то таблицы моста *B2* необходимо будет обновить и переобозначить порт *СБ* вместо *Б*. Если машина *F* уходит в серую сеть, то порт к ЛВС 2 надо будет переобозначить — вместо *СБ* он станет *C*.

Допустим теперь, что все машины ЛВС 2 и ЛВС 4 стали серыми. Тогда не только порты *B2*, соединенные с ЛВС 2 и ЛВС 4, получат обозначение *C*, но и порт *B1*, соединенный с *B2*, превратится в *C*, поскольку «белые» кадры, приходящие на *B1*, больше не будут нуждаться в ретрансляции на мост *B2*. На рис. 4.45, *б* изображена та же ситуация, только здесь все порты, соединенные с определенной машиной, отмечены только одним цветом, поскольку каждый компьютер может принадлежать только одной ВЛВС.

Итак, мы предположили, что мосты и коммутаторы каким-то образом узнают «цвет» приходящего кадра. Но как они это делают? Применяется один из следующих методов:

1. Каждому порту присваивается цвет.
2. Каждому MAC-адресу присваивается цвет.
3. Все протоколы 3-го уровня или IP-адреса соответствуют определенному цвету.

В первом методе каждый порт маркируется цветом какой-либо ВЛВС. Однако это работает только в том случае, если все машины порта принадлежат одной виртуальной сети. На рис. 4.45, *а* этим свойством обладает порт ЛВС 3 моста *B1*, а порт ЛВС 1 уже не может применять метод маркировки портов.

При использовании второго метода мост или коммутатор имеет таблицу, в которой представлены 48-битные MAC-адреса и названия виртуальных сетей всех станций, соединенных с устройством. В этом случае можно смешивать виртуальные сети внутри физической сети, как это происходит, скажем, с ЛВС 1 на рис. 4.45, *а*. Когда прибывает кадр, мосту или коммутатору необходимо лишь извлечь адрес MAC-уровня и найти его в таблице (это позволит понять, в какой виртуальной сети находится станция, с которой был отправлен кадр).

Третий метод заключается в том, что мост (коммутатор) просматривает поля данных кадров, чтобы, например, классифицировать все IP-машины как принадлежащие к одной виртуальной сети, а машины AppleTalk — к другой. В первом случае с помощью IP-адреса идентифицируется также конкретная машина. Такая стратегия особенно полезна, когда существенная часть станций представляют собой ноутбуки, которые могут подключаться в одном из нескольких мест. Поскольку у каждой из стыковочных станций есть свой MAC-адрес, то просто информация о том, какая стыковочная станция использовалась, ничего не скажет о том, в какой из виртуальных сетей находится сам ноутбук.

Единственная проблема, связанная с этим подходом, состоит в том, что он нарушает самый фундаментальный закон сетей — независимость уровней. Уровню передачи данных не должно быть никакого дела до содержимого поля данных. Он не должен просматривать его и тем более не имеет права принимать решения, исходя из него. Последствием использования этого метода стало то, что внесение изменений в протокол 3-го уровня (например, модернизация от IPv4 к IPv6) приводит к неработоспособности коммутаторов. К сожалению, такие коммутаторы все еще присутствуют на рынке.

Конечно, сама по себе маршрутизация, базирующаяся на IP-адресах (которой посвящена почти вся 5-я глава), является вполне легитимной, однако смешивание уровней — это оборотная и весьма нелицеприятная сторона медали. Производители коммутаторов могут пренебречь этим аргументом, заявляя, что их аппаратура понимает как IPv4, так и IPv6, так что все замечательно. Но что будет, если в один прекрасный день появится IPv7? Производители, вероятно, скажут: «Что ж, покупайте новые коммутаторы, что в этом такого?»

Стандарт IEEE 802.1Q

Если задуматься о том, как же работают виртуальные сети, то в голову приходит мысль, что все дело не в отправляющей машине, а в самом кадре ВЛВС. Если бы был какой-нибудь способ идентифицировать ВЛВС по заголовку кадра, отпала бы необходимость просмотра его содержимого. По крайней мере, в новых сетях типа 802.11 или 802.16 вполне можно было бы просто добавить специальное поле заголовка. Вообще-то *Идентификатор кадра* в стандарте 802.16 — это как раз нечто в этом духе. Но что делать с Ethernet — доминирующей сетью, у которой нет никаких «запасных» полей, которые можно было бы отдать под идентификатор виртуальной сети?

Комитет IEEE 802 озаботился этим вопросом в 1995 году. После долгих дискуссий было сделано невозможное — изменен формат заголовка кадра Ethernet! Новый формат было опубликован под именем 802.1Q в 1998 году. В заголовок

кадра был вставлен флаг ВЛВС, который мы сейчас вкратце рассмотрим. Понятно, что внесение изменений в нечто уже устоявшееся, такое как Ethernet, должно быть произведено каким-то нетривиальным образом. Встают, например, следующие вопросы:

1. И что, теперь надо будет выбросить на помойку несколько миллионов уже существующих сетевых карт Ethernet?
2. Если нет, то кто будет заниматься генерированием новых полей кадров?
3. Что произойдет с кадрами, которые уже имеют максимальный размер?

Конечно, комитет 802 тоже был озабочен этими вопросами, и решение, несмотря ни на что, было найдено.

Идея состоит в том, что на самом деле поля ВЛВС реально используются только мостами да коммутаторами, а не машинами пользователей. Так, скажем, сеть, изображенную на рис. 4.45, не очень-то волнует их наличие в каналах, идущих от оконечных станций, до тех пор, пока кадры не доходят до мостов или коммутаторов. Таким образом, чтобы была возможна работа с виртуальными сетями, про их существование должны знать мосты и коммутаторы, но это требование и так понятно. Теперь же мы выставляем еще одно требование: они должны знать про существование 802.1Q. Уже выпускается соответствующее оборудование.

Что касается старых сетевых карт Ethernet, то выкидывать их не приходится. Комитет 802.3 никак не мог заставить людей изменить поле *Тип* на поле *Длина*. Вы можете себе представить, какова была бы реакция на заявление о том, что все существующие карты Ethernet можно выбросить? Тем не менее, на рынке появляются новые модели, и есть надежда, что они теперь будут 802.1Q-совместимыми и смогут корректно заполнять поля идентификации виртуальных сетей.

Если отправитель не генерирует поле признака виртуальной сети, то кто же этим занимается? Ответ таков: первый встретившийся на пути мост или коммутатор, обрабатывающий кадры виртуальных сетей, вставляет это поле, а последний — вырезает его. Но как он узнает, в какую из виртуальных сетей передать? Для этого первое устройство, которое вставляет поле ВЛВС, может присвоить номер виртуальной сети порту, проанализировать MAC-адрес или (не дай Бог, конечно) подсмотреть содержимое поля данных. Пока все не перейдут на Ethernet-карты, совместимые со стандартом 802.1Q, все именно так и будет. Остается надеяться на то, что все сетевые платы гигабитного Ethernet будут придерживаться стандарта 802.1Q с самого начала их производства, и таким образом всем пользователям гигабитного Ethernet этой технологии автоматически станут доступны возможности 802.1Q. Что касается проблемы кадров, длина которых превышает 1518 байт, то в стандарте 802.1Q она решается путем повышения лимита до 1522 байт.

При передаче данных в системе могут встречаться как устройства, которым сокращение ВЛВС не говорит ровным счетом ни о чем (например, классический или быстрый Ethernet), так и совместимая с виртуальными сетями аппаратура (например, гигабитный Ethernet). Такая ситуация показана на рис. 4.46. Здесь затененные символы означают ВЛВС-совместимые устройства, а пустые квадратики — все остальные. Для простоты мы предполагаем, что все коммутаторы

ВЛВС-совместимы. Если же это не так, то первый такой ВЛВС-совместимый коммутатор добавит в кадр признак виртуальной сети, основываясь на информации, взятой из MAC- или IP-адреса.

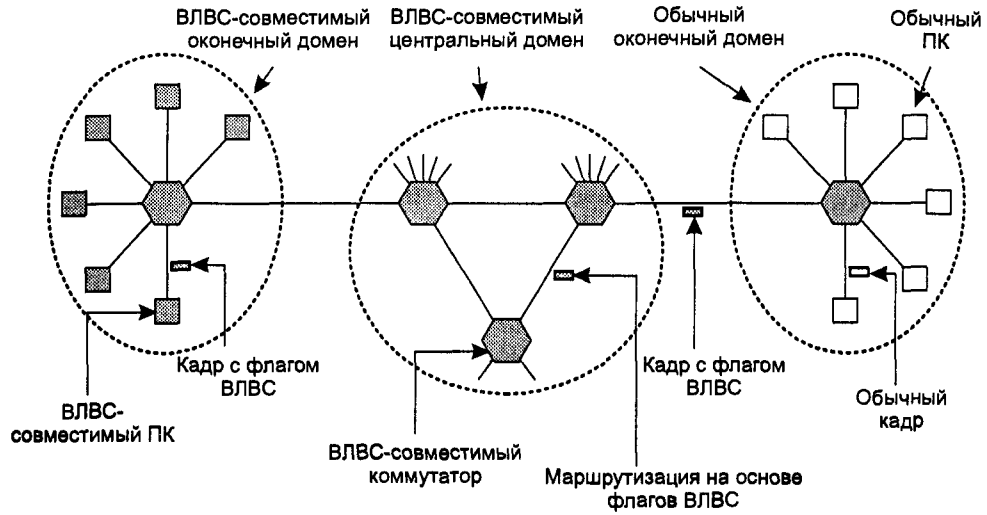


Рис. 4.46. Передача данных из обычного Ethernet в ВЛВС-совместимый Ethernet. Затененные символы — это ВЛВС-устройства. Все остальные несовместимы с виртуальными сетями

На этом рисунке мы видим, что ВЛВС-совместимые сетевые платы Ethernet генерируют кадры с флагами (то есть кадры стандарта 802.1Q), и дальнейшая маршрутизация производится уже с использованием этих флагов. Для осуществления маршрутизации коммутатор, как и раньше, должен знать, какие виртуальные сети доступны на всех портах. Информация о том, что кадр принадлежит серой виртуальной сети, еще, по большому счету, ни о чем не говорит, поскольку коммутатору еще нужно знать, какие порты соединены с машинами серой виртуальной сети. Таким образом, коммутатору нужна таблица соответствия портов виртуальным сетям, из которой также можно было бы узнать, являются ли порты ВЛВС-совместимыми.

Когда обычный, ничего не подозревающий о существовании виртуальных сетей компьютер посылает кадр на коммутатор виртуальной сети, последний генерирует новый кадр, вставляя в него флаг ВЛВС. Информацию для этого флага он получает с виртуальной сети отправителя (для ее определения используется номер порта, MAC- или IP-адрес.) Начиная с этого момента никто больше не переживает из-за того, что отправитель является машиной, не поддерживающей стандарт 802.1Q. Таким же образом коммутатор, желающий доставить кадр с флагом на такую машину, должен привести его к соответствующему формату.

Теперь рассмотрим собственно формат 802.1Q. Он показан на рис. 4.47. Единственное изменение, которое мы тут видим, — это пара 2-байтовых полей. Первое называется *Идентификатор протокола ВЛВС*. Оно всегда имеет значе-

ние 0x8100. Поскольку это число превышает 1500, то все сетевые карты Ethernet интерпретируют его как «тип», а не как «длину». Неизвестно, что будет делать карта, несовместимая с 802.1Q, поэтому такие кадры, по идее, не должны к ней никоим образом попадать.

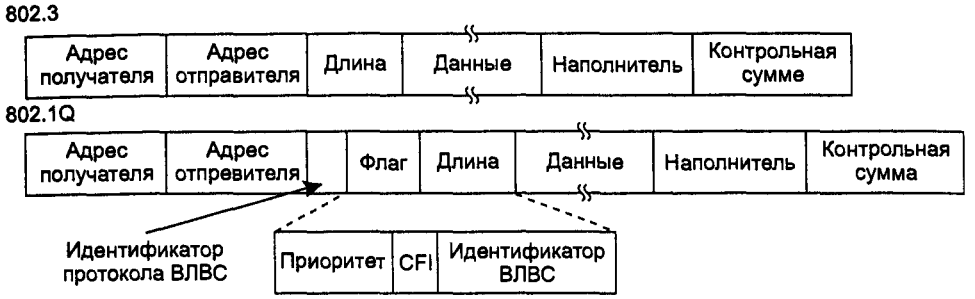


Рис. 4.47. Форматы кадров Ethernet-стандартов 802.3 или 802.1Q

Во втором двухбайтовом поле есть три вложенных поля. Главным из них является *Идентификатор ВЛВС*, который занимает 12 младших битов. Он содержит ту информацию, из-за которой все эти преобразования форматов, собственно, и были затеяны: в нем указано, какой виртуальной сети принадлежит кадр. Трехбитовое поле *Приоритет* не имеет совершенно ничего общего с виртуальными сетями. Просто изменение формата Ethernet-кадра — это такой ежедекадный ритуал, который занимает три года и исполняется какой-то сотней людей. Почему бы не оставить память о себе в виде трех дополнительных бит, да еще и с таким привлекательным назначением. Поле *Приоритет* позволяет различать трафик с жесткими требованиями к реальности масштаба времени, трафик со средними требованиями и трафик, для которого время передачи не критично. Это позволяет обеспечить более высокое качество обслуживания в Ethernet. Оно используется также при передаче голоса по Ethernet (хотя вот уже четверть века в IP имеется подобное поле, и никому никогда не требовалось его использовать).

Последний бит, *CFI* (Canonical Format Indicator — индикатор классического формата), следовало бы назвать Индикатором эгоизма компании. Изначально он предназначался для того, чтобы показывать, что применяется формат MAC-адреса с прямым порядком байтов (или, соответственно, с обратным порядком), однако в пылу дискуссий об этом как-то забыли. Его присутствие сейчас означает, что поле данных содержит усохший кадр 802.5, который ищет еще одну сеть формата 802.5 и в Ethernet попал совершенно случайно. То есть на самом деле он просто использует Ethernet в качестве средства передвижения. Все это, конечно, практически никак не связано с обсуждаемыми в данном разделе виртуальными сетями. Но политика комитета стандартизации не сильно отличается от обычной политики: если ты проголосуешь за введение в формат моего бита, то я проголосую за твой бит.

Как уже упоминалось ранее, когда кадр с флагом виртуальной сети приходит на ВЛВС-совместимый коммутатор, последний использует идентификатор вир-

туальной сети в качестве индекса таблицы, в которой он ищет, на какой бы порт послать кадр. Но откуда берется эта таблица? Если она разрабатывается вручную, это означает возврат в исходную точку: ручное конфигурирование коммутаторов. Вся прелесть прозрачности мостов состоит в том, что они настраиваются автоматически и не требуют для этого никакого вмешательства извне. Было бы очень стыдно потерять это свойство. К счастью, мосты для виртуальных сетей также являются самонастраивающимися. Настройка производится на основе информации, содержащейся во флагах приходящих кадров. Если кадр, помеченный как ВЛВС 4, приходит на порт 3, значит, несомненно, одна из машин, подключенных к этому порту, находится в виртуальной сети 4. Стандарт 802.1Q вполне четко поясняет, как строятся динамические таблицы. При этом делаются ссылки на соответствующие части алгоритма Перлмана (Perlman), который вошел в стандарт 802.1D.

Прежде чем завершить разговор о маршрутизации в виртуальных сетях, необходимо сделать еще одно замечание. Многие пользователи сетей Интернет и Ethernet фанатично привязаны к сетям без установления соединения и неистово противопоставляют их любым системам, в которых есть хотя бы намек на соединение на сетевом уровне или уровне передачи данных. Однако в виртуальных сетях один технический момент как раз-таки очень сильно напоминает установку соединения. Речь идет о том, что работа виртуальной сети невозможна без того, чтобы в каждом кадре был идентификатор, использующийся в качестве индекса таблицы, встроенной в коммутатор. По этой таблице определяется дальнейший вполне определенный маршрут кадра. Именно это и происходит в сетях, ориентированных на соединение. В системах без установления соединения маршрут определяется по адресу назначения, и там отсутствуют какие-либо идентификаторы конкретных линий, через которые должен пройти кадр. Более подробно мы рассмотрим эту тенденцию в главе 5.

Резюме

В некоторых сетях для любой связи используется единственный моноканал. При их разработке основной проблемой является распределение этого канала между соревнующимися за право его использования станциями. Разработаны различные алгоритмы распределения канала. Наиболее важные из них приведены в табл. 4.5.

Таблица 4.5. Методы распределения канала и системы с общим каналом

Метод	Описание
Частотное уплотнение (FDM)	Выделение каждой станции частотного диапазона
Спектральное уплотнение (WDM)	Динамическое частотное уплотнение для оптического волокна
Временное уплотнение	Выделение каждой станции временного интервала
Чистая система ALOHA	Несинхронизированная передача в любой момент времени

продолжение

Таблица 4.5 (*продолжение*)

Метод	Описание
Дискретная система ALOHA	Случайная передача в строго определенные временные интервалы
CSMA (множественный доступ с контролем несущей) с настойчивостью 1	Стандартная система множественного доступа с контролем несущей
Ненастойчивая система CSMA	Случайная задержка при занятом канале
CSMA с настойчивостью *p*	Система CSMA с вероятностью передачи *p*
CSMA/CD (CSMA с обнаружением столкновений)	Система CSMA с прекращением передачи при обнаружении столкновений
Бит-карта	Поочередное использование канала с использованием бит-карты для резервирования временных интервалов
Двоичный обратный отсчет	Передача в каждый следующий интервал времени готовой станцией с максимальным номером
Адаптивное дерево	Снижение остроты состязаний при помощи разбиения станций на группы
Спектральное разделение	Динамическая схема частотного уплотнения для оптоволоконного кабеля
MACA, MACAW	Беспроводные протоколы локальных сетей
Ethernet	CSMA/CD с удвоением времени ожидания
FHSS	Передача широкополосного сигнала по методу скачкообразного изменения частот
DSSS	Передача широкополосного сигнала по методу прямой последовательности
CSMA/CA	Множественный доступ с контролем несущей и предотвращением столкновений

Простейшими схемами распределения являются частотное и временное уплотнения. Они эффективны при небольшом количестве станций и постоянном трафике. Оба метода широко применяются в этих условиях, например, для разделения полосы пропускания в телефонных магистралях.

Однако когда количество станций велико и непостоянно или трафик является пульсирующим, частотное и временное уплотнения использовать нецелесообразно. В качестве альтернативы предлагается протокол ALOHA — дискретный и непрерывный, с управлением и без него. Система ALOHA, ее многочисленные варианты и производные системы широко обсуждались, анализировались и использовались в реальных системах.

Если состояние канала можно прослушивать, станции могут отказываться от передачи, когда слышат, что канал занят другой станцией. Применение метода контроля несущей повлекло за собой создание большого количества протоколов, применяемых в локальных и региональных сетях.

Широко известен класс протоколов, полностью устраняющий борьбу за канал или, по меньшей мере, значительно снижающий ее напряженность. Протокол с двоичным обратным отсчетом полностью устраняет состязание за канал. Протокол адаптивного дерева снижает его остроту, динамически деля станции

на две непересекающиеся группы, одной из которых разрешается передавать во время следующего интервала времени, а другой — нет. Алгоритм пытается делить станции таким образом, чтобы только одной станции, готовой к передаче, разрешалось это сделать.

У беспроводных локальных сетей есть свои проблемы и методы их решения. Наибольшую проблему представляют собой скрытые станции, поэтому алгоритм CSMA в данной ситуации не работает. Один из методов, применяемый, например, в протоколах MACA и MACAW, заключается в попытке стимулировать передачу вблизи получателя, что улучшает работу протокола CSMA. Также применяются алгоритмы передачи широкополосного сигнала по методам прямой последовательности и скачкообразного изменения частот. Стандарт IEEE 802.11 совмещает в себе методы CSMA и MACAW. Результатом такого комбинирования является метод CSMA/CA.

Ethernet является доминирующей технологией локальных вычислительных сетей. Распределение канала производится при помощи метода CSMA/CD. В старых версиях использовался кабель, который тянулся от машины к машине, однако теперь более распространены витые пары, идущие к концентраторам или коммутаторам. Скорости возросли от 10 Мбит/с до 1 Гбит/с и продолжают расти.

Беспроводные сети становятся все популярнее. Доминирует здесь стандарт 802.11. Физический уровень, определенный этим стандартом, позволяет работать в пяти различных режимах передачи, в которых используются инфракрасные сигналы, различные методы расширения спектра и многоканальная система частотного уплотнения. Работа может вестись как при наличии базовой станции в каждой ячейке, так и без нее. Используется протокол, аналогичный MACAW, с контролем виртуальной несущей.

Начинают появляться беспроводные региональные сети. Это широкополосные системы, которые решили проблему «последней мили» в телефонных системах путем применения радиосвязи. При этом используются традиционные методы узкополосной модуляции. Здесь важную роль играет качество обслуживания, и стандарт 802.16 определяет четыре класса сервисов (постоянная битовая скорость, два класса с переменной битовой скоростью, а также сервис с обязательствами приложения максимальных усилий).

Система Bluetooth — это еще один тип беспроводных сетей, но довольно специфический. Чаще всего Bluetooth используется для беспроводного соединения компьютера или мобильного телефона с периферийными устройствами. Как и в стандарте 802.11, используется передача широкополосного сигнала по методу скачкообразных изменений частоты в не подлежащем лицензированию диапазоне. Поскольку среда сильно зашумлена, а также благодаря необходимости передачи данных в реальном масштабе времени очень тщательно прорабатывались алгоритмы исправления ошибок, встраиваемые в различные протоколы.

В мире существует огромное количество вычислительных сетей. Понятно, что необходим какой-то способ их объединения. Для этих целей используются такие устройства, как мосты и коммутаторы. Для создания самонастраивающихся мостов применяется алгоритм связующего дерева. Новым словом в деле объединения ЛВС стали виртуальные локальные сети, которые позволили отделить

физическую топологию от логической. Был разработан новый формат кадров (стандарт 802.1Q), который позволил упростить процесс внедрения виртуальных сетей в организациях.

Вопросы

1. Для решения задачи используйте формулу, приведенную в данной главе, записав ее в общем виде. Кадры для передачи прибывают случайным образом на 100-мегабитный канал. Если в момент прибытия канал оказывается занят, кадр ставится в очередь ожидания. Длина кадра распределяется по экспоненциальному закону с математическим ожиданием, равным 10 000 бит/кадр. Для каждой из приведенных далее скоростей прибытия кадров вычислите задержку (включая время ожидания в очереди и время передачи) кадра средней длины.
   1) 90 кадров/с;
   2) 900 кадров/с;
   3) 9000 кадров/с.
2. Группа из N станций совместно использует канал чистой системы ALOHA, работающий со скоростью 56 Кбит/с. Каждая станция передает 1000-битный кадр в среднем каждые 100 с, даже если предыдущий кадр еще не был передан (например, станции могут буферизировать исходящие кадры). Каково максимальное значение N?
3. Сравните время задержки чистой и дискретной систем ALOHA при низкой нагрузке. У какой из систем это время будет меньшим? Поясните свой ответ.
4. 10 000 станций соревнуются за право использования единственного канала дискретной системы ALOHA. В среднем одна станция делает 18 запросов в час. Длительность интервала равна 125 мкс. Какова приблизительная суммарная загруженность канала?
5. Большая группа пользователей системы ALOHA формирует 50 запросов в секунду, включая первичные и повторные передачи. Время разделено на интервалы по 40 мс.
   1) Каковы шансы успеха с первой попытки?
   2) Какова вероятность того, что перед успехом произойдет ровно k столкновений?
   3) Чему равно среднее число попыток передачи?
6. Измерения канала дискретной системы ALOHA с бесконечным числом пользователей показали, что 10 % временных интервалов не используется.
   1) Какова загрузка канала G?
   2) Чему равна производительность канала?
   3) Канал перегружен или недогружен?

7. В дискретной системе ALOHA с бесконечным числом пользователей средний период ожидания станции между столкновением и повторной попыткой составляет 4 временных интервала. Нарисуйте зависимость задержки от потока в канале для данной системы.
8. Сколько времени в худшем случае придется ожидать начала передачи станции *s*, если в локальной сети применяется:
   1) базовый протокол бит-карты;
   2) протокол Мока (Mok) и Уорда (Ward) с перестановкой номеров виртуальных станций?
9. В локальной сети используется вариант двоичного обратного отсчета Мока (Mok) и Уорда (Ward). В некоторый момент времени десять станций имеют следующие виртуальные номера: 8, 2, 4, 5, 1, 7, 3, 6, 9, 0. Следующими передающими станциями являются 4, 3 и 9. Какими будут новые виртуальные номера станций после того, как эти три станции закончат свою передачу?
10. Шестнадцать станций, пронумерованных от 1 до 16, соревнуются за право использования общего канала, используя протокол движения по адаптивному дереву. Сколько интервалов времени потребуется для разрешения спора, если все станции, чьи номера являются простыми числами, одновременно станут готовыми к передаче?
11. Группа из 2^n станций использует протокол движения по адаптивному дереву для предоставления доступа к совместно используемому кабелю. В некоторый момент времени 2 станции оказываются готовыми к передаче. Чему равно минимальное, максимальное и среднее число интервалов времени, необходимое для прохождения по дереву, если 2^n много больше 1?
12. Беспроводная локальная сеть, которую мы изучали, использовала такие протоколы, как MACA, вместо CSMA/CD. При каких условиях было бы возможно вместо MACA использовать CSMA/CD?
13. Какие есть общие свойства у протоколов доступа к каналу WDMA и GSM? (Для ответа на этот вопрос см. главу 2, в которой рассказывается о системе GSM.)
14. Шесть станций, отмеченных буквами A — F, взаимодействуют друг с другом по протоколу MACA. Возможна ли ситуация двух одновременных передач данных? Ответ поясните.
15. В семиэтажном офисном здании на каждом этаже расположено по 15 офисов. В каждом офисе на стене установлен разъем для подключения терминала, так что в вертикальной плоскости эти разъемы образуют четырехугольную сетку с расстоянием по 4 м между гнездами как по горизонтали, так и по вертикали. Предполагая, что можно проложить кабель по прямой между любой парой гнезд по горизонтали, вертикали или диагонали, сосчитайте, сколько метров кабеля потребуется для соединения всех гнезд при помощи:
   1) конфигурации «звезда» с одним маршрутизатором посередине;
   2) сети 802.3.

16. Чему равна скорость в бодах стандартной локальной сети Ethernet со скоростью 10 Мбит/с?
17. Как будет выглядеть манчестерский код следующей двоичной последовательности: 0001110101?
18. Как будет выглядеть дифференциальный манчестерский код двоичной последовательности из предыдущего задания? Предполагается, что вначале линия находится в низком состоянии сигнала.
19. В сети с протоколом CSMA/CD (не 802.3) длиной 1 км со скоростью передачи данных 10 Мбит/с скорость распространения сигнала составляет 200 м/мкс. Длина кадров данных равна 256 бит, включая 32 бита заголовка, контрольную сумму и другие накладные расходы. Первый интервал времени после успешной передачи кадра резервируется для передачи получателем 32-битового кадра с подтверждением. Какова эффективная скорость передачи данных без учета накладных расходов, если предположить, что столкновений нет?
20. Две станции в сети с протоколом CSMA/CD пытаются передавать длинные (состоящие из нескольких кадров) файлы. После передачи каждого кадра они состязаются за канал при помощи алгоритма удвоения периода ожидания. Какова вероятность того, что борьба закончится в k-м раунде, и чему равно среднее число раундов в периоде состязания?
21. Как создать сеть CSMA/CD, работающую на скорости 1 Гбит/с по кабелю длиной в 1 км, без повторителей? Скорость распространения сигнала в кабеле равна 200 000 км/с. Чему равен минимальный размер кадра в этой сети?
22. IP-пакет необходимо передать по сети Ethernet. Длина пакета — 60 байт, включая все служебные поля. Если не используется LLC, необходимо ли заполнение Ethernet-кадра? Если да, то сколькими байтами?
23. Кадры Ethernet должны быть не короче 64 байт для того, чтобы в случае коллизии на дальнем конце провода передатчик все еще передавал тот же самый кадр. В сетях типа «быстрый Ethernet» минимальный размер кадра также равен 64 байтам, однако биты могут выдаваться в десять раз чаще, чем в классическом варианте Ethernet. Каким образом в системе удалось сохранить прежний минимальный размер кадра?
24. В некоторых изданиях можно прочесть, что максимальный размер кадра Ethernet равен 1518 байтам (а не 1500 байтам). Ошибаются ли авторы этих изданий? Ответ поясните.
25. Спецификация 1000Base-SX предписывает генератору тактов работать с частотой 1250 МГц, хотя гигабитный Ethernet рассчитан на максимальную скорость1 Гбит/с. Как вы думаете, это завышение частоты является следствием внесения в систему «запаса прочности» или имеются какие-то другие причины?
26. Сколько кадров в секунду может обрабатывать гигабитный Ethernet? Хорошо подумайте перед тем, как отвечать. *Подсказка*: имеет значение тот факт, что это именно *гигабитный* Ethernet.
27. Назовите две сетевые технологии, позволяющие паковать кадры друг за другом. Чем выгодно такое свойство?

28. На рис. 4.24 показаны четыре станции: *A*, *B*, *C* и *D*. Как вы думаете, какая из двух последних станций находится ближе к A, и почему вы так решили?
29. Пусть по 11-мегабитной локальной сети 802.11b передаются друг за другом по радиоканалу 64-байтные кадры с вероятностью ошибки 10^{-7} на бит. Сколько кадров в секунду в среднем будет искажаться при передаче?
30. Сеть стандарта 802.16 обладает каналом с шириной полосы пропускания 20 МГц. Сколько бит в секунду можно отправлять по ней абоненту?
31. Стандарт IEEE 802.16 описывает четыре класса сервисов. Какой из них лучше всего подходит для передачи несжатого видео?
32. Назовите две причины, по которым в сетях может быть предпочтительнее использовать исправление ошибок вместо обнаружения ошибок и повторной передачи.
33. На рис. 4.32 видно, что устройство системы Bluetooth может находиться одновременно в двух пикосетях. Почему одно и то же устройство не может являться главным сразу в двух пикосетях?
34. На рис. 4.22 показано несколько протоколов физического уровня. Какой из них больше всего напоминает протокол физического уровня Bluetooth? В чем состоит основная разница этих двух протоколов?
35. Bluetooth поддерживает два типа соединений главного узла с подчиненным. Как они называются и для чего используются?
36. Сигнальные кадры в варианте стандарта 802.11 со скачкообразными изменениями частот содержат время пребывания. Как вы думаете, содержат ли время пребывания подобные сигнальные кадры в системе Bluetooth?
37. Рассмотрите объединенные сети на рис. 4.40. Предположим, хосты *A* и *B* расположены в ЛВС 1, *C* — в ЛВС 2, а *D* — в ЛВС 8. Изначально хэш-таблицы всех мостов пусты, и используется связующее дерево, показанное на рис. 4.40, *б*. Как изменятся таблицы разных мостов после того как последовательно произойдут следующие события:
    1) *A* отправляет данные для *D*;
    2) *C* отправляет данные для *A*;
    3) *D* отправляет данные для *C*;
    4) *D* переходит в ЛВС 6;
    5) *D* отправляет данные для *A*?
38. Одним из результатов применения связующего дерева для передачи кадров в расширенной ЛВС является то, что некоторые мосты могут вообще не участвовать в работе системы. Найдите три таких моста на рис. 4.40. Есть ли какие-нибудь причины для того, чтобы оставить эти мосты, несмотря на то, что они бездействуют?
39. Допустим, у коммутатора имеются сетевые платы с четырьмя входами. Часто бывает так, что пришедший по одной из линий кадр должен быть перенаправлен на другую линию, подключенную к той же плате. Что должно быть предпринято разработчиками платы для оптимизации работы при такой ситуации?

40. Коммутатор, предназначенный для работы с быстрым Ethernet, имеет объединительную плату, которая может передавать данные со скоростью 10 Гбит/с. Сколько кадров в секунду может быть обработано таким коммутатором?
41. Рассмотрите сеть, показанную на рис. 4.45, *а*. Если машина *J* неожиданно станет «белой», необходимо ли будет вносить какие-либо изменения в маркировку? Если да, то какие?
42. Опишите вкратце разницу между коммутаторами с ожиданием и сквозными коммутаторами.
43. Коммутаторы с ожиданием имеют преимущество перед сквозными коммутаторами при обработке испорченных кадров. Объясните, почему.
44. Чтобы виртуальная сеть заработала, мостам и коммутаторам нужны конфигурационные таблицы. А что если в виртуальных сетях, показанных на рис. 4.45, *а*, использовать вместо моноканала концентраторы? Понадобятся ли им конфигурационные таблицы? Ответ поясните.
45. На рис. 4.46 коммутатор обычного оконечного домена (справа) является ВЛВС-совместимым. Можно было бы поставить здесь обычный коммутатор? Ответ поясните.
46. Напишите программу, симулирующую поведение протокола CSMA/CD в системе Ethernet с *N* станциями, готовыми к передаче во время передачи по каналу кадра. Программа должна выводить временные метки тех моментов, когда каждая из станций смогла успешно начать передачу своего кадра. Пусть часы изменяют свое состояние каждый такт (51,2 мкс), и обнаружение коллизии с отправкой преднамеренной помехи, сообщающей об этом, также занимает один такт. Все кадры имеют максимально допустимую длину.

Глава 5
Сетевой уровень

- Вопросы проектирования сетевого уровня
- Алгоритмы маршрутизации
- Алгоритмы борьбы с перегрузкой
- Качество обслуживания
- Объединение сетей
- Сетевой уровень в Интернете
- Резюме
- Вопросы

Сетевой уровень занимается разработкой маршрутов доставки пакетов от отправителя до получателя. Чтобы добраться до пункта назначения, пакету может потребоваться преодолеть несколько транзитных участков между маршрутизаторами. Функции, выполняемые на сетевом уровне, резко контрастируют с деятельностью уровня передачи данных, цель которого была более скромной — просто переместить кадры с одного конца провода на другой. Таким образом, сетевой уровень оказывается самым низким уровнем, который имеет дело с передачей данных по всему пути от одного конца до другого.

Для достижения этих целей сетевой уровень должен обладать информацией о топологии подсети связи (то есть о множестве всех маршрутизаторов) и выбирать нужный путь по этой подсети. Он должен также заботиться о том, чтобы нагрузка на маршрутизаторы и линии связи была, по возможности, более равномерной. Наконец, если источник и приемник находятся в различных сетях, именно сетевой уровень должен уметь решать проблемы, связанные с различиями в сетях. В данной главе мы рассмотрим все эти аспекты и проиллюстрируем их прежде всего на примере Интернета и его протокола сетевого уровня — IP, хотя и беспроводные сети мы также рассмотрим.

Вопросы проектирования сетевого уровня

В следующих разделах мы рассмотрим некоторые вопросы, с которыми приходится сталкиваться разработчикам сетевого уровня. К этим вопросам относятся сервисы, предоставляемые транспортному уровню, и внутреннее строение подсети.

Метод коммутации пакетов с ожиданием

Прежде чем начать подробное рассмотрение сетевого уровня, необходимо восстановить в памяти окружение, в котором ему приходится функционировать. Оно показано на рис. 5.1. Основными компонентами системы являются устройства оператора связи (маршрутизаторы, соединенные линиями связи), показанные внутри затененного овала, а также устройства, принадлежащие клиенту и показанные вне овала. Хост *H1* напрямую соединен по выделенной линии с одним из маршрутизаторов оператора связи, *A*. Хост *H2*, напротив, находится в ЛВС с маршрутизатором *F*, принадлежащим клиенту, который с ним работает. Этот маршрутизатор связывается с оператором также по выделенной линии. Мы показали *F* вне овала, потому что он не принадлежит оператору связи, однако с точки зрения устройства всей системы и используемых протоколов он ничем не отличается от маршрутизаторов оператора. Можно спорить о том, входит ли он в подсеть, однако в контексте данной главы мы будем считать маршрутизаторы клиента частью подсети, поскольку в них применяются те же самые алгоритмы, что и в маршрутизаторах операторов связи (а основным предметом рассмотрения будут именно алгоритмы).

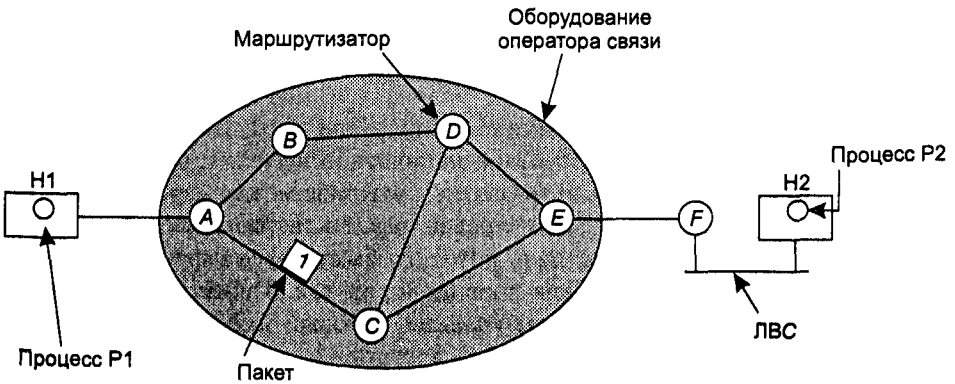


Рис. 5.1. Окружениа, в котором функционируют протоколы сетевого уровня

Система работает следующим образом. Хост, у которого есть пакет для передачи, посылает его либо на ближайший маршрутизатор своей ЛВС, либо по двухточечному соединению оператору связи. Там пакет хранится до тех пор, пока не будет принят целиком, включая верифицируемую контрольную сумму. Затем он передается по цепочке маршрутизаторов, которая в итоге приводит к пункту назначения. Такой механизм называется коммутацией пакетов с ожиданием, и мы уже рассматривали его в предыдущих главах.

Сервисы, предоставляемые транспортному уровню

Сетевой уровень предоставляет транспортному уровню сервисы в виде интерфейса между сетевым и транспортным уровнями. Важным вопросом является то, какой именно вид сервиса сетевой уровень предоставляет транспортному. При разработке сервисов сетевого уровня ставились следующие задачи:

- Сервисы сетевого уровня не должны зависеть от технологии маршрутизатора.
- Транспортный уровень должен быть независим от количества, типа и топологии присутствующих подсетей с маршрутизаторами.
- Сетевые адреса, доступные транспортному уровню, должны использовать единую систему нумерации в локальных и глобальных сетях.

Находясь в рамках поставленной перед ними задачи, разработчики оказываются абсолютно свободными в написании детальной спецификации сервисов, которые должны предоставляться транспортному уровню. Эта свобода часто вырождается в яростную борьбу между двумя непримиримыми группировками. В центре дискуссии оказывается вопрос о том, какие сервисы должен предоставлять сетевой уровень — ориентированные на соединение или не требующие соединений.

Один лагерь (представленный Интернет-сообществом) заявляет, что работа маршрутизатора заключается исключительно в перемещении с места на место пакетов и больше ни в чем. С их точки зрения (основанной на примерно тридцатилетнем опыте работы с реальными компьютерными сетями), подсеть обладает врожденной ненадежностью вне зависимости от того, как она спроектирована. Хосты должны учитывать это и защищаться от ошибок своими силами (то есть заниматься обнаружением и исправлением ошибок), а также самостоятельно управлять потоком.

Из этого следует, что сетевой сервис должен быть сервисом, не требующим установки соединения и состоящим в основном из примитивов SEND PACKET (послать пакет) и RECEIVE PACKET (принять пакет). В частности, сюда нельзя включать упорядочивание пакетов и контроль потока — все равно эти действия будет выполнять хост. От того, что одна и та же работа будет выполнена дважды, качество обслуживания не повысится. Кроме того, каждый пакет должен содержать полный адрес получателя, так как пересылка производится независимо от предшествующих пакетов.

Другой лагерь, представленный телефонными компаниями, возражает, что сеть должна предоставлять надежный, ориентированный на соединение сервис. Они утверждают, что 100 лет успешного управления телефонными системами по всему миру — это серьезный аргумент в их пользу. По их мнению, качество обслуживания является определяющим фактором, и без установления соединения в подсети очень сложно добиться каких-либо приемлемых результатов, особенно когда дело касается трафика реального масштаба времени — например, передачи голоса и видео.

Примерами технологий, защищаемых каждой из сторон, являются Интернет и ATM. Интернет предоставляет не требующие установления соединения сервисы сетевого уровня, а система ATM — ориентированные на соединение. Интересно, что в последнее время вопрос гарантии качества обслуживания становится все более важным, а Интернет при этом активно развивается. В частности, как мы увидим позже, ему все больше вменяются свойства, ассоциирующиеся с сервисами, ориентированными на соединение. Вообще-то мы уже даже намекали на это, когда рассматривали виртуальные сети в главе 4.

Реализация сервиса без установления соединения

Рассмотрев два класса сервисов, которые сетевой уровень может предоставлять своим пользователям, можно перейти к обсуждению устройства этого уровня. Возможны два варианта в зависимости от типа сервиса. Если предоставляется сервис без установления соединения, пакеты внедряются в подсеть по отдельности и их маршруты рассчитываются независимо. При этом никакой предварительной настройки не требуется. В этом случае пакеты часто называют **дейтаграммами**, по аналогии с телеграммами, а подсети, соответственно, — **дейтаграммными**. При использовании сервиса, ориентированного на соединение, путь от маршрутизатора отправителя до маршрутизатора получателя должен быть установлен до начала каких-либо передач пакетов. Такое соединение называется **виртуальным каналом**, по аналогии с физическими каналами, устанавливаемыми в телефонной системе. Подсеть при этом называется **подсетью виртуального канала**. В этом разделе мы обсудим дейтаграммные подсети; в следующем разделе — подсети виртуального канала.

Рассмотрим принцип работы дейтаграммных подсетей. Пусть процесс *P1* (рис. 5.2) хочет послать длинное сообщение для *P2*. Он передает свое послание транспортному уровню, сообщает ему о том, что доставить данные необходимо процессу *P2*, выполняющемуся на хосте *H2*. Код транспортного уровня исполняется на хосте *H1*; более того, обычно он является частью операционной системы. Заголовок транспортного уровня вставляется в начало сообщения, и в таком виде оно передается на сетевой уровень. Обычно это просто еще одна процедура операционной системы.

Предположим, что сообщение в четыре раза длиннее максимального размера пакета, поэтому сетевой уровень должен разбить его на четыре пакета (1, 2, 3 и 4) и послать их все поочередно на маршрутизатор *A* с использованием какого-нибудь протокола двухточечного соединения, например PPP. Здесь вступает в игру оператор связи. Каждый маршрутизатор имеет свою внутреннюю таблицу, по которой он определяет дальнейший путь пакета при каждом из возможных адресов назначения. Каждая запись таблицы состоит из двух полей: пункт назначения (адресат) и выходящая линия для данного адресата. Во втором поле могут использоваться только линии, непосредственно соединенные с данным маршрутизатором. Так, например, на рис. 5.2 у маршрутизатора *A* имеются только две исходящие линии — ведущие к *B* и к *C*, поэтому все входящие пакеты должны

пересылаться на какой-то из этих двух маршрутизаторов, даже если они не являются адресатами. Изначальная таблица маршрутизации *A* показана на рисунке под соответствующей надписью.

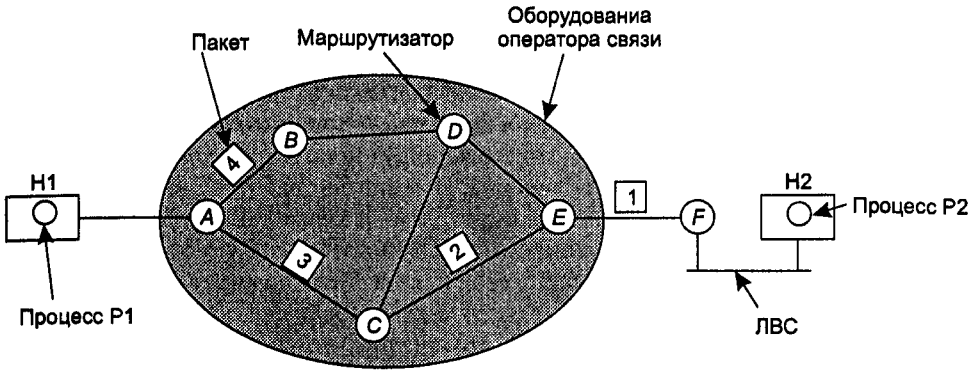


Таблица маршрутизатора *A*

В начале		В конце		Таблица маршрутизатора *C*		Таблица маршрутизатора *E*	
A	–	*A*	–	*A*	*A*	*A*	*C*
B	*B*	*B*	*B*	*B*	*A*	*B*	*D*
C	*C*	*C*	*C*	*C*	–	*C*	*C*
D	*B*	*D*	*B*	*D*	*D*	*D*	*D*
E	*C*	*E*	*B*	*E*	*E*	*E*	–
F	*C*	*F*	*C*	*F*	*E*	*F*	*F*

Назначение Линия

Рис. 5.2. Маршрутизация внутри дейтаграммной подсети

Пакеты 1, 2 и 3, прибывая на маршрутизатор *A*, кратковременно сохраняются для того, чтобы можно было проверить их корректную доставку по контрольной сумме. Затем в соответствии с таблицей *A* все они пересылаются на маршрутизатор *C*. После этого пакет 1 уходит на *E*, откуда доставляется на маршрутизатор локальной сети, *F*. Когда он прибывает на *F*, он инкапсулируется в кадр уровня передачи данных и передается на хост *H2* по локальной вычислительной сети. Пакеты 2 и 3 следуют по тому же маршруту.

Однако с пакетом 4 связана несколько иная история. После прибытия на *A* он пересылается на маршрутизатор *B* несмотря на то, что адресом назначения является *F*, как и у первых трех пакетов. По каким-то своим причинам маршрутизатор *A* решил послать пакет 4 по новому маршруту. Может быть, это стало следствием затора где-то на линии ACE, возникшего при пересылке трех пакетов, в результате чего маршрутизатор решил обновить свою таблицу (на рисунке показана под надписью «В конце»). Алгоритм, управляющий таблицами маршрутизации и принимающий решения, называется **алгоритмом маршрутизации**. Именно изучению алгоритмов маршрутизации будет уделено основное внимание в этой главе.

Реализация сервиса с установлением соединения

Сервису с установлением соединения нужна подсеть виртуального канала. Рассмотрим ее работу. Идея виртуальных каналов состоит в предотвращении выбора своего маршрута для каждого пакета, как было показано на рис. 5.2. Вместо этого устанавливается соединение, маршрут от отправляющей до получающей машины прописывается в настройках системы и хранится в специальных таблицах, встроенных в маршрутизаторы. Один и тот же маршрут используется для всего трафика, проходящего через данное соединение. Именно так работает телефонная система. Когда соединение разрывается, виртуальный канал также прекращает свое существование. При использовании сервиса, ориентированного на установление соединения, каждый пакет включает в себя идентификатор виртуального канала.

В качестве примера рассмотрим ситуацию, изображенную на рис. 5.3. Хост *H1* установил соединение с хостом *H2*. Это соединение запоминается и становится первой записью во всех таблицах маршрутизации. Так, первая строчка таблицы маршрутизатора *A* говорит о том, что если пакет с идентификатором соединения 1 пришел с хоста *H1*, то его нужно направить на *C* с идентификатором соединения 1. Точно так же первая запись *C* направляет пакет на *E* все с тем же идентификатором соединения 1.

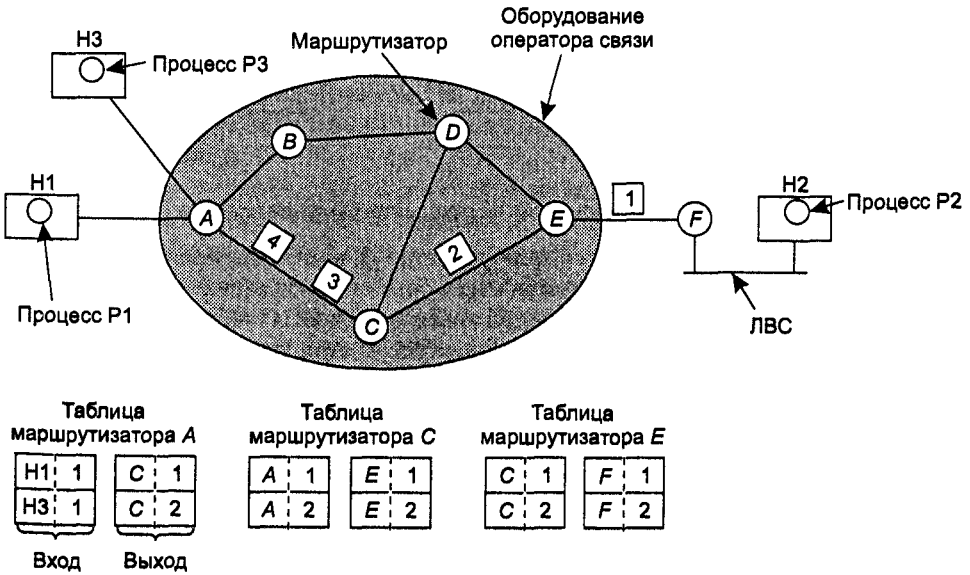


Рис. 5.3. Маршрутизация в подсети виртуального канала

Теперь рассмотрим, что будет, если хост *H3* захочет установить соединение с *H2*. Он выбирает идентификатор соединения 1 (у него просто нет выбора, поскольку это на данный момент единственное существующее соединение) и просит подсеть установить виртуальный канал. Таким образом, в таблице появляет-

ся вторая запись. Обратите внимание на то, что здесь возникает, на самом деле, конфликт, потому что если *A* еще может отличить пакеты соединения 1, пришедшие с *H1*, от пакетов соединения 1, пришедших с *H3*, то *C* такой возможности не имеет. По этой причине *A* присваивает новый идентификатор соединения исходящему трафику и тем самым создает второе соединение. Предотвращение конфликтов подобного рода является причиной того, почему маршрутизаторам нужна возможность изменения идентификаторов соединения в исходящих пакетах. Иногда это называется коммутацией меток.

Сравнение подсетей виртуальных каналов и дейтаграммных подсетей

Как виртуальные каналы, так и дейтаграммы имеют своих сторонников и противников. Попробуем обобщить аргументы обеих сторон. Основные аспекты сведены в табл. 5.1, хотя наверняка можно найти контраргументы для каждого пункта таблицы.

Таблица 5.1. Сравнение виртуальных каналов и дейтаграмм

Проблема	Дейтаграммы	Виртуальные каналы
Установка канала	Не требуется	Требуется
Адресация	Каждый пакет содержит полный адрес отправителя и получателя	Каждый пакет содержит короткий номер виртуального канала
Информация о состоянии	Подсеть не содержит информации о состоянии	Каждый виртуальный канал требует места в таблице подсети
Маршрутизация	Маршрут каждого пакета выбирается независимо	Маршрут выбирается при установке виртуального канала. Каждый пакет следует по этому маршруту
Эффект от выхода из строя маршрутизатора	Никакого, кроме потерянных пакетов	Все виртуальные каналы, проходившие через отказавший маршрутизатор, прекращают существование
Борьба с перегрузкой	Трудно реализовать	Легко реализуется при наличии достаточного количества буферов для каждого виртуального канала

Оба подхода к созданию подсетей в ряде вопросов находят некие компромиссы. Во-первых, существует компромисс между внутренней памятью маршрутизатора и пропускной способностью. Виртуальные каналы позволяют экономить на адресах получателя, указывая вместо них короткие номера виртуальных каналов. Если размер кадра мал, полный адрес получателя может составлять довольно существенную часть всего кадра, сильно снижая полезную пропускную способность сети. Однако для хранения в маршрутизаторе больших таблиц с адресами может потребоваться много памяти.

Второй компромисс — между временем установки соединения и временем обработки адреса. Виртуальный канал требует определенных затрат времени на его установку, однако последующая обработка пакетов для маршрутизатора оказывается проще и быстрее, чем в дейтаграммной подсети.

Виртуальные каналы обладают некоторыми преимуществами, помогающими им предоставлять гарантированное качество обслуживания и избегать заторов в подсети, так как ресурсы могут быть зарезервированы заранее, во время установки соединения. Когда начинают прибывать пакеты, необходимая пропускная способность и мощность маршрутизатора будут предоставлены. В дейтаграммной подсети предотвращение заторов реализовать значительно сложнее.

В системах обработки транзакций (например, при запросе магазина на верификацию кредитной карты) накладные расходы на установку соединения и удаление виртуального канала могут сильно снизить потребительские свойства сети. Если объем информации, передаваемой во время одного соединения, невелик, то использование виртуального канала не имеет смысла. Однако в данной ситуации могут оказаться полезными постоянные виртуальные каналы, установленные вручную и не разрываемые месяцами и даже годами.

Недостатком виртуальных каналов является их уязвимость в случае выхода из строя или временного выключения маршрутизатора. Даже если он будет быстро починен и снова включен, все виртуальные каналы, проходившие через него, будут прерваны. Если же в дейтаграммной подсети маршрутизатор выйдет из строя, то будут потеряны только те пакеты, которые находились в данный момент на маршрутизаторе (а возможно, лишь некоторые из них, в зависимости от того, были они подтверждены или нет). Обрыв линии связи для виртуальных каналов является фатальным, а в дейтаграммной системе может оказаться почти незамеченным. Кроме того, дейтаграммная система позволяет соблюдать баланс между загрузкой маршрутизаторов и линий связи.

Алгоритмы маршрутизации

Основная функция сетевого уровня заключается в выборе маршрута для пакетов от начальной до конечной точки. В большинстве сетей пакетам приходится проходить через несколько маршрутизаторов. Единственным исключением являются широковещательные сети, но даже в них маршрутизация является важным вопросом, если отправитель и получатель находятся в разных сетях. Алгоритмы выбора маршрутов и используемые ими структуры данных являются главной целью при проектировании сетевого уровня.

Алгоритм маршрутизации реализуется той частью программного обеспечения сетевого уровня, которая отвечает за выбор выходной линии для отправки пришедшего пакета. Если подсеть использует дейтаграммную службу, выбор маршрута для каждого пакета должен производиться заново, так как оптимальный маршрут мог измениться. Если подсеть использует виртуальные каналы, маршрут выбирается только при создании нового виртуального канала. После этого все информационные пакеты следуют по выбранному маршруту. Последний

случай иногда называют **сеансовой маршрутизацией**, так как маршрут остается в силе на протяжении всего сеанса связи с пользователем (например, сеанса регистрации на терминале или передачи файла).

Полезно понимать разницу между маршрутизацией, при которой системе приходится делать выбор определенного маршрута следования, и пересылкой — действием, происходящим при получении пакета. Можно представить себе маршрутизатор как устройство, в котором функционируют два процесса. Один из них обрабатывает приходящие пакеты и выбирает для них по таблице маршрутизации исходящую линию. Такой процесс называется **пересылкой**. Второй процесс отвечает за заполнение и обновление таблиц маршрутизации. Именно здесь в игру вступает алгоритм маршрутизации.

Вне зависимости от того, отдельно ли выбираются маршруты для каждого пакета или же только один раз для соединения, желательно, чтобы алгоритм выбора маршрута обладал определенными свойствами — корректностью, простотой, надежностью, устойчивостью, справедливостью и оптимальностью. Правильность и простота вряд ли требуют комментариев, а вот потребность в надежности не столь очевидна с первого взгляда. Во время работы большой сети постоянно происходят какие-то отказы аппаратуры и изменения топологии. Алгоритм маршрутизации должен уметь справляться с изменениями топологии и трафика без необходимости прекращения всех задач на всех хостах и перезагрузки сети при каждой поломке маршрутизатора.

Алгоритм маршрутизации должен также обладать устойчивостью. Существуют алгоритмы выбора маршрута, никогда не приходящие в состояние равновесия, независимо от того, как долго они работают. Такие цели, как справедливость и оптимальность, могут показаться очевидными — вряд ли кто-нибудь станет возражать против них, — однако они зачастую оказываются взаимоисключающими. Для примера рассмотрим ситуацию, показанную на рис. 5.4. Предположим, что трафик между станциями A и A', B и B', а также C и C' настолько интенсивный, что горизонтальные линии связи оказываются полностью насыщенными. Чтобы максимизировать общий поток данных, трафик между станциями X и X' следовало бы совсем отключить. Однако станции X и X', скорее всего, имеют другую точку зрения по данному вопросу. Очевидно, необходим компромисс между справедливым выделением трафика всем станциям и оптимальным использованием канала в глобальном смысле.

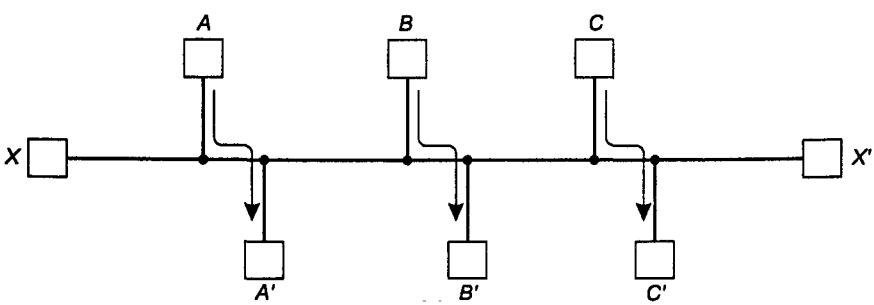


Рис. 5.4. Конфликт справедливости и оптимальности

Прежде чем пытаться искать приемлемое соотношение справедливости и оптимальности, следует решить, что именно мы будем стремиться оптимизировать. Можно попробовать минимизировать среднее время задержки или увеличить общую пропускную способность сети. Однако эти цели также противоречат друг другу, поскольку работа любой системы с очередями вблизи максимума производительности предполагает долгое стояние в очередях. В качестве компромисса многие сети стараются минимизировать количество пересылок для каждого пакета, поскольку при этом снижается время прохождения пакета по сети, а также снижается нагрузка на сеть, в результате чего улучшается пропускная способность.

Алгоритмы выбора маршрута можно разбить на два основных класса: адаптивные и неадаптивные. **Неадаптивные алгоритмы** не учитывают при выборе маршрута топологию и текущее состояние сети и не измеряют трафик на линиях. Вместо этого выбор маршрута для каждой пары станций производится заранее, в автономном режиме, и список маршрутов загружается в маршрутизаторы во время загрузки сети. Такая процедура иногда называется **статической маршрутизацией**.

Адаптивные алгоритмы, напротив, изменяют решение о выборе маршрутов при изменении топологии и также часто в зависимости от загруженности линий. Адаптивные алгоритмы отличаются источниками получения информации (такие источники могут быть, например, локальными, если это соседние маршрутизаторы, либо глобальными, если это вообще все маршрутизаторы сети), моментами изменения маршрутов (например, через определенные равные интервалы времени, при изменении нагрузки или при изменении топологии) и данными, использующимися для оптимизации (расстояние, количество транзитных участков или ожидаемое время пересылки). В следующих разделах мы обсудим различные алгоритмы маршрутизации, как статические, так и динамические.

Принцип оптимальности маршрута

Прежде чем перейти к рассмотрению отдельных алгоритмов, возможно, следует привести некие общие положения, описывающие оптимальные маршруты, вне зависимости от топологии или трафика. Такой основополагающей идеей является **принцип оптимальности**. В соответствии с этим принципом, если маршрутизатор B располагается на оптимальном маршруте от маршрутизатора A к маршрутизатору C, то оптимальный маршрут от маршрутизатора B к маршрутизатору C совпадет с частью первого маршрута. Чтобы убедиться в этом, обозначим часть маршрута от маршрутизатора A к маршрутизатору B как r_1, а остальную часть маршрута — r_2. Если бы существовал более оптимальный маршрут от маршрутизатора B к маршрутизатору C, чем r_2, то его можно было объединить с r_1, чтобы улучшить маршрут от маршрутизатора A к маршрутизатору C, что противоречит первоначальному утверждению о том, что маршрут r_1r_2 является оптимальным.

Прямым следствием принципа оптимальности является возможность рассмотрения множества оптимальных маршрутов от всех источников к приемникам в виде дерева. Такое дерево называется **входным деревом**. Оно изображено

на рис. 5.5. Расстояния измеряются количеством транзитных участков. Обратите внимание на то, что входное дерево не обязательно является уникальным. У одной сети могут существовать несколько входных деревьев с одинаковыми длинами путей. Цель всех алгоритмов выбора маршрутов заключается в вычислении и использовании входных деревьев для всех маршрутизаторов.

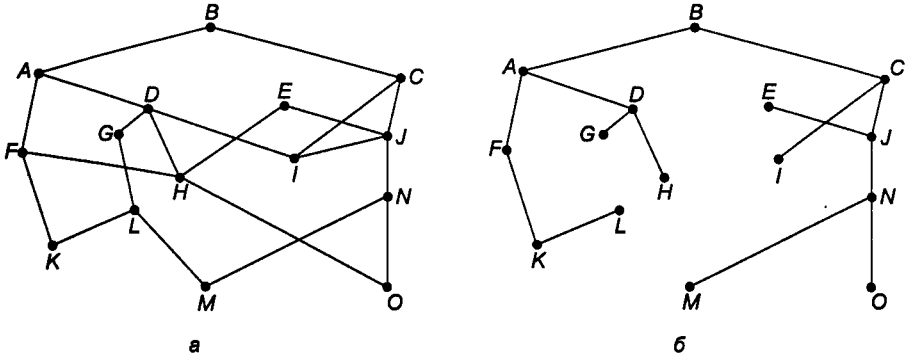


Рис. 5.5. Подсеть (*а*); входное дерево для маршрутизатора *B* (*б*)

Поскольку входное дерево действительно является деревом, оно не содержит петель, поэтому каждый пакет будет доставлен за конечное и ограниченное число пересылок. На практике все это не так просто. Линии связи и маршрутизаторы могут выходить из строя и снова появляться в сети во время выполнения операции, поэтому у разных маршрутизаторов могут оказаться различные представления о текущей топологии сети. Кроме того, мы обошли вопрос о том, собирает ли маршрутизатор информацию для вычисления входного дерева сам или эта информация каким-то другим образом поступает к нему. Мы вскоре рассмотрим этот вопрос. Тем не менее, принцип оптимальности и входное дерево — это те точки отсчета, относительно которых можно измерять эффективность различных алгоритмов маршрутизации.

Выбор кратчайшего пути

Начнем наше изучение алгоритмов выбора маршрутов с метода, широко применяемого в различных формах благодаря его простоте и понятности. Идея заключается в построении графа подсети, в котором каждый узел будет соответствовать маршрутизатору, а каждая дуга — линии связи (часто называемой просто связью). При выборе маршрута между двумя маршрутизаторами алгоритм просто находит кратчайший путь между ними на графе.

Концепция **кратчайшего пути** требует некоторого пояснения. Один из способов измерения длины пути состоит в подсчете количества транзитных участков. В таком случае пути *ABC* и *ABE* на рис. 5.6 имеют одинаковую длину. Можно измерять расстояния в километрах. В таком случае окажется, что путь *ABC* значительно длиннее пути *ABE* (предполагается, что рисунок изображен с соблюдением масштаба).

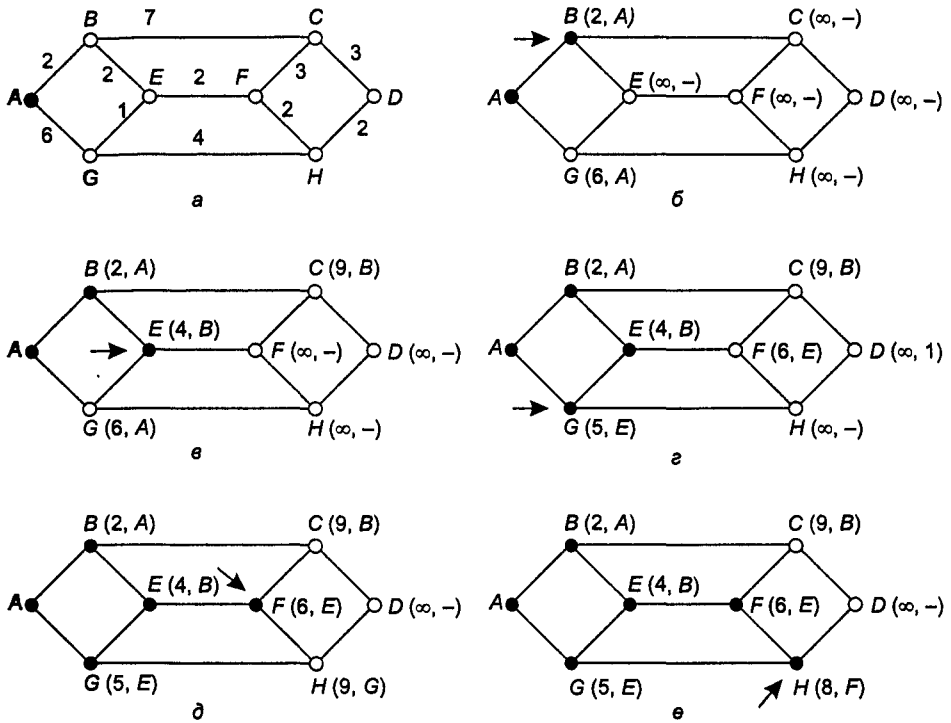


Рис. 5.6. Первые пять шагов вычисления кратчайшего пути от *A* к *D*. Стрелка указывает на рабочий узел

Однако помимо учета количества транзитных участков и физической длины линий возможен учет и других параметров. Например, каждой дуге графа можно поставить в соответствие среднюю длину очереди и время задержки пересылки, определяемые каждый час с помощью передачи специального тестового пакета. В таком графе кратчайший путь определяется как самый быстрый путь, а не путь с самой короткой длиной кабеля или путь, состоящий из минимального числа отдельных отрезков кабеля.

В общем случае параметры дуг графа являются функциями расстояния, пропускной способности, средней загруженности, стоимости связи средней длины очереди, измеренной величины задержки и других факторов. Изменяя весовую функцию, алгоритм может вычислять кратчайший путь с учетом любого количества критериев в различных комбинациях.

Известно несколько алгоритмов вычисления кратчайшего пути между двумя узлами графа. Один из них был создан знаменитым Дейкстрой (Dijkstra) в 1959 году. Каждый узел помечается (в скобках) расстоянием до него от узла отправителя по наилучшему известному пути. Вначале пути неизвестны, поэтому все узлы помечаются символом бесконечности. По мере работы алгоритма и нахождения путей отметки узлов изменяются, показывая оптимальные пути. Отметка может быть постоянной или экспериментальной. Вначале все отметки являются ориен-

тировочными. Когда выясняется, что отметка действительно соответствует кратчайшему возможному пути, она становится постоянной и в дальнейшем не изменяется.

Чтобы показать, как работает этот алгоритм, рассмотрим взвешенный ненаправленный граф (рис. 5.6, *а*), где весовые коэффициенты соответствуют, например, расстояниям. Мы хотим найти кратчайший путь от *A* к *D*. Для начала мы черным кружком помечаем узел *A* как постоянный. Затем мы исследуем все соседние с ним узлы, указывая около них расстояние до узла *A*. Если отыскивается более короткий путь к какому-либо узлу, то вместе с указанием расстояния в отметке меняется и узел, через который прошел более короткий путь. Таким образом, позднее можно восстановить весь путь. Рассмотрев все соседние с *A* узлы, мы помечаем ближайший узел как постоянный, как показано на рис. 5.6, *б*. Этот узел становится новым рабочим узлом.

Теперь мы повторяем ту же процедуру с узлом *B*, исследуя все его соседние узлы. Если сумма расстояния от узла *B* и значения отметки в узле *B* (расстояния от *A* до *B*) оказывается меньше, чем отметка у исследуемого узла (уже найденное другим путем расстояние от *A*), это значит, что найден более короткий путь, поэтому пометка узла изменяется.

После того как все соседние с рабочим узлы исследованы и временные отметки при необходимости изменены, по всему графу ищется узел с наименьшей временной отметкой. Этот узел помечается как постоянный и становится текущим рабочим узлом. На рис. 5.6 показаны первые пять этапов работы алгоритма.

Чтобы понять, как работает алгоритм, посмотрим на рис. 5.6, *в*. На данном этапе узел *E* только что был отмечен как постоянный. Предположим, что существует более короткий путь, нежели *ABE*, например *AXYZE*. В этом случае есть две возможности — либо узел *Z* уже сделан постоянным, либо еще нет. Если да, значит, узел *E* уже проверялся, когда узел *Z* был сделан постоянным и, следовательно, рабочим узлом. В этом случае путь *AXYZE* уже исследовался.

Теперь рассмотрим случай, когда узел *Z* все еще помечен как временный. В этом случае отметка узла *Z* либо больше или равна пометки узла *E*, либо меньше ее. В первом случае путь *AXYZE* не может быть короче, чем путь *ABE*. Если же отметка узла *Z* меньше пометки узла *E*, то тогда узел *Z* должен был стать постоянным раньше узла *E*, и узел *E* проверялся бы с узла *Z*.

Этот алгоритм приведен в листинге 5.1. Глобальные переменные *n* и *dist* описывают граф и инициализируются раньше, чем вызывается *shortest_path*. Единственное отличие программы от описанного выше алгоритма заключается в том, что вычисление кратчайшего пути в программе начинается не от узла-источника *s*, а от конечного узла *t*. Поскольку в однонаправленном графе кратчайший путь от *t* к *s* тот же самый, что и от *s* к *t*, не имеет значения, с какого конца начинать. (Если только не существует несколько кратчайших путей. В таком случае, двигаясь в противоположном направлении, мы можем обнаружить другой путь.) Причина поиска пути в обратном направлении заключается в том, что каждый узел помечается предшествующим узлом, а не следующим. Когда найденный путь копируется в выходную переменную *path*, он инвертируется. С помощью реверсивного поиска убирается неправильность направления поиска, и мы получаем путь, идущий в нужную сторону.

Листинг 5.1. Алгоритм Дейкстры вычисления кратчайшего пути по графу

```
#define MAX_NODES 1024                /* максимальное количество узлов */
#define INFINITY 1000000000           /* число, большее длины максимального пути */
int n, dist[MAX_NODES][MAX_NODES]; /* dist[i][j] - это расстояние от i до j */
void shortest_path(int s, int t, int path[])
{ struct state {                      /* рабочий путь */
    int predecessor;                  /* предыдущий узел */
    int length;                       /* расстояние от источника до этого узла */
    enum {permanent, tentative} label; /* метка состояния */
  } state[MAX_NODES];
  int i, k, min;
  struct state *p;
  for (p = &state[0]; p < &state[n]; p++) { /* инициализировать состояние */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0; state[t].label = permanent;
  k = t;                                  /* k - начальный рабочий узел */
  do {                                    /* Есть ли лучший путь от k? */
      for (i = 0; i < n; i++)             /* У этого графа n узлов */
          if (dist[k][i] != 0 && state[i].label == tentative) {
              if (state[k].length + dist[k][i] < state[i].length) {
                  state[i].predecessor = k;
                  state[i].length = state[k].length + dist[k][i];
              }
          }
      /* Найти узел, помеченный как предварительный с наименьшей меткой */
      k = 0; min = INFINITY;
      for (i = 0; i < n; i++)
          if (state[i].label == tentative && state[i].length < min) {
              min = state[i].length;
              k =i;
          }
      state[k].label = permanent;
  } while (k != s);
  /* Копировать путь в выходной массив */
  i = 0; k = s;
  do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Заливка

Метод **заливки** представляет собой еще один статический алгоритм, при котором каждый приходящий пакет посылается на все исходящие линии, кроме той, по которой пришел пакет. Очевидно, что алгоритм заливки порождает огромное количество дублированных пакетов, даже бесконечное количество в сетях с замкнутыми контурами, если не принять специальных мер. Одна из таких мер состоит в помещении в заголовок пакета счетчика преодоленных им транзитных участков, уменьшаемого при прохождении каждого маршрутизатора. Когда значение этого счетчика падает до нуля, пакет удаляется. В идеальном случае счетчик транзит-

ных участков должен вначале устанавливаться равным длине пути от отправителя до получателя. Если отправитель не знает расстояния до получателя, он может установить значение счетчика равным длине максимального пути (диаметру) в данной подсети.

Альтернативный способ ограничения количества тиражируемых пакетов заключается в учете проходящих через маршрутизатор пакетов. Это позволяет не посылать их повторно. Один из методов состоит в том, что каждый маршрутизатор помещает в каждый получаемый от своих хостов пакет порядковый номер. Все маршрутизаторы ведут список маршрутизаторов-источников, в котором сохраняются все порядковые номера пакетов, которые им встречались. Если пакет от данного источника с таким порядковым номером уже есть в списке, он дальше не распространяется и удаляется.

Чтобы предотвратить неограниченный рост размера списка, можно снабдить все списки счетчиком k, показывающим, что все порядковые номера вплоть до k уже встречались. И когда приходит пакет, можно очень легко проверить, не является ли он дубликатом. При положительном ответе такой пакет отвергается. Кроме того, не нужно хранить весь список до k, поскольку этот счетчик очень действенно подытоживает его.

На практике чаще применяется вариант данного алгоритма под названием **выборочная заливка**. В этом алгоритме маршрутизаторы посылают пакеты не по всем линиям, а только по тем, которые идут приблизительно в нужном направлении. Вряд ли есть смысл посылать пакет, направляющийся на запад, по линии, идущей на восток, если только топология сети не представляет собой лабиринт и маршрутизатор не знает об этом.

В большинстве случаев алгоритм заливки является непрактичным, но, тем не менее, иногда он применяется. Например, в военных приложениях, где большая часть маршрутизаторов в любой момент может оказаться уничтоженной, высокая надежность алгоритма заливки является, наоборот, желательной. В распределенных базах данных иногда бывает необходимо одновременно обновить все базы данных, и в этом случае заливка также оказывается полезной. Третье применение алгоритма заливки — эталонное тестирование других алгоритмов выбора маршрутов, так как заливка всегда находит все возможные пути в сети, а следовательно, и кратчайшие. Ухудшить эталонные показатели времени доставки могут разве что накладные расходы, вызванные огромным количеством пакетов, формируемых самим алгоритмом заливки.

Маршрутизация по вектору расстояний

Современные компьютерные сети обычно используют не статические, а динамические алгоритмы маршрутизации, поскольку статические просто не принимают во внимание текущую нагрузку на сеть. Самой большой популярностью пользуются два метода: маршрутизация по вектору расстояний и маршрутизация с учетом состояния каналов. В этом разделе мы изучим первый, в следующем — второй метод.

Алгоритмы маршрутизации по вектору расстояний работают, опираясь на таблицы (то есть векторы), поддерживаемые всеми маршрутизаторами и содержащие наилучшие известные пути к каждому из возможных адресатов. Для обновления данных этих таблиц производится обмен информацией с соседними маршрутизаторами.

Алгоритм маршрутизации по вектору расстояний иногда называют по именам его создателей распределенным алгоритмом **Беллмана—Форда** (Bellman—Ford) и алгоритмом **Форда—Фулкерсона** (Ford—Fulkerson), (Bellman, 1957; Ford and Filkerson, 1962). Этот алгоритм изначально применялся в сети ARPANET и в Интернете был известен под названием RIP.

При маршрутизации по вектору расстояний таблицы, с которыми работают и которые обновляют маршрутизаторы, содержат записи о каждом маршрутизаторе подсети. Каждая запись состоит из двух частей: предпочитаемого номера линии для данного получателя и предполагаемого расстояния или времени прохождения пакета до этого получателя. В качестве единиц измерения может использоваться число транзитных участков, миллисекунды, число пакетов, ожидающих в очереди в данном направлении, или еще что-нибудь подобное.

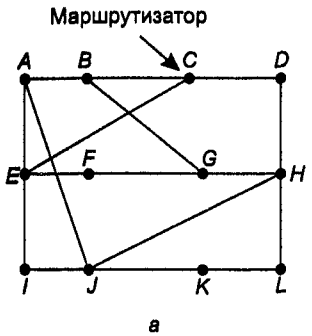
Предполагается, что маршрутизаторам известно расстояние до каждого из соседей. Если в качестве единицы измерения используется число транзитных участков, то расстояние равно одному транзитному участку. Если же дистанция измеряется временем задержки, то маршрутизатор может измерить его с помощью специального пакета ECHO (эхо), в который получатель помещает время получения и который отправляет обратно как можно быстрее.

Предположим, что в качестве единицы измерения используется время задержки, и этот параметр относительно каждого из соседей известен маршрутизатору. Через каждые T миллисекунды все маршрутизаторы посылают своим соседям список с приблизительными задержками для каждого получателя. Они, разумеется, также получают подобный список от всех своих соседей. Допустим, одна из таких таблиц пришла от соседа X, и в ней указывается, что время распространения от маршрутизатора X до маршрутизатора i равно X_i. Если маршрутизатор знает, что время пересылки до маршрутизатора X равно m, тогда задержка при передаче пакета маршрутизатору i через маршрутизатор X составит $X_i + m$. Выполнив такие расчеты для всех своих соседей, маршрутизатор может выбрать наилучшие пути и поместить соответствующие записи в новую таблицу. Обратите внимание на то, что старая таблица в расчетах не используется.

Процесс обновления таблицы проиллюстрирован на рис. 5.7. Слева показана подсеть (рис. 5.7, *а*). Первые четыре столбца на рис. 5.7, *б* показывают векторы задержек, полученные маршрутизатором J от своих соседей. Маршрутизатор A считает, что время пересылки от него до маршрутизатора B равно 12 мс, 25 мс — до маршрутизатора C, 40 мс — до D и т. д. Предположим, что маршрутизатор J измерил или оценил задержки до своих соседей A, I, H и K как 8, 10, 12 и 6 мс соответственно.

Теперь рассмотрим, как J рассчитывает свой новый маршрут к маршрутизатору G. Он знает, что задержка до A составляет 8 мс, а A думает, что от него до G

данные дойдут за 18 мс. Таким образом, *J* знает, что если он станет отправлять пакеты для *G* через *A*, то задержка составит 26 мс. Аналогично он вычисляет значения задержек для маршрутов от него до *G*, проходящих через остальных его соседей (*I*, *H* и *K*), и получает соответственно 41 (31 + 10), 18 (6 + 12) и 37 (31 + 6). Лучшим значением является 18, поэтому именно оно помещается в таблицу в запись для получателя *G*. Вместе с числом 18 туда же помещается обозначение линии, по которой проходит самый короткий маршрут до *G*, то есть *H*. Данный метод повторяется для всех остальных адресатов, и при этом получается новая таблица, показанная в виде правого столбца на рисунке.



а

K	A	I	H	K	Новая расчатная задержка для J	Линия
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	8	K
L	29	33	9	9	15	K

Задержка JA равна 8 | Задержка JI равна 10 | Задаржка JH равна 12 | Задаржка JK раана 6

Новая таблица маршрутов для J

Векторы, получанныа от четырех соседей J

б

Рис. 5.7. Подсеть (а); полученные от *A*, *I*, *H* и *K* векторы и новая таблица маршрутов для *J* (*б*)

Проблема счета до бесконечности

Алгоритм маршрутизации по вектору расстояний работает в теории, но обладает серьезным недостатком на практике: хотя правильный ответ в конце концов и находится, процесс его поиска может занять довольно много времени. В частности, такой алгоритм быстро реагирует на хорошие новости и очень лениво — на плохие. Рассмотрим маршрутизатор, у которого задержка до маршрутизатора *X* велика. Если при очередном обмене векторами его сосед сообщит ему, что от него до маршрутизатора *X* совсем недалеко, наш маршрутизатор просто переключится для передач маршрутизатору *X* на линию, проходящую через этого соседа. Таким образом, хорошая новость распространилась всего за один обмен информацией.

Чтобы увидеть, как быстро распространяются хорошие известия, рассмотрим линейную подсеть из пяти узлов, показанную на рис. 5.8, в которой мерой расстояния служит количество транзитных участков. Предположим, что вначале маршрутизатор *A* выключен и все остальные маршрутизаторы об этом знают. То есть они считают, что расстояние до *A* равно бесконечности.

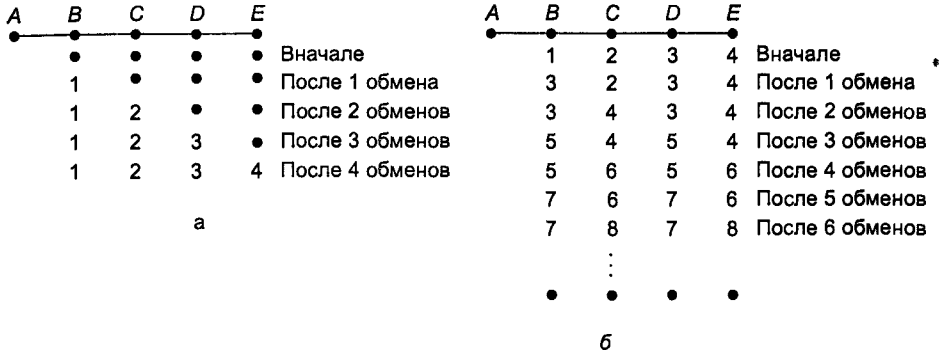


Рис. 5.8. Проблема счета до бесконечности

Когда в сети появляется *A*, остальные маршрутизаторы узнают об этом с помощью обмена векторами. Для простоты будем предполагать, что где-то в сети имеется гигантский гонг, в который периодически ударяют, чтобы инициировать одновременный обмен векторами. После первого обмена *B* узнает, что у его соседа слева нулевая задержка при связи с *A*, а *B* помечает в своей таблице маршрутов, что *A* находится слева на расстоянии одного транзитного участка. Все остальные маршрутизаторы в этот момент еще полагают, что *A* выключен. Значения задержек для *A* в таблицах на этот момент показаны во второй строке на рис. 5.8, *а*. При следующем обмене информацией *C* узнает, что у *B* есть путь к *A* длиной 1, поэтому он обновляет свою таблицу, указывая длину пути до *A*, равную 2, но *D* и *E* об этом еще не знают. Таким образом, хорошие вести распространяются со скоростью один транзитный участок за один обмен векторами. Если самый длинный путь в подсети состоит из *N* транзитных участков, то через *N* обменов все маршрутизаторы подсети будут знать о включенных маршрутизаторах и заработавших линиях.

Теперь рассмотрим ситуацию на рис. 5.8, *б*, в которой все линии и маршрутизаторы изначально находятся во включенном состоянии. Маршрутизаторы *B*, *C*, *D* и *E* находятся от *A* на расстоянии 1, 2, 3 и 4 соответственно. Внезапно *A* отключается или, возможно, происходит обрыв линии между *A* и *B*, что с точки зрения *B* одно и то же.

При первом обмене пакетами *B* не слышит ответа от *A*. К счастью, *C* говорит: «Не волнуйся. У меня есть путь к *A* длиной 2». *B* не знает, что путь от *C* к *A* проходит через *B*. *B* может только предполагать, что у *C* около 10 выходных линий с независимыми путями к *A*, кратчайшая из которых имеет длину 2. Поэтому теперь *B* думает, что может связаться с *A* через *C* по пути длиной 3. При этом первом обмене маршрутизаторы *D* и *E* не обновляют свою информацию об *A*.

При втором обмене векторами *C* замечает, что у всех его соседей есть путь к *A* длиной 3. Он выбирает один из них случайным образом и устанавливает свое расстояние до *A* равным 4, как показано в третьей строке на рис. 5.8, *б*. Результаты последующих обменов векторами также показаны на этом рисунке.

Теперь должно быть понятно, почему плохие новости медленно распространяются — ни один маршрутизатор не может установить значение расстояния, более чем на единицу превышающее минимальное значение этого расстояния, хранящееся у его соседей. Таким образом, все маршрутизаторы будут до бесконечности увеличивать значение расстояния до выключенного маршрутизатора. Количество необходимых для завершения этого процесса обменов векторами можно ограничить, если установить значение этой «бесконечности» равным длине самого длинного пути плюс 1. Если расстояния в подсети измеряются во временных задержках, такую верхнюю границу выбрать сложнее. Следует выбирать достаточно большое ограничивающее значение, чтобы линия с большой задержкой не была сочтена совсем не работающей. Неудивительно, что эта проблема называется **счетом до бесконечности**. Было сделано несколько попыток решить ее (например, алгоритм расколотого горизонта, весьма бесполезный во многих случаях, но внесенный в RFC 1058), но все они оказались безуспешными. Суть проблемы заключается в том, что когда *X* сообщает *Y* о том, что у него есть какой-то путь, у *Y* нет никакой возможности узнать, входит ли он сам в этот путь.

Маршрутизация с учетом состояния линий

Маршрутизация на основе векторов расстояний использовалась в сети ARPANET вплоть до 1979 года, когда ее сменил алгоритм маршрутизации с учетом состояния линий. Отказаться от прежнего алгоритма пришлось по двум причинам. Во-первых, старый алгоритм при выборе пути не учитывал пропускную способность линий. Пока все линии имели пропускную способность 56 Кбит/с, в учете пропускной способности не было необходимости. Однако стали появляться линии со скоростью 230 Кбит/с, а затем и 1,544 Мбит/с, и не принимать во внимание пропускную способность стало невозможно. Конечно, можно было ввести пропускную способность в качестве множителя для единицы измерения, но имелась еще и другая проблема, заключавшаяся в том, что алгоритм слишком долго приходил к устойчивому состоянию (проблема счета до бесконечности). Поэтому он был заменен полностью новым алгоритмом, который сейчас называется **маршрутизацией с учетом состояния линий**. Варианты этого алгоритма широко применяются в наши дни.

В основе алгоритма лежит простая идея, ее можно изложить в *пяти требованиях к маршрутизатору*. Каждый маршрутизатор должен:

1. Обнаруживать своих соседей и узнавать их сетевые адреса.
2. Измерять задержку или стоимость связи с каждым из своих соседей.
3. Создавать пакет, содержащий всю собранную информацию.
4. Посылать этот пакет всем маршрутизаторам.
5. Вычислять кратчайший путь ко всем маршрутизаторам.

В результате каждому маршрутизатору высылаются полная топология и все измеренные значения задержек. После этого для обнаружения кратчайшего пути к каждому маршрутизатору может применяться алгоритм Дейкстры. Далее мы рассмотрим каждый из этих пяти этапов более подробно.

Знакомство с соседями

Когда маршрутизатор загружается, его первая задача состоит в получении информации о своих соседях. Он достигает этой цели, посылая специальный пакет HELLO по всем двухточечным линиям. При этом маршрутизатор на другом конце линии должен послать ответ, сообщая информацию о себе. Имена маршрутизаторов должны быть совершенно уникальными, поскольку, если удаленный маршрутизатор слышит, что три маршрутизатора являются соседями маршрутизатора *F*, не должно возникать разночтений по поводу того, один и тот же маршрутизатор *F* имеется в виду или нет.

Когда два или более маршрутизаторов объединены в локальную сеть, ситуация несколько усложняется. На рис. 5.9, *а* изображена ЛВС, к которой напрямую подключены три маршрутизатора — *A*, *C* и *F*. Каждый из них, как показано на рисунке, соединен также с одним или несколькими дополнительными маршрутизаторами.

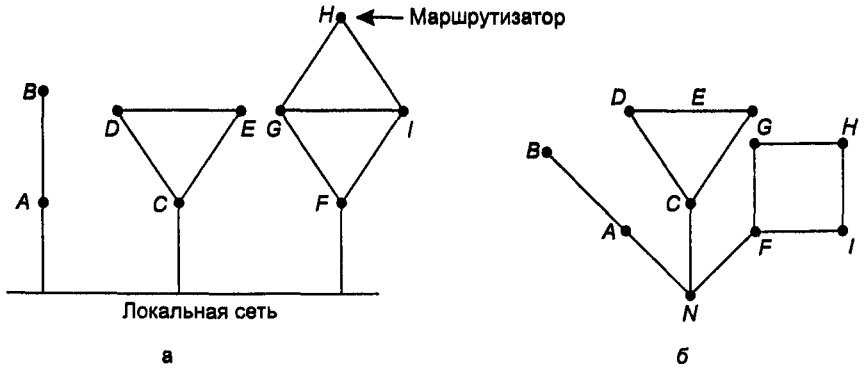


Рис. 5.9. Девять маршрутизаторов и локальная сать (*а*); графовая модель той же системы (*б*)

Один из способов моделирования локальной сети состоит в том, что ЛВС рассматривается в виде узла графа, как и маршрутизаторы. Это показано на рис. 5.9, *б*. На рисунке сеть изображена в виде искусственного узла *N*, с которым соединены маршрутизаторы *A*, *C* и *F*. Возможность передачи пакетов от *A* к *C* по локальной сети отражается здесь наличием пути *ANC*.

Измерение стоимости линии

Алгоритм маршрутизации с учетом состояния линии требует от каждого маршрутизатора знания или хотя бы обоснованной оценки задержки для всех линий связи со своими соседями. Наиболее прямой способ определить эту задержку заключается в посылке по линии специального пакета ECHO, на который другая сто-

рона обязана немедленно ответить. Измерив время двойного оборота этого пакета и разделив его на два, отправитель получает приемлемую оценку задержки. Чтобы получить более точный результат, это действие можно повторить несколько раз, после чего вычислить среднее арифметическое. Конечно, такой метод предполагает, что задержки являются симметричными, что не всегда так.

Возникает интересный вопрос: надо ли учитывать нагрузку на линию во время измерения задержки? Чтобы учесть загруженность линии, таймер должен включаться при отправке пакета ECHO. Чтобы игнорировать загрузку, таймер следует включать, когда пакет ECHO достигает начала очереди.

Оба способа могут быть аргументированы. Учет трафика в линии при измерении задержки означает, что когда у маршрутизатора есть выбор между двумя линиями с одинаковой пропускной способностью, маршрут по менее загруженной линии рассматривается как более короткий. Такой выбор приведет к более сбалансированному использованию линий связи и, следовательно, к более эффективной работе системы.

К сожалению, можно привести аргумент и против учета загруженности линии при расчете задержек. Рассмотрим подсеть, показанную на рис. 5.10. Она разделена на две части — восточную и западную, которые соединены двумя линиями, *CF* и *EI*.

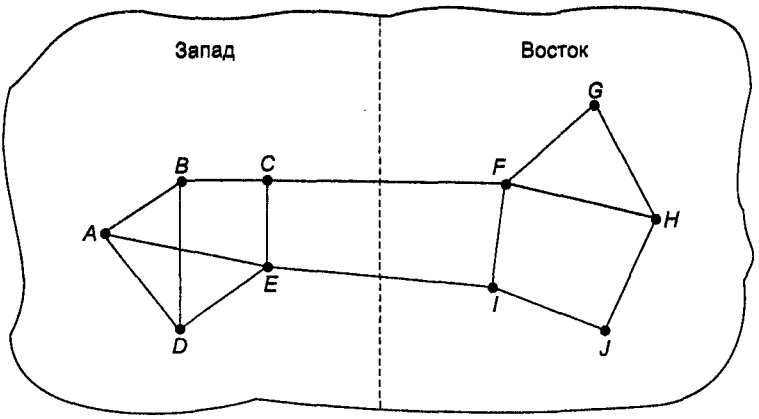


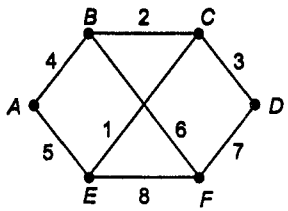
Рис. 5.10. Подсеть, в которой восточная и западная части соединены двумя линиями

Предположим, что основная часть потока данных между востоком и западом использует линию *CF*. В результате эта линия оказывается сильно загруженной и с большими задержками. Учет времени стояния пакета в очередях при подсчете кратчайшего пути сделает линию *EI* более привлекательной. После установки новых таблиц маршрутизации большая часть потока данных между востоком и западом переместится на линию *EI*, и ситуация повторится с точностью до смены одной линии на другую. Аналогично, после еще одного обновления уже линия *CF* окажется более привлекательной. В результате таблицы маршрутизации будут страдать от незатухающих колебаний, что сильно снизит эффективность

работы системы. Если же нагрузку не учитывать, то эта проблема не возникнет. Можно поступать по-другому: распределять нагрузку между двумя линиями. Однако такое решение приведет к неполному использованию наилучшего пути. Тем не менее, во избежание колебаний системы при выборе оптимального пути, по-видимому, лучше всего распределять нагрузку между несколькими линиями, пуская определенные части трафика по каждой из них.

Создание пакетов состояния линий

После того как информация, необходимая для обмена, собрана, следующий шаг, выполняемый каждым маршрутизатором, заключается в создании пакета, содержащего все эти данные. Пакет начинается с идентификатора отправителя, за которым следует порядковый номер и возраст (описываемый далее), а также список соседей. Для каждого соседа указывается соответствующая ему задержка. Пример подсети приведен на рис. 5.11, *а*, на котором показаны задержки для каждой линии. Соответствующие пакеты состояния линий для всех шести маршрутизаторов показаны на рис. 5.11, *б*.



а

Пакеты состояния линий

A	
Порядковый номер	
Возраст	
B	4
E	5

B	
Порядковый номер	
Возрест	
A	4
C	2
F	6

C	
Порядковый номер	
Возраст	
B	2
D	3
E	1

D	
Порядковый номер	
Возрест	
C	3
F	7

E	
Порядковый номер	
Возраст	
A	5
C	1
F	8

F	
Порядковый номер	
Возрвст	
B	6
D	7
E	6

б

Рис. 5.11. Подсеть (а); пакеты состояния линий для этой подсети (*б*)

Создаются пакеты состояния линий несложно. Самое трудное заключается в выборе момента времени для их создания. Их можно создавать периодически через равные интервалы времени. Другой вариант состоит в создании пакетов, когда происходит какое-нибудь значительное событие — например, линия или сосед выходит из строя или, наоборот, снова появляется в сети либо существенно изменяет свои свойства.

Распространение пакетов состояния линий

Самая сложная часть алгоритма заключается в распространении пакетов состояния линий. По мере распространения и установки пакетов маршрутизаторы, получившие первые пакеты, начинают изменять свои маршруты. Соответственно разные маршрутизаторы будут пользоваться разными версиями топологии, что может привести к противоречиям, появлению в маршрутах петель, недоступных машин, а также к другим проблемам.

Сначала мы опишем основной алгоритм распространения. Затем расскажем о некоторых улучшениях. Основная идея алгоритма распространения пакетов состояния линии состоит в использовании алгоритма заливки. Чтобы держать этот процесс под контролем, в каждый пакет помещают порядковый номер, увеличивающийся на единицу для каждого следующего пакета. Маршрутизаторы записывают все пары (источник, порядковый номер), которые им попадаются. Когда приходит новый пакет состояния линий, маршрутизатор ищет адрес его отправителя и порядковый номер пакета в своем списке. Если это новый пакет, он рассылается дальше по всем линиям, кроме той, по которой он пришел. Если же это дубликат, он удаляется. Если порядковый номер прибывшего пакета меньше, чем номер уже полученного пакета от того же отправителя, то такой пакет также удаляется как устаревший, поскольку очевидно, что у маршрутизатора есть более свежие данные.

С этим алгоритмом связано несколько проблем, но с ними можно справиться. Во-первых, если последовательный номер, достигнув максимально возможного значения, обнулится, возникнет путаница. Решение состоит в использовании 32-разрядных порядковых номеров. Даже если рассылать каждую секунду по пакету, то для переполнения 4-байтового целого числа понадобится 137 лет.

Во-вторых, если маршрутизатор выйдет из строя, будет потерян его порядковый номер. Если он будет снова загружен с нулевым порядковым номером, его пакеты будут игнорироваться как устаревшие.

В-третьих, может произойти искажение порядкового номера — например, вместо номера 4 будет принято число 65 540 (ошибка в 1-м бите); в этом случае пакеты с 5-го по 65 540-й будут игнорироваться некоторыми маршрутизаторами как устаревшие.

Решение этих проблем заключается в помещении в пакет после его порядкового номера возраста пакета и уменьшении его на единицу каждую секунду. Когда возраст уменьшается до нуля, информация от этого маршрутизатора удаляется. В нормальной ситуации новый пакет приходит, например, каждые 10 секунд; таким образом, сведения о маршрутизаторе устаревают, только когда маршрутизатор выключен (или в случае потери шести пакетов подряд, что маловероятно). Поле возраста также уменьшается на единицу каждым маршрутизатором во время начального процесса заливки, чтобы гарантировать, что ни один пакет не потеряется и не будет жить вечно.

Для повышения надежности этого алгоритма используются некоторые усовершенствования. Когда пакет состояния линий приходит на маршрутизатор для заливки, он не ставится сразу в очередь на отправку. Вместо этого он сохраняет-

ся в течение некоторого периода времени в области промежуточного хранения. Если за это время от того же отправителя успевает прийти еще один пакет, маршрутизатор сравнивает их порядковые номера. Более старый пакет удаляется. Если номера одинаковые, то удаляется дубликат. Для защиты от ошибок на линиях связи между маршрутизаторами получение всех пакетов состояния линий подтверждается. Когда линия освобождается, маршрутизатор сканирует область промежуточного хранения, из которой выбираются для передачи пакеты или подтверждения.

Структура данных, используемая маршрутизатором *B* для работы с подсетью, изображенной на рис. 5.11, *а*, показана на рис. 5.12. Каждый ряд здесь соответствует недавно полученному, но еще не полностью обработанному пакету состояния линий. В таблице записываются адрес отправителя, порядковый номер, возраст и данные. Кроме того, в таблице содержатся флаги рассылки и подтверждений для каждой из трех линий маршрутизатора *B* (к *A*, *C* и *F* соответственно). Флаги отсылки означают, что этот пакет следует отослать по соответствующей линии. Флаги подтверждений означают, что нужно подтвердить получение этого пакета по данной линии.

			Флаги отсылки			Флаги подтверждения			
Источник	Порядковый номер	Возраст	*A*	*C*	*F*	*A*	*C*	*F*	Данные
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Рис. 5.12. Буфер пакетов маршрутизатора *B* с рисунка 5.11

Как видно из рис. 5.12, пакет состояния линий от маршрутизатора *A* пришел напрямую, поэтому он должен быть отправлен маршрутизаторам *C* и *F*, а подтверждение о его получении следует направить маршрутизатору *A*, что и показывают флаговые биты. Аналогично, пакет от *F* следует переслать маршрутизаторам *A* и *C*, а *F* отослать подтверждение.

Однако ситуация с третьим пакетом, полученным от маршрутизатора *E*, отличается. Он был получен дважды, по линиям *EAB* и *EFB*. Следовательно, его нужно отослать только *C*, но подтверждения выслать и *A*, и *F*, как указывают биты.

Если в то время, когда оригинал еще находится в буфере, прибывает дубликат пакета, значение битов должно быть изменено. Например, если копия состояния маршрутизатора *C* прибудет от *F* прежде, чем четвертая строка таблицы будет разослана, шесть флаговых битов примут значение 100011, и это будет означать, что следует подтвердить получение пакета от *F*, но не пересылать его *F*.

Вычисление новых маршрутов

Собрав полный комплект пакетов состояния линий, маршрутизатор может построить полный граф подсети, так как он располагает данными обо всех линиях. На самом деле, каждая линия представлена даже дважды, по одному значению для каждого направления. Эти два значения могут усредняться или использоваться по отдельности.

Теперь для построения кратчайшего пути ко всем возможным адресатам может быть локально применен алгоритм Дейкстры. Результат вычислений может быть установлен в таблицах маршрутов, после чего можно возобновить нормальную работу маршрутизатора.

В подсети, состоящей из n маршрутизаторов, у каждого из которых k соседей, количество памяти, необходимой для хранения входной информации, пропорционально kn. Кроме того, может потребоваться много времени на обработку информации. В больших подсетях это может составлять проблему. Тем не менее, во многих практических ситуациях маршрутизация с учетом состояния линий работает вполне удовлетворительно.

Однако неисправности оборудования или программного обеспечения могут привести к очень серьезным проблемам при использовании данного алгоритма (а также других алгоритмов). Например, если маршрутизатор заявит о существовании линии, которой у него в действительности нет, или наоборот, забудет о существовании имеющейся у него линии, граф подсети окажется неверным. Если маршрутизатор не сможет переслать пакеты или повредит их при пересылке, также возникнет проблема. Наконец, если у маршрутизатора закончится свободная память или он ошибется в расчетах маршрутов, также возможны различные неприятности. При увеличении размера подсети до нескольких десятков или сотен тысяч маршрутизаторов вероятность выхода из строя одного из них перестает быть пренебрежимо малой. Все, что можно здесь сделать, — это попытаться ограничить вред, наносимый практически неизбежным выходом из строя оборудования. Эти проблемы и методы их разрешения подробно обсуждаются в (Perlman, 1988).

Маршрутизация с учетом состояния линий широко применяется в современных сетях, поэтому следует сказать несколько слов о некоторых примерах протоколов, использующих данный алгоритм. Одним из таких протоколов является протокол OSPF, все чаще применяемый в Интернете, о котором будет рассказано в разделе «Протокол внутреннего шлюза OSPF».

Другим важным протоколом с учетом состояния линий является **IS-IS** (Intermediate System to Intermediate System — связь между промежуточными системами) — протокол, разработанный для сети DECnet и принятый впоследствии Международной организацией по стандартизации ISO для использования вместе с протоколом сетевого уровня CLNP, не требующим соединений. С тех пор он был модифицирован для поддержки также и других протоколов, в частности IP. Протокол IS-IS используется в некоторых магистралях сети Интернет (включая старую магистраль NSFNET) и в некоторых цифровых сотовых системах, например, в CDPD. В сети Novell NetWare применяется разновидность протокола IS-IS (NLSP) для маршрутизации IPX-пакетов.

В основе работы протокола IS-IS лежит распространение картины топологии маршрутизаторов, по которой рассчитываются кратчайшие пути. Каждый маршрутизатор сообщает в информации о состоянии линий доступные ему напрямую адреса сетевого уровня. Эти адреса могут быть адресами IP, IPX, AppleTalk или другими. Протокол IS-IS может даже осуществлять одновременную поддержку нескольких протоколов сетевого уровня.

Многие новшества, разработанные для протокола IS-IS, были приняты несколько лет спустя при разработке протокола OSPF. К ним относятся метод саморегуляции лавинного потока обновлений информации о состоянии линий, концепция выделенного маршрутизатора в локальной сети, а также метод вычисления и поддержки расщепления пути и умножения метрик. Соответственно, между протоколами IS-IS и OSPF нет почти никакой разницы. Наиболее существенное различие между ними заключается в том, что способ кодирования в протоколе IS-IS, в отличие от OSPF, облегчает одновременную поддержку нескольких сетевых протоколов. Это свойство особенно важно в больших многопротокольных средах.

Иерархическая маршрутизация

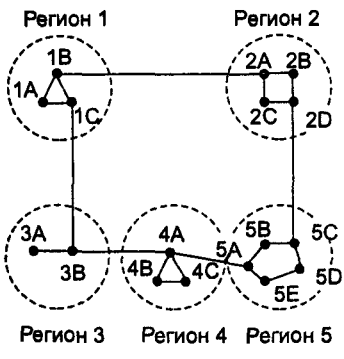
Размер таблиц маршрутов, поддерживаемых маршрутизаторами, увеличивается пропорционально увеличению размеров сети. При этом требуется не только большее количество памяти для хранения этой таблицы, но и большее время центрального процессора для ее обработки. Кроме того, возрастает размер служебных пакетов, которыми обмениваются маршрутизаторы, что увеличивает нагрузку на линии. В определенный момент сеть может вырасти до таких размеров, при которых перестанет быть возможным хранение на маршрутизаторах записи обо всех остальных маршрутизаторах. Поэтому в больших сетях маршрутизация должна осуществляться иерархически, как это делается в телефонных сетях.

При использовании иерархической маршрутизации маршрутизаторы разбиваются на отдельные так называемые **регионы**. Каждый маршрутизатор знает все детали выбора маршрутов в пределах своей области, но ему ничего не известно о внутреннем строении других регионов. При объединении нескольких сетей естественно рассматривать их как отдельные регионы, при этом маршрутизаторы одной сети освобождаются от необходимости знать топологию других сетей.

В очень больших сетях двухуровневой иерархии может оказаться недостаточно. Может потребоваться группировать регионы в кластеры, кластеры в зоны, зоны в группы, и т. д., пока у нас не иссякнет фантазия на названия для новых образований. В качестве примера многоуровневой иерархии рассмотрим маршрутизацию пакета, пересылаемого из университета Беркли (Berkeley), штат Калифорния, в Малинди (Malindi) в Кении. Маршрутизатор в Беркли знает детали топологии в пределах Калифорнии, но трафик, направляющийся за пределы штата, он посылает маршрутизатору в Лос-Анджелесе. Маршрутизатор в Лос-Анджелесе может выбрать маршрут для трафика в пределах США, но все пакеты, направляемые за рубеж, переправляет в Нью-Йорк. Нью-йоркский маршрутизатор направит трафик на маршрутизатор страны назначения, ответственный за прием

трафика из-за границы. Он может располагаться, например, в Найроби. Наконец, направляясь вниз по дереву иерархии уже в пределах Кении, пакет попадет в Малинди.

На рис. 5.13 приведен количественный пример маршрутизации в двухуровневой иерархии с пятью регионами. Полная таблица маршрутизатора *1A*, как показано на рис. 5.13, *б*, состоит из 17 записей. При использовании иерархической маршрутизации, как показано на рис. 5.13, *в*, таблица, как и прежде, содержит сведения обо всех локальных маршрутизаторах, но записи обо всех остальных регионах концентрируются в пределах одного маршрутизатора, поэтому трафик во второй регион по-прежнему пойдет по линии *1B — 2A*, а во все остальные регионы — по линии *1C — 3B*. При иерархической маршрутизации размер таблицы маршрутов уменьшается с 17 до 7 строк. Чем крупнее выбираются регионы, тем больше экономится места в таблице.



а

Полная таблица для 1A

Назначение	Линия	Транзитные участки
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

б

Иерархическая таблица для 1A

Назначение	Линия	Транзитные участки
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	3
4	1C	3
5	1C	4

в

Рис. 5.13. Иерархическая маршрутизация

К сожалению, этот выигрыш памяти не достается бесплатно. Платой за уменьшение размеров таблицы маршрутов является увеличение длины пути. Например, наилучший маршрут от *1A* до *5C* проходит через регион 2, однако при использовании иерархической маршрутизации весь трафик в регион 5 направляется через регион 3, поскольку так лучше для большинства адресатов в регионе 5.

Когда единая сеть становится очень большой, возникает интересный вопрос: сколько уровней должна иметь иерархия? Для примера рассмотрим подсеть с 720 маршрутизаторами. Если иерархии нет, то каждому маршрутизатору необхо-

димо поддерживать таблицу из 720 строк. Если подсеть разбить на 24 региона по 30 маршрутизаторов в каждом регионе, тогда каждому маршрутизатору потребуется 30 локальных записей плюс 23 записи об удаленных регионах, итого 53 записи. При выборе трехуровневой иерархии, состоящей из 8 кластеров по 9 регионов из 10 маршрутизаторов, каждому маршрутизатору понадобится 10 строк в таблице для локальных маршрутизаторов, 8 строк для маршрутизации в другие регионы в пределах своего кластера, плюс 7 строк для удаленных кластеров, итого 25 строк. Камоун (Kamoun) и Кляйнрок (Kleinrock) в 1979 году показали, что оптимальное количество уровней иерархии для подсети, состоящей из N маршрутизаторов, равно $\ln N$. При этом потребуется $e \ln N$ записей для каждого маршрутизатора. Они также показали, что увеличение длины эффективного среднего пути, вызываемое иерархической маршрутизацией, довольно мало и обычно является приемлемым.

Широковещательная маршрутизация

В некоторых приложениях хостам требуется посылать сообщения на множество хостов или даже на все сразу. Можно привести такие примеры, как распространение прогнозов погоды, обновление биржевых курсов ценных бумаг, радиопрограммы в прямом эфире. Эффективнее всего распространять соответствующие данные широковещательным способом, предоставляя возможность всем заинтересованным хостам получить их. Итак, **широковещанием** называется рассылка пакетов по всем пунктам назначения. Для ее реализации применяются разнообразные методы.

Один из методов широковещательной маршрутизации не требует никаких особых способностей от подсети и используется просто для того, чтобы рассылать отдельные пакеты по всем направлениям. Он не только отнимает у подсети пропускную способность, но и требует, чтобы у источника пакета был полный список всех хостов. На практике это может быть единственная возможность, но такой метод является наименее желательным.

Еще одним очевидным кандидатом является метод *заливки*. Хотя он плохо подходит для обычных двухточечных соединений, для широковещания это может быть серьезный претендент, особенно если нет возможности применить один из методов, описываемых ниже. Проблема с применением заливки в качестве метода широковещания такая же, как с двухточечным алгоритмом маршрутизации: при заливке генерируется очень много пакетов и отнимается весьма существенная часть пропускной способности.

Третий алгоритм называется **многоадресной маршрутизацией**. При использовании этого метода в каждом пакете содержится либо список адресатов, либо битовая карта, показывающая предпочитаемые хосты назначения. Когда такой пакет прибывает на маршрутизатор, последний проверяет список, содержащийся в пакете, определяя набор выходных линий, которые потребуются для дальнейшей рассылки. (Линия может потребоваться в том случае, если она входит в оптимальный путь к какому-то из адресатов списка.) Маршрутизатором создается копия пакета для каждой из используемых исходящих линий. В нее включаются

только те адресаты, для доступа к которым требуется данная линия. Таким образом, весь список рассылки распределяется между исходящими линиями. После определенного числа пересылок каждый из пакетов будет содержать только один адрес назначения и будет выглядеть как обычный пакет. Многоадресная маршрутизация подобна индивидуально адресуемым пакетам с той разницей, что в первом случае из нескольких пакетов, следующих по одному и тому же маршруту, только один «платит полную стоимость», а остальные едут бесплатно.

Еще один, четвертый алгоритм широковещательной маршрутизации в явном виде использует корневое дерево или любое другое связующее дерево. **Связующее дерево** представляет собой подмножество подсети, включающее в себя все маршрутизаторы, но не содержащее замкнутых путей. Если каждый маршрутизатор знает, какие из его линий принадлежат связующему дереву, он может отправить приходящий пакет по всем линиям связующего дерева, кроме той, по которой пакет прибыл. Такой метод оптимальным образом использует пропускную способность сети, порождая минимальное количество пакетов, требующихся для выполнения работы. Единственной проблемой этого метода является то, что каждому маршрутизатору необходимо обладать информацией о связующем дереве. Иногда такая информация доступна (например, в случае маршрутизации с учетом состояния линий), но иногда — нет (при маршрутизации по векторам расстояний).

Последний алгоритм широковещания, который мы рассмотрим, представляет собой попытку приблизиться к поведению предыдущего алгоритма, даже когда маршрутизаторы ничего не знают о связующих деревьях. Лежащая в основе данного алгоритма идея, называющаяся **продвижением по встречному пути**, замечательно проста. Когда прибывает широковещательный пакет, маршрутизатор проверяет, используется ли та линия, по которой он прибыл, для нормальной передачи пакетов *источнику широковещания*. В случае положительного ответа велика вероятность того, что широковещательный пакет прибыл по наилучшему маршруту и является, таким образом, первой копией, прибывшей на маршрутизатор. Тогда маршрутизатор рассылает этот пакет по всем линиям, кроме той, по которой он прибыл. Однако если пакет прибывает от того же источника по другой линии, он отвергается как вероятный дубликат.

Пример работы алгоритма продвижения по встречному пути показан на рис. 5.14. Слева изображена подсеть, посередине —входное дерево для маршрутизатора *I* этой подсети. На первом транзитном участке маршрутизатор *I* посылает пакеты маршрутизаторам *F*, *H*, *J* и *N*, являющимся вторым ярусом дерева. Все эти пакеты прибывают к *I* по предпочитаемым линиям (по пути, совпадающему с входным деревом), что обозначается кружками вокруг символов на рис. 5.14, *в*. На втором этапе пересылки формируются восемь пакетов — по два каждым маршрутизатором, получившим пакет после первой пересылки. Все восемь пакетов попадают к маршрутизаторам, не получавшим ранее пакетов, а пять из них приходят по предпочитаемым линиям. Из шести пакетов, формируемых на третьем транзитном участке, только три прибывают по предпочитаемым линиям (на маршрутизаторы *C*, *E* и *K*). Остальные оказываются дубликатами. После пяти транзитных участков широковещание заканчивается с общим количест-

вом переданных пакетов, равным 23, тогда как при использовании входного дерева потребовалось бы 4 транзитных участка и 14 пакетов.

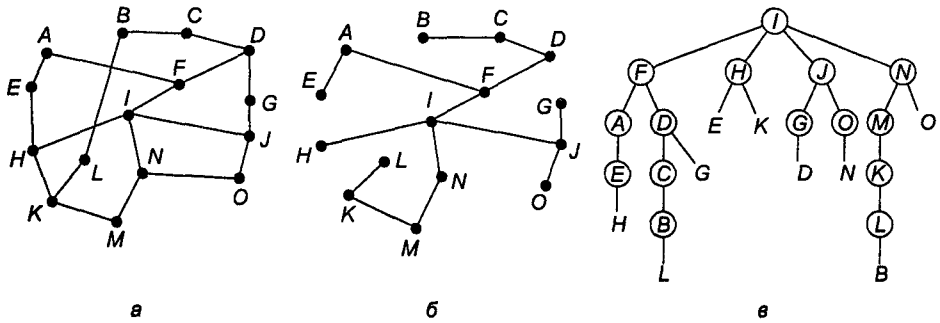


Рис. 5.14. Продвижение по встречному пути: подсеть (*а*); связующее дерево (*б*); дерево, построенное методом продвижения по встречному пути (*в*)

Принципиальное преимущество метода продвижения по встречному пути заключается в его вполне приемлемой эффективности при простоте реализации. Для использования этого метода маршрутизаторам не нужна никакая дополнительная информация о связующих деревьях. Не требуются и дополнительные расходы на список получателей или бит-карту в каждом распространяемом пакете, как в случае многоадресной рассылки. Также не требуется никакого специального механизма для прекращения процесса, как, например, в методе заливки (либо счетчик транзитных участков в каждом пакете и априорные сведения о диаметре сети, либо список уже встречавшихся пакетов от каждого источника).

Многоадресная рассылка

В некоторых приложениях сильно разделенные процессы работают совместными группами. Например, в виде группы процессов может быть реализована распределенная база данных. Часто одному процессу бывает необходимо послать сообщение всем остальным членам группы. Если группа невелика, то можно просто послать каждому члену группы отдельное сообщение. Если же группа достаточно большая, такая стратегия окажется весьма дорогостоящей. Иногда может быть использовано широковещание, но применять его для информирования 1000 машин в сети, состоящей из миллиона узлов, неэффективно, поскольку большинство получателей будут не заинтересованы в данном сообщении (или, что еще хуже, явно заинтересованы, но было бы крайне желательно от них эту информацию скрыть). Таким образом, требуется способ рассылки сообщений строго определенным группам, довольно большим по численности, но небольшим по сравнению со всей сетью.

Передача сообщения членам такой группы называется **многоадресной рассылкой**, а алгоритм маршрутизации этой операции — **многоадресной маршрутизацией**. В этом разделе будет описан один из способов реализации многоадресной маршрутизации. Дополнительные сведения см. в (Chu и др., 2000; Costa и др., 2001; Kasera и др., 2000; Madruga and Garcia-Luna-Aceves, 2001; Zhang and Ryu, 2001).

Многоадресной рассылке требуется управление группами, то есть способ создания и удаления групп, присоединения процесса к группе и ухода процесса из группы. Реализация данных задач, однако, не интересует алгоритм маршрутизации. Зато он заинтересован в том, чтобы процесс информировал свой хост о присоединении к какой-нибудь группе. Важно, чтобы маршрутизаторы знали, какой хост к какой группе принадлежит. Для этого либо хост должен сообщать своим маршрутизаторам об изменении в составе групп, либо маршрутизаторы должны сами периодически опрашивать свои хосты. В любом случае маршрутизаторы узнают, какие из их хостов к каким группам принадлежат. Маршрутизаторы сообщают об этом своим соседям, и таким образом эта информация распространяется по всей подсети.

Для многоадресной рассылки каждый маршрутизатор рассчитывает связующее дерево, покрывающее все остальные маршрутизаторы подсети. Например, на рис. 5.15, *а* мы видим подсеть с двумя группами, 1 и 2. Как показано на рисунке, маршрутизаторы соединены с хостами, принадлежащими к одной или обеим группам. Связующее дерево для самого левого маршрутизатора показано на рис. 5.15, *б*.

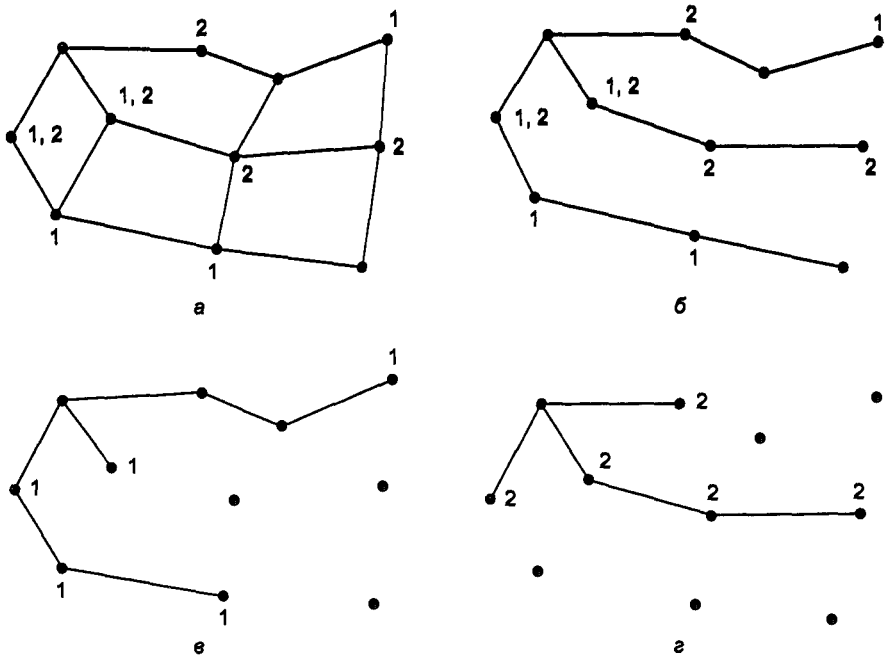


Рис. 5.15. Подсеть (*а*); связующее дерево для самого левого маршрутизатора (*б*); многоадресное дерево для группы 1 (*в*); многоадресное дереао для группы 2 (*г*)

Когда процесс посылает группе многоадресный пакет, первый маршрутизатор изучает свое связующее дерево и отсекает у него линии, не ведущие к хостам, являющимся членами группы. В нашем примере на рис. 5.15, *в* изображено усеченное связующее дерево для группы 1. Аналогично, на рис. 5.15, *г* показано усеченное связующее дерево для группы 2. Многоадресные пакеты рассылаются только вдоль соответствующего их группе усеченного связующего дерева.

Существует несколько способов усечения связующего дерева. Простейший способ может применяться при использовании маршрутизации с учетом состояния линий, когда каждому маршрутизатору известна полная топология подсети, в том числе и состав групп. При этом из связующего дерева могут быть удалены маршрутизаторы, не принадлежащие к данной группе, начиная с конца каждого пути вплоть до корня дерева.

При маршрутизации по векторам расстояний может быть применена другая стратегия усечения дерева. Для многоадресной рассылки здесь применяется алгоритм продвижения по встречному пути. Когда многоадресное сообщение получает маршрутизатор, у которого нет хостов, входящих в группу, и линий связи с другими маршрутизаторами, он может ответить сообщением PRUNE (отсечь), информируя отправителя, что сообщения для данной группы ему больше посылать не нужно. Такой же ответ может дать маршрутизатор, у которого нет хостов, входящих в группу, если он получил многоадресное сообщение по всем своим линиям. В результате подсеть постепенно рекурсивно усекается.

Недостаток данного алгоритма заключается в его плохой применимости к большим сетям. Предположим, что в сети есть n групп, каждая из которых в среднем состоит из m членов. Для каждой группы должно храниться m усеченных входных деревьев, то есть mn деревьев для всей сети. При большом количестве групп для хранения всех деревьев потребуется много памяти.

Альтернативный метод использует **деревья с основанием в сердцевине** (Ballardie и др., 1993). В этом методе для каждой группы рассчитывается единое связующее дерево с корнем (ядром) около середины группы. Хост посылает многоадресное сообщение ядру группы, откуда оно уже рассылается по всему связующему дереву группы. Хотя это дерево не является оптимальным для всех источников, единое дерево для группы снижает затраты на хранение информации о нем в m раз.

Алгоритмы маршрутизации для мобильных хостов

Сегодня миллионы людей обладают переносными компьютерами, и большинство из них желает читать свою электронную почту и получать доступ к нормальным файловым системам, находясь при этом в любой точке земного шара. Мобильные хосты привносят новое усложнение в и без того непростое дело выбора маршрутов в различных вычислительных сетях — чтобы направить пакет к мобильному хосту, его нужно сначала найти. Вопрос включения мобильных хостов в сети появился не так давно, но в данном разделе мы рассмотрим некоторые проблемы и приведем их возможные решения.

Модель мира, обычно используемая разработчиками сетей, показана на рис. 5.16. Здесь мы видим глобальную сеть, состоящую из маршрутизаторов и хостов. С глобальной сетью соединены локальные и региональные сети и беспроводные соты, которые рассматривались в главе 2.

Хосты, которые никогда не перемещаются, называются стационарными. Они соединены с сетью медными поводами или оптическими кабелями. Мы же будем различать две другие категории хостов. Мигрирующие хосты являются, в основном, стационарными пользователями, но время от времени перемещаются с од-

ного фиксированного места на другое и пользуются сетью только тогда, когда физически соединены с нею. Блуждающие хосты используют переносные компьютеры, и им требуется связь с сетью прямо во время перемещения в пространстве. Для обозначения этих двух категорий, то есть хостов, которые не имеют постоянного местоположения и тем не менее желают быть на связи, мы будем использовать термин **мобильные хосты**.

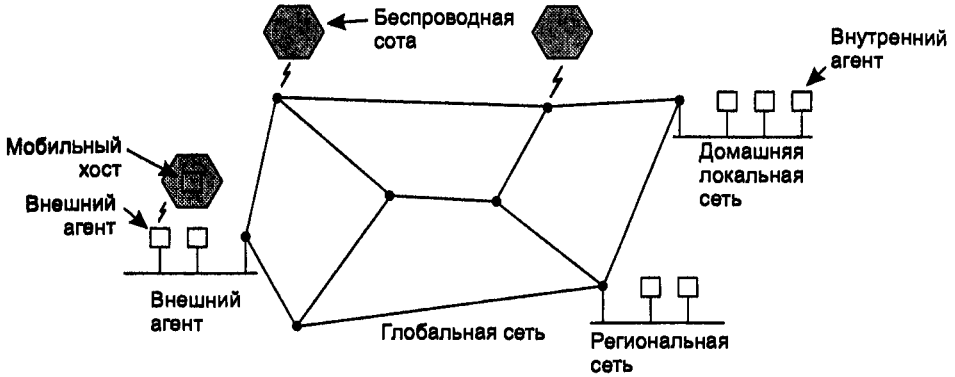


Рис. 5.16. Глобальная сать, с которой соединены локальные и региональные сети, а также беспроводные соты

Предполагается, что у всех хостов есть постоянное домашнее **местоположение**, которое никогда не меняется. Кроме того, у хостов есть постоянный домашний адрес, которым можно воспользоваться для определения домашнего местоположения, аналогично тому, как телефонный номер 1-212-5551212 обозначает США (страна с кодом 1) и Манхэттен (212). Целью маршрутизации в системах с мобильными хостами является обеспечение возможности передачи пакетов мобильным пользователям с помощью их домашних адресов. При этом пакеты должны эффективно достигать пользователей, независимо от их расположения. Самое сложное здесь, конечно, — найти пользователя.

В модели, показанной на рис. 5.16, мир разделен на небольшие единицы. Мы будем называть их областями, что обычно будет означать локальную сеть или беспроводную соту. Каждая область может содержать одного или более **внешних агентов**, следящих за всеми мобильными пользователями, посещающими область. Кроме того, в каждой области имеется **внутренний агент**, следящий за временно покинувшими свою область пользователями.

Когда в области появляется новый пользователь — либо подключившийся к ней (соединив свой компьютер с сетью), либо просто переместившийся в соту, — его компьютер должен зарегистрироваться в данной области, связавшись с местным внешним агентом. Процедура регистрации обычно выглядит следующим образом:

1. Периодически каждый внешний агент рассылает пакет, объявляя таким образом о своем существовании и местонахождении. Вновь прибывший мобильный хост может ждать подобного сообщения, но может и сам, не дождавшись его, передать пакет с запросом о наличии внешнего агента в данной области.

2. Мобильный хост регистрируется в данной области, сообщая внешнему агенту свой домашний адрес, текущий адрес уровня передачи данных, а также информацию, подтверждающую его подлинность.
3. Внешний агент связывается с внутренним агентом мобильного пользователя и сообщает ему: «Один из ваших хостов находится в нашей области». Это сообщение содержит адрес сети внешнего агента, а также информацию, подтверждающую подлинность мобильного хоста. Это позволяет убедить внутреннего агента в том, что мобильный хост действительно находится здесь.
4. Внутренний агент проверяет переданный ему идентификатор безопасности мобильного хоста, содержащий временной штамп, доказывающий, что идентификатор был создан буквально несколько секунд назад. Если проверка подлинности хоста проходит успешно, внутренний агент разрешает внешнему агенту продолжать связь.
5. Получив подтверждение от внутреннего агента, внешний агент заносит сведения о мобильном хосте в свою таблицу и сообщает ему, что он зарегистрирован.

В идеальном случае, покидая область, пользователь также должен сообщить об этом внешнему агенту, однако на практике многие пользователи, закончив свои дела, просто выключают свои компьютеры.

Когда пакет посылается мобильному пользователю, он направляется в его домашнюю локальную сеть на домашний адрес пользователя. На рис. 5.17 это действие показано как этап 1. Здесь отправитель из северо-западного города Сиэтла хочет отправить пакет хосту, который обычно находится по другую сторону США, в Нью-Йорке. Пакеты, посланные в домашнюю локальную сеть мобильного хоста (Нью-Йорк), перехватываются внутренним агентом, который узнает новое (временное) расположение мобильной станции (например, Лос-Анджелес) и адрес внешнего агента локальной сети, в которой она в данный момент находится.

Затем внутренний агент выполняет два действия. Во-первых, он помещает пакет, предназначающийся мобильному пользователю, в поле данных внешнего пакета, который посылается внешнему агенту (этап 2 на рис. 5.17). Такой прием называется туннелированием. Позднее мы обсудим ее подробнее. Получив пакет, внешний агент извлекает из поля данных оригинальный пакет, который пересылает мобильному пользователю в виде кадра уровня передачи данных.

Затем внутренний агент сообщает отправителю, что в дальнейшем следует не посылать пакеты мобильному хосту на домашний адрес, а вкладывать их в поле данных пакетов, явно адресованных внешнему агенту (этап 3 на рис. 5.17). Последующие пакеты теперь могут направляться напрямую пользователю через внешнего агента (этап 4), полностью минуя домашний адрес мобильного пользователя.

Различные предложенные схемы маршрутизации отличаются в нескольких аспектах. Во-первых, они отличаются в том, какая часть протокола выполняется маршрутизаторами, а какая — хостами, а также каким уровнем протоколов хостов. Во-вторых, в некоторых схемах маршрутизаторы записывают преобразованные адреса, поэтому они могут перехватывать и переадресовывать пакеты даже еще до того, как они успевают дойти до домашнего адреса мобильного пользователя. В-третьих, в одних схемах каждому посетителю дается уникальный вре-

менный адрес, а в других схемах временный адрес ссылается на агента, обрабатывающего трафик для всех посетителей.

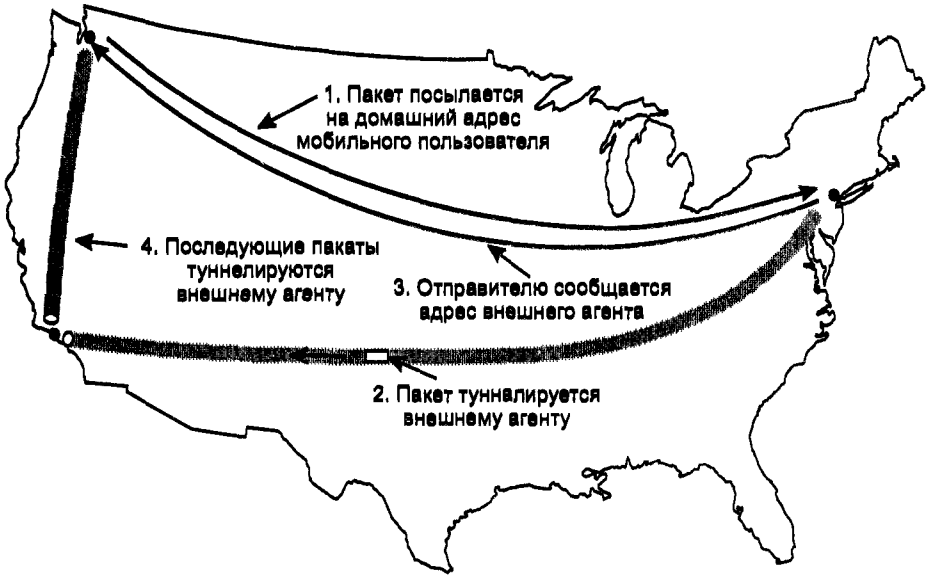


Рис. 5.17. Маршрутизация пакетов мобильным хостам

В-четвертых, схемы различаются способами переадресации пакетов. Один из способов заключается в изменении поля адреса получателя в пакете и передаче измененного пакета. Есть системы, в которых весь пакет, включая домашний адрес, может быть помещен внутрь другого пакета, посылаемого по временному адресу. В любом случае, когда хост или маршрутизатор получает сообщение вида «Начиная с этого момента, пожалуйста, пересылайте мне всю почту, адресуемую Стефани», у него могут возникнуть вопросы — например, с кем он разговаривал, соглашаться или нет на данное предложение. Несколько протоколов мобильных хостов обсуждаются и сравниваются в (Hac and Guo, 2000; Perkins, 1998a; Snoeren and Balakrishnah, 2000; Solomon, 1998; Wang and Chen, 2001).

Маршрутизация в специализированных сетях

Итак, мы рассмотрели, как производится маршрутизация в случаях, когда станции мобильны, а маршрутизаторы стационарны. Еще более занимательная ситуация возникает тогда, когда мобильны сами маршрутизаторы. Это возможно, например, в следующих случаях.

- Военная техника на поле боя при отсутствии инфраструктуры.
- Морская флотилия, находящаяся в плавании.
- Работники служб спасения в районах с разрушенной инфраструктурой.
- Собрание людей с портативными компьютерами при отсутствии в помещении сети 802.11.

Во всех подобных случаях каждый узел состоит из маршрутизатора и хоста одновременно, обычно они даже совмещены в пределах одного компьютера. Сети, состоящие из узлов, волею судеб оказавшихся недалеко друг от друга, называются **специализированными сетями**, или **мобильными специализированными сетями** (MANET, Mobile Ad hoc networks). Давайте их вкратце рассмотрим. Более подробную информацию можно найти в книге (Perkins, 2001).

Основное отличие специализированных сетей от обычных проводных сетей состоит в том, что все обычные законы, касающиеся фиксированной топологии, известных соседей, взаимосвязи между IP-адресом и расположением в специализированных сетях, перестают работать. Маршрутизаторы могут легко появляться в системе и так же легко из нее исчезать, появляясь в каком-то другом месте. В обычных сетях путь от маршрутизатора к какому-либо адресату продолжает оставаться реализуемым до тех пор, пока не произойдет какой-нибудь сбой системы. В специализированных сетях топология постоянно меняется, а с ней меняется и предпочтительность (и даже реализуемость) путей. Причем, это происходит спонтанно, безо всяких предупреждений. Надо ли говорить о том, что в таких условиях маршрутизация будет сильно отличаться от маршрутизации в стационарных сетях.

Известно множество алгоритмов выбора маршрута для специализированных сетей. Один из наиболее интересных — это алгоритм **AODV** (Ad hoc On-demand Distance Vector — маршрутизация по требованию в специализированных сетях на основе вектора расстояний). Об этом можно прочитать у (Perkins and Royer, 1999). AODV является дальним родственником алгоритма Беллмана—Форда (Bellman—Ford) (метод векторов расстояний), адаптированным для работы в мобильной среде и принимающим в расчет ограниченность пропускной способности и срока службы элементов питания — свойства, характерные для мобильных сетей. Еще одной необычной характеристикой является то, что AODV — это алгоритм «по требованию», то есть он вычисляет маршрут только в тот момент, когда появляется желающий отправить пакет тому или иному адресату. Посмотрим, что это значит.

Построение маршрута

Специализированная сеть в любой момент времени может быть описана с помощью графа узлов (маршрутизаторов и хостов). Два узла считаются соединенными (то есть между ними проведена дуга), если они могут связываться напрямую посредством радио. Поскольку у одного из них может быть более мощный передатчик, чем у другого, то возможна ситуация, когда узел *A* соединен с *B*, но *B* не соединен с *A*. Однако для простоты мы будем считать, что все соединения симметричны. Следует заметить, что нахождение одного из узлов в зоне действия другого еще не означает наличия связи между ними. Их могут разделять холмы, здания и другие местные предметы, блокирующие соединение.

Для описания алгоритма воспользуемся рис. 5.18, на котором изображен процесс, запущенный на узле *A*, которому необходимо отправить пакет на узел *I*. Алгоритм AODV на каждом узле ведет таблицу, доступ к которой осуществляется с помощью поля адреса. Таблица содержит информацию об адресате, в том числе

адрес ближайшего соседа, которому необходимо переслать пакет, чтобы он мог достичь пункта назначения. Допустим, *A* просматривает эту таблицу и не находит записи для *I*. Значит, нужно найти маршрут, ведущий к этому узлу. Итак, алгоритм начинает заниматься поисками маршрутов только тогда, когда они реально требуются. Это и делает его алгоритмом «по требованию».

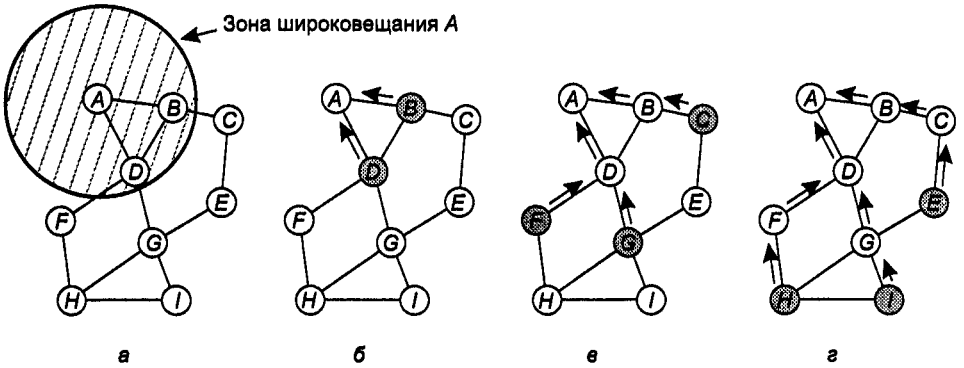


Рис. 5.18. Зона широковещания *A* (*а*); состояние после получения узлами *B* и *D* широковещательного пакета от *A* (*б*); состояние после получения узлами *C*, *F* и *G* широковещательного пакета от *A* (*в*); состояние после получения узлами *E*, *H* и *I* широковещательного пакета от *A* (*г*). Затененными кружочками обозначены новые получатели. Стрелками показаны возможные обратные маршруты

Для поиска *I* узел *A* генерирует специальный пакет запроса маршрута ROUTE REQUEST и распространяет его по сети широковещательным способом. На рис. 5.18, *а* показано, что этот пакет достигает узлов *B* и *D*. На самом деле, причиной установления именно узлами *B* и *D* соединения с *A* является то, что они могут получать пакеты от *A*. Например, *F* не соединен дугой с *A*, потому что он не может принимать радиосигнал от этого узла. То есть *F* не соединен с *A*.

Формат пакета запроса маршрута показан на рис. 5.19. В нем, как видно из этого рисунка, содержатся адреса источника и приемника (обычно IP-адреса), с помощью которых можно понять, кто кого ищет. Также содержится поле *Идентификатор запроса*, которое представляет собой локальный счетчик, обновляемый каждым узлом независимо и инкрементирующийся всякий раз, когда распространяется пакет запроса маршрута. Поля *Адрес источника* и *Идентификатор запроса* вместе единственным образом идентифицируют пакет ROUTE REQUEST, что позволяет узлам обнаруживать и отвергать любые дубликаты.

Адрес отправителя	Идентификатор запроса	Адрес получателя	Порядковый номер отправителя	Порядковый номер получатвля	Счетчик первходов

Рис. 5.19. Формат пакета ROUTE REQUEST

В дополнение к счетчику *Идентификатор запроса* каждый узел имеет второй счетчик, который инкрементируется всякий раз при отправке пакета для запроса маршрута или ответе на такой пакет. Его работа напоминает часы, и используется

он для того, чтобы можно было отличить новые маршруты от старых. Четвертое поле, показанное на рис. 5.19, это счетчик узла *A*; пятое — последнее значение порядкового номера пакета, полученного от *I* (оно равно 0, если такого пакета не было). Вскоре мы более подробно раскроем назначение этих полей Наконец, последнее поле — *Счетчик переходов* — запоминает количество пересылок, совершенных пакетом. В начале работы алгоритма оно равно нулю.

Когда пакет запроса маршрута прибывает на узел (например, на узлы *B* и *D*), с ним происходит следующее:

1. Пара значений полей *Адрес источника* и *Идентификатор запроса* ищется в таблице локальной истории. С их помощью можно выяснить, приходил ли уже этот запрос и обрабатывался ли он. Если обнаруживается, что пакет является дубликатом, он отвергается и его обработка прекращается. В противном случае указанная пара значений заносится в таблицу истории, чтобы в будущем можно было обнаружить дубликаты. Обработка запроса продолжается.
2. Приемник ищет адрес назначения в таблице маршрутов. Если известен достаточно свежий маршрут, отправителю посылается пакет наличия маршрута ROUTE REPLY, сообщающий ему о том, как можно достичь получателя (в двух словах: «Используй меня»). Что значит «свежий маршрут»? Имеется в виду, что поле *Порядковый номер получателя* в таблице маршрутизации имеет значение большее или равное *Порядковому номеру получателя* из пакета запроса маршрута. Если оно меньше, значит, хранящийся в таблице маршрут является более старым, нежели предыдущий маршрут, имевшийся у отправителя к тому же пункту назначения. В этом случае выполняется пункт 3.
3. Поскольку у приемника отсутствует свежий маршрут к адресату, он инкрементирует поле *Счетчик переходов* и вновь широковещательным образом распространяет пакет запроса маршрута. Из пакета извлекаются данные и сохраняются в виде новой записи в таблице обратных маршрутов. Эти данные будут использоваться для построения обратного пути, по которому впоследствии необходимо будет послать ответный пакет отправителю. Стрелки на рис. 5.18 как раз показывают процесс построения обратного пути. Для записи о только что созданном обратном пути запускается таймер. При наступлении таймаута запись удаляется.

Ни *B*, ни *D* не знают, где находится узел *I*, поэтому каждый из них создает обратный путь к *A*, как показано стрелками на рис. 5.18, и широковещательным способом распространяет пакет со *Счетчиком переходов*, установленным в единицу. Этот пакет от *B* достигает *C* и *D*. Узел *C* делает запись в таблице обратных путей и, в свою очередь, тоже широковещательным способом распространяет пакет далее. Что касается *D*, то он отвергает пакет: для него это дубликат. Разумеется, и *B* отвергает пакеты, полученные от *D*. Тем не менее, *F* и *G* принимают широковещательное сообщение от *D* и сохраняют его, как показано на рис. 5.18, *в*. После того как *E*, *H* и *I* получают широковещательный пакет, запрос маршрута наконец достигает узла назначения (*I*). Этот счастливый миг запечатлен на рис. 5.18, *г*. Обратите внимание: несмотря на то, что мы показали распростране-

ние широковещательного пакета в виде трех стадий, на самом деле рассылка этого пакета разными узлами никак не координируется.

В ответ на пришедший запрос узел *I* генерирует пакет наличия маршрута ROUTE REPLY, показанный на рис. 5.20. Поля *Адрес отправителя*, *Адрес получателя* и *Счетчик переходов* копируются из ROUTE REQUEST, а *Порядковый номер получателя* берется из собственного счетчика, хранящегося в памяти. Поле *Счетчик переходов* устанавливается в 0. Поле *Время существования* используется для управления реализуемостью маршрута. Данный пакет распространяется методом одноадресной передачи на тот узел, с которого пришел запрос маршрута. В данном случае он уходит на узел *G*. Затем, в соответствии с установленным обратным путем, он попадет на *D* и наконец на *A*. При проходе каждого узла *Счетчик переходов* инкрементируется, так что узел-отправитель может увидеть, насколько далеко от него находится узел-получатель (*I*).

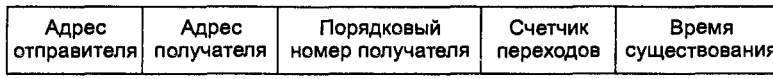


Рис. 5.20. Формат пакета ROUTE REPLY

Каждый узел, через который проходит пакет на обратном пути (к *A*), проверяет его. При выполнении хотя бы одного из трех условий на его основе строится запись в локальной таблице маршрутов о пути к *I*. Вот эти условия:

1. Не известен ни один маршрут к *I*.
2. Последовательный номер для *I* в пакете ROUTE REPLY больше, чем значение в таблице маршрутизации.
3. Последовательные номера равны, но новый путь короче.

Таким образом, все узлы, стоящие на обратном пути к *A*, совершенно бесплатно получают информацию о маршруте к узлу *I*. Это как бы побочный продукт построения маршрута для *A*. Узлы, получившие исходный пакет запроса маршрута, но не стоящие на обратном пути (узлы *B*, *C*, *E*, *F* и *H* в данном примере) удаляют запись в таблице обратных маршрутов, когда ассоциированный с ней таймер достигает тайм-аута.

В больших сетях алгоритмом генерируется много широковещательных пакетов даже для адресатов, расположенных довольно близко друг к другу. Число этих пакетов может быть уменьшено следующим образом. *Время жизни* IP-пакета устанавливается отправителем в значение, соответствующее ожидаемому диаметру сети, и декрементируется при каждой пересылке. Когда его значение становится равным 0, пакет отвергается, а не распространяется дальше.

При этом процесс поиска пути немного изменяется. Для обнаружения адресата отправитель рассылает пакет запроса маршрута с *Временем жизни*, равным 1. Если в течение разумного времени ответ не приходит, посылается еще один запрос с *Временем жизни*, равным 2, и т. д. Таким образом, поиск, начавшийся в какой-то локальной области, все больше расширяет свой охват.

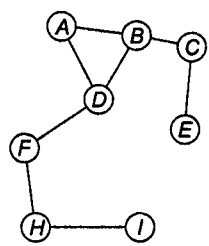
Обслуживание маршрута

Поскольку узлы могут перемещаться и выключаться, топология сети может изменяться совершенно спонтанно. Например, если на рис. 5.18 узел *G* выключится, *A* не поймет, что путь к *I* (*ADGI*) больше не может быть реализован. Алгоритму нужно как-то с этим бороться. Периодически все узлы рассылают сообщение приветствия *Hello*. Ожидается, что все узлы, будучи истинными джентльменами, ответят на него. Если ответ не приходит, значит, сосед вышел из зоны действия и больше не связан с данным узлом. Аналогичным образом, если он пытается послать пакет соседу, который не отвечает, он узнает, что связь с ним недоступна.

Эта информация используется для удаления нерабочих путей. Для каждого из возможных адресатов каждый узел *N* хранит историю о том, какие соседи снабжали узел пакетами для данных адресатов в течение последних ΔT секунд. Такие соседи называются **активными соседями** узла *N* для данного адресата. Узел *N* осуществляет сбор подобных сведений с помощью таблицы маршрутизации, которая, как известно, в качестве индекса использует адрес назначения. В этой таблице указан тот узел, на который нужно переслать пакет, чтобы он мог дойти до адресата. Кроме того, в ней имеются сведения об оставшемся числе переходов, последнем порядковом номере получателя, а также об активных соседях данного адресата. Вид возможной таблицы маршрутизации для узла *D* при топологии, рассматриваемой в нашем примере, показан на рис. 5.21, *а*.

Адресат	Следующий переход	Расстояние	Активные соседи	Прочие поля
A	*A*	1	*F*, *G*	
B	*B*	1	*F*, *G*	
C	*B*	2	*F*	
E	*G*	2		
F	*F*	1	*A*, *B*	
G	*G*	1	*A*, *B*	
H	*F*	2	*A*, *B*	
I	*G*	2	*A*, *B*	

а



б

Рис. 5.21. Таблица маршрутизации узла *D* перед выходом из сети узла *G* (*а*); граф-схема сети после выхода из нее *G* (*б*)

Когда какой-либо из соседей узла *N* становится недоступным, проверяется его таблица маршрутизации — ведь теперь нужно понять, к каким адресатам лежал путь через ушедший узел. Всем оставшимся активным соседям сообщается, что такие пути больше нельзя использовать и их следует удалить из таблиц маршрутизации. Активные соседи передают эти новости своим активным соседям, и так далее, пока все пути, зависевшие от ушедшего узла, не будут удалены из всех таблиц.

Рассмотрим наш предыдущий пример, предположив, что *G* внезапно выключился. Образовавшаяся в результате этого события топология показана на рис. 5.21, *б*. Когда *D* обнаруживает, что *G* ушел из сети, он просматривает свою таблицу маршрутизации и видит, что *G* стоял на пути к *E*, *G* и *I*. Объединением активных соседей для данных адресатов является множество {*A*, *B*}. Другими словами, *A* и *B* содержат записи о маршрутах, проходящих через *G*, поэтому их нужно проинформировать о том, что эти маршруты больше не работают. *D* сообщает им об этом, посылая специальные пакеты, заставляющие их обновить свои таблицы соответствующим образом. Сам узел *D* удаляет записи для адресатов *E*, *G* и *I* из таблицы маршрутизации.

Из приведенного описания это, может быть, и не очевидно, но основная разница между AODV и алгоритмом Беллмана—Форда состоит в том, что узлы не занимаются периодической широковещательной рассылкой пакетов, содержащих полные таблицы маршрутизации. Благодаря этому более эффективно используется полоса пропускания и увеличивается время работы элементов питания.

AODV, впрочем, может также заниматься широковещательной и групповой маршрутизацией. Детали см. в (Perkins and Royer, 2001). Маршрутизация в специализированных сетях — чрезвычайно популярная сегодня область исследований. Вопросам, связанным с ней, посвящено большое количество материалов. Например, (Chen и др., 2002; Hu and Johnson, 2001; Li и др., 2001; Raju and Garcia-Luna-Aceves, 2001; Ramanathan and Redi, 2002; Royer and Toh, 1999; Spohn and Garcia-Luna-Aceves, 2001; Tseng и др., 2001; Zadeh и др., 2002).

Поиск узла в равноранговых сетях

Относительно новым явлением являются равноранговые сети, в которых большое количество пользователей, имеющих обычно постоянное кабельное соединение с Интернетом, работает с разделяемыми ресурсами. Первым широко известным применением равноранговых сетей стало создание нелегальной системы Napster, 50 миллионов пользователей которой обменивались звукозаписями без разрешения обладателей авторских прав. Это продолжалось до тех пор, пока сеть Napster после бурной полемики не была закрыта решением суда. Тем не менее, у равноранговых сетей есть множество интересных и в то же время легальных применений. С ней связаны некоторые проблемы, сходные с проблемой маршрутизации, хотя и не такие же, как мы только что изучили. Сейчас мы их вкратце рассмотрим.

Что делает равноранговые системы интересными, так это их полная распределенность. Все узлы симметричны, никакого единого управления или иерархии не существует. В типичной равноранговой сети пользователи обладают какой-то информацией, могущей представлять интерес для других. Это может быть бесплатное программное обеспечение, музыка (являющаяся всеобщим достоянием), фотографии и т. д. Если пользователей много, они друг о друге ничего не знают и, возможно, никогда не узнают. Но совершенно непонятно, как искать что-то нужное в такой сети! Одним из решений является создание централизованной базы данных, но это может быть неосуществимо при некоторых условиях (напри-

мер, в сети нет добровольцев, желающих содержать и обслуживать такую базу). То есть проблема сводится к тому, что пользователю нужен какой-то метод поиска узла, на котором хранится интересующая его информация, в условиях отсутствия централизованной базы данных и даже централизованного индекса.

Предположим, что каждый пользователь владеет одной или более единицей информации, такой как песни, программы, фотографии, файлы и тому подобное — все то, чем другие пользователи, возможно, заинтересуются. У каждого такого объекта имеется строка ASCII, именующая его. Потенциальный пользователь знает только эту строку и стремится найти одного или нескольких пользователей, у которых есть данный объект, и узнать его (их) IP-адрес.

В качестве примера рассмотрим распределенную генеалогическую базу данных. У каждого генеалога есть онлайновый набор записей о его предках и родственниках, возможно, с фотографиями, аудиозаписями или даже видеозаписями. Понятно, что общий прадедушка был у многих, поэтому запись о нем будет иметься на нескольких узлах сети. Запись идентифицируется по имени человека, записанном в какой-нибудь канонической форме. И тут генеалог обнаруживает в архиве завещание прадедушки, в соответствии с которым золотые карманные часы он передает своему племяннику. Теперь необходимо узнать имя этого племянника и того, у кого могут быть какие-либо сведения о нем. Как же это сделать, не имея централизованной базы данных?

Для решения этой проблемы было предложено несколько различных алгоритмов. Мы рассмотрим **метод хорды** (Dabek и др., 2001a; Stoica и др., 2001). Несколько упрощенное объяснение принципа его работы таково. Пусть система состоит из n пользователей. У каждого из них есть какие-то записи, и каждый готов хранить биты и части индекса, которые могут быть использованы другими. У каждого узла есть IP-адрес, который может быть закодирован m-битным номером с помощью хэш-функции. В методе хорды используется алгоритм SHA-1 для вычисления значения хэш-функции. SHA-1 применяется в криптографии, и мы рассмотрим его в главе 8. А сейчас нам просто надо знать, что это некоторая функция с аргументом в виде строки переменной длины и значением — 160-битным числом высокой степени случайности. Таким образом, любой IP-адрес можно закодировать 160-битным числом, называемым **идентификатором узла**.

Концепция состоит в том, что вообще все 2^{160} идентификаторов располагаются в возрастающем порядке, образуя большой круг чисел. Некоторые из идентификаторов соответствуют реально существующим узлам, но это лишь малая часть из них. На рис. 5.22 показан круг идентификаторов для $m = 5$ (на дуги внутри круга пока внимания не обращайте). В этом примере реальным узлам соответствуют идентификаторы 1, 4, 7, 12, 15, 20 и 27. На рисунке кружочки с этими номерами закрашены. Всем остальным идентификаторам нельзя поставить в соответствие какие-либо узлы.

Определим функцию определения последователя *successor*(k) как идентификатор первого реального узла, следующего после k в описанном нами круге (по часовой стрелке). Например, *successor*(6) = 7, *successor*(8) = 12, *successor*(22) = 27.

Названия записей (например, названия песен, имена предков и т. п.) также обрабатываются хэш-функцией *hash* (с помощью алгоритма SHA-1) и превраща-

ются в 160-битные числа, называемые **ключами**. Таким образом, чтобы получить ключ (*key*) из названия записи (*name*), необходимо вычислить *key* = *hash*(*name*). Такое вычисление является просто локальной процедурой, вызовом функции *hash*. Если держатель генеалогической записи хочет сделать ее доступной всем, он должен построить кортеж (набор взаимосвязанных величин), состоящий из полей (*name*, *мой_IP-адрес*), и затем выполнить функцию *successor*(*hash*(*name*)) для сохранения кортежа в общепринятой форме. Если существует несколько записей (на разных узлах) для одного и того же названия, все их кортежи будут храниться на одном и том же узле. То есть индекс распределяется случайным образом между узлами. Во избежание сбоев системы, для хранения кортежей на p узлах используется p различных хэш-функций, но далее мы не будем учитывать этот нюанс.

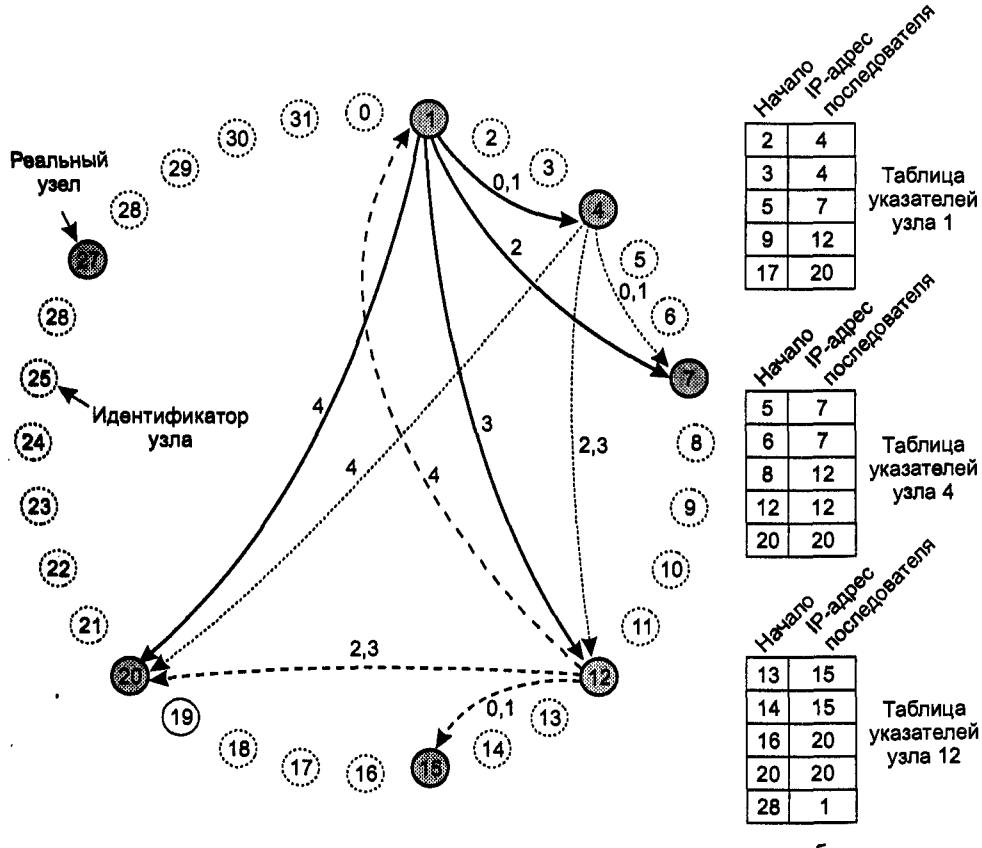


Рис. 5.22. Набор из 32 идентификаторов узлов, выстроенных по кругу (*а*). Закрашенные кружочки соответствуют реальным узлам. Дугами обозначены указатели с узлов 1, 4 и 12. Подписи на дугах соответствуют индексам таблицы. Примеры таблиц указателей (*б*)

Если кто-то из пользователей затем захочет найти название записи (*name*), он вычислит значение хэш-функции, получит ключ (*key*) и затем с помощью функции *successor*(*key*) сможет найти IP-адрес узла, хранящего кортежи индексов. Первый шаг осуществляется довольно просто, не в пример второму. Для того чтобы можно было найти IP-адрес узла, соответствующего какому-либо ключу, каждый узел должен поддерживать определенные служебные структуры данных. Одной из них является IP-адрес последователя. Например, на рис. 5.22 последователь узла 4 — это узел 7, а последователь узла 7 — узел 12.

Теперь можно начинать поиск. Запрашивающий узел посылает последователю пакет, содержащий его IP-адрес и ключ, который он ищет. Этот пакет пересылается по кругу до тех пор, пока не будет найден искомый узел. На нем проверяется соответствие имеющейся информации ключу, и при положительном результате проверки пакет возвращается напрямую запрашивающему узлу, чей IP-адрес содержится в запросе.

В качестве первой оптимизации алгоритма предлагается следующее: каждый узел должен знать IP-адрес не только последователя, но и предшественника, тогда запрос можно будет распространять одновременно в двух направлениях — по часовой стрелке и против часовой стрелки, в зависимости от того, какой из путей кажется более коротким. Например, на рис. 5.22 узел 7 может отправить пакет по часовой стрелке, если ищется узел 10. А если ищется узел 3, то логичнее направить пакет против часовой стрелки.

Даже с этой возможностью выбора направления линейный поиск остается крайне неэффективным методом для больших равноранговых сетей, поскольку среднее число узлов, которое потребуется обойти для поиска записи, равно $n/2$. Значительно ускорить поиск позволяет поддерживаемая каждым узлом специальная таблица, которая в методе хорд носит название **таблицы указателей**. В ней имеется m записей, пронумерованных от 0 до $m - 1$. Каждая запись содержит два поля: *начало* и IP-адрес последователя — *successor*(*start*), как показано на рис. 5.22, *б*. Значения полей записи i на узле k равны:

$$Start = k + 2^i \ (modulo 2^m),$$

$$\text{IP-адрес } successor \ (start \ [i]).$$

Обратите внимание: на узлах хранится сравнительно небольшое число адресов, и большинство из них расположено довольно близко друг от друга в терминах идентификаторов узлов.

С использованием таблицы указателей процесс поиска ключа на узле k выглядит следующим образом. Если ключ попадает в диапазон между k и *successor*(k), значит, он находится, на самом деле, на узле *successor*(k), и поиск прекращается. В противном случае таблица указателей просматривается на предмет поиска записи, поле *начало* которой является ближайшим предшественником ключа. Запрос посылается напрямую по IP-адресу из таблицы указателей. Узел с этим адресом просят перенять инициативу поиска. Поскольку искомый ключ находится где-то рядом, но идентификатор имеет меньшее значение, велика вероятность того, что конечный ответ будет дан очень скоро, спустя всего несколько дополнительных запросов. Поскольку каждая итерация при поиске вдвое уменьшает

последующую область рассмотрения (то есть оставшееся расстояние до цели), то можно доказать, что среднее число итераций равно $\log_2 n$.

В качестве первого примера рассмотрим поиск ключа *key* = 3 на узле 1. Узел с идентификатором 1 знает, что 3 лежит где-то между ним самим и его последователем — узлом 4. Он делает вывод о том, что ключ находится на узле 4, и поиск прекращается. Результатом является IP-адрес узла 4.

Второй пример. Пусть узлу 1 теперь понадобилось найти ключ *key* = 14. Число 14 не находится между 1 и 4, поэтому необходимо заглянуть в таблицу указателей. Ближайшим предшественником 14 является 9, поэтому запрос направляется на IP-адрес, хранящийся в записи с полем *начало*, равным 9. Конкретно, это узел 12. Последний видит, что 14 находится между ним самим и последователем — узлом 15, поэтому результатом поиска является IP-адрес узла 15.

Третий пример. Допустим, узлу 1 нужно найти ключ *key* = 16. Запрос, как и в предыдущем примере, отправляется на узел 12. Но на этот раз узел не может самостоятельно ответить на него. Он пытается найти тот узел, который находится ближе всего к 16, но имеет меньший идентификатор. Им является узел 14, который ссылается, в свою очередь, на узел 15. Запрос туда и посылается. Узел 15 видит, что 16 находится между ним самим и его последователем (20), поэтому он возвращает IP-адрес 20 просителю. Ответ возвращается сразу на узел 1.

Поскольку узлы могут появляться в сети и исчезать из нее в любое время, в методе хорд необходимо как-то обрабатывать подобные ситуации. Мы предполагаем, что когда система начинала работать, она была достаточно небольшой, и узлы могли обмениваться информацией напрямую, выстраивая первичный круг идентификаторов и таблицы указателей. После этого уже необходима автоматизированная процедура. Когда новый узел *r* собирается войти в сеть, он должен войти в контакт с какой-либо из уже находящихся в сети станций и попросить ее поискать для него IP-адрес *successor*(*r*). Затем новый узел выясняет, кто будет его предшественником (*successor*(*r*) для предшественника). Все это нужно для того, чтобы можно было определить свое место в круге идентификаторов. Например, если узел 24 на рис. 5.22 захочет войти в сеть, он может поинтересоваться у любого узла, чему равно значение *successor*(24). Выясняется, что оно равно 27. Затем у узла 27 надо узнать, кто является его предшественником (20). После того как обоим этим узлам сообщается о существовании нового узла, узел 20 помечает себе, что его последователем становится узел 24, а узел 27 — что этот узел становится его предшественником. Кроме того, узел 27 передает ключи диапазона 21–24 новичку. С этого момента последний считается зарегистрированным в равноранговой сети.

Тем не менее, ситуация изменилась, и многие таблицы указателей теперь содержат ошибки. Чтобы исправить их, все узлы запускают фоновые процессы, которые периодически пересчитывают таблицы указателей, обращаясь к функции *successor*. Когда какой-то из этих запросов наталкивается на новый узел, запись таблицы обновляется.

Если узел уходит вежливо, он передает свои ключи последователю и информирует предшественника о своем скором отбытии. При этом в цепочке идентификаторного круга предшественник теперь оказывается непосредственным соседом последователя. Однако если с узлом происходит какая-то авария, возникают

проблемы у его предшественника, поскольку он не знает, кто является его последователем. Решение данной проблемы состоит в том, что узлы запоминают не только своего прямого последователя, но и еще s последователей. Получается, что без тяжких последствий для окружающих из сети могут выпасть $s - 1$ последовательных узлов, и круг при этом не разорвется.

Метод хорд используется для создания распределенных файловых систем (Dabek и др., 2001b) и других приложений; научные изыскания в этой области идут и сейчас. Одна из равноранговых систем, Pastry, и ее приложения описаны в (Rowston and Druschel, 2001a; Rowston and Druschel, 2001b). Еще одна система, Freenet, обсуждается в (Clarke и др., 2002). Наконец, четвертая система этого типа описана в (Ratnasamy и др., 2001).

Алгоритмы борьбы с перегрузкой

Когда количество пакетов, передаваемых одновременно по подсети (или ее части), превышает некий пороговый уровень, производительность сети начинает снижаться. Такая ситуация называется **перегрузкой**. График на рис. 5.23 изображает симптомы этого явления. Когда число пакетов, посылаемых хостами в сеть, не превышает ее пропускной способности, все они доставляются адресатам (кроме небольшого процента поврежденных ошибками передачи). При этом количество доставленных пакетов пропорционально количеству посланных. Однако по мере роста трафика маршрутизаторы перестают успевать обрабатывать все пакеты и начинают их терять. При дальнейшем увеличении числа отправляемых пакетов ситуация продолжает ухудшаться. Когда число пакетов достигает максимального уровня, производительность сети начинает снижаться. При очень высоком уровне трафика производительность сети падает до совсем низкого уровня и практически никакие пакеты не доставляются.

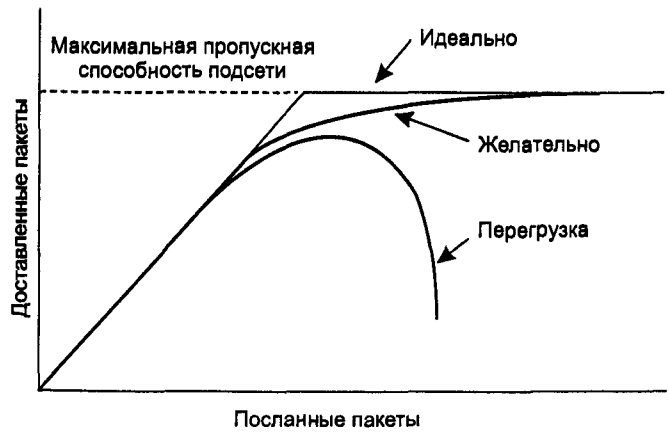


Рис. 5.23. При слишком высоком уровне трафика начинается перегрузка, и производительность сети резко снижается

Перегрузка может быть вызвана несколькими факторами. Если вдруг потоки пакетов начинают прибывать на маршрутизатор сразу по трем или четырем входным линиям и всем им нужна одна и та же выходная линия, то образуется очередь. Когда у маршрутизатора закончится свободная память для буферизации всех прибывающих пакетов, их негде будет сохранять и они начнут теряться. Увеличение объема памяти маршрутизаторов может в какой-то степени помочь, но Нэгл (Nagle) в 1987 году показал, что даже если у маршрутизаторов будет бесконечное количество памяти, ситуация с перегрузкой не улучшится, а, наоборот, ухудшится, так как к тому времени, когда пакеты доберутся до начала очереди, они уже запоздают настолько, что источником будут высланы их дубликаты. Все эти пакеты будут посланы следующему маршрутизатору, еще более увеличивая нагрузку на всем протяжении маршрута к получателю.

Медленные процессоры также могут служить причиной заторов. Если центральные процессоры маршрутизаторов слишком медленно выполняют свои задачи, связанные с учетом, управлением очередями, обновлением таблиц и т. д., то очереди будут появляться даже при достаточно высокой пропускной способности линий. Аналогично, линии с низкой пропускной способностью также могут вызывать заторы в сети. Если заменить линии более совершенными, но оставить старые процессоры, или наоборот, такие действия обычно немного помогают, но часто просто приводят к сдвигу узкого места, вызванного несоответствием производительности разных частей системы. Проблема узкого места сохраняется до тех пор, пока компоненты системы не будут должным образом сбалансированы.

Необходимо пояснить, в чем состоит разница между борьбой с перегрузкой и управлением потоком. Предотвращение перегрузки гарантирует, что подсеть справится с предлагаемым ей трафиком. Это глобальный вопрос, включающий поведение всех хостов и маршрутизаторов, процессов хранения и пересылки на маршрутизаторах, а также множество других факторов, снижающих пропускную способность подсети.

Управление потоком, напротив, относится к трафику между двумя конкретными станциями — отправителем и получателем. Задача управления потоком состоит в согласовании скорости передачи отправителя со скоростью, с которой получатель способен принимать поток пакетов. Управление потоком обычно реализуется при помощи обратной связи между получателем и отправителем.

Чтобы разница между этими двумя проблемами стала яснее, представьте себе оптоволоконную сеть с пропускной способностью 1000 Гбит/с, по которой суперкомпьютер пытается передать персональному компьютеру файл со скоростью 1 Гбит/с. Хотя перегрузки сети в данной ситуации не наблюдается, алгоритм управления потоком довольно часто заставляет суперкомпьютер приостанавливать передачу, чтобы персональный компьютер мог успевать принимать файл.

А вот другой пример. Рассмотрим сеть с промежуточным хранением, состоящую из 1000 больших компьютеров, соединенных линиями с пропускной способностью 1 Мбит/с. Одна половина компьютеров пытается передавать файлы другой половине со скоростью 100 Кбит/с. Здесь проблема заключается уже не в том, что медленные получатели не успевают принимать данные, посылаемые им

быстрыми отправителями, а просто в неспособности сети пропустить весь предлагаемый трафик.

Причина, по которой управление потоком и борьбу с перегрузкой часто путают, заключается в том, что алгоритмы борьбы с перегрузкой также используют обратную связь в виде специальных сообщений, посылаемых различным отправителям, с просьбой передавать данные помедленнее, когда в сети появляются заторы. Таким образом, хост может получить просьбу замедлить передачу в двух случаях: когда с передаваемым потоком не справляется получатель или когда с ним не справляется вся сеть. В дальнейшем мы еще будем рассматривать этот вопрос.

Мы начнем изучение алгоритмов борьбы с перегрузкой с рассмотрения общей модели. Затем мы познакомимся с общим подходом к предотвращению перегрузки, а также с различными динамическими алгоритмами борьбы с перегрузкой, которую не удалось предотвратить.

Общие принципы борьбы с перегрузкой

Многие проблемы, возникающие в сложных системах, таких как компьютерные сети, следует рассматривать с точки зрения теории управления. При таком подходе все решения делятся на две группы: без обратной связи и с обратной связью. Решения без обратной связи заключаются в попытках решить проблему с помощью улучшения дизайна системы, пытаясь, таким образом, в первую очередь предотвратить возникновение самой ситуации перегрузки. Никаких корректирующих действий во время работы системы не предпринимается.

К методам управления без обратной связи относятся решения о том, когда разрешать новый трафик, когда отвергать пакеты и какие именно, а также составление расписаний для различных участков сети. Общее в этих решениях то, что они не учитывают текущего состояния сети.

Решения с обратной связью, напротив, основываются на учете текущего состояния системы. Этот подход состоит из трех следующих частей:

1. Наблюдения за системой с целью определить, где и когда произойдет перегрузка.
2. Передачи информации о перегрузке в те места, где могут быть предприняты соответствующие действия.
3. Принятия необходимых мер при работе системы для устранения перегрузки.

При наблюдении за состоянием подсети с целью обнаружения перегрузки могут измеряться различные параметры. Среди них следует выделить следующие: процент пакетов, отвергаемых из-за отсутствия свободного места в буфере; средняя длина очереди; процент пакетов, переданных повторно по причине истекшего времени ожидания подтверждения; среднее время задержки пакетов и среднеквадратичное отклонение задержки пакетов. Во всех случаях увеличивающиеся значения параметров являются сигналами о растущей перегрузке.

Второй этап борьбы с перегрузкой состоит в передаче информации о перегрузке от места ее обнаружения туда, где могут быть приняты какие-то меры по

ее устранению. Очевидное решение заключается в том, чтобы маршрутизатор, обнаруживший перегрузку, пересылал источнику или источникам трафика пакет с извещением о наличии проблемы. Такие пакеты, конечно, окажут дополнительную нагрузку на сеть как раз в тот момент, когда нагрузку необходимо снизить.

Существуют, однако, и другие решения. Например, можно зарезервировать в каждом пакете бит или поле, которые будут заполняться маршрутизаторами при достижении перегрузкой порогового уровня. Таким образом, соседи этого маршрутизатора будут предупреждены о том, что на данном участке сети наблюдается перегрузка.

Еще один метод состоит в том, что хосты или маршрутизаторы периодически посылают пробные пакеты, явно спрашивая друг друга о перегрузке. Собранная таким образом информация может затем использоваться для выбора маршрутов в обход участков сети, в которых возникла проблема с перегрузкой. Так, некоторые радиостанции обзавелись вертолетами, летающими над городами и сообщающими слушателям о заторах на дорогах в надежде, что слушающие их водители выберут маршруты для своих пакетов (то есть машин) в обход пробок.

Все системы с обратной связью предполагают, что получившие информацию о перегрузке в сети хосты и маршрутизаторы предпримут какие-нибудь действия для устранения перегрузки. Чтобы данная схема работала, необходимо тщательно настроить временные параметры. Если каждый раз, когда два пакета приходят одновременно, какой-нибудь нервный маршрутизатор будет кричать «Стоп!», а простояв без работы 20 мкс, он же будет давать команду «Давай!», система будет находиться в состоянии постоянных незатухающих колебаний. С другой стороны, если маршрутизатор будет спокоен, как слон, и для большей надежности станет ждать 30 минут, прежде чем что-либо сообщить, то механизм борьбы с перегрузкой будет реагировать слишком медленно, чтобы приносить вообще какую-либо пользу. Для правильной работы необходимо некоторое усреднение, однако правильный выбор значения постоянной времени является нетривиальной задачей.

Известны различные алгоритмы борьбы с перегрузкой. Янг (Yang) и Редди (Reddy) (1995) даже разработали специальный метод классификации этих алгоритмов. Они начали с того, что разделили все методы на алгоритмы с обратной связью и без нее, как уже описывалось ранее. Затем они разделили алгоритмы без обратной связи на работающие у отправителя и у получателя. Алгоритмы с обратной связью также были разделены на две подкатегории: с явной и неявной обратной связью. В алгоритмах с явной обратной связью от точки возникновения перегрузки в обратном направлении посылаются пакеты, предупреждающие о заторе. В алгоритмах с неявной обратной связью источник приходит к выводу о наличии перегрузки, основываясь на локальных наблюдениях, — например, по значению интервала времени, требующегося для получения подтверждения.

Наличие перегрузки означает, что нагрузка временно превысила возможности ресурсов данной части системы. Есть два решения данной проблемы: увеличить ресурсы системы или снизить нагрузку. Например, подсеть может использовать телефонные линии с модемами, чтобы увеличить пропускную способность между определенными точками. В спутниковых системах большую пропускную спо-

собность часто дает увеличение мощности передатчика. Распределение трафика по нескольким маршрутам вместо постоянного использования одного и того же, пусть даже оптимального пути также может позволить ликвидировать местную перегрузку. Наконец, для увеличения пропускной способности сети в случае серьезных заторов могут быть задействованы запасные маршрутизаторы, которые обычно применяются для повышения устойчивости системы в случае сбоя.

Однако иногда увеличить пропускную способность бывает невозможно либо она уже увеличена до предела. В таком случае единственный способ борьбы с перегрузкой состоит в уменьшении нагрузки. Для этого существует несколько способов, включая отказ в обслуживании или снижение уровня обслуживания некоторых или всех пользователей, а также составление более четкого расписания потребностей пользователей в обслуживании.

Некоторые из этих методов, которые будут кратко рассмотрены далее, лучше всего применимы к виртуальным каналам. В подсетях, основанных на использовании виртуальных каналов, эти методы могут применяться на сетевом уровне. В дейтаграммных подсетях они иногда также могут применяться в соединениях транспортного уровня. В данной главе основное внимание будет уделено применению методов борьбы с перегрузкой на сетевом уровне. В следующей главе мы обсудим, что можно сделать на транспортном уровне.

Стратегии предотвращения перегрузки

Начнем изучение методов борьбы с перегрузкой с систем без обратной связи. Эти системы разработаны в первую очередь для предотвращения перегрузки, а не для борьбы с уже имеющей место перегрузкой. Они пытаются достичь своей цели, используя соответствующие стратегии на разных уровнях. В табл. 5.2 показаны различные стратегии уровней передачи данных, сетевого и транспортного, способные влиять на перегрузку [162].

Таблица 5.2. Стратегии предотвращения перегрузки

Уровень	Стратегии
Транспортный	Политика повторной передачи
	Политика кэширования пакетов, приходящих в неверном порядке
	Политика подтверждений
	Политика управления потоком
	Определение тайм-аутов
Сетевой	Виртуальные каналы против дейтаграмм в составе подсети
	Политика очередей пакетов и обслуживания
	Политика игнорирования пакетов
	Алгоритм маршрутизации
	Управление временем жизни пакетов
Передачи данных	Политика повторной передачи
	Политика кэширования пакетов, приходящих в неверном порядке
	Политика подтверждений
	Политика управления потоком

Начнем рассмотрение различных стратегий с уровня передачи данных. Стратегия повторной передачи определяет, насколько быстро у отправителя истекает время ожидания подтверждения и что он передает после того как время ожидания истекло. Нетерпеливый отправитель, у которого время ожидания истекает слишком быстро и который повторно посылает все неподтвержденные пакеты с помощью алгоритма возврата на n, окажет более сильную нагрузку на сеть, нежели ленивый отправитель, использующий выборочный повтор. Тесно связана с этим стратегия кэширования. Если получатели просто игнорируют все пакеты, приходящие не в том порядке, то все проигнорированные пакеты придется передавать позднее еще раз, что окажет дополнительную нагрузку на сеть.

Стратегия подтверждений также влияет на перегрузку. Если каждый пакет немедленно подтверждается получателем, то пакеты с подтверждениями образуют дополнительный трафик. Однако если подтверждения добираются обратно «верхом» на попутном потоке кадров, то количество трафика в сети снижается, зато увеличивается среднее время получения подтверждений, что может, в свою очередь, вызвать увеличение повторно переданных пакетов вследствие истечения времени ожидания подтверждений. Более жесткая схема управления потоком (например, с небольшим размером окна) уменьшает скорость передачи данных и помогает бороться с перегрузкой.

Существует также зависимость перегрузки от того, является ли сетевой уровень дейтаграммным или он основан на виртуальных каналах, так как многие алгоритмы борьбы с перегрузкой работают только в подсетях с виртуальным каналами. Политика очередей пакетов и обслуживания определяет количество очередей у каждого маршрутизатора — например, одна общая очередь для всех линий, или по очереди для каждой линии, или какой-нибудь комбинированный вариант. Она также определяет порядок обработки пакетов (например, поочередно или в порядке приоритетов). Политика игнорирования пакетов является правилом, определяющим набор пакетов, подлежащих отвержению, когда не хватает памяти. Хорошо продуманная стратегия может облегчить симптомы перегрузки, тогда как неудачная политика может даже ухудшить ситуацию.

Хороший алгоритм выбора маршрута может помочь избежать локальной перегрузки, перераспределяя трафик по всем линиям, тогда как неудачный алгоритм может направить слишком большое количество пакетов по одной линии и вызвать затор. Наконец, управление временем жизни пакетов определяет, как долго пакет может перемещаться по сети, прежде чем он будет проигнорирован очередным маршрутизатором. Если это время слишком велико, то потерянные пакеты могут засорять собою сеть, однако если время жизни пакета слишком мало, то пакеты не будут успевать достичь адресата, что приведет к необходимости повторных передач.

На транспортном уровне применяются те же стратегии, что и на уровне передачи данных, но к ним добавляется еще и проблема определения времени ожидания подтверждения, что на транспортном уровне осуществить значительно сложнее, поскольку время пересечения всей сети предсказать значительно сложнее, чем время передачи пакета от какого-либо маршрутизатора до его соседа. Если этот интервал слишком короток, будут высылаться излишние повторные пакеты,

а если слишком велик — перегрузка снизится, но увеличится задержка в случае потери пакета.

Борьба с перегрузкой в подсетях виртуальных каналов

Описанные ранее методы борьбы с перегрузкой в основном являются методами без обратной связи: они в первую очередь пытаются предотвратить перегрузку, а не занимаются устранением уже возникшей перегрузки. В данном разделе мы опишем несколько подходов к динамическому управлению перегрузкой в подсетях виртуальных каналов. В следующих двух разделах будут описаны методы, применимые в любой подсети.

Широко применяемым методом недопущения ухудшения уже начавшейся перегрузки является **управление допуском**. Идея этого метода проста: когда приходит сигнал о перегрузке, никакие новые виртуальные каналы не создаются до тех пор, пока проблема не разрешится. То есть любые попытки установить новые соединения транспортного уровня пресекаются. Понятно, что если пустить в сеть, в которой уже возникла перегрузка, дополнительных пользователей, то ситуация только ухудшится. Хотя такой метод несколько прямолинеен и не эстетичен, зато он дешев, надежен и практичен. В обычных телефонных системах этот метод тоже широко применяется: когда коммутатор оказывается перегруженным, вы поднимаете трубку и не слышите гудка.

Альтернативный подход заключается в том, что создание новых виртуальных каналов разрешается, но эти каналы тщательно прокладываются в обход заторов. Для примера рассмотрим подсеть, показанную на рис. 5.24, в которой два маршрутизатора перегружены.

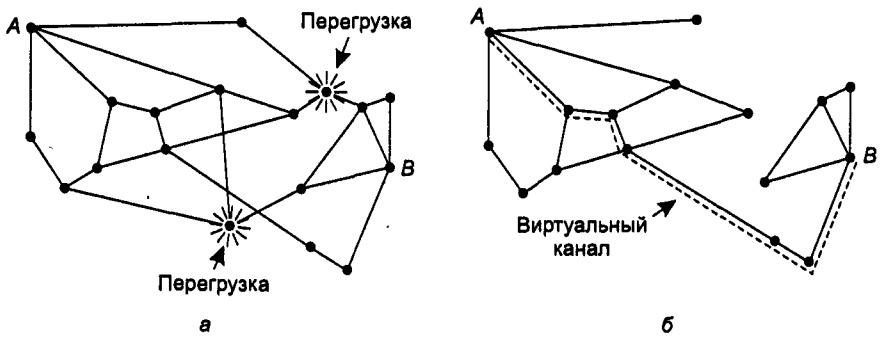


Рис. 5.24. Перегруженная подсеть (*а*); та же подсеть с устраненной перегрузкой (*б*). Показан виртуальный канал между *A* и *B*

Предположим, что хост, соединенный с маршрутизатором *A*, хочет установить соединение с хостом, соединенным с маршрутизатором *B*. В нормальных условиях это соединение прошло бы через один из перегруженных маршрутизаторов. Чтобы этого избежать, подсеть усекается, как показано на рис. 5.24, *б*. При

этом из нее удаляются перегруженные маршрутизаторы и все их линии связи. Пунктирной линией показан возможный маршрут виртуального канала в обход перегруженных маршрутизаторов.

Другая стратегия, связанная с виртуальными каналами, состоит в достижении соглашения между хостом и подсетью во время установки виртуального канала. Эта договоренность обычно описывает объем и форму трафика, требуемое качество обслуживания и другие параметры. В качестве выполнения своей части соглашения подсеть обычно резервирует ресурсы на пути следования создаваемого канала. К этим ресурсам относятся память для буферов и таблиц маршрутизаторов и пропускная способность линий. При таком подходе возникновение перегрузки в новом виртуальном канале маловероятно, так как все необходимые ресурсы были зарезервированы и их доступность гарантируется.

Подобное резервирование может выполняться как в виде постоянной стандартной процедуры, так и в виде действия, выполняемого только при возникновении перегрузки. Недостаток постоянного резервирования заключается в том, что на эту процедуру расходуются вычислительные ресурсы. Если шесть виртуальных каналов, которым разрешено использовать по 1 Мбит/с, проходят по одной и той же физической линии с пропускной способностью 6 Мбит/с, линия должна быть помечена как полная, хотя маловероятно, что все шесть виртуальных каналов передают данные одновременно, да еще и используют всю доступную пропускную способность. Следовательно, платой за резервирование будет неиспользованная пропускная способность.

Борьба с перегрузкой в дейтаграммных подсетях

Теперь обратимся к методу, применяемому в дейтаграммных подсетях (впрочем, он может использоваться и в подсетях с виртуальным каналом). Каждый маршрутизатор может запросто следить за использованием своих выходных линий и других ресурсов. Например, с каждой линией может быть связана вещественная переменная u, значение которой в пределах от 0,0 до 1,0 отражало бы использование линии за последнее время. Такую усредненную оценку загруженности линии можно получить с помощью несложных вычислений, периодически замеряя мгновенную загруженность линии f (0 либо 1) и рассчитывая новое значение переменной u по формуле

$$u_{\text{нов}} = au_{\text{ст}} + (1 - a)f,$$

где константа a определяет, насколько быстро маршрутизатор забывает свое прошлое.

Когда значение переменной u начинает превышать некий пороговый уровень, это означает, что линия переходит в опасное состояние. Каждый приходящий пакет проходит проверку: если ему предстоит следовать по линии, находящейся в близком к перегрузке состоянии, то выполняется одно из нескольких действий. Далее мы обсудим, как именно маршрутизатор может отреагировать на эту ситуацию.

Биты предупреждения

В старой архитектуре DECNET сигнализация опасного состояния производилась путем установки специального бита в заголовке пакета. То же самое делается в технологии ретрансляции кадров. Пакет доходит до места назначения, и в подтверждение о его доставке, отсылаемое источнику, включается бит предупреждения, увидев который, источник должен снизить трафик.

Продолжая находиться в опасном состоянии, маршрутизатор устанавливает биты предупреждения во все проходящие мимо него пакеты. Это означает, что хосты-источники информируются о проблеме. Со своей стороны источники изучают ту часть подтверждений, в которой установлены биты предупреждения, и соответствующим образом подстраивают свою скорость передачи. Если такие подтверждения продолжают прибывать, источник продолжает снижать скорость. Достигнув такой скорости, когда пакеты еле-еле просачиваются по линии, хост может начать увеличивать скорость. Обратите внимание: бит предупреждения может установить любой маршрутизатор, поэтому трафик может начать повышаться только тогда, когда со всеми маршрутизаторами все в порядке.

Сдерживающие пакеты

Предыдущий алгоритм борьбы с перегрузкой действует довольно хитро: он использует окольные средства для сообщения источнику о том, что неплохо бы умерить пыл. Но почему бы не сказать ему об этом прямо? Такая идея привела к созданию подхода, при котором маршрутизатор сам отправляет источнику **сдерживающий пакет**. Информация об источнике берется из задержанного пакета. Исходный пакет помечается (специальный бит в его заголовке устанавливается в единицу), чтобы он больше не порождал сдерживающих пакетов на пути следования, и отправляется дальше по своему обычному маршруту.

Когда хост-отправитель получает сдерживающий пакет, он должен уменьшить трафик к указанному получателю на некоторый процент X. Поскольку другие пакеты, направляющиеся к тому же адресату, скорее всего, в этот момент уже находятся в пути, они также породят сдерживающие пакеты. Поэтому в течение фиксированного интервала времени хост должен игнорировать сдерживающие пакеты, относящиеся к тому же получателю. По истечении этого периода времени хост начинает прослушивать другой интервал на предмет сдерживающих пакетов. Если приходит хотя бы один, это означает, что линия все еще перегружена, поэтому хост еще сильнее снижает выходной поток и снова начинает игнорировать последующие сдерживающие пакеты. Если в течение второго интервала времени (периода ожидания сдерживающих пакетов) сдерживающие пакеты не приходят, хост может снова увеличить поток. Обратная связь, присутствующая в данном протоколе, может помочь предотвратить перегрузку, не ограничивая поток до тех пор, пока не возникнет необходимость.

Хосты могут уменьшать трафик, изменяя свои стратегические параметры, например размер окна. Обычно первый сдерживающий пакет уменьшает скорость передачи данных в два раза, следующий — в четыре и т. д. Увеличения скорости производятся меньшими приращениями, чтобы избежать слишком быстрого повторения перегрузки.

Существуют различные варианты этого алгоритма борьбы с перегрузкой. В одном из них маршрутизаторами применяется несколько пороговых уровней загруженности линии. В зависимости от того, какой из порогов пересечен, сдерживающие пакеты могут содержать мягкое или строгое предупреждение либо ультиматум.

В качестве варианта может измеряться длина очереди или объем свободной памяти маршрутизатора. При этом можно использовать ту же экспоненциальную весовую функцию, что и для u.

Сдерживающие пакеты для ретрансляционных участков

При больших скоростях передачи данных и при сильной удаленности хостов с отправкой сдерживающих пакетов возникают проблемы, поскольку реакция на них оказывается крайне запоздалой. Рассмотрим, к примеру, хост в Сан-Франциско (маршрутизатор A на рис. 5.25), посылающий поток данных на хост, расположенный в Нью-Йорке (маршутизатор D на рис. 5.25), со скоростью 155 Мбит/с. Если у нью-йоркского хоста станет кончаться буферная память, сдерживающему пакету потребуется около 30 мс на то, чтобы добраться обратно в Сан-Франциско и сообщить о том, что необходимо снизить объем трафика. Распространение сдерживающего пакета схематично показано на второй, третьей и четвертой диаграммах рис. 5.25, *а*. За те 30 мс, пока этот пакет движется по сети, в сторону нью-йоркского маршрутизатора передается еще 4,6 Мбит данных, с которыми тоже надо как-то совладать. Только к седьмой диаграмме (рис. 5.25, *а*) маршрутизатор заметит начавшееся снижение потока.

Однако есть альтернативный метод, позволяющий бороться с этой проблемой. Он заключается в том, что сдерживающий пакет влияет на трафик каждого маршрутизатора, через который он проходит. Это показано на последовательности диаграмм на рис. 5.25, *б*. Как только сдерживающий пакет достигает точки F, поток данных от F в сторону D должен быть уменьшен. Таким образом, F резервирует для потока большее количество буферной памяти: источник все еще продолжает заваливать это направление своими данными. Нагрузка на D мгновенно спадает, как головная боль у страдальца, рекламирующего по телевизору чудодейственные пилюли. На следующем шаге сдерживающий пакет, продолжая свой путь, достигает E и приказывает уменьшить поток в сторону F. В результате в течение какого-то времени точке E приходится выдерживать повышенную нагрузку, но зато мгновенно освобождается от своего бремени точка F. Наконец, победный марш сдерживающего пакета приводит его к источнику всех бед — точке A, и теперь поток снижается на самом деле.

Результатом применения метода сдерживания трафика на ретрансляционных участках является максимально быстрое устранение перегрузки в самой горячей точке за счет использования большего объема буферной памяти промежуточных маршрутизаторов. Таким образом, перегрузка пресекается без потери пакетов. Эта идея обсуждается более подробно у (Mishra и Kanakia, 1992).

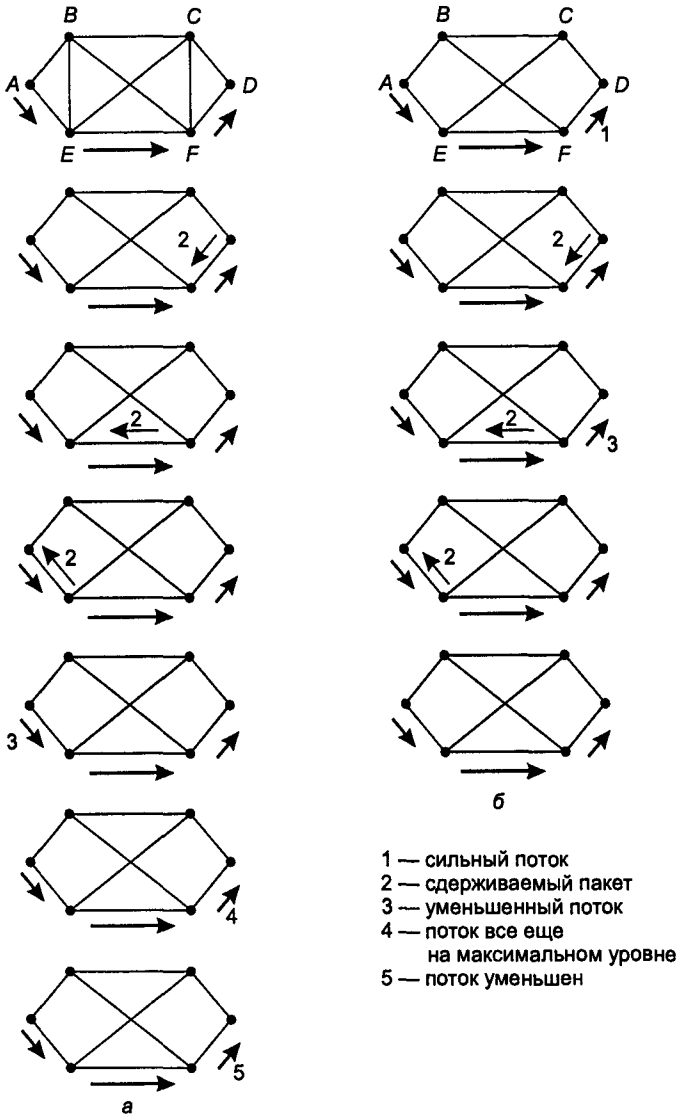


Рис. 5.25. Сдерживающий пакет влияет только на источник (*а*); сдерживающий пакет влияет на все промежуточные участки (*б*)

Сброс нагрузки

Когда ни один из описанных ранее методов не помогает в борьбе с перегрузкой, маршрутизаторы могут ввести в бой тяжелую артиллерию — сброс нагрузки. **Сбросом нагрузки** называется простое игнорирование маршрутизаторами пакетов, которые они не могут обработать. Своим происхождением этот термин обя-

зан системам электроснабжения, где он означает отключение в случае перегрузок отдельных участков во избежание выхода из строя всей системы. Обычно такое происходит в морозные зимние дни, когда потребности в электроэнергии для обогревателей резко возрастают.

Маршрутизатор, заваленный пакетами, может выбирать пакеты просто случайным образом, но обычно имеются более оптимальные варианты. Выбор пакета, который будет отвергнут, может зависеть от приложения, пересылающего этот пакет. Для передачи файла более старый пакет ценится выше нового, так как отвержение пакета номер 6 и сохранение пакетов с номерами с 7-го по 10-й может привести к тому, что получатель запросит еще раз пакеты с 6-го по 10-й (если получатель просто отвергает все пакеты, приходящие не в том порядке). В файле, состоящем из 12 пакетов, выбрасывание 6-го пакета может потребовать повторной передачи пакетов с 7-го по 12-й, тогда как выбрасывание пакета номер 10 может потребовать повторной передачи только пакетов с 10-го по 12-й. Для мультимедийных приложений, напротив, новый пакет важнее старого. Первую стратегию (старое лучше нового) часто называют **винной стратегией**, а вторую (новое лучше старого) — **молочной стратегией**.

Чтобы сделать этот алгоритм еще разумнее, необходимо участие в нем отправителей. Во многих приложениях одни пакеты могут быть значительно важнее других. Например, некоторые алгоритмы сжатия видеосигнала периодически посылают полный кадр, а последующие кадры представляют собой карты изменений относительно последнего полного кадра. В таком случае потеря пакета, содержащего разностный сигнал, не так страшна, как потеря полного кадра. Точно так же при передаче страницы, содержащей текст и рисунок, потеря линии пикселов рисунка может остаться почти незамеченной, тогда как потеря строки текста крайне нежелательна.

Для реализации интеллектуальной стратегии выбрасывания части информации приложения должны помечать свои пакеты классами приоритетов, соответствующими их важности. В этом случае маршрутизаторы смогут сначала выбросить пакеты нижнего класса, затем следующего за ним и т. д. Конечно, при отсутствии стимула все будут помечать свои пакеты не иначе как ОЧЕНЬ ВАЖНО — НИ В КОЕМ СЛУЧАЕ НЕ ВЫБРАСЫВАТЬ.

Стимулом может служить стоимость обслуживания, то есть пересылка пакетов низкоприоритетным классом может быть дешевле, чем высокоприоритетным. В качестве альтернативы источникам может быть ультимативно предложено отправлять высокоприоритетные пакеты только в условиях низкого трафика, а с повышением загрузки сети прекращать их отправку.

Еще один вариант состоит в разрешении хостам превышать пределы, указанные в соглашении, заключенном при создании виртуального канала (например, использовать большую пропускную способность, чем договаривались), но при условии, что весь дополнительный трафик будет помечаться как низкоприоритетный. Такая стратегия весьма удачна, поскольку более эффективно использует свободные ресурсы, разрешая хостам пользоваться ими, пока это никому не мешает, но не закрепляя за ними этого права.

Случайное раннее обнаружение

Хорошо известно, что при борьбе с перегрузкой гораздо проще вовремя обнаружить затор, чем дать ему развиться до критических размеров, а потом думать, что делать в сложившейся ситуации. Это соображение приводит к идее отвержения некоторых пакетов еще до того, как все буферное пространство будет заполнено скопившимися необработанными данными. Популярный алгоритм, реализующий данную идею, называется **случайным ранним обнаружением** (RED — Random Early Detection, Floyd и Jacobson, 1993). Некоторые транспортные протоколы (включая TCP) на потерю своих пакетов отвечают снижением трафика от источника, чего мы, в сущности, и добиваемся. Обоснование такой логики состоит в том, что TCP предназначен для проводных сетей, которые по сути своей являются очень надежными, и потеря пакетов в них чаще всего сигнализирует о переполнении буфера, а не об ошибках передачи. Этот факт и используется для уменьшения перегрузок.

Если заставить маршрутизаторы сознательно терять пакеты еще до того как ситуация станет безнадежной (именно такой момент зашифрован в слове «раннее» из названия подхода), то останется время на то, чтобы предпринять какие-то действия. Для определения условий, при которых следует начинать терять пакеты, маршрутизаторы постоянно высчитывают скользящее среднее длин своих очередей. Когда средняя длина очереди на какой-либо линии превышает пороговое значение, эта линия объявляется перегруженной и выполняются действия по предотвращению затора.

Маршрутизатор не всегда может определить, кто из отправителей больше всех виноват в заваливании линии данными, поэтому пакет из очереди выбирается случайным образом, и это — самое справедливое, что можно сделать в данной ситуации.

Но как маршрутизатор сообщит источнику о возникшей проблеме? Можно послать ему сдерживающий пакет, как описывалось ранее. Но это приведет к созданию дополнительной нагрузки на и так уже почти перегруженную сеть. Другой подход заключается в том, чтобы просто потерять выбранный пакет и никому не сообщать об этом. Источник в конечном счете среагирует на отсутствие подтверждения. Поскольку он знает, что потеря пакетов обычно связана с перегрузкой сети, он уменьшит скорость выдачи пакетов и не станет пытаться во что бы то ни стало пробиться к загруженному маршрутизатору. Такая неявная форма обратной связи может применяться только тогда, когда источник знает, что на потерю пакетов надо реагировать снижением скорости передачи. В беспроводных сетях, где большинство испорченных и потерянных пакетов обязано своим исчезновением шуму в эфире, такой подход не годится.

Борьба с флуктуациями

Для таких приложений как аудио- и видеопередача, не так уж важно, 20 или 30 мс занимает доставка пакетов, до тех пор, пока время доставки постоянно. Колебание (то есть, среднеквадратичное отклонение) времени доставки пакетов называется **флуктуацией**. Если одни пакеты будут доставляться за 20 мс, а другие — за 30 мс,

изображение или звук начнет дрожать. В этом случае говорят о наличии сильных флуктуаций. На рис. 5.26 изображены примеры флуктуаций. Внутри системы, с другой стороны, может существовать договоренность о том, что 99 % пакетов должны быть доставлены с задержкой в диапазоне от 24,5 до 25,5 мс, и качество при этом будет вполне приемлемым.

Выбранный диапазон должен быть, конечно, выполнимым. При вычислении времени задержки необходимо принимать во внимание время передачи по каналу со скоростью света, минимальную задержку при прохождении маршрутизаторов, а также некоторые другие неизбежные задержки.

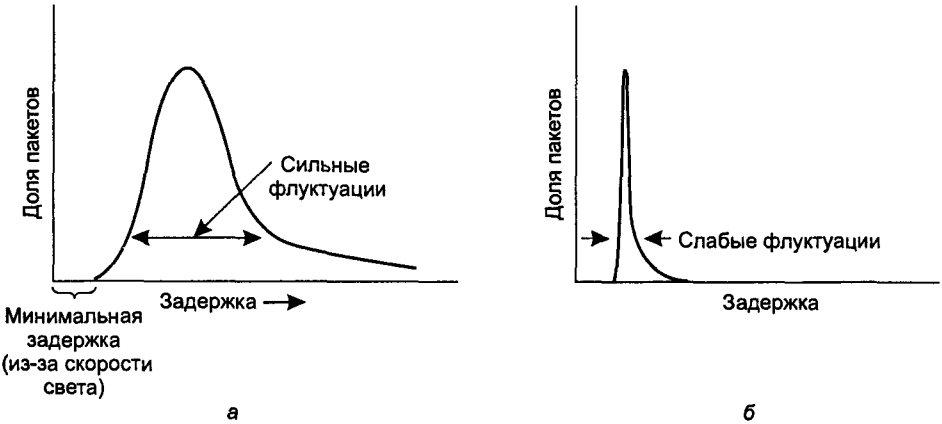


Рис. 5.26. Сильные флуктуации (*а*); слабые флуктуации (*б*)

Для ограничения флуктуаций должно быть вычислено ожидаемое время пересылки по каждому транзитному участку пути. Получив пакет, маршрутизатор проверяет, насколько пакет опаздывает или опережает график. Эта информация хранится в каждом пакете и обновляется каждым маршрутизатором. Если пакет приходит с опережением графика, он удерживается в течение требуемого интервала времени. Если же пакет запаздывает, маршрутизатор пытается отправить его дальше как можно быстрее.

В самом деле, алгоритм, определяющий, какие из пакетов отправить первыми по выходной линии, всегда может выбрать пакет, сильнее всего отстающий от расписания. При этом пакеты, опережающие график, замедляются, а опаздывающие пропускаются в первую очередь, что в обоих случаях уменьшает флуктуации времени доставки пакетов.

В некоторых приложениях, таких как видео по требованию, флуктуации могут быть снижены путем сохранения пакетов в буфере приемника с их последующей выдачей для отображения. При этом сеть не обязана работать в реальном масштабе времени. Тем не менее, в приложениях, которые должны обеспечивать межпользовательское взаимодействие в реальном времени (например, в интернет-телефонии или видеоконференциях), задержка, связанная с буферизацией, совершенно недопустима.

Борьба с перегрузками представляет собой область активных исследований. Текущее положение дел отражено в книге (Gevros и др., 2001).

Качество обслуживания

Методы, рассмотренные нами ранее, направлены на уменьшение перегрузок и повышение производительности в сетях передачи данных. Однако с ростом доли мультимедийной информации таких специализированных параметров оказывается недостаточно. Необходимо предпринимать серьезные попытки обеспечения гарантированного качества обслуживания сетей и улучшения протоколов. В следующих разделах мы продолжим изучение параметров производительности сетей, но больший упор сделаем на методах обеспечения высокого качества обслуживания, соответствующего требованиям конкретных приложений. Для начала следует отметить, что многие идеи пока еще не приобрели законченный вид и со временем могут измениться.

Требования

Последовательность пакетов, передающихся от источника к приемнику, называется **потоком**. При этом в сетях, ориентированных на соединение, все пакеты потока следуют по одному и тому же маршруту, а в сетях без установления соединения они могут идти разными путями. Каждому потоку требуются определенные условия, которые можно охарактеризовать следующими четырьмя основными параметрами: надежность, задержка, флуктуация и пропускная способность. Все вместе они формируют то, что называется **качеством обслуживания** (QoS — Quality of Service), необходимым потоку. Некоторые общие приложения особенно строго подходят к вопросу качества обслуживания, как показано в табл. 5.3.

Таблица 5.3. Строгие требования некоторых приложений к качеству обслуживания

Приложение	Надежность	Задержка	Флуктуации	Пропускная способность
Электронная почта	Высокая	Низкая	Слабые	Низкая
Передача файлов	Высокая	Низкая	Слабые	Средняя
Веб-доступ	Высокая	Средняя	Слабые	Средняя
Удаленный доступ	Высокая	Средняя	Средние	Низкая
Аудио по заказу	Низкая	Низкая	Сильные	Средняя
Видео по заказу	Низкая	Низкая	Сильные	Высокая
Телефония	Низкая	Высокая	Сильные	Низкая
Видеоконференции	Низкая	Высокая	Сильные	Высокая

Первые четыре приложения предъявляют высокие требования к надежности. Некорректная доставка битов должна быть исключена. Обычно это достигается подсчетом контрольной суммы для каждого пакета и ее проверкой у получателя.

Если пакет во время передачи был испорчен, подтверждение о его доставке не высылается, и источник вынужден передавать его повторно. Такая стратегия обеспечивает высокую надежность. Четыре последних (аудио/видео) приложения весьма толерантны к ошибкам, поэтому здесь нет никаких вычислений и проверок контрольных сумм.

Приложения, занимающиеся передачей файлов, включая электронную почту и видео, не чувствительны к задержкам. Даже если все пакеты будут доставляться с задержкой в несколько секунд, ничего страшного не произойдет. Однако интерактивные приложения — например, обеспечивающие веб-доступ или удаленный доступ, — к задержкам более критичны. Что касается приложений, работающих в реальном масштабе времени, их требования к задержкам очень строги. Если при телефонном разговоре все слова собеседников будут приходить с задержкой ровно 2,000 с, пользователи сочтут такую связь неприемлемой. С другой стороны, проигрывание видео- или аудиофайлов, хранящихся на сервере, допускает наличие некоторой задержки.

Первые три приложения спокойно отнесутся к неравномерной задержке доставки пакетов, а при организации удаленного доступа этот фактор имеет более важное значение, поскольку при сильных флуктуациях символы на экране будут появляться скачками. Видео- и особенно аудиоданные исключительно чувствительны к флуктуациям. Если пользователь просматривает видео, доставляемое на его компьютер по сети, и все кадры приходят с задержкой ровно 2,000 с, все нормально. Однако если время передачи колеблется от одной до двух секунд, то результат будет просто ужасен. При прослушивании звукозаписей будут заметны флуктуации даже в несколько миллисекунд.

Наконец, приложения могут иметь различные потребности в пропускной способности. Скажем, при передаче электронной почты или при удаленном доступе высокая пропускная способность не требуется, а вот для передачи видеоданных любых типов необходима высокая производительность сети.

В сетях ATM принята следующая классификация потоков по требованиям к качеству обслуживания:

1. Постоянная битовая скорость (например, телефония);
2. Переменная битовая скорость в реальном времени (например, сжатые видеоданные при проведении видеоконференций);
3. Переменная битовая скорость не в реальном времени (например, просмотр фильмов через Интернет);
4. Доступная битовая скорость (например, передача файлов).

Такое разбиение по категориям может оказаться полезным и для других целей, и для других сетей. Постоянная битовая скорость — это попытка моделирования проводной сети путем предоставления фиксированной пропускной способности и фиксированной задержки. Битовая скорость может быть переменной, например, при передаче сжатого видео, когда одни кадры удается сжать в большей степени, нежели другие. Кадр, содержащий множество разноцветных деталей, сожмется, скорее всего, плохо, и на его передачу придется потратить много битов, тогда как кадр, запечатлевший белую стену, сожмется очень хорошо.

Приложениям типа электронной почты нужно принципиальное наличие хоть какой-нибудь битовой скорости, они не чувствительны к задержкам и флуктуациям, поэтому говорят, что этим приложениям требуется «доступная битовая скорость».

Методы достижения хорошего качества обслуживания

Итак, мы узнали кое-что о требованиях, входящих в понятие «качество обслуживания». Как же заставить систему удовлетворять этим требованиям? Во-первых, необходимо учесть, что не существует никакой волшебной палочки. Ни один метод не обеспечивает безусловно высокое качество обслуживания. Напротив, разработано большое количество методов, практические реализации которых зачастую используют смешанные технологии. Далее мы рассмотрим некоторые методы, применяемые разработчиками систем для повышения качества обслуживания.

Избыточное обеспечение

Проще всего обеспечить такую емкость маршрутизаторов, буферной памяти и такую пропускную способность, при которых пакеты без затруднений пролетали бы по сети. Проблема здесь одна: такое решение обходится очень дорого. Со временем разработчики начинают понимать, какие параметры являются необходимыми и достаточными, и тогда такой подход оправдывает себя. Можно сказать, что телефонная сеть является системой с избыточным обеспечением. Довольно редко бывает, чтобы вы подняли трубку и не услышали гудка. Дело в том, что в систему заложена настолько большая пропускная способность, что превысить ее оказывается тяжело.

Буферизация

Потоки можно сохранять в буферной памяти на принимающей стороне перед тем, как доставлять потребителю. Буферизация не сказывается на надежности и пропускной способности, но сказывается на увеличении задержки. Зато с ее помощью можно снизить уровень флуктуаций. При передаче аудио и видео по требованию именно флуктуация представляет собой основную проблему, и буферизация помогает решить ее.

Мы уже видели различия характеристик сети при высокой и низкой флуктуации на рис. 5.26. На рис. 5.27 изображен поток пакетов, доставляемый при значительной флуктуации. Пакет 1 посылается с сервера в момент времени $t = 0$ с и прибывает в момент времени $t = 1$ с. Задержка пакета 2 составляет уже не 1, а 2 с. Прибывающие пакеты буферизируются на клиентской машине.

В момент времени $t = 10$ с начинается воспроизведение, при этом пакеты с 1-го по 6-й уже находятся в буфере, поэтому их можно оттуда извлекать через равные интервалы и воспроизводить. К сожалению, пакету 8 не повезло: он задержался настолько, что его невозможно воспроизвести вовремя. Выдача пакетов приостанавливается до его прибытия. Возникает досадная задержка в проигрывании му-

зыки или фильма. Проблему можно решить увеличением задержки начала выдачи пакетов, но для этого потребуется буфер большей емкости. Все коммерческие веб-сайты, на которых содержится потоковое видео или аудио, используют проигрыватели, которые начинают воспроизведение только после примерно десятисекундной буферизации.



Рис. 5.27. Сглаживание выходного потока путем буферизации пакетов

Формирование трафика

В приведенном ранее примере источник выдает пакеты через фиксированные интервалы времени, однако бывает, что этот процесс не является столь равномерным. Это может приводить к перегрузке сети. Неравномерный выходной поток — это обычное дело для серверов, поддерживающих множество потоков и различные виды действий, таких как быстрая прокрутка вперед и назад, идентификация пользователя и т. д. Подход, описанный ранее (буферизация), не всегда можно применить (например, при видеоконференциях). Тем не менее, если бы удалось заставить серверы (и хосты в целом) передавать данные с предсказуемой скоростью, качество обслуживания было бы лучше. Рассмотрим метод **формирования трафика**, сглаживающий выходной трафик на стороне сервера, а не на стороне клиента.

При формировании трафика происходит регулирование средней и пиковой *скорости* передачи данных. Изучавшиеся нами ранее протоколы скользящего окна ограничивают количество данных, посылаемых сразу, но не скорость, с которой они посылаются. Когда устанавливается виртуальный канал, пользователь и подсеть (то есть клиент и оператор связи) договариваются об определенной схеме (то есть форме) трафика для данного канала. Иногда это действие называется **соглашением об уровне обслуживания**. До тех пор пока клиент выполняет свою часть условий соглашения и посылает пакеты не чаще оговоренного в договоре графика, оператор связи обязуется доставлять их в определенный срок. Формирование трафика снижает перегрузку и, таким образом, помогает оператору связи выполнять свои обязательства. Подобные договоренности не столь важны при передаче файлов, но весьма существенны при передаче данных в режиме реального времени, как, например, для аудио- и видеосвязи, которые плохо переносят перегрузку.

В результате при использовании метода формирования трафика клиент сообщает оператору связи: «Мой график передачи будет выглядеть следующим образом. Сможете ли вы это обеспечить?». Если оператор связи соглашается, то возникает вопрос о том, как оператор связи будет сообщать клиенту, что тот соблюдает соглашение, и что делать, если клиент нарушит договор. Наблюдение за потоком трафика называется **политикой трафика**. Договориться о форме трафика и регулировать его впоследствии легче в подсетях с виртуальными каналами, чем в дейтаграммных подсетях. Тем не менее даже в дейтаграммных подсетях можно применить те же идеи к соединениям транспортного уровня.

Алгоритм дырявого ведра

Представьте себе ведро с маленькой дырочкой в днище, как показано на рис. 5.28, *а*. Независимо от скорости, с которой вода наливается в ведро, выходной поток обладает постоянной скоростью, когда в ведре есть вода, и нулевой скоростью, когда ведро пустое. Кроме того, когда ведро наполняется, вся лишняя вода выливается через край и теряется (то есть не попадает в выходной поток под дырочкой).

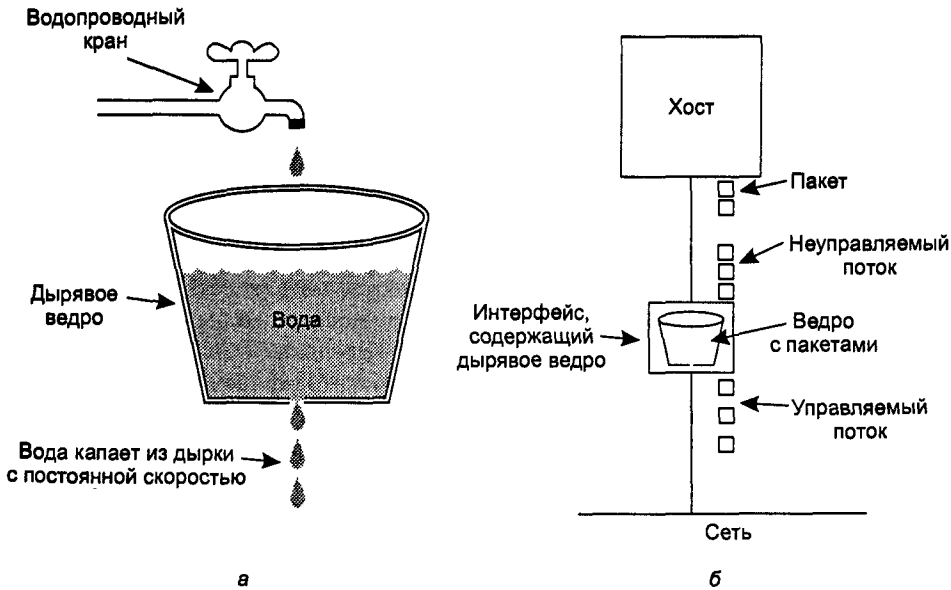


Рис. 5.28. Дырявое ведро с водой (*а*); дырявое ведро с пакетами (*б*)

Та же самая идея применима к пакетам, как показано на рис. 5.28, *б*. Принцип таков: каждый хост соединяется с сетью через интерфейс, содержащий дырявое ведро, то есть конечную внутреннюю очередь. Если пакет появляется в очереди, когда очередь полная, пакет игнорируется. Другими словами, если несколько процессов хоста пытаются послать пакеты, когда в очереди уже стоит максимально допустимое число пакетов, новый пакет игнорируется. Такой интерфейс может быть реализован как аппаратно, так и программно операционной систе-

мой хоста. Он был предложен Тернером (Turner, 1986) и называется **алгоритмом дырявого ведра**. По сути это не что иное, как однолинейная система массового обслуживания с постоянным временем обслуживания.

Хосту разрешается посылать в сеть один пакет за один такт. Опять же, это может быть реализовано интерфейсной картой либо операционной системой. Этот механизм преобразует неравномерный поток пакетов от процессов пользователя в равномерный поток пакетов в сети, сглаживая пики и значительно снижая вероятность перегрузки.

Когда размер всех пакетов одинаков (например, в ячейках ATM), этот алгоритм может применяться, как описано ранее. Однако при использовании пакетов переменного размера часто бывает лучше ограничивать количество байтов, переданных в сеть за такт, нежели передавать один пакет за такт. Так, если правилом установлена передача 1024 байт за тактовый интервал, то за этот период могут быть переданы в сеть либо один пакет размером 1024 байта, либо два пакета по 512 байт, либо четыре пакета по 256 байт и т. д. Если оставшееся количество байт меньше размера следующего пакета, следующий пакет должен ждать начала следующего такта.

Реализация исходного алгоритма дырявого ведра проста. Дырявое ведро состоит из конечной очереди. Когда прибывает пакет и в очереди есть место, пакет добавляется к очереди, в противном случае пакет игнорируется. Если очередь не пуста, то в течение каждого тактового интервала в сеть передается по одному пакету.

Алгоритм дырявого ведра со счетчиком байтов реализуется почти также. В каждом тактовом интервале значение счетчика устанавливается равным n. Если размер первого пакета в очереди меньше текущего значения счетчика, он передается, а значение счетчика уменьшается на его размер. Если значение счетчика еще достаточно велико, могут быть посланы и другие пакеты. Когда значение счетчика становится меньше размера следующего пакета в очереди, передача прекращается до следующего такта, после чего все начинается сначала, а остаток счетчика обнуляется.

В качестве примера представьте, что компьютер может производить данные со скоростью 25 млн байт в секунду (200 Мбит/с) и что сеть также работает на этой скорости. Однако маршрутизаторы могут поддерживать эту скорость передачи данных лишь на коротких интервалах (пока не заполнится их буферная память). В течение больших интервалов времени они могут обеспечить не более 2 млн байт в секунду. Теперь предположим, что данные поступают пачками по 1 млн байт, одна пачка продолжительностью 40 мс в каждую секунду. Чтобы уменьшить среднюю скорость до 2 Мбайт/с, можно воспользоваться алгоритмом дырявого ведра с выходной скоростью $\rho = 2$ Мбайт/с и емкостью $C = 1$ Мбайт. Это означает, что пачки до 1 Мбайта могут обрабатываться без потерь и что такие пачки будут передаваться в сеть за 500 мс независимо от того, как быстро они приходят.

На рис. 5.29, *а* показан вход дырявого ведра, на который со скоростью 25 Мбайт/с поступает пачка в течение 40 мс. На рис. 5.29, *б* показан выход, через который данные проходят с постоянной скоростью 2 Мбайт/с в течение 500 мс.

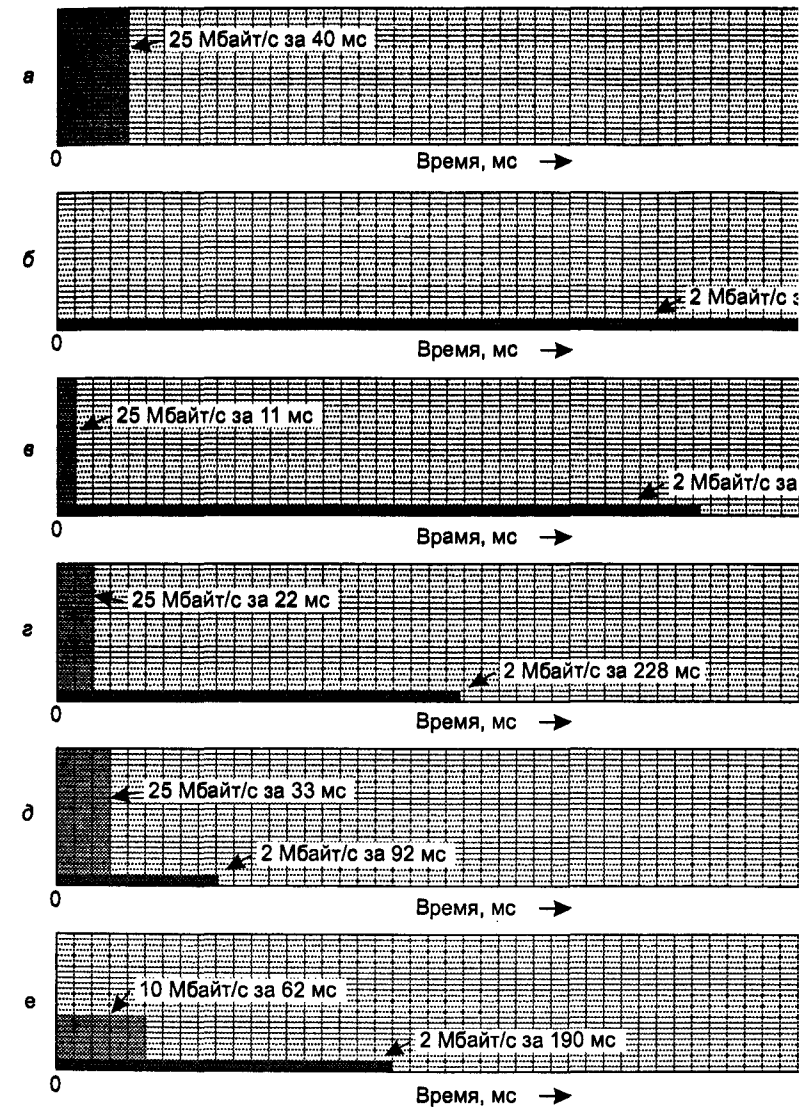


Рис. 5.29. Вход дырявого ведра (*а*); выход дырявого ведра (*б*); выход мар емкостью 250 Кбайт (*в*), 500 Кбайт (*г*) и 750 Кбайт (*д*); выход маркерного вед 500 Кбайт на входе дырявого ведра со скоростью протекания 10 Мба

Алгоритм маркерного ведра

Алгоритм дырявого ведра формирует строгий выходной поток с по ростью, не зависящей от неравномерности входного потока. Для жений было бы лучше при поступлении больших пакетов данных чивать выходную скорость. Таким образом можно было бы попы более гибкий алгоритм, желательно, не теряющий данные. Одним

ритмов является **алгоритм маркерного ведра**. В этом алгоритме ведро содержит маркеры, создаваемые через равные интервалы времени ΔT секунд. На рис. 5.30, *а* изображено ведро с тремя маркерами и пятью пакетами, стоящими в очереди. Чтобы передать один пакет, требуется удалить один маркер. На рис. 5.30, *б* мы видим, что три из пяти пакетов прошли в сеть, а оставшиеся два пакета остались ждать двух новых маркеров.

Алгоритм маркерного ведра формирует трафик не так, как алгоритм дырявого ведра. Алгоритм дырявого ведра не позволяет простаивающим хостам запасаться впрок разрешениями на передачу больших пакетов. Алгоритм маркерного ведра разрешает запасаться маркерами до определенного размера ведра *n*. Это свойство означает, что пачки (пакеты) с величиной до *n* могут быть переданы в сеть сразу, что создает некоторую неравномерность в выходном потоке, но обеспечивает быструю реакцию на неожиданные входные пачки.

Еще одно различие двух алгоритмов заключается в том, что при переполнении маркерного ведра алгоритм игнорирует маркеры, но никогда не отвергает пакеты. Алгоритм дырявого ведра, напротив, при переполнении выбрасывает сами пакеты.

Возможен вариант алгоритма, при котором маркер может предоставлять право пересылать не один пакет, а *k* байт. Пакет пересылается только при наличии достаточного числа маркеров, чтобы покрыть его длину. Лишние маркеры сохраняются для будущего использования.

Алгоритмы дырявого и маркерного ведра могут использоваться не только для регулирования выхода хостов, но и для сглаживания трафика между маршрутизаторами. А различаются эти два алгоритма тем, что применение алгоритма маркерного ведра может заставить маршрутизатор остановить передачу пакетов, что может привести к потере данных.

Реализация исходного алгоритма маркерного ведра подразумевает наличие переменной, считающей маркеры. Счетчик увеличивается на единицу через равные интервалы времени ΔT и уменьшается при посылке пакета. Когда счетчик уменьшается до нуля, передача пакетов останавливается. В варианте с учетом количества переданных байт счетчик увеличивается на *k* байт каждые ΔT секунд и уменьшается на размер переданного пакета.

Суть алгоритма маркерного ведра состоит в том, что он допускает передачу данных пачками, но ограничивает длительность пачки. Для примера рассмотрим рис. 5.29, *в*, на котором изображено маркерное ведро емкостью 250 Кбайт. Маркеры появляются с частотой, соответствующей выходной скорости 2 Мбайт/с. Предположим, что маркерное ведро заполнено, когда прибывает пакет данных размером 1 Мбайт. Ведро может быть освобождено с максимальной скоростью 25 Мбайт/с примерно за 11 мс. Затем оно должно уменьшить скорость передачи до 2 Мбайт/с, пока не будет передан весь входной пакет данных.

При расчете длительности выходной пачки (на максимальной скорости) нужно учитывать, что пока ведро опорожняется, появляются новые маркеры. При длительности пачки *S* секунд, емкости маркерного ведра *C* байт, скорости появления маркеров ρ байт/с, и максимальной выходной скорости *M* байт/с очевидно, что максимальное количество переданных байтов в пачке будет равно $C + \rho S$ байт.

Мы также знаем, что количество байтов, переданных в пачке с максимальной скоростью, равно *MS*. Таким образом,

$$C + \rho S = MS.$$

Решив это уравнение, получим: $S = C/(M - \rho)$. При наших параметрах $C = 250$ Кбайт, $M = 25$ Мбайт/с и $\rho = 2$ Мбайт/с получаем длительность пачки около 11 мс. На рис. 5.29, *г* и *д*, показаны маркерные ведра емкостью 500 и 750 Кбайт соответственно.

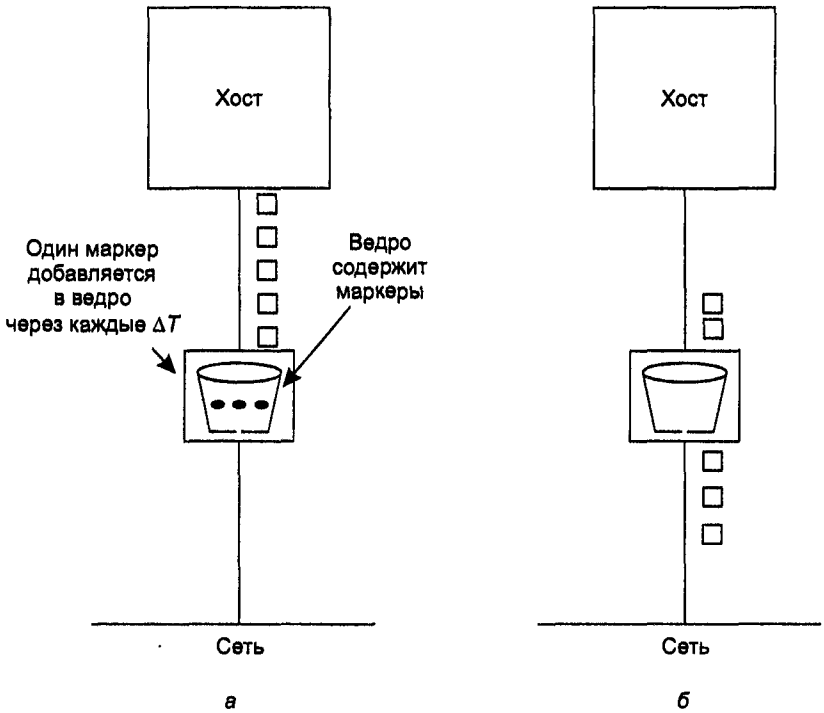


Рис. 5.30. Алгоритм маркерного ведра: до (*а*) и после (*б*)

Недостатком алгоритма маркерного ведра является слишком большая скорость передачи данных при опустошении ведра, несмотря на то что длительность пачки можно регулировать тщательным подбором ρ и M. Часто бывает желательно уменьшить пиковую скорость, не возвращаясь при этом к скорости алгоритма дырявого ведра.

Один из способов получения более гладкого трафика состоит в помещении дырявого ведра после маркерного ведра. Скорость дырявого ведра должна быть выше минимальной скорости маркерного ведра ρ, но ниже максимальной скорости сети. На рис. 5.29, *е* показан выходной поток маркерного ведра емкостью 500 Кбайт, за которым установлено дырявое ведро со скоростью протекания, равной 10 Мбайт/с.

Управление такими схемами может оказаться непростым. По сути, сеть должна имитировать алгоритм и гарантировать, что пакетов и байтов посылается не больше, чем разрешено. Тем не менее, эти методы позволяют формировать сетевой трафик, приводя его к более управляемому виду и обеспечивая тем самым выполнение требований к качеству обслуживания.

Резервирование ресурсов

Возможность управления трафиком — это неплохой начальный шаг в деле обеспечения гарантированного качества обслуживания. Однако на самом деле использование этих методов неявно означает, что все пакеты в потоке должны следовать по одному и тому же пути. При распределении их случайным образом между несколькими маршрутизаторами невозможно что-либо гарантировать. Следовательно, между источником и приемником должно быть установлено нечто вроде виртуального канала, и все пакеты, принадлежащие данному потоку, должны следовать по указанному маршруту.

Раз у нас есть особый путь, по которому направляется поток, становится возможным резервирование ресурсов вдоль этого пути, что позволяет гарантировать доступность необходимой емкости. Резервироваться могут три типа ресурсов:

1. Пропускная способность.
2. Буферное пространство.
3. Время центрального процессора.

Наиболее очевидно резервирование пропускной способности. Если потоку необходима скорость 1 Мбит/с, а исходящая линия может работать со скоростью 2 Мбит/с, то направить три потока с такими параметрами по этой линии не удастся. То есть резервирование пропускной способности означает предотвращение предоставления канала большему числу абонентов, чем канал может обработать.

Вторым дефицитным ресурсом является буферное пространство. Когда прибывает пакет, он обычно оседает на сетевой интерфейсной карте (это действие управляется аппаратно). Затем программному обеспечению маршрутизатора необходимо скопировать пакет в буфер оперативной памяти и поставить содержимое этого буфера в очередь на отправку по выбранной исходящей линии. Если буферное пространство недоступно, входящий пакет приходится игнорировать, поскольку его просто негде сохранить. Для обеспечения хорошего качества обслуживания можно резервировать некоторую часть буферной памяти под конкретный поток, чтобы ему не пришлось бороться за буфер с другими потоками. И тогда при передаче потока ему всегда будет предоставляться выделенная часть буфера, вплоть до некоторого максимума.

Наконец, время центрального процесса — это еще один очень ценный ресурс. На что расходуется время работы процессора в маршрутизаторе? На обработку пакетов. Поэтому существует предельная скорость, с которой маршрутизатор может обрабатывать пакеты. Необходимо быть уверенным в том, что процессор не перегружен, — это залог своевременной обработки каждого пакета.

На первый взгляд кажется, что если на обработку пакета уходит, скажем, 1 мкс, то маршрутизатор способен управиться с миллионом пакетов за секунду. Однако это предположение ошибочно, так как при доставке потока всегда есть промежутки времени, в течение которых ничего не передается. Если центральному процессору для совершения своей работы важен каждый отдельный такт, то пропуск нескольких тактов из-за молчания на линии приведет к накоплению невыполненных заказов, от которых невозможно избавиться.

Однако даже если нагрузка несколько меньше теоретической емкости, все равно могут образовываться очереди и возникать задержки. Рассмотрим ситуацию, когда пакеты прибывают нерегулярно со средней скоростью прибытия λ пакетов в секунду. Время, необходимое процессору на обработку каждого пакета, также меняется, но в среднем составляет μ пакетов в секунду. Предположим, что как скорость прибытия, так и скорость обслуживания имеют пуассоновское распределение. Тогда, используя теорию массового обслуживания, можно доказать, что средняя задержка T, присущая пакету, составляет

$$T = \frac{1}{\mu} \times \frac{1}{1-\lambda/\mu} = \frac{1}{\mu} \times \frac{1}{1-\rho}$$

где $\rho = \lambda/\mu$ — коэффициент использования центрального процессора. Первый сомножитель $1/\mu$ — это задержка при отсутствии конкуренции. Второй сомножитель представляет собой дополнительную задержку, возникающую в результате конкурентной борьбы с другими потоками. Например, если $\lambda = 950\,000$ пакетов/с, а $\mu = 1\,000\,000$ пакетов/с, тогда $\rho = 0{,}95$, и средняя задержка каждого пакета составляет 20 мкс вместо 1 мкс. Эти подсчеты учитывают и задержку доставки, и задержку обработки: при малом трафике отношение $\lambda/\mu \approx 0$. Если на пути потока стоят, скажем, 30 маршрутизаторов, то одна только задержка обслуживания составит 600 мкс.

Управление доступом

Итак, в результате проведенной работы мы получили входящий трафик в виде хорошо сформированного и, возможно, следующего по единому маршруту потока. На пути потока можно заранее резервировать ресурсы. Когда маршрутизатору предлагается обработать такой поток, он может принять или отвергнуть его, обосновывая свое решение доступной емкостью и количеством уже находящихся в обработке потоков.

Процесс принятия решения об обработке или игнорировании потока сложнее, нежели простое сравнение запрашиваемых потоком параметров (пропускной способности, буферной памяти, времени центрального процессора) с имеющимися. Во-первых, хотя многие приложения и знают свои требования к пропускной способности, они понятия не имеют, какой объем буферной памяти и сколько тактов работы процессора им требуется. Следовательно, нужен, по крайней мере, иной способ описания потоков. Далее, приложения весьма различаются по толерантности в отношении пропущенного предельного срока обработки. Наконец, некоторые приложения могут поторговаться за параметры пакетов, а некоторые не могут. Скажем, проигрыватель видео, предоставляющий обычно 30 кадров/с,

может согласиться работать на 25 кадрах/с, если для 30 не хватает пропускной способности. Аналогично, можно настраивать количество пикселов на кадр, полосу пропускания для аудиоданных и другие свойства потоков различных приложений.

Поскольку в спор по поводу того, что делать с потоком, вовлечено много сторон (отправитель, приемник и все маршрутизаторы на пути между ними), поток необходимо описывать крайне аккуратно с помощью параметров, о которых можно дискутировать. Набор таких параметров называется **спецификацией потока**. В типичном случае отправитель (например, сервер видеоданных) создает спецификацию потока, указывая параметры, которые он хотел бы использовать для аргументации. По мере того как эта спецификация распространяется по пути следования потока, содержащаяся в нем информация анализируется всеми маршрутизаторами, которые модифицируют параметры так, как считают нужным. Эти модификации могут быть направлены только на снижение трафика — никто не станет сознательно брать на себя больше работы, чем требует заказчик (например, указываемая в спецификации скорость передачи данных может быть понижена, но не повышена). Когда спецификация доходит до приемника, становятся понятны окончательные параметры.

В качестве содержимого спецификации потока рассмотрим пример, базирующийся на RFC 2210 и RFC 2211 (табл. 5.4). В спецификации содержится пять параметров, первый из которых, *Скорость маркерного ведра*, хранит число байтов, поступающих в «ведро» за секунду. Это максимум, который отправитель может поддерживать в течение длительного времени, усредненный по большому временному отрезку.

Таблица 5.4. Пример спецификации потока

Параметр	Единицы измерения
Скорость маркерного ведра	байт/с
Размер маркерного ведра	байт
Пиковая скорость передачи данных	байт/с
Минимальный размер пакета	байт
Максимальный размер пакета	байт

Второй параметр — размер маркерного ведра в байтах. Если, к примеру, *Скорость маркерного ведра* составляет 1 Мбит/с, а размер ведра равен 500 Кбайт, то его можно будет наполнять данными в течение 4 с. Все, что будет посылаться после этого, будет теряться.

Третий параметр, *Пиковая скорость передачи данных*, — это максимальная допустимая скорость даже для коротких промежутков времени. Отправитель ни в коем случае не должен превышать это значение.

Наконец, последние два параметра определяют минимальный и максимальный размеры пакетов, включая заголовки транспортного и сетевого уровней (например, TCP и IP). Минимальный размер важен, поскольку обработка каждого пакета занимает какое-то, пусть даже очень малое, время. Маршрутизатор, может быть, готов принимать 10 000 пакетов в секунду по 1 Кбайт каждый, но не

готов обрабатывать 100 000 пакетов по 50 байт в секунду несмотря на то, что во втором случае скорость передачи данных меньше, чем в первом. Максимальный размер пакета не менее важен, но уже по другой причине. Дело в том, что существуют определенные внутрисетевые ограничения, которые ни в коем случае не должны быть превышены. Например, если путь потока лежит через Ethernet, то максимальный размер пакета будет ограничен 1500 байтами независимо от того, какого размера пакеты могут поддерживать другие части сети.

Интересно, каким образом маршрутизатор преобразует спецификацию потока в набор определенных резервируемых ресурсов? Это отображение является специфическим и не стандартизованным действием. Допустим, маршрутизатор может обрабатывать 100 000 пакетов/с. Если ему предлагается пропустить через себя поток со скоростью 1 Мбайт/с с максимальным размером пакета, составляющим 512 байт, он может легко посчитать, что такой поток дает 2048 пакетов/с, значит, под него необходимо отвести 2 % времени работы процессора, а лучше немного больше, чтобы избежать больших задержек обслуживания. Если политика маршрутизатора не позволяет ему резервировать более 50 % процессорного времени (что подразумевает половинную задержку) и если 49 % уже зарезервировано, то поток будет отвергнут. Подобные вычисления необходимо производить для всех резервируемых ресурсов.

Чем строже спецификация потока, тем лучше для маршрутизаторов. Если же в спецификации говорится, что *Скорость маркерного ведра* составляет 5 Мбайт/с, однако пакеты могут быть размером от 50 до 1500 байт, значит, скорость передачи пакетов может колебаться от 3500 до 105 000 пакетов/с. Маршрутизатор, ужаснувшись при виде последнего числа, может отвергнуть такой поток. При минимальном размере пакета, равном 1000 байт, 5-мегабайтный поток тем же самым маршрутизатором мог бы быть принят.

Пропорциональная маршрутизация

Большинство алгоритмов маршрутизации пытаются искать наилучшие пути для каждого адресата и направлять весь трафик по оптимальному пути. Альтернативный подход, позволяющий повысить качество обслуживания, состоит в разделении трафика для одного и того же адресата между несколькими маршрутами. Поскольку маршрутизаторы обычно не следят за нагрузкой на всю сеть в целом, остается лишь один способ разделения трафика — на основе доступной локальной информации. Одним из простых методов является маршрутизация, пропорциональная или эквивалентная емкостям исходящих связей. Однако существуют и более сложные алгоритмы (Nelakuditi и Zhang, 2002).

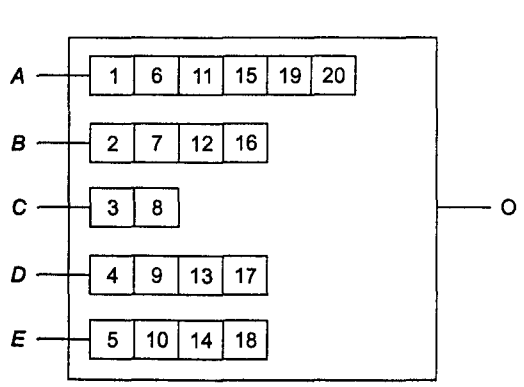
Диспетчеризация пакетов

Если маршрутизатор имеет поддержку нескольких потоков, существует опасность того, что один из них захватит слишком большую часть пропускной способности и не даст жить всем остальным потокам. Обработка пакетов в порядке поступления может привести к тому, что агрессивный источник загрузит все производственные мощности маршрутизаторов, через которые проходит его поток, и тем самым снизит качество обслуживания других источников. Для пресечения

подобных попыток были разработаны алгоритмы диспетчеризации пакетов (Bhatti и Crowcroft, 2000).

Одним из первых был алгоритм **справедливого обслуживания** (Nagle, 1987). Суть его состоит в том, что маршрутизаторы организуют отдельные очереди для каждой исходящей линии, по одной для каждого потока. Как только линия освобождается, маршрутизатор начинает циклически сканировать очереди, выбирая первый пакет следующей очереди. Таким образом, если за данную исходящую линию борются n хостов, то каждый из них имеет возможность отправить свой пакет, пропустив $n - 1$ чужих пакетов. Агрессивному хосту не поможет то, что в его очереди стоит больше пакетов, чем у остальных.

Однако и с этим алгоритмом связана одна проблема: предоставляемая им пропускная способность напрямую зависит от размера пакета, используемого хостом: большая часть предоставляется хостам с большими пакетами, и меньшая — хостам с небольшими пакетами. В книге (Demers и др., 1990) предлагается улучшенная версия, в которой циклический опрос производится с целью выхватывания не пакета, а байта. То есть очереди сканируются побайтно до того момента, пока не будет выхвачен последний байт последнего пакета. После этого пакеты отправляются в том порядке, в котором они заканчивались при опросе очередей. Алгоритм проиллюстрирован на рис. 5.31.



Пакет	Время финиширования
C	8
B	16
D	17
E	18
A	20

а *б*

Рис. 5.31. Маршрутизатор с пятью очередями пакетов для линии O (а); время окончания сканирования для пяти пакетов (*б*)

На рис. 5.31, *а* мы видим пакеты длиной от 2 до 6 байт. Во время (виртуального) первого такта извлекается первый байт пакета с линии *A*. Затем следует первый байт пакета с линии *B*, и т. д. Первым, через 8 тактов, закончится пакет *C*. Порядок сортировки пакетов приведен на рисунке справа (рис. 5.31, *б*). При отсутствии новых поступлений пакеты будут отсылаться в указанном порядке, начиная с *C* и заканчивая *A*.

Проблема данного алгоритма заключается в том, что он дает всем хостам одинаковые приоритеты. Во многих случаях желательно предоставлять, например,

видеосерверам большую пропускную способность, чем обычным файл-серверам, чтобы они могли посылать два или более байт за такт опроса. Такая модификация алгоритма называется **взвешенным справедливым обслуживанием**. Иногда весовой коэффициент эквивалентен числу потоков, генерируемых машиной, таким образом все процессы получают равные доли пропускной способности. Одна из эффективных реализаций данного алгоритма описывается в (Shreedhar и Varghese, 1995). Все чаще и чаще встречаются аппаратные реализации передачи пакетов через маршрутизаторы или коммутаторы (Elhanany и др., 2001).

Интегральное обслуживание

В 1995–1997 годы проблемная группа проектирования сети Интернет (IETF) прилагала множество усилий по продвижению архитектуры потокового мультимедиа. В результате появилось две дюжины документов RFC, начинающихся с префикса RFC и включающих порядковые номера 2205–2210. Общее название этих трудов — **потоковые алгоритмы** или **интегральное обслуживание**. Предлагаемая технология предназначена как для одноадресных, так и для многоадресных приложений. Примером первых может быть видеоклип на сайте новостей, доставляемый в виде потока пользователю, пожелавшему посмотреть его. Пример вторых — набор станций цифрового телевидения, осуществляющих широковещательное распространение своих программ в виде потоков IP-пакетов. Данной услугой может пользоваться большое число абонентов, расположенных в разных географических точках. Далее мы более подробно рассмотрим многоадресную рассылку, поскольку однонадресная передача — это лишь особый случай многоадресной. Во многих приложениях с многоадресной маршрутизацией группы пользователей могут меняться динамически. Например, люди могут подключаться к участию в видеоконференциях, но со временем это занятие им надоедает, и они переключаются на мыльные оперы или спортивные каналы. В данном случае стратегия предварительного резервирования пропускной способности не совсем подходит, потому что при этом каждому источнику пришлось бы запоминать все изменения в составе аудитории. В системах, предназначенных для передачи телевизионного сигнала миллионам абонентов, такой подход и вовсе не годится.

RSVP — протокол резервирования ресурсов

Главный протокол архитектуры интегрального обслуживания, разработанный IETF, называется **протоколом резервирования ресурсов** (RSVP — Resource reSerVation Protocol). Он описывается в документе RFC 2205 и других документах-протоколах. Как следует из названия, протокол предназначен для резервирования ресурсов; другие протоколы применяются для описания передачи данных. RSVP позволяет нескольким отправителям посылать данные нескольким группам абонентов, разрешает отдельным получателям переключать каналы и оптимизирует использование пропускной способности, в то же время устраняя возникновение перегрузки.

Простейшая форма этого протокола использует многоадресную маршрутизацию с применением связующих деревьев, обсуждавшуюся ранее. Каждой группе назначается групповой адрес. Чтобы послать данные группе, отправитель помещает ее адрес в заголовки пакетов. Затем стандартный алгоритм многоадресной маршрутизации строит связующее дерево, покрывающее всех членов группы. Алгоритм маршрутизации не является частью протокола RSVP. Единственное отличие от нормальной многоадресной маршрутизации состоит в том, что группе периодически рассылается дополнительная информация, с помощью которой маршрутизаторы обновляют определенные структуры данных.

Для примера рассмотрим сеть, показанную на рис. 5.32, *а*. Хосты 1 и 2 являются многоадресными передатчиками, а хосты 3, 4 и 5 — многоадресными приемниками. В данном примере передатчики и приемники разделены, однако в общем случае эти два множества могут перекрываться. Деревья многоадресной рассылки для хостов 1 и 2 показаны на рис. 5.32, *б* и *в* соответственно.

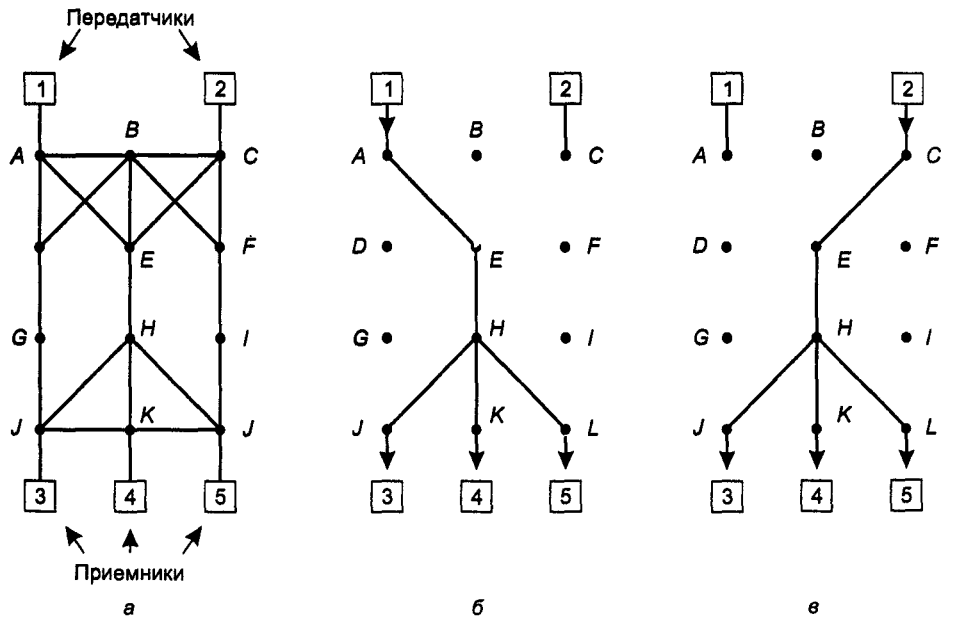


Рис. 5.32. Сеть (*а*); связующее дерево многоадресной рассылки для хоста 1 (*б*); связующее дерево многоадресной рассылки для хоста 2 (*в*)

Для улучшения качества приема и устранения перегрузки каждый получатель в группе может послать передатчику (вверх по дереву) запрос на резервирование. Запрос продвигается, используя обсуждавшийся ранее алгоритм обратного пути. На каждом транзитном участке маршрутизатор замечает запрос и резервирует необходимую пропускную способность. Если пропускной способности недостаточно, он отвечает сообщением об ошибке. К тому моменту как запрос доходит до передатчика, пропускная способность зарезервирована вдоль всего пути от отправителя к получателю.

Пример резервирования показан на рис. 5.33, *а*. Здесь хост 3 запросил канал к хосту 1. После создания канала поток пакетов от хоста 1 к хосту 3 может течь, не боясь попасть в затор. Рассмотрим, что произойдет, если теперь хост 3 зарезервирует канал к другому передатчику, хосту 2, чтобы пользователь мог одновременно смотреть две телевизионные программы. Зарезервированный второй канал показан на рис. 5.33, *б*. Обратите внимание: между хостом 3 и маршрутизатором *E* требуется наличие двух отдельных каналов, так как передаются два независимых потока.

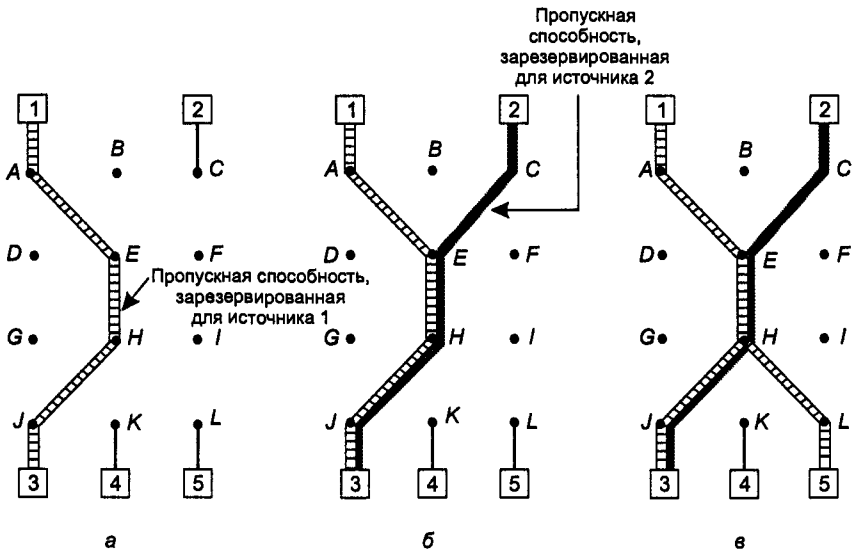


Рис. 5.33. Хост 3 запрашивает канал к хосту 1 (а); затем хост 3 запрашивает второй канал к хосту 2 (*б*); хост 5 запрашивает канал к хосту 1 (*в*)

Наконец, на рис. 5.33, *в* хост 5 решает посмотреть программу, передаваемую хостом 1, и также резервирует себе канал. Сначала требуемая пропускная способность резервируется до маршрутизатора *H*. Затем этот маршрутизатор замечает, что у него уже есть канал от хоста 1, поэтому дополнительное резервирование выше по дереву не требуется. Обратите внимание на то, что хосты 3 и 5 могут запросить различную пропускную способность (например, у хоста 3 черно-белый телевизор, поэтому ему не нужна информация о цвете), следовательно, маршрутизатор *H* должен зарезервировать пропускную способность, достаточную для удовлетворения самого жадного получателя.

При подаче запроса на резервирование получатель может (по желанию) указать один или несколько источников, от которых он хотел бы получать сигнал. Он также может указать, является ли выбор источников фиксированным в течение времени резервирования или он оставляет за собой право сменить источники. Данные сведения используются маршрутизаторами для оптимизации планирования пропускной способности. В частности, двум приемникам выделяется общий путь, только если они соглашаются не менять впоследствии свой источник.

В основе такой динамической стратегии лежит полная независимость зарезервированной пропускной способности от выбора источника. Получив зарезервированную пропускную способность, получатель может переключаться с одного источника на другой, сохраняя часть существующего пути, годящуюся для нового источника. Например, если хост 2 передает несколько видеопотоков, хост 3 может переключаться между ними по желанию, не меняя своих параметров резервирования: маршрутизаторам все равно, какую программу смотрит получатель.

Дифференцированное обслуживание

Потоковые алгоритмы способны обеспечивать хорошее качество обслуживания одного или нескольких потоков за счет резервирования любых необходимых ресурсов на протяжении всего маршрута. Однако есть у них и недостаток. Им требуется предварительная договоренность при установке канала для каждого потока. Это не позволяет в достаточной мере расширять систему, в которой применяются данные алгоритмы. Скажем, в системах с тысячами или миллионами потоков интегральное обслуживание применить не удастся. Кроме того, потоковые алгоритмы работают с внутренней информацией о каждом потоке, хранящейся в маршрутизаторах, что делает их уязвимыми к выходу из строя маршрутизаторов. Наконец, программные изменения, которые нужно производить в маршрутизаторах, довольно значительны и связаны со сложными процессами обмена между маршрутизаторами при установке потоков. В результате выжило крайне мало реализаций RSVP и чего-либо подобного.

По этим причинам IETF был создан упрощенный подход к повышению качества обслуживания. Его можно реализовать локально в каждом маршрутизаторе без предварительной настройки и без включения в процесс всех устройств вдоль маршрута. Подход известен как **ориентированное на классы** (в отличие от ориентированного на потоки) качество обслуживания. Проблемной группой IETF была стандартизована специальная архитектура под названием **дифференцированное обслуживание**, описываемая в документах RFC 2474, RFC 2475 и во многих других. Далее мы опишем ее.

Дифференцированное обслуживание (ДО) может предоставляться набором маршрутизаторов, образующих административный домен (например, интернет-провайдер или телефонную компанию). Администрация определяет множество классов обслуживания и соответствующие правила маршрутизации. Пакеты, приходящие от абонента, пользующегося ДО, содержат поле *Тип обслуживания*, и в зависимости от этого значения некоторым классам предоставляется улучшенный сервис по сравнению с остальными. К трафику класса могут предъявляться определенные требования, касающиеся его формы. Например, от него может потребоваться, чтобы он представлял собой «дырявое ведро» с определенной скоростью просачивания данных через «дырочку». Оператор, привыкший брать деньги за все, может взимать дополнительную плату за каждый пакет, обслуживаемый по высшему классу, либо может установить абонентскую плату за передачу N таких пакетов в месяц. Обратите внимания: здесь нет никакой предварительной

настройки, резервирования ресурсов и трудоемких согласований параметров для каждого потока, как при интегральном обслуживании. Это делает ДО относительно просто реализуемым.

Обслуживание, ориентированное на классы, возникает и в других областях. Например, службы доставки посылок могут предлагать на выбор несколько уровней обслуживания: доставка на следующий день, через день или через два дня. В самолетах обычно бывают первый класс, бизнесс-класс и второй класс. То же самое касается поездов дальнего следования. Даже в парижской подземке до недавних пор были вагоны двух классов. Что касается нашей тематики, то классы пакетов могут отличаться друг от друга задержками, флуктуациями времени доставки, вероятностью быть проигнорированными в случае коллизии, а также другими параметрами (коих, впрочем, не больше, чем у кадров Ethernet).

Чтобы разница между обслуживанием, ориентированным на классы, и обслуживанием, ориентированным на потоки, стала яснее, рассмотрим пример: интернет-телефонию. При потоковом алгоритме обслуживания каждому телефонному соединению предоставляются собственные ресурсы и гарантии. При классовом обслуживании все телефонные соединения совместно получают ресурсы, зарезервированные для телефонии данного класса. Эти ресурсы, с одной стороны, не может отнять никто извне (соединения других классов, потоки систем передачи файлов и т. п.), с другой стороны, ни одно телефонное соединение не может получить никакие ресурсы в частное пользование.

Срочная пересылка

Выбор классов обслуживания зависит от решения оператора, однако поскольку пакеты зачастую необходимо пересылать между подсетями, управляемыми разными операторами, проблемная группа IETF разрабатывает классы обслуживания, не зависящие от подсети. Простейший из них — класс **срочной пересылки**, с него и начнем. Он описывается документом RFC 3246.

Итак, идея, на которой построена срочная пересылка, очень проста. Существует два класса обслуживания: обычный и срочный. Ожидается, что подавляющая часть объема трафика будет использовать обычный класс обслуживания. Однако есть небольшая доля пакетов, которые необходимо передавать в срочном порядке. Их нужно пересылать между подсетями так, будто кроме них в сети больше нет вообще никаких пакетов. Графическое представление такой двухканальной системы показано на рис. 5.34. Имейте в виду, что физическая линия здесь только одна. Два логических пути — это своеобразный способ резервирования пропускной способности, а вовсе не протягивание второго провода для передачи данных рядом с основным.

Данную стратегию можно реализовать, запрограммировав маршрутизаторы таким образом, чтобы они формировали на выходе две очереди — для обычных и для срочных пакетов. Прибывший пакет ставится в очередь, соответствующую его классу обслуживания. Составление графика передачи пакетов должно опираться на алгоритм, подобный взвешенному справедливому обслуживанию. Например, если срочный трафик составляет 10 % общего объема, а обычный — 90 %, то 20 % пропускной способности можно выделить под срочные пакеты, а все ос-

тальное — под обычные. Если мы так сделаем, срочный трафик получит вдвое большую пропускную способность по сравнению со своими нуждами, за счет этого значительно уменьшатся задержки при передаче пакетов. Такое распределение может быть достигнуто, если на каждый срочный пакет приходится четыре обычных (при предположении, что средний размер пакетов обоих классов примерно одинаков). Таким образом, есть надежда, что срочный трафик будет думать, что подсеть пустынна и безжизненна, хотя на самом деле она может быть загружена чрезвычайно сильно.

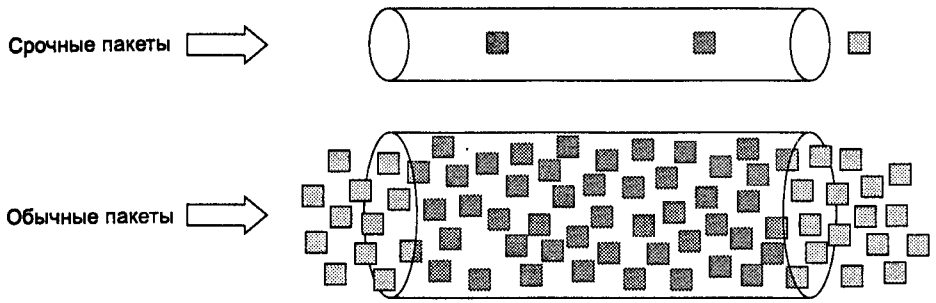


Рис. 5.34. Срочные пакеты движутся по свободной от трафика сети

Гарантированная пересылка

Более совершенная схема управления классами обслуживания называется схемой **гарантированной пересылки**. Эта стратегия описывается в документе RFC 2597. Подразумевается наличие четырех классов приоритетов, каждый из которых обладает своими ресурсами. Кроме того, определены три различных вероятности игнорирования пакетов, попавших в затор (низкая, средняя и высокая). Итого получается 12 сочетаний, то есть 12 классов обслуживания.

На рис. 5.35 показан один из способов обработки пакетов при гарантированной пересылке. На первом шаге пакеты разбиваются на четыре класса приоритетов. Эта процедура может выполняться на хосте-источнике (как показано на рисунке) или на первом маршрутизаторе. Преимуществом классификации пакетов на источнике является то, что в этой точке доступно больше информации о том, каким потокам принадлежат пакеты.

Вторым шагом является маркировка пакетов в соответствии с присвоенными им классами обслуживания. Для маркировки используется поле заголовка. К счастью, в заголовке IP-пакета, как мы вскоре увидим, имеется восьмибитное поле *Тип обслуживания*. Документ RFC 2597 говорит о том, что 6 из этих битов должны использоваться для обозначения класса обслуживания. Таким образом, есть возможность закодировать как все ныне существующие классы, так и те, которые могут появиться в будущем.

На третьем шаге пакеты прогоняются через фильтр, формирующий поток, который может задержать некоторые пакеты при организации четырех потоков. При этом могут использоваться, к примеру, алгоритмы дырявого или маркерного ведра. Если пакетов слишком много, некоторые из них могут вообще быть от-

вергнуты. Здесь большую роль играет категория игнорирования. Есть и более совершенные методы — с замерами и обратной связью.

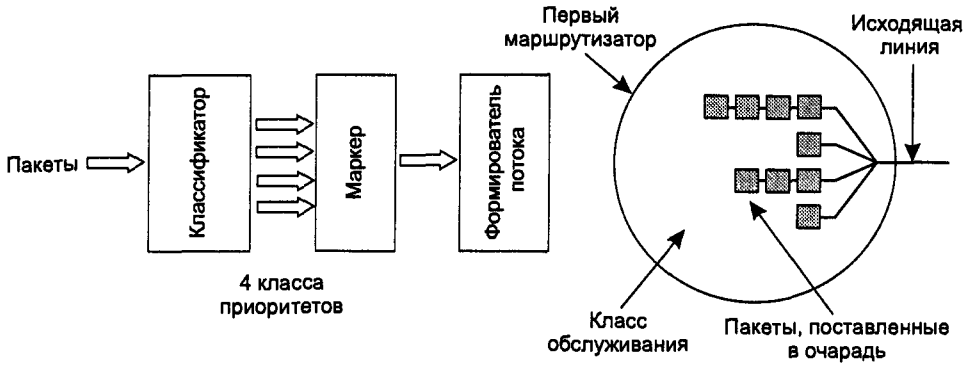


Рис. 5.35. Возможная реализация гарантированной пересылки потока данных

В приведенном примере все три шага выполняются на хосте-источнике, исходящий поток с которого направляется на первый маршрутизатор. Надо отметить, что все эти действия могут выполняться специальным сетевым программным обеспечением или даже операционной системой, что устраняет необходимость в замене существующих приложений.

Коммутация меток и MPLS

Пока проблемная группа IETF разрабатывала интегральные и дифференцированные виды обслуживания, производители маршрутизаторов стремились улучшить методы пересылки данных. Результатом их работы стали, в частности, появление метки в начале каждого пакета и маршрутизация, базирующаяся не на адресе назначения, а на этих метках. Если метку использовать в качестве индекса внутренней таблицы, то поиск подходящей исходящей линии сводится к просмотру таблицы. С использованием этого метода маршрутизация осуществляется очень быстро, и вдоль всего пути следования резервируются все необходимые ресурсы.

Конечно, метод расстановки меток рискует приблизиться слишком близко к виртуальным каналам. X.25, ATM, сети с ретрансляцией кадров, как и любые другие системы, в которых используются подсети с виртуальными каналами, также устанавливают метки (то есть идентификаторы виртуальных каналов) во все пакеты, затем осуществляют поиск по таблице и производят маршрутизацию на основе табличной записи. Несмотря на то, что очень многие представители Интернет-сообщества весьма презрительно относятся к сетям, ориентированным на соединение, похоже, что именно эта идея переживает второе рождение. На этот раз с ее помощью реализуются быстрая маршрутизация и высокое качество обслуживания. Тем не менее, есть существенные различия между тем, как в Интернете формируется маршрут и как это делается в сетях, ориентированных на соединение. Используется, конечно же, не архаичная коммутация пакетов, а нечто иное.

Это «иное» известно под самыми разными именами, включая **коммутацию меток** и **коммутацию тегов**. В конечном счете, проблемная группа IETF занялась стандартизацией своих идей, и это вылилось в появление стандарта **MPLS** (MultiProtocol Label Switching — мультипротокольная коммутация меток). Далее мы будем называть его MPLS. Стандарт описан в документе RFC 3031 и многих других.

Сделаем здесь небольшое отступление. Есть мнение, что *маршрутизация* и *коммутация* — это разные понятия. Маршрутизация — это процесс поиска адреса назначения по таблице и определения исходящей линии, на которую нужно послать данные, чтобы они дошли до адресата. Коммутация, напротив, использует метку, хранящуюся в пакете, в качестве индекса для таблицы пересылок. Впрочем, такие определения далеки от совершенства.

Первая проблема состоит вот в чем: куда поставить метку? Поскольку IP-пакеты не предназначены для виртуальных каналов, в их заголовке не предусмотрено место для номеров виртуальных каналов. Следовательно, нужно добавлять новый заголовок MPLS в начало IP-пакета. На линии между маршрутизаторами, использующей в качестве протокола кадрирования PPP, применяется формат, включающий в себя заголовки PPP, MPLS, IP и TCP, как показано на рис. 5.36. В каком-то смысле MPLS образует уровень с номером 2.5.

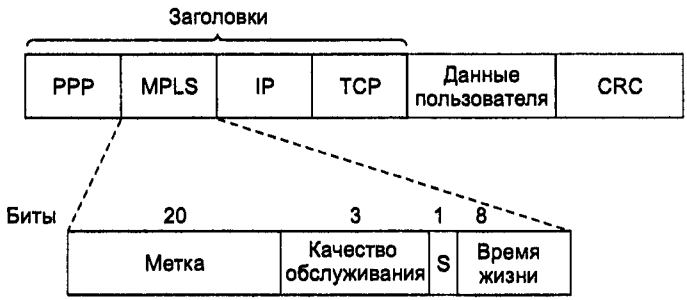


Рис. 5.36. Передача TCP-сегмента с использованием IP, MPLS и PPP

Обычно в заголовок MPLS входят четыре поля, наиболее важное из которых — поле *Метка*, значением которого является индекс. Поле *Качество обслуживания* указывает на применяемый класс обслуживания. Поле *S* связано со стеком меток в иерархических сетях (речь об этом пойдет далее). Если оно равно 0, пакет игнорируется. Благодаря этому исключаются бесконечные циклы в случае сбоя маршрутизации.

Заголовки MPLS не являются частью пакетов сетевого уровня, и к кадрам уровня передачи данных они отношения также не имеют. MPLS является методом, не зависящим от обоих этих уровней. Кроме всего прочего, это свойство означает, что можно создать такие коммутаторы MPLS, которые могут пересылать как IP-пакеты, так и ячейки ATM в зависимости от того, что необходимо в каждом конкретном случае. Именно отсюда следует «мультипротокольность» метода, отраженная в его названии.

Когда пакет (ячейка), расширенный за счет заголовка MPLS, прибывает на MPLS-совместимый маршрутизатор, извлеченная из него метка используется в качестве индекса таблицы, по которой определяются исходящая линия и значение новой метки. Смена меток используется во всех подсетях с виртуальными каналами, поскольку метки имеют только локальное значение, и два разных маршрутизатора могут снабдить независимые пакеты одной и той же меткой, если их нужно направить на одну и ту же линию третьего маршрутизатора. Поэтому, чтобы метки можно было различить на приемном конце, их приходится менять при каждом переходе. Мы видели этот механизм в действии — он был графически изображен на рис. 5.3. В MPLS используется такой же метод.

Еще одним отличием от традиционных виртуальных каналов является уровень агрегации. Конечно, можно каждому потоку, проходящему через подсеть, предоставить собственный набор меток. Однако более распространенным приемом является группировка потоков, заканчивающихся на данном маршрутизаторе или в данной ЛВС, и использование одной метки для всех таких потоков. О потоках, сгруппированных вместе и имеющих одинаковые метки, говорят, что они принадлежат одному **классу эквивалентности пересылок (FEC** — Forwarding Equivalence Class). В такой класс входят пакеты, не только идущие по одному и тому же маршруту, но и обслуживаемые по одному классу (в терминах дифференцированного обслуживания). Такие пакеты воспринимаются при пересылке одинаково.

При традиционной маршрутизации с использованием виртуальных каналов невозможно группировать разные пути с разными конечными пунктами в один виртуальный канал, потому что адресат не сможет их различить. В MPLS пакеты содержат не только метку, но и адрес назначения, поэтому в конце помеченного пути заголовок с меткой может быть удален, и дальнейшая маршрутизация может осуществляться традиционным способом — с использованием адреса назначения сетевого уровня.

Одним из основных отличий MPLS от обычных виртуальных каналов является способ построения таблицы маршрутизации. В традиционных сетях пользователь, желающий установить соединение, посылает установочный пакет для создания пути и соответствующей ему записи в таблице. В MPLS этого не происходит, потому что в этом методе вообще отсутствует установочная фаза для каждого соединения (в противном случае пришлось бы менять слишком большую часть существующего программного обеспечения Интернета).

Вместо этого существуют два альтернативных способа создания записей в таблице. При **методе, управляемом данными**, первый маршрутизатор, на который прибывает пакет, контактирует со следующим маршрутизатором на его пути и просит его создать метку для данного потока. Метод является рекурсивным. Можно сказать, что это своего рода создание виртуального канала по требованию.

Протоколы, обслуживающие этот метод, должны очень тщательно следить за предотвращением возникновения петель. Для этого часто используются так называемые **цветные потоки**. Обратное распространение FEC можно сравнить с передачей по подсети потока, окрашенного в уникальный цвет. Если маршрутизатор видит, что тот или иной цвет у него уже имеется, значит, возникла петля,

которую необходимо ликвидировать. Метод, управляемый данными, шире всего применяется в сетях с ATM в качестве транспортного уровня (например, в большинстве телефонных систем).

Второй метод, использующийся в не-ATM-сетях, — это **метод с явным управлением**. Имеется несколько вариантов этого подхода. Один из них работает следующим образом. При загрузке маршрутизатора выявляется, для каких маршрутов он является пунктом назначения (например, какие хосты находятся в его ЛВС). Для них создается один или несколько FEC, каждому из них выделяется метка, значение которой сообщается соседям. Соседи, в свою очередь, заносят эти метки в свои таблицы пересылки и посылают новые метки своим соседям. Процесс продолжается до тех пор, пока все маршрутизаторы не получат представление о маршрутах. По мере формирования путей могут резервироваться ресурсы, что позволяет обеспечить надлежащее качество обслуживания.

MPLS может работать на нескольких уровнях одновременно. На высшем уровне оператор связи может рассматриваться в качестве метамаршрутизатора; подразумевается, что между метамаршрутизаторами существует путь от источника до приемника. Этот путь может использоваться MPLS. Однако внутри сети каждого оператора также может применяться MPLS (второй уровень использования мультипротокольной коммутации меток). На самом деле, в пакете может содержаться целый стек меток. Бит *S* (см. рис. 5.36) позволяет маршрутизатору, удаляющему метку, узнать, остались ли у пакета еще метки. Единичное значение бита говорит о том, что метка — последняя в стеке, а нулевое значение говорит об обратном. На практике эта возможность чаще всего используется при реализации частных виртуальных сетей и рекурсивных каналов.

Хотя основные идеи MPLS довольно просты, детали этого метода чрезвычайно сложны, причем существует множество вариаций и улучшений. Поэтому мы не будем уходить вглубь вопроса. Дополнительную информацию можно найти в книгах (Davie и Rekhter, 2000; Lin и др., 2002; Pepelnjak и Guichard, 2001; Wang, 2001).

Объединение сетей

До сих пор мы неявно предполагали наличие единой однородной сети, в которой каждая машина использует один и тот же протокол на всех уровнях. К сожалению, данное предположение слишком оптимистично. Существует множество различных сетей, включая локальные, региональные и глобальные. На каждом уровне широко применяются многочисленные и разнообразные протоколы. В следующих разделах особое внимание будет уделено вопросам, возникающим при объединении двух или более сетей, формирующих **интерсеть**.

По вопросу о том, является ли сегодняшнее изобилие разнообразных типов сетей временным явлением, которое скоро перестанет иметь место, как только все наконец поймут, как замечательна сеть [вставьте свою любимую сеть], или оно является неизбежной данностью окружающего нас мира, единого мнения

нет. Наличие различных сетей всегда приводит к возникновению различных протоколов.

Мы полагаем, что разнообразие сетей (а следовательно, и протоколов) будет оставаться всегда по следующим причинам. Прежде всего, установленная база существующих сетей уже достаточно велика и продолжает расти. Почти все персональные компьютеры используют протокол TCP/IP. Во многих больших компаниях еще остались мейнфреймы, использующие протоколы SNA фирмы IBM. Существенная доля телефонных сетей ориентирована на ATM. Локальные сети персональных компьютеров все еще пользуются протоколами Novell NCP/IPX или AppleTalk. Наконец, беспроводные сети — эта бурно развивающаяся сегодня область — внедряют свои протоколы. Такая тенденция будет сохраняться в ближайшие годы благодаря наличию большого количества существующих сетей и еще благодаря тому, что некоторые производители считают, что возможность клиента легко переходить в системы других производителей не в их интересах.

Во-вторых, по мере того как компьютеры и сети становятся все дешевле, уровень принятия решения о выборе той или иной технологии все опускается и опускается, и теперь уже этим занимаются организации, желающие установить у себя сеть. Многие компании придерживаются политики, что приобретения стоимостью более миллиона долларов должны быть одобрены высшим руководством, покупки дороже 100 000 долларов — руководством среднего звена, а покупки дешевле 100 000 долларов могут совершаться главами отделов безо всякого согласования с вышестоящими руководителями. Такая политика может легко привести к тому, что в техническом отделе будут установлены рабочие станции UNIX, ориентированные на протокол TCP/IP, а отдел маркетинга установит у себя машины Macintosh с AppleTalk.

В-третьих, различные сети (например, ATM и беспроводные сети) основаны на принципиально разных технологиях, поэтому вряд ли стоит удивляться, что с появлением нового оборудования появится и новое программное обеспечение для него. Например, средний дом сейчас напоминает средний офис, каким он был десять лет назад: он битком набит компьютерами, не соединенными друг с другом. В будущем, возможно, будет нормой объединять в единую сеть телефон, телевизор и другую бытовую технику, так чтобы ею можно было управлять дистанционно. Появление этой новой технологии, несомненно, повлечет создание новых протоколов.

Рассмотрим следующий пример взаимодействия нескольких различных сетей, показанный на рис. 5.37. На нем изображена корпоративная сеть, части которой находятся далеко друг от друга и соединены глобальной сетью ATM. В одной из частей для объединения Ethernet, беспроводной ЛВС 802.11 и мейнфреймовой сети SNA корпоративного центра данных используется оптическая магистраль FDDI.

Целью объединения этих сетей является предоставление пользователям возможности общаться с пользователями любой другой из этих сетей, а также получать доступ к данным всех частей сети. Для реализации этих возможностей требуется посылать пакеты из одной сети в другую. Поскольку сети зачастую различаются довольно сильно, это действие осуществить не так уж просто, как мы увидим далее.

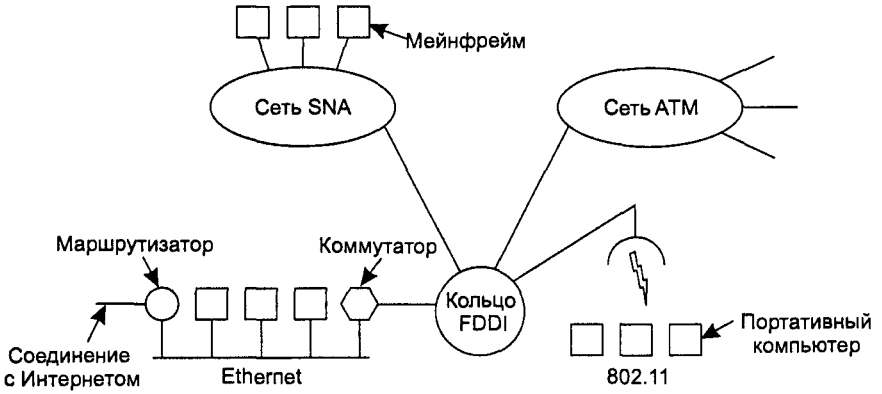


Рис. 5.37. Набор объединенных сетей

Различия сетей

Сети могут отличаться друг от друга довольно сильно и по разным параметрам. Некоторые из параметров, такие как методы модуляции или форматы кадров, нас сейчас не интересуют, поскольку они относятся к физическому уровню и уровню передачи данных. В табл. 5.5 приведен список некоторых параметров, которые могут встретиться на сетевом уровне. Именно сглаживание этих различий делает обеспечение работы объединенной сети значительно более сложным делом, чем обеспечение работы одной сети.

Когда пакетам приходится пересекать несколько сетей, отличных от исходной сети, может возникнуть много проблем, связанных с интерфейсами между сетями. Во-первых, когда пакеты из ориентированной на соединение сети должны пересечь не требующую соединений сеть, их порядок может нарушиться, причем для отправителя это может оказаться неожиданностью, а получатель может оказаться не подготовленным к такому событию. Часто будет требоваться преобразование протоколов, что может быть непросто, если необходимая функциональность не может быть выражена. Также понадобится преобразование форматов адресов, что может потребовать создания некой разновидности системы каталогов. Передача многоадресных пакетов через сеть, не поддерживающую многоадресную рассылку, потребует формирования отдельных пакетов для каждого адресата.

Различия в максимальном размере пакетов в разных сетях составляют главную головную боль. Как передать 8000-байтовый пакет по сети, в которой максимальный размер пакета равен 1500 байтам? При передаче пакета с обязательствами доставки в реальном масштабе времени по сети, не предоставляющей каких-либо гарантий работы в реальном времени, возникает проблема разницы в качестве обслуживания.

Методы обработки ошибок, управления потоком и борьбы с перегрузкой часто различаются в различных сетях. Если отправитель и получатель ожидают, что все пакеты будут доставлены без ошибок и с сохранением порядка, а сеть просто

игнорирует пакеты, когда ей угрожает перегрузка, или пакеты, направляясь различными путями, приходят к получателю совсем не в том порядке, в каком они были отправлены, то многие приложения просто не смогут работать в таких условиях. Различия в механизмах безопасности, установке параметров, правилах тарификации и даже в законах, охраняющих тайну переписки, могут послужить причиной многих проблем.

Таблица 5.5. Некоторые аспекты различия сетей

Аспект	Возможные значения
Предлагаемый сервис	Ориентированные на соединение или не требующие соединения
Протоколы	IP, IPX, SNA, ATM, MPLS, AppleTalk и др.
Адресация	Плоская (802) или иерархическая (IP)
Многоадресная рассылка	Присутствует или отсутствует (а также широковещание)
Размер пакета	У каждой сети есть свой максимум
Качество обслуживания	Может присутствовать и отсутствовать. Много разновидностей
Обработка ошибок	Надежная, упорядоченная и неупорядоченная доставка
Управление потоком	Скользящее окно, управление скоростью, другое или никакого
Борьба с перегрузкой	Дырявое ведро, маркерное ведро, сдерживающие пакеты, нерегулярное раннее обнаружение и др.
Безопасность	Правила секретности, шифрование и т. д.
Параметры	Различные тайм-ауты, спецификация потока и др.
Тарификация	По времени соединения, за пакет, побайтно или никак

Способы объединения сетей

Как уже было показано в главе 4, сети могут объединяться с помощью разных устройств. Давайте вкратце вспомним эту тему. На физическом уровне сети объединяются повторителями или концентраторами, которые просто переносят биты из одной сети в другую такую же сеть. Чаще всего это аналоговые устройства, ничего не смыслящие в цифровых протоколах (они просто-напросто регенерируют сигналы).

Если мы поднимемся на уровень выше, мы обнаружим мосты и коммутаторы, работающие на уровне передачи данных. Они могут принимать кадры, анализировать их MAC-адреса, направлять их в другие сети, осуществляя по ходу дела минимальные преобразования протоколов, например из Ethernet в FDDI или в 802.11.

На сетевом уровне у нас есть маршрутизаторы, соединяющие две сети. Если сетевые уровни у них разные, маршрутизатор может обеспечить перевод пакета из одного формата в другой, хотя такие преобразования сейчас выполняются все реже. Маршрутизатор может поддерживать несколько протоколов, тогда он называется **мультипротокольным маршрутизатором**.

На транспортном уровне существуют транспортные шлюзы, предоставляющие интерфейсы для соединений своего уровня. Транспортный шлюз позволяет,

к примеру, передавать пакеты из сети TCP в сеть SNA (протоколы транспортного уровня у них различаются), склеивая одно соединение с другим.

Наконец, на прикладном уровне шлюзы занимаются преобразованием семантики сообщений. Например, шлюзы между электронной почтой Интернета (RFC 822) и электронной почтой X.400 должны анализировать содержимое сообщений и изменять различные поля электронного конверта.

В данной главе мы сосредоточим наше внимание на объединении сетей на сетевом уровне. Чтобы понять, в чем состоит его отличие от объединения на уровне передачи данных, рассмотрим рис. 5.38. На рис. рис. 5.38, *а* источник *S* пытается послать пакет приемнику *D*. Эти две машины работают в разных сетях Ethernet, соединенных коммутатором. Источник *S* вставляет пакет в кадр и отправляет его. Кадр прибывает на коммутатор, который по MAC-адресу определяет, что его надо переслать в ЛВС 2. Коммутатор просто снимает кадр с ЛВС 1 и передает его в ЛВС 2.

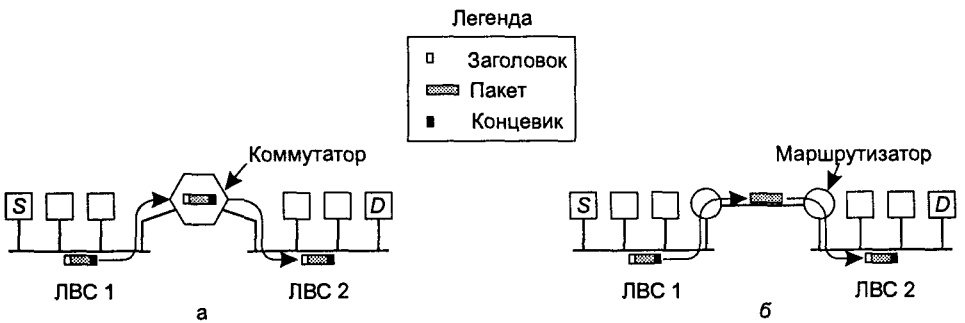


Рис. 5.38. Две сети Ethernet, объединенные коммутатором (*а*); две сети Ethernet, объединенные маршрутизаторами (*б*)

Теперь рассмотрим ту же ситуацию, но с применением другого оборудования. Допустим, две сети Ethernet объединены не коммутатором, а парой маршрутизаторов. Маршрутизаторы между собой соединены двухточечной линией, которая может представлять собой, например, выделенную линию длиной в тысячи километров. Что в данном случае будет происходить с кадром? Он принимается маршрутизатором, из его поля данных извлекается пакет. Далее маршрутизатор анализирует содержащийся в пакете адрес (например, IP-адрес). Этот адрес нужно отыскать в таблице маршрутизации. В соответствии с ним принимается решение об отправке пакета (возможно, упакованного в кадр нового вида — это зависит от протокола, используемого линией) на удаленный маршрутизатор. На противоположном конце пакет вставляется в поле данных кадра Ethernet и помещается в ЛВС 2.

В чем заключается основная разница между случаем коммутации (установки моста) и маршрутизации? Коммутатор (мост) пересылает весь пакет, обосновывая свое решение значением MAC-адреса. При применении маршрутизатора пакет извлекается из кадра, и для принятия решения используется адрес, содержащийся именно в пакете. Коммутаторы не обязаны вникать в подробности уст-

ройства протокола сетевого уровня, с помощью которого производится коммутация. А маршрутизаторы обязаны.

Сцепленные виртуальные каналы

Наиболее распространенными являются два стиля объединения сетей: ориентированное на соединение сцепление подсетей виртуальных каналов и дейтаграммный интерсетевой стиль. Мы рассмотрим их поочередно, однако необходимо предварить наше рассмотрение небольшим вступлением. В прошлом большинство сетей (общего пользования) были ориентированными на соединение (сети с ретрансляцией кадров, SNA, 802.16 и ATM по сей день являются таковыми). Со стремительным развитием Интернета все больше входили в моду дейтаграммы. Тем не менее, было бы ошибкой думать, что дейтаграммный способ будет существовать вечно. В этом деле единственное постоянство — это изменчивость. С ростом доли и важности мультимедийных данных в общем потоке растет вероятность того, что наступит эпоха возрождения для технологий, ориентированных на соединение. Причиной тому является тот простой факт, что при установлении соединения гораздо проще гарантировать определенный уровень обслуживания. Далее мы еще уделим некоторое место сетям, ориентированным на соединение.

В модели сцепленных виртуальных каналов, показанной на рис. 5.39, соединение с хостом в удаленной сети устанавливается способом, близким к тому, как устанавливаются обычные соединения. Подсеть видит, что адресат является удаленным, и создает виртуальный канал к ближайшему маршрутизатору из сети адресата. Затем строится виртуальный канал от этого маршрутизатора к внешнему **шлюзу** (многопротокольному маршрутизатору). Этот шлюз запоминает существование созданного виртуального канала в своих таблицах и строит новый виртуальный канал к маршрутизатору в следующей подсети. Процесс продолжается до тех пор, пока не будет достигнут хост-получатель.

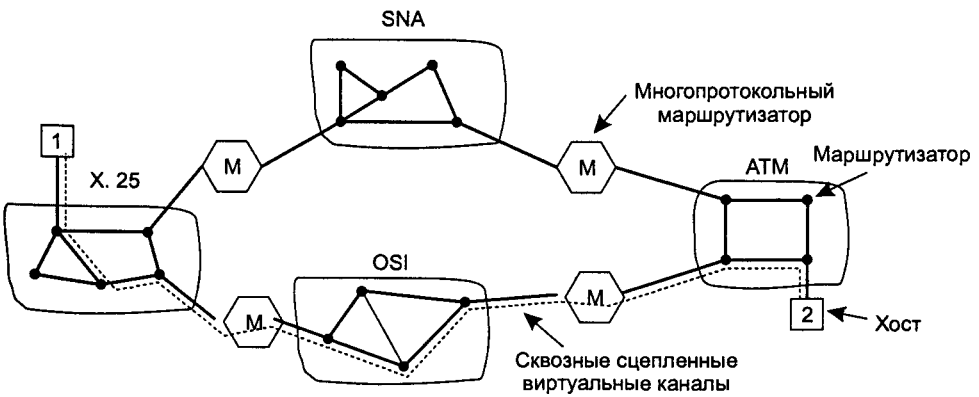


Рис. 5.39. Объединение сетей с помощью сцепленных виртуальных каналов

Когда по проложенному пути начинают идти пакеты данных, каждый шлюз переправляет их дальше, преобразуя формат пакетов и номера виртуальных каналов. Очевидно, что все информационные пакеты будут передаваться по одному и тому же пути и, таким образом, прибудут к пункту назначения с сохранением порядка отправления.

Существенной особенностью данного подхода является то, что последовательность виртуальных пакетов устанавливается от источника через один или более шлюзов к приемнику. Каждый шлюз хранит таблицы, содержащие информацию о проходящих через них виртуальных каналах, о том, как осуществлять маршрутизацию для них и каков номер нового виртуального канала.

Такая схема лучше всего работает, когда все сети обладают примерно одинаковыми свойствами. Например, если каждая из них гарантирует надежную доставку пакета сетевого уровня, то, исключив случай сбоя системы где-то на его пути, можно сказать, что и весь поток от источника до приемника будет надежным. С другой стороны, если машина-источник работает в сети, которая гарантирует надежную доставку, а какая-то промежуточная сеть может терять пакеты, то сцепление радикально изменит сущность сервиса.

Сцепленные виртуальные каналы часто применяются на транспортном уровне. В частности, можно построить битовый канал, используя, скажем, SNA, который заканчивается на шлюзе, и иметь при этом TCP-соединение между соседними шлюзами. Таким образом, можно построить сквозной виртуальный канал, охватывающий разные сети и протоколы.

Дейтаграммное объединение сетей

Альтернативной моделью объединения сетей является дейтаграммная модель, показанная на рис. 5.40. В данной модели единственный сервис, который сетевой уровень предоставляет транспортному уровню, состоит в возможности посылать в сеть дейтаграммы и надеяться на лучшее. На сетевом уровне нет никакого упоминания о виртуальных каналах, не говоря уже об их сцеплении. В этой модели пакеты не обязаны следовать по одному и тому же маршруту, даже если они принадлежат одному соединению. На рис. 5.40 показаны дейтаграммы, следующие от хоста 1 к хосту 2 и выбирающие различные маршруты по объединенной сети. Выбор маршрута производится независимо для каждого пакета. Он может зависеть от текущей загруженности сети. При такой стратегии могут использоваться различные маршруты, что позволяет достигать большей пропускной способности, чем при применении модели сцепленных виртуальных каналов. С другой стороны, не дается никакой гарантии того, что пакеты прибудут к получателю в нужном порядке, если они вообще прибудут.

Модель, изображенная на рис. 5.40, не так проста, как кажется. Во-первых, если каждая сеть обладает своим сетевым уровнем, то пакет из одной сети не сможет перейти в другую сеть. Можно представить себе многопротокольные маршрутизаторы, пытающиеся преобразовать пакет из одного формата в другой, но, если только эти форматы не являются близкими родственниками, такое преобразование всегда будет неполным и часто обречено на ошибку.

Еще более серьезные проблемы возникают с адресацией. Представьте себе простой случай: интернет-хост пытается послать IP-пакет хосту, находящемуся в соседней сети SNA. Адреса IP и SNA различаются. Значит, потребуется двухстороннее приведение одного формата адреса к другому. Более того, различается сама концепция адресации. В IP адресуемой единицей является хост (собственно, интерфейсная карта). В SNA адресуемыми могут быть не только хосты, но и другие сущности (например, физические устройства). В идеале необходимо поддерживать базу данных, отображающую одни адресуемые сущности на другие, но это задача исключительно трудоемкая, если не сказать невыполнимая.

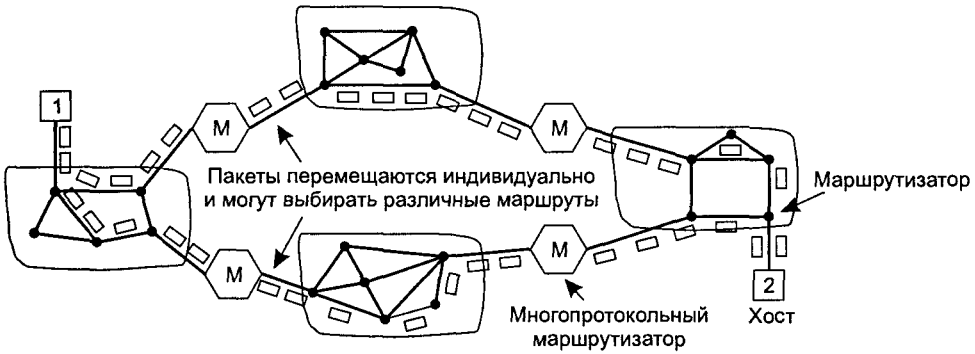


Рис. 5.40. Дейтаграммная объединенная сеть

Другая идея заключается в создании универсального межсетевого пакета, который распознавался бы всеми маршрутизаторами. Именно эта идея была взята за основу при разработке протокола IP: IP-пакеты созданы для передачи по разным сетям. Впрочем, может, конечно, оказаться так, что применяемый сегодня в Интернете протокол IPv4 вытеснит с рынка все остальные форматы, IPv6 (будущий протокол Интернета) не получит ожидаемого распространения и ничего нового вообще не будет изобретено, однако пока что исторически все складывается совсем не так. Заставить всех согласиться применять только один формат практически невозможно, особенно учитывая тот факт, что каждая компания считает самым большим своим достижением изобретение, внедрение и раскручивание собственного формата.

Подведем краткие итоги обсуждения двух способов объединения сетей. Модель сцепленных виртуальных каналов обладает теми же преимуществами, что и применение виртуальных каналов внутри единой подсети: буферы могут быть зарезервированы заранее, сохранение порядка пакетов гарантируется, могут использоваться короткие заголовки, кроме того, можно избежать неприятностей, вызываемых дубликатами пакетов.

Недостатки опять те же самые: маршрутизаторы должны хранить таблицы с записями для каждого открытого соединения, при возникновении затора обходные пути не используются, а выход маршрутизатора из строя обрывает все проходящие через него виртуальные каналы. Кроме того, очень сложно, может, даже невозможно реализовать систему виртуальных каналов, если в состав объединенной сети входит хотя бы одна ненадежная дейтаграммная сеть.

Свойства дейтаграммного подхода к объединению сетей те же самые, что и у дейтаграммных подсетей: риск возникновения перегрузки выше, но также больше возможностей для адаптации к ней, высокая надежность в случае отказов маршрутизаторов, однако требуются более длинные заголовки пакетов. В объединенной сети, как и в единой дейтаграммной сети, возможно применение различных адаптивных алгоритмов выбора маршрута.

Главное преимущество дейтаграммного подхода к объединению сетей заключается в том, что он может применяться в подсетях без виртуальных каналов. К дейтаграммным сетям относятся многие локальные сети, мобильные сети (в том числе применяемые в воздушном и морском транспорте) и даже некоторые глобальные сети. При включении одной из этих сетей в объединенную сеть стратегия объединения сетей на основе виртуальных каналов встречает серьезные трудности.

Туннелирование

Объединение сетей в общем случае является исключительно сложной задачей. Однако есть частный случай, реализация которого вполне осуществима. Это случай, при котором хост-источник и хост-приемник находятся в сетях одного типа, но между ними находится сеть другого типа. Например, представьте себе международный банк, у которого имеется одна TCP/IP-сеть на основе Ethernet в Париже и такая же сеть в Лондоне, а между ними находится какая-нибудь глобальная не-IP сеть (например, ATM), как показано на рис. 5.41.

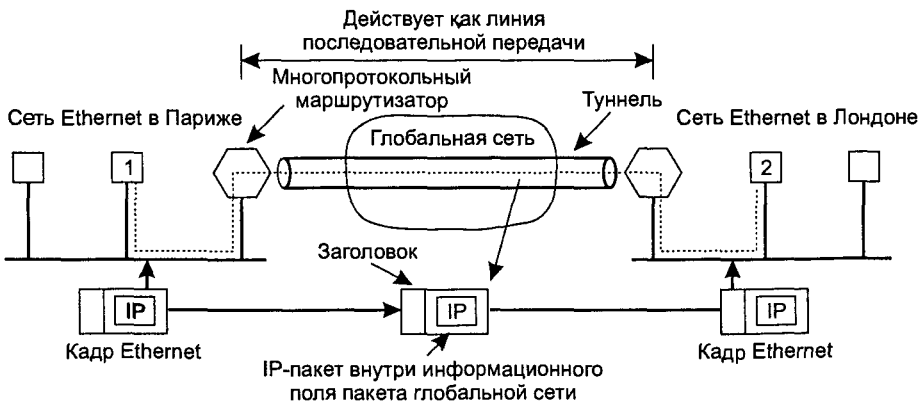


Рис. 5.41. Туннелирование пакета из Парижа в Лондон

Метод решения данной проблемы называется **туннелированием**. Чтобы послать IP-пакет хосту 2, хост 1 формирует пакет, содержащий IP-адрес хоста 2, помещает его в кадр Ethernet, адресованный парижскому многопротокольному маршрутизатору, и пересылает его по сети Ethernet. Получив кадр, многопротокольный маршрутизатор извлекает IP-пакет, помещает его в поле данных пакета сетевого уровня глобальной сети и пересылает его лондонскому многопрото-

кольному маршрутизатору. Когда пакет попадает туда, лондонский многопротокольный маршрутизатор извлекает IP-пакет и посылает его хосту 2 внутри кадра Ethernet.

Глобальную сеть при этом можно рассматривать как большой туннель, простирающийся от одного многопротокольного маршрутизатора до другого. IP-пакет, помещенный в красивую упаковку, просто перемещается от одного конца туннеля до другого. Ему не нужно беспокоиться о взаимодействии с глобальной сетью. Это же касается и хостов сетей Ethernet по обе стороны туннеля. Переупаковкой пакета и переадресацией занимаются многопротокольные маршрутизаторы. Для этого им нужно уметь разбираться в IP-адресах и обладать информацией о формате пакетов глобальной сети. В результате весь путь от середины одного многопротокольного маршрутизатора до середины другого работает, как линия последовательной передачи.

Чтобы сделать этот пример еще проще и понятнее, рассмотрим в качестве аналогии водителя автомобиля, направляющегося из Парижа в Лондон. По территории Франции автомобиль двигается по дороге сам. Но, достигнув Ла-Манша, он погружается в высокоскоростной поезд и транспортируется под проливом по туннелю (автомобилям не разрешается передвигаться по туннелю самим). Таким образом, автомобиль перевозится как груз (рис. 5.42). На другом конце туннеля автомобиль спускается с железнодорожной платформы на английское шоссе и снова продолжает путь своими силами. Точно такой же принцип применяется при туннелировании пакетов, проходящих через чужеродную сеть.

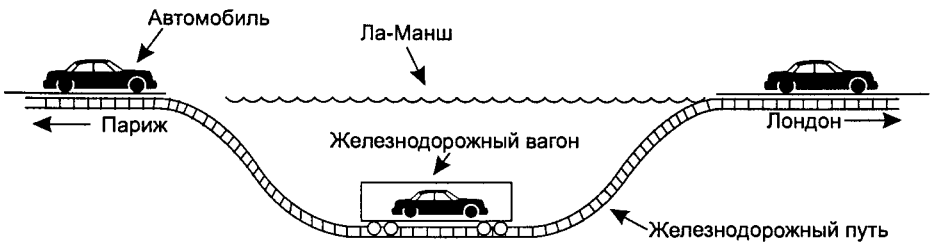


Рис. 5.42. Туннелирование автомобиля, едущего из Парижа в Лондон

Маршрутизация в объединенных сетях

Маршрутизация в объединенных сетях аналогична маршрутизации в единой подсети, но связана с некоторыми дополнительными сложностями. Рассмотрим, например, объединенную сеть, изображенную на рис. 5.43, *а*, в которой пять сетей соединены шестью (возможно, многопротокольными) маршрутизаторами. Построение графа для данной сети усложняется тем фактом, что каждый многопротокольный маршрутизатор может напрямую обращаться (то есть посылать пакеты) к любому другому многопротокольному маршрутизатору, соединенному с той же сетью, что и он. Например, многопротокольный маршрутизатор *B* на рис. 5.43, *а* может напрямую обращаться к многопротокольным маршрутизаторам *A* и *C* по сети 2,

а также к многопротокольному маршрутизатору *B* по сети 3. В результате мы получим граф, показанный на рис. 5.43, *б*.

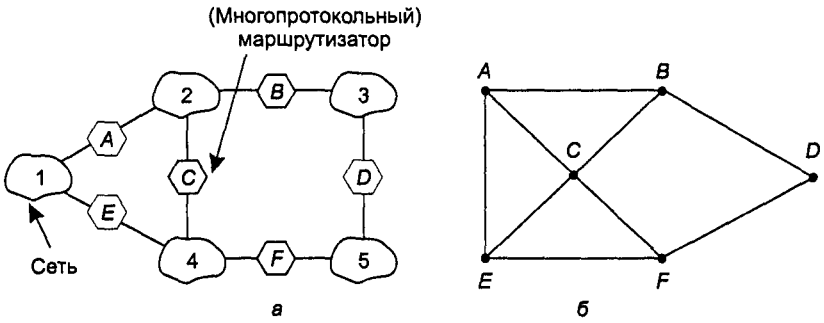


Рис. 5.43. Объединенная сеть (*а*); граф объединенной сети (*б*)

Когда граф построен, к множеству многопротокольных маршрутизаторов можно применить известные алгоритмы выбора маршрутов, такие как маршрутизация по вектору расстояний или алгоритм, учитывающий состояние линий. Таким образом, получается двухуровневый алгоритм маршрутизации: в пределах каждой сети используется **внутренний шлюзовый протокол**, но между сетями применяется **внешний шлюзовый протокол** (термин «шлюз» употреблялся раньше вместо термина «маршрутизатор»). Так как все сети независимы, в них могут применяться различные алгоритмы маршрутизации. Благодаря независимости сетей друг от друга, они часто называются **автономными системами (AS)** (АС).

Типичный интерсетевой пакет, отправляясь из своей локальной сети, адресуется своему локальному многопротокольному маршрутизатору (адрес прописывается в заголовке MAC-уровня). Когда он попадает туда, сетевой уровень с помощью своих таблиц решает, какому многопротокольному маршрутизатору переслать этот пакет. Если до соответствующего маршрутизатора можно добраться с помощью «родного» сетевого протокола пакета, он направляется туда напрямую. В противном случае он туннелируется туда, для чего используются пакет и протокол промежуточной сети. Этот процесс повторяется до тех пор, пока пакет не прибудет в сеть адресата.

Одно из различий внутрисетевой и межсетевой маршрутизаций состоит в том, что при межсетевой маршрутизации часто требуется пересечение международных границ. Неожиданно вступают в игру различные законы, как, например, строгое шведское законодательство, касающееся экспорта личных сведений о шведских гражданах за пределы Швеции. Другим примером является канадский закон, гласящий, что поток данных, начинающийся и заканчивающийся в Канаде, не может покидать пределы страны. Это означает, что трафик из Виндзора, штат Онтарио, в Ванкувер не может проходить через окрестности соседнего Детройта, даже если такой путь быстрее и дешевле.

Другим различием внутрисетевой и межсетевой маршрутизаций является их стоимость. В пределах одной сети обычно применяется единый алгоритм тарификации. Однако различные сети могут управляться различными организация-

ми, и один маршрут может оказаться дешевле другого. Аналогично, качество обслуживания, предлагаемое в разных сетях, также может различаться, что также может послужить причиной выбора того или иного маршрута.

Фрагментация

Все сети накладывают ограничения на размер своих пакетов. Эти пределы вызваны различными предпосылками, среди которых есть следующие:

1. Аппаратные (например, размер кадра Ethernet).
2. Операционная система (например, все буферы имеют размер 512 байт).
3. Протоколы (например, количество бит в поле длины пакета).
4. Соответствие какому-либо международному или национальному стандарту.
5. Желание снизить количество пакетов, пересылаемых повторно из-за ошибок передачи.
6. Желание предотвратить ситуацию, когда один пакет слишком долгое время занимает канал.

Результатом действия всех этих факторов является то, что разработчики не могут выбирать максимальный размер пакета по своему усмотрению. Максимальный размер поля полезной нагрузки варьируется от 48 байт (ATM-ячейки) до 65 515 байт (IP-пакеты), хотя на более высоких уровнях размер поля полезной нагрузки часто бывает больше.

Очевидно, возникает проблема, когда большой пакет хочет пройти по сети, в которой максимальный размер пакетов слишком мал. Одно из решений состоит в предотвращении возникновения самой проблемы. Другими словами, объединенная сеть должна использовать такой алгоритм маршрутизации, который не допускает пересылки пакетов по сетям, которые не могут их принять. Однако это решение вовсе не является решением. Что произойдет, если исходный пакет окажется слишком велик для сети адресата? Алгоритм маршрутизации будет в данном случае бессилен.

Следовательно, единственное решение проблемы заключается в разрешении шлюзам разбивать пакеты на **фрагменты** и посылать каждый фрагмент в виде отдельного межсетевого пакета. Однако, как вам скажет любой родитель маленького ребенка, преобразование объекта в небольшие фрагменты существенно проще, чем обратный процесс. (Физики даже дали этому эффекту ломания игрушек специальное название: второй закон термодинамики.) В сетях с коммутацией пакетов также существует проблема с восстановлением пакетов из фрагментов.

Для восстановления исходных пакетов из фрагментов применяются две противоположные стратегии. Первая стратегия заключается в том, чтобы фрагментация пакета, вызванная сетью с пакетами малых размеров, оставалась прозрачной для обоих хостов, обменивающихся пакетом. Этот вариант показан на рис. 5.44, *а*. «Мелкопакетная» сеть имеет шлюзы (скорее всего, это специализированные маршрутизаторы), предоставляющие интерфейсы другим сетям. Когда на такой шлюз приходит пакет слишком большого размера, он разбивается на фрагменты.

Каждый фрагмент адресуется одному и тому же выходному шлюзу, восстанавливающему из этих фрагментов исходный пакет. Таким образом, прохождение данных через мелкопакетную сеть оказывается прозрачным. Соседние сети даже не догадываются о том, что у них под боком пакеты страшным образом нарезаются, а потом снова склеиваются. В сетях ATM, например, есть даже специальная аппаратура для обеспечения прозрачной фрагментации пакетов (разбивания на ячейки) и обратной сборки ячеек в пакеты. В мире ATM фрагментацию называют сегментацией. Концепция та же самая, а отличия есть только в некоторых деталях.

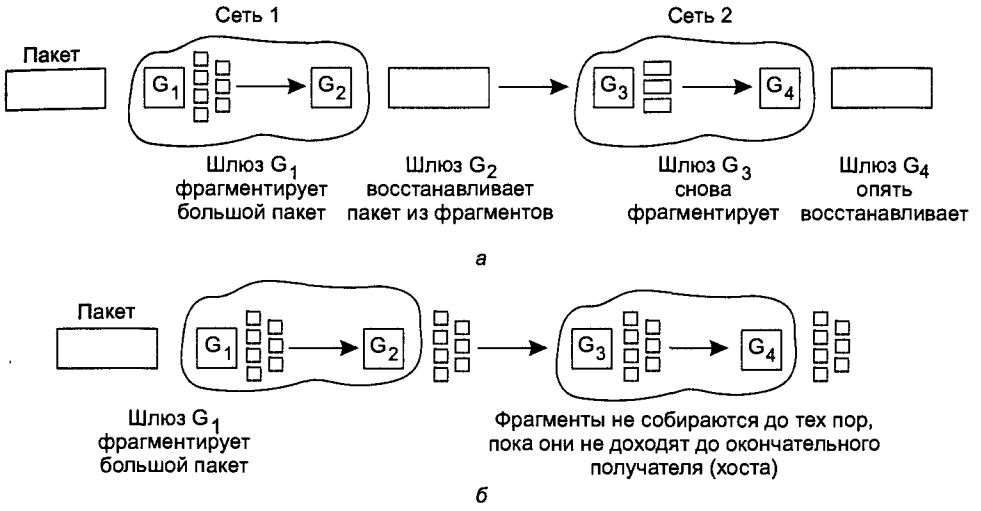


Рис. 5.44. Прозрачная фрагментация (а); непрозрачная фрагментация (*б*)

Прозрачная фрагментация проста, но, тем не менее, создает некоторые проблемы. Во-первых, выходной шлюз должен уметь определять момент получения последней части пакета, поэтому каждый фрагмент должен содержать либо поле счетчика, либо признак конца пакета. Во-вторых, все фрагменты должны выходить через один и тот же шлюз. Таким образом, налагается запрет на использование фрагментами различных путей к окончательному получателю, и в результате может оказаться потерянной часть производительности. Наконец, процессы фрагментации и последующей сборки пакетов при прохождении каждой сети с малым размером пакетов приводят к дополнительным накладным расходам. Сетям ATM требуется прозрачная фрагментация.

Другая стратегия фрагментации состоит в отказе от восстановления пакета из фрагментов на промежуточных маршрутизаторах. Как только пакет оказывается разбитым на отдельные фрагменты, с каждым фрагментом обращаются как с отдельным пакетом. Все фрагменты проходят через выходной шлюз (или несколько), как показано на рис. 5.44, *б*. Задача восстановления оригинального пакета возложена на получающий хост. Так работает IP.

С непрозрачной фрагментацией связаны свои проблемы. Например, она требует, чтобы *каждый* хост мог восстановить пакет из фрагментов. Кроме того, при фрагментации большого пакета возрастают суммарные накладные расходы, так

как каждый фрагмент должен иметь заголовок. В то время как в случае прозрачной фрагментации лишние заголовки при выходе из мелкопакетной сети исчезали, в данном методе накладные расходы сохраняются на протяжении всего пути. Однако преимущество непрозрачной фрагментации состоит в возможности использовать для передачи фрагментов несколько различных маршрутов, что повышает производительность. Естественно, при использовании модели сцепленных виртуальных каналов это преимущество оказывается бесполезным.

Фрагменты пакета должны нумероваться таким образом, чтобы можно было восстановить исходный поток данных. Один из способов нумерации фрагментов состоит в использовании дерева. Если пакет 0 должен быть расщеплен, фрагменты получают номера 0.0, 0.1, 0.2 и т. д. Если эти фрагменты в дальнейшем сами фрагментируются, получающиеся кусочки нумеруются так: 0.0.0, 0.0.1, 0.0.2, ..., 0.1.0, 0.1.1, 0.1.2 и т. д. Если в заголовках зарезервировано достаточно места для случая наиболее глубокого разбиения и при этом отсутствуют дубликаты, то такая схема гарантирует правильную сборку пакета получателем независимо от порядка, в котором будут получены отдельные фрагменты.

Однако если одна из сетей нечаянно потеряет или удалит один или несколько фрагментов, то понадобится повторная передача всего пакета с тяжелыми последствиями для системы нумерации. Представим, что передается пакет длиной 1024 байта. При первой передаче пакета он разбивается на четыре одинаковых фрагмента с номерами 0.0, 0.1, 0.2 и 0.3. Фрагмент 0.1 теряется по дороге, а остальные успешно добираются до получателя. Отправитель не получает подтверждения на переданный пакет и посылает его снова. Но на этот раз срабатывает закон бутерброда (или закон Мерфи): пакет пересылается по другому маршруту и проходит через сеть с 512-байтовым ограничением на размер пакетов, поэтому пакет разбивается всего на два фрагмента. Получив фрагмент с номером 0.1, получатель может подумать, что это — как раз недостающая деталь, и в результате соберет пакет неверно.

Совершенно иной и гораздо более совершенный подход к решению данной проблемы состоит в определении размера элементарного фрагмента, достаточно малого, чтобы он мог пройти по любой сети. То есть пакет разбивается на элементарные фрагменты одинакового размера плюс довесок, который может быть только короче всех остальных. Для пущей эффективности межсетевой пакет может содержать несколько фрагментов. Межсетевой заголовок должен содержать номер исходного пакета и номер (первого) элементарного фрагмента, содержащегося в нем. Как обычно, в заголовке должен содержаться признак конца исходного пакета.

Такой подход требует включения в заголовок межсетевого пакета двух полей: порядкового номера исходного пакета и порядкового номера фрагмента. Есть вполне очевидный компромисс между размером элементарного фрагмента и числом бит номера фрагмента. Поскольку размер элементарного пакета выбирается таким образом, что он может пройти по любой сети, дальнейшая фрагментация межсетевого пакета не составляет проблемы. Последним пределом является элементарный фрагмент размером с бит или байт, при этом номер фрагмента, содержащийся в заголовке пакета, превращается в смещение до этого бита или байта (рис. 5.45).

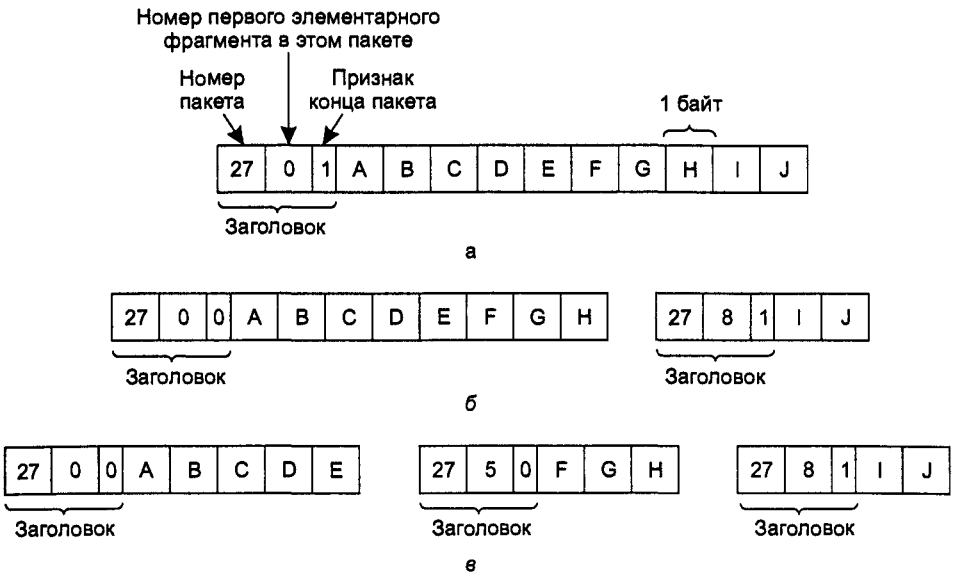


Рис. 5.45. Фрагментация при элементарном размере 1 байт: исходный пакет, содержащий 10 байт данных (*а*); фрагменты после прохождения через сеть с максимальным размером 8 байт (*б*); фрагменты после прохождения через шлюз размером 5 (*а*)

Некоторые межсетевые протоколы развивают этот метод дальше и рассматривают всю передачу по виртуальному каналу как один гигантский пакет, так что каждый фрагмент содержит абсолютный номер первого байта фрагмента.

Сетевой уровень в Интернете

Прежде чем начинать рассматривать особенности реализации сетевого уровня в интернет-технологиях, неплохо было бы вспомнить принципы, которые были основополагающими в прошлом, при его разработке, и которые обеспечили сегодняшний успех. В наше время люди все чаще пренебрегают ими. Между тем эти принципы пронумерованы и включены в документ RFC 1958, с которым полезно ознакомиться (а для разработчиков протоколов он должен быть просто обязательным для прочтения, с экзаменом в конце). Этот документ сильно перекликается с идеями, которые можно найти в книгах (Clark, 1988; Saltzer и др., 1984). Далее мы приведем 10 основных принципов, начиная с самых главных.

1. **Убедитесь в работоспособности.** Нельзя считать разработку (или стандарт) законченной, пока не проведен ряд успешных соединений между прототипами. Очень часто разработчики сначала пишут тысячестраничное описание стандарта, утверждают его, а потом обнаруживается, что он еще очень сырой или вообще неработоспособен. Тогда пишется версия 1.1 стандарта. Так бывает, но так быть не должно.

2. **Упрощайте.** Если есть сомнения, всегда самый простой выбор является самым лучшим. Уильям Оккам (William Occam) декларировал этот принцип еще в XIV веке («бритва Оккама»). Его можно выразить следующим образом: «*Борись с излишествами*». Если какое-то свойство не является абсолютно необходимым, забудьте про него, особенно если такого же эффекта можно добиться комбинированием уже имеющихся свойств.
3. **Всегда делайте четкий выбор.** Если есть несколько способов реализации одного и того же, необходимо выбрать один из них. Увеличение количества способов — это порочный путь. В стандартах часто можно встретить несколько опций, режимов или параметров. Почему так получается? Лишь потому, что при разработке было несколько авторитетных мнений на тему того, что является наилучшим. Разработчики должны избегать этого и сопротивляться подобным тенденциям. Надо просто уметь говорить «Нет».
4. **Используйте модульный принцип.** Этот принцип напрямую приводит к идее стеков протоколов, каждый из которых работает на одном из независимых уровней. Таким образом, если обстоятельства складываются так, что необходимо изменить один из модулей или уровней, то это не приводит к необходимости других частей системы.
5. **Предполагайте разнородность.** В любой большой сети могут сосуществовать различные типы оборудования, средства передачи данных и различные приложения. Сетевая технология должна быть достаточно гибкой, простой и обобщенной, чтобы работать в таких условиях.
6. **Избегайте статичности свойств и параметров.** Если есть какие-то обязательные параметры (например, максимальный размер пакета), то лучше заставить отправителя и получателя договариваться об их конкретных значениях, чем жестко закреплять их.
7. **Проект должен быть хорошим, но он не может быть идеальным.** Очень часто разработчики создают хорошие проекты, но не могут предусмотреть какие-нибудь причудливые частные случаи. Не стоит портить то, что сделано хорошо и работает в большинстве случаев. Вместо этого имеет смысл переложить все бремя ответственности за «улучшения» проекта на тех, кто предъявляет свои странные требования.
8. **Тщательно продумывайте отправку данных, но будьте снисходительны при приеме данных.** Другими словами, посылайте пакеты, только убедившись в том, что они полностью соответствуют всем требованиям стандартов. При этом надо иметь в виду, что приходящие пакеты не всегда столь идеальны и их тоже нужно обрабатывать.
9. **Продумайте масштабируемость.** Если в системе работают миллионы хостов и миллиарды пользователей, о централизации можно забыть. Нагрузка должна быть распределена максимально равномерно между имеющимися ресурсами.
10. **Помните о производительности и цене.** Никого не заинтересует сеть низкопроизводительная или дорогостоящая.

Теперь перейдем от общих принципов к деталям построения сетевого уровня Интернета. На сетевом уровне Интернет можно рассматривать как набор подсе-

тей или **автономных систем**, соединенных друг с другом. Структуры как таковой Интернет не имеет, но все же есть несколько магистралей. Они собраны из высокопроизводительных линий и быстрых маршрутизаторов. К магистралям присоединены региональные сети (сети среднего уровня), с которыми, в свою очередь, соединяются локальные сети многочисленных университетов, компаний и провайдеров. Схема этой квазииерархической структуры показана на рис. 5.46.

Вся эта конструкция «склеивается» благодаря протоколу сетевого уровня, **IP** (Internet Protocol — протокол сети Интернет). В отличие от большинства ранних протоколов сетевого уровня, IP с самого начала разрабатывался как протокол межсетевого обмена. Вот как можно описать данный протокол сетевого уровня: его работа заключается в приложении максимума усилий (тем не менее, без всяких гарантий) по транспортировке дейтаграмм от отправителя к получателю независимо от того, находятся эти машины в одной и той же сети или нет.

Соединение в сети Интернет представляет собой следующее. Транспортный уровень берет поток данных и разбивает его на дейтаграммы. Теоретически размер каждой дейтаграммы может достигать 64 Кбайт, однако на практике они обычно не более 1500 байт (укладываются в один кадр Ethernet). Каждая дейтаграмма пересылается по Интернету, возможно, разбиваясь при этом на более мелкие фрагменты, собираемые сетевым уровнем получателя в исходную дейтаграмму. Затем эта дейтаграмма передается транспортному уровню, вставляющему ее во входной поток получающего процесса. На рис. 5.46 видно, что пакет, посланный хостом 1, пересечет на своем пути шесть сетей, прежде чем доберется до хоста 2. На практике промежуточных сетей оказывается гораздо больше.

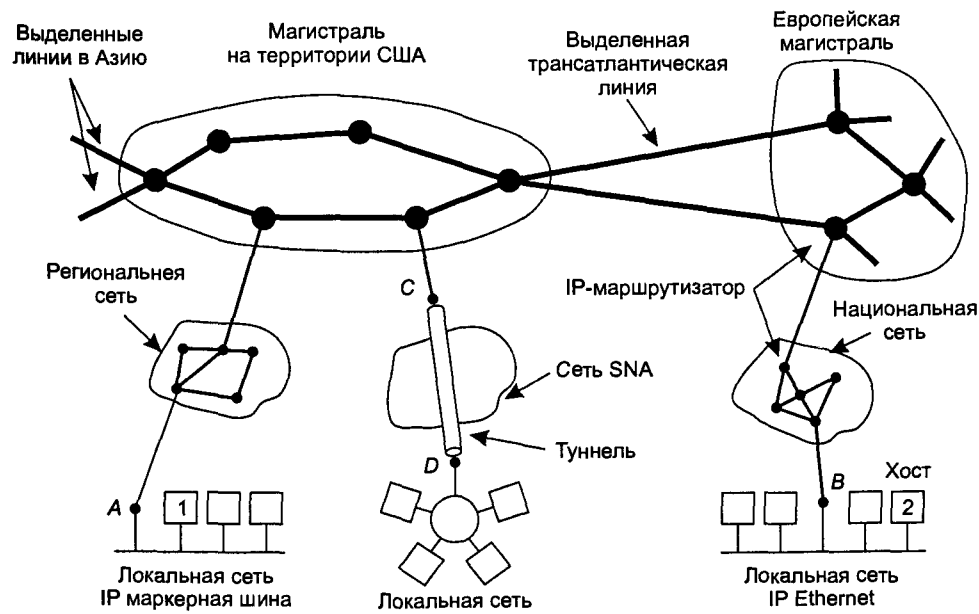


Рис. 5.46. Интернет представляет собой набор соединенных друг с другом сетей

Протокол IP

Начнем изучение сетевого уровня Интернета с формата IP-дейтаграмм. IP-дейтаграмма состоит из заголовка и текстовой части. Заголовок содержит обязательную 20-байтную часть, а также необязательную часть переменной длины. Формат заголовка показан на рис. 5.47. Он передается слева направо, то есть старший бит поля *Версия* передается первым. (В процессоре SPARC байты располагаются слева направо, в процессоре Pentium — наоборот, справа налево.) На машинах, у которых старший байт располагается после младшего, как, например, у семейства процессоров корпорации Intel, требуется программное преобразование как при передаче, так и при приеме.

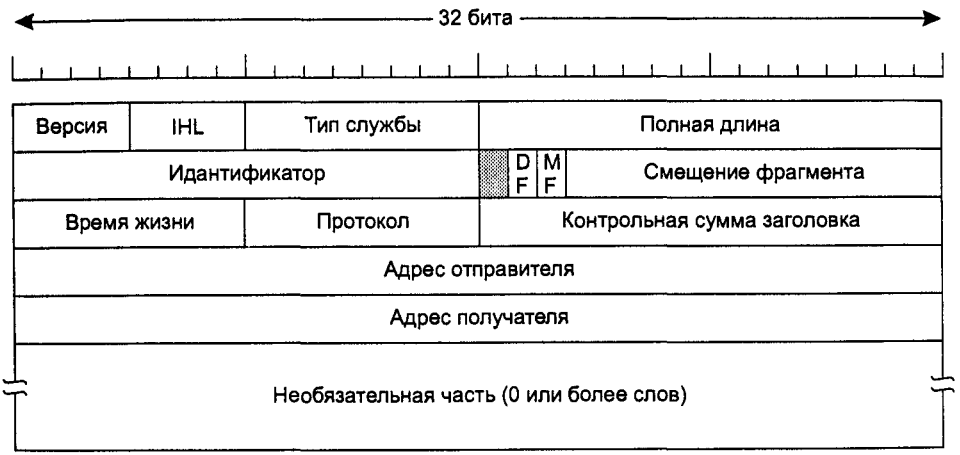


Рис. 5.47. Заголовок IP-дейтаграммы IPv4

Поле *Версия* содержит версию протокола, к которому принадлежит дейтаграмма. Включение версии в каждую дейтаграмму позволяет использовать разные версии протокола на разных машинах. Дело в том, что с годами протокол изменялся, и на одних машинах сейчас работают новые версии, тогда как на других продолжают использоваться старые. Сейчас происходит переход от версии IPv4 к версии IPv6. Он длится уже много лет, и не похоже, что скоро завершится (Durand, 2001; Wiljakka, 2002; Waddington и Chang, 2002). Некоторые даже считают, что это не произойдет никогда (Weiser, 2001). Что касается нумерации, то ничего странного в ней нет, просто в свое время существовал мало кому известный экспериментальный протокол реального масштаба времени IPv5.

Длина заголовка является переменной величиной, для хранения которой выделено поле *IHL* (информация в нем представлена в виде 32-разрядных слов). Минимальное значение длины (при отсутствии необязательного поля) равно 5. Максимальное значение этого 4-битового поля равно 15, что соответствует заголовку длиной 60 байт; таким образом, максимальный размер необязательного поля равен 40 байтам. Для некоторых приложений, например, для записи маршрута, по которому должен быть переслан пакет, 40 байт слишком мало. В данном случае дополнительное поле оказывается бесполезным.

Поле *Тип службы* — единственное поле, смысл которого с годами несколько изменился. Оно было (впрочем, и до сих пор) предназначено для различения классов обслуживания. Возможны разные комбинации надежности и скорости. Для оцифрованного голоса скорость доставки важнее точности. При передаче файла, наоборот, передача без ошибок важнее быстрой доставки.

Изначально 6-разрядное поле *Тип службы* состояло из трехразрядного поля *Precedence* и трех флагов — *D*, *T* и *R*. Поле *Precedence* указывало приоритет, от 0 (нормальный) до 7 (управляющий сетевой пакет). Три флаговых бита позволяли хосту указать, что беспокоит его сильнее всего, выбрав из набора {Delay, Throughput, Reliability} (Задержка, Пропускная способность, Надежность). Теоретически, эти поля позволяют маршрутизаторам выбрать, например, между спутниковой линией с высокой пропускной способностью и большой задержкой и выделенной линией с низкой пропускной способностью и небольшой задержкой. На практике сегодняшние маршрутизаторы часто вообще игнорируют поле *Тип службы*.

Поле *Полная длина* содержит длину всей дейтаграммы, включая как заголовок, так и данные. Максимальная длина дейтаграммы 65 535 байт. В настоящий момент этот верхний предел достаточен, однако с появлением гигабитных сетей могут понадобиться дейтаграммы большего размера.

Поле *Идентификатор* позволяет хосту-получателю определить, какой дейтаграмме принадлежат полученные им фрагменты. Все фрагменты одной дейтаграммы содержат одно и то же значение идентификатора.

Следом идет один неиспользуемый бит и два однобитных поля. Бит *DF* означает Don't Fragment (Не фрагментировать). Это команда маршрутизатору, запрещающая ему фрагментировать дейтаграмму, так как получатель не сможет восстановить ее из фрагментов. Например, при загрузке компьютера его ПЗУ может запросить образ памяти в виде единой дейтаграммы. Пометив дейтаграмму битом *DF*, отправитель гарантирует, что дейтаграмма дойдет единым блоком, даже если для ее доставки придется избегать сетей с маленьким размером пакетов. От всех машин требуется способность принимать фрагменты размером 576 байт и менее.

Бит *MF* означает More Fragments (Продолжение следует). Он устанавливается во всех фрагментах, кроме последнего. По этому биту получатель узнает о прибытии последнего фрагмента дейтаграммы.

Поле *Смещение фрагмента* указывает положение фрагмента в исходной дейтаграмме. Длина всех фрагментов в байтах, кроме длины последнего фрагмента, должна быть кратна 8. Так как на это поле выделено 13 бит, максимальное количество фрагментов в дейтаграмме равно 8192, что дает максимальную длину дейтаграммы 65 536 байт, на 1 байт больше, чем может содержаться в поле *Полная длина*.

Поле *Время жизни* представляет собой счетчик, ограничивающий время жизни пакета. Предполагалось, что он будет отсчитывать время в секундах, таким образом, допуская максимальное время жизни пакета в 255 с. На каждом маршрутизаторе это значение должно было уменьшаться как минимум на единицу плюс время стояния в очереди. Однако на практике этот счетчик просто считает

количество переходов через маршрутизаторы. Когда значение этого поля становится равным нулю, пакет отвергается, а отправителю отсылается пакет с предупреждением. Таким образом удается избежать вечного странствования пакетов, что может произойти, если таблицы маршрутизаторов по какой-либо причине испортятся.

Собрав дейтаграмму из фрагментов, сетевой уровень должен решить, что с ней делать. Поле *Протокол* сообщит ему, какому процессу транспортного уровня ее передать. Это может быть TCP, UDP или что-нибудь еще. Нумерация процессов глобально стандартизирована по всему Интернету. Номера протоколов вместе с некоторыми другими были сведены в RFC 1700, однако теперь доступна интернет-версия в виде базы данных, расположенной по адресу *www.iana.org*.

Поле *Контрольная сумма заголовка* защищает от ошибок только заголовок. Подобная контрольная сумма полезна для обнаружения ошибок, вызванных неисправными микросхемами памяти маршрутизаторов. Алгоритм вычисления суммы просто складывает все 16-разрядные полуслова в дополнительном коде, преобразуя результат также в дополнительный код. Таким образом, проверяемая получателем контрольная сумма заголовка (вместе с этим полем) должна быть равна нулю. Этот алгоритм надежнее, чем обычное суммирование. Обратите внимание на то, что значение *Контрольной суммы заголовка* должно подсчитываться заново на каждом транзитном участке, так как по крайней мере одно поле постоянно меняется (поле *Время жизни*). Для ускорения расчетов применяются некоторые хитрости.

Поля *Адрес отправителя* и *Адрес получателя* указывают номер сети и номер хоста. Интернет-адреса будут обсуждаться в следующем разделе. Поле *Необязательная часть* было создано для того, чтобы с появлением новых вариантов протокола не пришлось вносить в заголовок поля, отсутствующие в нынешнем формате. Оно же может служить пространством для различного рода экспериментов, испытания новых идей. Кроме того, оно позволяет не включать в стандартный заголовок редко используемую информацию. Размер поля *Необязательная часть* может варьироваться. В начале поля всегда располагается однобайтный идентификатор. Иногда за ним может располагаться также однобайтное поле длины, а затем один или несколько информационных байтов. В любом случае, размер поля *Необязательная часть* должен быть кратен 4 байтам. Изначально было определено пять разновидностей этого поля, перечисленных в табл. 5.6, однако с тех пор появилось несколько новых. Текущий полный список имеется в Интернете, его можно найти по адресу *www.iana.org/assignments/ip-parameters*.

Таблица 5.6. Некоторые типы необязательного поля IP-дейтаграммы

Тип	Описание
Безопасность	Указывает уровень секретности дейтаграммы
Свободная маршрутизация от источника	Задает список маршрутизаторов, которые нельзя миновать
Строгая маршрутизация от источника	Задает полный путь следования дейтаграммы

Тип	Описание
Запомнить маршрут	Требует от всех маршрутизаторов добавлять свой IP-адрес
Временной штамп	Требует от всех маршрутизаторов добавлять свой IP-адрес и текущее время

Параметр *Безопасность* указывает уровень секретности дейтаграммы. Теоретически, военный маршрутизатор может использовать это поле, чтобы запретить маршрутизацию дейтаграммы через территорию определенных государств. На практике все маршрутизаторы игнорируют этот параметр, так что его единственное практическое применение состоит в помощи шпионам в поиске ценной информации.

Параметр *Строгая маршрутизация от источника* задает полный путь следования дейтаграммы от отправителя до получателя в виде последовательности IP-адресов. Дейтаграмма обязана следовать именно по этому маршруту. Наибольшая польза этого параметра заключается в том, что с его помощью системный менеджер может послать экстренные пакеты, когда таблицы маршрутизатора повреждены, или замерить временные параметры сети.

Параметр *Свободная маршрутизация от источника* требует, чтобы пакет прошел через указанный список маршрутизаторов в указанном порядке, но при этом по пути он может проходить через любые другие маршрутизаторы. Обычно этот параметр указывает лишь небольшое количество маршрутизаторов. Например, чтобы заставить пакет, посылаемый из Лондона в Сидней, двигаться не на восток, а на запад, можно указать в этом параметре IP-адреса маршрутизаторов в Нью-Йорке, Лос-Анджелесе и Гонолулу. Этот параметр наиболее полезен, когда по политическим или экономическим соображениям следует избегать прохождения пакетов через определенные государства.

Параметр *Запомнить маршрут* требует от всех маршрутизаторов, встречающихся по пути следования пакета, добавлять свой IP-адрес к полю *Необязательная часть*. Этот параметр позволяет системным администраторам вылавливать ошибки в алгоритмах маршрутизации («Ну почему все пакеты, посылаемые из Хьюстона в Даллас, сначала попадают в Токио?»). Когда была создана сеть ARPANET, ни один пакет не проходил больше чем через девять маршрутизаторов, поэтому 40 байт для этого параметра было как раз достаточно. Как уже говорилось, сегодня размер поля *Необязательная часть* оказывается слишком мал.

Наконец, параметр *Временной штамп* действует полностью аналогично параметру *Запомнить маршрут*, но кроме 32-разрядного IP-адреса, каждый маршрутизатор записывает также 32-разрядную запись о текущем времени. Этот параметр также применяется в основном для отладки алгоритмов маршрутизации.

IP-адреса

У каждого хоста и маршрутизатора в Интернете есть IP-адрес, состоящий из номера сети и номера хоста. Эта комбинация уникальна: нет двух машин с одинаковыми IP-адресами. Все IP-адреса имеют длину 32 бита и используются в полях

Адрес отправителя и *Адрес получателя* IP-пакетов. Важно отметить, что IP-адрес, на самом деле, не имеет отношения к хосту. Он имеет отношение к сетевому интерфейсу, поэтому хост, соединенный с двумя сетями, должен иметь два IP-адреса. Однако на практике большинство хостов подключены к одной сети, следовательно, имеют один адрес.

В течение вот уже нескольких десятилетий IP-адреса делятся на пять классов, показанных на рис. 5.48. Такое распределение обычно называется **полноклассовой адресацией**. Сейчас такая адресация уже не используется, но ссылки на нее в литературе встречаются все еще довольно часто. Чуть позже мы обсудим то, что пришло ей на смену.

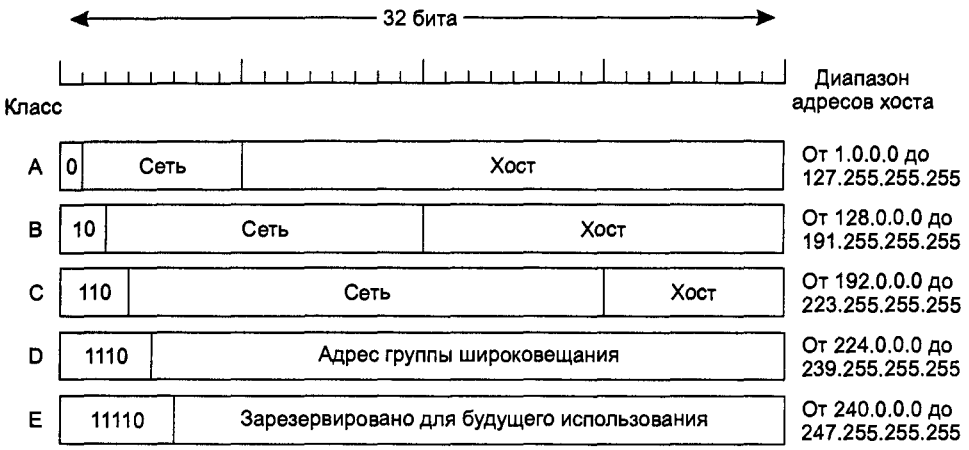


Рис. 5.48. Форматы IP-адреса

Форматы классов A, B, C и D позволяют задавать адреса до 128 сетей с 16 млн хостов в каждой, 16 384 сетей с 64 тысячами хостов или 2 миллионов сетей (например, ЛВС) с 256 хостами (хотя некоторые из них могут быть специализированными). Предусмотрен класс для многоадресной рассылки, при которой дейтаграммы рассылаются одновременно на несколько хостов. Адреса, начинающиеся с 1111, зарезервированы для будущего применения. В настоящее время к Интернету подсоединено более 500 000 сетей, и это число растет с каждым годом. Во избежание конфликтов, номера сетям назначаются некоммерческой **корпорацией по присвоению имен и номеров, ICANN** (Internet Corporation for Assigned Names and Numbers). В свою очередь, ICANN передала полномочия по присвоению некоторых частей адресного пространства региональным органам, занимающимся выделением IP-адресов провайдерам и другим компаниям.

Сетевые адреса, являющиеся 32-разрядными числами, обычно записываются в виде четырех десятичных чисел, которые соответствуют отдельным байтам, разделенных точками. Например, шестнадцатеричный адрес C0290614 записывается как 192.41.6.20. Наименьший IP-адрес равен 0.0.0.0, а наибольший — 255.255.255.255.

Как видно из рис. 5.49, числа 0 и –1 (единицы во всех разрядах) имеют особое назначение. Число 0 означает эту сеть или этот хост. Значение –1 используется для широковещания и означает все хосты указанной сети.



Рис. 5.49. Специальные IP-адреса

IP-адрес 0.0.0.0 используется хостом только при загрузке. IP-адреса с нулевым номером сети обозначают текущую сеть. Эти адреса позволяют машинам обращаться к хостам собственной сети, не зная ее номера (но они должны знать ее класс и количество используемых нулей). Адрес, состоящий только из единиц, обеспечивает широковещание в пределах текущей (обычно локальной) сети. Адреса, в которых указана сеть, но в поле номера хоста одни единицы, обеспечивают широковещание в пределах любой удаленной локальной сети, соединенной с Интернетом. Наконец, все адреса вида 127.xx.yy.zz зарезервированы для тестирования сетевого программного обеспечения методом обратной передачи. Отправляемые по этому адресу пакеты не попадают на линию, а обрабатываются локально как входные пакеты. Это позволяет пакетам перемещаться по локальной сети, когда отправитель не знает номера.

Подсети

Как было показано ранее, у всех хостов сети должен быть один и тот же номер сети. Это свойство IP-адресации может вызвать проблемы при росте сети. Например, представьте, что университет создал сеть класса B, используемую факультетом информатики в качестве Ethernet. Год спустя факультету электротехники понадобилось подключиться к Интернету, для чего был куплен повторитель для расширения сети факультета информатики и проложен кабель из его здания. Однако время шло, число компьютеров в сети росло, и четырех повторителей (максимальный предел для сети Ethernet) стало не хватать. Понадобилось создание новой архитектуры.

Получить второй сетевой адрес университету довольно сложно, потому что сетевые адреса — ресурс дефицитный, к тому же в одном сетевом адресе адресного пространства достаточно для подключения более 60 000 хостов. Проблема заключается в следующем: правилом установлено, что адрес одного класса (A, B или C) относится только к одной сети, а не к набору ЛВС. С этим столкнулось

множество организаций, в результате чего были произведены небольшие изменения в системе адресации.

Проблема решилась предоставлением сети возможности разделения на несколько частей с точки зрения внутренней организации. При этом с точки зрения внешнего представления сеть могла оставаться единой сущностью. Типичная сеть университетского городка в наши дни выглядит так, как показано на рис. 5.50. Здесь главный маршрутизатор соединен с провайдером или региональной сетью, а на каждом факультете может быть установлена своя локальная сеть Ethernet. Все сети Ethernet с помощью своих маршрутизаторов соединяются с главным маршрутизатором университетской сети (возможно, с помощью магистральной ЛВС, однако здесь важен сам принцип межмаршрутизаторной связи).

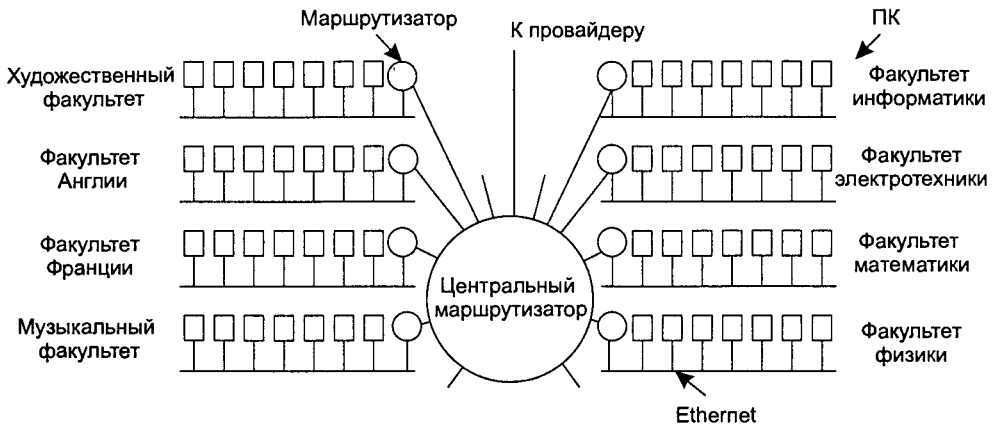


Рис. 5.50. Университетская сеть, состоящая из ЛВС разных факультетов

В литературе, посвященной интернет-технологиям, части сети называются **подсетями**. Как уже упоминалось в главе 1, подобное использование этого термина конфликтует со старым понятием «подсети», обозначающим множество всех маршрутизаторов и линий связи в сети. К счастью, по контексту обычно бывает ясно, какой смысл вкладывается в это слово. По крайней мере, в данном разделе «подсеть» будет употребляться только в новом значении.

Как центральный маршрутизатор узнает, в какую из подсетей (Ethernet) направить пришедший пакет? Одним из способов является поддержание маршрутизатором таблицы из 65 536 записей, говорящих о том, какой из маршрутизаторов использовать для доступа к каждому из хостов. Эта идея будет работать, но потребуется очень большая таблица и много операций по ее обслуживанию, выполняемых вручную, при добавлении, перемещении и удалении хостов.

Была изобретена альтернативная схема работы. Вместо одного адреса класса B с 14 битами для номера сети и 16 битами для номера хоста было предложено использовать несколько другой формат — формировать адрес подсети из нескольких битов. Например, если в университете существует 35 подразделений, то 6-битным номером можно кодировать подсети, а 10-битным — номера хостов. С помощью такой адресации можно организовать до 64 сетей Ethernet по

1022 хоста в каждой (адреса 0 и –1 не используются, как уже говорилось, поэтому не 1024 (2^{10}), а именно 1022 хоста). Такое разбиение может быть изменено, если окажется, что оно не очень подходит.

Все, что нужно маршрутизатору для реализации подсети, это наложить **маску подсети**, показывающую разбиение адреса на номер сети, подсети и хоста (рис. 5.51). Маски подсетей также записываются в виде десятичных чисел, разделенных точками, с добавлением косой черты, за которой следует число битов номера сети и подсети. Например, на рис. 5.51 маску подсети можно записать в виде 255.255.252.0. Альтернативная запись будет включать /22, показывая, что маска подсети занимает 22 бита.



Рис. 5.51. Сеть класса B, разделенная на 64 подсети

За пределами сети разделение на подсети незаметно, поэтому нет нужды с появлением каждой подсети обращаться в ICANN или изменять какие-либо внешние базы данных. В данном примере первая подсеть может использовать IP-адреса, начиная с 130.50.4.1; вторая — начиная с 130.50.8.1; третья — 130.50.12.1, и т. д. Чтобы понять, почему на каждую подсеть уходит именно четыре единицы в адресе, вспомните двоичную запись этих адресов:

```
Подсеть 1: 10000010 00110010 000001|00 00000001
Подсеть 2: 10000010 00110010 000010|00 00000001
Подсеть 3: 10000010 00110010 000011|00 00000001
```

Здесь вертикальная черта (|) показывает границу номера подсети и хоста. Слева расположен 6-битный номер подсети, справа — 10-битный номер хоста.

Чтобы понять, как функционируют подсети, следует рассмотреть процесс обработки IP-пакетов маршрутизатором. У каждого маршрутизатора есть таблица, содержащая IP-адреса сетей (вида *<сеть, 0>*) и IP-адреса хостов (вида *<эта_сеть, хост>*). Адреса сетей позволяют получать доступ к удаленным сетям, а адреса хостов — обращаться к локальным хостам. С каждой таблицей связан сетевой интерфейс, применяющийся для получения доступа к пункту назначения, а также другая информация.

Когда IP-пакет прибывает на маршрутизатор, адрес получателя, указанный в пакете, ищется в таблице маршрутизации. Если пакет направляется в удаленную сеть, он пересылается следующему маршрутизатору по интерфейсу, указанному в таблице. Если пакет предназначен локальному хосту (например, в локальной сети маршрутизатора), он посылается напрямую адресату. Если номера сети, в которую посылается пакет, в таблице маршрутизатора нет, пакет пересылается маршрутизатору *по* умолчанию, с более подробными таблицами. Такой алгоритм

означает, что каждый маршрутизатор должен учитывать только другие сети и локальные хосты, а не пары *<сеть, хост>*, что значительно уменьшает размер таблиц маршрутизатора.

При разбиении сети на подсети таблицы маршрутизации меняются — добавляются записи вида *<эта_сеть, подсеть, 0>* и *<эта_сеть, эта_подсеть, хост>*. Таким образом, маршрутизатор подсети *k* знает, как получить доступ ко всем другим подсетям и как добраться до всех хостов своей подсети. Ему нет нужды знать детали адресации хостов в других подсетях. На самом деле, все, что для этого требуется от маршрутизатора, это выполнить двоичную операцию И над маской подсети, чтобы избавиться от номера хоста, а затем найти получившийся адрес в таблицах (после определения класса сети). Например, пакет, адресованный хосту с IP-адресом 130.50.15.6 и прибывающий на центральный маршрутизатор, после выполнения операции И с маской 255.255.252.0/22 получает адрес 130.50.12.0. Это значение ищется в таблицах маршрутизации, и с его помощью определяется выходная линия маршрутизатора к подсети 3. Итак, разбиение на подсети уменьшает объем таблиц маршрутизаторов путем создания трехуровневой иерархии, состоящей из сети, подсети и хоста.

CIDR — бесклассовая междоменная маршрутизация

В течение многих лет IP оставался чрезвычайно популярным протоколом. Он работал просто прекрасно, и основное подтверждение тому - экспоненциальный рост сети Интернет. К сожалению, протокол IP скоро стал жертвой собственной популярности: стало подходить к концу адресное пространство. Эта угроза вызвала бурю дискуссий и обсуждений в Интернет-сообществе. В этом разделе мы опишем как саму проблему, так и некоторые предложенные способы ее решения.

В теперь уже далеком 1987 году некоторые дальновидные люди предсказывали, что настанет день, когда в Интернете будет 100 000 сетей. Большинство экспертов посмеивались над этим и говорили, что если такое когда-нибудь и произойдет, то не раньше, чем через много десятков лет. Стотысячная сеть была подключена к Интернету в 1996 году. И, как уже было сказано, нависла угроза выхода за пределы пространства IP-адресов. В принципе, существует 2 миллиарда адресов, но на практике благодаря иерархической организации адресного пространства (см. рис. 5.48) это число сократилось на миллионы. В частности, одним из виновников этого является класс сетей B. Для большинства организаций класс A с 16 миллионами адресов — это слишком много, а класс C с 256 адресами — слишком мало. Класс B с 65 536 адресами — это то, что нужно. В интернет-фольклоре такая дилемма известна под названием **проблемы трех медведей** (вспомним *Машу и трех медведей*).

На самом деле и класс B слишком велик для большинства контор, которые устанавливают у себя сети. Исследования показали, что более чем в половине случаев сети класса B включают в себя менее 50 хостов. Безо всяких сомнений, всем этим организациям хватило бы и сетей класса C, однако почему-то все уверены, что в один прекрасный день маленькое предприятие вдруг разрастется настолько, что сеть выйдет за пределы 8-битного адресного пространства хостов. Сейчас, оглядываясь назад, кажется, что лучше было бы использовать в классе C

10-битную адресацию (до 1022 хостов в сети). Если бы это было так, то, возможно, большинство организаций приняло бы разумное решение устанавливать у себя сети класса C, а не B. Таких сетей могло бы быть полмиллиона, а не 16 384, как в случае сетей класса B.

Нельзя обвинять в создавшейся ситуации проектировщиков Интернета за то, что они не увеличили (или не уменьшили) адресное пространство сетей класса B. В то время, когда принималось решение о создании трех классов сетей, Интернет был инструментом научно-исследовательских организаций США (плюс несколько компаний и военных организаций, занимавшихся исследованиями с помощью сети). Никто тогда не предполагал, что Интернет станет коммерческой системой коммуникации общего пользования, соперничающей с телефонной сетью. Тогда кое-кто сказал, ничуть не сомневаясь в своей правоте: «В США около 2000 колледжей и университетов. Даже если все они подключатся к Интернету и к ним присоединятся университеты из других стран, мы никогда не превысим число 16 000, потому что высших учебных заведений по всему миру не так уж много. Зато мы будем кодировать номер хоста целым числом байт, что ускорит процесс обработки пакетов».

Даже если выделить 20 бит под адрес сети класса B, возникнет другая проблема: разрастание таблиц маршрутизации. С точки зрения маршрутизаторов, адресное пространство IP представляет собой двухуровневую иерархию, состоящую из номеров сетей и номеров хостов. Маршрутизаторы не обязаны знать номера вообще всех хостов, но им необходимо знать номера всех сетей. Если бы использовалось полмиллиона сетей класса C, каждому маршрутизатору пришлось бы хранить в таблице полмиллиона записей, по одной для каждой сети, в которых сообщалось бы о том, какую выходную линию использовать, чтобы добраться до той или иной сети, а также о чем-нибудь еще.

Физическое хранение полумиллиона строк таблицы, вероятно, выполнимо, хотя и дорого для маршрутизаторов, хранящих таблицы в статической памяти плат ввода-вывода. Более серьезная проблема состоит в том, что сложность обработки этих таблиц растет быстрее, чем сами таблицы, то есть зависимость между ними не линейная. Кроме того, большая часть имеющихся программных и программно-аппаратных средств маршрутизаторов разрабатывалась в те времена, когда Интернет объединял 1000 сетей, а 10 000 сетей казались отдаленным будущим. Методы реализации тех лет в настоящее время далеки от оптимальных.

Различные алгоритмы маршрутизации требуют от каждого маршрутизатора периодической рассылки своих таблиц (например, протоколов векторов расстояний). Чем больше будет размер этих таблиц, тем выше вероятность потери части информации по дороге, что будет приводить к неполноте данных и возможной нестабильности работы алгоритмов выбора маршрутов.

Проблема таблиц маршрутизаторов может быть решена при помощи увеличения числа уровней иерархии. Например, если бы каждый IP-адрес содержал поля страны, штата, города, сети и номера хоста. В таком случае каждому маршрутизатору нужно будет знать, как добраться до каждой страны, до каждого штата или провинции своей страны, каждого города своей провинции или штата и до

каждой сети своего города. К сожалению, такой подход потребует существенно более 32 бит для адреса, а адресное поле будет использоваться неэффективно (для княжества Лихтенштейн будет выделено столько же разрядов, сколько для Соединенных Штатов).

Таким образом, большая часть решений разрешает одну проблему, но взамен создает новую. Одним из решений, реализуемым в настоящий момент, является алгоритм маршрутизации **CIDR** (Classless InterDomain Routing — бесклассовая междоменная маршрутизация). Идея маршрутизации CIDR, описанной в RFC 1519, состоит в объединении оставшихся адресов в блоки переменного размера, независимо от класса. Если кому-нибудь требуется, скажем, 2000 адресов, ему выделяется блок из 2048 адресов на границе, кратной 2048 байтам.

Отказ от классов усложнил процесс маршрутизации. В старой системе, построенной на классах, маршрутизация происходила следующим образом. По прибытии пакета на маршрутизатор копия IP-адреса, извлеченного из пакета и сдвинутого вправо на 28 бит, давала 4-битный номер класса. С помощью 16-альтернативного ветвления пакеты рассортировывались на A, B, C и D (если этот класс поддерживался): восемь случаев было зарезервировано для A, четыре для B, два для C и по одному для D и E. Затем при помощи маскировки по коду каждого класса определялся 8-, 16- или 32-битный сетевой номер, который и записывался с выравниванием по правым разрядам в 32-битное слово. Сетевой номер отыскивался в таблице A, B или C, причем для A и B применялась индексация, а для C — хэш-функция. По найденной записи определялась выходная линия, по которой пакет и отправлялся в дальнейшее путешествие.

В CIDR этот простой алгоритм применить не удается. Вместо этого применяется расширение всех записей таблицы маршрутизации за счет добавления 32-битной маски. Таким образом, образуется единая таблица для всех сетей, состоящая из набора троек (IP-адрес, маска подсети, исходящая линия). Что происходит с приходящим пакетом при применении метода CIDR? Во-первых, извлекается IP-адрес назначения. Затем (концептуально) таблица маршрутизации сканируется запись за записью, адрес назначения маскируется и сравнивается со значениями записей. Может оказаться, что по значению подойдет несколько записей (с разными длинами масок подсети). В этом случае используется самая длинная маска. То есть если найдено совпадение для маски /20 и /24, то будет выбрана запись, соответствующая /24.

Был разработан сложный алгоритм для ускорения процесса поиска адреса в таблице (Ruiz-Sanchez и др., 2001). В маршрутизаторах, предполагающих коммерческое использование, применяются специальные чипы VLSI, в которые данные алгоритмы встроены аппаратно.

Чтобы лучше понять алгоритм маршрутизации, рассмотрим пример. Допустим, имеется набор из миллионов адресов, начиная с 194.24.0.0. Допустим также, что Кембриджскому университету требуется 2048 адресов, и ему выделяются адреса от 194.24.0.0 до 194.24.7.255, а также маска 255.255.248.0. Затем Оксфордский университет запрашивает 4096 адресов. Так как блок из 4096 адресов должен располагаться на границе, кратной 4096, то ему не могут быть выделены адреса начиная с 194.24.8.0. Вместо этого он получает адреса от 194.24.16.0 до

194.24.31.255 вместе с маской 255.255.240.0. Затем Эдинбургский университет просит выделить ему 1024 адреса и получает адреса от 194.24.8.0 до 194.24.11.255 и маску 255.255.252.0. Все эти присвоенные адреса и маски сведены в табл. 5.7.

Таблица 5.7. Набор присвоенных IP-адресов

Университет	Первый адрес	Последний адрес	Количество	Форма записи
Кембридж	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Эдинбург	194.24.8.0	94.24.11.255	1024	194.24.8.0/22
(Свободно)	194.24.12.0	94.24.15.255	1024	194.24.12.0/22
Оксфорд	194.24.16.0	94.24.16.255	4096	194.24.16.0/20

После этого таблицы маршрутизаторов по всему миру получают три новые строки, содержащие базовый адрес и маску. В двоичном виде эти записи выглядят так:

Адрес	Маска
К: 11000010 00011000 00000000 00000000	11111111 11111111 11111000 00000000
Э: 11000010 00011000 00001000 00000000	11111111 11111111 11111100 00000000
О: 11000010 00011000 00010000 00000000	11111111 11111111 11110000 00000000

Теперь посмотрим, что произойдет, когда пакет придет по адресу 194.24.17.4. В двоичном виде этот адрес представляет собой следующую 32-битную строку:

11000010 00011000 00010001 00000100

Сначала на него накладывается (выполняется логическое И) маска Кембриджа, в результате чего получается

11000010 00011000 00010000 00000000

Это значение не совпадает с базовым адресом Кембриджа, поэтому на оригинальный адрес накладывается маска Оксфорда, что дает в результате

11000010 00011000 00010000 00000000

Это значение совпадает с базовым адресом Оксфорда. Если далее по таблице совпадений нет, то пакет пересылается Оксфордскому маршрутизатору.

Теперь посмотрим на эту троицу университетов с точки зрения маршрутизатора в Омахе, штат Небраска, у которого есть всего четыре выходных линии: на Миннеаполис, Нью-Йорк, Даллас и Денвер. Получив три новых записи, маршрутизатор понимает, что он может скомпоновать их вместе и получить **одну агрегированную запись**, состоящую из адреса 194.24.0.0/19 и подмаски:

11000010 00000000 00000000 00000000 11111111 11111111 11100000 00000000

В соответствии с этой записью пакеты, предназначенные для любого из трех университетов, отправляются в Нью-Йорк. Объединив три записи, маршрутизатор в Омахе уменьшил размер своей таблицы на две строки.

В Нью-Йорке весь трафик Великобритании направляется по лондонской линии, поэтому там также используется агрегированная запись. Однако если имеются отдельные линии в Лондон и Эдинбург, тогда необходимы три отдельные записи. Агрегация часто используется в Интернете для уменьшения размеров таблиц маршрутизации.

И последнее замечание по данному примеру. Та же самая агрегированная запись маршрутизатора в Омахе используется для отправки пакетов по не присвоенным адресам в Нью-Йорк. До тех пор, пока адреса остаются не присвоенными, это не имеет значения, поскольку предполагается, что они вообще не встретятся. Однако если в какой-то момент этот диапазон адресов будет присвоен какой-нибудь калифорнийской компании, для его обработки понадобится новая запись: 194.24.12.0/22.

NAT — трансляция сетевого адреса

IP-адреса являются дефицитным ресурсом. У провайдера может быть /16-адрес (бывший класс B), дающий возможность подключить 65 534 хоста. Если клиентов становится больше, начинают возникать проблемы. Хостам, подключающимся к Интернету время от времени по обычной телефонной линии, можно выделять IP-адреса динамически, только на время соединения. Тогда один /16-адрес будет обслуживать до 65 534 *активных* пользователей, и этого, возможно, будет достаточно для провайдера, у которого несколько сотен тысяч клиентов. Когда сессия связи завершается, IP-адрес присваивается новому соединению. Такая стратегия может решить проблемы провайдеров, имеющих не очень большое количество частных клиентов, соединяющихся по телефонной линии, однако не поможет провайдерам, большую часть клиентуры которых составляют организации.

Дело в том, что корпоративные клиенты предпочитают иметь постоянное соединение с Интернетом, по крайней мере в течение рабочего дня. И в маленьких конторах, например туристических агенствах, состоящих из трех сотрудников, и в больших корпорациях имеются локальные сети, состоящие из некоторого числа компьютеров. Некоторые компьютеры являются рабочими станциями сотрудников, некоторые служат веб-серверами. В общем случае имеется маршрутизатор ЛВС, соединенный с провайдером по выделенной линии для обеспечения постоянного подключения. Такое решение означает, что с каждым компьютером целый день связан один IP-адрес. Вообще-то даже все вместе взятые компьютеры, имеющиеся у корпоративных клиентов, не могут перекрыть имеющиеся у провайдера IP-адреса. Для адреса длины /16 этот предел равен, как мы уже отмечали, 65 534. Однако если у поставщика услуг Интернета число корпоративных клиентов исчисляется десятками тысяч, то этот предел будет достигнут очень быстро.

Проблема усугубляется еще и тем, что все большее число частных пользователей желают иметь ADSL или кабельное соединение с Интернетом. Особенности этих способов заключаются в следующем: а) пользователи получают постоянный IP-адрес; б) отсутствует повременная оплата (взимается только ежемесячная абонентская плата). Пользователи такого рода услуг имеют постоянное подключение к Интернету. Развитие в данном направлении приводит к возрастанию дефицита IP-адресов. Присваивать IP-адреса «на лету», как это делается при телефонном подключении, бесполезно, потому что число активных адресов в каждый момент времени может быть во много раз больше, чем имеется у провайдера.

Часто ситуация еще больше усложняется за счет того, что многие пользователи ADSL и кабельного Интернета имеют дома два и более компьютера (например, по одному на каждого члена семьи) и хотят, чтобы все машины имели выход в Интернет. Что же делать — ведь есть только один IP-адрес, выданный провайдером! Решение таково: необходимо установить маршрутизатор и объединить все компьютеры в локальную сеть. С точки зрения провайдера, в этом случае семья будет выступать в качестве аналога маленькой фирмы с несколькими компьютерами. Добро пожаловать в корпорацию Васильевых!

Проблема дефицита IP-адресов отнюдь не теоретическая и отнюдь не относится к отдаленному будущему. Она уже актуальна, и бороться с ней приходится здесь и сейчас. Долговременный проект предполагает тотальный перевод всего Интернета на протокол IPv6 со 128-битной адресацией. Этот переход действительно постепенно происходит, но процесс идет настолько медленно, что затягивается на годы. Видя это, многие поняли, что нужно срочно найти какое-нибудь решение хотя бы на ближайшее время. Такое решение было найдено в виде метода **трансляции сетевого адреса, NAT** (Network Address Translation), описанного в RFC 3022. Суть его мы рассмотрим позже, а более подробную информацию можно найти в (Dutcher, 2001).

Основная идея трансляции сетевого адреса состоит в присвоении каждой фирме одного IP-адреса (или, по крайней мере, небольшого числа адресов) для интернет-трафика. *Внутри* фирмы каждый компьютер получает уникальный IP-адрес, используемый для маршрутизации внутреннего трафика. Однако как только пакет покидает пределы здания фирмы и направляется к провайдеру, выполняется трансляция адреса. Для реализации этой схемы было создано три диапазона так называемых частных IP-адресов. Они могут использоваться внутри компании по ее усмотрению. Единственное ограничение заключается в том, что пакеты с такими адресами ни в коем случае не должны появляться в самом Интернете. Вот эти три зарезервированных диапазона:

10.0.0.0	– 10.255.255.255/8	(16 777 216 хостов)
172.16.0.0	– 172.31.255.255/12	(1 048 576 хостов)
192.168.0.0	– 192.168.255.255/16	(65 536 хостов)

Итак, первый диапазон может обеспечить адресами 16 777 216 хостов (кроме 0 и –1, как обычно), и именно его обычно предпочитают компании, даже если им на самом деле столько внутренних адресов и не требуется.

Работа метода трансляции сетевых адресов показана на рис. 5.52. В пределах территории компании у каждой машины имеется собственный уникальный адрес вида 10.*x.y.z*. Тем не менее, когда пакет выходит за пределы владений компании, он проходит через **NAT-блок**, транслирующий внутренний IP-адрес источника (10.0.0.1 на рисунке) в реальный IP-адрес, полученный компанией от провайдера (198.60.42.12 для нашего примера). NAT-блок обычно представляет собой единое устройство с брандмауэром, обеспечивающим безопасность путем строго отслеживания входящего и исходящего трафика компании. Брандмауэры мы будем изучать отдельно в главе 8. NAT-блок может быть интегрирован и с маршрутизатором компании.

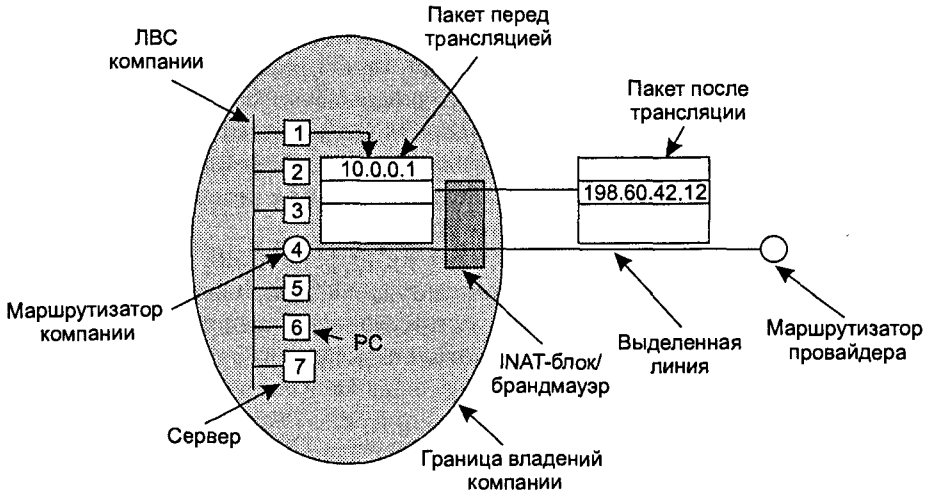


Рис. 5.52. Расположение и работа NAT-блока

Мы до сих пор обходили одну маленькую деталь: когда приходит ответ на запрос (например, от веб-сервера), он ведь адресуется 198.60.42.12. Как же NAT-блок узнает, каким внутренним адресом заменить общий адрес компании? Вот в этом и состоит главная проблема использования трансляции сетевых адресов. Если бы в заголовке IP-пакета было свободное поле, его можно было бы использовать для запоминания адреса того, кто посылал запрос. Но в заголовке остается неиспользованным всего один бит. В принципе, можно было бы создать такое поле для истинного адреса источника, но это потребовало бы изменения IP-кода на всех машинах по всему Интернету. Это не лучший выход, особенно если мы хотим найти быстрое решение проблемы нехватки IP-адресов.

На самом деле произошло вот что. Разработчики NAT подметили, что большая часть полезной нагрузки IP-пакетов — это либо TCP, либо UDP. Когда мы будем в главе 6 рассматривать TCP и UDP, мы увидим, что оба формата имеют заголовки, содержащие номера портов источника и приемника. Далее мы обсудим, что значит порт TCP, но надо иметь в виду, что с портами UDP связана точно такая же история. Номера портов представляют собой 16-разрядные целые числа, показывающие, где начинается и где заканчивается TCP-соединение. Место хранения номеров портов используется в качестве поля, необходимого для работы NAT.

Когда процесс желает установить TCP-соединение с удаленным процессом, он связывается со свободным TCP-портом на собственном компьютере. Этот порт становится **портом источника**, который сообщает TCP-коду информацию о том, куда направлять пакеты данного соединения. Процесс также определяет **порт назначения**. Посредством порта назначения сообщается, кому отдать пакет на удаленной стороне. Порты с 0 по 1023 зарезервированы для хорошо известных сервисов. Например, 80-й порт используется веб-серверами, соответственно, на них могут ориентироваться удаленные клиенты. Каждое исходящее сообщение

TCP содержит информацию о порте источника и порте назначения. Вместе они служат для идентификации процессов на обоих концах, использующих соединение.

Проведем аналогию, которая несколько прояснит принцип использования портов. Допустим, у компании есть один общий телефонный номер. Когда люди набирают его, они слышат голос оператора, который спрашивает, с кем именно они хотели бы соединиться, и подключают их к соответствующему добавочному телефонному номеру. Основной телефонный номер является аналогией IP-адреса компании, а добавочные на обоих концах аналогичны портам. Для адресации портов используется 16-битное поле, которое идентифицирует процесс, получающий входящий пакет.

С помощью поля *Порт источника* мы можем решить проблему отображения адресов. Когда исходящий пакет приходит в NAT-блок, адрес источника вида 10.*x.y.z* заменяется настоящим IP-адресом. Кроме того, поле *Порт источника* TCP заменяется индексом таблицы перевода NAT-блока, содержащей 65 536 записей. Каждая запись содержит исходный IP-адрес и номер исходного порта. Наконец, пересчитываются и вставляются в пакет контрольные суммы заголовков TCP и IP. Необходимо заменять поле *Порт источника*, потому что машины с местными адресами 10.0.0.1 и 10.0.0.2 могут случайно пожелать воспользоваться одним и тем же портом (5000-м, например). Так что для однозначной идентификации процесса отправителя одного поля *Порт источника* оказывается недостаточно.

Когда пакет прибывает на NAT-блок со стороны провайдера, извлекается значение поля *Порт источника* заголовка TCP. Оно используется в качестве индекса таблицы отображения NAT-блока. По найденной в этой таблице записи определяются внутренний IP-адрес и настоящий *Порт источника* TCP. Эти два значения вставляются в пакет. Затем заново подсчитываются контрольные суммы TCP и IP. Пакет передается на главный маршрутизатор компании для нормальной доставки с адресом вида 10.*x.y.z*.

В случае применения ADSL или кабельного Интернета трансляция сетевых адресов может применяться для облегчения борьбы с нехваткой адресов. Присваиваемые пользователям адреса имеют вид 10.*x.y.z*. Как только пакет покидает пределы владений провайдера и уходит в Интернет, он попадает в NAT-блок, который преобразует внутренний адрес в реальный IP-адрес провайдера. На обратном пути выполняется обратная операция. В этом смысле для всего остального Интернета провайдер со своими клиентами, использующими ADSL и кабельное соединение, представляется в виде одной большой компании.

Хотя описанная выше схема частично решает проблему нехватки IP-адресов, многие приверженцы IP рассматривают NAT как некую заразу, распространяющуюся по Земле. И их можно понять.

Во-первых, сам принцип трансляции сетевых адресов никак не вписывается в архитектуру IP, которая подразумевает, что каждый IP-адрес уникальным образом идентифицирует только одну машину в мире. Вся программная структура Интернета построена на использовании этого факта. При трансляции сетевых адресов получается, что тысячи машин могут (и так происходит в действительности) иметь адрес 10.0.0.1.

Во-вторых, NAT превращает Интернет из сети без установления соединения в нечто подобное сети, ориентированной на соединение. Проблема в том, что NAT-блок должен поддерживать таблицу отображения для всех соединений, проходящих через него. Запоминать состояние соединения — дело сетей, ориентированных на соединение, но никак не сетей без установления соединений. Если NAT-блок ломается и теряются его таблицы отображения, то про все TCP-соединения, проходящие через него, можно забыть. При отсутствии трансляции сетевых адресов выход из строя маршрутизатора не оказывает никакого эффекта на деятельность TCP. Отправляющий процесс просто выжидает несколько секунд и посылает заново все неподтвержденные пакеты. При использовании NAT Интернет становится таким же восприимчивым к сбоям, как сеть с коммутацией каналов.

В-третьих, NAT нарушает одно из фундаментальных правил построения многоуровневых протоколов: уровень k не должен строить никаких предположений относительно того, что именно уровень $k + 1$ поместил в поле полезной нагрузки. Этот принцип определяет независимость уровней друг от друга. Если когда-нибудь на смену TCP придет TCP-2, у которого будет другой формат заголовка (например, 32-битная адресация портов), то трансляция сетевых адресов потерпит фиаско. Вся идея многоуровневых протоколов состоит в том, чтобы изменения в одном из уровней никак не могли повлиять на остальные уровни. NAT разрушает эту независимость.

В-четвертых, процессы в Интернете вовсе не обязаны использовать только TCP или UDP. Если пользователь машины *A* решит придумать новый протокол транспортного уровня для общения с пользователем машины *B* (это может быть сделано, например, для какого-нибудь мультимедийного приложения), то ему придется как-то бороться с тем, что NAT-блок не сможет корректно обработать поле *Порт источника* TCP.

В-пятых, некоторые приложения вставляют IP-адреса в текст сообщений. Получатель извлекает их оттуда и затем обрабатывает. Так как NAT не знает ничего про такой способ адресации, он не сможет корректно обработать пакеты, и любые попытки использования этих адресов удаленной стороной приведут к неудаче. **Протокол передачи файлов, FTP** (File Transfer Protocol), использует именно такой метод и может отказаться работать при трансляции сетевых адресов, если только не будут приняты специальные меры. Протокол интернет-телефонии H.323 (мы будем изучать его в главе 7) также обладает подобным свойством. Можно улучшить метод NAT и заставить его корректно работать с H.323, но невозможно же дорабатывать его всякий раз, когда появляется новое приложение.

В-шестых, поскольку поле *Порт источника* является 16-разрядным, то на один IP-адрес может быть отображено примерно 65 536 местных адресов машин. На самом деле это число несколько меньше: первые 4096 портов зарезервированы для служебных нужд. В общем, если есть несколько IP-адресов, то каждый из них может поддерживать до 61 440 местных адресов.

Эти и другие проблемы, связанные с трансляцией сетевых адресов, обсуждаются в RFC 2993. Обычно противники использования NAT говорят, что решение проблемы нехватки IP-адресов путем создания временной уродливой заплатки

только мешает процессу настоящей эволюции, заключающемуся в переходе на IPv6. Поэтому NAT — это не добро, а зло для Интернета.

Управляющие протоколы Интернета

Помимо протокола IP, используемого для передачи данных, в Интернете есть несколько управляющих протоколов, применяемых на сетевом уровне, к которым относятся ICMP, ARP, RARP, BOOTP и DHCP. В данном разделе мы рассмотрим их все по очереди.

ICMP — протокол управляющих сообщений Интернета

За работой Интернета следят маршрутизаторы. Когда случается что-то неожиданное, о происшествии сообщается по протоколу **ICMP** (Internet Control Message Protocol — протокол управляющих сообщений Интернета), используемому также для тестирования Интернета. Протоколом ICMP определено около дюжины типов сообщений. Наиболее важные из них приведены в табл. 5.8. Каждое ICMP-сообщение вкладывается в IP-пакет.

Таблица 5.8. Основные типы ICMP-сообщений

Тип сообщения	Описание
Адресат недоступен	Пакет не может быть доставлен
Время истекло	Время жизни пакета упало до нуля
Проблема с параметром	Неверное поле заголовка
Гашение источника	Сдерживающий пакет
Переадресовать	Научить маршрутизатор географии
Запрос отклика	Спросить машину, жива ли она
Отклик	Да, я жива
Запрос временного штампа	То же, что и Запрос отклика, но с временным штампом
Отклик с временным штампом	То же, что и Отклик, но с временным штампом

Сообщение АДРЕСАТ НЕДОСТУПЕН используется, когда подсеть или маршрутизатор не могут обнаружить пункт назначения или когда пакет с битом *DF* (не фрагментировать) не может быть доставлен, так как путь ему преграждает сеть с маленьким размером пакетов.

Сообщение ВРЕМЯ ИСТЕКЛО посылается, когда пакет игнорируется, так как его счетчик уменьшился до нуля. Это событие является признаком того, что пакеты двигаются по замкнутым контурам, что имеется большая перегрузка или установлено слишком низкое значение таймера.

Сообщение ПРОБЛЕМА ПАРАМЕТРА указывает на то, что обнаружено неверное значение в поле заголовка. Это является признаком наличия ошибки в программном обеспечении хоста, отправившего этот пакет, или промежуточного маршрутизатора.

Сообщение ГАШЕНИЕ ИСТОЧНИКА ранее использовалось для усмирения хостов, которые отправляли слишком много пакетов. Хост, получивший такое сообщение,

должен был снизить обороты. В настоящее время подобное сообщение редко используется, так как при возникновении перегрузки подобные пакеты только подливают масла в огонь, еще больше загружая сеть. Теперь борьба с перегрузкой в Интернете осуществляется в основном на транспортном уровне. Это будет подробно обсуждаться в главе 6.

Сообщение ПЕРЕАДРЕСОВАТЬ посылается хосту, отправившему пакет, когда маршрутизатор замечает, что пакет адресован неверно.

Сообщения ЗАПРОС ОТКЛИКА и ОТКЛИК посылаются, чтобы определить, достижим ли и жив ли конкретный адресат. Получив сообщение ЗАПРОС ОТКЛИКА, хост должен отправить обратно сообщение ОТКЛИК. Сообщения ЗАПРОС ВРЕМЕННОГО ШТАМПА и ОТКЛИК С ВРЕМЕННЫМ ШТАМПОМ имеют то же назначение, но при этом в ответе проставляется время получения сообщения и время отправления ответа. Это сообщение используется для измерения производительности сети.

Кроме перечисленных сообщений, определены и другие. Их полный список хранится в Интернете по адресу www.iana.org/assignments/icmp-parameters.

ARP — протокол разрешения адресов

Хотя у каждой машины в Интернете есть один (или более) IP-адресов, они не могут использоваться для отправки пакетов, так как аппаратура уровня передачи данных не понимает интернет-адресов. В настоящее время большинство хостов соединены с локальными сетями с помощью интерфейсных карт, понимающих только адреса данной локальной сети. Например, каждая когда-либо выпущенная сетевая карта Ethernet имеет 48-разрядный Ethernet-адрес. Производители сетевых карт Ethernet запрашивают у центра блок адресов, что гарантирует уникальность Ethernet-адресов (это позволяет избежать конфликтов при наличии одинаковых сетевых карт в одной ЛВС). Сетевые карты отправляют и принимают кадры, основываясь на 48-разрядных Ethernet-адресах. О 32-разрядных IP-адресах им ничего не известно.

Таким образом, возникает вопрос: как устанавливается соответствие IP-адресов и адресов уровня передачи данных, таких как Ethernet-адреса? Чтобы понять, как это работает, рассмотрим показанный на рис. 5.53 пример, в котором изображен небольшой университет с несколькими сетями класса C (ныне называемыми сетями класса /24). На рисунке мы видим две сети Ethernet: одна на факультете кибернетики с IP-адресом 192.31.65.0, а другая — с IP-адресом 192.31.63.0 на электротехническом факультете. Они соединены кольцом FDDI с IP-адресом 192.31.60.0. У каждой машины сетей Ethernet есть уникальный Ethernet-адрес (на рисунке — от *E*1 до *E*6), а у каждой машины кольца FDDI есть FDDI-адрес (от *F*1 до *F*3).

Рассмотрим, как пользователь хоста 1 посылает пакет пользователю хоста 2. Допустим, отправителю известно имя получателя, например, mary@eagle.cs.uni.edu. Сначала надо найти IP-адрес для хоста 2, известного как eagle.cs.uni.edu. Этот поиск осуществляется службой имен доменов DNS (Domain Name System), которую мы рассмотрим в главе 7. На данный момент мы просто предположим, что служба DNS возвращает IP-адрес для хоста 2 (192.31.65.5).

Теперь программное обеспечение верхнего уровня хоста 1 создает пакет со значением 192.31.65.5 в поле *Адрес получателя* и передает его IP-программе для пересылки. Программное обеспечение протокола IP может посмотреть на адрес и увидеть, что адресат находится в его собственной сети, но ему нужно как-то определить Ethernet-адрес получателя. Одно из решений состоит в том, чтобы хранить в системе конфигурационный файл, в котором были бы перечислены соответствия всех локальных IP-адресов Ethernet-адресам. Такое решение, конечно, возможно, но в организациях с тысячами машин обновление этих файлов потребует много времени и подвержено ошибкам.

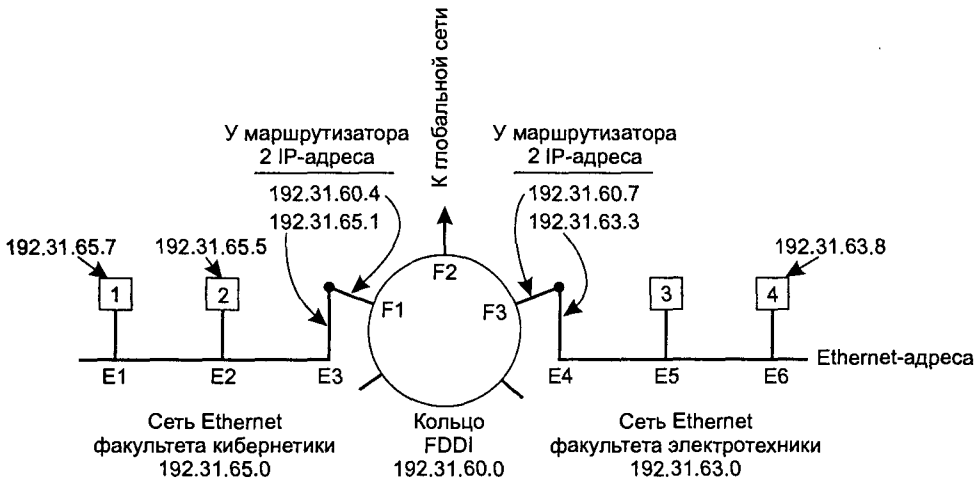


Рис. 5.53. Три объединенные сети класса /24: две сети Ethernet и кольцо FDDI

Более удачное решение заключается в рассылке хостом 1 по сети Ethernet широковещательного пакета с вопросом: «Кому принадлежит IP-адрес 192.31.65.5?» Этот пакет будет получен каждой машиной сети Ethernet 192.31.65.0, а хост 2 ответит на вопрос своим Ethernet-адресом *E2*. Таким образом, хост 1 узнает, что IP-адрес 192.31.65.5 принадлежит хосту с Ethernet-адресом *E2*. Протокол, который задает подобный вопрос и получает ответ на него, называется **ARP** (Address Resolution Protocol — протокол разрешения адресов) и описан в RFC 826. Он работает почти на каждой машине в Интернете.

Преимущество протокола ARP над файлами конфигурации заключается в его простоте. Системный администратор должен всего лишь назначить каждой машине IP-адрес и решить вопрос с маской подсети. Все остальное сделает протокол ARP.

Затем программное обеспечение протокола IP хоста 1 создает Ethernet-кадр для *E2*, помещает в его поле полезной нагрузки IP-пакет, адресованный 192.31.65.5, и посылает его по сети Ethernet. Сетевая карта Ethernet хоста 2 обнаруживает кадр, замечает, что он адресован ей, считывает его и вызывает прерывание. Ethernet-драйвер извлекает IP-пакет из поля полезной нагрузки и передает его IP-программе, которая, видя, что пакет адресован правильно, обрабатывает его.

Существуют различные методы повышения эффективности протокола ARP. Во-первых, машина, на которой работает протокол ARP, может запоминать результат преобразования адреса на случай, если ей придется снова связываться с той же машиной. В следующий раз она найдет нужный адрес в своем кэше, сэкономив, таким образом, на рассылке широковещательного пакета. Скорее всего, хосту 2 понадобится отослать ответ на пакет, что также потребует от него обращения к ARP для определения адреса отправителя. Этого обращения можно избежать, если отправитель включит в ARP-пакет свои IP- и Ethernet-адреса. Когда широковещательный ARP-пакет прибудет на хост 2, пара (192.31.65.7, *E*1) будет сохранена хостом 2 в ARP-кэше для будущего использования. Более того, эту пару адресов могут сохранить у себя все машины сети Ethernet.

Кроме того, каждая машина может рассылать свою пару адресов во время загрузки. Обычно эта широковещательная рассылка производится в виде ARP-пакета, запрашивающего свой собственный IP-адрес. Ответа на такой запрос быть не должно, но все машины могут запомнить эту пару адресов. Если ответ все же придет, это будет означать, что двум машинам назначен один и тот же IP-адрес. При этом вторая машина должна проинформировать системного администратора и прекратить загрузку.

Чтобы разрешить изменение соответствий адресов, например, при поломке и замене сетевой карты на новую (с новым Ethernet-адресом), записи в ARP-кэше должны устаревать за несколько минут.

Посмотрим снова на рис. 5.53. На этот раз хост 1 хочет послать пакет хосту 4 (192.31.63.8). Обращение к ARP не даст результата, так как хост 4 не увидит широковещательного пакета (маршрутизаторы не переправляют широковещательные пакеты Ethernet-уровня). Есть два решения данной проблемы. Во-первых, маршрутизатор факультета кибернетики можно настроить так, чтобы он отвечал на ARP-запросы к сети 192.31.63.0 (а также к другим местным сетям). В таком случае хост 1 добавит в свой ARP-кэш строку (192.31.63.8, E3) и будет счастлив посылать весь трафик для хоста 4 локальному маршрутизатору. Такой метод называется **ARP-прокси**. Второе решение состоит в том, чтобы хост 1 сразу выявлял нахождение адресата в удаленной сети и посылал весь внешний трафик по Ethernet-адресу, обрабатывающему все пакеты для удаленных адресатов, то есть по адресу маршрутизатора *E*3. В этом случае маршрутизатору факультета кибернетики не нужно знать, какую именно удаленную сеть он обслуживает.

В любом случае хост 1 помещает IP-пакет в поле полезной нагрузки Ethernet-кадра, адресованного маршрутизатору *E*3. Получив Ethernet-кадр, маршрутизатор факультета кибернетики извлекает из поля полезной нагрузки IP-пакет и ищет его IP-адрес в своих таблицах. Он обнаруживает, что пакеты, адресованные сети 192.31.63.0, должны пересылаться маршрутизатору 192.31.60.7. Если ему еще не известен FDDI-адрес маршрутизатора 192.31.60.7, то он посылает по кольцу широковещательный ARP-пакет и узнает, что нужный ему адрес *F*3. Затем он помещает IP-пакет в поле полезной нагрузки FDDI-кадра, адресованного маршрутизатору *F*3, и отправляет его по кольцу.

Когда кадр попадает на маршрутизатор факультета электротехники, FDDI-драйвер извлекает из поля полезной нагрузки IP-пакет и передает его IP-программе, которая понимает, что этот пакет следует переслать 192.31.63.8. Если та-

кого IP-адреса еще нет в ARP-кэше, маршрутизатор посылает широковещательный ARP-запрос по сети Ethernet факультета электротехники и узнает, что нужный ему адрес принадлежит хосту *E6*, поэтому он создает Ethernet-кадр, адресованный хосту *E6*, помещает IP-пакет в поле полезной нагрузки и передает его по сети Ethernet. Получив Ethernet-кадр, хост 4 извлекает из поля полезной нагрузки IP-пакет и передает его IP-программе для обработки.

Пересылка хостом 1 пакетов в удаленную сеть по глобальной сети работает аналогично, с той разницей, что на этот раз маршрутизатор факультета кибернетики узнает из своих таблиц, что пакет следует переслать маршрутизатору глобальной сети, чей FDDI-адрес равен *F2*.

Протоколы RARP, BOOTP и DHCP

Протокол ARP решает проблему определения по заданному IP-адресу Ethernet-адреса хоста. Иногда бывает необходимо решить обратную задачу, то есть по заданному Ethernet-адресу определить IP-адрес. В частности, эта проблема возникает при загрузке бездисковой рабочей станции. Обычно такая машина получает двоичный образ своей операционной системы от удаленного файлового сервера. Но как ей узнать его IP-адрес?

Первым для решения проблемы был разработан протокол **RARP** (Reverse Address Resolution Protocol — протокол обратного определения адреса), описанный в RFC 903. Этот протокол позволяет только что загрузившейся рабочей станции разослать всем свой Ethernet-адрес и сказать: «Мой 48-разрядный Ethernet-адрес — 14.04.05.18.01.25. Знает ли кто-нибудь мой IP-адрес?» RARP-сервер видит этот запрос, ищет Ethernet-адрес в своих файлах конфигурации и посылает обратно соответствующий IP-адрес.

Использование протокола RARP лучше внедрения IP-адреса в образ загружаемой памяти, так как это позволяет использовать данный образ памяти для разных машин. Если бы IP-адреса хранились бы где-то в глубине образа памяти, каждой машине понадобился бы свой отдельный образ.

Недостаток протокола RARP заключается в том, что в нем для обращения к RARP-серверу используется адрес, состоящий из одних единиц (ограниченное широковещание). Однако эти широковещательные запросы не переправляются маршрутизаторами в другие сети, поэтому в каждой сети требуется свой RARP-сервер. Для решения данной проблемы был разработан альтернативный загрузочный протокол **BOOTP**. В отличие от RARP, он использует UDP-сообщения, пересылаемые маршрутизаторами в другие сети. Он также снабжает бездисковые рабочие станции дополнительной информацией, включающей IP-адрес файлового сервера, содержащего образ памяти, IP-адрес маршрутизатора по умолчанию, а также маску подсети. Протокол BOOTP описан в документах RFC 951, 1048 и 1084.

Серьезной проблемой, связанной с применением BOOTP, является то, что таблицы соответствия адресов приходится настраивать вручную. Когда к ЛВС подключается новый хост, протокол BOOTP невозможно использовать до тех пор, пока администратор сети не присвоит ему IP-адрес и не пропишет вручную в конфигурационных таблицах пару (Ethernet-адрес, IP-адрес). Для устранения

влияния этого фактора протокол BOOTP был изменен и получил новое имя: **DHCP** (Dynamic Host Configuration Protocol — протокол динамической настройки хостов). DHCP позволяет настраивать таблицы соответствия адресов как вручную, так и автоматически. Этот протокол описан в RFC 2131 и 2132. В большинстве систем он уже практически заменил RARP и BOOTP.

Подобно RARP и BOOTP, DHCP основан на идее специализированного сервера, присваивающего IP-адреса хостам, которые их запрашивают. Такой сервер не обязательно должен быть подключен к той же ЛВС, что и запрашивающий хост. Поскольку сервер DHCP может быть недоступен с помощью широковещательной рассылки, в каждой ЛВС должен присутствовать **агент ретрансляции**, как показано на рис. 5.54.

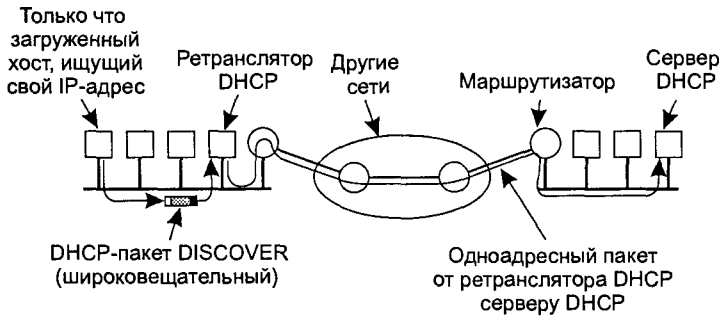


Рис. 5.54. Работа протокола DHCP

Для отыскания своего IP-адреса загружаемая машина широковещательным способом распространяет специальный пакет DISCOVER (Поиск). Агент ретрансляции DHCP перехватывает все широковещательные пакеты, относящиеся к протоколу DHCP. Обнаружив пакет DISCOVER, он превращает его из широковещательного в одноадресный и доставляет DHCP-серверу, который может находиться и в другой ЛВС. Агенту ретрансляции необходимо знать всего одну деталь: IP-адрес DHCP-сервера.

Встает вопрос: на какое время можно выдавать в автоматическом режиме IP-адреса из пула? Если хост покинет сеть и не освободит захваченный адрес, этот адрес будет навсегда утерян. С течением времени будет теряться все больше адресов. Для предотвращения этих неприятностей нужно выдавать IP-адреса не навсегда, а на определенное время. Такая технология называется **лизингом**. Перед окончанием срока действия лизинга хост должен послать на DHCP-сервер запрос о продлении срока пользования IP-адресом. Если такой запрос не был сделан или в просьбе было отказано, хост не имеет права продолжать использование выданного ранее адреса.

Протокол внутреннего шлюза OSPF

Итак, мы завершили изучение управляющих протоколов Интернета. Пришло время перейти к новой теме — маршрутизации в Интернете. Как уже упомина-

лось, Интернет состоит из большого количества автономных систем. Каждая автономная система управляется по-своему и может использовать внутри собственный алгоритм маршрутизации. Например, внутренние сети компаний X, Y и Z обычно рассматриваются как три автономные системы, если все они соединены с Интернетом. Они могут использовать различные внутренние алгоритмы маршрутизации. Тем не менее, наличие стандартов даже для внутренней маршрутизации упрощает реализацию на границах между автономными системами и позволяет повторно использовать программы. В данном разделе будет рассмотрена маршрутизация внутри автономной системы. В следующем разделе мы обсудим вопрос маршрутизации между автономными системами. Алгоритм маршрутизации внутри автономной системы называется **протоколом внутреннего шлюза**. Алгоритм маршрутизации между автономными системами называется **протоколом внешнего шлюза**.

Изначально в качестве протокола внутреннего шлюза в Интернете использовался протокол дистанционно-векторной маршрутизации RIP (Routing Information Protocol — протокол маршрутной информации), основанный на алгоритме Беллмана—Форда (Bellman—Ford) и унаследованный из ARPANET. Он хорошо работал в небольших системах, но по мере увеличения автономных систем стали проявляться его недостатки, такие как проблема счета до бесконечности и медленная сходимость, поэтому в мае 1979 года он был заменен протоколом состояния каналов. В 1988 году проблемная группа проектирования Интернета (IETF, Internet Engineering Task Force) начала работу над его преемником, которым в 1990 году стал алгоритм маршрутизации **OSPF** (Open Shortest Path First — открытый алгоритм предпочтительного выбора кратчайшего маршрута). В настоящее время он поддерживается многочисленными производителями маршрутизаторов и уже стал главным протоколом внутреннего шлюза. Далее будет дано краткое описание работы протокола OSPF. Более подробный рассказ о нем см. в RFC 2328.

Учитывая большой опыт работы с различными алгоритмами, группа разработчиков согласовывала свои действия с длинным списком требований, которые необходимо было удовлетворить.

Во-первых, этот алгоритм должен публиковаться в открытой литературе, откуда буква «O» (Open — открытый) в OSPF. Из этого следовало, что патентованный алгоритм, принадлежащий одной компании, не годится.

Во-вторых, новый протокол должен был уметь учитывать широкий спектр различных параметров, включая физическое расстояние, задержку и т. д.

В-третьих, этот алгоритм должен был быть динамическим, а также автоматически и быстро адаптирующимся к изменениям топологии.

В-четвертых (это требование впервые было предъявлено именно к OSPF), он должен был поддерживать выбор маршрутов, основываясь на типе сервиса. Новый протокол должен был уметь по-разному выбирать маршрут для трафика реального времени и для других видов трафика. IP-пакет уже давно содержит поле *Тип службы*, но ни один из имевшихся протоколов маршрутизации не использовал его.

В-пятых, новый протокол должен был уметь распределять нагрузку на линии. Это связано с предыдущим пунктом. Большинство протоколов посылало все па-

кеты по одному лучшему маршруту. Следующий по оптимальности маршрут не использовался совсем. Между тем во многих случаях распределение нагрузки по нескольким линиям дает лучшую производительность.

В-шестых, необходима поддержка иерархических систем. К 1988 году Интернет вырос настолько, что ни один маршрутизатор не мог вместить сведения о его полной топологии. Таким образом, требовалась разработка нового протокола.

В-седьмых, требовался необходимый минимум безопасности, защищающий маршрутизаторы от обманывающих их студентов-шутников, присылающих неверную информацию о маршруте. Наконец, требовалась поддержка для маршрутизаторов, соединенных с Интернетом по туннелю. Предыдущие протоколы справлялись с этим неудовлетворительно.

Протокол OSPF поддерживает три следующих типа соединений и сетей:

1. Двухточечные линии, соединяющие два маршрутизатора.
2. Сети множественного доступа с широковещанием (то есть большинство локальных сетей).
3. Сети множественного доступа без широковещания (то есть большинство глобальных сетей с коммутацией пакетов).

Сеть **множественного доступа** — это сеть, у которой может быть несколько маршрутизаторов, способных общаться друг с другом напрямую. Этим свойством обладают все локальные и глобальные сети. На рис. 5.55, *а* показана автономная система, содержащая все три типа сетей. Обратите внимание: хосты обычно не участвуют в отработке алгоритма OSPF.

В основе работы протокола OSPF лежит обобщенное представление о множестве сетей, маршрутизаторов и линий в виде направленного графа, в котором каждой дуге поставлена в соответствие ее цена (может выражаться в таких физических параметрах, как расстояние, задержка и т. д.). Затем, основываясь на весовых коэффициентах дуг, алгоритм вычисляет кратчайший путь. Последовательное соединение между двумя компьютерами представляется в виде пары дуг, по одной в каждом направлении. Их весовые коэффициенты могут быть различным. Сеть множественного доступа представляется в виде узла для самой сети, а также в виде узла для каждого маршрутизатора. Дуги, идущие от сетевого узла к узлам маршрутизатора, обладают нулевым весом и не включаются в граф.

Многие автономные системы в Интернете сами по себе довольно велики, и управлять ими непросто. Протокол OSPF позволяет делить их на пронумерованные **области**, то есть на сети или множества смежных сетей. Области не должны перекрываться, но не обязаны быть исчерпывающими, то есть некоторые маршрутизаторы могут не принадлежать ни одной области. Область является обобщением подсети. За пределами области ее топология и детали не видны.

У каждой автономной системы есть **магистральная область**, называемая областью 0. Все области соединены с магистралью, например, туннелями, так что по магистрали можно попасть из любой области автономной системы в ее любую другую область. Туннель представляется на графе в виде дуги и обладает определенной ценой. Каждый маршрутизатор, соединенный с двумя и более областями, является частью магистрали. Как и в случае других областей, топология магистрали за ее пределами не видна.

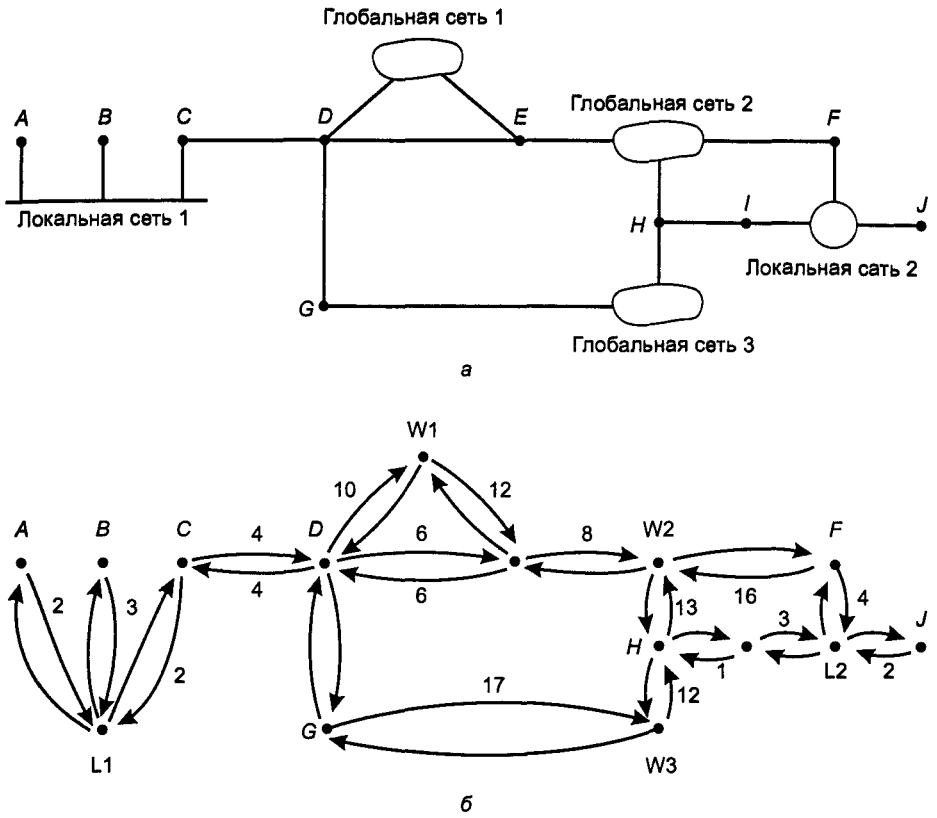


Рис. 5.55. Автономная система (а); представление *а* в виде графа (*б*)

У всех маршрутизаторов, принадлежащих к одной области, имеется одна и та же база данных состояния каналов и один алгоритм выбора кратчайшего пути. Работа маршрутизаторов заключается в расчете кратчайшего пути от себя до всех остальных маршрутизаторов этой области, включая маршрутизатор, соединенный с магистралью, который обязательно должен присутствовать в области, хотя бы один. Маршрутизатор, соединенный с двумя областями, должен иметь базы данных для каждой из них. Кратчайший путь для каждой области вычисляется отдельно.

При работе могут понадобиться три типа маршрутов: внутриобластные, межобластные и маршруты между автономными системами. Внутриобластные маршруты рассчитать легче всего, так как каждому маршрутизатору уже известен кратчайший путь до любого маршрутизатора своей области. Расчет межобластного маршрута состоит из трех этапов: от источника до магистрали, по магистрали до области назначения и от магистрали до адресата. Такой алгоритм приводит к конфигурации типа «звезда», в которой магистраль исполняет роль концентратора, а области являются лучами звезды. Пакеты направляются от отправителя к получателю в натуральном виде. Они не упаковываются в другие

пакеты и не туннелируются, кроме случаев, когда они направляются в области, с которыми магистраль соединена по туннелю. На рис. 5.56 показана часть Интернета с автономными системами и областями.

Протокол OSPF различает четыре класса маршрутизаторов:

1. Внутренние маршрутизаторы, расположенные целиком внутри области.
2. Маршрутизаторы границы области, соединяющие две и более областей.
3. Магистральные маршрутизаторы, находящиеся на магистрали.
4. Маршрутизаторы границы автономной системы, общающиеся с маршрутизаторами других автономных систем.

Эти классы могут перекрываться. Например, все пограничные маршрутизаторы автоматически являются магистральными. Кроме того, маршрутизатор, находящийся на магистрали, но не входящий ни в одну другую область, также является внутренним маршрутизатором. Примеры всех четырех классов показаны на рис. 5.56.

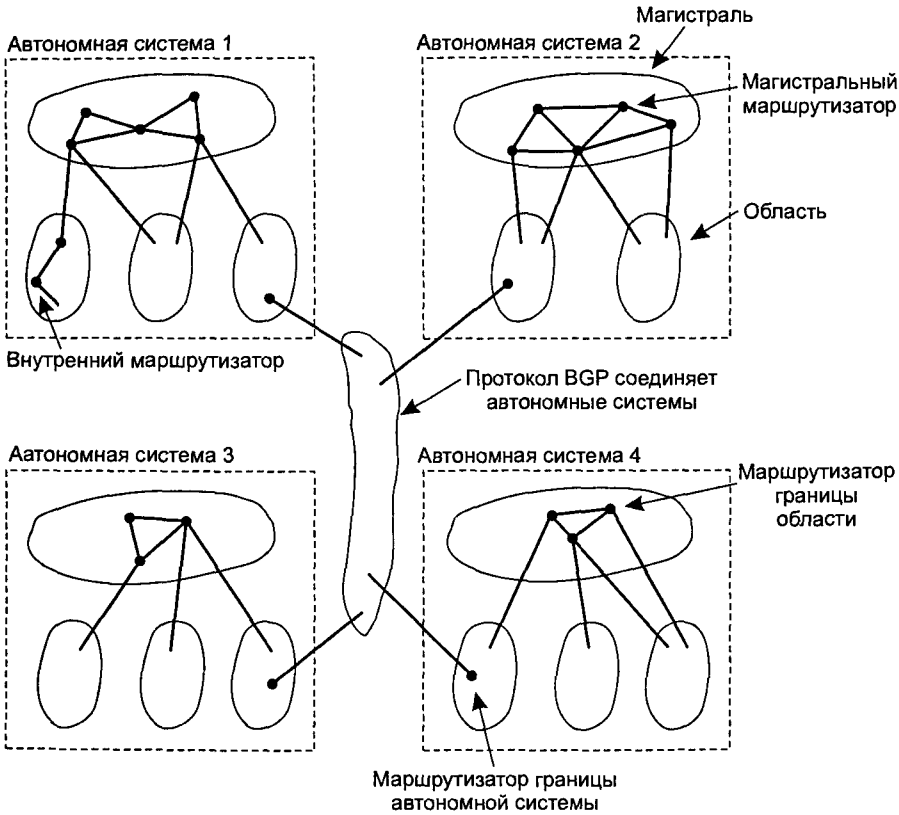


Рис. 5.56. Взаимосвязь между автономными системами, магистралями и областями в OSPF

При загрузке маршрутизатор рассылает сообщения HELLO по всем своим двухточечным линиям, производя многоадресную рассылку по локальным сетям для групп, состоящих из всех остальных маршрутизаторов. В глобальных сетях маршрутизатору требуется некая установочная информация, чтобы знать, с кем вступать в контакт. С помощью получаемых ответов каждый маршрутизатор знакомится со своими соседями.

Протокол OSPF работает при помощи обмена информацией между *смежными* маршрутизаторами, что не то же самое, что *соседние* маршрутизаторы. В частности, общение каждого маршрутизатора с каждым маршрутизатором локальной сети неэффективно. Поэтому один маршрутизатор выбирается **назначенным маршрутизатором**. Он считается **смежным** со всеми остальными маршрутизаторами и обменивается с ними информацией. Соседние маршрутизаторы, не являющиеся смежными, не обмениваются информацией друг с другом. На случай выхода из строя основного назначенного маршрутизатора всегда поддерживается в готовом состоянии запасной назначенный маршрутизатор.

При нормальной работе каждый маршрутизатор периодически рассылает методом заливки сообщение ОБНОВЛЕНИЕ СОСТОЯНИЯ КАНАЛОВ (LINK STATE UPDATE) всем своим смежным маршрутизаторам. Это сообщение содержит сведения о состоянии маршрутизатора и предоставляет информацию о цене, используемую в базах данных. В ответ на эти сообщения посылаются подтверждения, что повышает их надежность. Каждое сообщение получает последовательный номер, так что маршрутизатор может распознать, что новее: пришедшее сообщение или сообщение, хранимое им. Маршрутизаторы также рассылают эти сообщения, когда включается или выключается канал или изменяется его цена.

Сообщение ОПИСАНИЕ БАЗЫ ДАННЫХ (DATABASE DESCRIPTION) содержит порядковые номера всех записей о состоянии линий, которыми владеет отправитель. Сравнивая собственные значения со значениями отправителя, получатель может определить, у кого информация новее. Эти сообщения посылаются при восстановлении линии.

Каждый маршрутизатор может запросить информацию о состоянии линий у своего партнера с помощью сообщения ЗАПРОС О СОСТОЯНИИ КАНАЛА (LINK STATE REQUEST). В результате каждая пара смежных маршрутизаторов выясняет, чьи сведения являются более свежими, и, таким образом, по области распространяется наиболее новая информация. Все эти сообщения посылаются в виде IP-пакетов. Пять типов сообщений приведены в табл. 5.9.

Таблица 5.9. Пять типов сообщений протокола OSPF

Тип сообщения	Описание
Приветствие	Используется для знакомства с соседями
Обновление состояния каналов	Сообщает соседям информацию о каналах отправителя
Подтверждение состояния каналов	Подтверждает обновление состояния каналов
Описание базы данных	Сообщает о том, насколько свежей информацией располагает отправитель
Запрос состояния каналов	Запрашивает информацию у партнера

Подведем итоги. С помощью механизма заливки каждый маршрутизатор информирует все остальные маршрутизаторы своей области о своих соседях и цене каналов. Эта информация позволяет всем маршрутизаторам построить граф своей области и рассчитать кратчайшие пути. Маршрутизаторы магистральной области также занимаются этим. Кроме того, магистральные маршрутизаторы получают информацию от маршрутизаторов границ областей, с помощью которой они вычисляют оптимальные маршруты от каждого магистрального маршрутизатора до всех остальных маршрутизаторов. Эта информация рассылается обратно маршрутизаторам границ областей, которые распространяют ее в своих областях. С помощью этой информации маршрутизатор, собирающийся послать межобластной пакет, может выбрать оптимальный маршрутизатор, имеющий выход к магистрали.

Протокол внешнего шлюза BGP

В пределах одной автономной системы рекомендованным для применения в Интернете протоколом маршрутизации является OSPF (хотя он и не является единственным используемым протоколом). При выборе маршрута между различными автономными системами используется протокол **BGP** (Border Gateway Protocol — пограничный межсетевой протокол). Для выбора маршрута между различными автономными системами действительно требуется другой протокол, так как цели протоколов внутреннего и внешнего шлюзов различны. Задача протокола внутреннего шлюза ограничивается максимально эффективной передачей пакетов от отправителя к получателю. Политикой этот протокол не интересуется.

Протокол внешнего шлюза вынужден заниматься политикой (Metz, 2001). Например, корпоративной автономной системе может понадобиться возможность принимать и посылать пакеты на любой сайт Интернета. Однако прохождение через автономную систему пакета, отправитель и получатель которого находятся за пределами данного государства, может быть нежелательно, даже если кратчайший путь между отправителем и получателем пролегает через эту автономную систему («Это их заботы, а не наши»). С другой стороны, может оказаться желательным транзит трафика для других автономных систем, возможно, соседних, которые специально заплатили за эту услугу. Например, телефонные компании были бы рады оказывать подобные услуги, но только своим клиентам. Протоколы внешнего шлюза вообще и протокол BGP в частности разрабатывались для возможности учета различных стратегий при выборе маршрута между автономными системами.

Типичные стратегии выбора маршрутов учитывают политические, экономические факторы, а также соображения безопасности. К типичным примерам ограничений при выборе маршрутов относятся следующие:

1. Не пропускать трафик через определенные автономные системы.
2. Никогда не прокладывать через Ирак маршрут, начинающийся в Пентагоне.
3. Не использовать Соединенные Штаты при выборе маршрута из Британской Колумбии в штат Онтарио.

4. Прокладывать путь по Албании, только если нет альтернативных маршрутов.
5. Трафик, начинающийся или заканчивающийся в IBM®, не должен проходить через Microsoft®.

Стратегии настраиваются вручную на каждом BGP-маршрутизаторе (или представляют собой какой-нибудь скрипт). Они не являются частью протокола.

С точки зрения BGP-маршрутизатора, весь мир состоит из автономных систем и соединяющих их линий связи. Две автономные системы считаются соединенными, если есть общая линия между маршрутизаторами на их границах. Особая заинтересованность протокола BGP в транзитном трафике отразилась в разделении всех сетей на три категории. Первая категория представляет собой **тупиковые сети**, имеющие только одно соединение с BGP-графом. Они не могут использоваться для транзитного трафика, потому что на другой стороне ничего нет. Вторую категорию представляют **многосвязные сети**. Они могут применяться для транзитного трафика, если, только, конечно, согласятся на это. Наконец, имеются **транзитные сети** (например, магистрали), для которых транзитный трафик является желательным — возможно, с некоторыми ограничениями.

Пары BGP-маршрутизаторов общаются друг с другом, устанавливая TCP-соединения. Таким образом обеспечивается надежная связь и скрываются детали устройства сети, по которой проходит трафик.

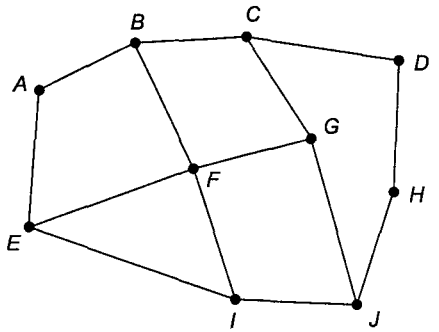
BGP по сути является протоколом маршрутизации по вектору расстояний, однако он значительно отличается от других подобных протоколов, например, протокола RIP (Routing Information Protocol — протокол маршрутной информации). Вместо того чтобы периодически сообщать всем своим соседям свои расчеты цены передачи до каждого возможного адресата, каждый BGP-маршрутизатор передает соседям точную информацию об используемых им маршрутах.

Для примера рассмотрим BGP-маршрутизаторы, показанные на рис. 5.57, *а*. В частности, рассмотрим таблицы маршрутизатора *F*. Предположим, что для доступа к маршрутизатору *D* он использует маршрут *FGCD*. Обмениваясь информацией о маршрутах, соседи сообщают друг другу полный используемый маршрут, как показано на рис. 5.57, *б* (для простоты показаны только пути к маршрутизатору *D*).

Получив все сведения о маршрутах от своих соседей, маршрутизатор *F* выбирает самый оптимальный из них. Он отбрасывает пути, используемые маршрутизаторами *I* и *E*, так как они проходят через маршрутизатор *F*. Таким образом, остается выбор между маршрутизаторами *B* и *G*. Каждый BGP-маршрутизатор содержит модуль, изучающий маршруты до каждого адресата и оценивающий расстояние до него. Каждый маршрут, нарушающий запрет политического характера, автоматически получает бесконечную оценку. Затем маршрутизатор принимает маршрут с кратчайшим расстоянием. Оценивающая функция не является частью протокола BGP, так что системный администратор может использовать любую оценивающую функцию.

Протокол BGP легко разрешает проблему счета до бесконечности, от которой страдают остальные алгоритмы дистанционно-векторной маршрутизации. Предположим, выходит из строя маршрутизатор *G* или линия *FG*. В этом случае маршрутизатор *F* получит информацию о маршрутах от своих трех оставшихся сосе-

дей. Этими маршрутами будут *BCD*, *IFGCD* и *EFGCD*. Второй и третий маршрут бесполезны для маршрутизатора *F*, так как они сами проходят через маршрутизатор *F*, поэтому в качестве нового пути к маршрутизатору *D* он выбирает маршрут *FBCD*. Другие алгоритмы дистанционно-векторной маршрутизации часто ошибаются, так как они не могут отличить соседей, обладающих независимыми маршрутами к адресату, от соседей, не обладающих такими маршрутами. Определение протокола BGP можно найти в RFC с 1771 по 1774.



а

Информация, которую маршрутизатор *F* получает от своих соседей о маршрутизаторе *D*

От *B*: «Я использую *BCD*»
От *G*: «Я использую *GCD*»
От *I*: «Я использую *IFGCD*»
От *E*: «Я использую *EFGCD*»

б

Рис. 5.57. Множество BGP-маршрутизаторов (*а*); информация, посланная маршрутизатору *F* (*б*)

Многоадресная рассылка в Интернете

Обычно IP-связь устанавливается между одним отправителем и одним получателем. Однако для некоторых приложений возможность послать сообщение одновременно большому количеству получателей является полезной. Такими приложениями могут быть, например, обновление реплицируемой распределенной базы данных, передача биржевых сводок брокерам и цифровые телеконференции (с участием нескольких собеседников).

Протокол IP поддерживает многоадресную рассылку при использовании адресов класса D. Каждый адрес класса D соответствует группе хостов. Для обозначения номера группы может быть использовано 28 бит, что делает возможным одновременное существование 250 миллионов групп. Когда процесс посылает пакет по адресу класса D, протокол прилагает максимальные усилия по его доставке всем членам группы, однако не дает гарантий доставки. Некоторые члены группы могут не получить пакета.

Поддерживаются два типа групповых адресов: постоянные и временные адреса. Постоянная группа не требует установки. У каждой постоянной группы есть постоянный адрес. Примерами постоянных групп являются:

- 224.0.0.1 — все системы локальной сети;
- 224.0.0.2 — все маршрутизаторы локальной сети;
- 224.0.0.5 — все OSPF-маршрутизаторы локальной сети;
- 224.0.0.6 — все назначенные OSPF-маршрутизаторы локальной сети.

Временные группы перед использованием следует создать. Процесс может попросить свой хост присоединиться к какой-либо группе. Когда последний процесс хоста покидает группу, группа более не присутствует на данном хосте. Каждый хост следит за тем, членами каких групп являются его процессы в текущий момент.

Многоадресная рассылка осуществляется специальными многоадресными маршрутизаторами, которые могут одновременно являться и стандартными маршрутизаторами. Примерно раз в минуту каждый многоадресный маршрутизатор совершает аппаратную (то есть на уровне передачи данных) многоадресную рассылку хостам на своей локальной сети (по адресу 224.0.0.1) с просьбой сообщить о группах, к которым в данный момент принадлежат их процессы. Каждый хост посылает обратно ответы для всех интересующих его адресов класса D.

Эти пакеты запросов и ответов используются протоколом **IGMP** (Internet Group Management Protocol — межсетевой протокол управления группами), являющимся грубым аналогом протокола ICMP (Internet Control Message Protocol — протокол контроля сообщений в сети Интернет). IGMP использует только два типа пакетов — запроса и ответа — фиксированного формата, содержащих управляющую информацию в первом слове поля полезной нагрузки и адрес класса D во втором. Этот формат описан в RFC 1112.

Многоадресная рассылка реализуется при помощи связующих деревьев. Каждый маршрутизатор многоадресной рассылки обменивается информацией со своими соседями с помощью модифицированного протокола дистанционно-векторной маршрутизации, что позволяет каждому построить для каждой группы связующее дерево, покрывающее всех членов группы. Для усечения дерева, чтобы исключить из него маршрутизаторы и сети, не являющиеся членами данной группы, применяются различные методы оптимизации. Чтобы миновать узлы сети, не являющиеся узлами связующего дерева, протокол использует туннелирование.

Мобильный IP

Многие пользователи Интернета обладают портативными компьютерами и заинтересованы в возможности оставаться в подключенном к сети состоянии, даже находясь в пути. К сожалению, система адресации IP такова, что реализовать это оказывается гораздо сложнее, чем кажется. В этом разделе мы познакомимся с этой проблемой и ее решением. Более подробное описание дано в (Perkins, 1998a).

Главным виновником проблемы является сама схема адресации. Каждый IP-адрес содержит номер сети и номер хоста. Например, рассмотрим машину с IP-адресом 160.80.40.20/16. Здесь 160.80 означает номер сети (8272 в десятичной системе счисления), 40.20 является номером хоста (10 260 в десятичной системе). Маршрутизаторы по всему миру содержат таблицы, в которых указывается, какую линию следует использовать, чтобы попасть в сеть 160.80. Когда приходит пакет с IP-адресом получателя вида 160.80.xxx.yyy, он отправляется по этой линии.

Если вдруг машина с этим адресом перевозится куда-то со своего места, адресованные ей пакеты будут продолжать направляться по ее домашнему адресу в ее локальную сеть (или ее маршрутизатору). До машины перестанет доходить электронная почта и т. п. Предоставление же машине нового адреса, соответствующего ее новому расположению, является нежелательным, так как об этом изменении придется информировать большое количество людей, программ и баз данных.

Другой подход состоит в использовании маршрутизаторами полного IP-адреса для определения маршрута, а не только полей класса и номера сети. Однако при такой стратегии каждый маршрутизатор должен будет содержать таблицы из миллионов записей, и стоимость поддержания Интернета в работоспособном состоянии составит астрономические суммы.

Когда потребность в мобильных хостах значительно возросла, проблемная группа проектирования Интернета (IETF, Internet Engineering Task Force) создала рабочую группу для поиска решения проблемы. Созданная рабочая группа быстро сформулировала набор целей, которых хотелось бы достичь, независимо от способа решения. Основными целями были признаны следующие:

1. Каждый мобильный хост должен иметь возможность использовать свой домашний IP-адрес где угодно.
2. Изменения программного обеспечения фиксированных хостов недопустимы.
3. Изменения программного обеспечения и таблиц маршрутизаторов недопустимы.
4. Большая часть пакетов, направляемых мобильным хостам, должны доставляться напрямую.
5. Не должно быть никаких дополнительных расходов, когда мобильный хост находится дома.

Рабочая группа выработала решение, описанное в разделе «Многоадресная рассылка». Суть его, напомним, заключалась в том, что везде, где требуется предоставить возможность перемещения в пространстве, следует создать внутреннего агента. Везде, где нужно принимать посетителей, следует создать внешнего агента. Когда мобильный хост прибывает на новое место, он связывается с местным внешним агентом и регистрируется. Затем внешний агент связывается с внутренним агентом пользователя и сообщает ему адрес для передачи сообщений прибывшему хосту. Обычно это IP-адрес внешнего агента.

Когда пакет прибывает в домашнюю локальную сеть пользователя, его получает маршрутизатор, соединенный с этой локальной сетью. При этом маршрутизатор пытается определить расположение хоста обычным способом, с помощью широковещательной рассылки ARP-пакета, спрашивая, например: «Каков Ethernet-адрес хоста 160.80.40.20?» Внутренний агент отвечает на этот запрос, выдавая свой собственный Ethernet-адрес. Маршрутизатор пересылает пакеты для 160.80.40.20 внутреннему агенту. Тот, в свою очередь, упаковывает их в поле данных IP-пакета, который туннелирует пакеты внешнему агенту. Внешний агент извлекает их и отсылает по адресу уровня передачи данных мобильного хоста. Внутренний агент также сообщает отправителю новый адрес мобильного хоста,

так что последующие пакеты могут быть туннелированы напрямую внешнему агенту. Это решение удовлетворяет всем перечисленным выше требованиям.

Следует, пожалуй, отметить одну небольшую деталь. Когда мобильный хост перемещается, у маршрутизатора, скорее всего, остается в памяти его Ethernet-адрес (который скоро станет недействительным). Чтобы заменить этот адрес адресом внутреннего агента, применяется хитрость, называемая **добровольным ARP-сообщением**. Это особое сообщение, предоставляемое маршрутизатору по инициативе хоста, которое заставляет маршрутизатор заменить в своей таблице запись о хосте, собирающемся покинуть свое место. Когда позднее мобильный хост возвращается, то же сообщение используется для повторного изменения памяти маршрутизатора.

Ничто не мешает мобильному хосту быть собственным внешним агентом, но такой подход будет работать только в том случае, когда мобильный хост (в качестве внешнего агента) логически связан с Интернетом на своем месте. Также он должен получить (временный) IP-адрес в текущей сети.

Решение, предложенное проблемной группой IETF, разрешает ряд других, еще не упомянутых проблем с мобильными хостами. Например, как обнаружить агента? Для этого агент периодически рассылает широковещательным способом свой адрес и тип услуг, которые он предоставляет (то есть пишет о том, кто он: внутренний агент, внешний агент или и то, и другое). Прибыв на новое место, хост может просто подождать рассылки этих широковещательных пакетов, называемых **рекламными объявлениями**. В качестве альтернативы он может сам разослать методом широковещания пакет с объявлением о своем прибытии и надеяться, что местный внешний агент на него отзовется.

Еще одна проблема состоит в том, что делать с невежливыми мобильными хостами, которые уходят не попрощавшись. Для решения этой проблемы регистрация хоста считается действительной только в течение ограниченного интервала времени. Если она периодически не обновляется, то считается устаревшей, после чего внешний агент может удалить запись о прибывшем хосте из своих таблиц.

Еще одним вопросом является безопасность. Когда внутренний агент получает просьбу пересылать все пакеты, приходящие на имя Натальи, на некий IP-адрес, он не должен подчиняться, пока он не убедится, что источником этого запроса является Наталья, а не кто-то пытающийся выдать себя за Наталью. Для этого применяются протоколы криптографической аутентификации, которые будут рассматриваться в главе 8.

Наконец, поговорим еще об одном вопросе, связанном с уровнями мобильности. Представьте себе самолет с установленной на борту сетью Ethernet, используемой навигационными и авиационными компьютерами. В этой сети есть стандартный маршрутизатор, общающийся с обычным стационарным Интернетом на земле по радиосвязи. В один прекрасный день кому-нибудь приходит в голову идея установить Ethernet-разъемы во всех подлокотниках кресел, так чтобы пассажиры с мобильными компьютерами могли подключаться к сети.

Таким образом, мы получаем два уровня мобильности: компьютеры самолета, неподвижные относительно сети Ethernet, и компьютеры пассажиров, являю-

щиеся мобильными относительно нее. Кроме того, бортовой маршрутизатор является мобильным относительно наземных маршрутизаторов. Мобильность относительно системы, которая сама является мобильной, может поддерживаться при помощи рекурсивного туннелирования.

Протокол IPv6

Хотя системы адресации CIDR и NAT и могут помочь нынешнему IP продержаться еще несколько лет, всем понятно, что дни IPv4 сочтены. Помимо перечисленных ранее технических проблем, имеется еще одна, угрожающе вырастающая на горизонте. До недавних пор Интернетом пользовались в основном университеты, высокотехнологичные предприятия и правительственные организации (особенно Министерство обороны). С лавинообразным ростом интереса к Интернету, начавшимся в середине 90-х годов, в третьем тысячелетии, скорее всего, им будет пользоваться гораздо большее количество пользователей с принципиально разными требованиями. Во-первых, пользователи портативных компьютеров могут пользоваться Интернетом для доступа к домашним базам данных. Во-вторых, при неминуемом сближении компьютерной промышленности, средств связи и индустрии развлечений, возможно, очень скоро каждый телевизор планеты станет узлом Интернета, что в результате приведет к появлению миллиардов машин, используемых для видео по заказу. В таких обстоятельствах становится очевидно, что протокол IP должен эволюционировать и стать более гибким.

Предвидя появление этих проблем, проблемная группа проектирования Интернета IETF начала в 1990 году работу над новой версией протокола IP, в которой никогда не должна возникнуть проблема нехватки адресов, а также будут решены многие другие проблемы. Кроме того, новая версия протокола должна была быть более гибкой и эффективной. Были сформулированы следующие основные цели:

1. Поддержка миллиардов хостов даже при неэффективном использовании адресного пространства.
2. Уменьшение размера таблиц маршрутизации.
3. Упрощение протокола для ускорения обработки пакетов маршрутизаторами.
4. Более надежное обеспечение безопасности (аутентификации и конфиденциальности), чем в нынешнем варианте IP.
5. Необходимость обращать больше внимания на тип сервиса, в частности, при передаче данных реального времени.
6. Упрощение работы многоадресных рассылок с помощью указания областей рассылки.
7. Возможность изменения положения хоста без необходимости изменять его адрес.
8. Возможность дальнейшего развития протокола в будущем.
9. Возможность сосуществования старого и нового протоколов в течение нескольких лет.

Чтобы найти протокол, удовлетворяющий всем этим требованиям, IETF издал в RFC 1550 приглашение к дискуссиям и предложениям. Был получен двадцать один ответ. Далеко не все варианты содержали предложения, полностью удовлетворяющие этим требованиям. В декабре 1992 года были рассмотрены семь серьезных предложений. Их содержание варьировалось от небольших изменений в протоколе IP до полного отказа от него и замены совершенно другим протоколом.

Одно из предложений состояло в использовании вместо IP протокола CLNP, который с его 160-разрядным адресом обеспечивал бы достаточное адресное пространство на веки вечные. Кроме того, это решение объединило бы два основных сетевых протокола. Однако все же сочли, что при подобном выборе придется признать, что кое-что в мире OSI было сделано правильно, что было бы политически некорректным в интернет-кругах. Протокол CLNP, на самом деле, очень мало отличается от протокола IP. Окончательный выбор был сделан в пользу протокола, отличающегося от IP значительно сильнее, нежели CLNP. Еще одним аргументом против CLNP была его слабая поддержка типа сервиса, требовавшегося для эффективной передачи мультимедиа.

Три лучших предложения были опубликованы в журнале *IEEE Network Magazine* (Deering, 1993; Francis, 1993; Katz и Ford, 1993). После долгих обсуждений, переработок и борьбы за первое место была выбрана модифицированная комбинированная версия Диринга (Deering) и Фрэнсиса (Francis), называемая в настоящий момент протоколом **SIPP** (Simple Internet Protocol Plus — простой интернет-протокол Плюс). Новому протоколу было дано обозначение **IPv6** (протокол IPv5 уже использовался в качестве экспериментального протокола потоков реального времени).

Протокол IPv6 прекрасно справляется с поставленными задачами. Он обладает достоинствами протокола IP и лишен некоторых его недостатков (либо они проявляются в меньшей степени), к тому же наделен некоторыми новыми особенностями. В общем случае протокол IPv6 несовместим с протоколом IPv4, но зато совместим со всеми остальными протоколами Интернета, включая TCP, UDP, ICMP, IGMP, OSPF, BGP и DNS, для чего иногда требуются небольшие изменения (в основном чтобы работать с более длинными адресами). Основные особенности протокола IPv6 обсуждаются далее. Дополнительные сведения о нем можно найти в RFC с 2460 по 2466.

Прежде всего, у протокола IPv6 поля адресов длиннее, чем у IPv4. Они имеют длину 16 байт, что решает основную проблему, поставленную при разработке протокола, — обеспечить практически неограниченный запас интернет-адресов. Мы еще кратко упомянем об адресах чуть позднее.

Второе заметное улучшение протокола IPv6 по сравнению с IPv4 состоит в более простом заголовке пакета. Он состоит всего из 7 полей (вместо 13 у протокола IPv4). Таким образом, маршрутизаторы могут быстрее обрабатывать пакеты, что повышает производительность. Краткое описание заголовков будет приведено далее.

Третье усовершенствование заключается в улучшенной поддержке необязательных параметров. Подобное изменение действительно было существенным,

так как в новом заголовке требуемые прежде поля стали необязательными. Кроме того, изменился способ представления необязательных параметров, что упростило для маршрутизаторов пропуск не относящихся к ним параметров и ускорило обработку пакетов.

В-четвертых, протокол IPv6 демонстрирует большой шаг вперед в области безопасности. У проблемной группы проектирования Интернета IETF была полная папка вырезок из газет с сообщениями о том, как 12-летние мальчишки с помощью своего персонального компьютера по Интернету вломились в банк или военную базу. Было ясно, что надо как-то улучшить систему безопасности. Аутентификация и конфиденциальность являются ключевыми чертами нового IP-протокола.

Наконец, в новом протоколе было уделено больше внимания типу предоставляемых услуг. Для этой цели в заголовке пакета IPv4 было отведено 8-разрядное поле (на практике не используемое), но при ожидаемом росте мультимедийного трафика в будущем требовалось значительно больше разрядов.

Основной заголовок IPv6

Заголовок IPv6 показан на рис. 5.58. Поле *Версия* содержит число 6 для IPv6 (и 4 для IPv4). На период перехода с IPv4 на IPv6, который, вероятно, займет около десяти лет, маршрутизаторы по значению этого поля смогут различать пакеты нового и старого стандарта. Подобная проверка потребует нескольких тактов процессора, что может оказаться нежелательным в некоторых ситуациях, поэтому многие реализации, вероятно, попытаются избежать этих накладных расходов и будут отличать пакеты IPv4 от пакетов IPv6 с помощью некоторого поля в заголовке уровня передачи данных. При этом пакеты будут передаваться напрямую нужному сетевому уровню. Однако знакомство уровня передачи данных с типами пакетов сетевого уровня полностью нарушает принцип разделения протоколов на уровни, при котором каждый уровень не должен знать назначения битов из пакетов более высокого уровня. Дискуссия между лагерями, руководствующимися принципами «Делай правильно» и «Делай быстро», несомненно, будет долгой и энергичной.

Поле *Класс трафика* используется для того, чтобы различать пакеты с разными требованиями к доставке в реальном времени. Такое поле присутствовало в стандарте IP с самого начала, однако оно реально обрабатывалось маршрутизаторами лишь в единичных случаях. Сейчас проводятся эксперименты, направленные на то, чтобы определить, как лучше всего использовать это поле для передачи данных в реальном времени.

Поле *Метка потока* также пока является экспериментальным, но будет применяться для установки между отправителем и получателем псевдосоединения с определенными свойствами и требованиями. Например, поток пакетов между двумя процессами на разных хостах может обладать строгими требованиями к задержкам, что потребует резервирования пропускной способности. Поток устанавливается заранее и получает идентификатор. Когда прибывает пакет с отличным от нуля содержимым поля *Метка потока*, все маршрутизаторы смотрят в свои таблицы, чтобы определить, какого рода особая обработка ему требуется.

Таким образом, новый протокол пытается соединить достоинства подсетей различных типов: гибкость дейтаграмм и гарантии виртуальных каналов.

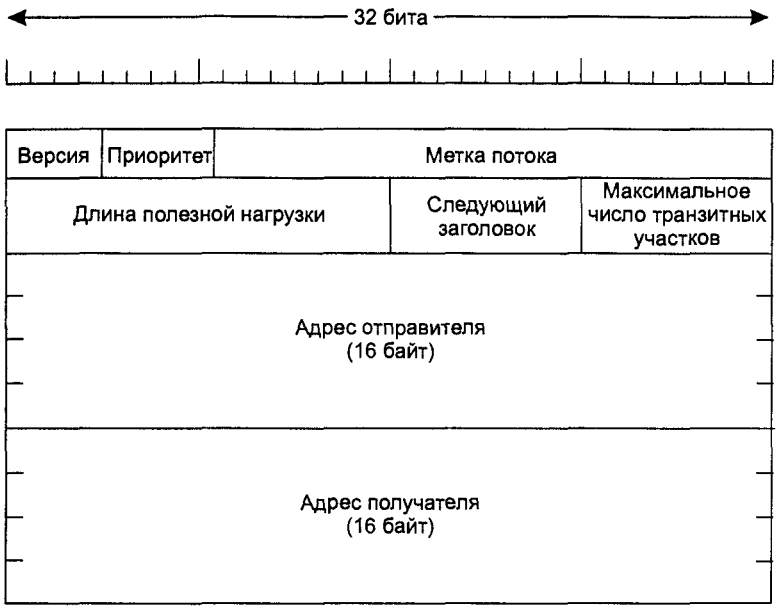


Рис. 5.58. Фиксированный заголовок IPv6 (обязательные поля)

Каждый поток описывается адресом источника, адресом назначения и номером потока, так что для каждой пары IP-адресов можно создать много активных потоков. Два потока с одинаковыми номерами, но различными адресами отправителя или получателя считаются разными потоками и различаются маршрутизаторами по адресам. Ожидается, что номера каналов будут выбираться случайным образом, а не назначаться подряд начиная с 1, что облегчит маршрутизаторам их распознавание.

Поле *Длина полезной нагрузки* сообщает, сколько байт следует за 40-байтовым заголовком, показанным на рис. 5.58. В заголовке IPv4 аналогичное поле называлось *Полная длина* и определяло весь размер пакета. В новом протоколе 40 байт заголовка учитываются отдельно.

Поле *Следующий заголовок* раскрывает секрет возможности использования упрощенного заголовка. Дело в том, что после обычного 40-байтового заголовка могут идти дополнительные (необязательные) расширенные заголовки. Это поле сообщает, какой из шести дополнительных заголовков (на текущий момент) следует за основным. В последнем IP-заголовке поле *Следующий заголовок* сообщает, какой обрабатывающей программе протокола транспортного уровня (то есть TCP или UDP) передать пакет.

Поле *Максимальное число транзитных участков* не дает пакетам вечно блуждать по сети. Оно имеет практически то же назначение, что и поле *Время жизни* в заголовке протокола IPv4. Это поле уменьшается на единицу на каждом тран-

зитном участке. Теоретически, в протоколе IPv4 это поле должно было содержать секунды времени жизни пакета, однако ни один маршрутизатор не использовал его подобным образом, поэтому имя поля было приведено в соответствие способу его применения.

Следом идут поля *Адрес отправителя* и *Адрес получателя*. В исходном предложении Диринга (протоколе SIPP) использовались 8-байтовые адреса, но при рассмотрении проекта было решено, что 8-байтовых адресов хватит лишь на несколько десятилетий, в то время как 16-байтовых адресов должно хватить навечно. Другие возражали, что 16 байтов для адресов слишком много, тогда как третьи настаивали на 20-байтных адресах для совместимости с дейтаграммным протоколом OSI. Еще одна фракция ратовала за адреса переменной длины. После продолжительных споров было решено, что наилучшим компромиссным решением являются 16-байтовые адреса фиксированной длины.

Для написания 16-байтовых адресов была выработана новая нотация. Адреса в IPv6 записываются в виде восьми групп по четыре шестнадцатеричных цифры, разделенных двоеточиями, например:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Поскольку многие адреса будут содержать большое количество нулей, были разрешены три метода сокращенной записи адресов. Во-первых, могут быть опущены ведущие нули в каждой группе, например, 0123 можно записывать как 123. Во-вторых, одна или более групп, полностью состоящих из нулей, могут заменяться парой двоеточий. Таким образом, приведенный выше адрес принимает вид

8000::123:4567:89AB:CDEF

Наконец, адреса IPv4 могут записываться как пара двоеточий, после которой пишется адрес в старом десятичном формате, например:

::192.31.20.46

Возможно, нет необходимости говорить об этом столь подробно, но количество всех возможных 16-байтовых адресов очень велико — 2^{128}, что приблизительно равно $3 \cdot 10^{38}$. Если покрыть компьютерами всю планету, включая сушу и океаны, то протокол IPv6 позволит иметь около $7 \cdot 10^{23}$ IP-адресов на квадратный метр. Кто изучал химию, может заметить, что это число больше числа Авогадро. Хотя в планы разработчиков не входило предоставление собственного IP-адреса каждой молекуле на поверхности Земли, они оказались не так уж далеко от обеспечения такой услуги.

На практике не все адресное пространство используется эффективно, как, например, не используются абсолютно все комбинации телефонных номеров. Например, телефонные номера Манхэттена (код 212) почти полностью заняты, тогда как в штате Вайоминг (код 307) они почти не используются. В RFC 3194 Дьюранд (Durand) и Хуйтема (Huitema) приводят свои вычисления. Утверждается, что если ориентироваться на использование телефонных номеров, то даже при самом пессимистическом сценарии все равно получается более 1000 IP-адресов на квадратный метр поверхности Земли (включая как сушу, так и море).

При любом более вероятном сценарии обеспечиваются триллионы адресов на квадратный метр. Таким образом, маловероятно, что в обозримом будущем обнаружится нехватка адресов. Также следует отметить, что на сегодня только для 28 % адресного пространства придуманы применения. Остальные 72 % зарезервированы на будущее.

Полезно сравнить заголовок IPv4 (рис. 5.47) с заголовком IPv6 (рис. 5.58), чтобы увидеть, что осталось от старого стандарта. Поле *IHL* исчезло, так как заголовок IPv6 имеет фиксированную длину. Поле *Протокол* также было убрано, поскольку поле *Следующий заголовок* сообщает, что следует за последним IP-заголовком (то есть UDP- или TCP-сегмент).

Были удалены все поля, относящиеся к фрагментации, так как в протоколе IPv6 используется другой подход к фрагментации. Во-первых, все хосты, поддерживающие протокол IPv6, должны динамически определять нужный размер дейтаграммы. Это правило делает фрагментацию маловероятной. Во-вторых, минимальный размер пакета был увеличен с 576 до 1280, чтобы можно было передавать 1024 байт данных, плюс множество заголовков. Кроме того, когда хост посылает слишком большой IPv6-пакет вместо того, чтобы его фрагментировать, то маршрутизатор, не способный переслать пакет дальше, посылает обратно сообщение об ошибке. Получив это сообщение, хост должен прекратить всю передачу этому адресату. Гораздо правильнее будет научить все хосты посылать пакеты требуемого размера, нежели учить маршрутизаторы фрагментировать их на лету.

Наконец, поле *Контрольная сумма* было удалено, так как ее подсчет значительно снижает производительность. Поскольку в настоящее время все шире используются надежные линии связи, а на уровне передачи данных и на транспортном уровне подсчитываются свои контрольные суммы, наличие еще одной контрольной суммы не стоило бы тех затрат производительности, которых требовал бы ее подсчет. В результате перечисленных удалений получился простой, быстрый и в то же время гибкий протокол сетевого уровня с огромным адресным пространством.

Дополнительные заголовки

В опущенных полях заголовка иногда возникает необходимость, поэтому в протоколе IPv6 была представлена новая концепция (необязательного) **дополнительного заголовка**. На сегодня определены шесть типов дополнительных заголовков, которые перечислены в табл. 5.9. Все они являются необязательными, но в случае использования более чем одного дополнительного заголовка они должны располагаться сразу за фиксированным заголовком, желательно в указанном порядке.

Таблица 5.9. Дополнительные заголовки IPv6

Дополнительный заголовок	Описание
Параметры маршрутизации	Разнообразная информация для маршрутизаторов
Параметры получателя	Дополнительная информация для получателя
Маршрутизация	Частичный список транзитных маршрутизаторов на пути пакета

продолжение

Таблица 5.9 (*продолжение*)

Дополнительный заголовок	Описание
Фрагментация	Управление фрагментами дейтаграмм
Аутентификация	Проверка подлинности отправителя
Шифрованные данные	Информация о зашифрованном содержимом

У некоторых заголовков формат фиксированный, другие содержат переменное количество полей переменной длины. Для них каждый пункт кодируется в виде тройки (Тип, Длина, Значение). *Тип* представляет собой однобайтовое поле, содержащее код параметра. Первые два бита этого поля сообщают, что делать с пакетом, маршрутизаторам, не знающим, как обрабатывать данный параметр. Возможны четыре следующих варианта: пропустить параметр, игнорировать пакет, игнорировать пакет и отослать обратно ICMP-пакет, а также то же самое, что и предыдущий вариант, но не отсылать обратно ICMP-пакет в случае многоадресной рассылки (чтобы один неверный многоадресный пакет не породил миллионы ICMP-донесений).

Поле *Длина* также имеет размер 1 байт. Оно сообщает, насколько велико значение (от 0 до 255 байт). Поле *Значение* содержит необходимую информацию, размером до 255 байт.

Заголовок параметров маршрутизации содержит информацию, которую должны исследовать маршрутизаторы на протяжении всего пути следования пакета. Пока что был определен один вариант использования этого параметра: поддержка дейтаграмм, превышающих 64 Кбайт. Формат заголовка показан на рис. 5.59.

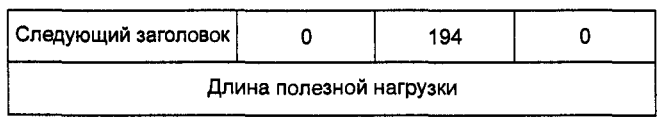


Рис. 5.59. Дополнительный заголовок для больших дейтаграмм

Как и все дополнительные заголовки, он начинается с байта, означающего тип следующего заголовка. Следующий байт содержит длину дополнительного заголовка в байтах, не считая первых 8 байт, являющихся обязательными. С этого начинаются все расширения.

Следующие два байта указывают, что данный параметр содержит размер дейтаграммы (код 194) в виде 4-байтового числа. Размеры меньше 65 536 не допускаются, так как могут привести к тому, что первый же маршрутизатор проигнорирует данный пакет и отошлет обратно ICMP-сообщение об ошибке. Дейтаграммы, использующие подобные расширения заголовка, называются **джамбограммами** (от слова «jumbo», означающего нечто большое и неуклюжее). Использование джамбограмм важно для суперкомпьютерных приложений, которым необходимо эффективно передавать по Интернету гигабайты данных.

Маршрутный заголовок содержит информацию об одном или нескольких маршрутизаторах, которые следует посетить по пути к получателю. Это очень

сильно напоминает свободную маршрутизацию стандарта IPv4 тем, что указанные в списке маршрутизаторы должны быть пройдены строго по порядку, тогда как не указанные проходятся между ними. Формат маршрутного заголовка показан на рис. 5.60.

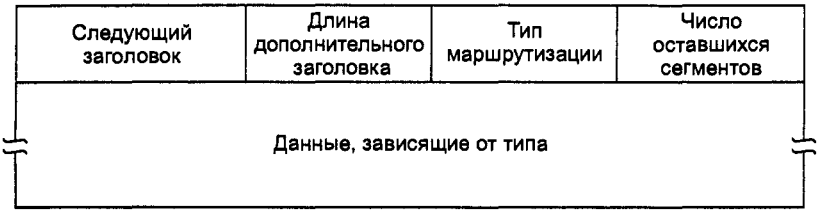


Рис. 5.60. Дополнительный заголовок для маршрутизации

Первые четыре байта дополнительного маршрутного заголовка содержат четыре однобайтовых целых числа. Поля *Следующий заголовок* и *Длина дополнительного заголовка* были описаны ранее. В поле *Тип маршрутизации* описывается формат оставшейся части заголовка. Если он равен 0, это означает, что далее следует зарезервированное 32-разрядное слово, а за ним — некоторое число адресов IPv6. В будущем, возможно, будут по мере необходимости изобретаться какие-то новые поля. Наконец, в поле *Число оставшихся сегментов* указывается, сколько адресов из списка еще осталось посетить. Его значение уменьшается при прохождении каждого адреса. Когда оно достигает нуля, пакет оставляется на произвол судьбы — никаких указаний относительно его дальнейшего маршрута не дается. Обычно в этот момент пакет уже находится достаточно близко к месту назначения, и оптимальный маршрут очевиден.

Заголовок фрагментации определяет фрагментацию способом, схожим с протоколом IPv4. Заголовок содержит идентификатор дейтаграммы, номер фрагмента и бит, информирующий о том, является ли этот фрагмент последним. В отличие от IPv4, в протоколе IPv6 фрагментировать пакет может только хост-источник. Маршрутизаторы фрагментировать пересылаемые пакеты не могут. Это порывающее с философией прошлого изменение в протоколе упрощает и ускоряет работу маршрутизаторов. Как уже было сказано, маршрутизатор отвергает слишком большие пакеты, посылая в ответ ICMP-пакет, указывающий хосту-источнику на необходимость заново передать пакет, выполнив его фрагментацию на меньшие части.

Заголовок аутентификации предоставляет механизм подтверждения подлинности отправителя пакета. Шифрование данных, содержащихся в поле полезной нагрузки, обеспечивает конфиденциальность: прочесть содержимое пакета сможет только тот, для кого предназначен пакет. Для выполнения этой задачи в заголовках используются криптографические методы.

Полемика

При той открытости, с которой происходил процесс разработки протокола IPv6, и при убежденности многочисленных разработчиков в собственной правоте не-

удивительно, что многие решения принимались в условиях весьма жарких дискуссий. О некоторых из них будет рассказано далее. Все кровавые подробности описаны в соответствующих RFC.

О спорах по поводу длины поля адреса уже упоминалось. В результате было принято компромиссное решение: 16-байтовые адреса фиксированной длины.

Другое сражение разгорелось из-за размера поля *Максимальное количество транзитных участков*. Один из противостоящих друг другу лагерей считал, что ограничение количества транзитных участков числом 255 (это явно следует из использования 8-битного поля) является большой ошибкой. В самом деле, маршруты из 32 транзитных участков уже стали обычными, а через 10 лет могут стать обычными более длинные маршруты. Сторонники этого лагеря заявляли, что использование полей адресов огромного размера было решением дальновидным, а применение крохотных счетчиков транзитных участков — недальновидным. Самый страшный грех, который, по их мнению, могут совершить специалисты по вычислительной технике, — это выделить для чего-нибудь недостаточное количество разрядов.

В ответ им было заявлено, что подобные аргументы можно привести для увеличения любого поля, что приведет к разбуханию заголовка. Кроме того, назначение поля *Максимальное количество транзитных участков* состоит в том, чтобы не допустить слишком долгого странствования пакетов, и 65 535 транзитных участков — это очень много. К тому же по мере роста Интернета будет создаваться все большее количество междугородных линий, что позволит передавать пакеты из любой страны в любую страну максимум за шесть транзитных пересылок. Если от получателя или отправителя до соответствующего международного шлюза окажется более 125 транзитных участков, то, видимо, что-то не в порядке с магистралями этого государства. В итоге эту битву выиграли сторонники 8-битового счетчика.

Еще одним предметом спора оказался максимальный размер пакета. Обладатели суперкомпьютеров настаивали на размере пакетов, превышающем 64 Кбайт. Когда суперкомпьютер начинает передачу, он занимается серьезным делом и не хочет, чтобы его прерывали через каждые 64 Кбайта. Аргумент против больших пакетов заключается в том, что если пакет размером в 1 Мбайт будет передаваться по линии T1 со скоростью 1,5 Мбит/с, то он займет линию на целых 5 секунд, что вызовет слишком большую задержку, заметную для интерактивных пользователей. В данном вопросе удалось достичь компромисса: нормальные пакеты ограничены размером 64 Кбайт, но с помощью дополнительного заголовка можно пересылать дейтаграммы огромного размера.

Еще одним спорным вопросом оказалось удаление контрольной суммы IPv4. Кое-кто сравнивал этот ход с удалением тормозов из автомобиля. При этом автомобиль становится легче и может двигаться быстрее, но если случится что-нибудь неожиданное, то могут быть проблемы.

Аргумент против контрольных сумм состоял в том, что каждое приложение, действительно заботящееся о целостности своих данных, все равно считает контрольную сумму на транспортном уровне, поэтому наличие еще одной на сетевом уровне является излишним (кроме того, контрольная сумма подсчитывается

еще и на уровне передачи данных). Более того, эксперименты показали, что вычисление контрольной суммы составляло основные затраты протокола IPv4. Это сражение было выиграно лагерем противников контрольной суммы, поэтому в протоколе IPv6, как мы знаем, контрольной суммы нет.

Вокруг мобильных хостов также разгорелся спор. Если мобильный компьютер окажется на другом конце Земли, сможет ли он продолжать использовать прежний IPv6-адрес или должен будет использовать схему с внутренним и внешним агентами? Мобильные хосты также вносят асимметрию в систему маршрутизации. Вполне реальна ситуация, когда маленький мобильный компьютер хорошо слышит мощный сигнал большого стационарного маршрутизатора, но стационарный маршрутизатор не слышит слабого сигнала, передаваемого мобильным хостом. Поэтому появилось много желающих создать в протоколе IPv6 явную поддержку мобильных хостов. Эти попытки потерпели поражение, поскольку ни по одному конкретному предложению не удалось достичь консенсуса.

Вероятно, самые жаркие баталии разгорелись вокруг вопроса безопасности. Все были согласны, что это необходимо. Спорным было то, где и как следует реализовывать безопасность. Во-первых, где. Аргументом за размещение системы безопасности на сетевом уровне было то, что при этом она становится стандартной службой, которой могут пользоваться все приложения безо всякого предварительного планирования. Контраргумент заключался в том, что по-настоящему защищенным приложениям подходит лишь сквозное шифрование, когда шифрование осуществляется самим источником, а дешифровка — непосредственным получателем. Во всех остальных случаях пользователь оказывается в зависимости от, возможно, содержащей ошибки реализации сетевого уровня, над которой у него нет контроля. В ответ на этот аргумент можно сказать, что приложение может просто отказаться от использования встроенных в IP функций защиты и выполнять всю эту работу самостоятельно. Возражение на этот контраргумент состоит в том, что пользователи, не доверяющие сетям, не хотят платить за не используемую ими функцию, реализация которой утяжеляет и замедляет работу протокола, даже если сама функция отключена.

Другой аспект вопроса расположения системы безопасности касается того факта, что во многих (но не во всех) странах приняты строгие экспортные законы, касающиеся криптографии. В некоторых странах, особенно во Франции и Ираке, строго запрещено использование криптографии даже внутри страны, чтобы у населения не могло быть секретов от полиции. В результате любая реализация протокола IP, использующая достаточно мощную криптографическую систему, не может быть экспортирована за пределы Соединенных Штатов (и многих других стран). Таким образом, приходится поддерживать два набора программного обеспечения — один для внутреннего использования, а другой для экспорта, против чего решительно выступает большинство производителей компьютеров.

Единственный вопрос, по которому не было споров, состоял в том, что никто не ожидает, что IPv4-Интернет будет выключен в воскресенье, а в понедельник утром будет включен уже IPv6-Интернет. Вместо этого вначале появятся «островки» IPv6, которые будут общаться по туннелям. По мере роста острова IPv6

будут объединяться в более крупные острова. Наконец все острова объединятся, и Интернет окажется полностью трансформированным. Учитывая огромные средства, вложенные в работающие сегодня IPv4-маршрутизаторы, процесс трансформации, вероятно, займет около десяти лет. Поэтому были приложены гигантские усилия, чтобы гарантировать максимальную безболезненность этого перехода. Дополнительную информацию, касающуюся IPv6, можно найти в (Loshin, 1999).

Резюме

Сетевой уровень предоставляет услуги транспортному уровню. В его основе могут лежать либо виртуальные каналы, либо дейтаграммы. В обоих случаях его основная работа состоит в выборе маршрута пакетов от источника до адресата. В подсетях с виртуальными каналами решение о выборе маршрута осуществляется при установке виртуального канала. В дейтаграммных подсетях оно принимается для каждого пакета.

В компьютерных сетях применяется большое количество алгоритмов маршрутизации. К статическим алгоритмам относятся определение кратчайшего пути и заливка. Динамические алгоритмы включают в себя дистанционно-векторную маршрутизацию и маршрутизацию с учетом состояния каналов. В большинстве имеющихся сетей применяется один из этих алгоритмов. К другим важным методам маршрутизации относятся иерархическая маршрутизация, маршрутизация для мобильных хостов, широковещательная маршрутизация, многоадресная маршрутизация и маршрутизация в равноранговых сетях.

Подсети подвержены перегрузкам, увеличивающим задержки и снижающим пропускную способность. Разработчики сетей пытаются предотвратить перегрузку разнообразными способами, включающими формирование трафика, спецификации потока и резервирование пропускной способности. Если перегрузку предотвратить не удалось, с ней нужно бороться. Для этого могут применяться посылаемые отправителю сдерживающие пакеты, перераспределение нагрузки и другие методы.

Существующие на сегодня сети имеют много различий, поэтому при объединении могут возникнуть проблемы. Разработчики пытаются избавиться от них, создавая соответствующие решения. Среди используемых методов надо выделить стратегии повторной передачи, кэширования, управления потоком и т. д. Если перегрузка сети все же возникает, с ней приходится бороться. Для этого можно посылать сдерживающие пакеты, сбрасывать нагрузку или применять другие методы.

Следующим шагом после разрешения проблем с перегрузкой является попытка достижения гарантированного качества обслуживания. Методы, используемые для этого, включают в себя буферизацию на стороне клиента, формирование трафика, резервирование ресурсов, контроль доступа. Подходы, разработанные для обеспечения хорошего качества обслуживания, включают в себя интегри-

рованное обслуживание (включая RSVP), дифференцированное обслуживание и MPLS.

Разные сети могут отличаться друг от друга весьма значительно, поэтому при попытке их объединения могут возникать определенные сложности. Иногда проблемы могут быть решены при помощи туннелирования пакета сквозь сильно отличающуюся сеть, но если отправитель и получатель находятся в сетях разных типов, этот подход не может быть применен. Если в сетях различаются ограничения на максимальный размер пакета, может быть применена фрагментация.

В Интернете существует большое разнообразие протоколов, относящихся к сетевому уровню. К ним относятся протокол транспортировки данных IP, управляющие протоколы ICMP, ARP и RARP, а также протоколы маршрутизации OSFP и BGP. Интернету перестает хватать IP-адресов, поэтому была разработана новая версия протокола IP, IPv6.

Вопросы

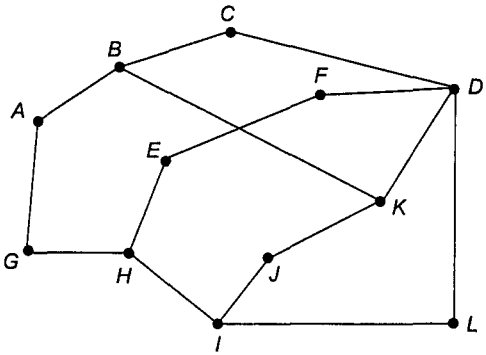
1. Приведите два примера приложений, для которых ориентированная на соединение служба является приемлемой. Затем приведите два примера приложений, для которых наилучшей будет служба без использования соединений.
2. Могут ли возникнуть обстоятельства, при которых служба виртуальных каналов доставлять пакеты без сохранения их порядка? Объясните.
3. Дейтаграммные подсети выбирают маршрут для каждого отдельного пакета независимо от других. В подсетях виртуальных каналов каждый пакет следует по заранее определенному маршруту. Означает ли это, что подсетям виртуальных каналов не требуется способность выбирать маршрут для отдельного пакета от произвольного источника произвольному адресату. Поясните свой ответ.
4. Приведите три примера параметров протокола, о значениях которых можно договориться при установке соединения.
5. Рассмотрите следующую проблему, касающуюся реализации службы виртуальных каналов. Если подсеть основана на виртуальных каналах, то каждый пакет данных должен иметь 3-байтовый заголовок, а каждому маршрутизатору требуется 8 байт для хранения информации о виртуальном канале. Если в подсети используется дейтаграммная служба, тогда необходимы 15-байтовые заголовки пакетов, но не требуется памяти для таблиц маршрутизаторов. Передача данных стоит 1 цент за 10^6 байт на транзитный участок. Очень быстрая память маршрутизаторов может быть приобретена по цене 1 цент за байт со сроком полной амортизации более двух лет (имеется в виду работа по 40 часов в неделю). Статистически усредненный сеанс продолжается 1000 с, за время которого передаются 200 пакетов. В среднем пакет пересылается через четыре транзитных участка. Какой способ реализации будет дешевле и насколько?

6. Предполагая, что все маршрутизаторы и хосты работают нормально и что их программное обеспечение не содержит ошибок, есть ли вероятность, хотя бы небольшая, что пакет будет доставлен неверному адресату?
7. Рассмотрите сеть на рис. 5.6, игнорируя весовые коэффициенты линий. Допустим, в качестве алгоритма маршрутизации применяется метод заливки. Пакет, посланный *A* для *D*, имеет ограничение на максимальное число переходов, равное 3. Перечислите все маршрутизаторы, через которые он пройдет. Сколько переходов относительно всей пропускной способности займет эта передача?
8. Укажите простой эвристический метод нахождения двух путей от данного источника к данному адресату, гарантирующий сохранение связи при обрыве любой линии (если такие два пути существуют). Маршрутизаторы считаются достаточно надежными, поэтому рассматривать возможность выхода их из строя не нужно.
9. Рассмотрите подсеть на рис. 5.11, *а*. Используется алгоритм дистанционно-векторной маршрутизации. На маршрутизатор *C* только что поступили следующие векторы: от *B* (5, 0, 8, 12, 6, 2); от *D* (16, 12, 6, 0, 9, 10); от *E* (7, 6, 3, 9, 0, 4). Измеренные задержки до *B*, *D* и *E* составляют 6, 3 и 5 соответственно. Какой будет новая таблица маршрутизатора *C*? Укажите используемые выходные линии и ожидаемое время задержки.
10. В сети, состоящей из 50 маршрутизаторов, значения времени задержки записываются как 8-битовые номера, а маршрутизаторы обмениваются векторами задержек дважды в секунду. Какая пропускная способность на каждой (дуплексной) линии съедается работой распределенного алгоритма маршрутизации? Предполагается, что каждый маршрутизатор соединен тремя линиями с другими маршрутизаторами.
11. На рис. 5.12 логическое ИЛИ двух наборов ACF-битов равно 111 для каждого ряда. Является ли это просто случайностью или же это сохраняется во всех подсетях при любых условиях?
12. Какие размеры регионов и кластеров следует выбрать для минимизации таблиц маршрутизации при трехуровневой иерархической маршрутизации, если количество маршрутизаторов равно 4800. Рекомендуется начать с гипотезы о том, что решение в виде k кластеров по k регионов из k маршрутизаторов близко к оптимальному. Это означает, что число k примерно равно корню кубическому из 4800 (около 16). Методом проб и ошибок подберите все три параметра так, чтобы они были близки к 16.
13. В тексте утверждалось, что когда мобильного хоста нет в его домашней сети, пакеты, посланные на адрес его домашней ЛВС, перехватываются внутренним агентом этой ЛВС. Как этот перехват осуществляет внутренний агент в IP-сети на основе локальной сети 802.3?
14. Сколько широковещательных пакетов формируется маршрутизатором *B* на рис. 5.5 с помощью:

    1) пересылки в обратном направлении;

    2) входного дерева?

15. Рассмотрите рис. 5.14, *а*. Допустим, добавляется одна новая линия между *F* и *G*, но входное дерево, показанное на рис. 5.14, *б*, остается без изменений. Какие изменения нужно внести в рис. 5.14, *в*?
16. Рассчитайте многоадресное связующее дерево для маршрутизатора *C* в подсети, показанной ниже, для группы, состоящей из маршрутизаторов *A*, *B*, *C*, *D*, *E*, *F*, *I* и *K*.



17. Рассмотрите рис. 5.18. При показанном поиске, начинающемся на узле *A*, будут ли когда-нибудь узлы *H* и *I* заниматься широковещанием?
18. Допустим, узел *B* на рис. 5.18 только что перезагрузился и не имеет никакой информации о маршрутизации в своих таблицах. Внезапно у него появляется необходимость в маршруте к узлу *H*. Он рассылает широковещательным способом наборы *TTL* на 1, 2, 3 и т. д. Сколько раундов потребуется на поиск пути?
19. В простейшем варианте алгоритма хорд при поиске в равноранговых сетях таблицы указателей не используются. Вместо этого производится линейный поиск по кругу в обоих направлениях. Может ли при этом узел предсказать, в каком направлении следует искать? Ответ аргументируйте.
20. Рассмотрите круг, используемый в алгоритме хорд и показанный на рис. 5.22. Допустим, узел 10 внезапно подключается к сети. Повлияет ли это на таблицу указателей узла 1, и если да, то как?
21. В качестве возможного механизма борьбы с перегрузкой в подсети, использующей виртуальные каналы, маршрутизатор может воздержаться от подтверждения полученного пакета в следующих случаях: 1) он знает, что его последняя передача по виртуальному каналу была получена успешно; 2) у него есть свободный буфер. Для простоты предположим, что маршрутизаторы используют протокол с ожиданием и что у каждого виртуального канала есть один буфер, выделенный ему для каждого направления трафика. Передача пакета (данных или подтверждения) занимает *T* секунд. Путь пакета проходит через *n* маршрутизаторов. С какой скоростью пакеты доставляются адресату? Предполагается, что ошибки очень редки, а связь между хостом и маршрутизатором почти не отнимает времени.

22. Дейтаграммная подсеть позволяет маршрутизаторам при необходимости выбрасывать пакеты. Вероятность того, что маршрутизатор отвергнет пакет, равна p. Рассмотрите маршрут, проходящий от хоста к хосту через два маршрутизатора. Если любой из маршрутизаторов отвергнет пакет, у хоста-отправителя в конце концов истечет интервал ожидания и он попытается переслать пакет еще раз. Если обе линии (хост—маршрутизатор и маршрутизатор—маршрутизатор) считать за транзитные участки, то чему равно среднее число:

    1) транзитных участков, преодолеваемых пакетом за одну передачу;

    2) передач для одного пакета;

    3) транзитных участков, необходимых для получения пакета?

23. В чем состоит основная разница между методом предупредительного бита и методом RED?

24. Почему алгоритм «дырявое ведро» должен позволять передачу лишь одного пакета за интервал времени, независимо от размеров пакета?

25. В некоторой системе используется вариант алгоритма «дырявое ведро» с подсчетом байтов. Правило гласит, что за один интервал времени может быть послан один 1024-байтовый пакет или два 512-байтовых пакета и т. д. В чем заключается не упомянутое в этом тексте серьезное ограничение такой системы?

26. Сеть ATM использует для формирования трафика схему маркерного ведра. Новый маркер помещается в ведро каждые 5 мкс. Чему равна максимальная скорость передачи данных в сети (не считая битов заголовка)?

27. Компьютер, подключенный к сети, скорость передачи в которой равна 6 Мбит/с, регулируется маркерным ведром. Маркерное ведро наполняется со скоростью 1 Мбит/с. Его начальная емкость составляет 8 Мбит. Как долго сможет передавать компьютер на полной скорости в 6 Мбит/с?

28. Представьте, что максимальный размер пакета в спецификации потока равен 1000 байт, скорость маркерного ведра равна 10 млн байт/с, объем маркерного ведра составляет 1 млн байт, а максимальная скорость передачи равна 50 млн байт/с. Как долго может продолжаться передача с максимальной скоростью?

29. Сеть на рис. 5.32 использует RSVP в деревьях групповой рассылки для хостов 1 и 2. Допустим, хост 3 запрашивает канал с пропускной способностью 2 Мбайт/с для потока от хоста 1 и еще один канал с пропускной способностью 1 Мбайт/с для потока от хоста 2. Одновременно хост 4 запрашивает 2-мегабайтный канал для потока от хоста 1, а хост 5 запрашивает 1-мегабайтный канал для потока от хоста 2. Какую суммарную пропускную способность необходимо зарезервировать для удовлетворения перечисленных запросов на маршрутизаторах A, B, C, E, H, J, K и L?

30. Центральный процессор маршрутизатора может обрабатывать 2 млн пакетов в секунду. Сколько времени уйдет на формирование очередей и обслужива-

ние пакетов процессорами, если путь от источника до приемника содержит 10 маршрутизаторов?

31. Допустим, пользователь получает дифференцированный сервис со срочной пересылкой. Есть ли гарантия того, что срочные пакеты будут испытывать меньшую задержку, чем обычные? Ответ поясните.
32. Нужна ли фрагментация в интерсетях с объединенными виртуальными каналами или она необходима только в дейтаграммных системах?
33. Туннелирование сквозь подсеть сцепленных виртуальных каналов осуществляется следующим образом: многопротокольный маршрутизатор на одном конце устанавливает виртуальный канал с другим концом и посылает по нему пакеты. Можно ли применить туннелирование в дейтаграммных подсетях? Если да, как?
34. Допустим, хост *A* соединен с маршрутизатором *R*1. Тот, в свою очередь, соединен с другим маршрутизатором, *R*2, а *R*2 — с хостом *B*. Сообщение TCP, содержащее 900 байт данных и 20 байт TCP-заголовка, передается IP-программе, установленной на хосте *A*, для доставки его хосту *B*. Каковы будут значения полей *Общая длина*, *Идентификатор*, *DF*, *MF* и *Сдвиг фрагмента* IP-заголовка каждого пакета, передающегося по трем линиям. Предполагается, что на линии *A*—*R*1 максимальный размер кадра равен 1024 байта, включая 14-байтный заголовок кадра, на линии *R*1-*R*2 максимальный размер кадра составляет 512 байт, включая 8-байтный заголовок кадра, и на линии *R*2—*B* максимальный размер кадра составляет 512 байт, включая 12-байтный заголовок кадра
35. Маршрутизатор освобождает IP-пакеты, общая длина которых (включая данные и заголовок) равна 1024 байт. Предполагая, что пакеты живут в течение 10 с, сосчитайте максимальную скорость линии, с которой может работать маршрутизатор без опасности зацикливания в пространстве идентификационных номеров IP-дейтаграммы.
36. IP-дейтаграмма, использующая параметр *Строгая маршрутизация от источника*, должна быть фрагментирована. Копируется ли этот параметр в каждый фрагмент, или достаточно поместить его в первый фрагмент? Поясните свой ответ.
37. Допустим, вместо 16 бит в адресе класса B для обозначения номера сети отводилось бы 20 бит. Сколько было бы тогда сетей класса B?
38. Преобразуйте IP-адрес, шестнадцатеричное представление которого равно C22F 1582, в десятичный формат, разделенный точками.
39. Маска подсети сети Интернета равна 255.255.240.0. Чему равно максимальное число хостов в ней?
40. Существует множество адресов, начинающихся с IP-адреса 198.16.0.0. Допустим, организации *A*, *B*, *C* и *D* запрашивают, соответственно, 4000, 2000, 4000 и 8000 адресов. Для каждой из них укажите первый и последний выданные адреса, а также маску вида *w.x.y.z/s*.

41. Маршрутизатор только что получил информацию о следующих IP-адресах: 57.6.96.0/21, 57.6.112.0/21 и 57.6.120.0/21. Если для них используется одна и та же исходящая линия, можно ли их агрегировать? Если та, то во что? Если нет, то почему?
42. Набор IP-адресов с 29.18.0.0 по 19.18.128.255 агрегирован в 29.18.0.0/17. Тем не менее, остался пробел из 1024 не присвоенных адресов, с 29.18.60.0 по 29.18.63.255, которые внезапно оказались присвоены хосту, использующему другую исходящую линию. Необходимо ли теперь разделить агрегированный адрес на составляющие, добавить в таблицу новый блок, а потом посмотреть, можно ли что-нибудь агрегировать? Если нет, тогда что можно сделать?
43. Маршрутизатор содержит следующие записи (CIDR) в своей таблице маршрутизации:

Адрес/маска	Следующий переход
135.46.56.0/22	Интерфейс 0
135.46.60.0/22	Интерфейс 1
192.53.40.0/23	Маршрутизатор 1
По умолчанию	Маршрутизатор 2

44. Куда направит маршрутизатор пакеты со следующими IP-адресами?
    1) 135.46.63.10;
    2) 135.46.57.14;
    3) 135.46.52.2;
    4) 192.53.40.7;
    5) 192.53.56.7.
45. Многие компании придерживаются стратегии установки двух и более маршрутизаторов, соединяющих компанию с провайдером, что гарантирует некоторый запас прочности на случай, если один из маршрутизаторов выйдет из строя. Применима ли такая политика при использовании NAT? Ответ поясните.
46. Вы рассказали товарищу про протокол ARP. Когда вы закончили объяснения, он сказал: «Ясно. ARP предоставляет услуги сетевому уровню, таким образом, он является частью уровня передачи данных». Что вы ему ответите?
47. Протоколы ARP и RARP оба устанавливают соответствия адресов из разных адресных пространств. В этом смысле они похожи. Однако способы их реализации в корне различны. В чем их основное отличие?
48. Опишите способ сборки пакета из фрагментов в пункте назначения.
49. В большинстве алгоритмов сборки IP-дейтаграмм из фрагментов используется таймер, чтобы из-за потерянного фрагмента буфер, в котором производится повторная сборка, не оказался занят остальными фрагментами дейтаграммы. Предположим, дейтаграмма разбивается на четыре фрагмента. Первые три фрагмента прибывают к получателю, а четвертый задерживается в пути.

У получателя истекает период ожидания, и три фрагмента, хранившиеся в его памяти, удаляются. Немного позднее наконец приползает последний фрагмент. Как следует с ним поступить?

50. Как в IP, так и в ATM контрольная сумма покрывает только заголовок, но не данные. Почему, как вы полагаете, была выбрана подобная схема?

51. Особа, живущая в Бостоне, едет в Миннеаполис и берет с собой свой персональный компьютер. К ее удивлению, локальная сеть в Миннеаполисе является беспроводной локальной сетью IP, поэтому ей нет необходимости подключать свой компьютер. Нужно ли, тем не менее, проходить процедуру с внутренним и внешним агентом, чтобы электронная почта и другой трафик прибывали правильно?

52. Протокол IPv6 использует 16-байтовые адреса. На какое время хватит этих адресов, если каждую пикосекунду назначать блок в 1 млн адресов?

53. Поле *Протокол*, используемое в заголовке IPv4, отсутствует в фиксированном заголовке IPv6. Почему?

54. Должен ли протокол ARP быть изменен при переходе на шестую версию протокола IP? Если да, то являются ли эти изменения концептуальными или техническими?

55. Напишите программу, моделирующую маршрутизацию методом заливки. Каждый пакет должен содержать счетчик, уменьшаемый на каждом маршрутизаторе. Когда счетчик уменьшается до нуля, пакет удаляется. Время дискретно, и каждая линия обрабатывает за один интервал времени один пакет. Создайте три версии этой программы: с заливкой по всем линиям, с заливкой по всем линиям, кроме входной линии, и с заливкой только k лучших линий (выбираемых статически). Сравните заливку с детерминированной маршрутизацией ($k = 1$) с точки зрения задержки и использования пропускной способности.

56. Напишите программу, моделирующую компьютерную сеть с дискретным временем. Первый пакет в очереди каждого маршрутизатора преодолевает по одному транзитному участку за интервал времени. Число буферов каждого маршрутизатора ограничено. Прибывший пакет, для которого нет свободного места, игнорируется и повторно не передается. Вместо этого используется сквозной протокол с тайм-аутами и пакетами подтверждения, который, в конце концов, вызывает повторную передачу пакета маршрутизатором-источником. Постройте график производительности сети как функции интервала сквозного времени ожидания при разных значениях частоты ошибок.

57. Напишите функцию, осуществляющую пересылку в IP-маршрутизаторе. У процедуры должен быть один параметр — IP-адрес. Имеется доступ к глобальной таблице, представляющей собой массив из троек значений. Каждая тройка содержит следующие целочисленные значения: IP-адрес, маску подсети и исходящую линию. Функция ищет IP-адрес в таблице, используя CIDR, и возвращает номер исходящей линии.

Используя программы *traceroute* (UNIX) или *tracert* (Windows), исследуйте маршрут от вашего компьютера до различных университетов мира. Составьте список трансокеанских линий. Вот некоторые адреса:

www.berkeley.edu (Калифорния);

www.mit.edu (Массачусетс);

www.vu.nl (Амстердам);

www.ucl.ac.uk (Лондон);

www.usyd.edu.au (Сидней);

www.u-tokyo.ac.jp (Токио);

www.uct.ac.za (Кейптаун).

Глава 6
Транспортный уровень

- Транспортная служба
- Элементы транспортных протоколов
- Простой транспортный протокол
- Транспортные протоколы Интернета: UDP
- Транспортные протоколы Интернета: TCP
- Вопросы производительности
- Резюме
- Вопросы

Транспортный уровень — это не просто очередной уровень. Это сердцевина всей иерархии протоколов. Его задача состоит в предоставлении надежной и экономичной передачи данных от машины-источника машине-адресату вне зависимости от физических характеристик использующейся сети или сетей. Без транспортного уровня вся концепция многоуровневых протоколов потеряет смысл. В данной главе мы подробно рассмотрим транспортный уровень, включая его сервисы, устройство, протоколы и производительность.

Транспортная служба

В следующих разделах мы познакомимся с транспортной службой. Мы рассмотрим виды сервисов, предоставляемых прикладному уровню. Чтобы наш разговор не был слишком абстрактным, мы разберем два набора примитивов транспортного уровня. Сначала рассмотрим простой (но не применяемый на практике) набор, просто чтобы показать основные идеи, а затем — реально применяемый в Интернете интерфейс.

Услуги, предоставляемые верхним уровнем

Конечная цель транспортного уровня заключается в предоставлении эффективных, надежных и экономичных услуг (сервисов) своим пользователям, которыми обычно являются процессы прикладного уровня. Для достижения этой цели транспортный уровень пользуется услугами, предоставляемыми сетевым уровнем. Аппаратура и/или программа, выполняющая работу транспортного уровня, называется **транспортной сущностью** или **транспортным объектом**. Транспортный объект может располагаться в ядре операционной системы, в отдельном пользовательском процессе, в библиотечном модуле, загруженном сетевым приложением, или в сетевой интерфейсной плате. Логическая взаимосвязь сетевого, транспортного и прикладного уровней проиллюстрирована на рис. 6.1.

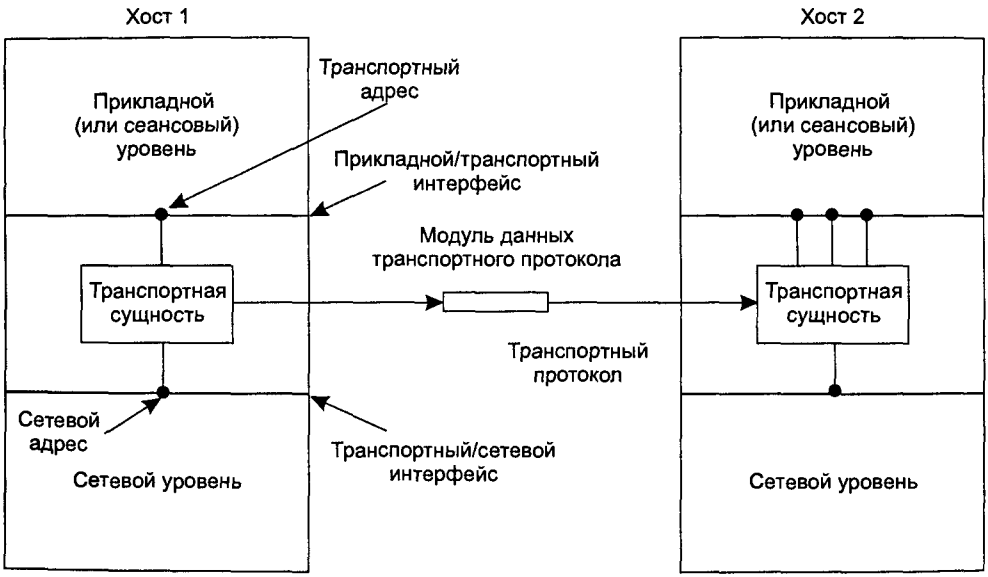


Рис. 6.1. Сетевой, транспортный и прикладной уровни

Сервисы транспортного уровня, как и сервисы сетевого уровня, могут быть ориентированными на соединение или не требующими соединений. Ориентированный на соединение транспортный сервис во многом похож на ориентированный на соединение сетевой сервис. В обоих случаях соединение проходит три этапа: установка, передача данных и разъединение. Адресация и управление потоком на разных уровнях также схожи. Более того, похожи друг на друга и не требующие соединений сервисы разных уровней.

Возникает закономерный вопрос: если сервис транспортного уровня так схож с сервисом сетевого уровня, то зачем нужны два различных уровня? Почему недостаточно одного уровня? Для ответа на этот важный вопрос следует вернуться к рис. 1.7. На рисунке мы видим, что транспортный код запускается целиком на пользовательских машинах, а сетевой уровень запускается в основном на машру-

тизаторах, которые управляются оператором связи (по крайней мере, в глобальных сетях). Что произойдет, если сетевой уровень будет предоставлять ориентированный на соединение сервис, но этот сервис будет ненадежным? Например, если он часто будет терять пакеты? Можно себе представить, что случится, если маршрутизаторы будут время от времени выходить из строя.

В этом случае пользователи столкнутся с большими проблемами. У них нет контроля над сетевым уровнем, поэтому они не смогут улучшить качество обслуживания, используя хорошие маршрутизаторы или совершенствуя обработку ошибок уровня передачи данных. Единственная возможность заключается в использовании еще одного уровня, расположенного над сетевым, для улучшения качества обслуживания. Если транспортный объект узнает, что его сетевое соединение было внезапно прервано, но не получит каких-либо сведений о том, что случилось с передаваемыми в этот момент данными, он может установить новое соединение с удаленной транспортной сущностью. С помощью нового сетевого соединения он может послать запрос к равноранговому объекту и узнать, какие данные дошли до адресата, а какие нет, после чего продолжить передачу данных.

По сути, благодаря наличию транспортного уровня транспортный сервис может быть более надежным, чем лежащий ниже сетевой сервис. Транспортным уровнем могут быть обнаружены потерянные пакеты и искаженные данные, после чего потери могут быть компенсированы. Более того, примитивы транспортной службы могут быть разработаны таким образом, что они будут независимы от примитивов сетевой службы, которые могут значительно варьироваться от сети к сети (например, сервис локальной сети без соединений может значительно отличаться от сервиса ориентированной на соединение глобальной сети). Если спрятать службы сетевого уровня за набором примитивов транспортной службы, то для изменения сетевой службы потребуется просто заменить один набор библиотечных процедур другими, делающими то же самое, но с помощью других сервисов более низкого уровня.

Благодаря наличию транспортного уровня прикладные программы могут использовать стандартный набор примитивов и сохранять работоспособность в самых различных сетях. Им не придется учитывать имеющееся разнообразие интерфейсов подсетей и беспокоиться о ненадежной передаче данных. Если бы все реальные сети работали идеально и у всех сетей был один набор служебных примитивов, то транспортный уровень, вероятно, был бы не нужен. Однако в реальном мире он выполняет ключевую роль изолирования верхних уровней от деталей технологии, устройства и несовершенства подсети.

Именно по этой причине часто проводится разграничение между уровнями с первого по четвертый и уровнями выше четвертого. Нижние четыре уровня можно рассматривать как **поставщика транспортных услуг**, а верхние уровни — как **пользователя транспортных услуг**. Разделение на поставщика и пользователя оказывает серьезное влияние на устройство уровней и помещает транспортный уровень в ключевую позицию, поскольку он формирует основную границу между поставщиком и пользователем надежной службы передачи данных.

Примитивы транспортной службы

Чтобы пользователи могли получить доступ к транспортной службе, транспортный уровень должен совершать некоторые действия по отношению к прикладным программам, то есть предоставлять интерфейс транспортной службы. У всех транспортных служб есть свои интерфейсы. В этом разделе мы вначале рассмотрим простой (но гипотетический) пример транспортной службы и ее интерфейсов, просто чтобы узнать основные принципы и понятия. Следующий раздел будет посвящен реальному примеру.

Транспортная служба подобна сетевой, но имеет и некоторые существенные отличия. Главное отличие состоит в том, что сетевая служба предназначена для моделирования сервисов, предоставляемых реальными сетями, со всеми их особенностями. Реальные сети теряют пакеты, поэтому в общем случае сетевая служба ненадежна.

Ориентированная на соединение транспортная служба, напротив, является надежной. Конечно, реальные сети содержат ошибки, но именно транспортный уровень как раз и должен обеспечивать надежность сервисов ненадежных сетей.

В качестве примера рассмотрим два процесса, соединенных каналами в системе UNIX. Эти процессы предполагают, что соединение между ними идеально. Они не желают знать о подтверждениях, потерянных пакетах, заторах и т. п. Им требуется стопроцентно надежное соединение. Процесс A помещает данные в один конец канала, а процесс B извлекает их на другом. Именно для этого и предназначена ориентированная на соединение транспортная служба — скрывать несовершенство сетевого уровня, чтобы пользовательские процессы могли считать, что существует безошибочный поток битов.

Кстати, транспортный уровень может также предоставлять ненадежный (дейтаграммный) сервис, но о нем сказать почти нечего, поэтому мы в данной главе сконцентрируемся на транспортной службе, ориентированной на соединение. Тем не менее, некоторые приложения, например, клиент-серверные вычислительные системы и потоковое мультимедиа, даже выигрывают от дейтаграммных сервисов, поэтому далее мы еще упомянем их.

Второе различие между сетевой и транспортной службами состоит в том, для кого они предназначены. Сетевая служба используется только транспортными объектами. Мало кто пишет свои собственные транспортные объекты, и поэтому пользователи и программы почти не встречаются с голой сетевой службой. Транспортные примитивы, напротив, используются многими программами, а следовательно, и программистами. Поэтому транспортная служба должна быть удобной и простой в употреблении.

Чтобы получить представление о транспортной службе, рассмотрим пять примитивов, перечисленных в табл. 6.1. Этот транспортный интерфейс сильно упрощен, но он дает представление о назначении ориентированного на соединение транспортного интерфейса. Он позволяет прикладным программам устанавливать, использовать и освобождать соединения, чего вполне достаточно для многих приложений.

Таблица 6.1. Примитивы простой транспортной службы

Примитив	Посланный модуль данных транспортного протокола	Значение
LISTEN (ОЖИДАТЬ)	(нет)	Блокировать сервер, пока какой-либо процесс не попытается соединиться
CONNECT (СОЕДИНИТЬ)	ЗАПРОС СОЕДИНЕНИЯ	Активно пытаться установить соединение
SEND (ПОСЛАТЬ)	ДАННЫЕ	Послать информацию
RECEIVE (ПОЛУЧИТЬ)	(нет)	Блокировать сервер, пока не прибудут данные
DISCONNECT (РАЗЪЕДИНИТЬ)	ЗАПРОС РАЗЪЕДИНЕНИЯ	Прервать соединение

Чтобы понять, как могут быть использованы эти примитивы, рассмотрим приложение, состоящее из сервера и нескольких удаленных клиентов. Вначале сервер выполняет примитив LISTEN — обычно для этого вызывается библиотечная процедура, которая обращается к системе. В результате сервер блокируется, пока клиент не обратится к нему. Когда клиент хочет поговорить с сервером, он выполняет примитив CONNECT. Транспортный объект выполняет этот примитив, блокируя обратившегося к нему клиента и посылая пакет серверу. Поле данных пакета содержит сообщение транспортного уровня, адресованное транспортному объекту сервера.

Следует сказать пару слов о терминологии. За неимением лучшего термина, для сообщений, посылаемых одной транспортной сущностью другой транспортной сущности, нам придется использовать несколько неуклюжее сокращение **TPDU** (Transport Protocol Data Unit — модуль данных транспортного протокола). Модули данных, которыми обмениваются транспортные уровни, помещаются в пакеты (которыми обмениваются сетевые уровни). Эти пакеты, в свою очередь, содержатся в кадрах, которыми обмениваются уровни передачи данных. Получив кадр, уровень передачи данных обрабатывает заголовок кадра и передает содержимое поля полезной нагрузки кадра наверх, сетевой сущности. Сетевая сущность обрабатывает заголовок пакета и передает содержимое поля полезной нагрузки пакета наверх, транспортной сущности. Эта вложенность проиллюстрирована на рис. 6.2.

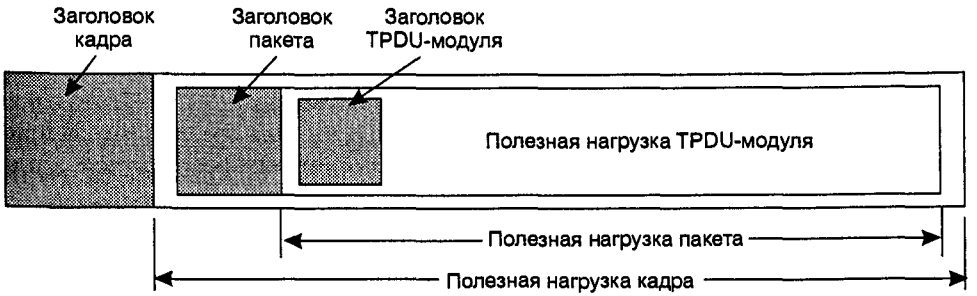


Рис. 6.2. Вложенность модулей данных транспортного протокола, пакетов и кадров

Итак, вернемся к нашему примеру общения клиента и сервера. В результате запроса клиента CONNECT серверу посылается модуль данных транспортного протокола, содержащий CONNECTION REQUEST (запрос соединения). Когда он прибывает, транспортная сущность проверяет, заблокирован ли сервер примитивом LISTEN (то есть заинтересован ли сервер в обработке запросов). Затем она разблокирует сервер и посылает обратно клиенту модуль данных CONNECTION ACCEPTED (соединение принято). Получив этот модуль, клиент разблокируется, после чего соединение считается установленным.

Теперь клиент и сервер могут обмениваться данными с помощью примитивов SEND и RECEIVE. В простейшем случае каждая из сторон может использовать блокирующий примитив RECEIVE для перехода в режим ожидания модуля данных, посылаемого противоположной стороной при помощи примитива SEND. Когда модуль данных прибывает, получатель разблокируется. Затем он может обработать полученный модуль и послать ответ. Такая схема прекрасно работает, пока обе стороны помнят, чей черед посылать, а чей — принимать.

Обратите внимание на то, что на сетевом уровне даже простая однонаправленная пересылка данных оказывается сложнее, чем на транспортном уровне. Каждый посланный пакет данных будет, в конце концов, подтвержден. Пакеты, содержащие управляющие модули данных, также подтверждаются, явно или неявно. Эти подтверждения управляются транспортными сущностями при помощи протокола сетевого уровня и не видны пользователям транспортного уровня. Аналогично транспортным сущностям нет необходимости беспокоиться о таймерах и повторных передачах. Все эти механизмы не видны пользователям транспортного уровня, для которых соединение представляется надежным битовым каналом. Один пользователь помещает в канал биты, которые волшебным образом появляются на другом конце канала. Эта способность скрывать сложность от пользователей свидетельствует о том, что многоуровневые протоколы являются довольно мощным инструментом.

Когда соединение больше не требуется, оно должно быть разорвано, чтобы можно было освободить место в таблицах двух транспортных сущностей. Разъединение существует в двух вариантах: симметричном и асимметричном. В асимметричном варианте любой пользователь транспортной службы может вызвать примитив DISCONNECT, в результате чего удаленной транспортной сущности будет послан управляющий модуль TPDU DISCONNECTION REQUEST (запрос разъединения). После получения модуля TPDU удаленной транспортной сущностью соединение разрывается.

В симметричном варианте каждое направление закрывается отдельно, независимо от другого. Когда одна сторона выполняет примитив DISCONNECT, это означает, что у нее больше нет данных для передачи, но что она все еще готова принимать данные от своего партнера. В этой схеме соединение разрывается, когда обе стороны выполняют примитив DISCONNECT.

Диаграмма состояний для установки и разрыва соединения показана на рис. 6.3. Каждый переход вызывается каким-то событием или примитивом, выполненным локальным пользователем транспортной службы или входящим пакетом. Для простоты мы будем считать, что каждый модуль TPDU подтверждается отдельно.

Мы также предполагаем, что используется модель симметричного разъединения, в которой клиент делает первый ход. Обратите внимание на простоту этой модели. Позднее мы рассмотрим более реалистичные модели.

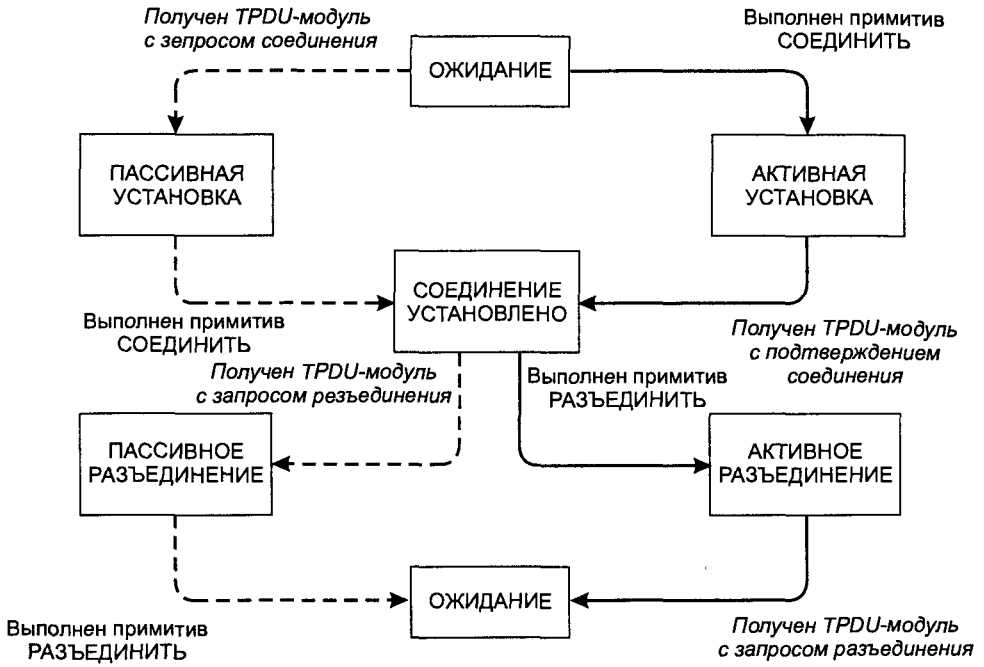


Рис. 6.3. Диаграмма состояний для простой схемы управления соединениями. Переходы, обозначенные курсивом, вызываются прибытием пакетов. Сплошными линиями показана последовательность состояний клиента. Пунктирными линиями показана последовательность состояний сервера

Сокеты Беркли

Теперь рассмотрим другой набор транспортных примитивов — примитивы сокетов (иногда называемых гнездами), используемые в операционной системе Berkeley UNIX для протокола TCP (Transmission Control Protocol — протокол управления передачей). Они приведены в табл. 6.2. Модель сокетов во многом подобна рассмотренной ранее модели транспортных примитивов, но обладает большей гибкостью и предоставляет больше возможностей. Модули TPDU, соответствующие этой модели, будут рассматриваться далее в этой главе, когда мы будем изучать TCP.

Первые четыре примитива списка выполняются серверами в таком же порядке. Примитив `SOCKET` создает новый сокет и выделяет для него место в таблице транспортной сущности. Параметры вызова указывают используемый формат адресов, тип требуемой услуги (например, надежный байтовый поток) и протокол. В случае успеха примитив `SOCKET` возвращает обычный описатель файла, ис-

пользуемого при вызове следующих примитивов, подобно тому, как возвращает описатель файла процедура OPEN.

Таблица 6.2. Примитивы сокетов для TCP

Примитив	Значение
SOCKET (СОКЕТ)	Создать новый сокет (гнездо связи)
BIND (СВЯЗАТЬ)	Связать локальный адрес с сокетом
LISTEN (ОЖИДАТЬ)	Объявить о желании принять соединение; указать размер очереди
ACCEPT (ПРИНЯТЬ)	Блокировать звонящего до получения попытки соединения
CONNECT (СОЕДИНИТЬ)	Активно пытаться установить соединение
SEND (ПОСЛАТЬ)	Посылать данные по соединению
RECEIVE (ПОЛУЧИТЬ)	Получать данные у соединения
CLOSE (ЗАКРЫТЬ)	Разрывать соединение

У только что созданного сокета нет сетевых адресов. Они назначаются с помощью примитива BIND. После того как сервер привязывает адрес к сокету, с ним могут связаться удаленные клиенты. Вызов SOCKET не создает адрес напрямую, так как некоторые процессы придают адресам большое значение (например, они использовали один и тот же адрес годами, и этот адрес всем известен), тогда как другим процессам это не важно.

Следом идет вызов LISTEN, который выделяет место для очереди входящих звонков на случай, если несколько клиентов попытаются соединиться одновременно. В отличие от примитива LISTEN в нашем первом примере, примитив LISTEN гнездовой модели не является блокирующим вызовом.

Чтобы заблокировать ожидание входящих соединений, сервер выполняет примитив ACCEPT. Получив TPDU-модуль с запросом соединения, транспортная сущность создает новый сокет с теми же свойствами, что и у исходного сокета, и возвращает описатель файла для него. При этом сервер может разветвить процесс или поток, чтобы обработать соединение для нового сокета и вернуться к ожиданию следующего соединения для оригинального сокета.

Теперь посмотрим на этот процесс со стороны клиента. В этом случае также сначала с помощью примитива SOCKET должен быть создан сокет, но примитив BIND здесь не требуется, так как используемый адрес не имеет значения для сервера. Примитив CONNECT блокирует вызывающего и инициирует активный процесс соединения. Когда этот процесс завершается (то есть когда соответствующий TPDU-модуль, посланный сервером, получен), процесс клиента разблокируется и соединение считается установленным. После этого обе стороны могут использовать примитивы SEND и RECV для передачи и получения данных по полнодуплексному соединению. Могут также применяться стандартные UNIX-вызовы READ и WRITE, если нет нужды в использовании специальных свойств SEND и RECV.

В модели сокетов используется симметричный разрыв соединения. Соединение разрывается, когда обе стороны выполняют примитив CLOSE.

Пример программирования сокета: файл-сервер для Интернета

В качестве примера использования вызовов сокета рассмотрим программу, демонстрирующую работу клиента и сервера, представленную в листинге 6.1. Имеется примитивный файл-сервер, работающий в Интернете и использующий его клиент. У программы много ограничений (о которых еще будет сказано), но, в принципе, данный код, описывающий сервер, может быть скомпилирован и запущен на любой UNIX-системе, подключенной к Интернету. Код, описывающий клиента, может быть запущен с определенными параметрами. Это позволит ему получить любой файл, к которому у сервера есть доступ. Файл отображается на стандартном устройстве вывода, но, разумеется, может быть перенаправлен на диск или какому-либо процессу.

Рассмотрим сперва ту часть программы, которая описывает сервер. Она начинается с включения некоторых стандартных заголовков, последние три из которых содержат основные структуры и определения, связанные с Интернетом. Затем SERVER_PORT определяется как 12 345. Значение выбрано случайным образом. Любое число от 1024 до 65 535 подойдет с не меньшим успехом, если только оно не используется каким-либо другим процессом. Понятно, что клиент и сервер должны обращаться к одному и тому же порту. Если сервер в один прекрасный день станет популярным во всем мире (что маловероятно, учитывая то, насколько он примитивен), ему будет присвоен постоянный порт с номером менее 1024, который появится на www.iana.org.

В последующих двух строках определяются две необходимые серверу константы. Первая из них задает размер участка данных для файловой передачи. Вторая определяет максимальное количество незавершенных соединений, после установки которых новые соединения будут отвергаться.

После объявления локальных переменных начинается сама программа сервера. Вначале она инициализирует структуру данных, которая будет содержать IP-адрес сервера. Эта структура будет связана с серверным сокетом. Вызов memset полностью обнуляет структуру данных. Последующие три присваивания заполняют три поля этой структуры. Последнее из них содержит порт сервера. Функции htonl и htons занимаются преобразованием значений в стандартный формат, что позволяет программе нормально выполняться на машинах с представлением числовых разрядов как в возрастающем порядке (например, SPARC), так и в убывающем (например, Pentium). Детали их семантики здесь роли не играют.

После этого сервером создается и проверяется на ошибки (определяется по $s < 0$) сокет. В конечной версии программы сообщение об ошибке может быть чуть более понятным. Вызов setsockopt нужен для того, чтобы порт мог использоваться несколько раз, а сервер — бесконечно, обрабатывая запрос за запросом. Теперь IP-адрес привязывается к сокету и выполняется проверка успешного завершения вызова bind. Конечным этапом инициализации является вызов listen, свидетельствующий о готовности сервера к приему входящих вызовов и сообщающий системе о том, что нужно ставить в очередь до QUEUE_SIZE вызовов, пока сервер обрабатывает текущий вызов. При заполнении очереди прибытие новых запросов спокойно игнорируется.

В этом месте начинается основной блок программы, который никогда не пропускается. Его можно остановить только извне. Вызов accept блокирует сервер на то время, пока клиент пытается установить соединение. Если вызов завершается успешно, accept возвращает дескриптор файла, который можно использовать для чтения и записи, аналогично тому, как файловые дескрипторы могут записываться и читаться в каналах. Однако, в отличие от однонаправленных каналов, сокеты двунаправлены, поэтому для чтения (и записи) данных из соединения надо использовать sa (адрес сокета).

После установки соединения сервер считывает имя файла. Если оно пока недоступно, сервер блокируется, ожидая его. Получив имя файла, сервер открывает файл и входит в цикл, который читает блоки данных из файла и записывает их в сокет. Это продолжается до тех пор, пока не будут скопированы все запрошенные данные. Затем файл закрывается, соединение разрывается, и начинается ожидание нового вызова. Данный цикл повторяется бесконечно.

Теперь рассмотрим часть кода, описывающую клиента. Чтобы понять, как работает программа, необходимо вначале разобраться, как она запускается. Если она называется client, ее типичный вызов будет выглядеть так:

```
client flits.cs.vu.nl /usr/tom/filename >f
```

Этот вызов сработает только в том случае, если сервер расположен по адресу flits.cs.vu.nl, файл usr/tom/filename существует и у сервера есть доступ для чтения этого файла. Если вызов произведен успешно, файл передается по Интернету и записывается на место f, после чего клиентская программа заканчивает свою работу. Поскольку серверная программа продолжает работать, клиент может запускать новые запросы на получение файлов.

Клиентская программа начинается с подключения файлов и объявлений. Работа начинается с проверки корректности числа аргументов (argc = 3 означает, что в строке запуска содержались имя программы и два аргумента). Обратите внимание на то, что argv[1] содержит имя сервера (например, flits.cs.vu.nl) и переводится в IP-адрес с помощью gethostbyname. Для поиска имени функция использует DNS. Мы будем изучать технологию DNS в главе 7. Затем создается и инициализируется сокет, после чего клиент пытается установить TCP-соединение с сервером посредством connect. Если сервер включен, работает на указанной машине, соединен с SERVER_PORT и либо простаивает, либо имеет достаточно места в очереди listen (очереди ожидания), то соединение с клиентом рано или поздно будет установлено. По данному соединению клиент передает имя файла, записывая его в сокет. Количество отправленных байтов на единицу превышает требуемое для передачи имени, поскольку нужен еще нулевой байт-ограничитель, с помощью которого сервер может понять, где кончается имя файла.

Теперь клиентская программа входит в цикл, читает файл блок за блоком из сокета и копирует на стандартное устройство вывода. По окончании этого процесса она просто завершается.

Процедура fatal выводит сообщение об ошибке и завершается. Серверу также требуется эта процедура, и она пропущена в листинге только из соображений экономии места. Поскольку программы клиента и сервера компилируются от-

дельно и в обычной ситуации запускаются на разных машинах, код процедуры fatal не может быть разделяемым.

Эти две программы (как и другие материалы, связанные с этой книгой) можно найти на веб-сайте книги по адресу http://www.prenhall.com/tanenbaum (ссылка рядом с изображением обложки). Их можно скачать и скомпилировать на любой UNIX-системе (например, Solaris, BSD, Linux). Делается это с помощью следующих командных строк:

```
cc -o client client.c -lsocket -lnsl
cc -o server server.c -lsocket -lnsl
```

Программу для сервера можно запустить, просто набрав

```
server
```

Клиентской программе нужны два аргумента, как описывалось ранее. На веб-сайте можно найти и Windows-версии программ.

Листинг 6.1. Программы использования сокетов для клиента и сервера

```
/* На этой странице содержится клиентская программа, запрашивающая файл у серверной
программы, расположенной на следующей странице. */
/* Сервер в ответ на запрос высылает файл.*/


#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345 /* По договоренности между клиентом и сервером */
#define BUF_SIZE 4096 /* Размер передаваемых блоков */

int main(int argc, char *argv)
{
  int c,s,bytes;
  char buf[BUF_SIZE];   /*буфер для входящего файла */
  struct hostent *h;    /*информация о сервере */
  struct sockaddr_in channel; /*хранит IP=адрес */

  if (argc!=3) fatal("Для запуска введите: клиент имя_сервера имя_файла");
  h = gethostbyname(argv[1]);     /* поиск IP-адреса хоста */
  if(!h) fatal("Ошибка выполнения gethostbyname")
  s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (s<0) fatal("Сокет");
  memset(&channel, 0, sizeof(channel));
  channel.sin_family=AF_INET;
  memcpy(&channel.sin_addr.s_addr,h>h_addr,h>h_length);
  channel.sin_port=htons(SERVER_PORT);

  c = connect(s,(struct sockaddr *) &channel, sizeof(channel));
  if (c<0) fatal("Ошибка соединения");

  /* Соединение установлено. Посылается имя файла с нулевым байтом на конце */
  write*s, argv[2], strlen(argv[2])+1);

  /* Получить файл, записать на стандартное устройство вывода */
```

```
  while (1) {
        bytes = read(s, buf, BUF_SIZE);  /* Читать из сокета */
        if (bytes <= 0) exit(0);         /* Проверка конца файла */
        write(1, buf, bytes);            /* Записать на стандартное устройство вывода
*/
  }
}
fatal(char *string)
{
  printf("%s\n", string);
  exit(1);
}

/* Код программы для сервера */
#include <sys/types.h>
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345  /* По договоренности между клиентом и сервером */
#define BUF_SIZE 4096      /* Размер передаваемых блоков */
#define QUEUE_SIZE 10

int main(int argc, char *argv[]);
{
  int s, b, l, fd, sa, bytes, on = 1;
  char buf[BUF_SIZE];            /* буфер для исходящего файла */
  struct sockaddr_in channel;    /* содержит IP-адрес */

  /* Создать структуру адреса для привязки к сокету */
  memset(&channel, 0, sizeof(channel));    /* Нулевой канал */
  channel.sin_family = AF_INET;
  channel.sin_addr.s_addr = htonl(INADDR_ANY);
  channel.sin_port = htons(SERVER_PORT);

  /* Пассивный режим. Ожидание соединения */
  s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP); /* создать сокет */
  if (s<0) fatal("ошибка сокета");
  setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));
  b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
  if (b<0) fatal("Ошибка связывания");

  l = listen(s, QUEUE_SIZE);    /* Определение размера очереди */
  if (l<0) fatal("Ошибка ожидания");

  /* Теперь сокет установлен и связан. Ожидание и обработка соединения */
  while (1) {
        sa = accept(s, 0, 0);   /* Блокировать обработку запроса */
        if (sa<0) fatal("Ошибка доступа");

        read(sa, buf, BUF_SIZE);        /* считать имя файла из сокета */

        /* Получить и вернуть файл */
        fd = open(buf, O_RDONLY);       /* Открыть файл для отсылки */
```

```
        if (fd < 0) fatal("Ошибка открытия файла");

        while (1) {
              bytes = read(fd, buf, BUF_SIZE); /* Читать из файла */
              if (bytes <= 0) break;           /* Проверка конца файла */
              write(sa, buf, bytes);           /* Записать байты в сокет */
        }
        close(fd);                             /* Закрыть файл */
        close(sa);                             /* Разорвать соединение */
  }
}
```

Кстати говоря, такой сервер построен далеко не по последнему слову техники. Осуществляемая проверка ошибок минимальна, а сообщения об ошибках реализованы весьма посредственно. Понятно, что ни о какой защите информации здесь говорить не приходится, а применение аскетичных системных вызовов UNIX — это не лучшее решение с точки зрения независимости от платформы. Делаются некоторые некорректные с технической точки зрения предположения, например, о том, что имя файла всегда поместится в буфер и будет передано без ошибок. Система будет обладать низкой производительностью, поскольку все запросы обрабатываются только последовательно (используется один поток запросов). Несмотря на эти недостатки, с помощью данной программы можно организовать полноценный работающий файл-сервер для Интернета. Более подробную информацию можно найти в (Stivens, 1997).

Элементы транспортных протоколов

Транспортная служба реализуется **транспортным протоколом**, используемым между двумя транспортными сущностями. В некоторых отношениях транспортные протоколы напоминают протоколы передачи данных, подробно изучавшиеся в главе 3. Все эти протоколы, наряду с другими вопросами, занимаются обработкой ошибок, управлением очередями и потоками.

Однако у протоколов разных уровней имеется и много различий, обусловленных различиями условий, в которых работают эти протоколы, как показано на рис. 6.4. На уровне передачи данных два маршрутизатора общаются напрямую по физическому каналу, тогда как на транспортном уровне физический канал заменен целой подсетью. Это отличие оказывает важное влияние на протоколы.

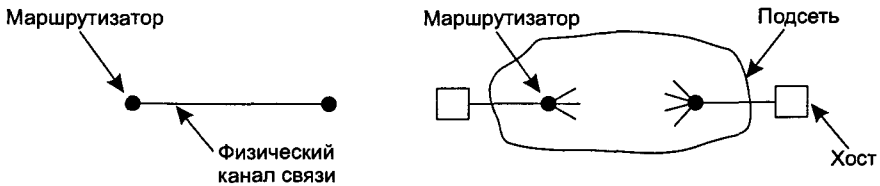


Рис. 6.4. Окружение уровня передачи данных (*а*); окружение транспортного уровня (*б*)

Во-первых, на уровне передачи данных маршрутизатору не требуется указывать, с каким маршрутизатором он хочет поговорить, — каждая выходная линия однозначно определяет маршрутизатор. На транспортном уровне требуется явно указывать адрес получателя.

Во-вторых, процесс установки соединения по проводу (рис. 6.4, *а*) прост: противоположная сторона всегда присутствует (если только она не вышла из строя). В любом случае, работы не очень много. На транспортном уровне начальная установка соединения, как будет показано далее, происходит более сложно.

Еще одно весьма досадное различие между уровнем передачи данных и транспортным уровнем состоит в том, что подсеть потенциально обладает возможностями хранения информации. Когда маршрутизатор посылает кадр, он может прибыть или потеряться, но кадр не может побродить где-то какое-то время, спрятаться в отдаленном уголке земного шара, а затем внезапно появиться в самый неподходящий момент 30 секунд спустя. Если подсеть использует дейтаграммы и адаптивную маршрутизацию, то всегда есть ненулевая вероятность того, что пакет будет храниться где-нибудь несколько секунд, а уже потом будет доставлен по назначению. Последствия способности подсети хранить пакеты иногда могут быть катастрофичными и требуют применения специальных протоколов.

Последнее различие между уровнем передачи данных и транспортным уровнем является скорее количественным, чем качественным. Буферизация и управление потоком необходимы на обоих уровнях, но наличие большого динамически изменяющегося количества соединений на транспортном уровне может потребовать принципиально другого подхода, нежели использовавшийся на уровне передачи данных. Некоторые протоколы, упоминавшиеся в главе 3, выделяют фиксированное количество буферов для каждой линии, так что, когда прибывает кадр, всегда имеется свободный буфер. На транспортном уровне из-за большого количества управляемых соединений идея выделения нескольких буферов каждому соединению выглядит не столь привлекательно. В следующих разделах мы изучим эти и другие важные вопросы.

Адресация

Когда один прикладной процесс желает установить соединение с другим прикладным процессом, он должен указать, с кем именно он хочет связаться. (У не требующей соединений транспортной службы проблемы такие же: кому следует посылать каждое сообщение?) Применяемый обычно метод состоит в определении транспортных адресов, к которым процессы могут посылать запросы на установку соединения. В Интернете такие конечные точки называются **портами**. В сетях ATM это точки доступа к службе **AAL-SAP** (Service Access Point). Мы будем пользоваться нейтральным термином **TSAP** (Transport Service Access Point — точка доступа к службам транспортного уровня). Аналогичные конечные точки сетевого уровня называются **NSAP** (Network Service Access Point — точка доступа к сетевому сервису). Примерами NSAP являются IP-адреса.

Рисунок 6.5 иллюстрирует взаимоотношения между NSAP, TSAP и транспортным соединением. Прикладные процессы как клиента, так и сервера могут

связываться с TSAP для установки соединения с удаленным TSAP. Такие соединения проходят через NSAP на каждом хосте, как показано на рисунке. TSAP нужны для того, чтобы различать конечные точки, совместно использующие NSAP, в сетях, где у каждого компьютера есть свой NSAP.

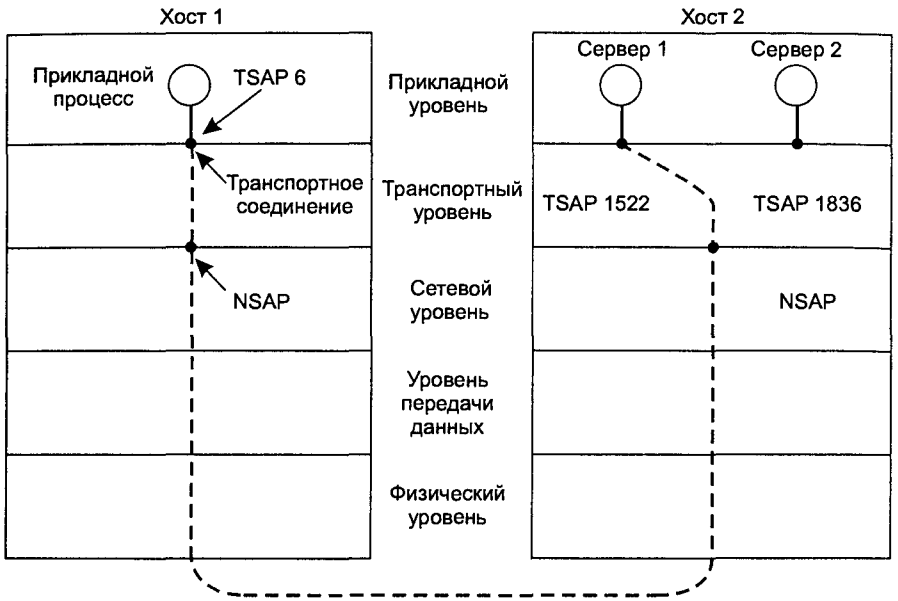


Рис. 6.5. Точки доступа к службам транспортного и сетевого уровня и транспортные соединения

Возможный сценарий для транспортного соединения выглядит следующим образом:

1. Серверный процесс хоста 2, сообщающий время суток, подсоединяется к точке доступа TSAP 1522 и ожидает входящего звонка. Вопрос о том, как процесс соединяется с TSAP, лежит за пределами сетевой модели и целиком зависит от локальной операционной системы. Например, может вызываться примитив, подобный LISTEN.
2. Прикладной процесс хоста 1 желает узнать, который час, поэтому он обращается к сети с запросом CONNECT, указывая TSAP 1208 в качестве адреса отправителя и TSAP 1522 в качестве адреса получателя. Это действие в результате приводит к установке транспортного соединения между прикладным процессом хоста 1 и сервером 1, расположенным на хосте 2.
3. Прикладной процесс отправляет запрос, надеясь выяснить, который час.
4. Сервер обрабатывает запрос и в качестве ответа посылает информацию о точном времени.
5. Транспортное соединение разрывается.

Обратите внимание на то, что на хосте 2 могут располагаться и другие серверы, соединенные со своими TSAP и ожидающие входящих запросов на соединение, приходящих с того же NSAP.

Нарисованная картинка всем хороша, но мы обошли стороной один маленький вопрос: как пользовательский процесс хоста 1 узнает, что сервер, сообщающий время, соединен с TSAP 1522? Возможно, сервер, сообщающий время, подключается к TSAP 1522 в течение долгих лет, и постепенно об этом узнают все пользователи сети. В этом случае службы имеют постоянные TSAP-адреса, хранящиеся в файлах, расположенных в известных местах, таких как etc/services в UNIX-системах. В файлах перечисляются серверы, за которыми жестко закреплены определенные порты.

Хотя постоянные TSAP-адреса могут хорошо подходить для небольшого количества никогда не меняющихся ключевых служб (например, таких как веб-сервер), в общем случае пользовательские процессы часто хотят пообщаться с другими пользовательскими процессами, существующими только в течение короткого времени и не обладающими постоянными TSAP-адресами, известным всем заранее. Кроме того, при наличии большого количества серверных процессов, большая часть которых редко используется, слишком расточительным делом оказывается поддержка всех их в активном состоянии с постоянными TSAP-адресами. То есть требуется другая модель.

Одна такая модель показана в упрощенном виде на рис. 6.6. Она называется **протоколом начального соединения**. Вместо того чтобы назначать всем возможным серверам хорошо известные TSAP-адреса, каждая машина, желающая предоставлять услуги удаленным пользователям, обзаводится специальным **обрабатывающим сервером**, действующим как прокси (посредник) для менее активно используемых серверов. Он прослушивает одновременно несколько портов, ожидая запроса на соединение. Потенциальные пользователи этой услуги начинают с того, что посылают запрос CONNECT, указывая TSAP-адрес нужной им службы. Если никакой сервер их не ждет, они получают соединение с обрабатывающим сервером, как показано на рис. 6.6, *а*.

Получив запрос, обрабатывающий сервер порождает подпроцесс на запрошенном сервере, позволяя ему унаследовать существующее соединение с пользователем. Новый сервер выполняет требуемую работу, в то время как обрабатывающий сервер возвращается к ожиданию новых запросов, как показано на рис. 6.6, *б*.

Хотя протокол начального соединения прекрасно работает с серверами, которые можно создавать по мере надобности, есть много ситуаций, в которых службы существуют независимо от обрабатывающего сервера. Например, файловый сервер должен работать на специальном оборудовании (машине с диском) и не может быть создан на ходу, когда кто-нибудь захочет к нему обратиться.

Чтобы справиться с этой ситуацией, часто используется другая схема. В этой модели используется специальный процесс, называющийся **сервером имен** или иногда **каталоговым сервером**. Чтобы найти TSAP-адрес, соответствующий данному имени службы, например «время суток», пользователь устанавливает соединение с сервером имен (TSAP-адрес которого всем известен). Затем пользователь посылает сообщение с указанием названия нужной ему услуги, и сервер

имен сообщает ему TSAP-адрес этой службы. После этого пользователь разрывает соединение с сервером имен и устанавливает новое соединение с нужной ему службой.

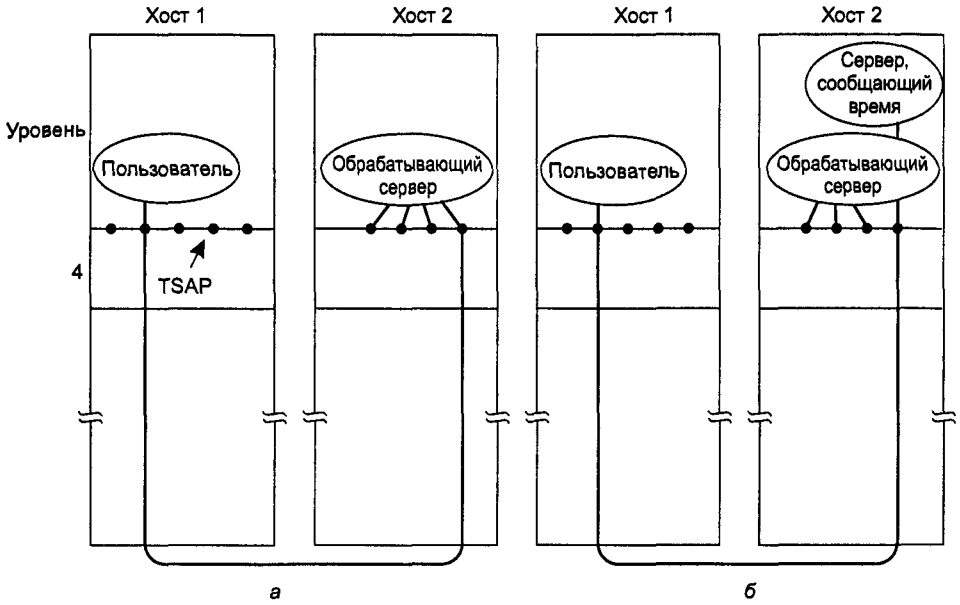


Рис. 6.6. Пользовательский процесс хоста 1 устанавливает соединение с сервером хоста 2

В этой модели, когда создается новая служба, она должна зарегистрироваться на сервере имен, сообщив ему название услуги (обычно строка ASCII) и TSAP-адрес. Сервер имен сохраняет полученную информацию в своей базе данных, чтобы иметь возможность отвечать на будущие запросы.

Функция сервера имен аналогична работе оператора телефонной справочной службы — он преобразует имена в номера. Как и в телефонной системе, важно, чтобы TSAP-адрес сервера имен (или обрабатывающего сервера в протоколе начального соединения) был действительно хорошо известен. Если вы не знаете номера телефонной справочной, вы не сможете позвонить оператору. Если вы полагаете, что номер справочной является очевидным, попытайтесь угадать его, находясь в другой стране.

Установка соединения

Установка соединения, хотя и просто звучит, неожиданно оказывается весьма непростым делом. На первый взгляд, должно быть достаточно одной транспортной сущности, для того чтобы послать адресату TPDU-модуль с запросом соединения CONNECTION REQUEST и услышать в ответ CONNECTION ACCEPTED (соединение принято). Неприятность заключается в том, что сеть может потерять, задержать или дублировать пакеты.

Представьте себе подсеть настолько перегруженную, что подтверждения практически никогда не доходят вовремя, каждый пакет опаздывает и пересылается повторно по два-три раза. Предположим, что подсеть основана на дейтаграммах и что каждый пакет следует по своему маршруту. Некоторые пакеты могут застрять в давке и прийти с большим опозданием.

Самый кошмарный сценарий выглядит следующим образом. Пользователь устанавливает соединение с банком и посылает сообщение с требованием банку перевести крупную сумму денег на счет не совсем надежного человека, после чего разрывает соединение. К несчастью, каждый пакет этого сценария дублируется и сохраняется в подсети. После разрыва соединения эти дубликаты пакетов наконец, добираются до адресата в нужном порядке. У банка нет способа определить, что это дубликаты. Он решает, что это вторая независимая транзакция, и еще раз переводит деньги. В оставшейся части этого раздела мы будем изучать проблему задержавшихся дубликатов, уделяя особое внимание алгоритмам, устанавливающим соединение надежным образом.

Основная проблема заключается в наличии задержавшихся дубликатов. Эту проблему можно попытаться решить несколькими способами, ни один из которых, на самом деле, не является удовлетворительным. Так, например, можно использовать одноразовые транспортные адреса. При таком подходе каждый раз, когда требуется транспортный адрес, генерируется новый адрес. Когда соединение разрывается, этот адрес уничтожается. Такая стратегия делает невозможной реализацию модели обрабатывающего сервера, изображенной на рис. 6.6.

Другая возможность состоит в том, что каждому соединению присваивается идентификатор соединения (то есть последовательный номер, который возрастает на единицу для каждого установленного соединения), выбираемый инициатором соединения и помещаемый в каждый TPDU-модуль, включая тот, который содержит запрос на соединение. После разрыва каждого соединения каждая транспортная сущность может обновить таблицу, в которой хранятся устаревшие соединения в виде пар (одноранговая транспортная сущность, идентификатор соединения). Для каждого приходящего запроса соединения может быть проверено, не хранится ли уже его идентификатор в таблице (он мог остаться там со времен разорванного ранее соединения).

К сожалению, у этой схемы есть существенный изъян: требуется, чтобы каждая транспортная сущность хранила неопределенно долго некоторое количество информации об истории соединений. Если машина выйдет из строя и потеряет свою память, она не сможет определить, какие соединения уже использовались, а какие нет.

Вместо этого можно применить другой подход. Следует разработать механизм, уничтожающий устаревшие заблудившиеся пакеты и не позволяющий им существовать в сети бесконечно долго. Если мы сможем гарантировать, что ни один пакет не сможет жить дольше определенного периода времени, проблема станет более управляемой.

Время жизни пакета может быть ограничено до известного максимума с помощью одного из следующих методов:

1. Проектирование подсети с ограничениями.

2. Помещение в каждый пакет счетчика транзитных участков.
3. Помещение в каждый пакет временного штампа.

К первому способу относятся все методы, предотвращающие зацикливание пакетов, в комбинации с ограничением задержки, вызванной перегрузкой по самому длинному возможному пути. Второй метод заключается в начальной установке счетчика на определенное значение и уменьшении на единицу этого значения на каждом маршрутизаторе. Сетевой протокол передачи данных просто игнорирует все пакеты, значение счетчика которых дошло до нуля. Третий метод состоит в том, что в каждый пакет помещается время его создания, а маршрутизаторы договариваются игнорировать все пакеты старше определенного времени. Для последнего метода требуется синхронизация часов маршрутизаторов, что само по себе является нетривиальной задачей. Впрочем, иногда синхронизацию удается производить с помощью внешнего источника, например GPS или радиостанции, передающей сигналы точного времени.

На практике нужно гарантировать не только то, что пакет мертв, но и что все его подтверждения также мертвы. Поэтому вводится некий интервал времени T, который в несколько раз превышает максимальное время жизни пакета. На какое число умножается максимальное время жизни пакета, зависит от протокола, и это влияет только на длительность интервала времени T. Если подождать в течение интервала времени T секунд момента отправки пакета, то можно быть уверенным, что все его следы уничтожены и что пакет не возникнет вдруг как гром среди ясного неба.

При ограниченном времени жизни пакетов можно разработать надежный способ безопасной установки соединений. Автором описанного далее метода является Томлинсон (Tomlinson) (1975). Этот метод решает проблему, но привносит некоторые собственные странности. Впоследствии (в 1978 году) этот метод был улучшен Саншайном (Sunshine) и Далалом (Dalal). Его варианты широко применяются на практике, и одним из примеров применения является TCP.

Чтобы обойти проблему потери машиной памяти предыдущих состояний (при выходе ее из строя), Томлинсон предложил снабдить каждый хост часами. Часы разных хостов нужно было синхронизировать. Предполагалось, что часы представляют собой двоичный счетчик, увеличивающийся через равные интервалы времени. Кроме того, число разрядов счетчика должно равняться числу битов в последовательных номерах (или превосходить его). Последнее и самое важное предположение состоит в том, что часы продолжают идти, даже если хост зависает.

Основная идея заключается в том, что два одинаково пронумерованных TPDU-модуля никогда не отправляются одновременно. При установке соединения младшие k битов часов используются в качестве начального порядкового номера (также k битов). Таким образом, в отличие от протоколов, описанных в главе 3, каждое соединение начинает нумерацию своих TPDU-модулей с разных чисел. Диапазон этих номеров должен быть достаточно большим, чтобы к тому моменту, когда порядковые номера сделают полный круг, старые TPDU-модули с такими же номерами уже давно исчезли. Линейная зависимость порядковых номеров от времени показана на рис. 6.7.

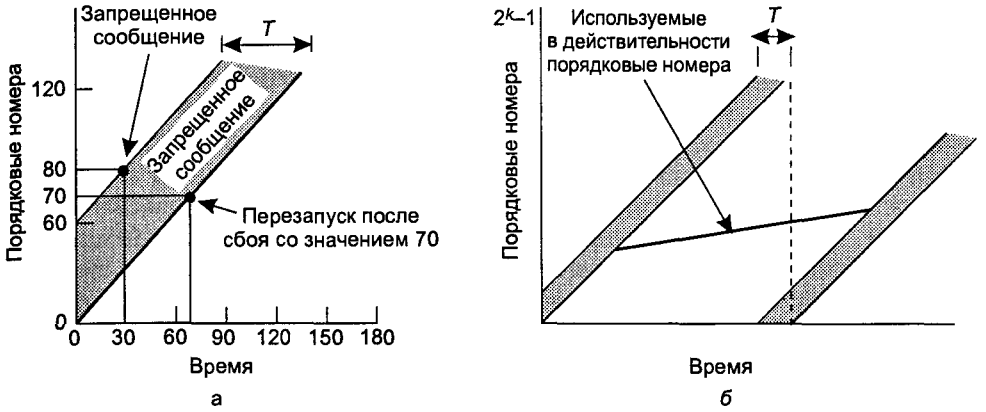


Рис. 6.7. TPDU-модули не могут заходить в запретную зону (а); проблема ресинхронизации (*б*)

Как только обе транспортные сущности договариваются о начальном порядковом номере, для управления потоком данных может применяться любой протокол скользящего окна. В действительности график порядковых номеров (показанный жирной линией) не прямой, а ступенчатый, так как показания часов увеличиваются дискретно. Впрочем, для простоты мы проигнорируем эту деталь.

Проблема возникает при выходе хоста из строя. Когда он снова включается, его транспортная сущность не помнит, где она находилась в пространстве порядковых номеров. Одно из решений заключается в том, что транспортная сущность должна подождать T секунд после восстановления, чтобы всех старых TPDU-модулей в сети не осталось. Однако в сложных сетях такая стратегия выглядит непривлекательно, так как значение T может оказаться довольно большим.

Чтобы не терять дополнительно T секунд после восстановления, необходимо ввести новое ограничение на использование порядковых номеров. Легче всего понять необходимость этого ограничения на примере. Пусть максимальное время жизни пакета $T = 60$ с, а часы тикают один раз в секунду. Как показано жирной линией на рис. 6.7, *а*, начальный порядковый номер для соединения, открываемого в момент времени x, будет равен x. Представим себе, что в момент времени $t = 30$ с по соединению номер 5 (открытому ранее) посылается обычный информационный TPDU-модуль, которому дается порядковый номер 80. Назовем его TPDU X. Немедленно после TPDU X хост сбрасывается и затем быстро перезагружается. В момент времени $t = 60$ с он начинает повторно открывать соединения с 0 по 4. В момент времени $t = 70$ с хост открывает повторно соединение 5, используя, как и требуется, начальный порядковый номер 70. В течение последующих 15 с он посылает TPDU-модули с порядковыми номерами от 70 до 80. Таким образом, в момент времени $t = 85$ с новый TPDU-модуль с порядковым номером 80 и номером соединения 5 попадает в подсеть. К несчастью, TPDU X еще жив. Если он прибудет к получателю ранее нового TPDU 80, TPDU X будет принят, а правильный TPDU 80 будет отвергнут как дубликат.

Чтобы избежать подобных ситуаций, необходимо запретить использование порядковых номеров на период времени T. Недопустимые комбинации времени и порядкового номера обозначены на рис. 6.7, *а* в виде **запретной зоны**. Прежде чем послать TPDU-модуль по любому соединению, транспортная сущность должна сначала прочитать показания часов и убедиться, что она не находится в запретной зоне.

Неприятности у протокола могут возникнуть по двум причинам. Если хост посылает слишком быстро и слишком много данных, кривая используемых в действительности порядковых номеров может оказаться круче линии зависимости начальных номеров от времени. Это означает, что скорость передачи данных в каждом открытом соединении должна быть ограничена одним TPDU-модулем за единицу времени. Кроме того, это означает, что транспортная сущность после восстановления, прежде чем открывать новое соединение, должна подождать, пока изменят свое состояние часы, чтобы один и тот же номер не использовался дважды. Следовательно, интервал изменения состояния часов должен быть коротким (несколько миллисекунд).

К сожалению, в запретную зону можно попасть не только снизу, передавая данные слишком быстро. Как видно из рис. 6.7, *б*, при любой скорости передачи данных, меньшей скорости часов, кривая используемых в действительности порядковых номеров попадет в запретную зону слева. Чем круче наклон этой кривой, тем дольше придется ждать этого события. Как уже упоминалось, непосредственно перед отправкой TPDU-модуля транспортная сущность должна проверить, не попадет ли она в ближайшее время в запретную зону, — и, если она находится близко от запретной зоны, отложить оправку TPDU-модуля на T секунд либо изменить синхронизацию порядковых номеров.

Использующий показания часов метод решает проблему опаздывающих дубликатов для информационных TPDU-модулей, но чтобы воспользоваться этим методом, соединение необходимо вначале установить. Так как управляющие TPDU-модули также могут задержаться в пути, возникает потенциальная проблема договоренности обеих сторон о начальном порядковом номере. Предположим, что для установления соединения хост 1 посылает TPDU-модуль с запросом соединения CONNECTION REQUEST, содержащим предлагаемый начальный порядковый номер и номер порта получателя, удаленному хосту 2. Получатель подтверждает получение этого запроса, посылая обратно TPDU-модуль CONNECTION ACCEPTED (соединение принято). Если оригинальный TPDU-модуль CONNECTION REQUEST будет потерян, а его дубликат неожиданно появится на хосте 2, соединение будет установлено некорректно.

Для разрешения этой проблемы Томлинсон (1975) предложил **«тройное рукопожатие»**. Этот протокол установки соединения не требует, чтобы обе стороны начинали передачу с одинаковыми порядковыми номерами, поэтому он может применяться вместе с методами синхронизации, отличными от метода глобальных часов. Нормальная процедура установки соединения показана на рис. 6.8, *а*. Хост 1 инициирует установку, выбирая порядковый номер x, и посылает TPDU-модуль CONNECTION REQUEST, содержащий этот начальный порядковый номер, хосту 2. Хост 2 отвечает TPDU-модулем ACK, подтверждая x и объявляя свой начальный порядковый номер y. Наконец, хост 1 подтверждает выбранный

хостом 2 начальный порядковый номер в первом посылаемом им информационном TPDU-модуле.

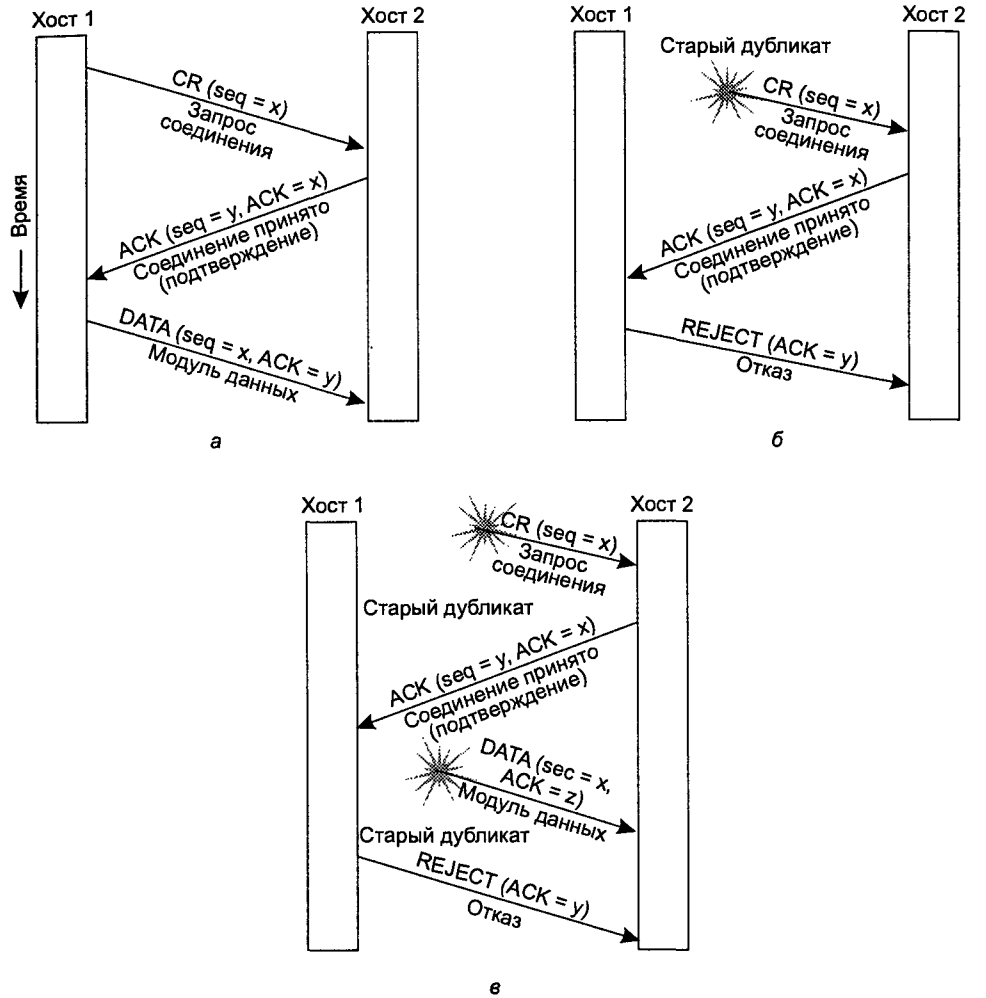


Рис. 6.8. Три сценария установки соединения с помощью «тройного рукопожатия». CR означает CONNECTION REQUEST. Нормальная работа (*а*); появление старого дубликата CONNECTION REQUEST (*б*); дубликат модуля CONNECTION REQUEST и дубликат модуля ACK (*в*)

Рассмотрим теперь работу «тройного рукопожатия» в присутствии задержавшегося дубликата управляющего TPDU-модуля. На рис. 6.8, *б* первый TPDU-модуль представляет собой задержавшийся дубликат модуля CONNECTION REQUEST от старого соединения. Этот TPDU-модуль прибывает на хост 2 тайком от хоста 1. Хост 2 реагирует на этот TPDU-модуль отправкой хосту 1 TPDU-модуля ACK, таким образом прося хост 1 подтвердить, что тот действительно пытался установить новое соединение. Когда хост 1 отказывается это сделать, хост 2 пони-

мает, что он был обманут задержавшимся дубликатом, и прерывает соединение. Таким образом, задержавшийся дубликат не причиняет вреда.

При наихудшем сценарии оба TPDU-модуля — CONNECTION REQUEST и ACK — блуждают по подсети. Этот случай показан на рис. 6.8, *в*. Как и в предыдущем примере, хост 2 получает задержавшийся модуль CONNECTION REQUEST и отвечает на него. В этом месте следует обратить внимание на то, что хост 2 предложил использовать y в качестве начального порядкового номера для трафика от хоста 2 к хосту 1, хорошо зная, что TPDU-модулей, содержащих порядковый номер y, или их подтверждений в данный момент в сети нет. Когда хост 2 получает второй задержавшийся TPDU-модуль, он понимает, что это дубликат, так как в этом модуле подтверждается не y, а z. Здесь важно понять, что не существует такой комбинации TPDU-модулей, которая заставила бы протокол ошибиться и случайно установить соединение, когда оно никому не нужно.

Разрыв соединения

Разорвать соединение проще, чем установить. Тем не менее, здесь также имеются подводные камни. Как уже было сказано, существует два стиля разрыва соединения: асимметричный и симметричный. Асимметричный разрыв связи соответствует принципу работы телефонной системы: когда одна из сторон вешает трубку, связь прерывается. При симметричном разрыве соединение рассматривается в виде двух отдельных однонаправленных связей, и требуется раздельное завершение каждого соединения.

Асимметричный разрыв связи является внезапным и может привести к потере данных. Рассмотрим сценарий, показанный на рис. 6.9. После установки соединения хост 1 посылает TPDU-модуль, который успешно добирается до хоста 2. Затем хост 1 посылает другой TPDU-модуль. К несчастью, хост 2 посылает DISCONNECTION REQUEST (запрос разъединения) прежде, чем прибывает второй TPDU-модуль. В результате соединение разрывается, а данные теряются.

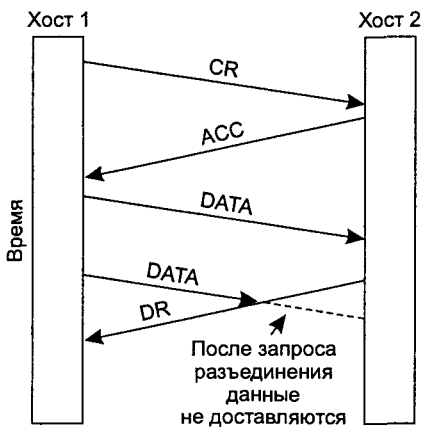


Рис. 6.9. Внезапное разъединение с потерей данных

Очевидно, требуется более сложный протокол, позволяющий избежать потери данных. Один из способов состоит в использовании симметричного разъединения, при котором каждое направление разъединяется независимо от другого. В этом случае хост может продолжать получать данные даже после того, как сам послал запрос разъединения.

Симметричное разъединение хорошо подходит для тех случаев, когда у каждой стороны есть фиксированное количество данных для передачи и каждая сторона точно знает, когда эти данные заканчиваются. В других случаях определить, что работа закончена и соединение может быть прервано, не так просто. Можно представить себе протокол, в котором хост 1 говорит: «Я закончил. Вы тоже закончили?» Если хост 2 отвечает: «Я тоже закончил. До свидания», соединение можно безо всякого риска разъединять.

К сожалению, этот протокол работает не всегда. Существует знаменитая проблема, называемая **проблемой двух армий**. Представьте, что армия белых расположилась в долине, как показано на рис. 6.10. На возвышенностях по обеим сторонам долины расположились две армии синих. Белая армия больше, чем любая из армий синих, но вместе синие превосходят белых. Если любая из армий синих атакует белых в одиночку, она потерпит поражение, но если синие сумеют атаковать белых одновременно, они могут победить.

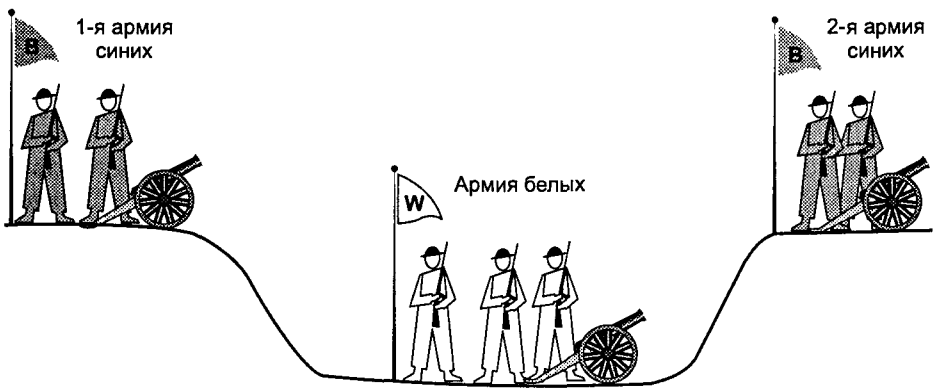


Рис. 6.10. Проблема двух армий

Синие армии хотели бы синхронизировать свое выступление. Однако единственный способ связи заключается в отправке вестового пешком по долине, где он может быть схвачен, а донесение потеряно (то есть приходится пользоваться ненадежным каналом). Спрашивается: существует ли протокол, позволяющий армиям синих победить?

Предположим, командир 1-й армии синих посылает следующее сообщение: «Я предлагаю атаковать 29 марта, на рассвете. Сообщите ваше мнение». Теперь предположим, что сообщение успешно доставляется и что командир 2-й армии синих соглашается, а его ответ успешно доставляется обратно в 1-ю армию синих. Состоится ли атака? Вероятно, нет, так как командир 2-й армии не уверен,

что его ответ получен. Если нет, то 1-я армия синих не будет атаковать, и было бы глупо с его стороны в одиночку ввязываться в сражение.

Теперь улучшим протокол с помощью «тройного рукопожатия». Инициатор оригинального предложения должен подтвердить ответ. Но и в этом случае останется неясным, было ли доставлено последнее сообщение. Протокол четырехкратного рукопожатия здесь также не поможет.

В действительности, можно доказать, что протокола, решающего данную проблему, не существует. Предположим, что такой протокол все же существует. В этом случае последнее сообщение протокола либо является важным, либо нет. Если оно не является важным, удалим его (а также все остальные несущественные сообщения), пока не останется протокол, в котором все сообщения являются существенными. Что произойдет, если последнее сообщение не дойдет до адресата? Мы только что сказали, что сообщение является важным, поэтому, если оно потеряется, атака не состоится. Поскольку отправитель последнего сообщения никогда не сможет быть уверен в его получении, он не станет рисковать. Другая синяя армия это знает и также воздержится от атаки.

Чтобы увидеть, какое отношение проблема двух армий имеет к разрыву соединения, просто замените слово «атаковать» на «разъединить». Если ни одна из сторон не готова разорвать соединение до тех пор, пока она не уверена, что другая сторона также готова к этому, то разъединение не произойдет никогда.

На практике к разрыву соединения обычно готовятся лучше, чем к атаке, поэтому ситуация не совсем безнадежна. На рис. 6.11 показаны четыре сценария разъединения, использующих «тройное рукопожатие». Хотя этот протокол и не безошибочен, обычно он работает успешно.

На рис. 6.11, *а* показан нормальный случай, в котором один из пользователей посылает запрос разъединения DR (DISCONNECTION REQUEST), чтобы инициировать разрыв соединения. Когда он прибывает, получатель посылает обратно также запрос разъединения DR и включает таймер на случай, если запрос потеряется. Когда запрос прибывает, первый отправитель посылает в ответ на него TPDU-модуль с подтверждением ACK и разрывает соединение. Наконец, когда прибывает ACK, получатель также разрывает соединение. Разрыв соединения означает, что транспортная сущность удаляет информацию об этом соединении из своей таблицы открытых соединений и сигнализирует о разрыве соединения владельцу соединения (пользователю транспортной службы). Эта процедура отличается от использования пользователем примитива DISCONNECT.

Если последний TPDU-модуль с подтверждением теряется (рис. 6.11, *б*), ситуацию спасает таймер. Когда время истекает, соединение разрывается в любом случае.

Теперь рассмотрим случай потери второго запроса разъединения DR. Пользователь, инициировавший разъединение, не получит ожидаемого ответа, у него истечет время ожидания, и он начнет все сначала. На рис. 6.11, *в* показано, как это происходит в случае, если все последующие запросы и подтверждения успешно доходят до адресатов.

Последний сценарий (рис. 6.11, *г*) аналогичен предыдущему — с той лишь разницей, что в этом случае предполагается, что все повторные попытки передать

запрос разъединения DR также терпят неудачу, поскольку все TPDU-модули теряются. После N повторных попыток отправитель наконец сдается и разрывает соединение. Тем временем у получателя также истекает время, и он тоже разрывает соединение.

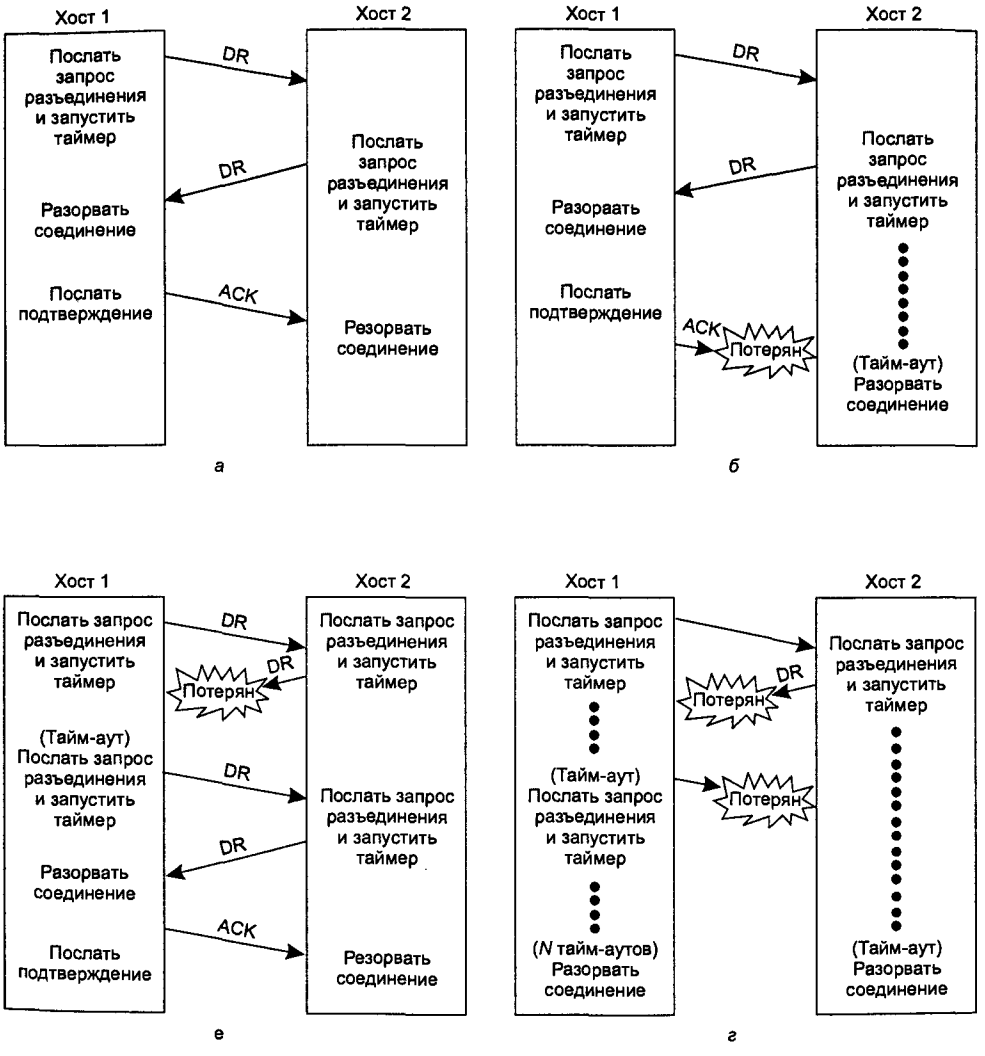


Рис. 6.11. Четыре сценария разрыва соединения: нормальный случай «тройного рукопожатия» (а); потеряно последнее подтверждение (*б*); потерян ответ (*в*); потерян ответ и последующие запросы (*г*)

Хотя такого протокола обычно бывает вполне достаточно, теоретически он может ошибиться, если потеряются начальный запрос разъединения DR и все N повторных попыток. Отправитель сдается и разрывает соединение, тогда как

другая сторона ничего не знает о попытках разорвать связь и сохраняет активность. В результате получается полуоткрытое соединение.

Этой ситуации можно было бы избежать, если не позволять отправителю сдаваться после N повторных попыток, а заставить его продолжать попытки, пока не будет получен ответ. Однако если другой стороне будет разрешено разрывать связь по таймеру, тогда отправитель действительно будет вечно повторять попытки, так как ответа он не получит никогда. Если же получающей стороне также не разрешать разрывать соединение по таймеру, тогда протокол зависнет в ситуации, изображенной на рис. 6.11, *г*.

Чтобы удалять полуоткрытые соединения, можно применять правило, гласящее, что если по соединению в течение определенного времени не прибывает ни одного TPDU-модуля, соединение автоматически разрывается. Таким образом, если одна сторона отсоединится, другая обнаружит отсутствие активности и также отсоединится. Для реализации этого правила каждая сторона должна управлять таймером, перезапускаемым после передачи каждого TPDU-модуля. Если этот таймер срабатывает, посылается пустой TPDU-модуль — лишь для того, чтобы другая сторона не повесила трубку. С другой стороны, если применяется правило автоматического разъединения и теряется слишком большое количество TPDU-модулей подряд, то соединение автоматически разрывается сначала одной стороной, а затем и другой.

На этом мы заканчиваем обсуждение этого вопроса, но теперь должно быть ясно, что разорвать соединение без потери данных не так просто, как это кажется на первый взгляд.

Управление потоком и буферизация

Изучив процессы установки и разрыва соединения, рассмотрим, как происходит управление соединением во время его использования. Одним из ключевых вопросов, уже обсуждавшихся ранее, является управление потоком. В некоторых аспектах проблема управления потоком на транспортном уровне аналогична той же проблеме на уровне передачи данных, но в то же время они различаются по целому ряду аспектов. Основное сходство состоит в том, что на обоих уровнях для согласования скорости передатчика и приемника требуется некая схема, например протокол скользящего окна. Основным различием является то, что у маршрутизатора обычно относительно небольшое количество линий, тогда как хост может иметь большое число соединений. Из-за этого различия использование на транспортном уровне стратегии буферизации, применяемой на уровне передачи данных, является непрактичным.

В протоколах передачи данных, обсуждавшихся в главе 3, кадры буферировались как отправляющим, так и получающим маршрутизаторами. Например, в протоколе 6 и отправитель, и получатель должны были отвести по MAX_SEQ + 1 буферов для каждой линии — половину для входного потока, половину для выходного. Так, для хоста с максимальным количеством соединений, равным 64, и 4-битовым порядковым номером этот протокол потребует 1024 буфера.

На уровне передачи данных отправитель должен буферизировать выходные кадры, так как может понадобиться их повторная передача. Если подсеть предоставляет дейтаграммные услуги, отправляющая транспортная сущность должна также использовать буфер по той же самой причине. Если получатель знает, что отправитель хранит в буферах все TPDU-модули до тех пор, пока они не будут подтверждены, тогда получатель сможет использовать свои буферы по своему усмотрению. Например, получатель может содержать единый буферный накопитель, используемый всеми соединениями. Когда приходит TPDU-модуль, предпринимается попытка динамически выделить ему новый буфер. Если это удается, то TPDU-модуль принимается, в противном случае он отвергается. Поскольку отправитель готов к тому, чтобы передавать потерянные TPDU-модули повторно, игнорирование TPDU-модулей получателем не наносит вреда, хотя и расходует некоторые ресурсы. Отправитель просто повторяет попытки до тех пор, пока не получит подтверждения.

В итоге, если сетевая служба является ненадежной, отправитель должен буферизировать все посланные TPDU-модули, как и на уровне передачи данных. Однако при надежной сетевой службе возможны другие варианты. В частности, если отправитель знает, что у получателя всегда есть место в буфере, ему не нужно хранить копии посланных TPDU-модулей. Однако если получатель не может гарантировать, что каждый приходящий TPDU-модуль будет принят, получателю придется буферизировать посланные TPDU-модули. В последнем случае отправитель не может доверять подтверждениям сетевого уровня, так как они означают лишь то, что TPDU-модуль прибыл, но не означают, что он был принят. Позднее мы вернемся к этому важному пункту.

Даже если получатель соглашается буферизовать принимаемые TPDU-модули, остается вопрос о том, какого размера должен быть буфер. Если большинство TPDU-модулей имеют примерно одинаковые размеры, естественно организовать буферы в виде массива буферов равной величины, каждый из которых может вместить один TPDU-модуль, как показано на рис. 6.12, *а*. Однако если TPDU-модули сильно различаются по размеру — от нескольких символов, набранных на терминале, до нескольких тысяч символов при передаче файлов, — то массив из буферов фиксированного размера окажется неудобным. Если размер буфера выбирать равным наибольшему возможному TPDU-модулю, то при хранении небольших TPDU-модулей память будет расходоваться неэффективно. Если же сделать размер буфера меньшим, тогда для хранения большого TPDU-модуля потребуется несколько буферов с сопутствующими сложностями.

Другой метод решения указанной проблемы состоит в использовании буферов переменного размера, как показано на рис. 6.12, *б*. Преимущество этого метода заключается в более оптимальном использовании памяти, но платой за это является усложненное управление буферами. Третий вариант состоит в выделении соединению единого большого циклического буфера, как показано на рис. 6.12, *в*. Такая схема также довольно хорошо использует память, если все соединения сильно нагружены, однако при невысокой нагрузке некоторых соединений ее эффективность снижается.

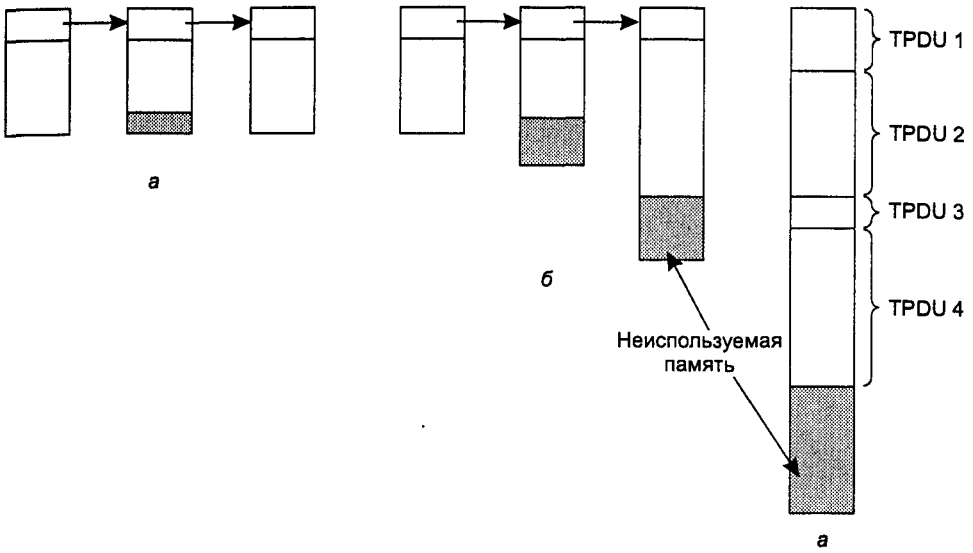


Рис. 6.12. Цепочка буферов фиксированного размера (*а*); цепочка буферов фиксированного размера (*б*); один большой циклический буфер для одного соединения (*в*)

Выбор компромиссного решения между буферизацией у отправителя и у получателя зависит от типа трафика соединения. Если трафик импульсный, небольшой мощности, как, например, трафик интерактивного терминала, лучше не выделять никаких буферов, а получать их динамически на обоих концах. Так как отправитель не уверен, что получатель сможет получить буфер, он должен будет сохранять копию TPDU-модуля, пока не получит относящееся к нему подтверждение. С другой стороны, при передаче файла будет лучше, если получатель выделит целое окно буферов, чтобы данные могли передаваться с максимальной скоростью. Таким образом, при пульсирующем трафике малой мощности буферизацию лучше производить у отправителя, а для трафика с постоянной большой скоростью — у получателя.

При открытии и закрытии соединений и при изменении формы трафика отправитель и получатель должны динамически изменять выделенные буферы. Следовательно, транспортный протокол должен позволять отправителю посылать запросы на выделение буфера на другом конце. Буферы могут выделяться для каждого соединения или коллективно на все соединения между двумя хостами. В качестве альтернативы запросам буферов получатель, зная состояние своих буферов, но не зная, какой трафик ему будет предложен, может сообщить отправителю, что он зарезервировал для него X буферов. При увеличении числа открытых соединений может потребоваться уменьшить количество или размеры выделенных буферов. Протокол должен предоставлять такую возможность.

В отличие от протоколов скользящего окна, описанных в главе 3, для реализации динамического выделения буферов следует отделить буферизацию от подтверждений. Динамическое выделение буферов означает, на самом деле, использование окна переменного размера. Вначале отправитель, основываясь на своих

потребностях, запрашивает определенное количество буферов. Получатель выделяет столько, сколько может. При отправлении каждого TPDU-модуля отправитель должен уменьшать на единицу число буферов, а когда это число достигнет нуля, он должен остановиться. Получатель отправляет обратно на попутных TPDU-модулях отдельно подтверждения и информацию об имеющихся у него свободных буферах.

На рис. 6.13 показан пример управления динамическим окном в дейтаграммной подсети с 4-битными порядковыми номерами. Предположим, что запрос на предоставление буферов пересылается в отдельных TPDU-модулях, а не добирается «автостопом» на попутных модулях. Вначале хост *A* запрашивает 8 буферов, но ему выделяется только 4. Затем он посылает три TPDU-модуля, из которых последний теряется. На шаге 6 хост *A* получает подтверждение получения посланных им TPDU-модулей 0 и 1, разрешает хосту *A* освободить буферы и послать еще три модуля (с порядковыми номерами 2, 3 и 4). Хост *A* знает, что TPDU-модуль номер 2 он уже посылал, поэтому он думает, что может послать модули 3 и 4, что он и делает. На этом шаге он блокируется, так как его счетчик буферов достиг нуля, и ждет предоставления новых буферов. На шаге 9 наступает таймаут хоста *A*, так как он до сих пор не получил подтверждения для TPDU-модуля 2. Этот модуль посылается еще раз. В строке 10 хост *B* подтверждает получение всех TPDU-модулей, включая 4-й, но отказывается предоставлять буферы хосту *A*. Такая ситуация невозможна в протоколах с фиксированным размером окна, описанных в главе 3. Следующий TPDU-модуль, посланный хостом *B*, разрешает хосту *A* передать еще один TPDU-модуль.

	A	Сообщение	B	Комментарии
1	→	< request 8 buffers>	→	*A* хочет 8 буферов
2	←	<ack = 15, buf = 4>	←	*B* позволяет переслать только сообщения 0–3
3	→	<seq = 0, data = m0>	→	У *A* теперь осталось 3 буфера
4	→	<seq = 1, data = m1>	→	У *A* теперь осталось 2 буфера
5	→	<seq = 2, data = m2>	•••	Сообщения потерялось, но *A* думает, что у него остался 1 буфер
6	←	<ack = 1, buf = 3>	←	*B* подтверждает получение модулей 0 и 1, разрешает передать со 2-го по 4-й
7	→	<seq = 3, data = m3>	→	У *A* остался буфер
8	→	<seq = 4, data = m4>	→	У *A* осталось 0 буферов, и он должен остановиться
9	→	<seq = 2, data = m2>	→	У *A* истекло время ожидания, и он передает еще раз
10	←	<ack = 4, buf = 0>	←	Все модули подтверждены, и он должен остановиться
11	←	<ack = 4, buf = 1>	←	Теперь *A* может послать модуль 5
12	←	<ack = 4, buf = 2>	←	*B* где-то нашел новый буфер
13	→	<seq = 5, data = m5>	→	У *A* остался 1 буфер
14	→	<seq = 6, data = m6>	→	*A* снова блокирован
15	←	<ack = 6, buf = 0>	←	*A* все еще блокирован
16	•••	<ack = 6, buf = 4>	←	Потенциальный тупик

Рис. 6.13. Динемическое выделение буферов. Стрелками показано направление передачи. Многоточие (...) означает потерянный TPDU-модуль

Потенциальные проблемы при такой схеме выделения буферов в дейтаграммных сетях могут возникнуть при потере управляющего TPDU-модуля. Взгляните на строку 16. Хост *B* выделил хосту *A* дополнительные буферы, но сообщение об этом было потеряно. Поскольку получение управляющих TPDU-модулей не подтверждается и, следовательно, управляющие TPDU-модули не посылаются повторно по тайм-ауту, хост *A* теперь оказался блокированным всерьез и надолго. Для предотвращения такой тупиковой ситуации каждый хост должен периодически посылать управляющий TPDU-модуль, содержащий подтверждение и состояние буферов для каждого соединения. Это позволит в конце концов выбраться из тупика.

До сих пор мы по умолчанию предполагали, что единственное ограничение, накладываемое на скорость передачи данных, состоит в количестве свободного буферного пространства у получателя. По мере все продолжающегося снижения цен на микросхемы памяти и винчестеры становится возможным оборудовать хосты таким количеством памяти, что проблема нехватки буферов будет возникать очень редко, если вообще будет возникать.

Если размер буферов перестанет ограничивать максимальный поток, возникнет другое узкое место: пропускная способность подсети. Если максимальная скорость обмена кадрами между соседними маршрутизаторами будет x кадров в секунду и между двумя хостами имеется k непересекающихся путей, то, сколько бы ни было буферов у обоих хостов, они не смогут пересылать друг другу больше чем kx TPDU-модулей в секунду. И если отправитель будет передавать с большей скоростью, то подсеть окажется перегружена.

Требуется механизм, основанный не столько на емкости буферов получателя, сколько на пропускной способности подсети. Очевидно, управление потоком должен проводить отправитель, в противном случае у него будет слишком много неподтвержденных TPDU-модулей. В 1975 году Белснес (Belsnes) предложил использовать схему управления потоком скользящего окна, в которой отправитель динамически приводит размер окна в соответствие с пропускной способностью сети. Если сеть может обработать c TPDU-модулей в секунду, а время цикла (включая передачу, распространение, ожидание в очередях, обработку получателем и возврат подтверждения) равно r, тогда размер окна отправителя должен быть равен cr. При таком размере окна отправитель работает, максимально используя канал. Любое уменьшение производительности сети приведет к его блокировке.

Для периодической настройки размера окна отправитель может отслеживать оба параметра и вычислять требуемое значение. Пропускная способность может быть определена простым подсчетом числа TPDU-модулей, получивших подтверждения за определенный период времени, и делением этого числа на тот же период времени. Во время измерения отправитель должен посылать данные с максимальной скоростью, чтобы удостовериться, что ограничивающим фактором является пропускная способность сети, а не низкая скорость передачи. Время, необходимое для получения подтверждения, может быть замерено аналогично. Так как пропускная способность сети зависит от количества трафика в ней, размер окна должен настраиваться довольно часто, чтобы можно было отслеживать

изменения пропускной способности. Как будет показано далее, в Интернете используется похожая схема.

Мультиплексирование

Объединение нескольких разговоров в одном соединении, виртуальном канале и по одной физической линии играет важную роль в нескольких уровнях сетевой архитектуры. Потребность в подобном уплотнении возникает в ряде случаев и на транспортном уровне. Например, если у хоста имеется только один сетевой адрес, он используется всеми соединениями транспортного уровня. Нужен какой-то способ, с помощью которого можно было бы различать, какому процессу следует передать входящий TPDU-модуль. Такая ситуация, называемая **восходящим мультиплексированием**, показана на рис. 6.14, *а*. На рисунке четыре различных соединения транспортного уровня используют одно сетевое соединение (например, один IP-адрес) с удаленным хостом.

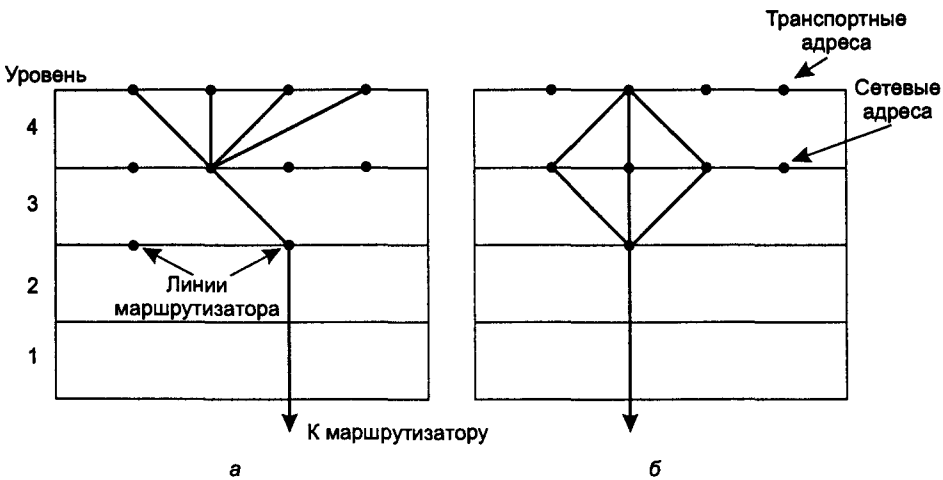


Рис. 6.14. Восходящее мультиплексирование (*а*); нисходящее мультиплексирование (*б*)

Уплотнение может играть важную роль на транспортном уровне и по другой причине. Предположим, например, что подсеть построена на основе виртуальных каналов и на каждом из них данные передаются с максимальной скоростью. Если пользователю требуется большая пропускная способность, нежели может предоставить один виртуальный канал, то можно попробовать решить эту проблему путем открытия нескольких сетевых соединений и распределения трафика между ними, используя виртуальные каналы поочередно, как показано на рис. 6.14, *б*. Такой метод называется **нисходящим мультиплексированием**. При k открытых сетевых соединениях эффективная пропускная способность увеличивается в k раз. В качестве примера можно привести нисходящее мультиплексирование, осуществляемое при работе частных пользователей, имеющих доступ к каналам ISDN. Такая линия обеспечивает установку двух отдельных соедине-

ний по 64 Кбит/с. Использование обоих соединений для доступа в Интернет и разделение трафика позволяют достигать эффективной пропускной способности 128 Кбит/с.

Восстановление после сбоев

Поскольку хосты и маршрутизаторы подвержены сбоям, следует рассмотреть вопрос восстановления после сбоев. Если транспортная сущность целиком помещается в хостах, восстановление после отказов сети и маршрутизаторов не вызывает затруднений. Если сетевой уровень поставляет дейтаграммные услуги, транспортные сущности постоянно ожидают потери TPDU-модулей и знают, как с этим бороться. Если сетевой уровень предоставляет услуги, ориентированные на соединение, тогда потеря виртуального канала обрабатывается при помощи установки нового виртуального канала с последующим запросом у удаленной транспортной сущности ее текущего состояния. При этом выясняется, какие TPDU-модули были получены, а какие нет. Потерянные TPDU-модули передаются повторно.

Более серьезную проблему представляет восстановление после сбоя хоста. В частности, для клиентов может быть желательной возможность продолжать работу после отказа и быстрой перезагрузки сервера. Чтобы пояснить, в чем тут сложность, предположим, что один хост — клиент — посылает длинный файл другому хосту — файловому серверу — при помощи простого протокола с ожиданием. Транспортный уровень сервера просто передает приходящие TPDU-модули один за другим пользователю транспортного уровня. Получив половину файла, сервер сбрасывается и перезагружается, после чего все его таблицы переинициализируются, поэтому он уже не помнит, что с ним было раньше.

Пытаясь восстановить предыдущее состояние, сервер может разослать широковещательный TPDU-модуль всем хостам, объявляя им, что он только что перезагрузился, и прося своих клиентов сообщить ему о состоянии всех открытых соединений. Каждый клиент может находиться в одном из двух состояний: один неподтвержденный TPDU-модуль (состояние *S1*) или ни одного неподтвержденного TPDU-модуля (состояние *S0*). Этой информации клиенту должно быть достаточно, чтобы решить, передавать ему повторно последний TPDU-модуль или нет.

На первый взгляд, здесь все очевидно: узнав о перезапуске сервера, клиент должен передать повторно последний неподтвержденный TPDU-модуль. То есть повторная передача требуется, если клиент находится в состоянии *S1*. Однако при более детальном рассмотрении оказывается, что все не так просто, как мы наивно предположили. Рассмотрим, например, ситуацию, в которой транспортная сущность сервера сначала посылает подтверждение, а уже затем передает пакет прикладному процессу. Запись TPDU-модуля в выходной поток и отправка подтверждения являются двумя различными неделимыми событиями, которые не могут быть выполнены одновременно. Если сбой произойдет после отправки подтверждения, но до того как выполнена запись, клиент получит подтверждение, а при получении объявления о перезапуске сервера окажется в состоянии *S0*.

Таким образом, клиент не станет передавать TPDU-модуль повторно, так как будет считать, что TPDU-модуль уже получен, что в конечном итоге приведет к отсутствию TPDU-модуля.

В этом месте вы, должно быть, подумаете: «А что если поменять местами последовательность действий, выполняемых транспортной сущностью сервера, чтобы сначала осуществлялась запись, а потом высылалось подтверждение?» Представим, что запись сделана, но сбой произошел до отправки подтверждения. В этом случае клиент окажется в состоянии *S1* и поэтому передаст TPDU-модуль повторно, и мы получим дубликат TPDU-модуля в выходном потоке.

Таким образом, независимо от того, как запрограммированы клиент и сервер, всегда могут быть ситуации, в которых протокол не сможет правильно восстановиться. Сервер можно запрограммировать двумя способами: так, чтобы он сначала передавал подтверждение, или так, чтобы сначала записывал TPDU-модуль. Клиент может быть запрограммирован одним из четырех способов: всегда передавать повторно последний TPDU-модуль, никогда не передавать повторно последний TPDU-модуль, передавать повторно TPDU-модуль только в состоянии *S0* и передавать повторно TPDU-модуль только в состоянии *S1*. Таким образом, получаем восемь комбинаций, но, как будет показано, для каждой комбинации имеется набор событий, заставляющий протокол ошибиться.

На сервере могут происходить три события: отправка подтверждения (*A*), запись TPDU-модуля в выходной процесс (*W*) и сбой (*C*). Они могут произойти в виде шести возможных последовательностей: *AC*(*W*), *AWC*, *C*(*AW*), *C*(*WA*), *WAC* и *WC*(*A*), где скобки означают, что после события *C* событие *A* или *B* может и не произойти (то есть уж сломался — так сломался). На рис. 6.15 показаны все восемь комбинаций стратегий сервера и клиента, каждая со своими последовательностями событий. Обратите внимание на то, что для каждой комбинации существует последовательность событий, приводящая к ошибке протокола. Например, если клиент всегда передает повторно неподтвержденный TPDU-модуль, событие *AWC* приведет к появлению неопознанного дубликата, хотя при двух других последовательностях событий протокол будет работать правильно.

Усложнение протокола не помогает. Даже если клиент и сервер обменяются несколькими TPDU-модулями, прежде чем сервер попытается записать полученный пакет, так что клиент будет точно знать, что происходит на сервере, у него нет возможности определить, когда произошел сбой на сервере: до или после записи. Отсюда следует неизбежный вывод: невозможно сделать отказ и восстановление хоста прозрачными для более высоких уровней.

В более общем виде это может быть сформулировано следующим образом: восстановление от сбоя уровня *N* может быть осуществлено только уровнем $N + 1$ и только при условии, что на более высоком уровне сохраняется достаточное количество информации о состоянии процесса. Как упоминалось ранее, транспортный уровень может обеспечить восстановление от сбоя на сетевом уровне, если каждая сторона соединения отслеживает свое текущее состояние.

Эта проблема подводит нас к вопросу о значении так называемого сквозного подтверждения. В принципе, транспортный протокол является сквозным, а не цепным, как более низкие уровни. Теперь рассмотрим случай обращения пользо-

вателя к удаленной базе данных. Предположим, что удаленная транспортная сущность запрограммирована сначала передавать TPDU-модуль вышестоящему уровню, а затем отправлять подтверждение. Даже в этом случае получение подтверждения машиной пользователя не означает, что удаленный хост успел обновить базу данных. Настоящее сквозное подтверждение, получение которого означает, что работа была сделана, и, соответственно, отсутствие которого означает обратное, вероятно, невозможно. Более подробно этот вопрос обсуждается в (Saltzer и др., 1984)

Стратегия, используемая получающим хостом

Стратегия, используемая передающим хостом	Сначала подтверждение, потом запись			Сначала запись, потом подтверждение		
	AC(W)	*AWC*	*C(AW)*	*C(WA)*	*WAC*	*WC(A)*
Всегда повторять передачу	OK	DUP	OK	OK	DUP	DUP
Никогда не повторять передачу	LOST	OK	LOST	LOST	OK	OK
Повторять передачу в *S0*	OK	DUP	LOST	LOST	DUP	OK
Повторять передачу в *S1*	LOST	OK	OK	OK	OK	DUP

OK = Протокол работает корректно
DUP = Протокол формирует дубликат сообщения
LOST = Протокол теряет сообщение

Рис. 6.15. Различные комбинации стратегий сервера и клиента

Простой транспортный протокол

Чтобы конкретизировать обсуждавшиеся ранее идеи, в данном разделе мы подробно изучим пример реализации транспортного уровня. В качестве абстрактных служебных примитивов будут использоваться ориентированные на соединение примитивы из табл. 6.1. Такие примитивы были выбраны, чтобы сделать пример похожим (с некоторыми упрощениями) на популярный протокол TCP.

Служебные примитивы примера транспортного протокола

Первая наша задача будет состоять в том, чтобы как можно более конкретно представить примитивы. С примитивом `CONNECT` (соединить) все довольно просто: у нас просто будет библиотечная процедура `connect`, которую можно вызывать с соответствующими параметрами для установки соединения. Параметрами этой процедуры являются локальный и удаленный TSAP-адреса. Программа, обра-

щающаяся к этой процедуре, блокируется (то есть приостанавливается) на время, пока транспортная сущность пытается установить соединение. Если установка соединения проходит успешно, программа разблокируется и может начинать передавать данные.

Когда процесс желает принимать входящие звонки, он обращается к процедуре `listen` (ожидать), указывая TSAP-адрес, соединение с которым ожидается. При этом процесс блокируется, пока какой-либо удаленный процесс не попытается установить соединение с его TSAP-адресом.

Обратите внимание: такая модель обладает сильной асимметрией. Пассивная сторона выполняет процедуру `listen` и ждет какого-либо события. Активная сторона инициирует соединение. Возникает интересный вопрос: что делать, если активная сторона начнет первой? Первая стратегия такова: при отсутствии ожидания на пассивной стороне попытка соединения считается неудачной. Другая стратегия заключается в блокировании инициатора (возможно, навсегда), пока на противоположном конце устанавливаемого соединения процесс не перейдет в режим ожидания.

Компромиссное решение, используемое в нашем примере, состоит в том, что на установку соединения процедуре `connect` отводится определенный интервал времени. Если процесс хоста, с которым пытаются установить связь, вызовет процедуру `listen` прежде, чем истечет интервал ожидания, соединение будет установлено. В противном случае звонящий получает отказ, разблокируется и получает сообщение об ошибке.

Для разрыва соединения мы будем применять процедуру `disconnect`. Соединение будет считаться разорванным, когда обе стороны вызовут эту процедуру. Другими словами, мы используем симметричную модель разъединения.

При передаче данных появляется та же проблема, что и при установлении соединения: передатчик активен, а получатель пассивен. Мы будем использовать при передаче данных то же решение, что и при установке соединения: активную процедуру `send`, передающую данные, и пассивную процедуру `receive`, блокирующую процесс до тех пор, пока не прибудет TPDU-модуль.

Таким образом, наша услуга определяется пятью примитивами: `CONNECT`, `LISTEN`, `DISCONNECT`, `SEND` и `RECEIVE`. Каждому примитиву соответствует библиотечная служба, выполняющая примитив. Параметры для служебных примитивов и библиотечных процедур следующие:

```
connum = LISTEN (local)
connum = CONNECT (local, remote)
status = SEND (connum, buffer, bytes)
status = RECEIVE (connum, buffer, bytes)
status = DISCONNECT (connum)
```

Примитив `LISTEN` объявляет о желании обращающейся к нему стороны принимать запросы соединения, обращенные к указанному TSAP-адресу. Пользователь примитива блокируется до тех пор, пока кто-либо не попытается с ним связаться. Понятия тайм-аута здесь нет.

Примитив `CONNECT` имеет на входе два параметра: локальный TSAP-адрес `local` и удаленный TSAP-адрес `remote`. Он пытается установить транспортное соедине-

ние между ними. Если это удается, он возвращает в качестве выходного параметра connum неотрицательное число, используемое для идентификации соединения при следующих вызовах процедур. Если же установить соединение не удалось, то причина неудачи помещается в connum в виде отрицательного числа. В нашей простой модели каждый TSAP-адрес может участвовать только в одном транспортном соединении, поэтому возможной причиной отказа может быть занятость одного из транспортных адресов. Среди других причин могут быть следующие: удаленный хост выключен, неверен локальный адрес или неверен удаленный адрес.

Примитив SEND передает содержимое буфера в виде сообщения по указанному транспортному соединению — может быть, в несколько приемов, если сообщение слишком велико. Возможные ошибки, возвращаемые в виде значения переменной status, таковы: нет соединения, неверный адрес буфера или отрицательное число байт.

Примитив RECEIVE означает готовность вызывающего его процесса принимать данные. Размер полученного сообщения помещается в переменную bytes. Если удаленный процесс разорвал соединение или адрес буфера указан неверно (например, за пределами программы пользователя), переменной status присваивается значение кода ошибки, указывающего на причину возникновения проблемы.

Примитив DISCONNECT разрывает транспортное соединение. Параметр connum сообщает номер соединения, которое следует разорвать. Могут возникать, например, такие ошибки: connum принадлежит другому процессу или connum является неверным идентификатором соединения. Переменной status присваивается 0 в случае успеха, в противном случае — код ошибки.

Транспортная сущность примера транспортного протокола

Прежде чем перейти к рассмотрению программы моделирования транспортной сущности, обратите внимание на то, что этот пример аналогичен примерам, приведенным в главе 3: они приводятся скорее в педагогических целях, нежели как серьезное предложение. Многие технические детали (как, например, исчерпывающая обработка ошибок), необходимые для действительно рабочей системы, ради простоты были здесь опущены.

Транспортный уровень использует примитивы сетевой службы для отправки и получения TPDU-модулей. Нам нужно будет выбрать примитивы сетевой службы, чтобы использовать их в этом примере. Одним из вариантов могла бы быть ненадежная дейтаграммная служба. Чтобы сохранить простоту примера, мы не стали останавливать свой выбор на этом варианте, так как в этом случае транспортная программа была бы большой и сложной и занималась бы в основном потерянными и опаздывающими пакетами. Кроме того, большая часть этих идей уже достаточно подробно обсуждалась в главе 3.

Вместо этого мы решили использовать ориентированную на соединение надежную сетевую службу. При этом мы сможем уделить максимум внимания

транспортным вопросам, не встречавшимся на более низких уровнях. Среди прочих, к ним относятся установка соединения, разрыв соединения и управление кредитованием. Подобным образом могла бы выглядеть простая транспортная служба, построенная на базе сети ATM.

В общем случае транспортная сущность может быть либо частью операционной системы, либо набором библиотечных процедур, работающих в адресном пространстве пользователя. В целях упрощения нашего примера мы будем полагать, что здесь используются библиотечные процедуры, однако с помощью небольших изменений можно добиться того, чтобы транспортная сущность стала частью операционной системы (эти изменения связаны в основном со способами доступа к буферам пользователя).

Следует отметить, тем не менее, что в этом примере «транспортная сущность» в действительности вообще не является отдельной сущностью, а представляет собой часть пользовательского процесса. В частности, когда пользователь выполняет некий блокирующий примитив, например LISTEN, вся транспортная сущность также блокируется. Такое решение является вполне удовлетворительным для однозадачного хоста, но на хосте с несколькими пользовательскими процессами более естественным было бы иметь транспортную сущность в виде отдельного процесса, отличного от всех пользовательских процессов.

Интерфейс с сетевым уровнем реализуется с помощью процедур to_net и from_net (не показаны). У каждой из них имеется по шесть параметров. Первый параметр означает идентификатор соединения, один в один соответствующий сетевому виртуальному каналу. Следом идут биты *Q* и *M*, которые, будучи установленными в 1, означают, соответственно, управляющее сообщение и то, что сообщение будет продолжено в следующем пакете. Следом за ними идет тип пакета, выбираемый из шести возможных, приведенных в табл. 6.3. Последние два параметра — это указатель на данные и целое число, указывающее на количество байтов данных.

Таблица 6.3. Пакеты сетевого уровня, используемые в примере

Сетевой пакет	Значение
CALL REQUEST	Запрос соединения. Посылается для установки соединения
CALL ACCEPTED	Вызов принят. Ответ на CALL REQUEST
CLEAR REQUEST	Запрос разъединения. Посылается для разрыва соединения
CLEAR CONFIRMATION	Подтверждение разъединения. Ответ на CLEAR REQUEST
DATA	Данные
CREDIT	Кредит. Служебный пакет для управления окном

При обращении к процедуре to_net транспортная сущность заполняет все значения параметров и передает их сетевому уровню. При вызове процедуры from_net сетевой уровень передает входящий пакет транспортной сущности. Благодаря передаче информации сетевому уровню в виде параметров процедуры вместо передачи самого входящего или исходящего пакета транспортная сущность оказывается защищенной от деталей протокола сетевого уровня. Если транспортная

сущность попытается послать пакет, когда в скользящем окне более низкого уровня нет свободных буферов, она приостановится в процедуре to_net до тех пор, пока в окне не появится место. Для транспортной сущности этот механизм прозрачен и управляется сетевым уровнем с помощью команд типа enable_transport_layer и disable_transport_layer, аналогичных описанным в протоколах главы 3. Управление окном также осуществляется сетевым уровнем.

Помимо этого прозрачного механизма приостановки есть также процедуры sleep и wakeup (не показаны), вызываемые транспортной сущностью. Процедура sleep вызывается, когда транспортная сущность логически блокирована ожиданием внешнего события, обычно прибытия пакета. После вызова процедуры sleep транспортная сущность (и пользовательский процесс, конечно) останавливаются.

Программа транспортной сущности показана в листинге 6.2. Каждое соединение может находиться в одном из следующих семи состояний:

1. IDLE — соединение еще не установлено.
2. WAITING — примитив CONNECT выполнен, пакет CALL REQUEST послан.
3. QUEUED — пакет CALL REQUEST прибыл. Примитив LISTEN еще не вызывался.
4. ESTABLISHED — соединение установлено.
5. SENDING — пользователь ожидает разрешения отправить пакет.
6. RECEIVING — примитив RECEIVE выполнен.
7. DISCONNECTING — примитив DISCONNECT выполнен локально.

Между состояниями могут происходить переходы при одном из следующих событий: выполняется примитив, прибывает пакет или истекает время ожидания.

Листинг 6.2. Пример транспортной сущности

```
#define MAX_CONN 32                     /* максимальное число одновременных
соединений */
#define MAX_MSG_SIZE 8192               /* максимальный размер сообщения в байтах */
#define MAX_PKT_SIZE 512                /* максимальный размер пакета в байтах */
#define TIMEOUT 20
#define CRED 1
#define OK 0

#define ERR_FULL -1
#define ERR_REJECT -2
#define ERR_CLOSED -3
#define LOW_ERR -3

typedef int transport_address;
typedef enum {CALL_REQ,CALL_ACC,CLEAR_REQ,CLEAR_CONF,DATA_PKT,CREDIT} pkt_type;
typedef enum {IDLE,WAITING,QUEUED,ESTABLISHED,SENDING,RECEIVING,DISCONN} cstate;
/* глобальные переменные */
transport_address listen_address;       /* локальный прослушиваемый адрес */
int listen_conn;                        /* идентификатор прослушиваемого соединения
*/
unsigned char data[MAX_PKT_SIZE];       /* временная область для пакетных данных */
```

```
struct conn {
  transport_address local_address, remote_address;
  cstate state;                          /* состояние этого соединения */
  unsigned char *user_buf_addr;          /* указатель на приемный буфер */
  int byte_count;                        /* счетчик передачи/приема */
  int clr_req_received;                  /* устанавливается при получении пакета
CLEAR_REQ */
  int timer;                             /* используется для ожидания подтверждений
для пакетов CALL_REQ */
  int credits;                           /* число сообщений, которые разрешается
послать */
} conn[MAX_CONN+1];                      /* 0-й интервал не используется */

void sleep(void);                        /* прототипы */
void wakeup(void);
void to_net(int cid, int q, int m, pkt_type pt, unsigned char *p, int bytes);
void from_net(int *cid, int *q, int *m, pkt_type *pt, unsigned char *p, int *bytes);

int listen(transport_address t)
{/* Пользователь ждет соединения. Посмотреть, не прибыл ли уже пакет CALL_REQ. */
  int i = 1, found = 0;

  for (i = 1; i <= MAX_CONN; i++)        /* поиск CALL_REQ в таблице */
      if (conn[i].state == QUEUED && conn[i].local_address == t) {
          found = i;
          break;
      }

  if (found == 0) {
      /* Нет пакетов CALL_REQ, ждущих подтверждения. Можно спать, пока не прибудет
пакет или не сработает таймер. */
      listen_address = t; sleep(); i = listen_conn ;
  }
  conn[i].state = ESTABLISHED;           /* соединение установлено */
  conn[i].timer = 0;                     /* таймер не используется */
  listen_conn = 0;                       /* 0 считается неверным адресом */
  to_net(i, 0, 0, CALL_ACC, data, 0);    /* велеть сетевому уровню принять сообщение
*/
  return(i);                             /* вернуть идентификатор соединения */
}

int connect(transport_address l, transport_address r)
{/* Пользователь желает установить соединение с удаленным процессом; послать пакет
CALL_REQ. */
  int i;
  struct conn *cptr;

  data[0] = r; data[1] = l;              /* это нужно для пакета CALL_REQ */
  i = MAX_CONN;                          /* поиск в таблице с конца */
  while (conn[i].state != IDLE && i > 1) i = i -1;
  if (conn[i].state == IDLE) {
      /* отметить в таблице, что CALL_REQ послан */
      cptr = &conn[i];
      cptr->local_address = l; cptr->remote_address = r;
```

```
      cptr->state = WAITING; cptr->clr_req_received = 0;
      cptr->credits = 0; cptr->timer = 0;
      to_net(i, 0, 0, CALL_REQ, data, 2);
      sleep();                              /* ждать CALL_ACC или CLEAR_REQ */
      if (cptr->state == ESTABLISHED) return(i);
      if (cptr->clr_req_received) {
          /* другая сторона отказалась от соединения */
          cptr->state = IDLE;               /* назад в состояние ожидания */
          to_net(i, 0, 0, CLEAR_CONF, data, 0);
          return(ERR_REJECT);
      }
  } else return(ERR_FULL);                  /* отказаться от соединения: нет места в
таблице */
}

int send(int cid, unsigned char bufptr[], int bytes)
{/* Пользователь хочет послать сообщение. */
  int i, count, m;
  struct conn *cptr = &conn[cid];
  /* вход в состояние передачи */
  cptr->state = SENDING;
  cptr->byte_count = 0;                     /* количество посланных байтов сообщения */
  if (cptr->clr_req_received == 0 && cptr->credits == 0) sleep();
  if (cptr->clr_req_received == 0) {
      /* кредит имеется; разбить сообщение на пакеты, если нужно */
      do {
          if (bytes - cptr->byte_count > MAX_PKT_SIZE) {/* многопакетное сообщение */
              count = MAX_PKT_SIZE; m = 1; /* остальные пакеты позже */
          } else { /* однопакетное сообщение */
              count = bytes .cptr->byte_count; m = 0;/* последний пакет сообщения */
          }
          for (i = 0; i < count; i++) data[i] = bufptr[cptr->byte_count + i];
          to_net(cid, 0, m, DATA_PKT, data, count);/* послать 1 пакет */
          cptr->byte_count = cptr->byte_count + count;/* увеличить число посланных
байтов */
      } while (cptr->byte_count < bytes); /* цикл, пока не будет послано все сообщение
*/
      cptr->credits--;                      /* на каждое сообщение расходуется 1 кредит
*/
      cptr->state = ESTABLISHED;
      return(OK);
  } else {
      cptr->state = ESTABLISHED;
      return(ERR_CLOSED);                   /* ошибка передачи: другая сторона хочет
разорвать соединение */
  }
}

int receive(int cid, unsigned char bufptr[], int *bytes)
{/* пользователь готов принять сообщение */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received == 0) {
      /* соединение установлено; попытка получения */
```

```
      cptr->state = RECEIVING;
      cptr->user_buf_addr = bufptr;
      cptr->byte_count = 0;
      data[0] = CRED;
      data[1] = 1;
      to_net(cid, 1, 0, CREDIT, data, 2); /* послать кредит */
      sleep(); /* ожидание данных */
      *bytes = cptr->byte_count;
  }
  cptr->state = ESTABLISHED;
  return(cptr->clr_req_received ? ERR_CLOSED : OK);
}

int disconnect(int cid)
{/* пользователь хочет разорвать соединение */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received) {          /* другая сторона инициировала окончание
связи */
      cptr->state = IDLE;                /* теперь соединение разорвано */
      to_net(cid, 0, 0, CLEAR_CONF, data, 0);
  } else {                               /* мы сами инициировали окончание связи */
      cptr->state = DISCONN;             /* соединение не разорвано, пока другая
сторона не согласится */
      to_net(cid, 0, 0, CLEAR_REQ, data, 0);
  }
  return(OK);
}

void packet_arrival(void)
{/* Прибыл пакет. Получить и обработать его. */
  int cid;                               /* соединение, по которому прибыл пакет */
  int count, i, q, m;
  pkt_type ptype;    /* CALL_REQ, CALL_ACC, CLEAR_REQ, CLEAR_CONF, DATA_PKT, CREDIT
  unsigned char data[MAX_PKT_SIZE];      /* часть данных из пришедшего пакета */
  struct conn *cptr;

  from_net(&cid, &q, &m, &ptype, data, &count); /* получить пакет */
  cptr = &conn[cid];
  switch (ptype) {
      case CALL_REQ: /* удаленный пользователь хочет установить соединение */
          cptr->local_address = data[0]; cptr->remote_address = data[1];
          if (cptr->local_address == listen_address) {
              listen_conn = cid; cptr->state = ESTABLISHED; wakeup();
          } else {
              cptr->state = QUEUED; cptr->timer = TIMEOUT;
          }
          cptr->clr_req_received = 0; cptr->credits = 0;
          break;

      case CALL_ACC: /* удаленный пользователь принял наш CALL_REQ */
          cptr->state = ESTABLISHED;
          wakeup();
          break;
```

```
      case CLEAR_REQ: /* удаленный пользователь хочет разорвать соединение или
отвергнуть вызов */
          cptr->clr_req_received = 1;
          if (cptr->state == DISCONN) cptr->state = IDLE;/* clear collision */
          if (cptr->state == WAITING || cptr->state == RECEIVING || cptr->state ==
SENDING) wakeup();
          break;

      case CLEAR_CONF: /* удаленный пользователь согласен разорвать соединение */
          cptr->state = IDLE;
          break;

      case CREDIT: /* удаленный пользователь ожидает данные */
          cptr->credits += data[1];
          if (cptr->state == SENDING) wakeup();
          break;

      case DATA_PKT: /* удаленный пользователь послал данные */
          for (i = 0; i < count; i++) cptr->user_buf_addr[cptr->byte_count + i] =
data[i];
          cptr->byte_count += count;
          if (m == 0 ) wakeup();
  }
}

void clock(void)
{/* часы тикнули, проверить на тайм-ауты стоящие в очереди запросы на соединение */
  int i;
  struct conn *cptr;
  for (i = 1; i <= MAX_CONN; i++) {
      cptr = &conn[i];
      if (cptr->timer > 0) { /* таймер запущен */
          cptr->timer--;
          if (cptr->timer == 0) { /* теперь время истекло */
              cptr->state = IDLE;
              to_net(i, 0, 0, CLEAR_REQ, data, 0);
          }
      }
  }
}
```

В листинге 6.2 приведены процедуры двух типов. Большинство из них вызываются напрямую пользовательскими программами. Однако процедуры packet_arrival и clock отличаются от остальных. Они вызываются внешними событиями — прибытием пакета и срабатыванием таймера соответственно. Таким образом, они являются процедурами обработки прерываний. Мы будем предполагать, что они никогда не вызываются во время работы процедуры транспортной сущности, а вызываются только тогда, когда пользовательский процесс находится в режиме ожидания или управление находится за пределами транспортной сущности. Это их свойство является существенным для корректной работы транспортной сущности.

Наличие бита Q (Qualifier — спецификатор) в заголовке пакета позволяет избежать накладных расходов в заголовке транспортного уровня. Обычные информационные сообщения посылаются в виде пакетов данных с $Q = 0$. Управляющие сообщения транспортного протокола посылаются как информационные пакеты с $Q = 1$. В нашем примере такое сообщение только одно — CREDIT. Эти управляющие сообщения обнаруживаются и обрабатываются принимающей транспортной сущностью.

Основной структурой данных, используемой транспортной сущностью, является массив *conn*. Каждый элемент этого массива предназначается для одного потенциального соединения и содержит информацию о состоянии соединения, включая транспортные адреса обоих его концов, число посланных и полученных сообщений, текущее состояние, указатель на буфер пользователя, количество уже посланных и полученных байтов, бит, указывающий, что от удаленного пользователя получен запрос на разъединение, таймер и счетчик разрешений на передачу сообщений. Не все эти поля используются в нашем простом примере, но для полной реализации транспортной сущности потребовались бы все эти значения и, возможно, даже некоторые дополнительные. Предполагается, что изначально поле состояния соединения всех элементов массива *conn* инициализируется значением *IDLE*.

Когда пользователь обращается к примитиву CONNECT, сетевой уровень получает указание послать удаленной машине пакет CALL REQUEST, а пользователь переводится в режим ожидания. Когда этот пакет прибывает по указанному адресу, транспортная сущность удаленной машины прерывается на выполнение процедуры packet_arrival, проверяющей, ожидает ли локальный пользователь соединения с указанным адресом. Если да, то обратно отправляется пакет CALL ACCEPTED, а удаленный пользователь переводится в активное состояние. В противном случае запрос соединения ставится в очередь на период времени *TIMEOUT*. Если в течение этого интервала времени пользователь вызывает примитив LISTEN, соединение устанавливается, в противном случае время ожидания истекает, а просящий соединения получает отказ в виде пакета CLEAR REQUEST. Этот механизм необходим, чтобы инициатор соединения не оказался заблокированным навсегда, если удаленный процесс не желает устанавливать с ним соединение.

Хотя мы удалили заголовок транспортного протокола, нам, тем не менее, нужен метод, при помощи которого можно было бы отслеживать принадлежность пакетов тому или иному транспортному соединению, так как несколько соединений могут существовать одновременно. Проще всего в качестве номера соединения использовать номер виртуального канала сетевого уровня. Более того, номер виртуального канала может использоваться как индекс массива *conn*. Когда пакет приходит по виртуальному каналу k, он принадлежит транспортному соединению k, состояние которого хранится в *conn*[k]. Для соединений, инициированных на данном хосте, номер соединения выбирается инициирующей соединение транспортной сущностью.

Чтобы избежать необходимости предоставления буферов и управления ими в транспортной сущности, здесь используется механизм управления потоком, от-

личный от традиционного скользящего окна. Суть его в следующем: когда пользователь вызывает примитив RECEIVE, транспортной сущности посылающей машины отправляется специальное **кредитное сообщение**, содержащее разрешение на передачу определенного количества пакетов данных. Это число сохраняется в массиве *conn*. Когда вызывается примитив SEND, транспортная сущность проверяет, получен ли кредит указанным соединением. Если кредит не нулевой, сообщение посылается (при необходимости в нескольких пакетах), а значение кредита уменьшается, в противном случае транспортная сущность переходит в режим ожидания кредитов. Такой механизм гарантирует, что ни одно сообщение не будет послано, если другая сторона не вызвала примитив RECEIVE. В результате, когда сообщение прибывает, для него гарантированно имеется свободный буфер. Эту схему несложно усовершенствовать, позволив получателям предоставлять сразу несколько буферов и запрашивать несколько сообщений.

Необходимо помнить, что программа, приведенная в листинге 6.2, является сильно упрощенной. Настоящая транспортная сущность должна проверять правильность всех предоставляемых пользователем параметров, обеспечивать восстановление от сбоев сетевого уровня, обрабатывать столкновение вызовов и поддерживать более общие транспортные услуги, включающие такие возможности, как прерывания, дейтаграммы и неблокирующие версии примитивов SEND и RECEIVE.

Пример протокола как конечного автомата

Написание транспортной сущности является сложной и кропотливой работой, особенно для протоколов, применяющихся в действительности. Чтобы снизить вероятность ошибки, полезно представлять состояния протокола в виде конечного автомата.

Как мы уже видели, у соединений нашего протокола есть семь состояний. Можно выделить 12 событий, переводящих соединение из одного состояния в другое. Пять из этих событий являются служебными примитивами. Еще шесть соответствуют получению шести типов пакетов. Последнее событие — истечение времени ожидания. На рис. 6.16 в виде матрицы показаны основные действия протокола. Столбцы матрицы представляют собой состояния, а строки — 12 событий.

Каждая ячейка матрицы на рисунке (то есть модели конечного автомата) содержит до трех полей: предикат, действие и новое состояние. Предикат указывает, при каких условиях производилось действие. Например, в левом верхнем углу матрицы, если выполняется примитив LISTEN и нет свободного места в таблице (предикат *P1*), выполнение примитива LISTEN завершается неудачно, и состояние не изменяется. С другой стороны, если пакет CALL REQUEST для ожидаемого транспортного адреса уже прибыл (предикат *P2*), соединение устанавливается незамедлительно. Другая возможность состоит в том, что утверждение *P2* ложно, то есть пакет CALL REQUEST не прибыл. В этом случае соединение остается в состоянии *IDLE*, ожидая пакета CALL REQUEST.

		Состояние						
		Простой	Ожидание	В очереди	Установлено	Отправление	Получение	Разъединение
Примитивы	LISTEN	P1: ~/Простой P2: A1/Установлено $\overline{P2}$: A2/Простой		~/Установлено				
	CONNECT	P1: ~/Простой $\overline{P1}$: A3/Ожидание						
	DISCONNECT				P4: A5/Простой $\overline{P4}$: A6/Разъединение			
	SEND				P5: A7/Установлено $\overline{P5}$: A8/Отправление			
	RECEIVE				A9/Получение			
Входящие пакеты	Запрос соединения	P3: A1/Установлено $\overline{P3}$: A4/В очереди						
	Запрос принят		~/Установлено					
	Запрос разъединения		~/Простой		A10/Установлено	A10/Установлено	A10/Установлено	~/Простой
	Подтверждение разъединения							~/Простой
	Пакет данных						A12/Установлено	
	Кредит				A11/Установлено	A7/Установлено		
Часы	Тайм-аут			~/Простой				

Предикаты (утверждения)
- P1: Таблица соединений полная
- P2: Выполняется запрос соединения
- P3: Выполняется примитив LISTEN
- P4: Выполняется запрос разъединения
- P5: Кредит есть

Действия
- A1: Послать подтверждение соединения
- A2: Ждать запрос соединения
- A3: Послать подтверждение соединения
- A4: Запустить таймер
- A5: Послать подтверждения разъединения
- A6: Послать запрос разъединения
- A7: Послать сообщение
- A8: Ждать кредита
- A9: Послать кредит
- A10: Установить флаг «Запрос разъединения получен»
- A11: Залисать кредит
- A12: Принять сообщение

Рис. 6.16. Пример протокола как конечного автомата. У каждой ячейки матрицы может быть предикат, действие и новое состояние. Тильда означает, что основное действие не предпринималось. Черта над предикатом означает отрицание предиката. Пустые ячейки соответствуют невозможным или неверным событиям

Следует отметить, что выбор используемых состояний обусловлен не только самим протоколом. В нашем примере нет состояния *LISTENING*, которое вполне могло бы следовать после вызова примитива LISTEN. Состояния *LISTENING* нет потому, что оно связано с полем записи соединения, а примитив LISTEN не создает

записи соединения. Почему же? Потому что мы решили использовать в качестве идентификаторов соединений номера виртуальных каналов сетевого уровня, а для примитива LISTEN номер виртуального канала в конечном счете выбирается сетевым уровнем, когда прибывает пакет CALL REQUEST.

Действия с *A1* по *A12* представляют собой значительные действия, такие как отправление пакетов и запуск таймера. В таблице не перечислены менее значимые события, как, например, инициализация полей записи соединения. Если действие включает активизацию спящего процесса, то действие, следующее за активизацией, также учитывается. Например, если прибывает пакет CALL REQUEST и процесс находился в состоянии ожидания этого пакета, то передача пакета CALL ACCEPT, следующая за активизацией, рассматривается как часть реакции на прибытие пакета CALL REQUEST. После выполнения каждого действия соединение может переместиться в новое состояние, как показано на рис. 6.16.

Представление протокола в виде матрицы состояний обладает тремя преимуществами. Во-первых, такая форма значительно упрощает программисту проверку всех комбинаций состояний и событий при принятии решения о том, не требуется ли какое-либо действие со стороны протокола. В рабочих версиях протокола некоторые комбинации использовались бы для обработки ошибок. В матрице, показанной на рисунке, не проведено различия между невозможными и запрещенными состояниями. Например, если соединение находится в состоянии ожидания (waiting), то событие DISCONNECT является невозможным, так как пользователь блокирован и не может выполнять никакие примитивы. С другой стороны, в состоянии передачи (sending) получение пакета данных не ожидается, так как кредит на них не отпускался. Прибытие пакета данных в этом состоянии является ошибкой протокола.

Второе преимущество представления протокола в виде матрицы состоит в его реализации. Можно представить себе двумерный массив, в котором элемент $a[i][j]$ является указателем или индексом процедуры обработки события i в состоянии j. При этом можно написать транспортную сущность в виде короткого цикла ожидания события. Когда событие происходит, берется соответствующее соединение и извлекается его состояние. При известных значениях события и состояния транспортная сущность просто обращается к массиву a и вызывает соответствующую процедуру обработки. Результатом такого подхода будет более регулярное и систематическое строение программы, нежели в нашем примере транспортной сущности.

Третье преимущество использования конечного автомата заключается в способе описания протокола. В некоторых стандартных документах протоколы описываются как конечные автоматы того же вида, что и на рис. 6.17. Создать работающую транспортную сущность по такому описанию значительно легче, чем по многочисленным разрозненным документам.

Основной недостаток подхода на основе конечного автомата состоит в том, что он может оказаться более сложным для понимания, чем приведенный ранее в листинге пример программы. Однако частично эту проблему можно решить, изобразив конечный автомат в виде графа, как это сделано на рис. 6.17.

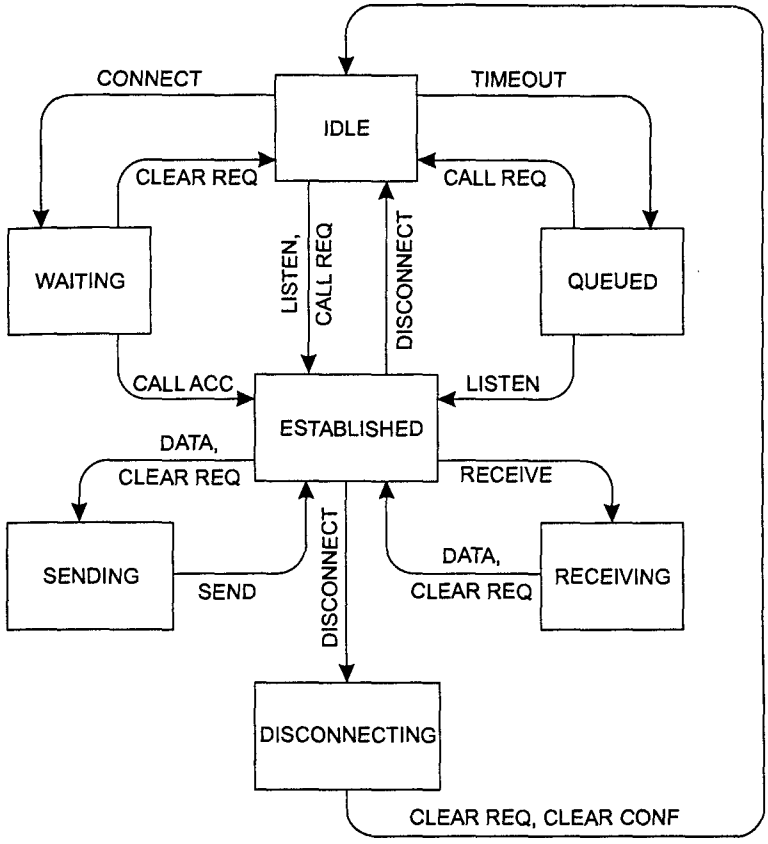


Рис. 6.17. Пример протокола в виде графа. Для простоты переходы, не изменяющие состояние соединения, были опущены

Транспортные протоколы Интернета: UDP

В Интернете нашли применение два основных протокола транспортного уровня, один из которых ориентирован на соединение, другой — нет. В следующих разделах мы изучим их. Протоколом без установления соединения является UDP. Протокол TCP, напротив, ориентирован на соединение. Так как UDP — это, на самом деле, просто IP с добавлением небольшого заголовка, мы изучим сперва его. Рассмотрим также два практических применения UDP.

Основы UDP

Среди набора протоколов Интернета есть транспортный протокол без установления соединения, **UDP** (User Datagram Protocol — пользовательский дейтаграммный протокол). UDP позволяет приложениям отправлять инкапсулированные IP-дейтаграммы без установления соединений. UDP описан в RFC 768.

С помощью протокола UDP передаются сегменты, состоящие из 8-байтного заголовка, за которым следует поле полезной нагрузки. Заголовок показан на рис. 6.18. Два номера портов служат для идентификации конечных точек внутри отправляющей и принимающей машин. Когда прибывает пакет UDP, содержимое его поля полезной нагрузки передается процессу, связанному с портом назначения. Это связывание происходит при выполнении примитива типа BIND. Это было продемонстрировано в листинге 6.1 применительно к TCP (в UDP процесс связывания происходит точно так же). В сущности, весь смысл использования UDP вместо обычного IP заключается как раз в указании портов источника и приемника. Без этих двух полей на транспортном уровне невозможно было бы определить действие, которое следует произвести с пакетом. В соответствии с полями портов производится корректная доставка сегментов.



Рис. 6.18. Заголовок UDP

Информация о порте источника требуется прежде всего при создании ответа, пересылаемого отправителю. Копируя значения поля *Порт источника* из входящего сегмента в поле *Порт назначения* исходящего сегмента, процесс, посылающий ответ, может указать, какому именно процессу на противоположной стороне он предназначается.

Поле *Длина UDP* содержит информацию о длине сегмента, включая заголовок и полезную нагрузку. *Контрольная сумма UDP* не является обязательной. Если она не подсчитывается, ее значение равно 0 (настоящая нулевая контрольная сумма кодируется всеми единицами).

Отключать функцию подсчета контрольной суммы глупо, за исключением одного случая — когда нужна высокая производительность (например, при передаче оцифрованной речи).

Наверное, стоит прямо сказать о том, чего UDP *не* делает. Итак, UDP не занимается контролем потока, контролем ошибок, повторной передачей после приема испорченного сегмента. Все это перекладывается на пользовательские процессы. Что же он делает? UDP предоставляет интерфейс для IP путем демультиплексирования нескольких процессов, использующих порты. Это все, что он делает. Для процессов, которым хочется управлять потоком, контролировать ошибки и временные интервалы, протокол UDP — это как раз то, что доктор прописал.

Одной из областей, где UDP применяется особенно широко, является область клиент-серверных приложений. Зачастую клиент посылает короткий запрос серверу и надеется получить короткий ответ. Если запрос или ответ теряется, клиент по прошествии определенного временного интервала может попытаться еще раз. Это позволяет не только упростить код, но и уменьшить требуемое количе-

ство собщений по сравнению с протоколами, которым требуется начальная настройка.

DNS (Domain Name System — служба имен доменов) — это приложение, которое использует UDP именно так, как описано ранее. Мы изучим его в главе 7. В двух словах, если программе нужно найти IP-адрес по имени хоста, например, www.cs.berkeley.edu, она может послать UDP-пакет с этим именем на сервер DNS. Сервер в ответ на запрос посылает UDP-пакет с IP-адресом хоста. Никакой предварительной настройки не требуется, как не требуется и разрыва соединения после завершения задачи. По сети просто передаются два сообщения.

Вызов удаленной процедуры

В определенном смысле процессы отправки сообщения на удаленный хост и получения ответа очень похожи на вызов функции в языке программирования. В обоих случаях необходимо передать один или несколько параметров, и вы получаете результат. Это соображение навело разработчиков на мысль о том, что можно попробовать организовать запросно-ответное взаимодействие по сети, выполняемое в форме вызовов процедур. Такое решение позволяет упростить и сделать более привычной разработку сетевых приложений. Например, представьте себе процедуру с именем `get_IP_address(имя_хоста)`, работающую посредством отправки UDP-пакетов на сервер DNS, ожидания ответа и отправки повторного запроса в случае наступления тайм-аута (если одна из сторон работает недостаточно быстро). Таким образом, все детали, связанные с особенностями сетевых технологий, скрыты от программиста.

Ключевая работа в этой области написана в 1984 году Бирреллом (Birrell) и Нельсоном (Nelson). По сути дела, было предложено разрешить программам вызывать процедуры, расположенные на удаленных хостах. Когда процесс на машине 1 вызывает процедуру, находящуюся на машине 2, вызывающий процесс машины 1 блокируется и выполняется вызванная процедура на машине 2. Информация от вызывающего процесса может передаваться в виде параметров и приходить обратно в виде результата процедуры. Передача сообщений по сети скрыта от программиста. Такая технология известна под названием **RPC** (Remote Procedure Call — удаленный вызов процедуры) и стала основой многих сетевых приложений. Традиционно вызывающая процедура считается клиентом, а вызываемая — сервером. Мы и здесь будем называть их так же.

Идея RPC состоит в том, чтобы сделать вызов удаленной процедуры максимально похожим на локальный вызов. В простейшем случае для вызова удаленной процедуры клиентская программа должна быть связана с маленькой библиотечной процедурой, называемой **клиентской заглушкой**, которая отображает серверную процедуру в пространство адресов клиента. Аналогично сервер должен быть связан с процедурой, называемой **серверной заглушкой**. Эти процедуры скрывают тот факт, что вызов клиентом серверной процедуры осуществляется не локально.

Реальные шаги, выполняемые при удаленном вызове процедуры, показаны на рис. 6.19. Шаг 1 заключается в вызове клиентом клиентской заглушки. Это ло-

кальный вызов процедуры, параметры которой самым обычным образом помещаются в стек. Шаг 2 состоит в упаковке параметров клиентской заглушки в сообщение и в осуществлении системного вызова для отправки этого сообщения. Упаковка параметров называется **маршалингом**. На шаге 3 ядро системы передает сообщение с клиентской машины на сервер. Шаг 4 заключается в том, что ядро передает входящий пакет серверной заглушке. Последняя на пятом шаге вызывает серверную процедуру с демаршализованными параметрами. При ответе выполняются те же самые шаги, но передача происходит в обратном направлении.

Важнее всего здесь то, что клиентская процедура, написанная пользователем, выполняет обычный (то есть локальный) вызов клиентской заглушки, имеющей то же имя, что и серверная процедура. Поскольку клиентская процедура и клиентская заглушка существуют в одном и том же адресном пространстве, параметры передаются обычным образом. Аналогично серверная процедура вызывается процедурой, находящейся в том же адресном пространстве, с ожидаемыми параметрами. С точки зрения серверной процедуры, не происходит ничего необычного. Таким образом, вместо ввода-вывода с помощью сокетов сетевая коммуникация осуществляется обычным вызовом процедуры.

Несмотря на элегантность концепции RPC, в ней есть определенные подводные камни. Речь идет прежде всего об использовании указателей в качестве параметров. В обычной ситуации передача указателя процедуре не представляет никаких сложностей. Вызываемая процедура может использовать указатель так же, как и вызывающая, поскольку они обе существуют в одном и том же виртуальном адресном пространстве. При удаленном вызове процедуры передача указателей невозможна, потому что адресные пространства клиента и сервера различаются.

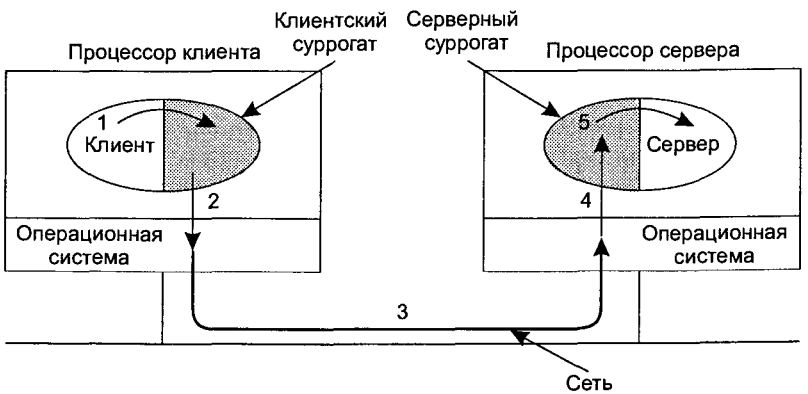


Рис. 6.19. Этапы выполнения удаленного вызова процедуры. Заглушки затенены

Иногда с помощью некоторых уловок все же удается передавать указатели. Допустим, первым параметром является указатель на целое число k. Клиентская заглушка может выполнить маршализацию k и передать его серверу. Серверная

заглушка создаст указатель на полученную переменную *k* и передаст его серверной процедуре. Именно этого та и ожидала. Когда серверная процедура возвращает управление серверной заглушке, последняя отправляет *k* обратно клиенту, где обновленное значение этой переменной записывается вместо старого (если оно было изменено сервером). В принципе, стандартная последовательность действий, выполняемая при вызове по ссылке, заменилась последовательностью копирования и восстановления. Увы, этот трюк не всегда удается применить — в частности, нельзя это сделать, если указатель ссылается на граф или иную сложную структуру данных. По этой причине на параметры удаленно вызываемых процедур должны быть наложены определенные ограничения.

Вторая проблема заключается в том, что в языках со слабой типизацией данных (например, в C) можно совершенно законно написать процедуру, которая подсчитывает скалярное произведение двух векторов (массивов), не указывая их размеры. Каждая из этих структур в качестве ограничителя имеет какое-то значение, известное только вызывающей и вызываемой процедурам. При этих обстоятельствах клиентская заглушка не способна маршализовать параметры: нет никакой возможности определить их размеры.

Третья проблема заключается в том, что не всегда можно распознать типы параметров по спецификации или по самому коду. В качестве примера можно привести процедуру `printf`, у которой может быть любое число параметров (не меньше одного), и они могут представлять собой смесь целочисленных, коротких вещественных, длинных вещественных, символьных, строковых, вещественных с плавающей запятой различной длины и других типов. Задача удаленного вызова процедуры `printf` может оказаться практически невыполнимой из-за такой своеобразной толерантности языка C. Тем не менее, нет правила, говорящего, что удаленный вызов процедур возможен только в том случае, если используется не C (C++), — это подорвало бы репутацию метода RPC.

Четвертая проблема связана с применением глобальных переменных. В нормальной ситуации вызывающая и вызываемая процедуры могут общаться друг с другом посредством глобальных переменных (кроме общения с помощью параметров). Если вызываемая процедура переедет на удаленную машину, программа, использующая глобальные переменные, не сможет работать, потому что глобальные переменные больше не смогут служить в качестве разделяемого ресурса.

Эти проблемы не означают, что метод удаленного вызова процедур безнадежен. На самом деле, он широко используется, просто нужны некоторые ограничения для его нормальной практической работы.

Конечно, RPC не нуждается в UDP-пакетах, однако они хорошо подходят друг для друга, и UDP обычно используется совместно с RPC. Тем не менее, когда параметры или результаты оказываются больше максимальных размеров UDP-пакетов или запрашиваемая операция не идемпотентна (то есть не может повторяться без риска сбоя — например, операция инкрементирования счетчика), может понадобиться установка TCP-соединения и отправки запроса по TCP, а не по UDP.

Транспортный протокол реального масштаба времени

Клиент-серверный удаленный вызов процедур — это та область, в которой UDP применяется очень широко. Еще одной такой областью являются мультимедийные приложения реального времени. В частности, с ростом популярности интернет-радио, интернет-телефонии, музыки и видео по заказу, видеоконференций и других мультимедийных приложений стало понятно, что все они пытаются заново изобрести велосипед, используя более или менее одинаковые транспортные протоколы реального времени. Вскоре пришли к мысли о том, что было бы здорово иметь один общий транспортный протокол для мультимедийных приложений. Так появился на свет протокол **RTP** (Real-Time Transport Protocol — транспортный протокол реального масштаба времени). Он описан в RFC 1889 и ныне широко используется.

RTP занимает довольно странное положение в стеке протоколов. Было решено, что RTP будет принадлежать пользовательскому пространству и работать (в обычной ситуации) поверх UDP. Делается это так. Мультимедийное приложение может состоять из нескольких аудио-, видео-, текстовых и некоторых других потоков. Они прописываются в библиотеке RTP, которая, как и само приложение, находится в пользовательском пространстве. Библиотека уплотняет потоки и помещает их в пакеты RTP, которые, в свою очередь, отправляются в сокет. На другом конце сокета (в ядре операционной системы) генерируются UDP-пакеты, которые внедряются в IP-пакеты. Теперь остается передать IP-пакеты по сети. Если компьютер подключен к локальной сети Ethernet, IP-пакеты для передачи разбиваются на кадры Ethernet. Стек протоколов для описанной ситуации показан на рис. 6.20, *а*. Система вложенных пакетов показана на рис. 6.20, *б*.

В результате оказывается непросто определить, к какому уровню относится RTP. Так как протокол работает в пользовательском пространстве и связан с прикладной программой, он, несомненно, выглядит, как прикладной протокол. С другой стороны, он является общим, независимым от приложения протоколом, который просто предоставляет транспортные услуги, не более. С этой точки зрения он может показаться транспортным протоколом. Наверное, лучше всего будет определить его таким образом: RTP — это транспортный протокол, реализованный на прикладном уровне.

Основной функцией RTP является уплотнение нескольких потоков реального масштаба времени в единый поток пакетов UDP. Поток UDP можно направлять либо по одному, либо сразу по нескольким адресам. Поскольку RTP использует обычный UDP, его пакеты не обрабатываются маршрутизаторами каким-либо особым образом, если только не включены свойства доставки с качеством обслуживания уровня IP. В частности, нет никаких гарантий касательно доставки, дрожания (неустойчивой синхронизации) и т. д.

Каждый пакет, посылаемый с потоком RTP, имеет номер, на единицу превышающий номер своего предшественника. Такой способ нумерации позволяет получателю определить пропажу пакетов. Если обнаруживается исчезновение какого-либо пакета, то лучшее, что может сделать получатель, — это путем ин-

терполяции аппроксимировать пропущенное значение. Повторная передача в данном случае не является хорошим решением, поскольку это займет много времени, и повторно переданный пакет окажется уже никому не нужным. Поэтому протокол RTP не осуществляет управление потоком, контроль ошибок, и в стандарте не предусмотрены никакие подтверждения и механизмы запроса повторной передачи.

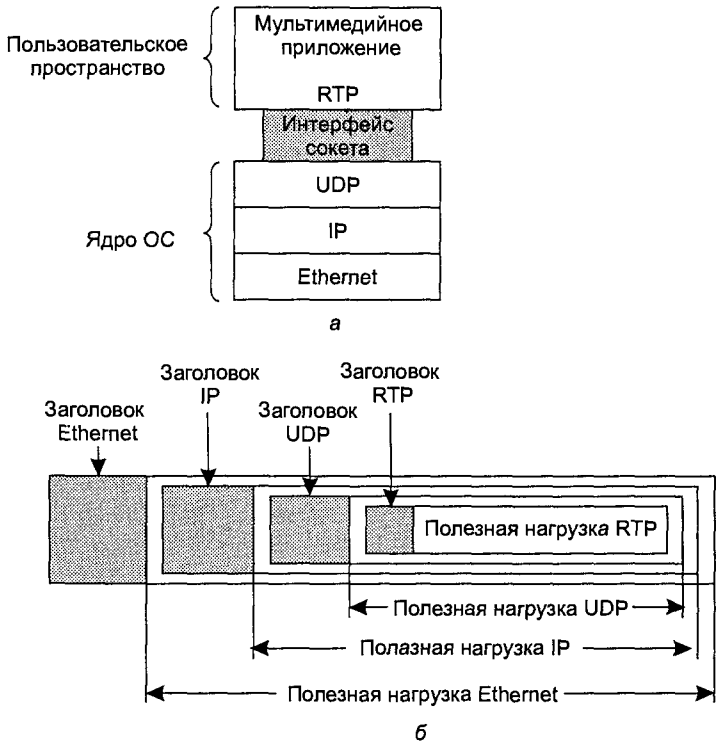


Рис. 6.20. Положение RTP в стеке протоколов (а); вложение пакетов (*б*)

В поле полезной нагрузки RTP может содержаться несколько символов данных, которые могут иметь формат, соответствующий отправившему их приложению. Межсетевое взаимодействие обеспечивается в протоколе RTP за счет определения нескольких профилей (например, отдельных аудиопотоков), каждому из которых может сопоставляться несколько форматов кодирования. Скажем, аудиопоток может кодироваться при помощи PCM (8-битными символами с полосой 8 кГц), дельта-кодирования, кодирования с предсказанием, GSM, MP3 и т. д. В RTP имеется специальное поле заголовка, в котором источник может указать метод кодирования, однако далее источник никак не влияет на процесс кодирования.

Еще одна функция, необходимая приложениям реального времени, — это отметки времени. Идея состоит в том, чтобы позволить источнику связать отметку времени с первым символом каждого пакета. Отметки времени ставятся относи-

тельно момента начала передачи потока, поэтому важны только интервалы *между* отметками. Абсолютные значения, по сути дела, никакой роли не играют. Такой механизм позволяет приемнику буферизировать небольшое количество данных и проигрывать каждый отрезок спустя правильное число миллисекунд после начала потока независимо от того, когда на самом деле приходит пакет, содержащий данный отрезок. За счет этого не только снижается эффект джиттера (флуктуаций, неустойчивости синхронизации), но и появляется возможность синхронизации между собой нескольких потоков. Например, в цифровом телевидении может быть видеопоток и два аудиопотока. Второй аудиопоток обычно нужен либо для обеспечения стереозвучания, либо для дублированной на иностранный язык звуковой дорожки фильма. У каждого потока свой физический источник, однако с помощью временных отметок, генерируемых единым таймером, эти потоки при воспроизведении можно синхронизовать даже в том случае, если они приходят не совсем одновременно.

Заголовок RTP показан на рис. 6.21. Он состоит из трех 32-разрядных слов и некоторых возможных расширений. Первое слово содержит поле *Версия*, которое в настоящий момент уже имеет значение 2. Будем надеяться, что текущая версия окажется окончательной или хотя бы предпоследней, поскольку в идентифицирующем ее двухбитном поле осталось место только для одного нового номера (впрочем, код 3 может обозначать, что настоящий номер версии содержится в поле расширения).

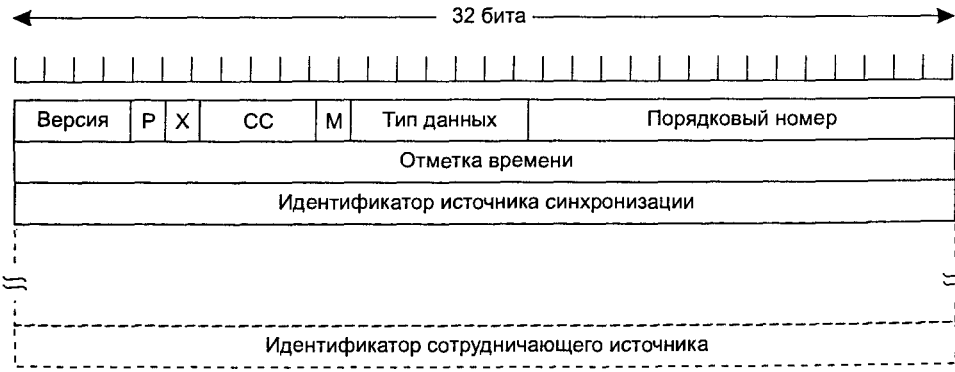


Рис. 6.21. Заголовок RTP

Бит *P* указывает на то, что размер пакета сделан кратным 4 байтам за счет байтов заполнения. При этом в последнем байте заполнения содержится общее число байтов заполнения. Бит *X* говорит о том, что присутствует расширенный заголовок. Формат и назначение расширенного заголовка не определяются. Обязательным для него является только то, что первое слово расширения должно содержать общую длину расширения. Это запасная возможность для разнообразных непредсказуемых будущих требований.

Поле *CC* говорит о том, сколько сотрудничающих источников формируют поток. Их число может колебаться от 0 до 15 (см. далее). Бит *M* — это маркер, свя-

занный с конкретным приложением. Он может использоваться для обозначения начала видеокадра, начала слова в аудиоканале или еще для чего-нибудь, важного и понятного для приложения. Поле *Тип данных* содержит информацию об использующемся алгоритме кодирования (например, несжатое 8-битное аудио, MP3 и т. д.). Поскольку такое поле есть в каждом пакете, метод кодирования может изменяться прямо во время передачи потока. *Порядковый номер* — это просто счетчик, который инкрементируется в каждом пакете RTP. Он используется для определения потерявшихся пакетов.

Отметка времени генерируется источником потока и служит для записи момента создания первого слова пакета. Отметки времени помогают снизить эффект джиттера на приемнике за счет того, что момент воспроизведения делается независимым от времени прибытия пакета. *Идентификатор источника синхронизации* позволяет определить, какому потоку принадлежит пакет. Применяется метод уплотнения и распределения потоков данных, следующих в виде единого потока UDP-пакетов. Наконец, *Идентификаторы сотрудничающих источников*, если таковые имеются, используются, когда конечный поток формируется несколькими источниками. В этом случае микширующее устройство является источником синхронизации, а в полях идентификаторов источников перечисляются смешиваемые потоки.

У протокола RTP есть небольшой родственный протокол под названием **RTCP** (Real-Time Transport Control Protocol — управляющий транспортный протокол реального времени). Он занимается поддержкой обратной связи, синхронизацией, обеспечением пользовательского интерфейса, однако не занимается передачей каких-либо данных. Первая его функция может использоваться для обратной связи по задержкам, джиттеру, пропускной способности, перегрузке и другим свойствам сети, о которых сообщается источникам. Полученная информация может приниматься во внимание кодировщиком для увеличения скорости передачи данных (что приведет к улучшению качества), когда это позволяет делать состояние сети, или уменьшения скорости при возникновении в сети каких-либо проблем. Постоянная обратная связь обеспечивает динамическую настройку алгоритмов кодирования на обеспечение наилучшего качества при текущих обстоятельствах. Например, пропускная способность при передаче потока может как увеличиваться, так и уменьшаться, и в соответствии с этим могут изменяться методы кодирования — скажем, MP3 может заменяться 8-битным PCM или дельта-кодированием. Поле *Тип данных* сообщает приемнику о том, какой алгоритм кодирования применяется для данного пакета, что позволяет изменять их по требованию при передаче потока.

RTCP также обеспечивает межпотоковую синхронизацию. Проблема состоит в том, что разные потоки могут использовать разные таймеры с разной степенью разрешения и разными скоростями дрейфа. RTCP помогает решить эти проблемы и синхронизировать потоки с разными параметрами.

Наконец, RTCP позволяет именовать различные источники (например, с помощью обычного ASCII-текста). Эта информация может отображаться на приемнике, позволяя определить источник текущего потока.

Более подробную информацию о протоколе RTP можно найти в (Perkins, 2002).

Транспортные протоколы Интернета: TCP

UDP является простым протоколом и имеет определенную область применения. В первую очередь, это клиент-серверные взаимодействия и мультимедиа. Тем не менее, большинству интернет-приложений требуется надежная, последовательная передача. UDP не удовлетворяет этим требованиям, поэтому требуется иной протокол. Такой протокол называется TCP, и он является рабочей лошадкой Интернета. Позже мы рассмотрим его детально.

Основы TCP

Протокол **TCP** (Transmission Control Protocol — протокол управления передачей) был специально разработан для обеспечения надежного сквозного байтового потока по ненадежной интерсети. Объединенная сеть отличается от отдельной сети тем, что ее различные участки могут обладать сильно различающейся топологией, пропускной способностью, значениями времени задержки, размерами пакетов и другими параметрами. При разработке TCP основное внимание уделялось способности протокола адаптироваться к свойствам объединенной сети и отказоустойчивости при возникновении различных проблем.

Протокол TCP описан в RFC 793. Со временем были обнаружены различные ошибки и неточности, и по некоторым пунктам требования были изменены. Подробное описание этих уточнений и исправлений дается в RFC 1122. Расширения протокола приведены в RFC 1323.

Каждая машина, поддерживающая протокол TCP, обладает транспортной сущностью TCP, являющейся либо библиотечной процедурой, либо пользовательским процессом, либо частью ядра системы. В любом случае, транспортная сущность управляет TCP-потоками и интерфейсом с IP-уровнем. TCP-сущность принимает от локальных процессов пользовательские потоки данных, разбивает их на куски, не превосходящие 64 Кбайт (на практике это число обычно равно 1460 байтам данных, что позволяет поместить их в один кадр Ethernet с заголовками IP и TCP), и посылает их в виде отдельных IP-дейтаграмм. Когда IP-дейтаграммы с TCP-данными прибывают на машину, они передаются TCP-сущности, которая восстанавливает исходный байтовый поток. Для простоты мы иногда будем употреблять «TCP» для обозначения транспортной сущности TCP (части программного обеспечения) или протокола TCP (набора правил). Из контекста будет понятно, что имеется в виду. Например, в выражении «Пользователь передает данные TCP» подразумевается, естественно, транспортная сущность TCP.

Уровень IP не гарантирует правильной доставки дейтаграмм, поэтому именно TCP приходится следить за истекшими интервалами ожидания и в случае необходимости заниматься повторной передачей пакетов. Бывает, что дейтаграммы прибывают в неправильном порядке. Восстанавливать сообщения из таких дейтаграмм обязан также TCP. Таким образом, протокол TCP призван обеспечить надежность, о которой мечтают многие пользователи и которая не предоставляется протоколом IP.

Модель службы TCP

В основе службы TCP лежат так называемые сокеты (гнезда или конечные точки), создаваемые как отправителем, так и получателем. Они обсуждались в разделе «Сокеты Беркли». У каждого сокета есть номер (адрес), состоящий из IP-адреса хоста и 16-битного номера, локального по отношению к хосту, называемого **портом**. Портом в TCP называют TSAP-адрес. Для обращения к службе TCP между сокетом машины отправителя и сокетом машины получателя должно быть явно установлено соединение. Примитивы сокетов приведены в табл. 6.2

Один сокет может использоваться одновременно для нескольких соединений. Другими словами, два и более соединений могут оканчиваться одним сокетом. Соединения различаются по идентификаторам сокетов на обоих концах — (*socket1*, *socket2*). Номера виртуальных каналов или другие идентификаторы не используются.

Номера портов со значениями ниже 1024, называемые **популярными портами**, зарезервированы стандартными сервисами. Например, любой процесс, желающий установить соединение с хостом для передачи файла с помощью протокола FTP, может связаться с портом 21 хоста-адресата и обратиться, таким образом, к его FTP-демону. Список популярных портов приведен на сайте www.iana.org. Таких портов на данный момент более 300. Некоторые из них включены в табл. 6.4.

Можно было бы, конечно, связать FTP-демона с портом 21 еще во время загрузки, тогда же связать демона telnet с портом 23, и т. д. Однако если бы мы так сделали, мы бы только зря заняли память информацией о демонах, которые, на самом деле, большую часть времени простаивают. Вместо этого обычно пользуются услугами одного демона, называемого в UNIX **inetd**, который связывается с несколькими портами и ожидает первое входящее соединение. Когда оно происходит, inetd создает новый процесс, для которого вызывается подходящий демон, обрабатывающий запрос. Таким образом, постоянно активен только inetd, остальные вызываются только тогда, когда для них есть работа. Inetd имеет специальный конфигурационный файл, из которого он может узнать о назначении портов. Это значит, что системный администратор может настроить систему таким образом, чтобы с самыми загруженными портами (например, 80) были связаны постоянные демоны, а с остальными — inetd.

Таблица 6.4. Некоторые зарезервированные порты

Порт	Протокол	Использование
21	FTP	Передача файлов
23	Telnet	Дистанционный вход в систему
25	SMTP	Электронная почта
69	TFTP	Простейший протокол передачи файлов
79	Finger	Поиск информации о пользователе
80	HTTP	Мировая Паутина
110	POP-3	Удаленный доступ к электронной почте
119	NNTP	Группы новостей

Все TCP-соединения являются полнодуплексными и двухточечными. Полный дуплекс означает, что трафик может следовать одновременно в противоположные стороны. Двухточечное соединение подразумевает, что у него имеются ровно две конечные точки. Широковещание и многоадресная рассылка протоколом TCP не поддерживаются.

TCP-соединение представляет собой байтовый поток, а не поток сообщений. Границы между сообщениями не сохраняются. Например, если отправляющий процесс записывает в TCP-поток четыре 512-байтовых порции данных, эти данные могут быть доставлены получающему процессу в виде четырех 512-байтовых порций, двух 1024-байтовых порций, одной 2048-байтовой порции (см. рис. 6.22) или как-нибудь еще. Нет способа, которым получатель смог бы определить, каким образом записывались данные.

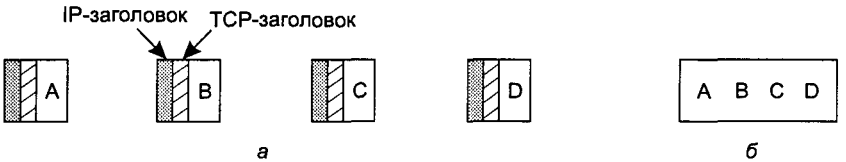


Рис. 6.22. Четыре 512-байтовых сегмента, посланные как отдельные IP-дейтаграммы (а); 2048 байт данных, доставленные приложению с помощью отдельного вызова процедуры READ (*б*)

Файлы в системе UNIX также обладают этим свойством. Программа, читающая файл, не может определить, как был записан этот файл: поблочно, побайтно или сразу целиком. Как и в случае с файлами системы UNIX, TCP-программы не имеют представления о назначении байтов и не интересуются этим. Байт для них — просто байт.

Получив данные от приложения, протокол TCP может послать их сразу или поместить в буфер, чтобы послать сразу большую порцию данных, по своему усмотрению. Однако иногда приложению бывает необходимо, чтобы данные были посланы немедленно. Допустим, например, что пользователь регистрируется на удаленной машине. После того как он ввел команду и нажал клавишу Enter, важно, чтобы введенная им строка была доставлена на удаленную машину сразу же, а не помещалась в буфер, пока не будет введена следующая строка. Чтобы вынудить передачу данных без промедления, приложение может установить флаг PUSH (протолкнуть).

Некоторые старые приложения использовали флаг PUSH как разделитель сообщений. Хотя этот трюк иногда срабатывает, не все реализации протокола TCP передают флаг PUSH принимающему приложению. Кроме того, если прежде чем первый пакет с установленным флагом PUSH будет передан в линию, TCP-сущность получит еще несколько таких пакетов (то есть выходная линия будет занята), TCP-сущность будет иметь право послать все эти данные в виде единой дейтаграммы, не разделяя их на отдельные порции.

Последней особенностью службы TCP, о которой следует упомянуть, являются **срочные данные**. Когда пользователь, взаимодействующий с программой

в интерактивном режиме, нажимает клавишу Delete или Ctrl-C, чтобы прервать начавшийся удаленный процесс, посылающее приложение помещает в выходной поток данных управляющую информацию и передает ее TCP-службе вместе с флагом URGENT (срочно). Этот флаг заставляет TCP-сущность прекратить накопление данных и без промедления передать в сеть все, что у нее есть для данного соединения.

Когда срочные данные прибывают по назначению, получающее приложение прерывается (то есть «получает сигнал», в терминологии UNIX), после чего оно может считать данные из входного потока и найти среди них срочные. Конец срочных данных маркируется, так что приложение может распознать, где они заканчиваются. Начало срочных данных не маркируется. Приложение должно само догадаться. Такая схема представляет собой грубый сигнальный механизм, оставляя все прочее приложению.

Протокол TCP

В данном разделе будет рассмотрен протокол TCP в общих чертах. В следующем разделе мы обсудим заголовок протокола, поле за полем.

Ключевым свойством TCP, определяющим всю структуру протокола, является то, что в TCP-соединении у каждого байта есть свой 32-разрядный порядковый номер. В первые годы существования Интернета базовая скорость передачи данных между маршрутизаторами по выделенным линиям составляла 56 Кбит/с. Хосту, постоянно выдающему данные с максимальной скоростью, потребовалось бы больше недели на то, чтобы порядковые номера совершили полный круг. При нынешних скоростях порядковые номера могут кончиться очень быстро, об этом еще будет сказано позже. Отдельные 32-разрядные порядковые номера используются для подтверждений и для механизма скользящего окна, о чем также будет сказано позже.

Отправляющая и принимающая TCP-сущности обмениваются данными в виде сегментов. **Сегмент** состоит из фиксированного 20-байтового заголовка (плюс необязательная часть), за которой могут следовать байты данных. Размер сегментов определяется программным обеспечением TCP. Оно может объединять в один сегмент данные, полученные в результате нескольких операций записи, или, наоборот, распределять результат одной записи между несколькими сегментами. Размер сегментов ограничен двумя пределами. Во-первых, каждый сегмент, включая TCP-заголовок, должен помещаться в 65 515-байтное поле полезной нагрузки IP-пакета. Во-вторых, в каждой сети есть **максимальная единица передачи (MTU**, Maximum Transfer Unit), и каждый сегмент должен помещаться в MTU. На практике размер максимальной единицы передачи составляет обычно 1500 байт (что соответствует размеру поля полезной нагрузки Ethernet), и таким образом определяется верхний предел размера сегмента.

Основным протоколом, используемым TCP-сущностями, является протокол скользящего окна. При передаче сегмента отправитель включает таймер. Когда сегмент прибывает в пункт назначения, принимающая TCP-сущность посылает обратно сегмент (с данными, если есть что посылать, или без данных) с номером

подтверждения, равным порядковому номеру следующего ожидаемого сегмента. Если время ожидания подтверждения истекает, отправитель посылает сегмент еще раз.

Хотя этот протокол кажется простым, в нем имеется несколько деталей, которые следует рассмотреть подробнее. Сегменты могут приходить в неверном порядке. Так, например, возможна ситуация, в которой байты с 3072-го по 4095-й уже прибыли, но подтверждение для них не может быть выслано, так как байты с 2048-го по 3071-й еще не получены. К тому же сегменты могут задерживаться в сети так долго, что у отправителя истечет время ожидания и он передаст их снова. Переданный повторно сегмент может включать в себя уже другие диапазоны фрагментов, поэтому потребуется очень аккуратное администрирование для определения номеров байтов, которые уже были приняты корректно. Тем не менее, поскольку каждый байт в потоке единственным образом определяется по своему сдвигу, эта задача оказывается реальной.

Протокол TCP должен уметь справляться с этими проблемами и решать их эффективно. На оптимизацию производительности TCP-потоков было потрачено много сил. В следующем разделе мы обсудим несколько алгоритмов, используемых в различных реализациях протокола TCP.

Заголовок TCP-сегмента

На рис. 6.23 показана структура заголовка TCP-сегмента. Каждый сегмент начинается с 20-байтного заголовка фиксированного формата. За ним могут следовать дополнительные поля. После дополнительных полей может располагаться до 65 535 – 20 – 20 = 65 495 байт данных, где первые 20 байт — это IP-заголовок, а вторые — TCP-заголовок. Сегменты могут и не содержать данных. Такие сегменты часто применяются для передачи подтверждений и управляющих сообщений.

Рассмотрим TCP-заголовок поле за полем. Поля *Порт получателя* и *Порт отправителя* являются идентификаторами локальных конечных точек соединения. Популярные номера портов перечислены на www.iana.org, однако, что касается всех остальных портов, то каждый хост может сам решать, как их распределять. Номер порта вместе с IP-адресом хоста образуют уникальный 48-битный идентификатор конечной точки. Пара таких идентификаторов, относящихся к источнику и приемнику, однозначно определяет соединение.

Поля *Порядковый номер* и *Номер подтверждения* выполняют свою обычную функцию. Обратите внимание: поле *Номер подтверждения* относится к следующему ожидаемому байту, а не к последнему полученному. Оба они 32-разрядные, так как в TCP-потоке нумеруется каждый байт данных.

Поле *Длина TCP-заголовка* содержит размер TCP-заголовка, выраженный в 32-разрядных словах. Эта информация необходима, так как поле *Факультативные поля*, а вместе с ним и весь заголовок, может быть переменной длины. По сути, это поле указывает смещение от начала сегмента до поля данных, измеренное в 32-битных словах. Это то же самое, что длина заголовка.

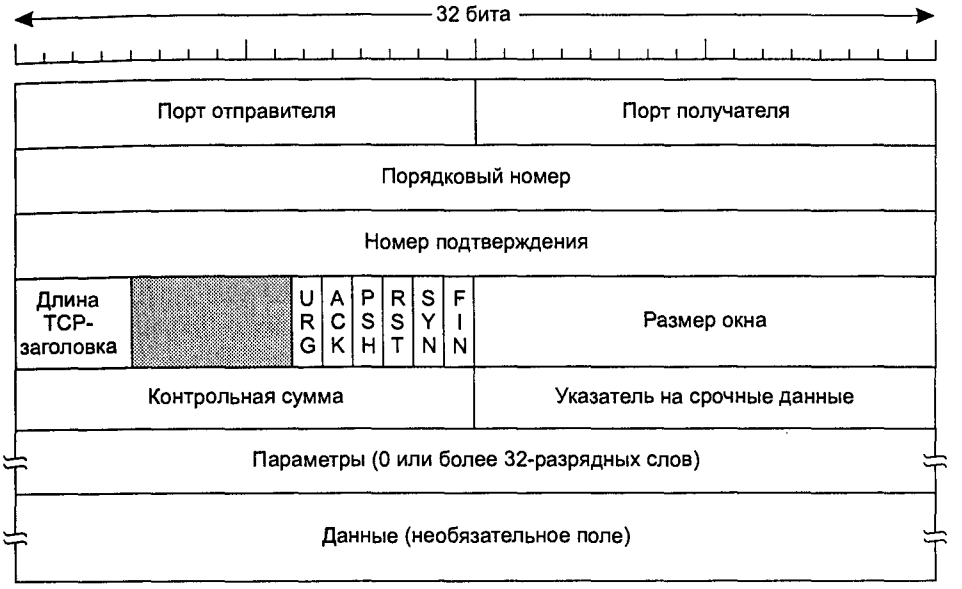


Рис. 6.23. Заголовок TCP

Следом идет неиспользуемое 6-битное поле. Тот факт, что это поле выжило в течение четверти века, является свидетельством того, насколько хорошо продуман дизайн TCP.

Затем следуют шесть 1-битовых флагов. Бит *URG* устанавливается в 1 в случае использования поля *Указатель на срочные данные*, содержащего смещение в байтах от текущего порядкового номера байта до места расположения срочных данных. Таким образом в протоколе TCP реализуются прерывающие сообщения. Как уже упоминалось, протокол TCP лишь обеспечивает доставку сигнала пользователя до получателя, не интересуясь причиной прерывания.

Если бит *ACK* установлен в 1, значит, поле *Номер подтверждения* содержит осмысленные данные. В противном случае данный сегмент не содержит подтверждения, и поле *Номер подтверждения* просто игнорируется.

Бит *PSH* является, по сути, PUSH-флагом, с помощью которого отправитель просит получателя доставить данные приложению сразу по получении пакета, а не хранить их в буфере, пока тот не наполнится. (Получатель может заниматься буферизацией для достижения большей эффективности.)

Бит *RST* используется для сброса состояния соединения, которое из-за сбоя хоста или по другой причине попало в тупиковую ситуацию. Кроме того, он используется для отказа от неверного сегмента или от попытки создать соединение. Если вы получили сегмент с установленным битом *RST*, это означает наличие какой-то проблемы.

Бит *SYN* применяется для установки соединения. У запроса соединения бит *SYN* = 1, а бит *ACK* = 0, что означает, что поле подтверждения не используется. В ответе на этот запрос содержится подтверждение, поэтому значения этих би-

тов в нем равны: $SYN = 1$, $ACK = 1$. Таким образом, бит *SYN* используется для обозначения сегментов CONNECTION REQUEST и CONNECTION ACCEPTED, а бит *ACK* — чтобы отличать их друг от друга.

Бит *FIN* используется для разрыва соединения. Он указывает на то, что у отправителя больше нет данных для передачи. Однако, даже закрыв соединение, процесс может продолжать получать данные в течение неопределенного времени. У сегментов с битами *FIN* и *SYN* есть порядковые номера, что гарантирует правильный порядок их выполнения.

Управление потоком в протоколе TCP осуществляется при помощи скользящего окна переменного размера. Поле *Размер окна* сообщает, сколько байт может быть послано после байта, получившего подтверждение. Значение поля *Размер окна* может быть равно нулю, что означает, что все байты вплоть до *Номер подтверждения*-1 получены, но у получателя в данный момент какие-то проблемы, и остальные байты он пока принять не может. Разрешение на дальнейшую передачу может быть получено путем отправки сегмента с таким же значением поля *Номер подтверждения* и ненулевым значением поля *Размер окна*.

В главе 3 мы обсуждали протоколы, в которых подтверждения приема кадров были связаны с разрешениями на продолжение передачи. Эта связь была следствием жестко закрепленного размера скользящего окна в этих протоколах. В TCP подтверждения отделены от разрешений на передачу данных. В сущности, приемник может сказать: «Я получил байты вплоть до k-го, но я сейчас не хочу продолжать прием данных». Такое разделение (выражающееся в скользящем окне *переменного размера*) придает протоколу дополнительную гибкость. Далее мы обсудим этот аспект более детально.

Поле *Контрольная сумма* служит для повышения надежности. Оно содержит контрольную сумму заголовка, данных и псевдозаголовка, показанного на рис. 6.24. При выполнении вычислений поле *Контрольная сумма* устанавливается равным нулю, а поле данных дополняется нулевым байтом, если его длина представляет собой нечетное число. Алгоритм вычисления контрольной суммы просто складывает все 16-разрядные слова в дополнительном коде, а затем вычисляет дополнение для всей суммы. В результате, когда получатель считает контрольную сумму всего сегмента, включая поле *Контрольная сумма*, результат должен быть равен 0.

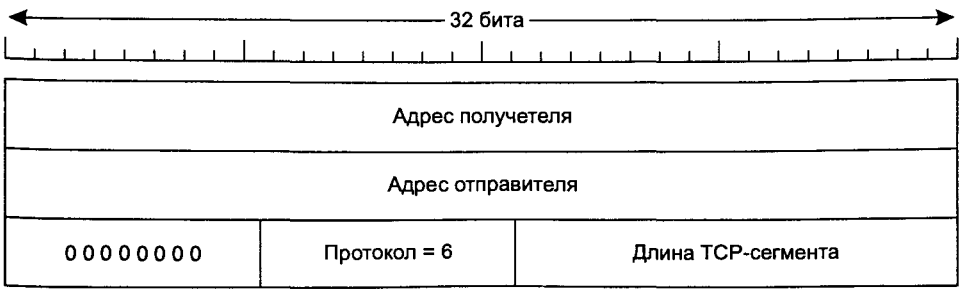


Рис. 6.24. Псевдозаголовок, аключаемый в контрольную сумму TCP

Псевдозаголовок содержит 32-разрядные IP-адреса отправителя и получателя, номер протокола для TCP (6) и счетчик байтов для TCP-сегмента (включая заголовок). Включение псевдозаголовка в контрольную сумму TCP помогает обнаружить неверно доставленные пакеты, хотя это нарушает иерархию протоколов, так как IP-адреса в нем принадлежат IP-уровню, а не TCP-уровню. В UDP для контрольной суммы используется такой же псевдозаголовок.

Поле *Факультативные поля* предоставляет дополнительные возможности, не покрываемые стандартным заголовком. С помощью одного из таких полей каждый хост может указать максимальный размер поля полезной нагрузки, который он может принять. Чем больше размер используемых сегментов, тем выше эффективность, так как при этом снижается удельный вес накладных расходов в виде 20-байтных заголовков, однако не все хосты способны принимать очень большие сегменты. Хосты могут сообщить друг другу максимальный размер поля полезной нагрузки во время установки соединения. По умолчанию этот размер равен 536 байтам. Все хосты обязаны принимать TCP-сегменты размером 536 + 20 = 556 байт. Для каждого из направлений может быть установлен свой максимальный размер поля полезной нагрузки.

Для линий с большой скоростью передачи и/или большой задержкой окно размером в 64 Кбайт оказывается слишком маленьким. Так, для линии T3 (44,736 Мбит/с) полное окно может быть передано в линию всего за 12 мс. Если значение времени распространения сигнала в оба конца составляет 50 мс (что типично для трансконтинентального оптического кабеля), 3/4 времени отправитель будет заниматься ожиданием подтверждения. При связи через спутник ситуация будет еще хуже. Больший размер окна мог бы улучшить эффективность, но 16-битовое поле *Размер окна* не позволяет этого сделать. В RFC 1323 был предложен новый параметр *Масштаб окна*, о значении которого два хоста могли договориться при установке соединения. Это число позволяет сдвигать поле *Размер окна* до 14 разрядов влево, обеспечивая расширение размера окна до 2^{30} байт (1 Гбайт). В настоящее время большинство реализаций протокола TCP поддерживают эту возможность.

Еще одна возможность, предложенная в RFC 1106 и широко применяемая сейчас, состоит в использовании протокола выборочного повтора вместо возврата на *n*. Если адресат получает один плохой сегмент и следом за ним большое количество хороших, у нормального TCP-протокола в конце концов истечет время ожидания и он передаст повторно все неподтвержденные сегменты, включая те, что были получены правильно. В документе RFC 1106 было предложено использовать отрицательные подтверждения (NAK), позволяющие получателю запрашивать отдельный сегмент или несколько сегментов. Получив его, принимающая сторона может подтвердить все хранящиеся в буфере данные, уменьшая таким образом количество повторно передаваемых данных.

Установка TCP-соединения

В протоколе TCP соединения устанавливаются с помощью «тройного рукопожатия», описанного в разделе «Установка соединения». Чтобы установить соедине-

ние, одна сторона (например, сервер) пассивно ожидает входящего соединения, выполняя примитивы LISTEN и ACCEPT, либо указывая конкретный источник, либо не указывая его.

Другая сторона (например, клиент) выполняет примитив CONNECT, указывая IP-адрес и порт, с которым он хочет установить соединение, максимальный размер TCP-сегмента и, по желанию, некоторые данные пользователя (например, пароль). Примитив CONNECT посылает TCP-сегмент с установленным битом *SYN* и сброшенным битом *ACK* и ждет ответа.

Когда этот сегмент прибывает в пункт назначения, TCP-сущность проверяет, выполнил ли какой-нибудь процесс примитив LISTEN, указав в качестве параметра тот же порт, который содержится в поле *Порт получателя*. Если такого процесса нет, она отвечает отправкой сегмента с установленным битом *RST* для отказа от соединения.

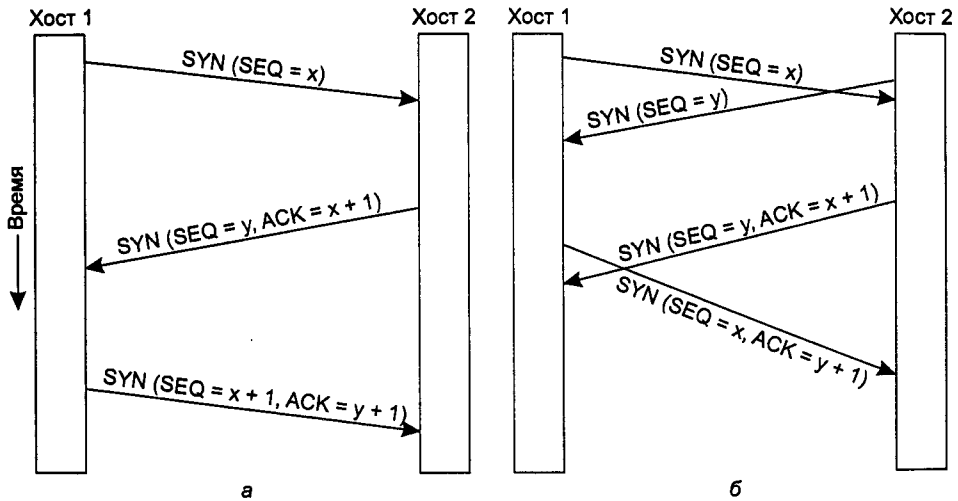


Рис. 6.25. Установка TCP-соединения в нормальном случае (*а*); столкновение вызовов (*б*)

Если какой-либо процесс прослушивает какой-либо порт, то входящий TCP-сегмент передается этому процессу. Последний может принять соединение или отказаться от него. Если процесс принимает соединение, он отсылает в ответ подтверждение. Последовательность TCP-сегментов, посылаемых в нормальном случае, показана на рис. 6.25, *а*. Обратите внимание на то, что сегмент с установленным битом *SYN* занимает 1 байт пространства порядковых номеров, что позволяет избежать неоднозначности в их подтверждениях.

Если два хоста одновременно попытаются установить соединение друг с другом, то последовательность происходящих при этом событий будет соответствовать рис. 6.25, *б*. В результате будет установлено только одно соединение, а не два, так как пара конечных точек однозначно определяет соединение. То есть если оба соединения пытаются идентифицировать себя с помощью пары (x, y), делается всего одна табличная запись для (x, y).

Начальное значение порядкового номера соединения не равно нулю по обсуждавшимся выше причинам. Используется схема, основанная на таймере, изменяющем свое состояние каждые 4 мкс. Для большей надежности хосту после сбоя запрещается перезагружаться ранее чем по прошествии максимального времени жизни пакета. Это позволяет гарантировать, что ни один пакет от прежних соединений не бродит где-нибудь в Интернете.

Разрыв соединения TCP

Хотя TCP-соединения являются полнодуплексными, чтобы понять, как происходит их разъединение, лучше считать их парами симплексных соединений. Каждое симплексное соединение разрывается независимо от своего напарника. Чтобы разорвать соединение, любая из сторон может послать TCP-сегмент с установленным в единицу битом *FIN*, что означает, что у него больше нет данных для передачи. Когда этот TCP-сегмент получает подтверждение, это направление передачи закрывается. Тем не менее, данные могут продолжать передаваться неопределенно долго в противоположном направлении. Соединение разрывается, когда оба направления закрываются. Обычно для разрыва соединения требуются четыре TCP-сегмента: по одному с битом *FIN* и по одному с битом *ACK* в каждом направлении. Первый бит *ACK* и второй бит *FIN* могут также содержаться в одном TCP-сегменте, что уменьшит количество сегментов до трех.

Как при телефонном разговоре, когда оба участника могут одновременно попрощаться и повесить трубки, оба конца TCP-соединения могут послать *FIN*-сегменты в одно и то же время. Они оба получают обычные подтверждения, и соединение закрывается. По сути, между одновременным и последовательным разъединениями нет никакой разницы.

Чтобы избежать проблемы двух армий, используются таймеры. Если ответ на посланный *FIN*-сегмент не приходит в течение двух максимальных интервалов времени жизни пакета, отправитель *FIN*-сегмента разрывает соединение. Другая сторона в конце концов заметит, что ей никто не отвечает, и также разорвет соединение. Хотя такое решение и не идеально, но, учитывая недостижимость идеала, приходится пользоваться тем, что есть. На практике проблемы возникают довольно редко.

Модель управления TCP-соединением

Этапы, необходимые для установки и разрыва соединения, могут быть представлены в виде модели конечного автомата, 11 состояний которого перечислены в табл. 6.6. В каждом из этих состояний могут происходить разрешенные и запрещенные события. В ответ на какое-либо разрешенное событие может осуществляться определенное действие. При возникновении запрещенных событий сообщается об ошибке.

Каждое соединение начинается в состоянии *CLOSED* (закрытое). Оно может выйти из этого состояния, предпринимая либо активную (CONNECT), либо пассивную (LISTEN) попытку открыть соединение. Если противоположная сторона осущест-

вляет противоположные действия, соединение устанавливается и переходит в состояние *ESTABLISHED*. Инициатором разрыва соединения может выступить любая сторона. По завершении процесса разъединения соединение возвращается в состояние *CLOSED*.

Таблица 6.5. Состояния конечного автомата, управляющего TCP-соединением

Состояние	Описание
CLOSED	Закрыто. Соединение не является активным и не находится в процессе установки
LISTEN	Ожидание. Сервер ожидает входящего запроса
SYN RCVD	Прибыл запрос соединения. Ожидание подтверждения
SYN SENT	Запрос соединения послан. Приложение начало открывать соединение
ESTABLISHED	Установлено. Нормальное состояние передачи данных
FIN WAIT 1	Приложение сообщило, что ему больше нечего передавать
FIN WAIT 2	Другая сторона согласна разорвать соединение
TIMED WAIT	Ожидание, пока в сети не исчезнут все пакеты
CLOSING	Обе стороны попытались одновременно закрыть соединение
CLOSE WAIT	Другая сторона инициировала разъединение
LAST ACK	Ожидание, пока в сети не исчезнут все пакеты

Конечный автомат показан на рис. 6.26. Типичный случай клиента, активно соединяющегося с пассивным сервером, показан жирными линиями — сплошными для клиента и пунктирными для сервера. Тонкие линии обозначают необычные последовательности событий. Каждая линия на рис. 6.26 маркирована парой *событие/действие*. Событие может представлять собой либо обращение пользователя к системной процедуре (CONNECT, LISTEN, SEND или CLOSE), либо прибытие сегмента (*SYN*, *FIN*, *ACK* или *RST*), либо, в одном случае, окончание периода ожидания, равного двойному времени жизни пакетов. Действие может состоять в отправке управляющего сегмента (*SYN*, *FIN* или *RST*). Впрочем, может не предприниматься никакого действия, что обозначается прочерком. В скобках приводятся комментарии.

Диаграмму легче всего понять, если сначала проследовать по пути клиента (сплошная жирная линия), а затем — по пути сервера (жирный пунктир). Когда приложение на машине клиента вызывает примитив CONNECT, локальная TCP-сущность создает запись соединения, помечает его состояние как *SYN SENT* и посылает *SYN*-сегмент. Примечательно, что несколько приложений одновременно могут открыть несколько соединений, поэтому свое состояние, хранящееся в записи соединения, имеется у каждого отдельного соединения. Когда прибывает сегмент *SYN* + *ACK*, TCP-сущность посылает последний *ACK*-сегмент «тройного рукопожатия» и переключается в состояние *ESTABLISHED*. В этом состоянии можно пересылать и получать данные.

Когда у приложения заканчиваются данные для передачи, оно выполняет примитив CLOSE, заставляющий локальную TCP-сущность послать *FIN*-сегмент и ждать ответного *ACK*-сегмента (пунктирный прямоугольник с пометкой «активное

разъединение»). Когда прибывает подтверждение, происходит переход в состояние *FIN WAIT 2*, и одно направление соединения закрывается. Когда приходит встречный *FIN*-сегмент, в ответ на него также высылается подтверждение, и второе направление соединения также закрывается. Теперь обе стороны соединения закрыты, но TCP-сущность выжидает в течение времени, равного максимальному времени жизни пакета, чтобы можно было гарантировать, что все пакеты этого соединения больше не перемещаются по сети даже в том случае, если подтверждение было потеряно. Когда этот период ожидания истекает, TCP-сущность удаляет запись о соединении.

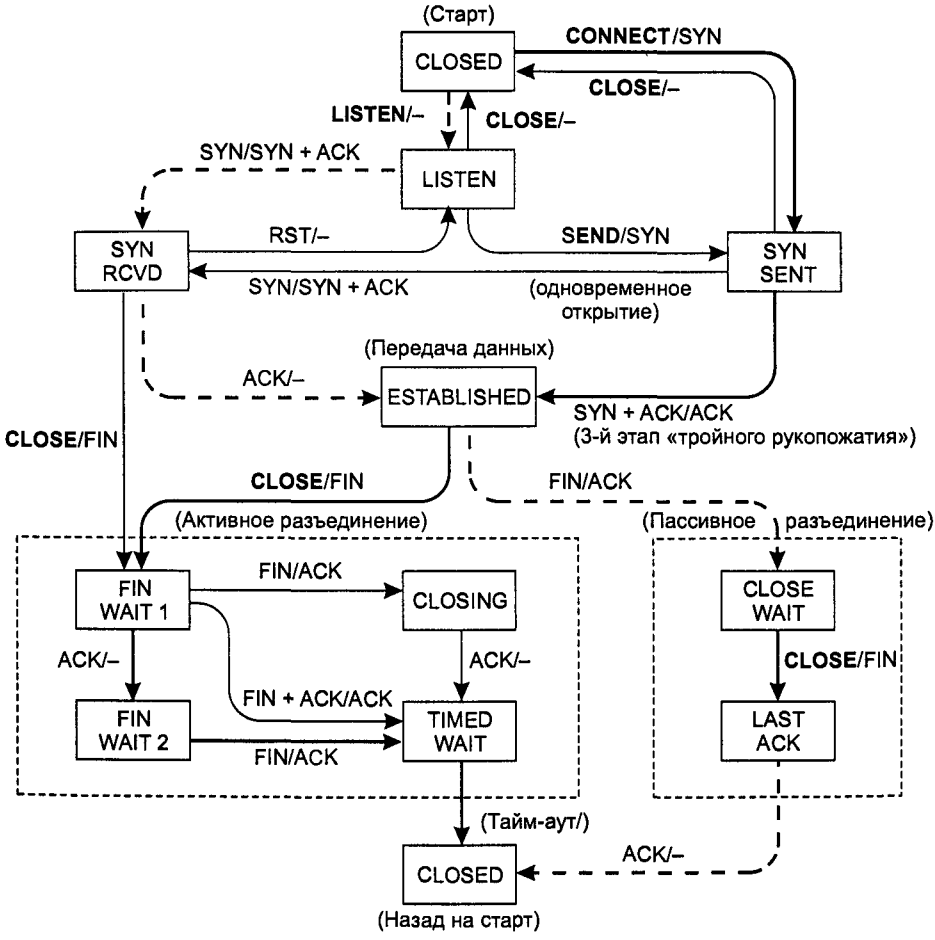


Рис. 6.26. Конечный автомат TCP-соединения. Жирная сплошная линия показывает нормальный путь клиента. Пунктиром показан нормальный путь сервера. Тонкими линиями обозначены необычные события

Рассмотрим теперь управление соединением с точки зрения сервера. Сервер выполняет примитив LISTEN и переходит в режим ожидания запросов соедине-

ния. Когда приходит *SYN*-сегмент, в ответ на него высылается подтверждение, после чего сервер переходит в состояние *SYN RCVD* (запрос соединения получен). Когда в ответ на *SYN*-подтверждение сервера от клиента приходит *ACK*-сегмент, процедура «тройного рукопожатия» завершается и сервер переходит в состояние *ESTABLISHED*. Теперь можно пересылать данные.

По окончании выполнения своей задачи клиент запускает примитив CLOSE, в результате чего на сервер прибывает *FIN*-сегмент (пунктирный прямоугольник, обозначенный как пассивное разъединение). Теперь сервер выполняет примитив CLOSE, а *FIN*-сегмент посылается клиенту. Когда от клиента прибывает подтверждение, сервер разрывает соединение и удаляет запись о нем.

Управление передачей в TCP

Как уже было сказано ранее, управление окном в TCP не привязано напрямую к подтверждениям, как это сделано в большинстве протоколов передачи данных. Например, предположим, что у получателя есть 4096-байтовый буфер, как показано на рис. 6.27. Если отправитель передает 2048-байтовый сегмент, который успешно принимается получателем, то получатель подтверждает его получение. Однако при этом у получателя остается всего лишь 2048 байт свободного буферного пространства (пока приложение не заберет сколько-нибудь данных из буфера), о чем он и сообщает отправителю, указывая соответствующий размер окна (2048) и номер следующего ожидаемого байта.

После этого отправитель посылает еще 2048 байт, получение которых подтверждается, но размер окна объявляется равным 0. Отправитель должен прекратить передачу до тех пор, пока получающий хост не освободит место в буфере и не увеличит размер окна.

При нулевом размере окна отправитель не может посылать сегменты, за исключением двух случаев. Во-первых, разрешается посылать срочные данные, например, чтобы пользователь мог уничтожить процесс, выполняющийся на удаленной машине. Во-вторых, отправитель может послать 1-байтовый сегмент, прося получателя повторить информацию о размере окна и ожидаемом следующем байте. Стандарт TCP явно предусматривает эту возможность для предотвращения тупиковых ситуаций в случае потери объявления о размере окна.

Отправители не обязаны передавать данные сразу, как только они приходят от приложения. Также никто не требует от получателей посылать подтверждения как можно скорее. Например, на рис. 6.27 TCP-сущность, получив от приложения первые 2 Кбайт данных и зная, что доступный размер окна равен 4 Кбайт, была бы совершенно права, если бы просто сохранила полученные данные в буфере до тех пор, пока не прибудут еще 2 Кбайт данных, чтобы передать сразу сегмент с 4 Кбайт полезной нагрузки. Эта свобода действий может использоваться для улучшения производительности.

Рассмотрим TELNET-соединение с интерактивным редактором, реагирующим на каждое нажатие клавиши. В худшем случае, когда символ прибывает к передающей TCP-сущности, она создает 21-байтовый TCP-сегмент и передает его IP-уровню, который, в свою очередь, посылает 41-байтовую IP-дейтаграмму.

На принимающей стороне TCP-сущность немедленно отвечает 40-байтовым подтверждением (20 байт TCP-заголовка и 20 байт IP-заголовка). Затем, когда редактор прочитает этот байт из буфера, TCP-сущность пошлет обновленную информацию о размере буфера, передвинув окно на 1 байт вправо. Размер этого пакета также составляет 40 байт. Наконец, когда редактор обработает этот символ, он отправляет обратно эхо, передаваемое 41-байтовым пакетом. Итого для каждого введенного с клавиатуры символа пересылается четыре пакета общим размером 162 байта. В условиях дефицита пропускной способности линий этот метод работы нежелателен.

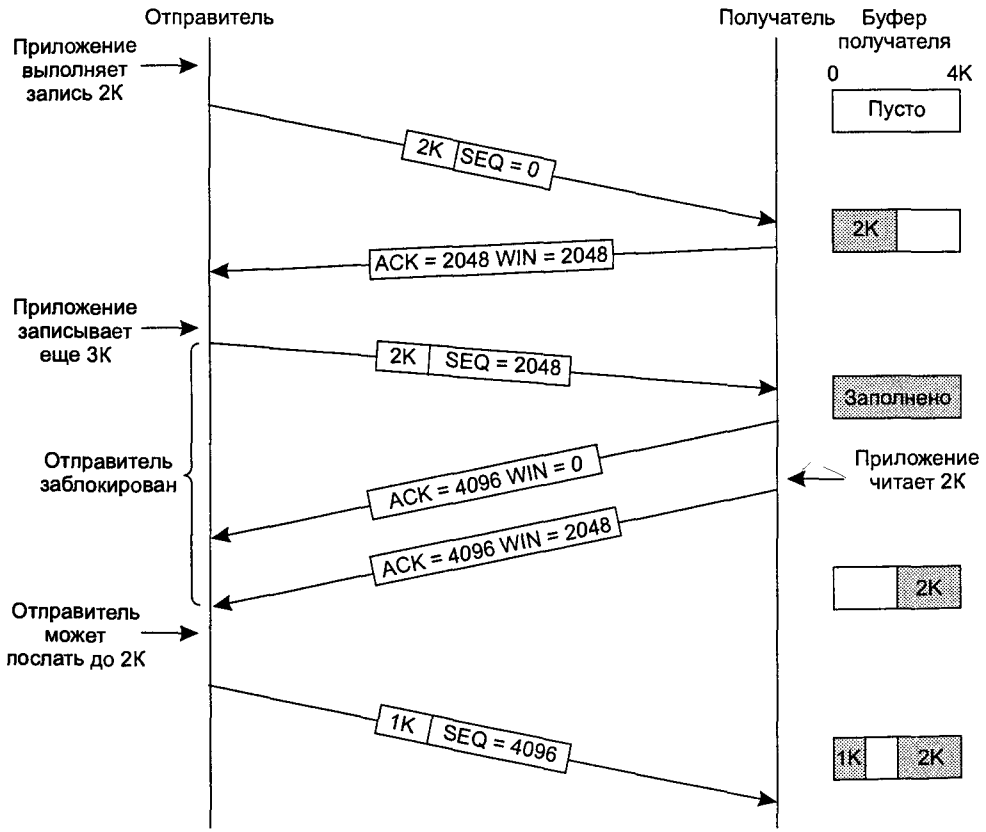


Рис. 6.27. Управление окном в TCP

Для улучшения ситуации многие реализации TCP используют задержку подтверждений и обновлений размера окна на 500 мс в надежде получить дополнительные данные, вместе с которыми можно будет отправить подтверждение одним пакетом. Если редактор успеет выдать эхо в течение 500 мс, удаленному пользователю нужно будет выслать только один 41-байтовый пакет, таким образом, нагрузка на сеть снизится вдвое.

Хотя такой метод задержки и снижает нагрузку на сеть, тем не менее, эффективность использования сети отправителем продолжает оставаться невысокой, так как каждый байт пересылается в отдельном 41-байтовом пакете. Метод, позволяющий повысить эффективность, известен как **алгоритм Нагля** (Nagle, 1984). Предложение Нагля звучит довольно просто: если данные поступают отправителю по одному байту, отправитель просто передает первый байт, а остальные помещает в буфер, пока не будет получено подтверждение приема первого байта. После этого можно переслать все накопленные в буфере символы в виде одного TCP-сегмента и снова начать буферизацию до получения подтверждения отосланных символов. Если пользователь вводит символы быстро, а сеть медленная, то в каждом сегменте будет передаваться значительное количество символов, таким образом, нагрузка на сеть будет существенно снижена. Кроме того, этот алгоритм позволяет посылать новый пакет, даже если число символов в буфере превышает половину размера окна или максимальный размер сегмента.

Алгоритм Нагля широко применяется различными реализациями протокола TCP, однако иногда бывают ситуации, в которых его лучше отключить. В частности, при работе приложения X-Windows в Интернете информация о перемещениях мыши пересылается на удаленный компьютер. (X-Window — это система управления окнами в большинстве ОС типа UNIX). Если буферизировать эти данные для пакетной пересылки, курсор будет перемещаться рывками с большими паузами, в результате чего пользоваться программой будет очень сложно, почти невозможно.

Еще одна проблема, способная значительно снизить производительность протокола TCP, известна под именем **синдрома глупого окна** (Clark, 1982). Суть проблемы состоит в том, что данные пересылаются TCP-сущностью крупными блоками, но принимающая сторона интерактивного приложения считывает их посимвольно. Чтобы ситуация стала понятнее, рассмотрим рис. 6.28. Начальное состояние таково: TCP-буфер приемной стороны полон, и отправителю это известно (то есть размер его окна равен 0). Затем интерактивное приложение читает один символ из TCP-потока. Принимающая TCP-сущность радостно сообщает отправителю, что размер окна увеличился, и что он теперь может послать 1 байт. Отправитель повинуется и посылает 1 байт. Буфер снова оказывается полон, о чем получатель и извещает, посылая подтверждение для 1-байтового сегмента с нулевым размером окна. И так может продолжаться вечно.

Дэвид Кларк (David Clark) предложил запретить принимающей стороне отправлять информацию об однобайтовом размере окна. Вместо этого получатель должен подождать, пока в буфере не накопится значительное количество свободного места. В частности, получатель не должен отправлять сведения о новом размере окна до тех пор, пока он не сможет принять сегмент максимального размера, который он объявлял при установке соединения, или его буфер не освободился хотя бы наполовину.

Кроме того, увеличению эффективности отправки может способствовать сам отправитель, отказываясь от отправки слишком маленьких сегментов. Вместо этого он должен подождать, пока размер окна не станет достаточно большим для

того, чтобы можно было послать полный сегмент или, по меньшей мере, равный половине размера буфера получателя. (Отправитель может оценить этот размер по последовательности сообщений о размере окна, полученных им ранее.)

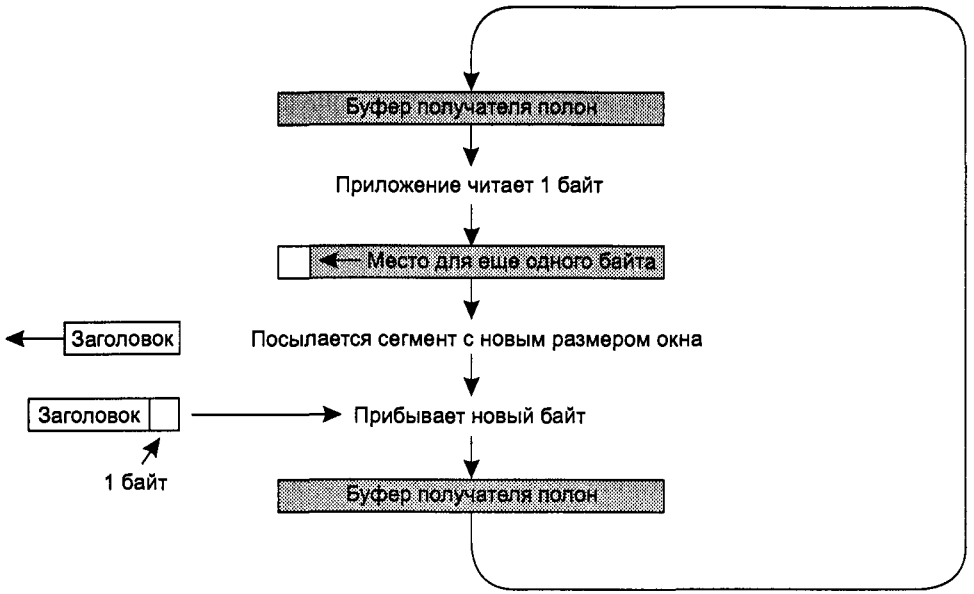


Рис. 6.28. Синдром глупого окна

В задаче избавления от синдрома глупого окна алгоритм Нагля и решение Кларка дополняют друг друга. Нагль пытался решить проблему приложения, предоставляющего данные TCP-сущности посимвольно. Кларк старался разрешить проблему приложения, посимвольно получающего данные у TCP. Оба решения хороши и могут работать одновременно. Суть их состоит в том, чтобы не посылать и не просить передавать данные слишком малыми порциями.

Принимающая TCP-сущность может пойти еще дальше в деле повышения производительности, просто обновляя информацию о размере окна большими порциями. Как и отправляющая TCP-сущность, она также может буферизировать данные и блокировать запрос на чтение READ, поступающий от приложения, до тех пор, пока у нее не накопится большого объема данных. Таким образом, снижается количество обращений к TCP-сущности и, следовательно, снижаются накладные расходы. Конечно, такой подход увеличивает время ожидания ответа, но для неинтерактивных приложений, например при передаче файла, сокращение времени, затраченного на всю операцию, значительно важнее увеличения времени ожидания ответа на отдельные запросы.

Еще одна проблема получателя состоит в сегментах, полученных в неправильном порядке. Они могут храниться или отвергаться по усмотрению получателя. Разумеется, подтверждение может быть выслано, только если все данные вплоть до подтверждаемого байта получены. Если до получателя доходят сег-

менты 0, 1, 2, 4, 5, 6 и 7, он может подтвердить получение данных вплоть до последнего байта сегмента 2. Когда у отправителя истечет время ожидания, он передаст сегмент 3 еще раз. Если к моменту прибытия сегмента 3 получатель сохранит в буфере сегменты с 4-го по 7-й, он сможет подтвердить получение всех байтов, вплоть до последнего байта сегмента 7.

Борьба с перегрузкой в TCP

Когда в какую-либо сеть поступает больше данных, чем она способна обработать, в сети образуются заторы. Интернет в этом смысле не является исключением. В данном разделе мы обсудим алгоритмы, разработанные (за последние двадцать пять лет) для борьбы с перегрузкой сети. Хотя сетевой уровень также пытается бороться с перегрузкой, основной вклад в решение этой проблемы, заключающееся в снижении скорости передачи данных, осуществляется протоколом TCP.

Теоретически, с перегрузкой можно бороться с помощью принципа, заимствованного из физики, — закона сохранения пакетов. Идея состоит в том, чтобы не передавать в сеть новые пакеты, пока ее не покинут (то есть не будут доставлены) старые. Протокол TCP пытается достичь этой цели с помощью динамического управления размером окна.

Первый шаг в борьбе с перегрузкой состоит в том, чтобы обнаружить ее. Пару десятилетий назад обнаружить перегрузку в сети было сложно. Трудно было понять, почему пакет не доставлен вовремя. Помимо возможности перегрузки сети имелась также большая вероятность потерять пакет вследствие высокого уровня помех на линии.

В настоящее время потери пакетов при передаче случаются относительно редко, так как большинство междугородных линий связи являются оптоволоконными (хотя в беспроводных сетях процент пакетов, теряемых из-за помех, довольно высок). Соответственно, большинство потерянных пакетов в Интернете вызвано заторами. Все TCP-алгоритмы Интернета предполагают, что потери пакетов вызываются перегрузкой сети, и следят за тайм-аутами как за предвестниками проблем, подобно шахтерам, наблюдающим за своими канарейками.

Прежде чем перейти к обсуждению того, как TCP реагирует на перегрузку, опишем сначала способы ее предотвращения, применяемые протоколом. При обнаружении перегрузки должен быть выбран подходящий размер окна. Получатель может указать размер окна, исходя из количества свободного места в буфере. Если отправитель будет иметь в виду размер отведенного ему окна, переполнение буфера у получателя не сможет стать причиной проблемы, однако она все равно может возникнуть из-за перегрузки на каком-либо участке сети между отправителем и получателем.

На рис. 6.29 эта проблема проиллюстрирована на примере водопровода. На рис. 6.29, *а* мы видим толстую трубу, ведущую к получателю с небольшой емкостью. До тех пор, пока отправитель не посылает воды больше, чем может поместиться в ведро, вода не будет проливаться. На рис. 6.29, *б* ограничительным фак-

тором является не емкость ведра, а пропускная способность сети. Если из крана в воронку вода будет литься слишком быстро, то уровень воды в воронке начнет подниматься и, в конце концов, часть воды может перелиться через край воронки.

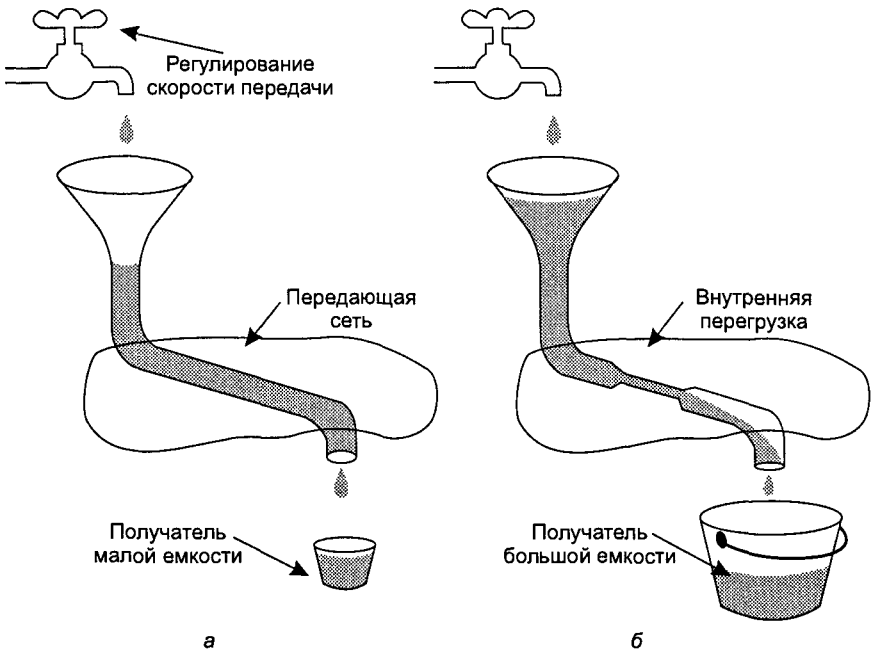


Рис. 6.29. Быстрая сеть и получатель малой емкости (*а*); медленная сеть и получатель большой емкости (*б*)

Решение, применяемое в Интернете, состоит в признании существования двух потенциальных проблем: низкой пропускной способности сети и низкой емкости получателя — и в раздельном решении обеих проблем. Для этого у каждого отправителя есть два окна: окно, предоставленное получателем, и **окно перегрузки**. Размер каждого из них соответствует количеству байтов, которое отправитель имеет право передать. Отправитель руководствуется минимальным из этих двух значений. Например, получатель говорит: «Посылайте 8 Кбайт», но отправитель знает, что если он пошлет более 4 Кбайт, то в сети образуется затор, поэтому он посылает все же 4 Кбайт. Если же отправитель знает, что сеть способна пропустить и большее количество данных, например 32 Кбайт, он передаст столько, сколько просит получатель (то есть 8 Кбайт).

При установке соединения отправитель устанавливает размер окна перегрузки равным размеру максимального используемого в данном соединении сегмента. Затем он передает один максимальный сегмент. Если подтверждение получения этого сегмента прибывает прежде, чем истекает период ожидания, к размеру окна добавляется размер сегмента, то есть размер окна перегрузки удваивается, и посылаются уже два сегмента. В ответ на подтверждение получения каждого из сегментов производится расширение окна перегрузки на величину одного мак-

симального сегмента. Допустим, размер окна равен *n* сегментам. Если подтверждения для всех сегментов приходят вовремя, окно увеличивается на число байтов, соответствующее *n* сегментам. По сути, подтверждение каждой последовательности сегментов приводит к удвоению окна перегрузки.

Этот процесс экспоненциального роста продолжается до тех пор, пока не будет достигнут размер окна получателя или не будет выработан признак тайм-аута, сигнализирующий о перегрузке в сети. Например, если пакеты размером 1024, 2048 и 4096 байт дошли до получателя успешно, а в ответ на передачу пакета размером 8192 байта подтверждение не пришло в установленный срок, окно перегрузки устанавливается равным 4096 байтам. Пока размер окна перегрузки остается равным 4096 байтам, более длинные пакеты не посылаются, независимо от размера окна, предоставляемого получателем. Этот алгоритм называется **затяжным пуском**, или **медленным пуском**. Однако он не такой уж и медленный (Jacobson, 1988). Он экспоненциальный. Все реализации протокола TCP обязаны его поддерживать.

Рассмотрим теперь механизм борьбы с перегрузкой, применяемый в Интернете. Помимо окон получателя и перегрузки, в качестве третьего параметра в нем используется **пороговое значение**, которое изначально устанавливается равным 64 Кбайт. Когда возникает ситуация тайм-аута (подтверждение не возвращается в срок), новое значение порога устанавливается равным половине текущего размера окна перегрузки, а окно перегрузки уменьшается до размера одного максимального сегмента. Затем, так же как и в предыдущем случае, используется алгоритм затяжного пуска, позволяющий быстро обнаружить предел пропускной способности сети. Однако на этот раз экспоненциальный рост размера окна останавливается по достижении им порогового значения, после чего окно увеличивается линейно, на один сегмент для каждой следующей передачи. В сущности, предполагается, что можно спокойно урезать вдвое размер окна перегрузки, после чего постепенно наращивать его.

Иллюстрация работы данного алгоритма борьбы с перегрузкой приведена на рис. 6.30. Максимальный размер сегмента в данном примере равен 1024 байт. Сначала окно перегрузки было установлено равным 64 Кбайт, но затем произошел тайм-аут, и порог стал равным 32 Кбайт, а окно перегрузки — 1 Кбайт (передача 0). Затем размер окна перегрузки удваивается на каждом шаге, пока не достигает порога (32 Кбайт). Начиная с этого момента, размер окна увеличивается линейно.

Передача 13 оказывается несчастливой (как и положено), так как срабатывает тайм-аут. При этом пороговое значение устанавливается равным половине текущего размера окна (40 Кбайт пополам, то есть 20 Кбайт), и опять происходит затяжной пуск. После достижения порогового значения экспоненциальный рост размера окна сменяется линейным.

Если тайм-аутов больше не возникает, окно перегрузки может продолжать расти до размера окна получателя. Затем рост прекратится, и размер окна останется постоянным, пока не произойдет тайм-аут или не изменится размер окна получателя. Прибытие ICMP-пакета SOURCE QUENCH (гашение источника) обрабатывается таким же образом, как и тайм-аут. Альтернативный (и более современный) подход описан в RFC 3168.

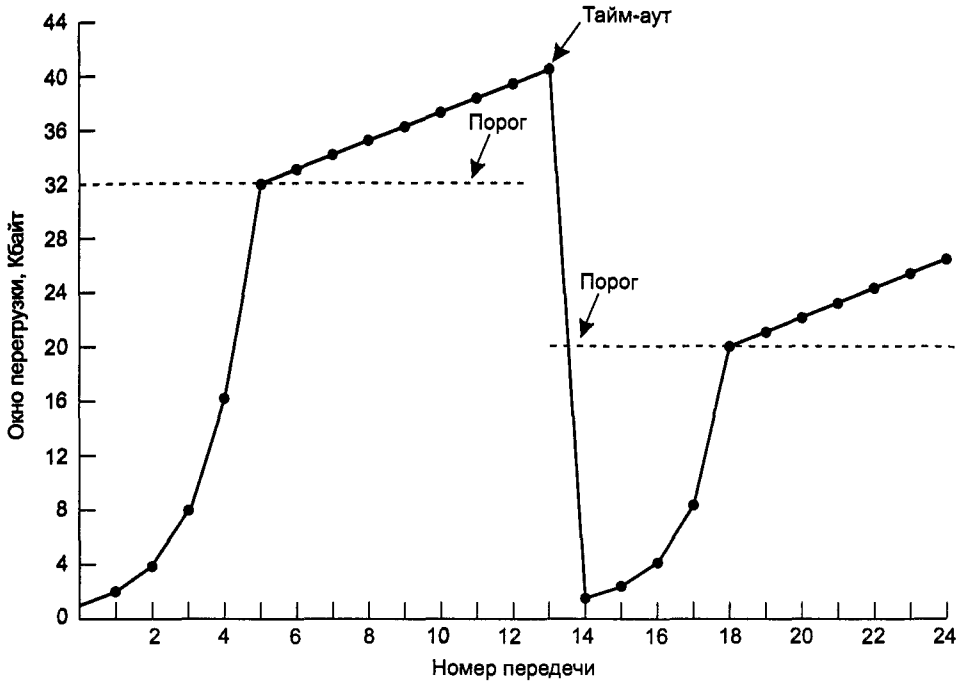


Рис. 6.30. Пример алгоритма борьбы с перегрузкой, применяемого в Интернете

Управление таймерами в TCP

В протоколе TCP используется много различных таймеров (по крайней мере, такова концепция). Наиболее важным из них является **таймер повторной передачи**. Когда посылается сегмент, запускается таймер повторной передачи. Если подтверждение получения сегмента прибывает раньше, чем истекает период таймера, таймер останавливается. Если же, наоборот, период ожидания истечет раньше, чем прибудет подтверждение, сегмент передается еще раз (а таймер запускается снова). Соответственно возникает вопрос: каким должен быть интервал времени ожидания?

На транспортном уровне Интернета эта проблема оказывается значительно сложнее, чем на уровне передачи данных, описанном в главе 3. На уровне передачи данных величину ожидаемой задержки довольно легко предсказать (ее разброс невелик), поэтому таймер можно установить на момент времени чуть позднее ожидаемого прибытия подтверждения (рис. 6.31, *а*). Поскольку на уровне передачи данных подтверждения редко запаздывают на больший срок (благодаря тому, что нет заторов), отсутствие подтверждения в течение установленного временного интервала с большой вероятностью означает потерю кадра или подтверждения.

Протокол TCP вынужден работать в совершенно иных условиях. Функция распределения плотности вероятности времени прибытия подтверждения на этом

уровне выглядит значительно более полого (рис. 6.31, *б*). Поэтому предсказать, сколько потребуется времени для прохождения данных от отправителя до получателя и обратно, весьма непросто. Если выбрать значение интервала ожидания слишком малым (например, T_1 на рис. 6.31, *б*), возникнут излишние повторные передачи, забивающие Интернет бесполезными пакетами. Если же установить значение этого интервала слишком большим (T_2), то из-за увеличения времени ожидания в случае потери пакета пострадает производительность. Более того, среднее значение и величина дисперсии времени прибытия подтверждений может изменяться всего за несколько секунд при возникновении и устранении перегрузки.

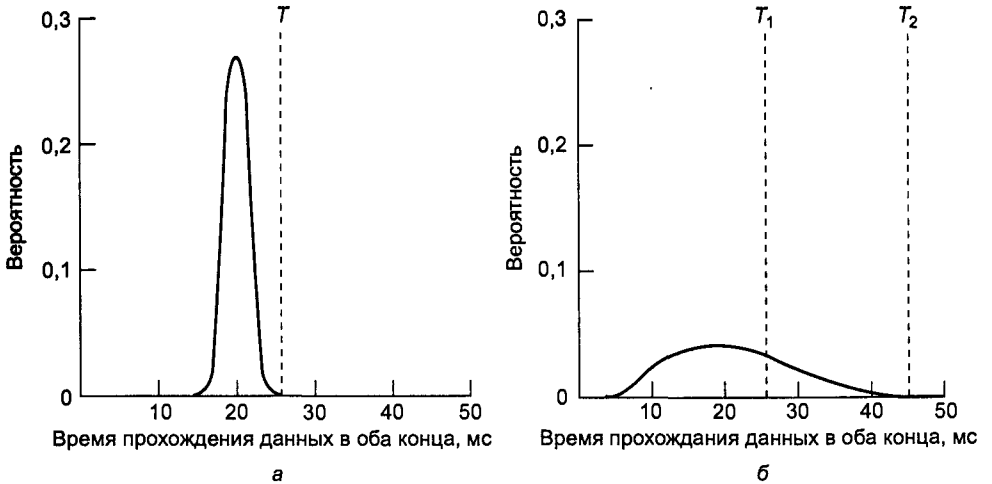


Рис. 6.31. Плотность вероятности времени прибытия подтверждения на уровне передачи данных (а); плотность вероятности времени прибытия подтверждения на транспортном уровне (*б*)

Решение заключается в использовании крайне динамичного алгоритма, постоянно изменяющего величину периода ожидания, основываясь на измерениях производительности сети. Алгоритм, применяемый в TCP, разработан Джекобсоном (Jacobson) в 1988 году и работает следующим образом. Для каждого соединения в протоколе TCP предусмотрена переменная *RTT* (Round-Trip Time — время перемещения в оба конца), в которой хранится наименьшее время получения подтверждения для данного соединения. При передаче сегмента запускается таймер, который измеряет время, требуемое для получения подтверждения, а также запускает повторную передачу, если подтверждение не приходит в срок. Если подтверждение успевает вернуться прежде чем истечет период ожидания, TCP-сущность измеряет время, потребовавшееся для его получения (*M*). Затем значение переменной *RTT* обновляется по следующей формуле:

$$RTT = \alpha RTT + (1 - \alpha)M,$$

где α — весовой коэффициент, обычно равный 7/8.

Даже при известном значении переменной *RTT* выбор периода ожидания подтверждения оказывается задачей нетривиальной. Обычно в протоколе TCP это значение вычисляется как βRTT. Остается только выбрать каким-нибудь хитрым образом соответствующее значение коэффициента β. В ранних реализациях протокола использовалось $\beta = 2$, однако экспериментально было показано, что постоянное значение β является негибким и плохо учитывает ситуации, при которых разброс значений времени прибытия подтверждения увеличивается.

В 1988 г. Джекобсон предложил использовать значение β, грубо пропорциональное среднеквадратичному отклонению (дисперсии) функции плотности вероятности времени прибытия подтверждения. Таким образом, при увеличении разброса значений времени прибытия увеличивалось бы и значение β, и наоборот. В частности, он предложил использовать *среднее линейное отклонение* в качестве легко вычисляемой оценки *среднеквадратичного отклонения*. В его версии алгоритма для каждого соединения вводилась еще одна сглаженная переменная *D*, отклонение. При получении каждого подтверждения вычислялась абсолютная величина разности между ожидаемым и измеренным значениями $|RTT - M|$. Сглаженное значение этой величины сохранялось в переменной *D*, вычисляемой по формуле

$$D = \alpha D + (1 - \alpha)\,|RTT - M|,$$

где β в общем случае могло выбираться отличным от значения, используемого в предыдущей формуле. Хотя значение переменной *D* и отличается от среднеквадратичного отклонения, оно достаточно хорошо подходит для данного алгоритма. Кроме того, Джекобсон показал, как можно вычислить значение этой переменной, используя только целочисленные сложение, вычитание и сдвиг, что явилось большим плюсом. В настоящее время этот алгоритм применяется в большинстве реализаций протокола TCP, а значение интервала ожидания устанавливается по формуле

$$\text{Время ожидания} = RTT + 4D.$$

Выбор множителя 4 является произвольным, однако это значение обладает двумя преимуществами. Во-первых, умножение целого числа на 4 может быть выполнено одной командой сдвига. Во-вторых, относительное количество излишних повторных передач при таком значении времени ожидания не превысит одного процента. (Вначале Джекобсон предлагал умножать *D* на 2, но последующие исследования показали, что 4 дает лучшую производительность.)

При динамической оценке величины *RTT* возникает вопрос, что делать со значением *RTT* при повторной передаче сегмента. Когда, наконец, прибывает подтверждение для такого сегмента, непонятно, относится это подтверждение к первой передаче пакета или же к последней. Неверная догадка может серьезно снизить точность оценки *RTT*. Эта проблема была обнаружена радиолюбителем Филом Карном (Phil Karn). Его интересовал вопрос передачи TCP/IP-пакетов с помощью коротковолновой любительской радиосвязи, известной своей ненадежностью (в лучшем случае до адресата доходит лишь половина пакетов). Предложение Ф. Карна было очень простым: не обновлять значение *RTT* для переданных

повторно пакетов. Вместо этого при каждой повторной передаче время ожидания можно удваивать до тех пор, пока сегменты не пройдут с первой попытки. Это исправление получило название **алгоритма Карна**. Он применяется в большинстве реализаций протокола TCP.

В протоколе TCP используется не только таймер повторной передачи. Еще один применяемый в этом протоколе таймер называется **таймером настойчивости**. Он предназначен для предотвращения следующей тупиковой ситуации. Получатель посылает подтверждение, в котором указывает окно нулевого размера, давая тем самым отправителю команду подождать. Через некоторое время получатель посылает пакет с новым размером окна, но этот пакет теряется. Теперь обе стороны ожидают действий противоположной стороны. Когда срабатывает таймер настойчивости, отправитель посылает получателю пакет с вопросом, не изменилось ли текущее состояние. В ответ получатель сообщает текущий размер окна. Если он все еще равен нулю, таймер настойчивости запускается снова, и весь цикл повторяется. Если же окно увеличилось, отправитель может передавать данные.

В некоторых реализациях протокола используется третий таймер, называемый **дежурным таймером**. Когда соединение не используется в течение долгого времени, срабатывает дежурный таймер, заставляя одну сторону проверить, есть ли еще кто живой на том конце соединения. Если проверяющая сторона не получает ответа, соединение разрывается. Эта особенность протокола довольно противоречива, поскольку она приносит дополнительные накладные расходы и может разорвать вполне жизнеспособное соединение из-за кратковременной потери связи.

Последний таймер, используемый в каждом TCP-соединении, — это таймер, запускаемый в состоянии *TIMED WAIT* конечного автомата при закрытии соединения. Он отсчитывает двойное время жизни пакета, чтобы гарантировать, что после закрытия соединения в сети не останутся созданные им пакеты.

Беспроводные протоколы TCP и UDP

Теоретически, транспортные протоколы не должны зависеть от технологии, применяемой на расположенном ниже сетевом уровне. В частности, протоколу TCP должно быть все равно, передаются его сегменты по радио или по оптоволоконному кабелю. На практике, тем не менее, это имеет значение, так как основная часть реализаций протокола TCP подверглась детальной оптимизации, основанной на предположениях, справедливых для проводных сетей, но неверных для беспроводных. Игнорирование свойств беспроводной связи может привести к появлению реализации протокола TCP, которая будет логически корректной, но в то же время будет характеризоваться ужасающе низкой производительностью.

Основным вопросом является алгоритм борьбы с перегрузкой. Почти все нынешние реализации протокола TCP предполагают, что тайм-ауты вызываются заторами, а не потерей пакетов. Соответственно, когда время ожидания истекает, протокол TCP снижает скорость передачи (например, по алгоритму затяжного

пуска Джекобсона), чтобы ослабить нагрузку на сеть и тем самым способствовать устранению затора.

К сожалению, беспроводные линии передачи являются чрезвычайно ненадежными. Они постоянно теряют пакеты. Правильная реакция на потерю пакета должна состоять в его скорейшей повторной отправке. Снижение скорости может лишь ухудшить ситуацию. Если, к примеру, 20 % всех пакетов теряется, а отправитель передает по 100 пакетов в секунду, то до получателя доходит около 80 пакетов в секунду. Если отправитель снизит скорость до 50 пакетов в секунду, на выходе скорость упадет до 40 пакетов в секунду.

Таким образом, если пакет теряется в проводной сети, отправитель должен снизить скорость. Если же пакет теряется в беспроводной сети, отправитель должен не снижать скорость, а, возможно, даже увеличить ее (это напоминает разницу в способах вывода из заноса переднеприводных и заднеприводных автомобилей. — *Примеч. перев.*). Если отправитель не знает, в какой сети он находится, ему трудно принять верное решение.

Часто путь от отправителя до получателя оказывается неоднородным. Первые 1000 км могут проходить по проводной сети, но последний километр может оказаться беспроводным. В такой ситуации принять правильное решение в случае тайм-аута еще сложнее, так как его причины могут быть различными. Предложенное решение, получившее название **непрямого протокола TCP** (Bakne и Badrinath, 1995), состояло в разбиении TCP-соединения на два отдельных соединения, как показано на рис. 6.32. Первое соединение тянется от отправителя до базовой станции, а второе — от базовой станции до получателя. Базовая станция просто копирует пакеты из одного соединения в другое в обоих направлениях.

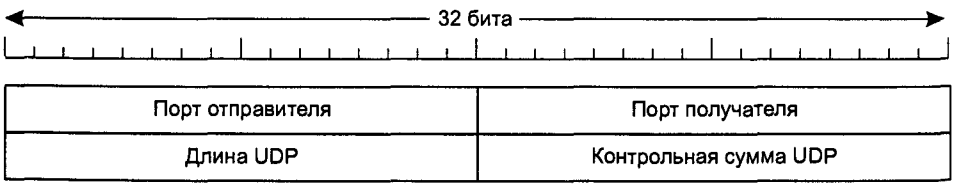


Рис. 6.32. Разбиение TCP-соединения на два

Преимущество этой схемы состоит в том, что два соединения, на которые разбивается TCP-соединение, оказываются однородными. В ответ на тайм-ауты в первом соединении отправитель может снизить скорость, а на тайм-ауты во втором — увеличить. Другие параметры также могут настраиваться независимо для каждого соединения. Недостаток заключается в том, что такое решение нарушает семантику протокола TCP. Поскольку каждая часть соединения представляет собой полноценное TCP-соединение, базовая станция сама подтверждает получение каждого сегмента обычным способом. Только теперь получение подтверждения отправителем означает не то, что сегмент успешно добрался до получателя, а только то, что его получила базовая станция.

Другое решение, разработанное Балакришнаном (Balakrishnan и др., 1995), не нарушает семантики протокола TCP. В его основе лежат небольшие изменения

в программе сетевого уровня, работающей на базовой станции. Одно из изменений состоит в добавлении специального следящего агента, просматривающего и кэширующего сегменты, направляемые мобильному хосту, и подтверждений, посылаемых им в ответ. Если следящий агент замечает, что в ответ на TCP-сегмент, пересылаемый им мобильному хосту, не поступает подтверждения, он просто передает этот сегмент еще раз, не информируя об этом отправителя. У следящего агента есть свой таймер для отслеживания подтверждений, устанавливаемый на относительно небольшой интервал времени. Он также повторно передает сегменты, когда получает от мобильного хоста дубликаты подтверждений, означающие, что хосту чего-то не хватает для счастья. Дубликаты подтверждений уничтожаются на месте, чтобы отправитель на другом конце кабельной сети не принял их за сигнал о перегрузке.

Недостаток такой прозрачности состоит в том, что при частых потерях сегментов на беспроводном участке у отправителя может истечь интервал ожидания, и он может запустить механизм борьбы с перегрузкой. В предыдущем варианте составного TCP-соединения, напротив, алгоритм борьбы с перегрузкой никогда бы не запустился, если только в сети действительно не образовался бы затор.

Метод Балакришнана также решает проблему потерянных сегментов, отправляемых мобильным хостом. Если базовая станция замечает разрыв в порядковых номерах получаемых сегментов, она просит повторить недостающие байты. Благодаря этим двум исправлениям участок беспроводной связи стал более надежным в обоих направлениях, причем для этого не потребовалось изменять семантику протокола TCP и даже информировать о возникающих проблемах отправителя.

Хотя протокол UDP и не страдает от тех же самых проблем, что и протокол TCP, беспроводная связь также представляет для него некоторые трудности. Основная сложность состоит в том, что программы, использующие протокол UDP, ожидают от него высокой надежности. Они знают, что гарантии не предоставляются, но, тем не менее, надеются, что протокол UDP почти совершенен. В условиях беспроводной связи до совершенства будет очень далеко. Программы, способные справляться с потерей UDP-сообщений (но довольно значимой ценой), попадая из окружения, в котором сообщения теоретически могут теряться, но теряются очень редко, в окружение, в котором они теряются постоянно, могут очень сильно потерять в производительности.

Беспроводная связь затрагивает также аспекты, отличные от производительности. Например, как мобильному хосту найти локальный принтер, чтобы не связываться со своим домашним принтером? В чем-то близкой является проблема получения веб-страницы для локальной соты, даже если ее имя неизвестно. Кроме того, веб-дизайнеры обычно рассчитывают на большую пропускную способность сети. Они размещают на каждой странице гигантский логотип, для передачи которого по беспроводному каналу потребуется 10 с при каждом обращении к этой странице, что чрезвычайно раздражает пользователей.

Беспроводные сети распространяются все шире, поэтому проблемы работы в них протокола TCP становятся все более острыми. Дополнительную информацию, касающуюся данных вопросов, можно найти в (Barakat и др., 2000; Ghani и Dixit, 1999; Huston, 2001; Xylomenos и др., 2001).

Транзакционный TCP

Мы уже рассматривали в этой главе, как осуществляется удаленный вызов процедуры в клиент-серверных системах. Если запрос и ответ достаточно малы, чтобы поместиться в один пакет, а операция идемпотентна, то можно смело использовать UDP. Однако если эти условия не выполняются, применение протокола UDP оказывается менее привлекательным. Например, если ответ может быть довольно объемным, то нужен механизм для последовательного выстраивания частей сообщения и повторной передачи потерянных пакетов. Получается, что речь идет о том, что приложению следует заново изобрести TCP.

Это, понятное дело, не очень привлекательная перспектива. Однако и TCP сам по себе в данном случае не кажется слишком привлекательным. Проблема заключается, прежде всего, в эффективности. На рис. 6.33, *а* показана обычная последовательность пакетов, необходимая для удаленного вызова процедуры. В лучшем случае потребуется обменяться девятью пакетами. Вот они:

1. Клиент посылает пакет *SYN* для установки соединения.
2. Сервер посылает пакет *ACK* для подтверждения приема *SYN*.
3. Клиент выполняет «тройное рукопожатие».
4. Клиент посылает, собственно, свой *запрос*.
5. Клиент посылает пакет *FIN*, сигнализирующий об окончании передачи.
6. Сервер подтверждает запрос и *FIN*.
7. Сервер посылает *ответ* клиенту.
8. Сервер посылает пакет *FIN*, сообщающий о том, что он также закончил передачу.
9. Клиент подтверждает *FIN* сервера.

И это в лучшем случае! Если же обстоятельства складываются не очень удачно, то запрос и *FIN* клиента подтверждаются раздельно. Раздельно могут подтверждаться и ответ и *FIN* сервера.

Само собой, возникает вопрос: нельзя ли объединить эффективность выполнения RPC с помощью UDP (всего два сообщения) с надежностью, которую гарантирует TCP? Ответ: отчасти. Можно попытаться применить экспериментальный вариант протокола TCP, называемый **транзакционным TCP (T/TCP**, Transactional TCP). Протокол T/TCP описан в RFC 1379 и 1644.

Центральная идея протокола состоит в том, чтобы немного изменить стандартную последовательность действий, выполняемых при установке соединения, и предоставить возможность передачи данных непосредственно во время установки соединения. Работа T/TCP показана на рис. 6.33, *б*. В первом пакете клиента содержится бит *SYN*, собственно запрос и *FIN*. В сущности, клиент говорит: «Я хочу установить соединение, вот данные для передачи, и на этом я считаю свою миссию выполненной».

Получив запрос, сервер ищет или вычисляет содержимое ответа и выбирает способ ответа. Если ответ укладывается в один пакет, он будет послан так, как показано на рис. 6.33, *б*, то есть сервер как бы говорит: «Подтверждаю получение

вашего *FIN*, вот мой ответ на запрос, и на этом я считаю свою миссию выполненной». Клиент подтверждает *FIN* сервера, и на этом работа протокола заканчивается. Обратите внимание: на весь процесс потребовалось всего три сообщения.

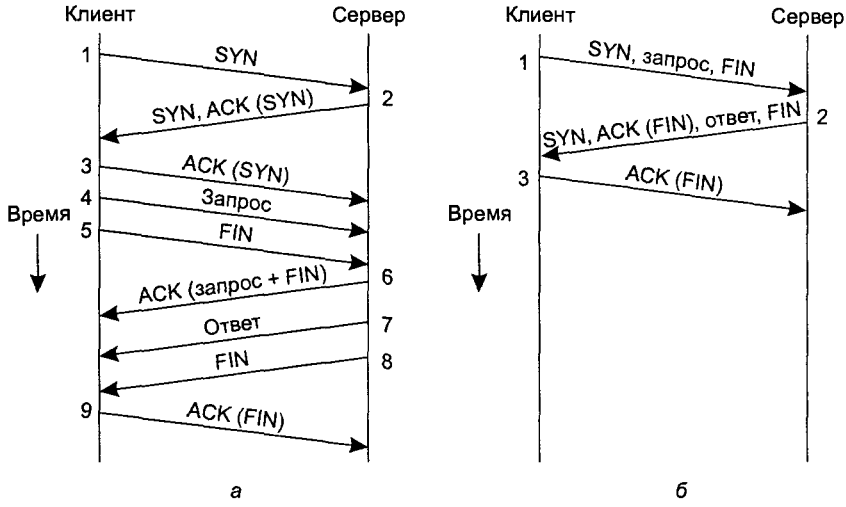


Рис. 6.33. Удаленный вызов процедуры с помощью обычного TCP (*а*); удаленный вызов процедуры с помощью T/TCP (*б*)

Однако если ответ не укладывается в один пакет, сервер может и не устанавливать бит *FIN* в первом же пакете. Он посылает столько пакетов, сколько нужно, прежде чем закрыть соединение в данном направлении.

Стоит отметить, что кроме T/TCP известны и другие вариации на тему TCP. Одним из таких протоколов является **протокол передачи с контролем потока (SCTP**, Stream Control Transmission Protocol). Среди его свойств можно выделить сохранение границ сообщений, несколько режимов доставки (например, неупорядоченная), множественную адресацию (дублирование адресатов), селективные подтверждения (Stewart и Metz, 2001). Тем не менее, когда кто-то предлагает улучшения того, что и так прекрасно работает в течение долгих лет, обычно возникают споры между сторонниками двух диаметрально противоположных принципов: «Все для расширения функциональности» и «Не трогать то, что пока еще не сломалось».

Вопросы производительности

Вопросы производительности играют важную роль в компьютерных сетях. Когда сотни или тысячи компьютеров соединены вместе, их взаимодействие становится очень сложным и может привести к непредсказуемым последствиям. Часто эта сложность приводит к низкой производительности, причины которой довольно трудно определить. В следующих разделах мы рассмотрим многие вопросы, свя-

занные с производительностью сетей, определим круг существующих проблем и обсудим методы их разрешения.

К сожалению, понимание производительности сетей — это скорее искусство, чем наука. Теоретическая база, допускающая хоть какое-то практическое применение, крайне скудна. Лучшее, что мы можем сделать, — это представить несколько практических методов, полученных в результате долгих экспериментов, а также привести несколько реально действующих примеров. Мы намеренно отложили эту дискуссию до того момента, когда будет изучен транспортный уровень в сетях TCP, чтобы иметь возможность иллюстрировать некоторые места примерами из TCP.

Вопросы производительности возникают не только на транспортном уровне. С некоторыми из них мы уже сталкивались в предыдущей главе. Тем не менее, сетевой уровень в основном занят вопросами маршрутизации и борьбы с перегрузкой. Более глобальные вопросы, касающиеся производительности всей системы в целом, оказываются прерогативой транспортного уровня, поэтому они будут рассматриваться именно в этой главе.

В следующих разделах мы рассмотрим следующие пять аспектов производительности сети.

1. Причины снижения производительности.
2. Измерение производительности сети.
3. Проектирование производительных систем.
4. Быстрая обработка TPDU-модулей.
5. Протоколы для будущих высокопроизводительных сетей.

Нам потребуется ввести название для единиц данных, которыми обмениваются транспортные сущности. Термин «сегмент», применяемый в протоколе TCP, в лучшем случае запутывает и никогда не используется в этом смысле за пределами мира TCP. Соответствующие ATM-термины — CS-PDU (протокольная единица обмена подуровня конвергенции), SAR-PDU и CPCS-PDU — применяются только в сетях ATM. Термин «пакет» относится к сетевому уровню, а сообщения применяются на прикладном уровне. За отсутствием стандартного термина мы будем продолжать называть единицы данных, которыми обмениваются транспортные сущности, TPDU-модулями. Когда будет иметься в виду TPDU-модуль вместе с пакетом, мы будем использовать в качестве обобщающего термина понятие «пакет», например: «Центральный процессор должен быть достаточно быстрым, чтобы успевать обрабатывать приходящие пакеты в режиме реального времени». При этом будет иметься в виду как пакет сетевого уровня, так и помещенный в него TPDU-модуль.

Причины снижения производительности компьютерных сетей

Некоторые виды снижения производительности вызваны временным отсутствием свободных ресурсов. Если на маршрутизатор вдруг прибывает трафик больше,

чем он способен обработать, образуется затор, и производительность резко падает. Вопросы перегрузки подробно рассматривались в предыдущей главе.

Производительность также снижается, если возникает структурный дисбаланс ресурсов. Например, если гигабитная линия связи присоединена к компьютеру с низкой производительностью, то несчастный центральный процессор не сможет достаточно быстро обрабатывать приходящие пакеты, что приведет к потере некоторых пакетов. Эти пакеты рано или поздно будут переданы повторно, что приведет к увеличению задержек, непроизводительному использованию пропускной способности и снижению общей производительности.

Перегрузка может также возникать синхронно. Например, если TPDU-модуль содержит неверный параметр (например, номер порта назначения), во многих случаях получатель заботливо пошлет обратно сообщение об ошибке. Теперь рассмотрим, что случится, если неверный TPDU-модуль будет разослан широковещательным способом 10 000 машин. Каждая машина может послать обратно сообщение об ошибке. Образовавшийся в результате **широковещательный шторм** может надолго остановить нормальную работу сети. Протокол UDP страдал от подобной проблемы, пока не было решено, что хосты должны воздерживаться от отправки сообщений об ошибке в ответ на широковещательные TPDU-модули UDP.

Второй пример синхронной перегрузки может быть вызван временным отключением электроэнергии. Когда питание снова включается, все машины одновременно обращаются к своей постоянной памяти и начинают перезагружаться. Типичная последовательность загрузки может требовать обращения к какому-нибудь DHCP-серверу (сервер динамической конфигурации хоста), чтобы узнать свой истинный адрес, а затем к файловому серверу, чтобы получить копию операционной системы. Если сотни машин обратятся к серверу одновременно, он не сможет обслужить сразу всех.

Даже при отсутствии синхронной перегрузки и при достаточном количестве ресурсов производительность может снижаться из-за неверных системных настроек. Например, у машины может быть мощный процессор и много памяти, но недостаточно памяти выделено под буфер. В этом случае буфер будет переполняться, и часть TPDU-модулей потеряется. Аналогично, если процессу обработки поступающих пакетов дан недостаточно высокий приоритет, некоторые TPDU-модули могут быть потеряны.

Также на производительность могут повлиять неверно установленные значения таймеров ожидания. Когда посылается TPDU-модуль, обычно включается таймер — на случай, если этот модуль потеряется. Выбор слишком короткого интервала ожидания подтверждения приведет к излишним повторным передачам TPDU-модулей. Если же интервал ожидания сделать слишком большим, это приведет к увеличению задержки в случае потери TPDU-модуля. К настраиваемым параметрам также относятся интервал ожидания попутного модуля данных для отправки подтверждения и количество попыток повторной передачи в случае отсутствия подтверждений.

С появлением гигабитных сетей появились и новые проблемы. Рассмотрим, например, процесс отправки 64 Кбайт данных из Сан-Диего в Бостон при раз-

мере буфера получателя 64 Кбайт. Пусть скорость передачи данных в линии составляет 1 Гбит/с, а время прохождения сигнала в одну сторону, ограниченное скоростью света в стекле, равно 20 мс. Вначале ($t = 0$), как показано на рис. 6.34, *а*, канал пуст. Только 500 мкс спустя все TPDU-модули попадут в канал (рис. 6.34, *б*). Первый TPDU-модуль оказывается где-то в окрестностях Броли, все еще в Южной Калифорнии. Тем не менее, передатчик уже должен остановиться, пока он не получит в ответ новую информацию об окне.

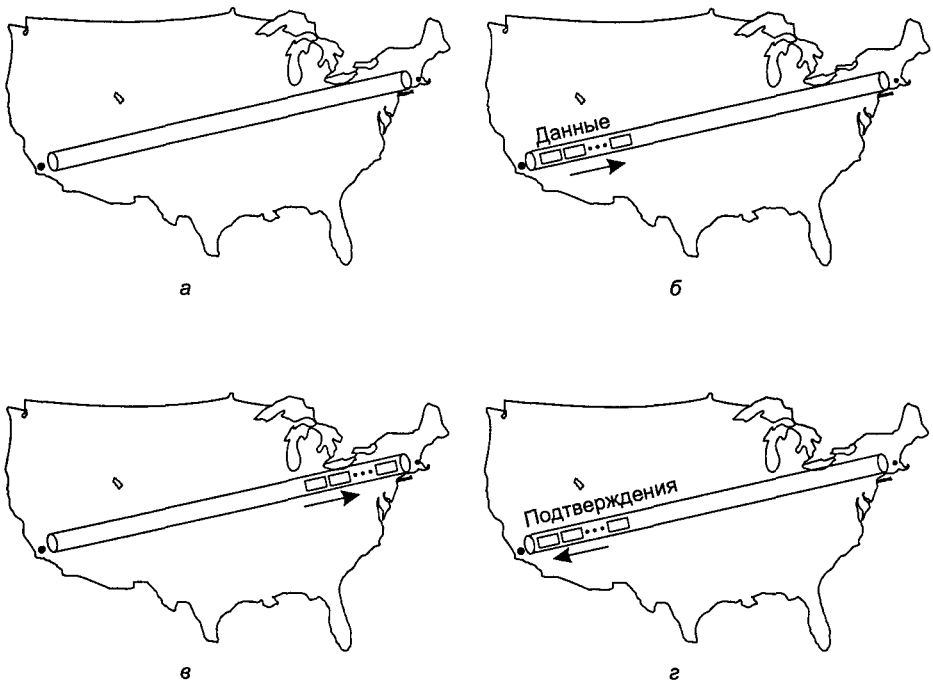


Рис. 6.34. Передача половины мегабита из Сан-Диего в Бостон: в момент времени $t = 0$ (*а*); через 500 мкс (*б*); через 20 мс (*в*); через 40 мс (*г*)

Через 20 мс первый TPDU-модуль, как показано на рис. 6.34, *в*, достигнет Бостона, и в ответ будет передано подтверждение. Наконец, через 40 мс после начала операции первое подтверждение возвращается к отправителю, после чего передается следующая порция данных. Поскольку линия передачи использовалась всего 0,5 мс из 40 мс, эффективность ее использования составит около 1,25 %. Такая ситуация является типичной для работы старых протоколов по гигабитным линиям.

При анализе производительности сетей полезно обращать внимание на **произведение пропускной способности и времени задержки**. Пропускная способность канала (в битах в секунду) умножается на время прохождения сигнала в оба конца (в секундах). В результате получается емкость канала в битах.

В примере на рис. 6.34 произведение пропускной способности и времени задержки равно 40 млн бит. Другими словами, отправитель к моменту получения

ответа успеет переслать 40 млн бит, если будет передавать с максимальной скоростью. Столько бит потребуется, чтобы заполнить канал в обоих направлениях. Таким образом, порция данных в полмиллиона бит составляет всего 1,25 % емкости канала, что и выражается в 1,25-процентной эффективности использования канала.

Отсюда следует, что для эффективного использования канала размер окна получателя должен быть, по меньшей мере, равен произведению пропускной способности и времени задержки, а лучше — превосходить его, так как получатель может сразу и не ответить. Для трансконтинентальной гигабитной линии каждому соединению потребуется, по меньшей мере, по 5 Мбайт.

Если даже при пересылке одного мегабита эффективность использования канала оказывается столь ужасной, представьте, что происходит в случае передачи нескольких сот байт при вызове удаленной процедуры. Если не найти этой линии еще какого-либо применения, пока клиент ожидает ответа, гигабитная линия будет ничем не лучше мегабитной, только значительно более дорогой.

Еще одна проблема производительности связана с приложениями типа видео и аудио, для которых временные параметры являются критическими. Обеспечить малое среднее время передачи здесь недостаточно. Требуется также обеспечить небольшое значение его среднеквадратичного отклонения. Для достижения обеих целей требуется немало инженерных усилий.

Измерение производительности сети

Когда качество работы сети оказывается не слишком хорошим, ее пользователи часто жалуются сетевым операторам, требуя усовершенствований. Чтобы улучшить производительность сети, операторы должны сначала точно определить, в чем суть проблемы. Чтобы выяснить текущее состояние сети, операторы должны произвести измерения. В данном разделе мы рассмотрим вопрос измерения производительности сети. Приводимое ниже обсуждение основано на работе Могола (Mogul, 1993).

Основной цикл работ по совершенствованию производительности сети включает следующие этапы.

1. Измерение наиболее важных сетевых параметров и производительности сети.
2. Попытка понять, что происходит.
3. Изменение одного из параметров.

Эти шаги повторяются до тех пор, пока производительность не увеличится достаточно или пока не станет ясно, что этими методами производительность уже больше не увеличить.

Измерения могут быть произведены разными способами и во многих местах (как физически, так и в стеке протоколов). Наиболее распространенный тип измерений представляет собой включение таймера при начале какой-либо активности и измерение продолжительности этой активности. Например, одним из ключевых измерений является измерение времени, необходимого для получения отправителем подтверждения в ответ на отправку TPDU-модуля. Другие изме-

рения производятся при помощи счетчиков, в которых учитывается частота некоторых событий (например, количество потерянных TPDU-модулей). Наконец, часто измеряются такие количественные показатели как число байтов, обработанных за определенный временной интервал.

Процедура измерения производительности сети и других параметров содержит множество подводных камней. Далее мы перечислим некоторые из них. Следует тщательно избегать подобных ошибок при любых попытках измерить производительность сети.

Убедитесь, что выборка достаточно велика

Не следует ограничиваться единственным измерением какого-нибудь параметра — например, времени, необходимого для передачи одного TPDU-модуля. Повторите измерение, скажем, миллион раз и вычислите среднее значение. Чем больше будет выборка, тем выше окажется точность оценки среднего значения и его среднеквадратичного отклонения. Погрешность может быть вычислена при помощи стандартных формул статистики.

Убедитесь, что выборка является репрезентативной

Следует повторить всю последовательность миллиона измерений параметров в различное время суток и в разные дни недели, чтобы заметить влияние различной загруженности системы на измеряемые параметры. Так, измерение перегрузки вряд ли принесет пользу, если эти измерения производить, когда перегрузки нет. Иногда результаты могут показаться на первый взгляд странными, как, например, наличие серьезных заторов в сети в 10, 11, 13 и 14 часов, но их отсутствие в полдень (когда все пользователи обедают).

Используя часы с грубой шкалой, будьте внимательны

Компьютерные часы работают, добавляя единицу к некоему счетчику через равные интервалы времени. Например, миллисекундный таймер увеличивает на единицу значение счетчика раз в 1 мс. Применение такого таймера для измерения длительности события, занимающего менее 1 мс, возможно, но требует осторожности.

Например, чтобы измерить время, необходимое для передачи одного TPDU-модуля, следует считывать показания системных часов (скажем, в миллисекундах) при входе в программу транспортного уровня и выходе из нее. Если время, требуемое для передачи одного TPDU-модуля, равно 300 мкс, то измеряемая величина будет равна либо 0, либо 1 мс. Однако если повторить измерения миллион раз, сложить все значения и разделить на миллион, то полученное среднее значение будет отличаться от истинного значения менее чем на 1 мкс.

Убедитесь, что во время ваших тестов не происходит ничего неожиданного

Если проводить измерения в университетской системе в день сдачи главного лабораторного проекта, то полученные результаты могут сильно отличаться от результатов измерений, произведенных на следующий день. Аналогично, если в то

время, когда вы производите тестирование сети, какой-нибудь исследователь решит устроить в сети видеоконференцию, вы также можете получить искаженные результаты. Лучше всего запускать тесты на пустой системе, создавая всю нагрузку самому. Хотя и в этом случае можно ошибиться. Вы можете предполагать, что никто не пользуется сетью в 3 часа ночи, но может оказаться, что именно в это время программа автоматического резервного копирования начинает свою работу по архивации всех жестких дисков на магнитную ленту. Кроме того, именно в это время пользователи, находящиеся в другой временной зоне на другой стороне Земного шара, могут создавать довольно сильный трафик, просматривая ваши замечательные веб-страницы.

Кэширование может сильно исказить ваши измерения

Чтобы измерить время операции по передаче файла, самый очевидный путь состоит в том, чтобы открыть большой файл, прочитать его целиком и закрыть, измерив при этом время, затраченное на всю последовательность операций. Затем можно повторить измерение много раз, чтобы получить точное значение средней величины. Беда заключается в том, что система может, считав файл по сети всего один раз, запомнить его в локальном кэше. Таким образом, правильным будет только первое измерение, а при остальных операциях обращения к сети не будет. Результат такого многократного измерения будет бесполезен (если только вы не измеряете производительность кэша).

Часто можно обмануть алгоритм кэширования, просто заставляя переполняться кэш. Например, если размер кэша составляет 10 Мбайт, в тестовый цикл можно включить поочередное открытие, чтение и закрытие двух 10-мегабайтных файлов, пытаясь заставить систему каждый раз читать файлы по сети. Тем не менее, следует быть уверенным в том, что вы понимаете, как работает алгоритм кэширования.

Буферизация пакетов может производить аналогичный эффект. Одна популярная TCP/IP-программа измерения производительности славилась тем, что сообщала, что протокол UDP может достичь производительности, значительно превышающей максимально допустимую для данной физической линии. Как это происходило? Обращение к UDP обычно возвращает управление сразу, как только сообщение принимается ядром системы и добавляется в очередь на передачу. При достаточном размере буфера время выполнения 1000 обращений к UDP не означает, что за это время все данные были переданы в линию. Большая часть их все еще находится в буфере ядра.

Следует хорошо понимать, что измеряется

Когда вы измеряете время чтения удаленного файла, результаты ваших измерений зависят от сети, операционной системы клиента и сервера, аппаратного интерфейса сетевых карт, их драйверов и других факторов. Если все делать внимательно, то в конечном итоге вы получите значение времени передачи файла для данной конфигурации. Если вы ставите перед собой цель настроить именно эту конкретную конфигурацию, то беспокоиться не о чем.

Однако если вы проводите сходные измерения на трех различных системах, чтобы выбрать, какую сетевую карту купить, то ваши результаты могут оказаться никуда не годными из-за того, что какой-нибудь сетевой драйвер окажется неудачным и использующим лишь 10 % производительности сетевой карты.

Будьте осторожны с экстраполяцией результатов

Предположим, вы что-нибудь измеряете (например, время ответа удаленного хоста), моделируя нагрузку сети от 0 (простой) до 0,4 (40 % мощности), как показано жирной линией на рис. 6.35. Может оказаться соблазнительным линейно экстраполировать полученную кривую (пунктир). Однако в действительности многие параметры в теории массового обслуживания содержат в качестве сомножителя $1/(1 - \rho)$, где ρ — нагрузка, поэтому истинная кривая зависимости будет больше походить на гиперболу, показанную штриховой линией.

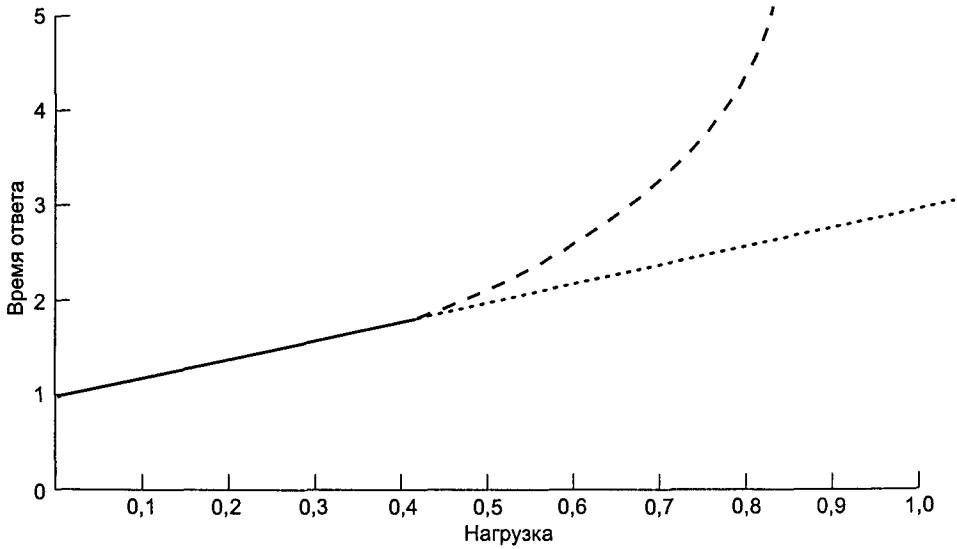


Рис. 6.35. Зависимость времени ответа от нагрузки

Проектирование производительных систем

Измерения и настройки часто позволяют значительно улучшить производительность сети, однако они никогда не заменят хорошо разработанного проекта. Плохо спроектированная сеть может быть усовершенствована только до определенного уровня. Для дальнейшего увеличения ее производительности ее потребуется переделать с нуля.

В данном разделе мы рассмотрим несколько эмпирических правил, основанных на опыте работы со многими сетями. Эти правила касаются не только устройства сети, но и системных аспектов, так как программное обеспечение и операционные системы часто оказываются важнее, чем маршрутизаторы и интерфейсные

карты. Большинство этих идей известны разработчикам сетей уже много лет и передаются из уст в уста от поколения к поколению. Впервые они были открыто записаны Моголом (Mogul, 1993). Наше повествование во многом пересекается с его книгой. Другим источником по этой же теме является (Metcalfe, 1993).

Правило 1: скорость центрального процессора важнее скорости сети

Длительные эксперименты показали, что почти во всех сетях накладные расходы операционной системы и протокола составляют основное время задержки сетевой операции. Например, теоретически минимальное время вызова удаленной процедуры (RPC, Remote Procedure Call) в сети Ethernet составляет 102 мкс, что соответствует минимальному запросу (64 байта), на который приходит минимальный (64-байтовый) ответ. На практике значительным достижением считается хотя бы какое-нибудь снижение накладных расходов, возникающих за счет программного обеспечения при вызове удаленной процедуры.

Аналогично, при работе с гигабитной линией основная задача заключается в достаточно быстрой передаче битов из буфера пользователя в линию, а также в том, чтобы получатель смог обработать их с той скоростью, с которой они приходят. Короче говоря, удвоение производительности процессора нередко может привести к почти удвоению пропускной способности канала. Удвоение же пропускной способности линии часто не дает никакого эффекта, поскольку узким местом обычно являются хосты.

Правило 2: уменьшайте число пакетов, чтобы уменьшить программные накладные расходы

Обработка каждого TPDU-модуля подразумевает определенное количество накладных расходов на каждый модуль (то есть на обработку его заголовка) и определенные затраты при обработке каждого байта (например, при подсчете контрольной суммы). При отправке 1 млн байт побайтовые затраты времени процессора на обработку не зависят от размера TPDU-модуля. Однако при использовании 128-байтовых TPDU-модулей затраты на обработку заголовков будут в 32 раза выше, чем для TPDU-модулей размером 4 Кбайт. И эти затраты увеличиваются очень быстро.

Помимо накладных расходов на обработку заголовков TPDU-модулей, следует рассмотреть накладные расходы нижних уровней. Прибытие каждого пакета вызывает прерывание. В современных RISC-процессорах каждое прерывание нарушает работу процессорного конвейера, снижает эффективность работы кэша, требует изменения контекста управления памятью и сохранения в стеке значительного числа регистров процессора. Таким образом, уменьшение на n числа посылаемых TPDU-модулей дает снижение числа прерываний и накладных расходов в целом в n раз.

Данное наблюдение служит аргументом в пользу сбора значительного количества данных перед их отправкой с целью снизить количество прерываний у по-

лучателя. Алгоритм Нагля и предложенный Кларком метод избавления от синдрома глупого окна действуют именно в этом направлении.

Правило 3: минимизируйте количество переключений контекста

Переключения контекста (например, из режима ядра в режим пользователя) обладают рядом неприятных свойств, в этом они сходны с прерываниями. Самое неприятное — потеря содержимого кэша. Количество переключений контекста может быть снижено при помощи библиотечной процедуры, посылающей данные во внутренний буфер до тех пор, пока их не наберется достаточное количество. Аналогично, на получающей стороне небольшие TPDU-модули следует собирать вместе и передавать пользователю за один раз, минимизируя количество переключений контекста.

В лучшем случае прибывший пакет вызывает одно переключение контекста из текущего пользовательского процесса в режим ядра, а затем еще одно переключение контекста при передаче управления принимающему процессу и предоставлении ему прибывших данных. К сожалению, во многих операционных системах происходит еще одно переключение контекста. Например, если сетевой менеджер работает в виде отдельного процесса в пространстве пользователя, поступление пакета вызывает передачу управления от процесса пользователя ядру, затем от ядра сетевому менеджеру, затем снова ядру и, наконец, от ядра получающему процессу. Эта последовательность переключений контекста показана на рис. 6.36. Все переключения контекста при получении каждого пакета сильно расходуют время центрального процессора и существенно снижают производительность сети.

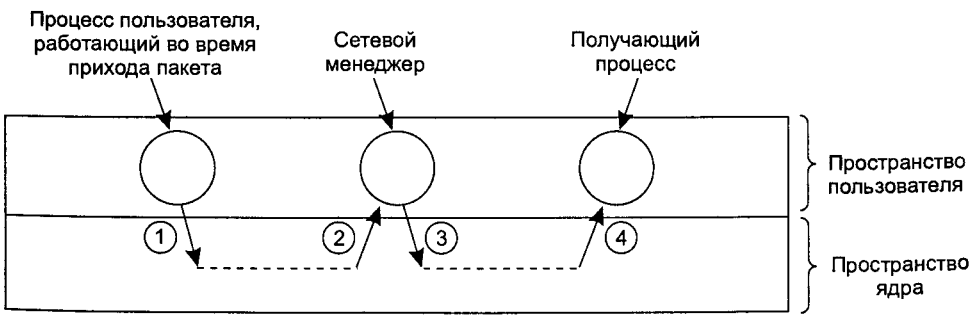


Рис. 6.36. Четыре переключения контекста для обработки одного пакета в сети, в которой сетевой менеджер находится в пространстве пользователя

Правило 4: минимизируйте число операций копирования

Еще больше времени процессора отнимается излишним копированием пакета. Часто полученный пакет копируется три или четыре раза, прежде чем содержащийся в нем TPDU-модуль доставляется по назначению. Сначала пакет принимается сетевым интерфейсом в специальный аппаратный буфер, расположенный

на сетевой карте. Из аппаратного буфера пакет копируется в системный буфер ядра, откуда он копируется в буфер сетевого уровня, а затем — в буфер транспортного уровня и, наконец, доставляется получающему приложению.

Грамотно разработанные операционные системы копируют по одному машинному слову за такт процессора, но нередки случаи, когда копирование одного слова требует пять инструкций процессора (считывание, запись, увеличение на единицу индексного регистра, проверка на конец данных и условный переход на начало цикла). Если на одно копирование 32-битного слова требуется пять инструкций процессора, то при трех операциях копирования каждого пакета на каждый скопированный байт приходится 15/4, или почти 4 команды. На машине с производительностью в 500 млн инструкций в секунду (500 MIPS) каждая команда выполняется за 2 нс, следовательно, копирование каждого байта будет производиться процессором в течение 8 нс (около 1 нс на бит). Значит, максимальная скорость ограничивается 1 Гбит/с. С учетом накладных расходов на обработку заголовка и переключения контекстов, возможно, удастся достичь скорости в 500 Мбит/с, а ведь мы еще не учли обработку самих данных. Очевидно, что о поддержке 10-гигабитной линии не может быть и речи.

В действительности поддержка линии со скоростью в 500 Мбит/с также может оказаться абсолютно невозможной. В приведенных расчетах мы предполагали, что машина с производительностью 500 MIPS может выполнить 500 млн любых инструкций в секунду. На самом деле, машина может работать с такой скоростью, только если она не обращается к памяти. Операции с памятью часто оказываются в десятки раз медленнее, чем операции с использованием только внутренних регистров (на выполнение инструкции расходуется около 20 нс). Если 20 % инструкций связано с обращением к памяти (то есть имеются потери кэшируемых данных) — а это вполне вероятная цифра при обработке входящих пакетов, — среднее время выполнения инструкции будет равно 5,6 нс ($0{,}8 \cdot 2 + 0{,}2 \cdot 20$). Предполагая, что на обработку байта требуются 4 инструкции, нам понадобится 22,4 нс/байт (или около 2,8 нс/бит), что в результате дает суммарную скорость около 357 Мбит/с. Допустим, половина этой производительности уйдет на обработку заголовков, тогда остается 178 Мбит/с. Обратите внимание на то, что аппаратные улучшения здесь не помогут. Проблема состоит в слишком большом числе операций копирования, выполняемых операционной системой.

Правило 5: можно купить более высокую пропускную способность, но не низкую задержку

Следующие три правила относятся не к протоколам, а к линиям связи. Первое правило утверждает, что если вам нужна более высокая пропускная способность, вы можете просто купить ее. Если проложить второй оптоволоконный кабель параллельно первому, пропускная способность удвоится, но время задержки от этого меньше не станет. Чтобы снизить задержку, следует улучшить программное обеспечение протоколов, операционную систему или сетевой интерфейс. И даже это не поможет, если загвоздка состоит во времени передачи по линии.

Правило 6: лучше избегать перегрузки, чем бороться с уже возникшей перегрузкой

Старая пословица, гласящая, что профилактика лучше лечения, справедлива и в деле борьбы с перегрузками в сетях. Когда в сети образуется затор, пакеты теряются, пропускная способность растрачивается впустую, увеличиваются задержки и т. п. Процесс восстановления после перегрузки требует времени и терпения. Гораздо более эффективной стратегией является предотвращение перегрузки, напоминающее прививку от болезни — это несколько неприятно, зато избавляет от возможных больших неприятностей.

Правило 7: избегайте тайм-аутов

Таймеры необходимы в сетях, но их следует применять умеренно, и следует минимизировать количество тайм-аутов. Когда срабатывает таймер, обычно повторяется какое-либо действие. Если повтор этого действия необходим, его следует повторить, однако повторение действия без особой необходимости является расточительным.

Чтобы избегать излишней работы, следует устанавливать период ожидания с небольшим запасом. Таймер, срабатывающий слишком поздно, несколько увеличивает задержку в (маловероятном) случае потери TPDU-модуля. Преждевременно срабатывающий таймер растрачивает попусту время процессора, пропускную способность и напрасно увеличивает нагрузку на, возможно, десятки маршрутизаторов.

Быстрая обработка TPDU-модулей

Мораль приведенной истории состоит в том, что основным препятствием на пути ускорения сетей является программное обеспечение протоколов. В данном разделе мы рассмотрим некоторые способы ускорения этих программ. Дополнительные сведения по этой теме см. в (Clark и др., 1989; Chase и др., 2001).

Накладные расходы по обработке TPDU-модулей состоят из двух компонентов: затрат по обработке заголовка и затрат по обработке каждого байта. Следует вести наступление сразу на обоих направлениях. Ключ к быстрой обработке TPDU-модулей лежит в выделении нормального случая (односторонней передачи данных) и отдельной обработке этого случая. Хотя для перехода в состояние *ESTABLISHED* требуется передача последовательности специальных TPDU-модулей, но как только это состояние достигнуто, обработка TPDU-модулей не вызывает затруднений, пока одна из сторон не начнет закрывать соединение.

Начнем с рассмотрения посылающей стороны, находящейся в состоянии *ESTABLISHED*, когда должны отправляться данные. Для простоты мы предположим, что транспортная сущность расположена в ядре, хотя те же самые идеи применимы и в случае, когда она представляет собой процесс, находящийся в пространстве пользователя, или набор библиотечных процедур в посылающем процессе. На рис. 6.37 отправляющий процесс эмулирует прерывание, выполняя примитив `SEND`, и передает управление ядру. Прежде всего, транспортная сущ-

ность проверяет, находится ли она в нормальном состоянии, то есть таком, когда установлено состояние *ESTABLISHED*, ни одна сторона не пытается закрыть соединение, посылается стандартный полный TPDU-модуль и у получателя достаточный размер окна. Если все эти условия выполнены, то никаких дополнительных проверок не требуется, и алгоритм транспортной сущности может выбрать быстрый путь. В большинстве случаев именно так и происходит.

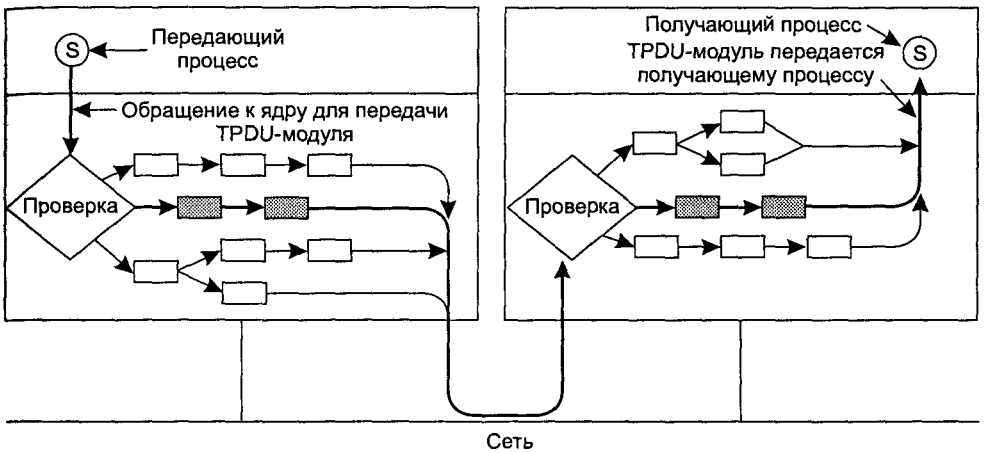
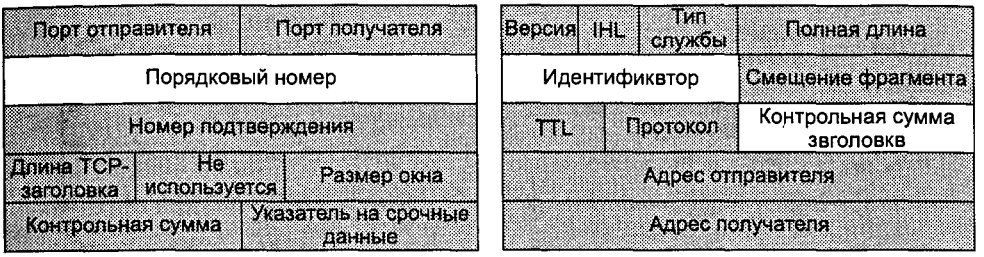


Рис. 6.37. Быстрый путь от отправителя до получателя показан жирной линией. Шаги обработки вдоль этого пути показаны затененными прямоугольниками

В нормальной ситуации заголовки соседних TPDU-модулей почти одинаковы. Чтобы использовать этот факт, транспортная сущность сохраняет в своем буфере прототип заголовка. В начале быстрого пути он как можно быстрее пословно копируется в буфер нового заголовка. Затем поверх перезаписываются все отличающиеся поля. Часто эти поля легко выводятся из переменных состояния — например, следующий порядковый номер TPDU-модуля. Затем указатель на полный TPDU-модуль и указатель на данные пользователя передаются сетевому уровню. Здесь может быть применена та же стратегия (на рис. 6.37 это не показано). Наконец, сетевой уровень передает полученный в результате пакет уровню передачи данных для отправки.

Чтобы увидеть, как работает этот принцип на практике, рассмотрим случай TCP/IP. На рис. 6.38, *а* изображен TCP-заголовок. Поля, одинаковые для заголовков последующих TPDU-модулей в однонаправленном потоке, затенены. Все, что нужно сделать передающей транспортной сущности, это скопировать пять слов заголовка-прототипа в выходной буфер, заполнить поле порядкового номера (скопировав одно слово), сосчитать контрольную сумму и увеличить на единицу значение переменной, хранящей текущий порядковый номер. Затем она может передать заголовок и данные специальной IP-процедуре, предназначенной для отправки стандартного максимального TPDU-модуля. Затем IP-процедура копирует свой заголовок-прототип из пяти слов (см. рис. 6.38, *б*) в буфер, заполняет поле *Идентификатор* и вычисляет контрольную сумму заголовка. Теперь пакет готов к передаче.

Рассмотрим теперь быстрый путь обработки пакета получающей стороной на рис. 6.37. Первый шаг состоит в нахождении для пришедшего TPDU-модуля записи соединения. В протоколе TCP запись соединения может храниться в хэш-таблице, ключом к которой является какая-нибудь простая функция двух IP-адресов и двух портов. После обнаружения записи соединения следует проверить адреса и номера портов, чтобы убедиться, что найдена верная запись.



а *б*

Рис. 6.38. TCP-заголовок (*а*); IP-заголовок (*б*). В обоих случаях затененные поля взяты у прототипа без изменений

Процесс поиска нужной записи можно дополнительно ускорить, если установить указатель на последнюю использовавшуюся запись и сначала проверить ее. Кларк с соавторами книги, вышедшей в 1989 году, исследовал этот вопрос и пришел к выводу, что в этом случае доля успешных обращений превысит 90 %. Другие эвристические методы поиска описаны в (McKenney и Dove, 1992).

Затем TPDU-модуль проверяется на адекватность: соединение в состоянии *ESTABLISHED*, ни одна сторона не пытается его разорвать, TPDU-модуль является полным, специальные флаги не установлены, и порядковый номер соответствует ожидающемуся. Программа, выполняющая всю эту проверку, состоит из довольно значительного количества инструкций. Если все эти условия удовлетворяются, вызывается специальная процедура быстрого пути TCP-уровня.

Процедура быстрого пути обновляет запись соединения и копирует данные пользователю. Во время копирования она одновременно подсчитывает контрольную сумму, что уменьшает количество циклов обработки данных. Если контрольная сумма верна, запись соединения обновляется и отправляется подтверждение. Метод, реализованный в виде отдельной процедуры, сначала производящей быструю проверку заголовка, чтобы убедиться, что заголовок именно такой, какой ожидается, называется **предсказанием заголовка**. Он применяется в большинстве реализаций протокола TCP. Использование этого метода оптимизации вместе с остальными, описанными в данном разделе, позволяет протоколу TCP достичь 90 % от скорости локального копирования из памяти в память при условии, что сама сеть достаточно быстрая.

Две другие области, в которых возможны основные улучшения производительности, — это управление буферами и управление таймерами. Как уже было сказано, при управлении буферами следует пытаться избегать излишнего копирования. При управлении таймерами следует учитывать, что они почти никогда не срабатывают. Они предназначены для обработки нестандартного случая по-

терь TPDU-модулей, но большинство TPDU-модулей и их подтверждений прибывают успешно, и поэтому истечение периода ожидания является редким событием.

В программе таймеры обычно реализуются в виде связанного списка таймеров, отсортированного по времени срабатывания. Заглавный элемент списка содержит счетчик, хранящий число минимальных интервалов времени, оставшихся до истечения периода ожидания. В каждом последующем элементе списка содержится счетчик, указывающий, через какой интервал времени относительно предыдущего элемента списка истечет время ожидания данного таймера. Например, если таймеры сработают через 3, 10 и 12 интервалов времени, счетчики списка будут содержать значения 3, 7 и 2 соответственно.

При каждом импульсе сигнала времени часов счетчик головного элемента списка уменьшается на единицу. Когда значение счетчика достигает нуля, обрабатывается связанное с этим таймером событие, а головным элементом списка становится следующий элемент. При такой схеме добавление и удаление таймера является операцией, требующей затрат ресурсов, при этом время выполнения операции пропорционально длине списка.

Если максимальное значение таймера ограничено и известно заранее, то может быть применен более эффективный метод. Здесь можно использовать массив, называющийся **колесом времени** (рис. 6.39). Каждое гнездо соответствует одному импульсу сигнала времени. Текущее время, показанное на рисунке, — $T = 4$. Таймеры должны сработать через 3, 10 и 12 импульсов сигнала времени. Если новый таймер должен сработать через 7 тиков, то все, что нужно сделать, — задать значение указателя в гнезде 11 на новый список таймеров. Аналогично, если таймер, установленный на время $T + 10$, должен быть отключен, нужно всего лишь обнулить запись в гнезде 14. Обратите внимание на то, что массив на рис. 6.39 не может поддерживать таймеры за пределами $T + 15$.



Рис. 6.39. Колесо времени

При каждом импульсе сигнала времени часов указатель текущего времени перемещается по колесу времени вперед (циклично) на одно гнездо. Если гнездо, на которое он указывает, не нулевое, обрабатываются все таймеры списка, на который указывает гнездо. Многочисленные варианты основной идеи обсуждаются в (Varghese и Lauck, 1989).

Протоколы для гигабитных сетей

Появление гигабитных сетей приходится на начало 90-х годов. Сначала к ним пытались применить старые протоколы, но при этом сразу же возникло множество проблем. Некоторые из этих проблем, а также методы их решения в протоколах будущих, даже более быстрых, сетей будут обсуждаться в данном разделе.

Первая проблема заключается в том, что многие протоколы используют 32-разрядные порядковые номера. В прежние годы, когда типичная скорость выделенных линий между маршрутизаторами равнялась 56 Кбит/с, хосту, который, бешено вращая глазами, постоянно выдавал бы данные в сеть, потребовалось бы больше недели на то, чтобы у него закончились порядковые номера. С точки зрения разработчиков TCP, 2^{32} считалось неплохим приближением к бесконечности, поскольку вероятность блуждания пакетов по сети в течение недели практически равна нулю. В Ethernet со скоростью 10 Мбит/с критическое время снизилось с одной недели до 57 минут. Это, конечно, гораздо меньше, но даже с таким интервалом еще можно иметь дело. Когда же Ethernet выдает в Интернет данные со скоростью 1 Гбит/с, порядковые номера закончатся примерно через 34 с. Это уже никуда не годится, поскольку максимальное время жизни пакета в Интернете равно 120 с. Внезапно оказалось, что 2^{32} совершенно не подходит в качестве значения, приближенного к бесконечности, поскольку стало очевидно, что отправителю, посылающему достаточно много пакетов, придется повторять их порядковые номера в то время, как старые пакеты все еще будут блуждать по сети. В RFC 1323 предлагается метод борьбы с этой ситуацией.

Проблема состоит в том, что многие разработчики протоколов просто предполагали, что время цикла пространства порядковых номеров значительно превосходит максимальное время жизни пакетов. Соответственно, в этих протоколах даже не рассматривалась сама возможность того, что старые пакеты еще могут находиться где-то в сети, когда порядковые номера опишут полный круг. В гигабитных сетях это предположение оказалось неверным.

Вторая проблема возникла, когда выяснилось, что скорости передачи данных увеличиваются значительно быстрее, чем скорости обработки данных. (Обращение к разработчикам компьютеров: идите и побейте разработчиков средств связи! Мы рассчитываем на вас.) В 70-е годы объединенная сеть ARPANET работала на скорости 56 Кбит/с и состояла из компьютеров с производительностью около 1 MIPS (1 млн инструкций в секунду). Использовались пакеты размером 1008 бит — таким образом, сеть ARPANET могла доставлять около 56 пакетов в секунду. Поскольку время передачи пакета составляло около 18 мс, хост мог затратить 18 000 инструкций на обработку одного пакета. Конечно, это полностью загрузило бы центральный процессор, однако можно было снизить это число до

9000 инструкций при половинной занятости процессора, предоставляя ему возможность выполнять всю необходимую работу.

Сравните эти цифры с цифрами для современных компьютеров, имеющих производительность 1000 MIPS и обменивающихся 1500-байтными пакетами по гигабитной линии. Пакеты могут прибывать с частотой свыше 80 000 пакетов в секунду, поэтому, если мы хотим зарезервировать половину мощности процессора для приложений, обработка пакета должна быть завершена за 6,25 мкс. За это время компьютер с производительностью 1000 млн инструкций в секунду может выполнить 6250 инструкций — лишь треть того, что было доступно хостам сети ARPANET. Более того, современные RISC-инструкции выполняют меньше действий, чем старые CISC-инструкции, так что проблема, на самом деле, оказывается еще тяжелее. Отсюда можно сделать следующий вывод: у протокола осталось меньше времени на обработку пакетов, поэтому протоколы должны стать проще.

Третья проблема состоит в том, что протокол с возвратом на n плохо работает в линиях с большим значением произведения пропускной способности на задержку. Рассмотрим, к примеру, линию длиной 4000 км, работающую со скоростью 1 Гбит/с. Время прохождения сигнала в оба конца равно 40 мс. За это время отправитель успевает передать 5 Мбайт. Если обнаруживается ошибка, потребуется 40 мс, чтобы оповестить об этом отправителя. При использовании протокола с возвратом на n отправителю потребуется повторять передачу не только поврежденного пакета, но также и до 5 Мбайт пакетов, переданных после поврежденного. Очевидно, что этот протокол использует ресурсы очень неэффективно.

Суть четвертой проблемы состоит в том, что гигабитные линии принципиально отличаются от мегабитных — в длинных гигабитных линиях главным ограничивающим фактором является не пропускная способность, а задержка. На рис. 6.40 изображена зависимость времени, требующегося для передачи файла размером 1 Мбит по линии длиной в 4000 км, от скорости передачи. На скоростях до 1 Мбит/с время передачи, в основном, зависит от скорости передачи данных. При скорости 1 Гбит/с задержка в 40 мс значительно превосходит 1 мс (время, требующееся для передачи битов по оптоволоконному кабелю). Дальнейшее увеличение скорости вообще не оказывает почти никакого действия на время передачи файла.

Изображенная на рис. 6.40 зависимость демонстрирует ограничения сетевых протоколов. Она показывает, что протоколы с ожиданием подтверждений, такие как удаленный вызов процедуры (RPC, Remote Procedure Call), имеют врожденное ограничение на производительность. Это ограничение связано со скоростью света. Никакие технологические прорывы в области оптики здесь не помогут (хотя могли бы помочь новые законы физики).

Пятая проблема, которую стоит упомянуть, связана не с протоколами или технологиями, а с появлением новых гигабитных мультимедийных приложений, для которых постоянство времени передачи пакета так же важно, как и само среднее значение задержки. Низкая, но постоянная скорость доставки часто предпочтительнее высокой, но непостоянной.

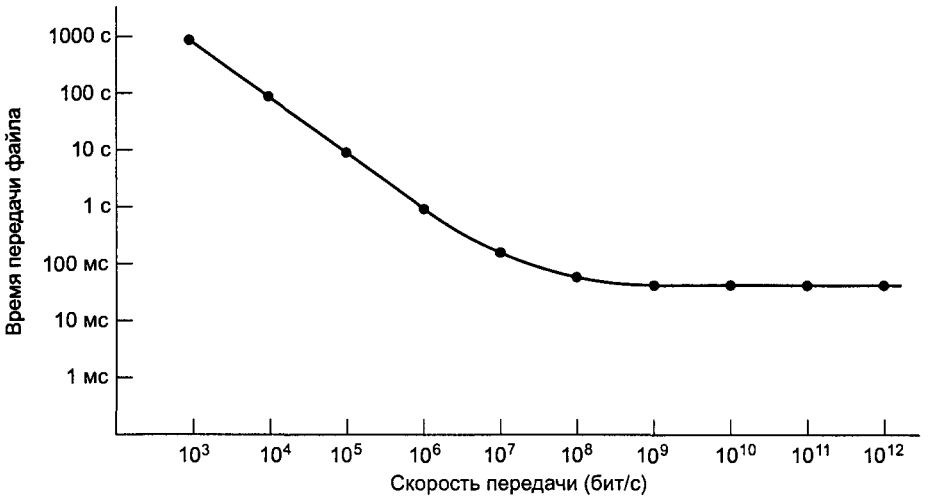


Рис. 6.40. Время передачи и подтверждения файла размером 1 Мбит по линии длиной 4000 км

Перейдем теперь к рассмотрению методов решения всего этого набора проблем. Сначала будет сделано несколько общих замечаний, затем мы рассмотрим механизмы протоколов, формат пакетов и программное обеспечение протоколов.

Главный принцип, который каждый разработчик гигабитных сетей должен выучить наизусть, звучит так:

Проектируя, стремись увеличить скорость, а не оптимизировать пропускную способность.

При разработке старых протоколов обычно ставилась задача минимизировать количество битов, переданных в линию, часто за счет использования полей малого размера и упаковывания нескольких полей вместе в один байт или слово. В настоящее время экономить пропускную способность смысла нет: ее более чем достаточно. Проблема заключается в обработке протоколами пакетов, поэтому при разработке новых протоколов минимизировать нужно именно время обработки. Разработчики IPv6, к счастью, хорошо понимают это.

Конечно, заманчивы попытки ускорить работу сетей, создав быстрые сетевые интерфейсы на аппаратном уровне. Сложность такого подхода состоит в том, что если только протокол не является чрезвычайно простым, аппаратный интерфейс представляет собой дополнительную плату с отдельным процессором и своей программой. Чтобы сетевой сопроцессор не был столь же дорогим, он, как и центральный процессор, часто представляет собой более медленную микросхему. В результате такого подхода быстрый центральный процессор ждет, пока медленный сопроцессор выполнит свою работу. Было бы неверным полагать, что у центрального процессора на время ожидания обязательно найдется другая работа. Более того, для общения двух процессоров общего назначения потребуются тщательно продуманные протоколы, обеспечивающие их корректную синхронизацию. Как правило, наилучший метод заключается в создании простых протоколов, чтобы всю работу мог выполнять центральный процессор.

Рассмотрим теперь вопрос обратной связи в высокоскоростных протоколах. Из-за относительно длинного цикла задержки желателен отказ от обратной связи: на оповещение отправителя уходит слишком много времени. Один пример обратной связи — управление скоростью передачи с помощью протокола скользящего окна. Чтобы избежать долгих задержек, связанных с передачей получателем информации о состоянии окна отправителю, лучше использовать протокол, основанный на скорости. В таком протоколе отправитель может посылать не быстрее, чем с некоторой скоростью, с которой получатель заранее согласился.

Второй пример обратной связи — алгоритм затяжного пуска, разработанный Джекобсоном. Этот алгоритм проводит многочисленные пробы, пытаясь определить пропускную способность сети. В высокоскоростных сетях на проведение пяти-шести проб для определения ответной реакции сети тратится огромное количество сетевых ресурсов. Более эффективная схема состоит в резервировании ресурсов отправителем, получателем и сетью в момент установки соединения. Кроме того, заблаговременное резервирование ресурсов позволяет несколько снизить эффект неравномерности доставки (джиттер). Короче говоря, повышение скоростей передачи в сетях неумолимо заставляет разработчиков выбирать ориентированную на соединение (или близкую к этому) схему работы сети.

Крайне важен в гигабитных сетях формат пакета. В заголовке должно быть как можно меньше полей — это позволит снизить время его обработки. Сами поля должны быть достаточно большими, чтобы нести в себе как можно больше полезной (служебной) информации. Кроме того, они не должны пересекать границы слов, тогда их будет проще обработать. В данном контексте под «достаточно большим» подразумевается такой размер, который исключает возникновение проблем зацикливания порядковых номеров при нахождении в сети старых пакетов, учитывает отсутствие у получателя возможности объявить реально доступный размер окна из-за слишком малого служебного поля размера окна, и т. д.

Кроме того, контрольные суммы заголовка и данных следует считать раздельно. Причин здесь две. Во-первых, чтобы предоставить возможность считать контрольную сумму только заголовка, без данных, что значительно быстрее. Во-вторых, чтобы иметь возможность убедиться в правильности заголовка, прежде чем начать копировать данные в пространство пользователя. Контрольную сумму данных лучше проверять во время копирования данных в пространство пользователя, но если заголовок неверен, то может оказаться, что будет производиться копирование не того процесса. Во избежание ненужного копирования необходимо считать контрольные суммы отдельно.

Максимальный размер поля данных должен быть достаточно большим, чтобы обеспечить возможность эффективной работы сети даже при наличии больших задержек. Кроме того, чем больше размер поля данных, тем меньшую долю в общем потоке данных составляют заголовки.

Еще одно полезное свойство протокола — возможность посылать нормальное количество данных вместе с запросом соединения. При этом можно сэкономить время одного запроса и ответа.

Наконец, следует сказать несколько слов о программном обеспечении протокола. Следует сконцентрировать внимание на успешном варианте работы. Мно-

гие старые протоколы были рассчитаны на работу в условиях плохих линий связи и особое внимание уделяли обработке ошибок (например, при потере пакета). Чтобы сделать протокол быстрее, разработчик должен, в первую очередь, постараться минимизировать время обработки в нормальном случае. Минимизация времени обработки в случае ошибок на линии должна быть на втором месте.

Второй аспект, касающийся программного обеспечения, состоит в минимизации времени копирования. Как уже было сказано ранее, копирование данных часто оказывается основным источником накладных расходов. В идеале, аппаратура должна отображать каждый приходящий пакет в память в виде непрерывного блока данных. Затем программа должна скопировать этот пакет в буфер пользователя за одну операцию блочного копирования. В зависимости от принципа работы кэша может быть желателен отказ от циклической организации копирования. Другими словами, чтобы скопировать 1024 слова, возможно, самым быстрым способом будет использование 1024 инструкций MOVE подряд (или 1024 пар инструкций чтения и записи). Процедура копирования является столь критичной, что ее следует очень аккуратно писать вручную на ассемблере, если только нет возможности заставить компилятор произвести самый оптимальный код.

Резюме

Транспортный уровень — это ключ к пониманию многоуровневых протоколов. Он предоставляет различные услуги, наиболее важной из которых является сквозной, надежный, ориентированный на соединение поток байтов от отправителя к получателю. Доступ к нему предоставляется при помощи сервисных примитивов, позволяющих устанавливать, использовать и разрывать соединения.

Транспортные протоколы должны обладать способностью управлять соединением в ненадежных сетях. Установка соединения осложняется возможностью существования дубликатов пакетов, которые могут появляться в самый неподходящий момент. Для борьбы с этими дубликатами при установке соединения применяется алгоритм «тройного рукопожатия». Разрыв соединения проще установки и, тем не менее, далеко не тривиален из-за наличия проблемы двух армий.

Даже если сетевой уровень абсолютно надежен, у транспортного уровня полно работы. Он должен обрабатывать все служебные примитивы, управлять соединениями и таймерами, а также предоставлять и использовать кредиты.

Основными транспортными протоколами Интернета являются TCP и UDP. UDP — это протокол без установления соединения, который работает с IP-пакетами и занимается обеспечением мультиплексирования и демультиплексирования нескольких процессов с использованием единого IP-адреса. UDP может использоваться при клиент-серверных взаимодействиях, например, при удаленном вызове процедур. Кроме того, на его основе можно создавать протоколы реального времени, такие как RTP.

Наиболее распространенным протоколом Интернета является TCP. Он обеспечивает надежную двухстороннюю потоковую байтовую передачу. Он использует

20-байтный заголовок для всех сегментов. Сегменты могут фрагментироваться маршрутизаторами Интернета, поэтому хосты должны уметь восстанавливать исходные сегменты из отдельных фрагментов. Оптимизации производительности протокола TCP было уделено много внимания. Для этого в нем применяются алгоритмы Нагля (Nagle), Кларка (Clark), Джекобсона (Jacobson), Карна (Karn) и др. Беспроводные линии связи приводят к усложнению протокола TCP. Транзакционный TCP — это расширение традиционного протокола TCP, предназначенное для поддержки клиент-серверного взаимодействия с использованием упрощенной процедуры обмена пакетами.

Производительность сети обычно в основном определяется протоколом и накладными расходами по обработке TPDU-модулей, причем с увеличением скорости передачи данных эта ситуация ухудшается. При разработке протоколов следует стараться минимизировать количество TPDU-модулей, количество переключений контекста и время копирования TPDU-модулей. В гигабитных сетях требуются простые протоколы.

Вопросы

1. В нашем примере транспортных примитивов, приведенных в табл. 6.1, LISTEN является блокирующим вызовом. Обязательно ли это? Если нет, объясните, как следует пользоваться неблокирующим примитивом. Какое преимущество это даст по сравнению со схемой, описанной в тексте?
2. В модели, лежащей в основе диаграммы состояний на рис. 6.3, предполагается, что пакеты могут теряться на сетевом уровне и поэтому должны подтверждаться индивидуально. Допустим, что сетевой уровень обеспечивает 100-процентную надежность доставки и никогда не теряет пакеты. Нужны ли какие-либо изменения в диаграмме состояний, показанной на рис. 6.3, и если да, то какие?
3. В обеих частях листинга 6.1 значение SERVER_PORT должно быть одинаковым у клиента и у сервера. Почему это так важно?
4. Предположим, что используется управляемая часами схема генерирования начальных порядковых номеров с 15-разрядным счетчиком часов. Часы тикают раз в 100 мс, а максимальное время жизни пакета равно 60 с. Как часто должна производится ресинхронизация:
   1) В худшем случае?
   2) Когда данные потребляют 240 порядковых номеров в минуту?
5. Почему максимальное время жизни пакета T должно быть достаточно большим, чтобы гарантировать, что не только пакет, но и его подтверждение исчезли?
6. Представьте, что для установки соединений вместо «тройного рукопожатия» использовалось бы двойное (то есть третье сообщение не требовалось). Возможны ли при этом тупиковые ситуации? Приведите пример или докажите, что тупиковых ситуаций нет.

Представьте себе обобщенную проблему n армий, в которой договоренность двух любых армий достаточна для победы. Существует ли протокол, позволяющий армиям синих выиграть?

Рассмотрим проблему восстановления от сбоев хостов (рис. 6.15). Если бы интервал между записью и отправкой подтверждения, или наоборот, можно было сделать относительно небольшим, какими были бы две лучшие стратегии отправителя и получателя, минимизирующие шансы ошибки протокола?

Возможны ли тупиковые ситуации для транспортной сущности, описанной в тексте (листинг 6.2)?

Из любопытства разработчик транспортной сущности, представленной в листинге 6.2, решил поместить внутрь процедуры `sleep` счетчики для сбора статистики о массиве *conn*. Среди прочих параметров определялось число соединений в каждом из семи возможных соединений, n_i (i = 1, ..., 7). Написав на языке FORTRAN большую программу для анализа данных, разработчик обнаружил, что соотношение $\Sigma n_i = MAX_CON$ оказывается всегда верным. Есть ли еще другие инварианты, включающие только эти семь переменных?

Что происходит, когда пользователь транспортной сущности, представленной в листинге 6.2, посылает сообщение нулевой длины? Обсудите значение этого факта.

Для каждого события, которое потенциально может произойти в транспортной сущности, представленной в листинге 6.2, определите, является ли оно разрешенным или нет, когда пользователь находится в состоянии *sending*.

Обсудите преимущества и недостатки схемы кредитов по сравнению с протоколами скользящего окна.

Зачем нужен протокол UDP? Разве не достаточно было бы просто позволить пользовательским процессам посылать необработанные IP-пакеты?

Рассмотрите простой протокол прикладного уровня, построенный на основе UDP, который позволяет клиенту запрашивать файл с удаленного сервера, расположенного по популярному адресу. Клиент вначале посылает запрос с именем файла, а сервер отвечает последовательностью информационных пакетов, содержащих различные части запрошенного файла. Для обеспечения надежности и доставки частей в правильном порядке клиент и сервер используют протокол с ожиданием. Какие проблемы могут возникнуть с таким протоколом, кроме очевидных проблем с производительностью? Обратите внимание на вероятность сбоя процессов.

Клиент посылает 128-байтный запрос на сервер, удаленный от него на 100 км, по оптоволокну со скоростью 1 Гбит/с. Какова эффективность линии во время выполнения удаленного вызова процедуры?

Рассмотрите снова ситуацию, описанную в предыдущем вопросе. Вычислите минимально возможное время ответа для данной линии со скоростью 1 Гбит/с и для 1-мегабитной линии. Какой вывод можно сделать, исходя из полученных значений?

18. Как в UDP, так и в TCP номера портов используются для идентификации принимающей сущности при доставке сообщения. Назовите две причины того, почему для этих протоколов были изобретены новые абстрактные идентификаторы (номера портов) и не использовались идентификаторы процессов, уже существовавшие на момент появления данных протоколов?
19. Каков суммарный размер минимального MTU протокола TCP, включая накладные расходы TCP и IP, но не включая накладные расходы уровня передачи данных?
20. Фрагментация и дефрагментация дейтаграмм выполняется протоколом IP и невидима для протокола TCP. Означает ли это, что протокол TCP не должен беспокоиться о данных, приходящих в неверном порядке?
21. RTP используется для передачи звукозаписей, по качеству соответствующих компакт-дискам. При этом для передачи одного отсчета каждого из стереоканалов используется пара 16-битных слов, передающихся 44 100 раз в секунду. Сколько пакетов в секунду должен уметь передавать RTP?
22. Возможно ли поместить код RTP в ядро операционной системы наряду с UDP? Ответ поясните.
23. Процессу хоста 1 был назначен порт p, а процессу хоста 2 —порт q. Может ли быть одновременно несколько соединений между этими двумя портами?
24. На рис. 6.23 мы видели, что в дополнение к 32-разрядному полю *Подтверждение* в четвертом слове имеется бит *ACK*? Приносит ли он какую-либо пользу? Ответ поясните.
25. Максимальный размер полезной нагрузки TCP-сегмента может быть равен 65 495 байт. Почему было выбрано такое странное число?
26. Опишите два способа, которыми можно попасть в состояние *SYN RCVD* на рис. 6.26.
27. В чем состоят потенциальные недостатки использования алгоритма Нагля в сильно перегруженной сети?
28. Рассмотрите эффект использования алгоритма затяжного пуска в линии со значением времени прохождения сигнала в оба конца, равным 10 мс, без перегрузок. Размер окна получателя 24 Кбайт, а максимальный размер сегмента равен 2 Кбайт. Через какое время может быть послано полное окно?
29. Предположим, окно перегрузки протокола TCP установлено на 18 Кбайт, когда происходит тайм-аут. Каким будет размер окна, если четыре последующих передачи будут успешными? Максимальный размер предполагается равным 1 Кбайт.
30. Текущее значение оценки времени прохождения сигнала в оба конца протокола TCP *RTT* равно 30 мс, а следующие подтверждения приходят через 26, 32 и 24 мс. Каково будет новое значение *RTT*? Используйте $\alpha = 0{,}9$.
31. TCP-машина посылает окна по 65 535 байт по гигабитному каналу, в котором время прохождения сигнала в один конец равно 10 мс. Какова максимальная достижимая пропускная способность канала? Чему равна эффективность использования линии?

Какова максимальная скорость, с которой хост может посылать в линию TCP-пакеты, содержащие 1500 байт полезной нагрузки, если максимальное время жизни пакета в сети равно 120 с? Требуется, чтобы порядковые номера не зацикливались. При расчете учитывайте накладные расходы на TCP, IP и Ethernet. Предполагается, что кадры Ethernet могут посылаться непрерывно.

Чему равна максимальная скорость передачи данных для каждого соединения, если максимальный размер TPDU-модуля равен 128 байт, максимальное время жизни TPDU-модуля равно 30 с и используются 8-разрядные порядковые номера TPDU-модулей?

Предположим, вы измеряется время, необходимое для получения TPDU-модуля. Когда возникает прерывание, вы читаете показания системных часов в миллисекундах. После полной обработки TPDU-модуля вы снова читаете показания часов. В результате миллиона измерений вы получаете значения 0 мс 270 000 раз и 1 мс 730 000 раз. Какой вывод можно сделать на основании полученных результатов?

Центральный процессор выполняет 1000 млн инструкций в секунду (1000 MIPS). Данные могут копироваться 64-разрядными словами. На копирование каждого слова требуется 10 инструкций. Может ли такая система управлять гигабитной линией, если каждый приходящий пакет должен быть скопирован четырежды? Для простоты предположим, что все инструкции, даже обращения к памяти, выполняются с максимальной скоростью 1000 MIPS.

Для решения проблемы повторного использования старых порядковых номеров пакетов, в то время как старые пакеты еще существуют, можно использовать 64-разрядные порядковые номера. Однако теоретически оптоволоконный кабель может обладать пропускной способностью до 75 Тбит/с. Каким следует выбрать максимальное время жизни пакетов, чтобы гарантировать отсутствие в сети будущего пакетов с одинаковыми номерами при скорости линий 75 Тбит/с и 64-разрядных порядковых номерах? Предполагается, что порядковый номер дается каждому байту, как в протоколе TCP.

Назовите одно преимущество RPC на основе UDP перед T/TCP. Теперь назовите одно преимущество T/TCP перед RPC.

На рис. 6.33, *а* видно, что для выполнения удаленного вызова процедуры требуется 9 пакетов. Могут ли возникнуть обстоятельства, при которых понадобятся ровно 10 пакетов?

В разделе «Протоколы для гигабитных сетей» мы подсчитали, что гигабитная линия отправляет хосту 80 000 пакетов в секунду, оставляя ему только 6250 инструкций на их обработку — половина производительности процессора отдается приложениям. В результате выяснилось, что размер пакета должен быть 1500 байт. Выполните подсчеты заново для пакетов ARPANET (128 байт). В обоих случаях предполагается, что в размер пакета включены все накладные расходы.

В гигабитной линии протяженности более 4000 км ограничивающим фактором является не пропускная способность, а время задержки. Рассмотрим ре-

гиональную сеть со средней удаленностью отправителя от получателя 20 км. При какой скорости передачи данных время прохождения сигнала в оба конца будет равно времени передачи одного килобайтного пакета?

41. Рассчитайте произведение пропускной способности на задержку в следующих сетях: 1) T1 (1,5 Мбит/с), 2) Ethernet (10 Мбит/с), 3) T3 (45 Мбит/с), 4) STS-3 (155 Мбит/с). Предполагается, что RTT = 100 мс. Не забудьте о том, что в заголовке TCP на размер окна отводится 16-разрядное поле. Как это замечание отразится на результатах вычислений?
42. Чему равно произведение пропускной способности на задержку для 50-мегабитного канала геостационарной спутниковой связи? Если все пакеты имеют размер 1500 байт (включая накладные расходы), какого размера должно быть окно в пакетах?
43. Файловый сервер, моделируемый листингом 6.1, далек от совершенства. Можно внести некоторые улучшения. Проделайте следующие изменения:
    1) пусть у клиента появится третий аргумент, указывающий байтовый диапазон;
    2) добавьте флаг –w в программу клиента, который позволил бы записывать файл на сервер.
44. Измените программу, приведенную в листинге 6.2, таким образом, чтобы она выполняла восстановление после сбоев. Добавьте новый тип пакета *reset*, который может прибывать после того, как соединение было открыто обеими сторонами и никем не закрыто. Это событие, происходящее на обоих концах соединения, означает, что любые пакеты, находившиеся в пути, были либо доставлены, либо уничтожены, но в любом случае они больше не находятся в сети.
45. Напишите программу, моделирующую управление буфером транспортной сущностью и использующую алгоритм скользящего окна вместо примененной в листинге 6.2 системы кредитов. Пусть процесс более высокого уровня случайным образом открывает соединения, посылает данные и закрывает соединения. Для простоты, пусть все данные передаются только с машины *A* на машину *B*. Поэкспериментируйте с различными стратегиями выделения буферов на машине *B*, например, выделяя буферы каждому соединению и организуя общий буферный пул, и измерьте полную пропускную способность, достижимую в каждом случае.
46. Разработайте и реализуйте систему сетевого общения (чат), рассчитанную на несколько групп пользователей. Координатор чата должен располагаться по хорошо известному сетевому адресу, использовать для связи с клиентами протокол UDP, настраивать чат-серверы перед каждой сессией общения и поддерживать каталог чат-сессий. На каждую сессию должен выделяться один обслуживающий сервер. Для связи с клиентами сервер должен использовать TCP. Клиентская программа должна позволять пользователям начинать разговор, присоединяться к уже ведущейся дискуссии и покидать сессию. Разработайте и реализуйте код координатора, сервера и клиента.

Глава 7
Прикладной уровень

- Служба имен доменов DNS
- Электронная почта
- Всемирная паутина (WWW)
- Мультимедиа
- Резюме
- Вопросы

Покончив с изучением базовых сведений о компьютерных сетях, мы переходим к уровню, на котором расположены все приложения. Все уровни, находящиеся в модели OSI ниже прикладного, служат для обеспечения надежной доставки данных, но никаких полезных для пользователя действий не производят. В этой главе мы изучим некоторые реальные сетевые приложения.

Разумеется, даже прикладной уровень нуждается в обслуживающих протоколах, с помощью которых осуществляется функционирование приложений. Соответственно, прежде чем начать рассмотрение самих приложений, мы изучим один из таких протоколов. Речь идет о службе имен доменов, DNS, обеспечивающей присвоение имен в Интернете. Затем мы рассмотрим три реально действующих приложения: электронную почту, Всемирную паутину и, наконец, мультимедиа.

Служба имен доменов DNS

Хотя программы теоретически могут обращаться к хостам, почтовым ящикам и другим ресурсам по их сетевым адресам (например, IP), пользователям запоминать их тяжело. Кроме того, отправка электронной почты на адрес Tanya@128.111.24.41 будет означать, что в случае переезда сервера таниного про-

вайдера или организации на новое место с новым IP-адресом придется изменить ее адрес e-mail. Для отделения имен машин от их адресов было решено использовать текстовые ASCII-имена. Поэтому танин адрес более привычно выглядит в таком виде: Tanya@art.ucsb.edu. Тем не менее, сеть сама по себе понимает только численные адреса, поэтому нужен механизм преобразования ASCII-строк в сетевые адреса. В следующих разделах мы изучим, как производится это отображение в Интернете.

Когда-то давно в сети ARPANET соответствие между текстовыми и двоичными адресами просто записывалось в файле *hosts.txt*, в котором перечислялись все хосты и их IP-адреса. Каждую ночь все хосты получали этот файл с сайта, на котором он хранился. В сети, состоящей из нескольких сотен больших машин, работающих под управлением системы с разделением времени, такой подход работал вполне приемлемо.

Но когда к сети подключились тысячи рабочих станций, всем стало ясно, что этот способ не сможет работать вечно. Во-первых, размер файла рано или поздно стал бы слишком большим. Однако, что еще важнее, если управление именами хостов не осуществлять централизованно, неизбежно возникновение конфликтов имен. В то же время, представить себе централизованное управление именами всех хостов гигантской международной сети довольно сложно. Для разрешения всех этих проблем и была разработана **служба имен доменов (DNS**, Domain Name System).

Суть системы DNS заключается в иерархической схеме имен, основанной на доменах, и распределенной базе данных, реализующей эту схему имен. В первую очередь эта система используется для преобразования имен хостов и пунктов назначения электронной почты в IP-адреса, но также может использоваться и в других целях. Определение системы DNS дано в RFC 1034 и 1035.

В общих чертах система DNS применяется следующим образом. Для преобразования имени в IP-адрес прикладная программа обращается к библиотечной процедуре, называющейся **распознавателем**, передавая ей имя в качестве параметра. Распознаватель посылает UDP-пакет локальному DNS-серверу, который ищет имя в базе данных и возвращает соответствующий IP-адрес распознавателю, который, в свою очередь, передает этот адрес вызвавшей его прикладной программе. Имея IP-адрес, программа может установить TCP-соединение с адресатом или послать ему UDP-пакеты.

Пространство имен DNS

Управление большим и постоянно изменяющимся набором имен представляет собой нетривиальную задачу. В почтовой системе на письмах требуется указывать (явно или неявно) страну, штат или область, город, улицу, номер дома, квартиру и фамилию получателя. Благодаря использованию такой иерархической схемы не возникает путаницы между Марвином Андерсоном, живущим на Мейн-стрит в Уайт Плейнс, штат Нью-Йорк, и Марвином Андерсоном с Мейн-стрит в Остине, штат Техас. Система DNS работает аналогично.

Интернет концептуально разделен на 200 **доменов** верхнего уровня. Доменами называют в Интернете множество хостов, объединенное в логическую группу. Каждый домен верхнего уровня подразделяется на поддомены, которые, в свою очередь, также могут состоять из других доменов, и т. д. Все эти домены можно рассматривать в виде дерева, показанного на рис. 7.1. Листьями дерева являются домены, не разделяющиеся на поддомены (но состоящие из хостов, конечно). Такой конечный домен может состоять из одного хоста или может представлять компанию и содержать в себе тысячи хостов.

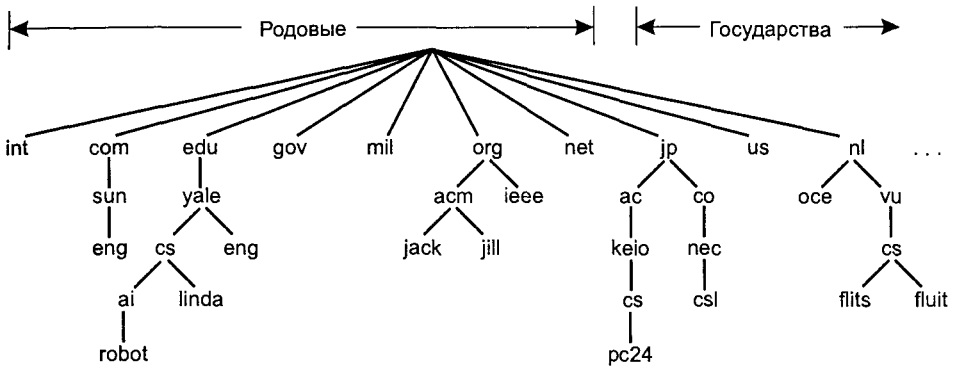


Рис. 7.1. Часть доменного пространства имен Интернета

Домены верхнего уровня разделяются на две группы: родовые домены и домены государств. К родовым относятся домены *com* (commercial — коммерческие организации), *edu* (educational — учебные заведения), *gov* (government — федеральное правительство США), *int* (international — определенные международные организации), *mil* (military — вооруженные силы США), *net* (network — сетевые операторы связи) и *org* (некоммерческие организации). За каждым государством в соответствии с международным стандартом ISO 3166 закреплен один домен государства.

В ноябре 2000 года ICANN было утверждено 4 новых родовых имени доменов верхнего уровня, а именно: *biz* (бизнес), *info* (информация), *name* (имена людей) и *pro* (специалисты, такие как доктора и адвокаты). Кроме того, по просьбе соответствующих отраслевых организаций были введены еще три специализированных имени доменов верхнего уровня: *aero* (аэрокосмическая промышленность), *coop* (кооперативы) и *museum* (музеи). В будущем появятся и другие домены верхнего уровня.

Между прочим, по мере коммерциализации Интернета появляется все больше спорных вопросов. Взять хотя бы домен *pro*. Он предназначен для сертифицированных специалистов. Но кто является специалистом, а кто нет? Кем должны быть эти специалисты сертифицированы? Понятно, что доктора и адвокаты — это профессионалы, спору нет. А что делать со свободными художниками, учителями музыки, заклинателями, водопроводчиками, парикмахерами, мусорщиками, рисователями татуировок, наемниками и проститутками? Имеют ли право

квалифицированные представители всех этих и многих других профессий получать домены *pro*? Если да, то кто выдаст сертификат каждому из этих специалистов?

В принципе, получить домен второго уровня типа *name-of-company.com* несложно. Надо лишь проверить, не занято ли желаемое имя домена кем-то другим и не является ли оно чьей-нибудь торговой маркой. Для этого надо зайти на сайт регистрационного бюро верхнего уровня (в данном случае *com*). Если все в порядке, заказчик регистрируется и за небольшую ежегодную абонентскую плату получает домен второго уровня. На сегодняшний день в качестве имен поддоменов *com* уже используются практически все общеупотребительные английские слова. Попробуйте набрать какое-нибудь слово, касающееся домашнего хозяйства, животных, растений, частей тела и т. д. Вряд ли ошибетесь.

Имя каждого домена, подобно полному пути к файлу в файловой системе, состоит из пути от этого домена до (безымянной) вершины дерева. Компоненты пути разделяются точками. Так, домен технического отдела корпорации Sun Microsystems может выглядеть как *eng.sun.com*, а не так, как это принято в стиле UNIX (*/com/sun/eng*). Следует отметить, что *eng.sun.com* не конфликтует с потенциальным использованием имени *eng* в домене *eng.yale.edu*, где он может обозначать факультет английского языка Йельского университета.

Имена доменов могут быть абсолютными и относительными. Абсолютное имя домена всегда оканчивается точкой (например, *eng.sun.com.*), тогда как относительное имя — нет. Для того чтобы можно было единственным образом определить истинные значения относительных имен, они должны интерпретироваться в некотором контексте. В любом случае именованный домен означает определенный узел дерева и все узлы под ним.

Имена доменов нечувствительны к изменению регистра символов. Так, например, *edu* и *EDU* означают одно и то же. Длина имен компонентов может достигать 63 символов, а длина полного пути не должна превосходить 255 символов.

В принципе, новые домены могут добавляться в дерево двумя разными путями. Например, *cs.yale.edu* можно без проблем поместить в домен *us* под именем *cs.yale.ct.us*. На практике, однако, почти все организации в США помещаются под родовыми доменами, тогда как почти все организации за пределами Соединенных Штатов располагаются под доменами их государств. Не существует каких-либо правил, запрещающих регистрацию под двумя доменами верхнего уровня, однако использует эту возможность лишь небольшое число организаций (за исключением интернациональных, например, *sony.com* и *sony.nl*).

Каждый домен управляет доступом к доменам, расположенным под ним. Например, в Японии домены *ac.jp* и *co.jp* соответствуют американским доменам *edu* и *com*. В Голландии подобное различие не используется, и все домены организаций помещаются прямо под доменом *nl*. В качестве примера приведем имена доменов факультетов компьютерных наук трех университетов.

1. cs.yale.edu (Йельский университет, США).
2. cs.vu.nl (университет Врийе, Нидерланды).
3. cs.keio.ac.jp (университет Кейо, Япония).

Для создания нового домена требуется разрешение домена, в который он будет включен. Например, если в Йеле образовалась группа VLSI, которая хочет зарегистрировать домен *vlsi.cs.yale.edu*, ей нужно разрешение от того, кто управляет доменом *cs.yale.edu*. Аналогично, если создается новый университет, например, университет Северной Южной Дакоты, он должен попросить менеджера домена *edu* присвоить их домену имя *unsd.edu*. Таким образом удается избежать конфликта имен, а каждый домен отслеживает состояние всех своих поддоменов. После того как домен создан и зарегистрирован, в нем могут создаваться поддомены, например *cs.unsd.edu*, для чего уже не требуется разрешения вышестоящих доменов.

Структура доменов отражает не физическое строение сети, а логическое разделение между организациями и их внутренними подразделениями. Так, если факультеты компьютерных наук и электротехники располагаются в одном здании и пользуются одной общей локальной сетью, они, тем не менее, могут иметь различные домены. И наоборот, если, скажем, факультет компьютерных наук располагается в двух различных корпусах университета с различными локальными сетями, логически все хосты обоих зданий обычно принадлежат к одному и тому же домену.

Записи ресурсов

У каждого домена, независимо от того, является ли он одиноким хостом или доменом верхнего уровня, может быть набор ассоциированных с ним **записей ресурсов**. Для одинокого хоста запись ресурсов чаще всего представляет собой просто его IP-адрес, но существует также много других записей ресурсов. Когда распознаватель передает имя домена DNS-серверу, то, что он получает обратно, представляет собой записи ресурсов, ассоциированные с его именем. Таким образом, истинное назначение системы DNS заключается в преобразовании доменных имен в записи ресурсов.

Запись ресурса состоит из пяти частей. Хотя для эффективности они часто перекодируются в двоичную форму, в большинстве описаний записи представляются в виде ASCII-текста, по одной строке на запись ресурса. Мы будем использовать следующий формат:

```
Domain_name Time_to_live Class Type Value
```

Поле *Domain_name* (имя домена) обозначает домен, к которому относится текущая запись. Обычно для каждого домена существует несколько записей ресурсов, и каждая копия базы данных хранит информацию о нескольких доменах. Поле имени домена является первичным ключом поиска, используемым для выполнения запросов. Порядок записей в базе данных значения не имеет. В ответ на запрос о домене возвращаются все удовлетворяющие запросу записи требуемого класса.

Поле *Time_to_live* (время жизни) указывает, насколько стабильно состояние записи. Редко меняющимся данным присваивается высокое значение этого поля, например, 86 400 (число секунд в сутках). Непостоянная информация помечает-

ся небольшим значением, например, 60 (1 минута). Мы вернемся к этому вопросу позднее, когда будем обсуждать кэширование.

Третьим полем каждой записи является поле *Class* (класс). Для информации Интернета значение этого поля всегда равно *IN*. Для прочей информации применяются другие коды, однако на практике они встречаются редко.

Поле *Type* (тип) означает тип записи. Наиболее важные типы записей перечислены в табл. 7.1.

Таблица 7.1. Основные типы записей ресурсов DNS для IPv4

Тип	Смысл	Значение
SOA	Начальная запись зоны	Параметры для этой зоны
A	IP-адрес хоста	32-разрядное целое число
MX	Обмен почтой	Приоритет, с которым домен желает принимать электронную почту
NS	Сервер имен	Имя сервера для этого домена
CNAME	Каноническое имя	Имя домена
PTR	Указатель	Псевдоним IP-адреса
HINFO	Описание хоста	Описание центрального процессора и ОС в виде ASCII-текста
TXT	Текст	Не интерпретируемый ASCII-текст

Запись *SOA* (Start Of Authority — начальная точка полномочий) сообщает имя первичного источника информации о зоне сервера имен (описанного ниже), адрес электронной почты его администратора, уникальный порядковый номер, различные флаги и тайм-ауты.

Самой важной является запись *A* (Address — адрес). Она содержит 32-разрядный IP-адрес хоста. У каждого хоста в Интернете должен быть по меньшей мере один IP-адрес, чтобы другие машины могли с ним общаться. На некоторых хостах может быть одновременно установлено несколько сетевых соединений. В этом случае им требуется по одной записи типа *A* для каждого сетевого соединения (для каждого IP-адреса). DNS можно настроить на циклический перебор этих записей, чтобы в ответ на первый запрос возвращалась первая запись, в ответ на второй запрос — вторая запись, и т. д.

Следующей по важности является запись *MX*. В ней указывается имя хоста, готового принимать почту для указанного домена. Дело в том, что не каждая машина может заниматься приемом почты. Если кто-нибудь хочет послать письмо на адрес, например, bill@microsoft.com, то отправляющему хосту нужно будет вначале найти почтовый сервер на microsoft.com. Запись *MX* может помочь в этих поисках.

Записи *NS* содержат информацию о серверах имен. Например, в каждой базе данных DNS содержится *NS*-запись для каждого домена верхнего уровня, что позволяет пересылать электронную почту на удаленные участки дерева имен. Позднее мы вернемся к этому вопросу.

Записи *CNAME* позволяют создавать псевдонимы. Представим себе, что человек, знакомый в общих чертах с формированием имен в Интернете, хочет по-

слать сообщение человеку с регистрационным именем *paul* на отделении компьютерных наук Массачусетского технологического института (M.I.T.). Он может попытаться угадать нужный ему адрес, составив строку *paul@cs.mit.edu*. Однако этот адрес работать не будет, так как домен отделения компьютерных наук Массачусетского технологического института на самом деле называется *lcs.mit.edu*. Таким образом, для удобства тех, кто этого не знает, M.I.T. может создать запись *CNAME*, позволяющую обращаться к нужному домену по обоим именам. Такая запись будет иметь следующий вид:

```
cs.mit.edu 86400 IN CNAME lcs.mit.edu
```

Как и *CNAME*, запись PTR указывает на другое имя. Однако в отличие от записи *CNAME*, являющейся, по сути, макроопределением, PTR представляет собой регулярный тип данных DNS, интерпретация которого зависит от контекста. На практике запись PTR почти всегда используется для ассоциации имени с IP-адресом, что позволяет по IP-адресу находить имя соответствующей машины. Это называется **обратным поиском**.

Запись *HINFO* позволяет определять тип машины и операционной системы, которой соответствует домен. Наконец, *TXT*-записи позволяют доменам идентифицировать себя произвольным образом. Оба эти типа записей разработаны для удобства пользователей. Ни один из них не является обязательным, поэтому рассчитывать на их наличие не следует, особенно при обработке записей программами (тем более что программы практически невозможно научить обрабатывать эти текстовые данные).

Наконец, последнее поле записи ресурса — это поле *Value* (значение). Это поле может быть числом, именем домена или текстовой ASCII-строкой. Смысл поля зависит от типа записи. Краткое описание поля *Value* для каждого из основных типов записей дано в табл. 7.1.

Пример информации, хранящейся в базе данных DNS домена, приведен в листинге 7.1. В нем показана часть (почти что гипотетической) базы данных домена *cs.vu.nl*, представленного также в виде узла дерева доменов на рис. 7.1. В базе данных содержится семь типов записей ресурсов.

Листинг 7.1. Часть возможной базы данных домена cs.vu.nl

```
; Официальная информация для cs.vu.nl
cs.vu.nl. 86400 IN SOA star boss (952771,7200,7200,2419200,86400)
cs.vu.nl. 86400 IN TXT "Faculteit Wiskunde en Informatica."
cs.vu.nl. 86400 IN TXT "Vrije Universiteit Amsterdam."
cs.vu.nl. 86400 IN MX 1 zephyr.cs.vu.nl.
cs.vu.nl. 86400 IN MX 2 top.cs.vu.nl.
flits.cs.vu.nl. 86400 IN HINFO Sun Unix
flits.cs.vu.nl. 86400 IN A 130.37.16.112
flits.cs.vu.nl. 86400 IN A 192.31.231.165
flits.cs.vu.nl. 86400 IN MX 1 flits.cs.vu.nl.
flits.cs.vu.nl. 86400 IN MX 2 zephyr.cs.vu.nl.
flits.cs.vu.nl. 86400 IN MX 3 top.cs.vu.nl.
www.cs.vu.nl.86400 IN CNAME star.cs.vu.nl
ftp.cs.vu.nl. 86400 IN CNAME zephyr.cs.vu.nl
rowboat IN A 130.37.56.201
```

```
IN MX 1 rowboat
IN MX 2 zephyr
IN HINFO Sun Unix
little-sister IN A 130.37.62.23
IN HINFO Mac MacOS
laserjet IN A 192.31.231.216
IN HINFO "HP Laserjet IIISi" Proprietary
```

В первой строке листинга, не являющейся комментарием, дается основная информация о домене, которая в дальнейшем нас интересовать не будет. В следующих двух строках приводится текстовая информация об организации, которой принадлежит домен. Следующие две строки определяют два хоста, с которыми следует связаться в первую очередь при попытке доставить электронную почту, посланную по адресу person@cs.vu.nl. Хост по имени *zephyr* (специальная машина) следует опросить первым. В случае неудачи следует попробовать доставить письмо машине по имени *top*.

После пустой строки, добавленной для удобства чтения, следуют строки, сообщающие о том, что хост *flits* является рабочей станцией Sun, работающей под управлением операционной системы UNIX, а также даются оба ее IP-адреса. Следующие три строки указывают хосты, которым следует доставлять письма, посылаемые по адресу *flits.cs.vu.nl*. В первую очередь, естественно, следует пытаться доставить письмо самому компьютеру *flits*. Но если этот хост выключен, следует продолжать попытки, обращаясь к хостам *zephyr* и *top*. Следом указаны псевдонимы *www.cs.vu.nl* и *ftp.cs.vu.nl*, позволяющие домену *cs.vu.nl* изменять свой WWW и FTP-серверы, не меняя адресов, по которым пользователи смогут продолжать к ним обращаться.

Следующие четыре строки содержат обычные записи для рабочих станций, в данном случае для *rowboat.cs.vu.nl*. Хранящаяся в базе данных информация содержит IP-адрес, имена первого и второго хостов для доставки почты и информацию о машине. Следом идут две записи о машине *little-sister*, работающей под управлением системы MacOS, отличной от системы UNIX (поэтому эта машина не может сама получать электронную почту). Последние две строки описывают лазерный принтер, подключенный к Интернету.

В этом файле нет IP-адресов доменов верхнего уровня, так как они не принадлежат домену *cs.vu.nl*. Эти адреса поставляются корневыми серверами, чьи IP-адреса присутствуют в файле конфигурации системы и загружаются в DNS-кэш, когда загружается DNS-сервер. Существует около дюжины корневых серверов по всему миру, и каждый из них знает IP-адреса всех серверов доменов верхнего уровня. То есть если машине известен IP-адрес хотя бы одного корневого сервера, она может узнать любое имя DNS.

Серверы имен

Теоретически один сервер мог бы содержать всю базу данных DNS и отвечать на все запросы к ней. На практике этот сервер оказался бы настолько перегруженным, что был бы просто бесполезным. Более того, если бы с ним когда-нибудь что-нибудь случилось, то весь Интернет не работал бы.

Чтобы избежать проблем, связанных с хранением всей информации в одном месте, пространство имен DNS разделено на непересекающиеся **зоны**. Один возможный способ разделения пространства имен, показанного на рис. 7.1, на зоны, изображен на рис. 7.2. Каждая зона содержит часть общего дерева доменов, а также в нее входят серверы имен, хранящие управляющую информацию об этой зоне. Обычно в каждой зоне находится один основной сервер зоны, получающий информацию из файла на своем диске, и несколько дополнительных серверов имен, которые получают информацию от основного сервера имен. Для большей надежности некоторые серверы, обслуживающие зону, могут находиться за пределами самой зоны.

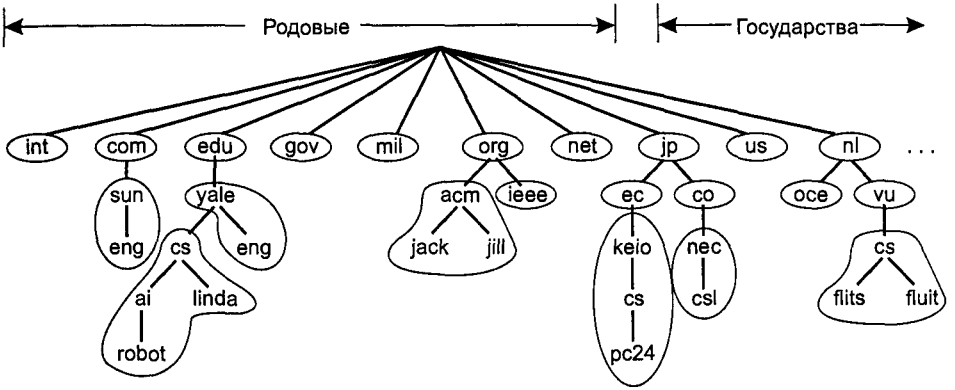


Рис. 7.2. Часть пространства имен DNS, разделенная на зоны

Расстановка границ зон целиком зависит от администратора зоны. Это решение основывается на том, сколько серверов имен требуется в той или иной зоне. Например, на рис. 7.2 у Йельского университета есть сервер для *yale.edu*, управляющий доменом *eng.yale.edu*, но не доменом *cs.yale.edu*, расположенным в отдельной зоне со своими серверами имен. Подобное решение может быть принято, когда факультет английского языка не хочет управлять собственным сервером имен, но этого хочет факультет компьютерных наук. Соответственно, домен *cs.yale.edu* выделен в отдельную зону, а домен *eng.yale.edu* — нет.

Распознаватель обращается с запросом разрешения имени домена к одному из локальных серверов имен. Если искомый домен относится к сфере ответственности данного сервера имен, как, например, домен *ai.cs.yale.edu* подпадает под юрисдикцию домена *cs.yale.edu*, тогда данный DNS-сервер сам отвечает распознавателю на его запрос, передавая ему **авторитетную запись** ресурса. Авторитетной называют запись, получаемую от официального источника, хранящего данную запись и управляющего ее состоянием. Поэтому такая запись всегда считается верной, в отличие от кэшируемых записей, которые могут устаревать.

Однако если домен удаленный, и информацию о запрашиваемом домене нельзя получить от данного сервера имен, последний посылает сообщение с запросом серверу домена верхнего уровня запрашиваемого домена. Поясним данный про-

цесс на примере, показанном на рис. 7.3. В данном случае распознаватель на компьютере *flits.cs.vu.nl* хочет узнать IP-адрес хоста *linda.cs.yale.edu*. На первом шаге он посылает запрос локальному серверу имен, *cs.vu.nl*. Этот запрос содержит имя искомого домена, тип (*A*) и класс (*IN*).

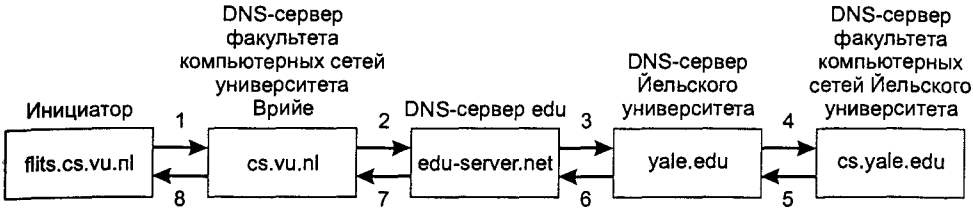


Рис. 7.3. Восемь этапов поиска распознавателем имени удаленного хоста

Предположим, что локальный сервер имен никогда ранее не получал запроса об этом домене и поэтому ничего о нем не знает. Он может опросить несколько находящихся рядом серверов имен, но если они также не знают ответа, данный локальный сервер посылает UDP-пакет серверу домена *edu*, адрес которого содержится в его базе данных. Как видно из рисунка, его имя *edu-server.net*. Маловероятно, чтобы этот сервер знал адрес хоста *linda.cs.yale.edu*. Скорее всего, он даже не знает адреса сервера *cs.yale.edu*, однако он должен знать все свои дочерние домены, поэтому он направляет запрос серверу имен домена *yale.edu* (шаг 3). Тот, в свою очередь, пересылает этот запрос серверу *cs.yale.edu* (шаг 4), у которого должны быть требуемые авторитетные записи ресурсов. Поскольку каждый запрос был от клиента к серверу, посылаемые в ответ записи ресурсов проделывают тот же путь в обратном направлении (шаги 5–8).

Когда записи ресурсов попадают на сервер имен *cs.vu.nl*, они помещаются в кэш на случай, если они понадобятся еще раз. Однако эта информация не является авторитетной, так как изменения в домене *cs.yale.edu* не будут распространяться автоматически на все кэши, в которых может храниться копия этой информации. По этой причине записи кэша обычно долго не живут. В каждой записи ресурса присутствует поле *Time_to_live*. Оно сообщает удаленным серверам, насколько долго следует хранить эту запись в кэше. Если какая-либо машина сохраняет постоянный адрес годами, возможно, будет достаточно надежно хранить эту информацию в кэше в течение одного дня. Для более непостоянной информации, вероятно, более осмотрительно удалять все записи через несколько секунд или одну минуту[1].

Следует отметить, что серия запросов, описанная выше, называется **рекурсивным запросом**, так как каждый сервер, который не владеет запрашиваемой информацией, передает запрос дальше, а затем ответ, также поэтапно, пересылается обратно. Возможны и другие схемы реализации запросов. Так, например,

[1] Практика показывает, что удобнее, наоборот, хранить эту информацию на своем хосте, для чего существуют специальные программы, поддерживающие мини-базы DNS для часто используемых адресов на компьютерах пользователя. — *Примеч. перев.*

возможна форма запроса, в которой в случае отсутствия требуемой информации на локальном сервере клиент сразу получает сообщение о неуспешном выполнении запроса, но при этом ему сообщается имя следующего сервера, который можно об этом спросить. Такая процедура, дающая клиенту больше контроля над процессом поиска, реализована на некоторых серверах. Некоторые серверы вообще не выполняют рекурсивных запросов, а всегда возвращают имя сервера для следующего запроса.

Также следует сказать, что когда DNS-клиент не получает ответа на посланный запрос, он обычно пытается получить искомую информацию у другого сервера, прежде чем истечет период ожидания. Это делается исходя из предположения о том, что сервер, которому адресовался запрос, может в данный момент оказаться выключенным (а не о том, что запрос или ответ потерялся).

Хотя система DNS чрезвычайно важна для обеспечения корректной работы Интернета, основная ее задача заключается в отображении текстовых имен машин на пространство их IP-адресов. Она не помогает найти людей, ресурсы, услуги или другие объекты. Для этого существует иная услуга, называемая **LDAP** (Light-weight Directory Access Protocol — облегченный протокол службы каталогов). Это упрощенная версия стандартной службы каталогов OSI X.500, описание LDAP содержится в RFC 2251. Информация организуется в древовидной форме, и по этому дереву можно осуществлять поиск различных компонентов. Это напоминает телефонную книгу типа «белые страницы». Мы не будем более возвращаться к этой системе, а дополнительную информацию все желающие могут найти в (Weltman и Dahbura, 2000).

Электронная почта

Электронная почта, или **e-mail**, как называют ее многочисленные любители, существует уже более двух десятилетий. До 1990 года она использовалась преимущественно в научных организациях. В 90-е годы она получила широкую известность, и с тех пор количество отправляемых с помощью электронной почты писем стало расти экспоненциально. Среднее число сообщений, посылаемых ежедневно, скоро во много раз превзошло число писем, отправляемых с помощью обычной, бумажной почты.

Электронной почте, как и любой форме коммуникаций, присущ определенный стиль и набор соглашений. В частности, общение по электронной почте носит очень неформальный и демократичный характер. Скажем, человек, который никогда бы не осмелился позвонить или даже написать бумажное письмо какой-нибудь Особо Важной Персоне, запросто может сесть и написать ей небрежное электронное сообщение.

В электронной почте люди обожают использовать особый жаргон и сокращения, такие как BTW (By The Way — между прочим), ROTFL (Rolling On The Floor Laughing — катаюсь по полу от смеха), IMHO (In My Humble Opinion — по моему скромному мнению) и т. д. Кроме того, чрезвычайно популярны так называемые **смайлики**, или **эмотиконы**. Самые интересные из них представлены в табл. 7.2. Чтобы понять, в чем состоит их смысл, бывает полезно повернуть

книгу на 90° по часовой стрелке (или голову — на 90° против часовой стрелки). Существует брошюра (Sanderson и Dougherty, 1993), в которой перечислено свыше 650 смайликов.

Таблица 7.2. Некоторые смайлики. Их не надо зубрить перед выпускным экзаменом :-)

Смайлик	Значение	Смайлик	Значение	Смайлик	Значение
:-)	Не воспринимай всерьез	=:-)	Г-н Линкольн	:+)	Большой нос
:-(	Я зол/огорчен	=):-)	Дядя Сэм	:-))	Двойной подбородок
:-|	Мне безразлично это	*<:-)	Дед Мороз	:-{)	С усами
;-)	Подмигиваю	<:-(	Болван	#:-)	Взъерошенные волосы
:(O)	Громко кричу	(-:	Австралиец (левша)	8-)	Носит очки
:-(*)	Мне тошно от этого	:-)X	С бабочкой	C:-)	Очень умный

Первые системы электронной почты состояли просто из протоколов передачи файлов и договоренности указывать адрес получателя в первой строке каждого сообщения (то есть файла). Со временем недостатки данного метода стали очевидны. Перечислим некоторые из них:

1. Было очень неудобно отсылать сообщения группе получателей. Эта возможность часто требовалась менеджерам для рассылки уведомлений своим подчиненным.
2. Сообщения не обладали внутренней структурой, что затрудняло их компьютерную обработку. Например, если переадресованное сообщение было помещено в тело другого сообщения, извлечь одно сообщение из другого было довольно сложно.
3. Отправитель сообщения никогда не знал, дошло ли его сообщение до адресата.
4. Если кто-либо собирался уехать на несколько недель по делам и хотел, чтобы вся его почта переправлялась его секретарю, организовать это было непросто.
5. Интерфейс пользователя практически отсутствовал. Пользователь должен был сначала в текстовом редакторе набрать сообщение, затем выйти из редактора и запустить программу передачи файла.
6. Было невозможно создавать и посылать сообщения, содержащие смесь текста, изображений (рисунков, факсов, фото) и звука.

Со временем, когда накопился опыт работы, были предложены более сложные системы электронной почты. В 1982 году предложения по работе с электронной почтой, выдвинутые администрацией сети ARPANET, были опубликованы в виде RFC 821 (протокол передачи) и RFC 822 (формат сообщений). Подверг-

шись минимальным изменениям, нашедшим отражение в RFC 2821 и 2822, они фактически стали стандартами Интернета, однако все равно, говоря об электронной почте Интернета, многие ссылаются на RFC 822.

В 1984 году консультативный комитет по международной телефонии и телеграфу (CCITT) впервые представил стандарт X.400. После двух десятков лет борьбы электронная почта, основанная на стандарте RFC 822, получила широкое применение, тогда как системы, базирующиеся на стандарте X.400, практически исчезли. Получилось так, что система, созданная горсткой аспирантов-компьютерщиков, смогла превзойти официальный международный стандарт, имевший серьезную поддержку всех управлений почтово-телеграфной и телефонной связи во всем мире, многих правительств и значительной части компьютерной промышленности. Все это напоминает библейскую историю о Давиде и Голиафе.

Причина успеха стандарта RFC 822 кроется не столько в его достоинствах, сколько в недостатках стандарта X.400. Последний был настолько плохо продуман и так сложен, что никто просто не мог его нормально реализовать. Большинство организаций предпочли простую, но работающую систему электронной почты, основанную на RFC 822, возможно, действительно замечательной, но неработающей системе, в основе которой был стандарт X.400. Может быть, это когда-нибудь послужит кому-нибудь уроком. Итак, далее в нашем обсуждении мы сконцентрируемся на используемых в Интернете системах электронной почты.

Архитектура и службы

В данном разделе мы рассмотрим возможности и организацию систем электронной почты. Обычно они состоят из двух подсистем: **пользовательских агентов**, позволяющих пользователям читать и отправлять электронную почту, и **агентов передачи сообщений**, пересылающих сообщения от отправителя к получателю. Пользовательские агенты представляют собой локальные программы, предоставляющие различные методы взаимодействия пользователя с почтовой системой. Эти методы (или интерфейсы) могут быть командными, графическими или основанными на меню. Агенты передачи сообщений обычно являются системными демонами, работающими в фоновом режиме и перемещающими электронную почту по системе.

Обычно системами электронной почты поддерживаются следующие пять основных функций.

Составление — процесс создания сообщений и ответов. Хотя для создания тела сообщения можно использовать любой текстовый редактор, система поможет в составлении адреса и многочисленных полей заголовков, добавляемых к каждому сообщению. Например, при составлении ответа на сообщение система электронной почты может извлечь адрес отправителя из полученного письма и автоматически поместить его в нужное место в ответе.

Передача — перемещение сообщений от отправителя к получателю. Для этого требуется установить соединение с адресатом или с какой-либо промежуточной машиной, переслать сообщение и разорвать соединение. Система электронной

почты должна выполнять все эти действия автоматически, не беспокоя пользователя.

Уведомление — информирование отправителя о состоянии сообщения. Что с ним стало? Доставлено оно, потеряно или отвергнуто? Существует множество приложений, в которых подтверждение доставки имеет большую важность и даже может иметь юридическую значимость («Ваша честь, моя электронная система не очень надежна, поэтому я полагаю, что повестка с вызовом в суд просто где-то потерялась»).

Отображение приходящих сообщений на экране необходимо, чтобы пользователи имели возможность читать свою электронную почту. Иногда требуется преобразование текста или вызов специальной программы просмотра, например, для просмотра файла формата PostScript или прослушивания оцифрованного звукового сообщения. Иногда также применяются простые преобразования и форматирование текста.

Наконец, на последнем этапе работы с электронным письмом решается дальнейшая судьба полученного сообщения. Получатель может удалить его, не читая, удалить после прочтения, сохранить и т. д. Также должна быть обеспечена возможность найти полученное ранее письмо и прочитать его еще раз, переслать его другому адресату или обрабатывать полученную почту другим способом.

Помимо этих пяти основных услуг большинство систем электронной почты предоставляют великое множество дополнительных функций. Перечислим кратко некоторые из них. Когда пользователи переезжают с места на место или отсутствуют в течение некоторого срока, им может понадобиться автоматическая переадресация их почты.

Большинство систем позволяют пользователям создавать **почтовые ящики** для хранения приходящей почты. Для работы с почтовыми ящиками нужны специальные команды, позволяющие создавать и удалять почтовые ящики, исследовать их содержимое, добавлять и удалять сообщения и т. д.

Менеджерам корпораций часто бывает нужно послать сообщение всем своим подчиненным, заказчикам или поставщикам. Для облегчения выполнения этой задачи применяются **списки рассылки**, представляющие собой список адресов электронной почты. При отправлении сообщения с помощью списка рассылки всем адресатам, числящимся в этом списке, посылаются идентичные копии сообщения.

Среди других полезных дополнительных функций можно перечислить следующие: рассылка копий писем «под копирку» (Carbon copy), рассылка копий без уведомления о других получателях (Blind carbon copy), письма с высоким приоритетом, секретная (то есть зашифрованная) почта, возможность доставки письма альтернативному получателю, если основной временно недоступен, а также возможность перепоручать обработку почты секретарям.

Сегодня электронная почта получила широкое применение в промышленности для обмена информацией внутри компании, что делает возможным совместную работу над сложными проектами далеко удаленных друг от друга (возможно, даже находящихся в различных временных зонах) сотрудников. Более того, общение по электронной почте устраняет концентрацию внимания на различиях

в положении, возрасте и поле, часто мешающих непредвзятому рассмотрению идей. Блестящая идея, посланная студентом-практикантом по электронной почте, имеет шанс выиграть у идеи не столь блестящей, но высказанной вице-президентом компании.

В основе всех современных систем электронной почты лежит ключевая идея о разграничении **конверта** и содержимого письма. Конверт заключает в себе сообщение. Он содержит всю информацию, необходимую для доставки сообщения — адрес получателя, приоритет, уровень секретности и т. п. Все эти сведения отделены от самого сообщения. Агенты передачи сообщений используют конверт для маршрутизации, аналогично тому, как это делает обычная почтовая служба.

Сообщение внутри конверта состоит из двух частей: **заголовка** и **тела письма**. Заголовок содержит управляющую информацию для пользовательских агентов. Тело письма целиком предназначается для человека-получателя. Примеры конвертов и сообщений показаны на рис. 7.4.

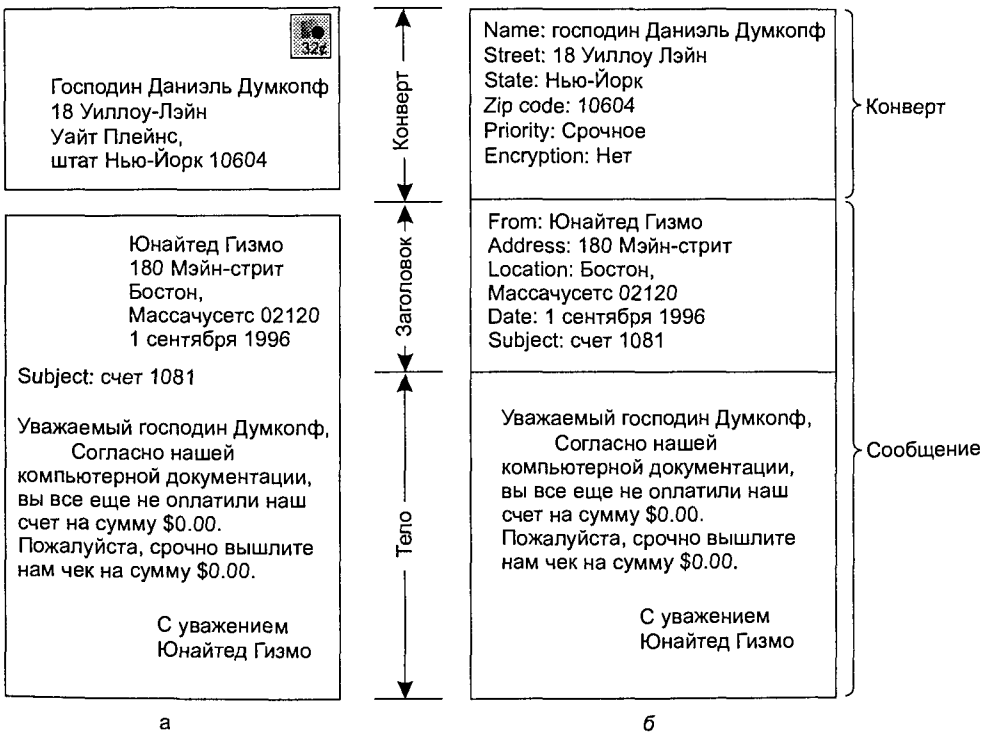


Рис. 7.4. Конверты и сообщения: обычное письмо (а); электронное письмо (*б*)

Пользовательский агент

Как уже было показано, системы электронной почты состоят из двух основных частей: пользовательских агентов и агентов передачи сообщений. В данном разделе мы рассмотрим пользовательских агентов. Как правило, пользовательский

агент — это программа (иногда называемая почтовым редактором), управляемая множеством команд для составления и получения сообщений, а также для ответа на сообщения и управления почтовыми ящиками. Некоторые пользовательские агенты снабжены причудливым интерфейсом с использованием меню или графическим интерфейсом, для работы с которым требуется мышь, тогда как другие управляются односимвольными командами, вводимыми с клавиатуры. Функционально все они примерно одинаковы. В некоторых из них есть меню или ярлыки, но для основных действий обычно существуют комбинации клавиш.

Отправление электронной почты

Чтобы послать письмо электронной почтой, пользователь должен составить текст сообщения, указать адрес получателя, а также, возможно, некоторые дополнительные параметры. Текст сообщения может быть набран в отдельном текстовом редакторе, с помощью программы изготовления документов или в текстовом редакторе, встроенном в почтовый редактор. Адрес получателя должен быть указан в формате, понятном для пользовательского агента. Многие пользовательские агенты ожидают DNS-адреса вида *пользователь@DNS-адрес*. Система доменных имен DNS уже рассматривалась ранее в этой главе, поэтому сейчас мы не станем подробно останавливаться на этом вопросе.

Однако следует отметить, что существуют и другие формы адресации. В частности, адреса стандарта X.400 абсолютно не похожи на DNS-адреса и состоят из пар *атрибут* = значение, разделенных косыми чертами, например:

```
/C=US/ST=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR./CN=KEN SMITH/
```

В этом адресе указаны государство, штат, местоположение, личный адрес и имя получателя (Ken Smith). Возможно также использование различных других атрибутов, что делает возможным отправку электронного письма человеку, чьего имени вы не знаете, при условии, что вам известны другие атрибуты (например, название компании и должность получателя). Хотя форма адресации X.400 значительно менее удобна, чем DNS, большинством систем электронной почты поддерживаются так называемые **псевдонимы**, с помощью которых пользователи могут вводить или выбирать имя адресата и получать корректный электронный адрес. Поэтому даже при использовании адресации X.400 пользователю не приходится вводить такие, мягко говоря, странные строки.

Большинством систем электронной почты поддерживаются списки рассылки, позволяющие пользователю рассылать одно и то же сообщение группе получателей при помощи одной команды. Если список рассылки хранится локально, пользовательский агент может просто послать каждому получателю отдельное сообщение. Однако при удаленном расположении списка рассылки сообщения будут тиражироваться там, где хранится список. Допустим, у группы исследователей птиц есть список рассылки, называющийся *birders* (птицеловы), установленный на домене *meadowlark.arizona.edu*. Тогда любое сообщение, посланное по адресу *birders@meadowlark.arizona.edu*, будет сначала отправляться в университет штата Аризона, а затем рассылаться оттуда в виде отдельных сообщений всем членам списка рассылки, где бы они ни находились. Для отправителя письма список

рассылки внешне не отличается от обычного индивидуального адреса. Если отправитель не знает, что *birders* — это список рассылки, он вполне может подумать, что он посылает письмо лично некоему профессору по имени Gabriel O. Birders.

Чтение электронной почты

Обычно при запуске пользовательский агент просматривает содержимое почтового ящика пользователя на предмет наличия новой почты. Затем он может объявить пользователю число новых сообщений в почтовом ящике или отобразить по одной строке сведений о каждом письме, после чего перейти в режим ожидания команды пользователя.

В качестве примера работы пользовательского агента рассмотрим типичный сценарий. Запустив пользовательский агент, пользователь запрашивает краткую сводку о своей почте. На экране при этом появляется список писем (см. табл. 7.3). Каждая строка соответствует одному полученному письму. В данном примере в почтовом ящике содержится восемь сообщений.

Таблица 7.3. Пример отображения содержимого почтового ящика

#	Флаги	Размер	Отправитель	Тема
1	K	1030	asw	Изменение в системе MINIX
2	KA	6348	vovka	Не все Вовки так уж противны
3	KF	4519	Amy N. Wong	Запрос сведений
4		1236	bal	Биоинформатика
5		104110	kaashoek	Материалы по одноранговым сетям
6		1223	Frank	Re: Вы рассмотрите мой запрос на грант?
7		3110	guido	Наша статья принята
8		1204	dmr	Re: посещение моего студента

Каждая отображаемая строка содержит несколько полей, извлеченных из конверта или заголовка соответствующего сообщения. В простой системе электронной почты список отображаемых полей встроен в программу. В более сложных системах пользователь может выбрать отображаемые поля, а настройки пользователя будут храниться в специальном файле, называющемся **профилем пользователя**. В данном примере первое отображаемое поле — номер сообщения. Второе поле, *Flags* (флаги) может содержать флаг *K*, означающий, что сообщение не является новым, уже было прочитано и хранится в почтовом ящике; флаг *A*, означающий, что на данное сообщение уже был отправлен ответ; и/или флаг *F*, означающий, что сообщение было переадресовано кому-то еще. Возможно также использование и других флагов.

Третье поле одержит размер сообщения в байтах, а в четвертом поле указывается отправитель сообщения. Поскольку значение этого поля просто извлекается из заголовка сообщения, это поле может содержать имена, полные имена, инициалы, имена регистрации в системе, а также все, что отправитель захочет указать в качестве своего имени. Наконец, поле *Subject* (тема) содержит краткое из-

ложение содержания сообщения. Пользователи, забывающие заполнять поле *Subject*, часто обнаруживают, что их письма читаются респондентами далеко не в первую очередь.

После того как программа отобразила заголовки, пользователь может выполнить одну из нескольких команд: чтение, удаление письма и т. д. Старые системы с текстовым интерфейсом обычно управлялись с помощью односимвольных команд, таких как T (вывести сообщение), A (создать ответ), D (удалить сообщение) и F (переслать). Более современные системы имеют графический интерфейс. Обычно пользователь выбирает сообщение с помощью мыши, затем щелкает на значке, соответствующем определенному действию (выводу сообщения, созданию ответа, удалению и переадресации).

Электронная почта сильно изменилась с тех пор, когда она представляла собой простую передачу файлов. Пользовательские агенты с развитым набором услуг позволяют управляться с огромными потоками почты. Для всех, кому приходится получать и отправлять тысячи писем в год, такие программы просто незаменимы.

Форматы сообщений

Перейдем теперь от рассмотрения пользовательского интерфейса к формату самих сообщений электронной почты. Сначала мы рассмотрим основной ASCII-формат электронного письма стандарта RFC 822. Затем мы познакомимся с мультимедийным расширением этого стандарта.

RFC 822

Сообщения состоят из примитивного конверта (описанного в RFC 821), нескольких полей заголовка, пустой строки и, наконец, тела сообщения. Каждое поле заголовка (логически) состоит из одной строки ASCII-текста, содержащей имя поля, двоеточие и (в большинстве случаев) значение поля. RFC 822 был создан несколько десятилетий назад, и в нем нет четкого разграничения конверта и заголовка. Хотя частично стандарт был пересмотрен в RFC 2822, целиком обновить его было невозможно, поскольку RFC 822 уже был очень широко распространен. Обычно пользовательский агент формирует сообщение и передает его агенту передачи сообщений, который с помощью одного из полей заголовка создает конверт нового вида, представляющий собой некую старомодную смесь сообщения и конверта.

Основные поля заголовка, связанные с транспортировкой сообщения, перечислены в табл. 7.4. Поле *To:* содержит DNS-адрес основного получателя. Возможно наличие и нескольких получателей. В поле *Cc:* указываются адреса дополнительных получателей. С точки зрения качества доставки, никакой разницы между основным и дополнительными получателями нет. Разница между ними чисто психологическая и может быть важна для людей, но совершенно не существует для почтовой системы. Термин *Cc:* (carbon copy — экземпляр, сделанный «под копирку») несколько устарел, так как при работе с компьютерами копировальная бумага вообще-то не используется, тем не менее, он прочно обосновался

в электронной почте. Поле *Bcc:* (Blind carbon copy — слепая копия) аналогично предыдущему, с той разницей, что в последнем случае строка этого поля не видна получателям (как основному, так и дополнительным). Это свойство позволяет рассылать одно письмо одновременно нескольким получателям так, что получатели не будут знать, что это письмо послано еще кому-либо кроме них.

Таблица 7.4. Поля заголовка стандарта RFC 822, связанные с транспортировкой сообщения

Поле	Значение
To:	Электронный адрес (адреса) основного получателя (получателей)
Cc:	Электронный адрес (адреса) дополнительного получателя (получателей)
Bcc:	Электронный адрес (адреса) слепой копии
From:	Автор (авторы) сообщения
Sender:	Электронный адрес отправителя
Received:	Строка, добавляемая каждым агентом передачи сообщений на протяжении маршрута
Return-Path:	Может быть использовано для идентификации обратного пути к отправителю

Следующие два поля, *From:* и *Sender:*, сообщают, соответственно, кто составил и отправил сообщение. Это могут быть разные люди. Например, написать письмо может руководитель предприятия, а отослать — его секретарша. В этом случае руководитель будет числиться в поле *From:*, а секретарша — в поле *Sender:*. Поле *From:* является обязательным, тогда как поле *Sender:* может быть опущено, если его содержимое не отличается от содержимого поля *From:*. Эти поля нужны на случай, если сообщение доставить невозможно и об этом следует проинформировать отправителя. Кроме того, по адресам, указанным в этих полях, может быть оправлен ответ.

Строка, содержащая поле *Received:*, добавляется каждым агентом передачи сообщений на пути следования сообщения. В это поле помещаются идентификатор агента, дата и время получения сообщения, а также другая информация, которая может быть использована для поиска неисправностей в системе маршрутизации.

Поле *Return-Path:* добавляется последним агентом передачи сообщений. Предполагалось, что это поле будет сообщать, как добраться до отправителя. Теоретически, эта информация может быть собрана из всех полей *Received:* заголовка (кроме имени почтового ящика отправителя), однако на практике оно редко заполняется подобным образом и обычно просто содержит адрес отправителя.

Помимо полей, показанных в табл. 7.4, сообщения стандарта RFC 822 могут также содержать широкий спектр полей заголовка, используемых пользовательским агентом или самим пользователем. Наиболее часто используемые поля заголовка приведены в табл. 7.5. Информации в таблице достаточно, чтобы понять назначение полей, поэтому мы не станем рассматривать их все подробно.

Таблица 7.5. Некоторые поля, используемые в заголовке сообщения стандарта RFC 822

Поле	Значение
Date:	Дата и время отправки сообщения
Reply-to:	Электронный адрес, на который следует присылать ответ
Message-Id:	Уникальный номер для последующей ссылки на это сообщение
In-Reply-To:	Идентификатор Message-Id сообщения, в ответ на которое посылается это сообщение
References:	Другие важные ссылки (идентификаторы Message-Id)
Keywords:	Ключевые слова, выбираемые пользователем
Subject:	Краткое изложение темы сообщения для отображения в одной строке

Поле *Reply-to:* иногда используется в случае, если ни составитель письма, ни его отправитель не хотят получать на него ответ. Например, управляющий отделом сбыта пишет письмо, информирующее клиента о новом продукте. Это письмо отправляется его секретарем, но в поле *Reply-to:* указан адрес менеджера отдела продаж, который может ответить на вопросы и принять заказы.

В документе RFC 822 открыто сказано, что пользователям разрешается изобретать собственные заголовки для своих нужд при условии, что эти заголовки начинаются со строки *X-*. Гарантируется, что в будущем никакие стандартные заголовки не будут начинаться с этих символов, чтобы избежать конфликтов между официальными и частными заголовками. Иногда умники-студенты включают поля вроде *X-Fruit-of-the-Day:* (сегодняшний плод) или *X-Disease-of-the-Week:* (недуг недели), использование которых вполне законно, хотя их смысл и не всегда понятен.

После заголовков идет тело самого сообщения. Пользователь может разместить в нем все, что ему угодно. Некоторые люди завершают свои послания сложными подписями, включающими рисунки, созданные из ASCII-символов, популярными и малоизвестными цитатами, политическими заявлениями и разнообразными объявлениями (например, «Корпорация АБВ не несет ответственности за высказанное выше мнение. Собственно, она даже не в силах постичь его»).

MIME — многоцелевые расширения электронной почты в сети Интернет

На заре существования сети ARPANET электронная почта состояла исключительно из текстовых сообщений, написанных на английском языке и представленных символами ASCII. Для подобного применения стандарта RFC 822 было вполне достаточно: он определял формат заголовков, но оставлял содержимое сообщения полностью на усмотрение пользователей. На сегодняшний день такой подход уже не удовлетворяет пользователей, привыкших к Интернету. Требуется обеспечить возможность оправления сообщений со следующими характеристиками.

1. Сообщения на языках с надстрочными знаками (например, на французском, немецком, испанском и т. д.).

2. Сообщения на языках, использующих алфавиты, отличные от латинского (например, на иврите или русском).
3. Сообщения на языках без алфавитов (например, китайском, японском, корейском).
4. Сообщения, вообще не являющиеся текстом (например, аудио или видео).

Решение было предложено в документе RFC 1341, а более новая редакция была опубликована в RFC 2045–2049. Это решение, получившее название **MIME** (Multipurpose Internet Mail Extensions, — многоцелевые расширения электронной почты в Интернете), широко применяется в настоящий момент. Далее приведено его описание. Дополнительную информацию о наборе стандартов MIME см. в RFC.

Основная идея стандартов MIME — продолжить использование формата RFC 822, но с добавлением структуры к телу сообщения и с определением правил кодировки не-ASCII-сообщений. Не отклоняясь от стандарта RFC 822, MIME-сообщения могут передаваться с помощью обычных почтовых программ и протоколов. Все, что нужно изменить, — это отправляющие и принимающие программы, которые пользователи могут создать для себя сами.

Стандартами MIME определяются пять новых заголовков сообщения, приведенных в табл. 7.6. Первый заголовок просто информирует пользовательского агента, получающего сообщение, что тот имеет дело с сообщением MIME, а также сообщает ему номер версии MIME, используемой в этом сообщении. Если сообщение не содержит такого заголовка, то оно считается написанным на английском языке и обрабатывается соответствующим образом.

Таблица 7.6. Заголовки RFC 822, добавленные MIME

Заголовок	Описание
MIME-Version:	Указывает версию стандартов MIME
Content-Description:	Описание содержимого. Строка обычного текста, информирующая о содержимом сообщения
Content-Id:	Уникальный идентификатор
Content-Transfer-Encoding:	Указывает способ кодировки тела сообщения для его передачи
Content-Type:	Тип и формат содержимого сообщения

Заголовок *Content-Description* представляет собой ASCII-строку, информирующую о том, что находится в сообщении. Этот заголовок позволяет пользователю принять решение о том, нужно ли ему декодировать и читать сообщение. Если в строке сказано: «Фотография тушканчика Барбары», а получивший это сообщение не является любителем тушканчиков, то, вероятнее всего, он сразу удалит это сообщение, а не станет перекодировать его в цветную фотографию высокого разрешения.

Заголовок *Content-Id* содержит идентификатор содержимого сообщения. В нем используется тот же формат, что и в стандартном заголовке *Message-Id:*.

Заголовок *Content-Transfer-Encoding* сообщает о способе упаковки тела сообщения для его передачи по сети, которая может возражать против символов, отличных от букв, цифр и знаков препинания. Разработано пять схем (имеется

возможность добавления новых схем). Простейшая из них заключается в передаче просто ASCII-текста. Символы ASCII используют 7 разрядов и могут передаваться напрямую протоколом электронной почты при условии, что строка не превышает 1000 символов.

Следующая по простоте схема аналогична предыдущей, но использует 8-разрядные символы, то есть все значения байта от 0 до 255 включительно. Такая схема кодировки нарушает (исходный) протокол электронной почты Интернета, но используется в некоторых частях Интернета, в которых реализовано некоторое расширение исходного протокола. Хотя объявление о способе кодирования не делает его использование автоматически законным, открытое объявление может, по крайней мере, в случае чего объяснить неправильную работу почтовых агентов. Сообщения, использующие 8-разрядную кодировку, также должны соблюдать правило о максимальной длине строки.

Еще хуже обстоит дело с сообщениями в двоичной кодировке. К ним относятся произвольные двоичные файлы, которые не только используют все 8 разрядов в байте, но еще и не соблюдают ограничение на 1000 символов в строке. К этой категории относятся исполняемые программные файлы. Не дается никакой гарантии, что эти двоичные сообщения будут доставлены корректно, но, тем не менее, очень многие пользователи все равно пересылают их друг другу.

Корректный способ кодирования двоичных сообщений состоит в использовании кодировки **base64** (64-символьная кодировка), иногда называемой **ASCII armor** (ASCII-оплетка). При использовании данного метода группы по 24 бита разбиваются на четыре 6-разрядные единицы, каждая из которых посылается в виде обычного разрешенного ASCII-символа. В этой кодировке 6-разрядный символ 0 кодируется ASCII-символом «A», 1 — ASCII-символом «B» и т. д. Затем следуют 26 строчных букв — это уже 10 разрядов, и наконец, + и / для кодирования 62 и 63 соответственно. Последовательности == и = говорят о том, что последняя группа содержит только 8 или 16 бит соответственно. Символы перевода строки и возврата каретки игнорируются, поэтому их можно вставлять в любом месте для того, чтобы строки выглядели не слишком длинными. Таким способом можно передать любой двоичный код.

Для сообщений, почти полностью состоящих из символов ASCII, но с небольшими включениями не-ASCII-символов, подобный метод несколько неэффективен. Вместо него лучше применять кодировку **quoted-printable** (цитируемое печатное кодирование). Это просто 7-битный ASCII, в котором символы, соответствующие значениям ASCII-кода свыше 127, кодируются знаком равенства, за которым следуют две шестнадцатеричных цифры — ASCII-код символа.

Итак, двоичные данные следует посылать в кодировке Base64 или quoted-printable. Когда имеются веские причины не использовать эти методы, можно указать в заголовке *Content-Transfer-Encoding:* свою кодировку.

Последний заголовок в табл. 7.6 представляет наибольший интерес. Он указывает тип тела сообщения. В документе RFC 2045 определены семь типов содержимого сообщений, каждый из которых распадается на несколько подтипов. Подтип отделяется от типа косой чертой, например,

```
Content-Type: video/mpeg
```

Подтип должен быть явно указан в заголовке; подтипов по умолчанию нет. Начальный список типов и подтипов, определенный в документе RFC 2045, приведен в табл. 7.7. С тех пор к ним было добавлено много новых типов и подтипов. Этот список пополняется всякий раз при возникновении соответствующей необходимости.

Таблица 7.7. Типы стандарта MIME и подтипы, определенные в RFC 2045

Тип	Подтип	Описание
Text	Plain	Неформатированный текст
	Enriched	Текст с включением простых команд форматирования
Image	Gif	Неподвижное изображение в формате GIF
	Jpeg	Неподвижное изображение в формате JPEG
Audio	Basic	Слышимый звук
Video	Mpeg	Видеофильм в формате MPEG
Application	Octet-stream	Неинтерпретируемая последовательность байтов
	Postscript	Документ для печати в формате PostScript
Message	Rfc822	Сообщение MIME RFC 822
	Partial	Сообщение разбито на части для передачи
	External-body	Само сообщение должно быть получено по сети
Multipart	Mixed	Независимые части в указанном порядке
	Alternative	То же сообщение в другом формате
	Parallel	Части сообщения следует просматривать одновременно
	Digest	Каждая часть является законченным сообщением стандарта RFC 822

Рассмотрим перечисленные в таблице типы сообщений. Тип *text* означает обычный текст. Комбинация *text/plain* служит для обозначения обычного текстового сообщения, которое может быть отображено на экране компьютера сразу после получения. Для этого не требуется дополнительной обработки или перекодировки. Это значение поля заголовка позволяет передавать обычные сообщения в MIME с добавлением небольшого количества дополнительных заголовков.

Подтип *text/enriched* позволяет включать в текст простой язык разметки документа. Этот язык обеспечивает системно-независимый способ выделять участки текста жирным или наклонным стилями, использовать шрифты самых разных размеров и цветов, отступы, выравнивание, верхние и нижние индексы и формировать простой макет страницы. В основе этого языка разметки лежит язык SGML (Standard Generalized Markup Language — стандартный обобщенный язык разметки), на базе которого был также создан язык HTML (HyperText Markup Language), применяемый в WWW. Например, сообщение

```
<bold> Момент </bold> настал, - сказал <italic> моряк </italic> ...
```

будет отображаться как

Момент настал, — сказал *моряк* ...

Интерпретация зависит от принимающей сообщение системы. Если стили «жирный» и «курсив» доступны, они будут применены. Если нет, для выделения можно использовать подчеркивание, мигание, инверсную печать, выделение другим цветом и т. д. Разные системы могут применять и применяют свои стили.

Когда веб-технологии стали популярны, был добавлен (в RFC 2854) новый тип *text/html*, который позволил пересылать веб-страницы в теле письма RFC 822. В RFC 3023 определен подтип для расширяемого языка разметки страниц, *text/xml*. Далее в этой главе мы рассмотрим HTML и XML.

Следующим типом MIME является *image*. Он используется для передачи неподвижных изображений. На сегодняшний день существует множество различных форматов хранения и передачи изображений, как с использованием сжатия, так и без него. Два формата — GIF и JPEG — встроены практически во все браузеры, однако существует еще множество других, которые, надо полагать, вскоре дополнят этот список.

Типы *audio* и *video* предназначены, соответственно, для передачи звука и двигающегося изображения. Обратите внимание на то, что подтип *video* включает только визуальную информацию, а не звуковую дорожку. Если необходимо передать по электронной почте видеофильм со звуком, то, возможно, придется посылать видеоряд и звуковую дорожку отдельно друг от друга. Это зависит от системы кодирования. Первым видеоформатом, который был определен стандартом MIME, стал формат со скромным названием MPEG (Motion Pictures Experts Group — экспертная группа по вопросам движущегося изображения).

Тип *application* (приложение) предназначен для всех форматов, требующих внешней обработки, не обеспечиваемой ни одним из других типов. Тип *octet-stream* (байтовый поток) представляет собой просто последовательность никак не обрабатываемых байтов. Получив такой поток, пользовательский агент должен, вероятно, предложить пользователю сохранить его в виде файла и запросить для этого файла имя. Последующая обработка файла целиком зависит от пользователя.

Подтип *postscript* означает язык PostScript, созданный компанией Adobe Systems и широко используемый для описания страниц, предназначенных для печати. Многие принтеры имеют встроенные интерпретаторы языка PostScript. Хотя пользовательский агент может для отображения полученных PostScript-файлов просто вызвать внешний интерпретатор языка PostScript, подобные действия, вообще говоря, небезопасны. PostScript является полноценным языком программирования. При достаточном количестве свободного времени и некоторой склонности к самоистязанию на языке PostScript можно написать компилятор языка C или систему управления базами данных. Отображение документа на языке PostScript осуществляется программой на языке PostScript, содержащейся в этом сообщении. Помимо отображения текста, эта программа способна читать, изменять или удалять файлы пользователя, а также может обладать еще целым рядом неприятных побочных эффектов.

Тип *message* позволяет помещать одно сообщение в другое. Это может быть полезно для переадресации письма. Если внутри одного сообщения заключено полное сообщение стандарта RFC 822, следует использовать подтип *rfc822*.

Подтип *partial* позволяет разбивать сообщение на отдельные части и передавать их отдельно (например, в случае, когда инкапсулированное сообщение является слишком длинным). Параметры обеспечивают восстановление сообщения получателем из отдельных фрагментов в правильном порядке.

Подтип *external-body* (внешнее тело) может применяться для очень длинных сообщений (таких как видеофильмы). Вместо того чтобы помещать MPEG-файл в тело письма, в нем сообщается FTP-адрес, по которому пользовательский агент может получить этот файл тогда, когда нужно. Это особенно удобно в случае рассылки по списку рекламных роликов. При этом пользователям, не желающим просматривать ролик, не придется скачивать его по сети в свой почтовый ящик. Даже страшно подумать, что было бы, если бы по электронной почте стали рассылать спам в виде рекламных видеороликов.

Наконец, тип *multipart* позволяет составлять сообщение из нескольких частей, при этом начало и конец каждой части отчетливо указываются. Подтип *mixed* позволяет создавать сообщение из частей различных форматов. В случае применения подтипа *alternative*, напротив, каждая часть должна содержать одно и то же сообщение, но в другом виде или другой кодировке. Например, сообщение может быть послано в виде простого ASCII-текста, а также в форматах RTF и PostScript. Грамотно созданный пользовательский агент, получив такое сообщение, сначала попытается отобразить его в формате PostScript. Если это по какой-либо причине невозможно, тогда производится попытка отобразить часть в формате RTF. Если и это невозможно, отображается ASCII-текст. Части следует располагать в порядке увеличения сложности, чтобы даже старые (до MIME) пользовательские агенты смогли отобразить сообщение хотя бы в виде простого ASCII-текста.

Подтип *alternative* также может использоваться для сообщений, посылаемых одновременно на разных языках. В этом контексте знаменитый розеттский камень, найденный в Египте, может считаться одним из ранних вариантов сообщений типа *multipart/alternative*.

Мультимедийный пример приведен в листинге 7.2. В данном случае поздравление с днем рождения посылается одновременно в виде текста и в виде песни. Если у получателя есть возможность воспроизвести звуковой файл *birthday.snd*, пользовательский агент скачает этот файл с указанного адреса и воспроизведет его. В противном случае на экране получателя в полной тишине отобразится текст сообщения. Части письма разделены двойными дефисами, за которыми следует (определяемая пользователем) строка, указанная как значение параметра *boundary* (граница).

Листинг 7.2. Сообщение, состоящее из RTF-текств и альтернативы в виде звукового файла

```
From: elinor@abc.com
To: carolyn@xyz.com
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@abc.com>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Земля обошла вокруг Солнца целое число раз
Это преамбула. Пользовательский агент игнорирует ее. Ку-ку.
```

```
--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/enriched
Happy birthday to you
Happy birthday to you
Happy birthday dear <bold> Carolyn </bold>
Happy birthday to you
--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
access-type="anon-ftp";
site="bicycle.abc.com";
directory="pub";
name="birthday.snd"
content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

Обратите внимание: заголовок *Content-Type* трижды встречается в данном сообщении. На верхнем уровне он указывает, что сообщение состоит из нескольких частей. Для каждой части он сообщает ее тип и подтип. Наконец, в теле второй части сообщения он указывает пользовательскому агенту тип внешнего файла. Чтобы подчеркнуть это различие, мы использовали в последнем случае строчные символы, хотя для всех заголовков регистр символа не имеет значения. Для внешнего тела в формате, отличном от 7-разрядного ASCII, также требуется заголовок *content-transfer-encoding*.

Для сообщений, состоящих из отдельных частей, существует еще два подтипа. Подтип *parallel* используется в том случае, когда требуется одновременный «просмотр» всех частей сообщения. Например, часто бывает так, что в фильмах звуковой канал отделен от изображения, однако эти два канала удобнее воспроизводить одновременно, а не последовательно.

Наконец, подтип *digest* используется, когда много сообщений упаковываются в одно сообщение. Например, какая-нибудь дискуссионная группа в Интернете может собирать сообщения от подписчиков, а затем высылать их в виде единого сообщения типа *multipart/digest*.

Пересылка писем

Система пересылки писем занимается доставкой электронных сообщений от отправителя получателю. Для этого проще всего установить транспортное соединение от машины-источника до машины-приемника, а затем просто передать по нему сообщение. Вначале мы познакомимся с тем, как это осуществляется в действительности, после чего рассмотрим несколько ситуаций, в которых подобный подход не работает, и обсудим способы разрешения возникающих в связи с этим проблем.

SMTP — простой протокол электронной почты

В Интернете для доставки электронной почты машина-источник устанавливает TCP-соединение с портом 25 машины-приемника. Этот порт прослушивается почтовым демоном, и их общение происходит с помощью протокола **SMTP** (Simple

Mail Transfer Protocol — простой протокол электронной почты). Этот демон принимает входящие соединения и копирует сообщения из них в соответствующие почтовые ящики. Если письмо невозможно доставить, отправителю возвращается сообщение об ошибке, содержащее первую часть этого письма.

Протокол SMTP представляет собой простой ASCII-протокол. Установив TCP-соединение с портом 25, передающая машина, выступающая в роли клиента, ждет запроса принимающей машины, работающей в режиме сервера. Сервер начинает диалог с того, что посылает текстовую строку, содержащую его идентификатор и сообщающую о его готовности (или неготовности) к приему почты. Если сервер не готов, клиент разрывает соединение и повторяет попытку позднее.

Если сервер готов принимать почту, клиент объявляет, от кого поступила почта и кому она предназначается. Если получатель почты существует, сервер дает клиенту добро на пересылку сообщения. Затем клиент посылает сообщение, а сервер подтверждает его получение. Контрольные суммы не проверяются, так как транспортный протокол TCP обеспечивает надежный байтовый поток. Если у отправителя есть еще почта, она также отправляется. После передачи всей почты в обоих направлениях соединение разрывается. Пример диалога клиента и сервера при передаче сообщения из листинга 7.2 показан в листинге 7.3. Строки, посланные клиентом, помечены буквой *C:*, а посланные сервером — *S:*.

Листинг 7.3. Передача сообщения от elinor@abc.com для carolyn@xyz.com

```
S: 220 xyz.com служба SMTP готова
C: HELO abc.com
S: 250 xyz.com приветствует abc.com
C: MAIL FROM: <elinor@abc.com>
S: 250 подтверждаю отправителя
C: RCPT TO: <carolyn@xyz.com>
S: 250 подтверждаю получателя
C: DATA
S: 354 Отправляйте письмо; конец письма обозначается строкой, состоящей из символа "."
C: From: elinor@abc.com
C: To: carolyn@xyz.com
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@abc.com>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: Земля обошла вокруг Солнца целое число раз
C:
C: Это преамбула. Пользовательский агент игнорирует ее. Ку-ку.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/richtext
C:
C: Happy birthday to you
C: Happy birthday to you
C: Happy birthday dear <bold> Carolyn </bold>
C: Happy birthday to you
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
```

```
C: access-type="anon-ftp";
C: site="bicycle.abc.com";
C: directory="pub";
C: name="birthday.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
S: 250 сообщение принято
C: QUIT
S: 221 xyz.com разрываю соединение
```

Несколько комментариев к листингу 7.3. Сначала клиент, естественно, посылает приветствие серверу. Таким образом, первая команда клиента выглядит как *HELO*, что представляет собой наиболее удачный из двух вариантов сокращения слова *HELLO* до четырех символов. Зачем все эти команды было нужно сокращать до четырех букв, сейчас уже никто не помнит.

В нашем примере сообщение должно быть послано только одному получателю, поэтому используется только одна команда *RCPT* (сокращение от слова recipient — получатель). Использование этой команды несколько раз позволяет посылать одно сообщение нескольким получателям. Каждое из них подтверждается или отвергается индивидуально. Несмотря на то, что попытки пересылки сообщения некоторым получателям оказываются неудачными (например, из-за отсутствия адресатов), это сообщение все равно может быть доставлено остальным адресатам, числящимся в списке рассылки.

Наконец, хотя синтаксис четырехсимвольных команд строго определен, синтаксис ответов не столь строг. Правила определяют только числовой код в начале строки. Все, что следует за этим кодом, может считаться комментарием и зависит от конкретной реализации протокола.

Чтобы лучше понять, как работают SMTP и другие рассмотренные в этой главе протоколы, попробуйте сами поработать с ними. В любом случае, для начала найдите машину, подключенную к Интернету. В системе UNIX наберите в командной строке:

```
telnet mail.isp.com 25
```

подставив вместо *mail.isp.com* DNS-имя почтового сервера провайдера. В системе Windows щелкните на кнопке Пуск, затем на кнопке Выполнить и наберите команду в диалоговом окне. В результате выполнения этой команды будет установлено telnet-соединение (то есть соединение TCP) с портом 25 данной машины. Как было показано в табл. 6.3, порт 25 является SMTP-портом. В ответ на введенную команду вы получите что-то вроде этого:

```
Trying 192.30.200.66...
Connected to mail.isp.com
Escape character is '^]'.
220 mail.isp.com Smail #74 ready at Thu, 25 Sept 2002 13:26 +0200
```

Первые три строки посылаются telnet и поясняют для вас происходящее. Последняя строка посылается сервером SMTP удаленной машины и сообщает о го-

товности к общению с вашей машиной и приему почты. Чтобы узнать о доступных командах, наберите

```
HELP
```

Начиная с этого момента, возможен обмен последовательностями команд, показанными в листинге 7.3. Начинаться общение должно с команды клиента HELO.

Стоит отметить, что использование строк ASCII-текста в качестве команд не случайно. Большинство протоколов Интернета работают именно таким образом. Применение ASCII-текста упрощает тестирование и отладку протоколов. Тестирование можно производить, набирая команды вручную, как было показано ранее. Легко читаются выведенные в ответ сообщения.

Несмотря на то, что протокол SMTP определен довольно четко, все же могут возникать определенные проблемы. Одна из них связана с длиной сообщений. Некоторые старые реализации не поддерживали сообщения длиннее 64 Кбайт. Еще одна проблема связана с тайм-аутами. Если таймеры клиента и сервера настроены на разное время ожидания, один из них может внезапно разорвать соединение, в то время как противоположная сторона будет продолжать передачу. Наконец, иногда могут возникать бесконечные «почтовые штормы». Например, если хост 1 хранит список рассылки *A*, а хост 2 — список рассылки *B* и они содержат записи друг о друге, то письмо, посланное по одному из списков, будет создавать бесконечный объем трафика, пока кто-нибудь не заметит это.

Для решения некоторых из этих проблем был разработан расширенный протокол SMTP, **ESMTP**. Он описан в RFC 2821. Клиенты, желающие использовать его, должны начинать сессию связи с посылки приветствия EHLO вместо HELO. Если команда не принимается сервером, значит, сервер поддерживает только обычный протокол SMTP и клиенту следует работать в обычном режиме. Если же EHLO принято, значит, установлена сессия ESMTP и возможна работа с новыми параметрами и командами.

Доставка сообщений

До сих пор мы предполагали, что все пользователи работают на машинах, способных посылать и получать электронную почту. Как мы уже знаем, электронная почта доставляется, когда отправитель устанавливает TCP-соединение с получателем и посылает по нему сообщения. Такая модель прекрасно работала тогда, когда все хосты сети ARPANET (а позднее — Интернет) были, по сути, постоянно на линии и могли принимать TCP-соединения.

Однако с появлением пользователей, связывающихся с провайдерами с помощью модема, такой подход перестал оправдывать себя. Проблема вот в чем: что будет, если Элинор захочет отправить письмо Кэролайн, а та в данный момент не работает в Интернете? Получается, что Элинор не сможет установить TCP-соединение с Кэролайн, следовательно, невозможно будет запустить протокол SMTP, и Кэролайн так и не получит поздравление с днем рождения.

Одно из решений заключается в создании агента передачи сообщений на машине провайдера, который бы принимал и хранил почту для своих пользовате-

лей в их почтовых ящиках. Поскольку такой агент может быть на линии постоянно, электронная почта может отправляться ему круглосуточно.

POP3

К сожалению, такое решение создает новую проблему: как пользователю забрать свою почту у агента передачи сообщений провайдера? Ответ таков: следует создать специальный протокол, который позволил бы пользовательскому агенту (на машине клиента) соединиться с агентом передачи сообщений провайдера (на машине провайдера) и скопировать хранящуюся для него почту. Одним из таких протоколов является **POP3** (Post Office Protocol v. 3 — почтовый протокол, 3-я версия), определенный в документе RFC 1939.

Ситуация, при которой доставка осуществляется в условиях постоянного соединения с Интернетом отправителя и получателя, показана на рис. 7.5, *а*. Иллюстрация ситуации, в которой отправитель находится в текущий момент на линии, а приемник — нет, приведена на рис. 7.5, *б*.

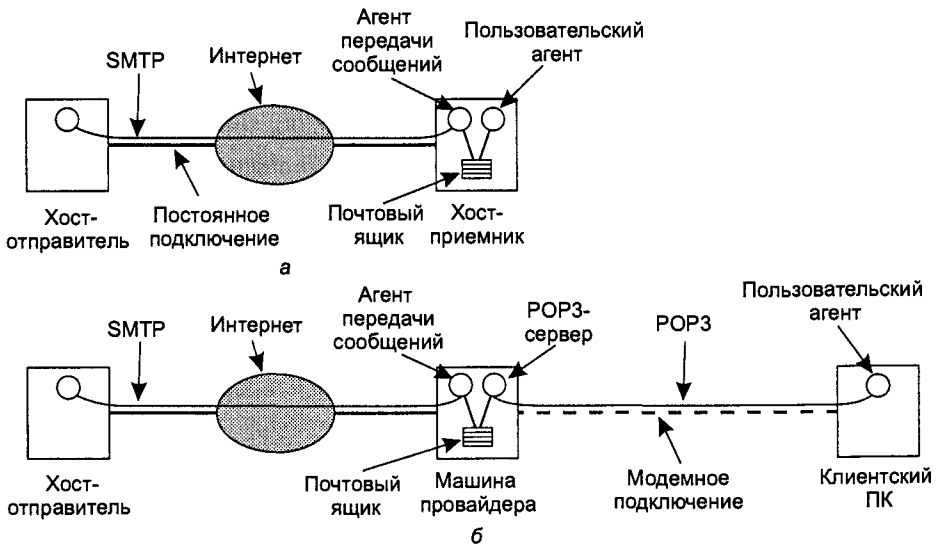


Рис. 7.5. Отправка и прием почты, когда приемник постоянно находится в подключенном состоянии и пользовательский агент работает на одной машине с агентом передачи сообщений (а); прием почты при модемном соединении получателя с провайдером (*б*)

Протокол POP3 начинает свою работу, когда пользователь запускает почтовый редактор. Последний дозванивается до провайдера (если только машина уже не находится в подключенном состоянии) и устанавливает TCP-соединение с агентом передачи сообщений с использованием порта 110. После установки соединения протокол POP3 проходит три последовательных состояния.

1. Авторизация.
2. Транзакции.
3. Обновление.

Авторизация связана с процессом входа пользователя в систему. В состоянии транзакций пользователь забирает свою почту и может пометить ее для удаления из почтового ящика. В состоянии обновления происходит удаление помеченной корреспонденции.

Можно посмотреть, как все это происходит, набрав команду вида

```
telnet mail.isp.com 110,
```

где *mail.isp.com* следует заменить на DNS-имя почтового сервера провайдера. Telnet устанавливает TCP-соединение с портом 110, прослушиваемым POP3-сервером. После установки TCP-соединения сервер посылает ASCII-сообщение, объявляя о своем присутствии. Обычно оно начинается с +OK, затем следует комментарий. Возможный сценарий после установки TCP-соединения показан в листинге 7.4. Как и раньше, строки, начинающиеся с C:, говорят о том, что данная команда исходит от клиента (пользователя), а начинающиеся с S: — что это сообщения сервера (агента передачи сообщений на машине провайдера).

Листинг 7.4. Получение трех сообщений по протоколу POP3

```
S: +OK POP3-сервер готов
C: USER carolyn
S: +OK
C: PASS vegetables
S: OK вход в систему произведен
C: LIST
S: 1 2505
S: 2 14302
S: 3 8122
S: .
C: RETR 1
S: (отправляет сообщение 1)
C: DELE 1
C: RETR 2
S: (отправляет сообщение 2)
C: DELE 2
C: RETR 3
S: (отправляет сообщение 3)
C: DELE 3
C: QUIT
S: +DK Конец соединения с POP3-сервером
```

В состоянии авторизации клиент должен сообщить имя пользователя и пароль. После успешного входа в систему клиент может послать команду LIST для запроса списка писем, хранящихся в почтовом ящике. Каждая строка списка соответствует одному письму, в ней указываются его номер и размер. Точка является признаком конца списка.

После этого пользователь может запросить сообщения командой RETR и пометить их для удаления командой DELE. После получения (и, возможно, установки меток удаления) всех писем пользователь посылает команду QUIT для завершения состояния транзакций и входа в состояние обновления. После удаления сервером всех сообщений он посылает ответ и разрывает TCP-соединение.

Несмотря на то, что протокол POP3 действительно поддерживает возможность получения одного или нескольких писем и оставления их на сервере, большинство программ обработки электронной почты просто скачивают все письма и опустошают почтовый ящик на сервере. Такие действия означают, что реально хранится только одна копия писем — на жестком диске пользователя. Если с ним что-то случается, корреспонденция пропадает безвозвратно.

Теперь подведем небольшие итоги того, как происходит работа с электронной почтой клиентов провайдера. Элинор создает сообщение для Кэролайн с помощью редактора электронной почты (то есть пользовательского агента) и щелкает на значке, чтобы отослать его. Программа передает письмо агенту передачи сообщений на хосте Элинор. Агент передачи сообщений видит, что письмо адресовано carolyn@xyz.com, и использует DNS для поиска записи *MX* для *xyz.com* (где *xyz.com* — провайдер Кэролайн). В ответ на запрос возвращается DNS-имя почтового сервера *xyz.com*. Агент передачи сообщений после этого снова обращается к DNS (например, используя `gethostbyname`): на этот раз ему нужно найти IP-адрес этой машины. Затем с помощью порта 25 найденной машины устанавливается TCP-соединение с SMTP-сервером. Передавая команды SMTP, аналогичные показанным в листинге 7.3, агент пересылает сообщение в почтовый ящик для Кэролайн и разрывает TCP-соединение.

Через некоторое время Кэролайн загружает свой компьютер, соединяется с провайдером и запускает программу электронной почты. Та устанавливает TCP-соединение через порт 110 POP3-сервера, работающего на машине провайдера. Имя DNS или IP-адрес этой машины обычно указывается при установке программы электронной почты либо его получают у провайдера. После установки TCP-соединения почтовая программа Кэролайн запускает протокол POP3 для копирования содержимого почтового ящика на локальный жесткий диск. При этом происходит обмен командами, аналогичными показанным в листинге 7.4. По окончании передачи электронной почты TCP-соединение разрывается. На самом деле в тот же момент можно разорвать и соединение с провайдером, поскольку вся почта уже находится на жестком диске у Кэролайн. Конечно, чтобы отправить ответ на письма, Кэролайн придется снова соединяться с провайдером, поэтому не всегда пользователи разрывают соединение сразу после загрузки почты.

IMAP

Пользователю, имеющему одну учетную запись у одного провайдера и всегда соединяющемуся с провайдером с одной и той же машины, вполне достаточно протокола POP3. Этот протокол и используется повсеместно благодаря его простоте и надежности. Однако в компьютерной индустрии есть такое незыблемое правило: если имеется нечто, что работает безупречно, всегда найдется некто, который захочет снабдить это нечто дополнительными возможностями (и тем самым снабдить его ошибками). Так произошло и с электронной почтой. У многих пользователей есть одна учетная запись в учебном заведении или на работе, но они хотят иметь доступ к ней и из дома, и с работы (учебы), и во время поездок (с портативного компьютера), и из интернет-кафе во время так называемого

отпуска. Хотя POP3 и предоставляет возможность разрешения такой ситуации (так как с его помощью все могут получить всю хранящуюся почту), но проблема в том, что корреспонденция пользователя очень быстро распространится более или менее случайным образом по всем машинам, с которых он получает доступ в Интернет, и некоторые из этих машин могут даже не принадлежать этому пользователю.

Это неудобство привело к созданию альтернативного протокола доставки сообщений, **IMAP** (Interactive Mail Access Protocol — протокол интерактивного доступа к электронной почте), определенного в RFC 2060. В отличие от протокола POP3, который подразумевает, что пользователь будет очищать почтовый ящик после каждого контакта с провайдером и будет работать с почтой в отключенном режиме, протокол IMAP предполагает, что вся почта будет оставаться в почтовых ящиках на сервере неограниченно долго. IMAP обладает широким набором механизмов для чтения сообщений или даже частей сообщений. Такое свойство полезно при использовании медленных модемов, поскольку можно прочесть только текстовую часть письма, к которому приложены большие видео- и аудиофрагменты. Поскольку основное предположение состоит в том, что пользователь не будет копировать на свой компьютер письма, в IMAP входят также инструменты для создания, удаления и других видов управления почтовыми ящиками, размещающимися на сервере. Таким образом, пользователь может завести собственный почтовый ящик для каждого лица, с которым ведется переписка, и переносить сообщения из почтового ящика для всех входящих писем в эти персональные ящики.

Протокол IMAP обладает разнообразными возможностями, например, способностью упорядочивать почту не по порядку ее поступления, как показано в табл. 7.3, а по атрибутам писем (например, «сначала дайте мне письмо от Бобби»). В отличие от POP3, IMAP может заниматься как доставкой исходящей почты от пользователя в направлении места назначения, так и доставлять входящую почту пользователя.

В целом стиль протокола IMAP подобен POP3, пример работы которого показан в листинге 7.4. Различаются они количеством команд — в IMAP их десятки. Сервер IMAP прослушивает порт 143. Сравнение протоколов POP3 и IMAP приведено в табл. 7.8. Следует заметить, что не все провайдеры и не все программы работы с электронной почтой поддерживают оба протокола. Поэтому, выбирая программу и провайдера, следует выяснить, могут ли они работать хоть с одним из этих протоколов, и если да, то с какими именно протоколами.

Таблица 7.8. Сравнение протоколов POP3 и IMAP

Свойство	POP3	IMAP
Где определен	RFC 1939	RFC 2060
Используемый порт TCP	110	143
Место хранения почты	ПК пользователя	Сервер
Способ чтения почты	В автономном режиме	В подключенном режиме
Требуемое время нахождения на линии	Небольшое	Большое

Свойство	POP3	IMAP
Использование ресурсов сервера	Минимальное	Значительное
Поддержка нескольких почтовых ящиков	Отсутствует	Есть
Кто делеет резервные копии почтовых ящиков	Пользователь	Провайдер
Удобство для мобильных пользователей	Нет	Да
Контроль загружаемой почты пользователем	Низкий	Полный
Возможность частичной загрузки сообщений	Нет	Есть
Наличие проблем с нехваткой места на диске	Нет	Есть
Простота реализации	Да	Нет
Популярность	Широкая	Растет

Особенности доставки

Независимо от того, куда доставляется почта — напрямую на рабочую станцию пользователя или на удаленный сервер, — многие системы предоставляют средства дополнительной обработки поступающей почты. Особенно большую ценность для пользователей представляет возможность устанавливать **фильтры** — наборы правил, выполняющихся при получении нового письма или при запуске пользовательского агента. Каждое правило определяет некоторое условие и действие при его выполнении. Например, правило может гласить, что любое сообщение от начальника следует помещать в почтовый ящик номер 1, любое сообщение, пришедшее от кого-либо из группы друзей, следует помещать в почтовый ящик номер 2, а любое сообщение, содержащее определенные неприятные слова в поле *Subject*, следует удалять без вопросов.

Некоторые провайдеры предоставляют фильтры для борьбы со спамом. Эти фильтры автоматически классифицируют приходящие сообщения как нормальные или как спам (нежелательная реклама) и сохраняют каждое письмо в соответствующем ящике. Обычно такие фильтры определяют спам по адресу отправителя, если он является известным распространителем нежелательной почты. Затем анализируется поле *Subject*. Если сотни пользователей только что получили письмо с одинаковыми темами, возможно, это спам. Известны и другие методы выявления спама.

Еще одна предоставляемая услуга — возможность (временной) переадресации приходящей почты по другому адресу. Этим адресом может быть даже компьютер, управляемый коммерческой пейджинговой службой, передающей пользователю на пейджер, по радио или через спутник строку *Subject* каждого сообщения.

Еще одной часто используемой услугой доставки писем является возможность установки специального **каникулярного демона**. Это программа, отсылающая в ответ на приходящие письма сообщение типа

```
Привет! Я в отпуске. Вернусь 24 августа. Желаю веселых каникул!
```

В подобных ответах можно также указывать способы решения срочных проблем, помещать адреса людей, которые могут решить специфические проблемы, и т. п. Большинство каникулярных демонов обычно формируют список лиц, которым они посылали подобные заранее заготовленные ответы, и воздерживаются от отправки такого ответа повторно тому же лицу. Грамотно написанные демоны также проверяют, не было ли полученное сообщение прислано по списку рассылки, и если да, то вообще не отвечают на него. (Люди, посылающие сообщения в большие списки рассылки в летние месяцы, обычно не любят получать в ответ сотни сообщений с подробностями планов на отпуск каждого адресата.)

Автор однажды испытал на себе одну из экстремальных форм обработки электронной почты, когда послал сообщение человеку, утверждающему, что он получает по 600 писем в день. Его личность будет сохранена в тайне, в противном случае по меньшей мере половина читателей этой книги также пошлют ему по письму. Будем условно называть его Джоном.

Джон установил на своей машине почтовый робот, просматривающий каждое прибывающее письмо, проверяя, от нового ли оно отправителя. Если да, то робот посылает стандартный ответ, разъясняющий, что Джон более не в состоянии сам читать всю присылаемую ему почту. Вместо того чтобы отвечать каждому в отдельности, он создал личный FAQ-документ (Frequently Asked Questions — часто задаваемые вопросы). Обычно подобные документы формируются не отдельными людьми, а конференциями.

В качестве ответов на типичные вопросы Джон сообщает свои адрес, номера факса и телефона и объясняет, как связаться с его компанией. Он разъясняет, как можно организовать его лекцию, и сообщает, где можно получить его статьи и другие документы. В этом перечне также даются ссылки на программы, которые он написал, конференцию, которую он ведет, стандарт, который он редактировал, и т. д. Возможно, такие меры и являются необходимыми, хотя, не исключено, что личный перечень типичных вопросов является просто символом исключительного статуса.

Веб-почта

Нельзя под конец не сказать нескольких слов о веб-почте. Некоторые веб-сайты, например Hotmail и Yahoo!, предоставляют электронную почту всем желающим. Работает эта система следующим образом. Имеются обычные агенты передачи сообщений, прослушивающие порт 25 в ожидании входящих SMTP-соединений. Чтобы связаться, например, со службой Hotmail, вам необходимо получить DNS-запись *MX*. Это можно сделать, например, набрав в командной строке UNIX

```
host -a -v hotmail.com
```

Если предположить, что почтовый сервер называется *mx10.hotmail.com*, то для установки TCP-соединения, по которому можно привычным образом обмениваться командами SMTP, следует набрать:

```
telnet mx10.hotmail.com 25
```

Итак, ничего необычного в этой процедуре нет, если не принимать во внимание то, что большие серверы часто бывают заняты и иногда приходится предпринимать несколько попыток установки TCP-соединения.

Интересен здесь вопрос доставки электронной почты. Обычно, когда пользователь заходит на веб-страницу почтового сервера, он видит форму, в которой ему нужно заполнить поля *Имя пользователя* и *Пароль*. После того как он щелкает на кнопке *Войти*, пароль и имя пользователя отправляются на сервер, где проверяется их правильность. Если вход в систему осуществлен корректно, сервер находит почтовый ящик пользователя и строит список, подобный табл. 7.3, только оформленный в виде веб-страницы на HTML. Эта веб-страница отсылается на браузер клиента. Пользователь может щелкать на многих элементах страницы, выполняющих функции работы с почтовым ящиком, — чтения, удаления писем и т. д.

Всемирная паутина (WWW)

Всемирная паутина (WWW, World Wide Web) — это архитектура, являющаяся основой для доступа к связанным между собой документам, находящимся на миллионах машин по всему Интернету. За 10 лет своего существования из средства распространения информации на тему физики высоких энергий она превратилась в приложение, о котором миллионы людей с разными интересами думают, что это и есть «Интернет». Огромная популярность этого приложения стала следствием цветного графического интерфейса, благодаря которому даже новички не встречают затруднений при его использовании. Кроме того, Всемирная паутина предоставляет огромное количество информации практически по любому вопросу, от африканских муравьедов до яшмового фарфора.

Всемирная паутина была создана в 1989 году в Европейском центре ядерных исследований CERN (Conseil Européen pour la Recherche Nucleaire) в Швейцарии. В этом центре есть несколько ускорителей, на которых большие группы ученых из разных европейских стран занимаются исследованиями в области физики элементарных частиц. В эти команды исследователей часто входят ученые из пяти-шести и более стран. Эксперименты очень сложны, для их планирования и создания оборудования требуется несколько лет. Программа Web (паутина) появилась в результате необходимости обеспечить совместную работу находящихся в разных странах больших групп ученых, которым нужно было пользоваться постоянно меняющимися отчетами о работе, чертежами, рисунками, фотографиями и другими документами.

Изначальное предложение, создать паутину из связанных друг с другом документов пришло от физика центра CERN Тима Бернерс-Ли (Tim Berners-Lee) в марте 1989 года. Первый (текстовый) прототип заработал спустя 18 месяцев. В декабре 1991 году на конференции Hypertext'91 в Сан-Антонио в штате Техас была произведена публичная демонстрация.

Эта демонстрация, сопровождаемая широкой рекламой, привлекла внимание других ученых. Марк Андрессен (Marc Andreessen) в университете Иллинойса

начал разработку первого графического браузера, Mosaic. Программа увидела свет в феврале 1993 года и стала настолько популярной, что год спустя ее автор Марк Андрессен решил сформировать собственную компанию Netscape Communications Corp., чьей целью была разработка клиентов, серверов и другого программного обеспечения для веб-приложений. Когда в 1995 году корпорация Netscape получила известность, инвесторы, полагая, очевидно, что появилась еще одна корпорация типа Microsoft, заплатили за пакет ее акций 1,5 млрд. долларов. Это было тем более удивительно, что номенклатура продукции компании состояла всего из одного наименования, компания имела устойчивый пассивный баланс, а в своих проспектах корпорация Netscape объявила, что в обозримом будущем не рассчитывает на получение прибыли. В течение последующих трех лет между Netscape Navigator и Internet Explorer от Microsoft развернулась настоящая «война браузеров». Разработчики с той и с другой стороны в безумном порыве пытались напичкать свои программы как можно большим числом функций (а следовательно, и ошибок) и в этом превзойти соперника. В 1998 году America Online купила корпорацию Netscape Communications за 4,2 млрд долларов, положив таким образом конец весьма непродолжительному независимому существованию компании.

В 1994 году CERN и Массачусетский технологический институт (M.I.T., Massachusetts Institute of Technologies) подписали соглашение об основании **WWW-консорциума** (World Wide Web Consortium, иногда применяется сокращение **W3C**) — организации, цель которой заключалась в дальнейшем развитии приложения Web, стандартизации протоколов и поощрении взаимодействия между отдельными сайтами. Бернерс-Ли стал директором консорциума. Хотя о Всемирной паутине уже написано очень много книг, лучшее место, где вы можете получить самую свежую информацию о ней, это сама Всемирная паутина. Домашнюю страницу консорциума можно найти по адресу http://www.w3.org. На этой странице заинтересованный читатель найдет ссылки на другие страницы, содержащие информацию обо всех документах консорциума и о его деятельности.

Представление об архитектуре

С точки зрения пользователя Всемирная паутина состоит из огромного собрания документов, расположенных по всему миру. Документы обычно называют для краткости просто **страницами**. Каждая страница может содержать ссылки (указатели) на другие связанные с ней страницы в любой точке мира. Пользователи могут следовать по ссылке (например, просто щелкнув на ней мышью), при этом страница, на которую указывает ссылка, загружается и появляется в окне браузера. Этот процесс можно повторять бесконечно. Идея страниц, связанных между собой гиперссылками (**гипертекст**), была впервые пророчески предложена в 1945 году, задолго до появления Интернета, Ванневаром Бушем (Vannevar Bush), профессором из Массачусетского университета, занимавшимся электротехникой.

Страницы просматриваются специальной программой, называемой **браузером**. Самыми популярными браузерами являются программы Internet Explorer и Netscape. Браузер предоставляет пользователю запрашиваемую страницу, интерпрети-

рует ее текст и содержащиеся в нем команды форматирования для вывода страницы на экран. Как и большинство веб-страниц, страница, показанная на рис. 7.6, *а* начинается с заголовка, содержит некоторую информацию и заканчивается адресом электронной почты организации, отвечающей за состояние этой страницы. Строки текста, представляющие собой ссылки на другие страницы, называются **гиперссылками**. Они обычно выделяются либо подчеркиванием, либо другим цветом, либо и тем и другим. Чтобы перейти по ссылке, пользователь перемещает указатель мыши в выделенную область, при этом меняется форма указателя, и щелкает на ней. Хотя существуют и текстовые браузеры, как, например, Lynx, они не так популярны, как графические, поэтому мы сконцентрируем наше внимание на последних. Следует заметить, что также разрабатываются браузеры, управляемые голосом.

ДОБРО ПОЖАЛОВАТЬ НА ВЕБ-СТРАНИЦУ УНИВЕРСИТЕТА ВОСТОЧНОГО ПОДУНКА

- Территория университета
  - Информация о приеме в университет
  - Карта территории университета
  - Как добраться до университета
  - Студенчество университета UEP
- Факультеты университета
  - Факультет психологии животных
  - Факультет альтернативных наук
  - Факультет микробиотической кулинарии
  - Факультет нетрадиционных наук
  - Факультет традиционных наук

Webmaster@eastrodunk.edu

а

ФАКУЛЬТЕТ ПСИХОЛОГИИ ЖИВОТНЫХ

- Информация для будущих студентов
- Персонал
  - Преподаватели факультета
  - Аспиранты
  - Неакадемический штат
- Исследовательские проекты
- Вакансии
- Наши самые популярные курсы
  - Обращение с травоядными животными
  - Управление лошадьми
  - Переговоры с вашей зверушкой
  - Сооружение удобной собачьей будки
- Полный список курсов

Webmaster@animalpsyc.eastrodunk.edu

б

Рис. 7.6. Веб-страница (*а*); страница, отображаемая при щелчке на строке Факультет психологии животных (*б*)

Пользователи, интересующиеся факультетом психологии животных, могут получить подробную информацию о нем, щелкнув мышью на подчеркнутой строке

с его названием. При этом браузер получит из сети страницу, на которую указывает соответствующая ссылка, и отобразит эту страницу (показана на рис. 7.6, *б*) на экране. Подчеркнутые строки этой страницы содержат ссылки на другие страницы, и т. д. Каждая новая страница, на которую указывает ссылка, может располагаться как на той же самой машине, что и первая страница, так и на компьютере, расположенном на противоположном конце Земли. Для пользователя это не заметно. Браузер добывает запрашиваемые страницы без участия пользователя. Если пользователь вернется к странице, на которой он уже был, то использованные им ссылки могут быть выделены другим цветом (или подчеркнуты пунктиром), что позволяет отличать их от тех, на которых пользователь еще не щелкал мышью. Обратите внимание что щелчок мышью на любой не подчеркнутой строке (например, *Campus Information*) не вызовет никакого эффекта. Это просто текст, не связанный ни с какой страницей.

Основной принцип работы Паутины показан на рис. 7.7. Браузер отображает веб-страницу на клиентской машине. Когда пользователь щелкает на строке, которая является ссылкой на страницу, расположенную на сервере *abcd.com*, браузер следует по этой гиперссылке. Реально при этом на *abcd.com* отправляется служебное сообщение с запросом страницы. Получив страницу, браузер показывает ее. Если на этой странице содержится гиперссылка на страницу с сервера *xyz.com*, то браузер обращается с запросом к *xyz.com*, и так далее до бесконечности.

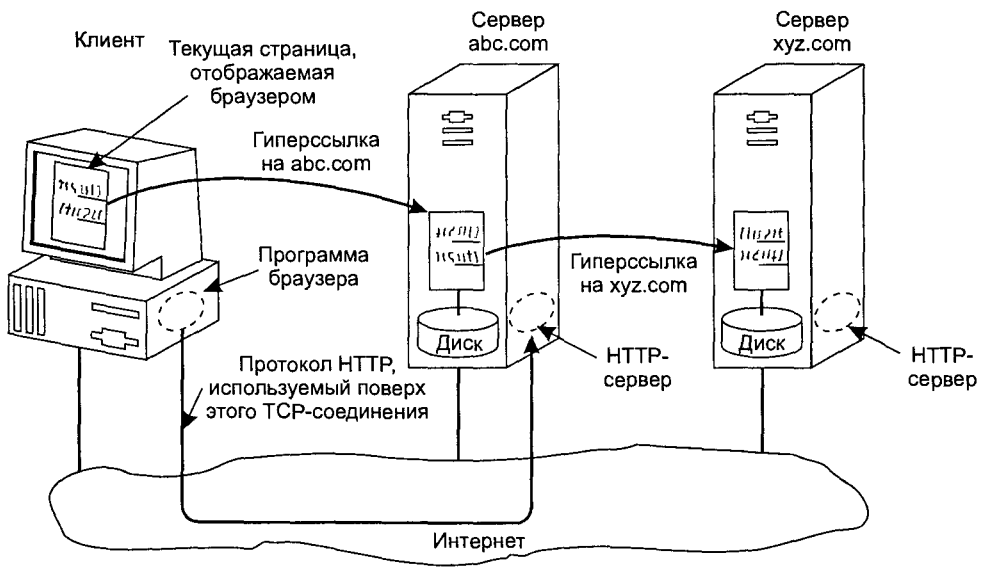


Рис. 7.7. Части Всемирной паутины

Сторона клиента

Давайте теперь более детально рассмотрим сторону клиента, основываясь на рис. 7.7. По сути дела, браузер — это программа, которая может отображать веб-страницы и распознавать щелчки мыши на элементах активной страницы. При

выборе элемента браузер следует по гиперссылке и получает с сервера запрашиваемую страницу. Следовательно, гиперссылка внутри документа должна быть устроена так, чтобы она могла указывать на любую страницу Всемирной паутины. Страницы именуются с помощью **URL** (Uniform Resource Locator — унифицированный указатель информационного ресурса). Типичный указатель выглядит так: http://www.abcd.com/products.html.

Более подробно про унифицированные указатели URL будет рассказано далее в этой главе. Пока что достаточно знать, что URL состоит из трех частей: имени протокола (*http*), DNS-имени машины, на которой расположена страница (www.abcd.com), и (обычно) имени файла, содержащего эту страницу (*products.html*). Между щелчком пользователя и отображением страницы происходят следующие события.

Когда пользователь щелкает мышью на гиперссылке, браузером выполняется ряд действий, приводящих к загрузке страницы, на которую указывает ссылка. Предположим, пользователь, блуждая по Интернету, находит ссылку на документ, рассказывающий про интернет-телефонию, а конкретно — на домашнюю страницу ITU, расположенную по адресу http://www.itu.org/home/index.html. Рассмотрим каждое действие, происходящее после выбора этой ссылки.

1. Браузер определяет URL (по выбранному элементу страницы).
2. Браузер запрашивает у службы DNS IP-адрес *www.itu.org*.
3. DNS дает ответ 156.106.192.32.
4. Браузер устанавливает TCP-соединение с 80-м портом машины 156.106.192.32.
5. Браузер отправляет запрос на получение файла */home/index.html*.
6. Сервер *www.itu.org* высылает файл */home/index.html*.
7. TCP-соединение разрывается.
8. Браузер отображает весь текст файла */home/index.html*.
9. Браузер получает и выводит все изображения, прикрепленные к данному файлу.
10. Многие браузеры отображают текущее выполняемое ими действие в строке состояния внизу экрана. Это позволяет пользователю понять причину низкой производительности: например, не отвечает служба DNS или сервер или просто сильно перегружена сеть при передаче страницы.
11. Для отображения каждой новой страницы браузер должен понять ее формат. Чтобы все браузеры могли отображать любые страницы, они пишутся на стандартизованном языке HTML, описывающем веб-страницы. Более детально мы рассмотрим его далее.

Несмотря на то что браузер, по сути дела, представляет собой интерпретатор HTML, большинство браузеров оснащаются многочисленными кнопками и функциями, облегчающими навигацию по Всемирной паутине. У многих браузеров есть кнопки для возврата на предыдущую страницу и перехода на следующую страницу (последняя доступна только в том случае, если пользователь уже возвращался назад), а также кнопка для прямого перехода на домашнюю страницу

пользователя. Большинство браузеров поддерживают в меню команды для установки закладки на текущей странице и отображения списка закладок, что позволяет попадать на любую страницу при помощи всего одного щелчка мышью. Страницы также могут быть сохранены на диске или распечатаны на принтере. Кроме того, доступны многочисленные функции управления отображением страницы и установки различных настроек пользователя.

Помимо обычного текста (не подчеркнутого) и гипертекста (подчеркнутого), веб-страницы могут также содержать значки, рисунки, чертежи и фотографии. Все они могут быть связаны ссылкой с другой страницей. Щелчок на элементе, содержащем ссылку, также вызовет смену текущей страницы, отображаемой браузером. С большими изображениями, такими как фотографии или карты, может быть ассоциировано несколько ссылок, при этом следующая отображаемая страница будет зависеть от того, на каком месте изображения произведен щелчок мышью.

Далеко не все страницы написаны исключительно с применением HTML. Например, страница может состоять из документа в формате PDF, значка в формате GIF, фотографии в JPEG, звукозаписи в формате MP3, видео в MPEG или чего-то еще в одном из сотни форматов. Поскольку стандартные HTML-страницы могут иметь ссылки на любые файлы, у браузера возникает проблема обработки страницы, которую он не может интерпретировать.

Вместо того чтобы наращивать возможности и размеры браузеров, встраивая в них интерпретаторы для различных типов файлов (количество которых быстро растет), обычно применяется более общее решение. Когда сервер возвращает в ответ на запрос какую-либо страницу, вместе с ней высылается некоторая дополнительная информация о ней. Эта информация включает MIME-тип страницы (см. табл. 7.7). Страницы типа *text/html* выводятся браузером напрямую, как и страницы некоторых других встроенных типов. Если же для данного MIME-типа внутренняя интерпретация невозможна, браузер определяет, как выводить страницу, по своей таблице MIME-типов. В данной таблице в соответствие каждому типу ставится программа просмотра.

Существуют два способа отображения: с помощью подключаемого модуля (**plug-in**) или вспомогательных приложений. Подключаемый модуль представляет собой особый код, который браузер извлекает из специального каталога на жестком диске и устанавливает в качестве своего расширения, как показано на рис. 7.8, *а*. Поскольку подключаемые модули работают внутри браузера, у них есть доступ к текущей странице, вид которой они могут изменять. После завершения своей работы (обычно это связано с переходом пользователя на другую страницу) подключаемый модуль удаляется из памяти браузера.

Каждый браузер имеет набор процедур, которые должны реализовывать все подключаемые модули. Это нужно для того, чтобы браузер мог обращаться к последним. Например, существует стандартная процедура, с помощью которой базовый код браузера передает подключаемому модулю данные для отображения. Набор этих процедур образует интерфейс подключаемого модуля и является специфичным для каждого конкретного браузера.

Кроме того, браузер предоставляет подключаемому модулю определенный набор своих процедур. Среди них в интерфейс браузера обычно включаются про-

цедуры распределения и освобождения памяти, вывода сообщений в строке статуса браузера и опроса параметров браузера.

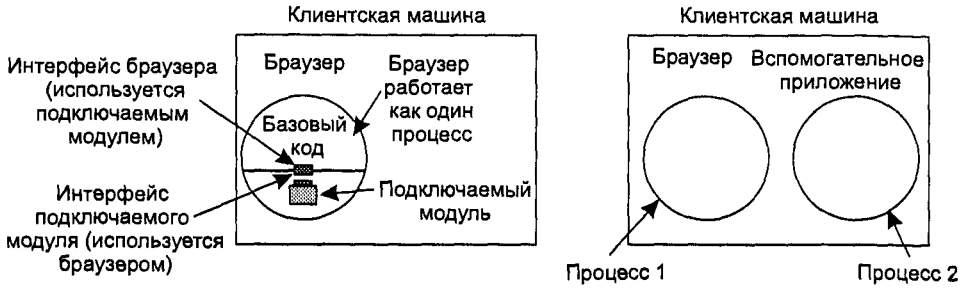


Рис. 7.8. Браузер с подключаемым модулем (*а*); вспомогательное приложение (*б*)

Перед использованием подключаемого модуля его нужно установить. Этот процесс подразумевает, что пользователь копирует с веб-сайта производителя модуля файл установки. В Windows он обычно представляет собой самораспаковывающийся архив ZIP с расширением .exe. Если дважды щелкнуть на таком файле, запускается небольшая программа, включенная в начало архива. Она распаковывает подключаемый модуль и копирует его в соответствующий каталог, известный браузеру. Затем она регистрирует MIME-тип, обрабатываемый модулем, и ассоциирует этот тип с модулем. В системах UNIX установочный файл зачастую представляет собой основной сценарий, осуществляющий копирование и регистрацию.

Вторым способом расширения возможностей браузера является использование **вспомогательных приложений**. Вспомогательное приложение — это полноценная программа, работающая как отдельный процесс. Это показано на рис. 7.8, *б*. Поскольку она никак не связана с браузером, между ними отсутствует какой бы то ни было интерфейс. Вместо этого обычно вспомогательное приложение вызывается с параметром, представляющим собой имя временного файла, содержащего данные для отображения. Получается, что браузер можно настроить на обработку практически любого типа файлов, не внося в него никаких изменений. Современные веб-серверы часто содержат сотни комбинаций типов и подтипов файлов. Новые типы файлов появляются всякий раз при установке новых программ.

Вспомогательные приложения не обязательно связаны только с MIME-типом *application* (приложение). Например, Adobe Photoshop будет работать с *image/x-photoshop*, а RealOne Player может поддерживать *audio/mp3*.

В Windows каждая устанавливаемая на компьютер программа регистрирует типы, которые она хочет поддерживать. Такой механизм приводит к конфликту, когда несколько программ могут обрабатывать один и тот же подтип, например, *video/mpg*. Конфликт разрешается следующим образом: программа, которая регистрируется последней, затирает своей записью существующую ассоциацию (тип MIME, вспомогательное приложение) для тех типов, которые она готова обрабатывать самостоятельно. Следствием этого является то, что каждая уста-

навливаемая программа может изменить метод отображения браузером некоторых типов.

В UNIX этот процесс обычно не является автоматическим. Пользователь должен вручную обновлять конфигурационные файлы. Такой подход приводит к уменьшению числа неожиданных сюрпризов, но при этом у пользователя появляются дополнительных заботы.

Браузеры могут работать и с локальными файлами, не запрашивая информацию с удаленных серверов. Поскольку локальные файлы не сообщают свои MIME-типы, браузеру нужно каким-то хитрым образом определить, какой подключаемый модуль или вспомогательное приложение использовать для типов, отличных от встроенных (таких, как *text/html* или *image/jpeg*). Для поддержки локальных файлов вспомогательные модули и приложения должны быть ассоциированы как с расширениями файлов, так и с типами MIME. При стандартных настройках попытка открыть файл foo.pdf приведет к его открытию в Acrobat, а файл bar.doc будет открыт в Word. Некоторые браузеры для определения типа MIME используют не только данные о типе MIME, но и расширение файла и даже данные, взятые из самого файла. В частности, Internet Explorer по возможности старается ориентироваться на расширение файла, а не на тип MIME.

Здесь также возможны конфликты, поскольку многие программы страстно желают поддерживать, например, .mpg. Профессиональные программы при установке обычно выводят флажки, позволяющие выбрать поддерживаемые типы MIME и расширения. Таким образом, пользователь может выбрать то, что ему требуется, и таким образом избежать случайного затирания существующих ассоциаций. Программы, нацеленные на массового потребителя, полагают, что большинство пользователей понятия не имеют о типах MIME и просто захватывают все что могут, совершенно не обращая внимания на ранее установленные программы.

Расширение возможностей браузера по поддержке новых типов файлов — это удобно, но может также привести к возникновению некоторых проблем. Когда Internet Explorer получает файл с расширением exe, он думает, что это исполняемая программа и никаких вспомогательных средств для нее не требуется. Очевидно, следует просто запустить программу. Однако такой подход может оказаться серьезной дырой в системе защиты информации. Злоумышленнику требуется лишь создать нехитрый сайт с фотографиями, скажем, знаменитых киноактеров или спортсменов и поставить ссылки на вирусы. Один-единственный щелчок мышкой на фотографии может в этом случае привести к запуску непредсказуемой и, возможно, опасной программы, которая будет действовать на машине пользователя. Для предотвращения подобных нежелательных ситуаций Internet Explorer можно настроить на избирательный запуск неизвестных программ, однако не все пользователи в состоянии справиться с настройкой браузера.

В UNIX похожая проблема может возникнуть с основными сценариями, однако при этом требуется, чтобы пользователь сознательно стал устанавливать вспомогательную программу. К счастью, это процесс довольно сложный, и «случайно» установить что-либо практически невозможно (немногие могут сделать это и преднамеренно).

Сторона сервера

О стороне клиента сказано уже достаточно много. Поговорим теперь о стороне сервера. Как мы уже знаем, когда пользователь вводит URL или щелкает на гиперссылке, браузер производит структурный анализ URL и интерпретирует часть, заключенную между *http://* и следующей косой чертой, как имя DNS, которое следует искать. Вооружившись IP-адресом сервера, браузер устанавливает TCP-соединение с портом 80 этого сервера. После этого отсылается команда, содержащая оставшуюся часть URL, в которой указывается имя файла на сервере. Сервер возвращает браузеру запрашиваемый файл для отображения.

В первом приближении веб-сервер напоминает сервер, представленный в листинге 6.1. Этому серверу, как и настоящему веб-серверу, передается имя файла, который следует найти и отправить. В обоих случаях в основном цикле сервер выполняет следующие действия:

1. Принимает входящее TCP-соединение от клиента (браузера).
2. Получает имя запрашиваемого файла.
3. Получает файл (с диска).
4. Возвращает файл клиенту.
5. Разрывает TCP-соединение.

Современные веб-серверы обладают более широкими возможностями, однако существенными в их работе являются именно перечисленные шаги.

Проблема данного подхода заключается в том, что каждый запрос требует обращения к диску для получения файла. В результате число обращений к веб-серверу за секунду ограничено максимальной скоростью обращений к диску. Среднее время доступа к высокоскоростному диску типа SCSI составляет около 5 мс, то есть сервер может обрабатывать не более 200 обращений в секунду. Это число даже меньше, если часто запрашиваются большие файлы. Для крупных веб-сайтов это слишком мало.

Очевидным способом решения проблемы является кэширование в памяти n последних запрошенных файлов. Прежде чем обратиться за файлом к диску, сервер проверяет содержимое кэша. Если файл обнаруживается в кэше, его можно сразу выдать клиенту, не обращаясь к диску. Несмотря на то, что для эффективного кэширования требуются большие объемы памяти и некоторое дополнительное время на проверку кэша и управление его содержимым, суммарный выигрыш во времени почти всегда оправдывает эти накладные расходы и стоимость.

Следующим шагом, направленным на повышение производительности, является создание многопоточных серверов. Одна из реализаций подразумевает, что сервер состоит из входного модуля, принимающего все входящие запросы, и k обрабатывающих модулей, как показано на рис. 7.9. Все $k + 1$ потоков принадлежат одному и тому же процессу, поэтому у обрабатывающих модулей есть доступ к кэшу в адресном пространстве процесса. Когда приходит запрос, входящий модуль принимает его и создает краткую запись с его описанием. Затем запись передается одному из обрабатывающих модулей. Другая возможная реализация подразумевает отсутствие входного модуля; все обрабатывающие модули пыта-

ются получить запросы, однако здесь требуется блокирующий протокол, помогающий избежать конфликтов.

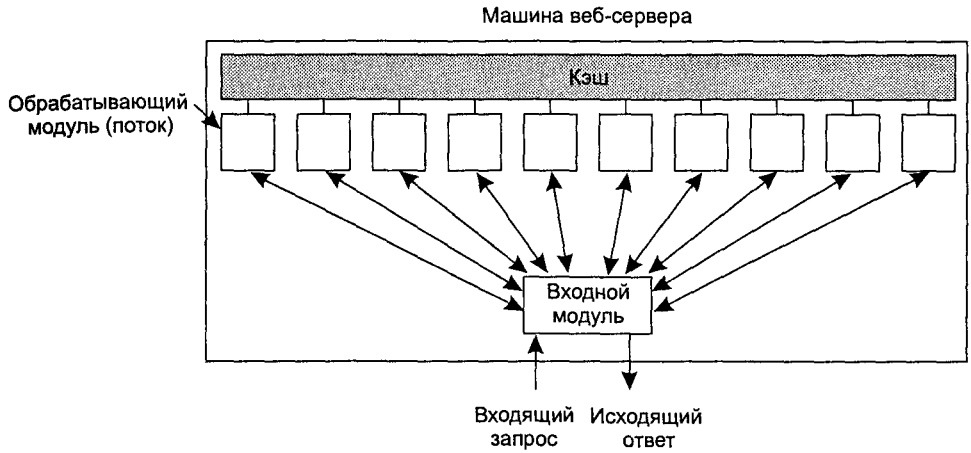


Рис. 7.9. Многопоточный веб-сервер с входным и обрабатывающими модулями

Обрабатывающий модуль вначале проверяет кэш на предмет нахождения там нужных файлов. Если они там действительно есть, он обновляет запись, включая в нее указатель на файл. Если искомого файла в кэше нет, обрабатывающий модуль обращается к диску и считывает файл в кэш (при этом, возможно, затирая некоторые хранящиеся там файлы, чтобы освободить место). Считанный с диска файл попадает в кэш и отсылается клиенту.

Преимущество такой схемы заключается в том, что пока один или несколько обрабатывающих модулей заблокированы в ожидании окончания дисковой операции (при этом такие модули не потребляют мощности центрального процессора), другие модули могут активно обрабатывать захваченные ими запросы. Разумеется, реального повышения производительности за счет многопоточной схемы можно достичь, только если установить несколько дисков, чтобы в каждый момент времени можно было обращаться более чем к одному диску. Имея k обрабатывающих модулей и k дисков, производительность можно повысить в k раз по сравнению с однопоточным сервером и одним диском.

Теоретически, однопоточный сервер с k дисками тоже должен давать прирост производительности в k раз, однако программирование и администрирование такой схемы оказывается очень сложным, так как в этом случае невозможно использование обычных блокирующих системных вызовов READ для чтения с диска. Многопоточные серверы такого ограничения не имеют, поскольку READ будет блокировать только тот поток, который осуществил системный вызов, а не весь процесс.

Современные веб-серверы выполняют гораздо больше функций, чем просто прием имен файлов и отправка файлов. На самом деле, реальная обработка каждого запроса может оказаться довольно сложной. По этой причине на многих серверах каждый обрабатывающий модуль выполняет серии действий. Входной

модуль передает каждый входящий запрос первому доступному модулю, который обрабатывает его путем выполнения некоторого подмножества указанных далее шагов в зависимости от того, что именно требуется для данного запроса:

- вычисление имени запрашиваемой веб-страницы;
- регистрация клиента;
- осуществление контроля доступа для клиента;
- осуществление контроля доступа для веб-страницы;
- проверка кэша;
- получение запрошенной страницы с диска;
- определение типа MIME для включения этой информации в ответ клиенту;
- аккуратное выполнение различных дополнительных задач;
- возвращение ответа клиенту;
- добавление записи в журнал активности сервера.

Шаг 1 необходим, потому что входящий запрос может и не содержать реального имени файла в виде строкового литерала. Например, URL может быть вот таким: http://www.cs.vu.nl. Здесь имя файла отсутствует. Этот URL необходимо дополнить неким именем файла по умолчанию. К тому же современные браузеры могут указывать язык пользователя по умолчанию (например, итальянский или английский), что позволяет серверу выбирать веб-страницу на соответствующем языке, если таковая существует. Вообще говоря, расширение имени — задача не такая уж тривиальная, как может показаться, поскольку существует множество соглашений об именовании файлов.

Шаг 2 состоит в проверке идентификационных данных клиента. Это нужно для отображения страниц, недоступных для широкой публики. Мы обсудим один из способов такой проверки далее в этой главе.

Шаг 3 проверяет наличие каких-либо ограничений, накладываемых на данного клиента и его местоположение. На шаге 4 проверяются ограничения на доступ к запрашиваемой странице. Если определенный файл (например, .htaccess) присутствует в том же каталоге, что и нужная страница, он может ограничивать доступ к файлу. Например, можно установить доступ к странице только для сотрудников компании.

Шаги 5 и 6 включают в себя получение страницы. Во время выполнения шага 6 должна быть обеспечена возможность одновременного чтения с нескольких дисков.

Шаг 7 связан с определением типа MIME, исходя из расширения файла, первых нескольких байтов, конфигурационного файла или каких-то иных источников. Шаг 8 предназначен для различных задач, таких как построение профиля пользователя, сбор статистики и т. д.

На шаге 9 наконец отсылается результат, что фиксируется в журнале активности сервера на шаге 10. Последний шаг требуется для нужд администрирования. Из подобных журналов можно впоследствии узнать ценную информацию

о поведении пользователей — например, о том, в каком порядке люди посещают страницы на сайте.

Если приходит слишком много запросов в секунду, центральный процессор может перестать справляться с их обработкой вне зависимости от того, сколько дисков параллельно работают на сервере. Решается эта проблема установкой на сервере нескольких узлов (компьютеров). Их полезно укомплектовывать реплицированными (содержащими одинаковую информацию) дисками во избежание ситуации, когда узким местом становится дисковый накопитель. В результате возникает многомашинная система, организованная в виде **серверной фермы** (рис. 7.10). Входной модуль по-прежнему принимает входящие запросы, однако распределяет их на сей раз не между потоками, а между центральными процессорами, снижая тем самым нагрузку на каждый компьютер. Отдельные машины сами по себе могут быть многопотоковыми с конвейеризацией, как и в рассматриваемом ранее случае.

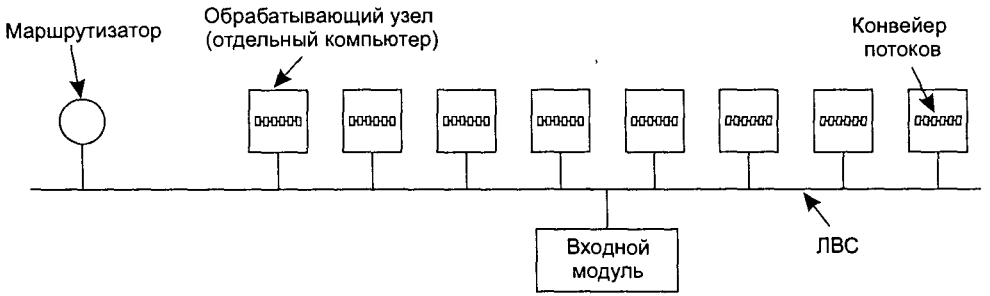


Рис. 7.10. Серверная ферма

Одна из проблем, связанных с серверными фермами, заключается в отсутствии общего кэша — каждый обрабатывающий узел обладает собственной памятью. Эта проблема может быть решена установкой дорогостоящей мультипроцессорной системы с разделяемой памятью, однако существует и более дешевый способ. Он заключается в том, что входной модуль запоминает, на какой узел он посылал запросы конкретных страниц. Последующие запросы тех же страниц он сможет тогда направлять на те же узлы. Таким образом, получается, что каждый узел специализируется по своему набору страниц; и отпадает необходимость хранения одних и тех же файлов в кэшах разных компьютеров.

Другая проблема, возникающая при использовании серверных ферм, состоит в том, что TCP-соединение клиента заканчивается на входном модуле, то есть ответ в любом случае должен пройти через входной модуль. Эта ситуация показана на рис. 7.11, *а*. Здесь как входящий запрос (1), так и исходящий ответ (2) проходят через входной модуль. Иногда для обхода этой проблемы применяется хитрость под названием **передача TCP**. Суть ее в том, что TCP-соединение продлевается до конечного (обрабатывающего) узла, и он может самостоятельно отправить ответ напрямую клиенту (рис. 7.11, *б*). Эта передача соединения для клиента незаметна.

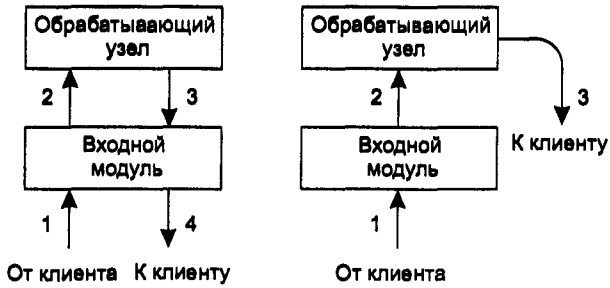


Рис. 7.11. Обычный запрос-ответный обмен (а); обмен запросами и ответами при передаче TCP (б)

URL — унифицированные указатели информационных ресурсов

Мы несколько раз упоминали о том, что веб-страницы могут быть связаны между собой ссылками. Пора познакомиться с тем, как эти ссылки реализованы. Уже при создании Паутины было очевидно, что для реализации ссылок с одних страниц на другие необходим механизм именования и указания расположения страниц. В частности, прежде чем выводить выбранную страницу на экран, нужно узнать ответы на три следующих вопроса.

1. Как называется эта страница?
2. Где она расположена?
3. Как получить к ней доступ?

Если бы каждой странице можно было присвоить уникальное имя, то в идентификации страниц не было бы никакой неоднозначности. Тем не менее, проблему бы это не решило. Для примера проведем параллель между страницами и людьми. В Соединенных Штатах почти у всех граждан есть номер карточки социального страхования, представляющий собой уникальный идентификатор, так как нет двух людей с одинаковым номером. Тем не менее, зная только номер карточки социального страхования, нет способа узнать адрес владельца и, конечно, нельзя определить, следует ли писать этому гражданину по-английски, по-испански или по-китайски. Во Всемирной паутине проблемы, в принципе, те же самые.

В результате было принято решение идентифицировать страницы способом, решающим сразу все три проблемы. Каждой странице назначается унифицированный указатель информационного ресурса (**URL**, Uniform Resource Locator), который служит уникальным именем страницы. URL состоят из трех частей: протокола (также называемого **схемой**), DNS-имени машины, на которой расположена страница, и локального имени, единственным образом идентифицирующего страницу в пределах этой машины (обычно это просто имя файла). Например, веб-сайт факультета, на котором работает автор, содержит несколько видеофрагментов об университете и городе Амстердаме. Унифицированный указатель страницы с видео выглядит следующим образом:

http://www.cs.vu.nl/video/index-en.html

Этот URL состоит из трех частей: протокола (*http*), DNS-имени хоста (*www.cs.vu.nl*) и имени файла (*video/index-en.html*). Отдельные части URL-указателя разделяются специальными знаками пунктуации. Имя файла представляет собой относительный путь по отношению к веб-катологу *cs.vu.vu.nl.*

У сайтов могут быть сокращенные имена для ускоренного доступа к определенным файлам. Скажем, при отсутствии в URL имени файла может выводиться главная (домашняя) страница сайта. Если имя файла заканчивается именем каталога, то из него по умолчанию выбирается файл с именем *index.html.* Наконец, имя *~user/* может соответствовать WWW-каталогу пользователя, причем может быть также задано имя файла по умолчанию, например, *index.html.* Так, на домашнюю страницу автора можно попасть по адресу

http://www.cs.vu.nl/~ast/

несмотря на то, что действительное имя файла (*index.html*) отличается от указанного.

Теперь надо понять, как работает гипертекст. Чтобы на неком участке текста браузер мог реагировать на щелчок мыши, при написании веб-страницы нужно обозначить два элемента: отображаемый на экране текст ссылки и URL страницы, которая должна стать текущей при щелчке мышью. Синтаксис такой команды будет пояснен далее в этой главе.

При выборе ссылки браузер с помощью службы DNS ищет имя хоста. Зная IP-адрес хоста, браузер устанавливает с ним TCP-соединение. По этому соединению с помощью указанного протокола браузер посылает имя файла, содержащего страницу. Вот, собственно, и все. Назад по соединению передается страница.

Такая схема является открытой в том смысле, что она позволяет использовать разные протоколы для доставки информационных единиц разного типа. Определены URL-указатели для других распространенных протоколов, понимаемые многими браузерами. Слегка упрощенные формы наиболее употребительных URL-указателей приведены в табл. 7.9.

Таблица 7.9. Некоторые распространенные URL-указатели

Имя	Применение	Пример
http	Гипертекст (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Локальный файл	file:////usr/suzanne/prog.c
news	Телеконференция	news:comp.os.minix
news	Статья новостей	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Отправка электронной почты	mailto:JohnUser@acm.org
telnet	Удаленный терминал	telnet://www.w3.org:80

Кратко рассмотрим этот список. Протокол *http* является родным языком Всемирной паутины, на нем разговаривают веб-серверы. **HTTP** — это сокращение, которое расшифровывается как HyperText Transfer Protocol (протокол передачи гипертекста). Более подробно мы рассмотрим его далее в этой главе.

Протокол *ftp* используется для доступа к файлам по FTP — протоколу передачи файлов по Интернету. За двадцать лет своего существования он достаточно хорошо укоренился в сети. Многочисленные FTP-серверы по всему миру позволяют пользователям в любых концах Интернета регистрироваться на сервере и скачивать разнообразные файлы, размещенные на сервере. Всемирная паутина здесь не вносит особых изменений. Она просто упрощает доступ к FTP-серверам и работу с файлами, ибо само по себе FTP имеет несколько загадочный интерфейс (однако более мощный, чем HTTP: например, он позволяет пользователю машины *A* передать файл с машины *B* на машину *C*).

К локальному файлу также можно обратиться как к веб-странице, либо используя протокол *file*, либо просто указав имя файла. Такой подход напоминает FTP, но не требует наличия сервера. Разумеется, он работает только с локальными файлами, а не с расположенными на удаленных терминалах.

Задолго до появления Интернета появилась система групп новостей USENET. Она состоит примерно из 30 000 конференций, в которых миллионы людей обсуждают широкий круг вопросов, отправляя и читая сообщения, связанные с тематикой данной конференции. Протокол *news* позволяет пользователю вызывать на экран статью с новостями, как если бы она была обычной веб-страницей. Это означает, что веб-браузер легким движением руки превращается в элегантную программу чтения новостей. На самом деле, благодаря кнопкам и пунктам меню многих браузеров чтение новостей USENET становится даже удобнее, чем с помощью специальных программ чтения сетевых новостей.

Для протокола *news* поддерживается два формата URL-указателей. Первый формат указывает телеконференцию, и с его помощью можно получить список новых статей с указанного заранее сайта новостей. Второй формат позволяет получить конкретную статью по ее идентификатору, например, *AA0134223112@cs.utah.edu*. Для получения этой статьи с заранее настроенного сайта браузер использует протокол NNTP (Network News Transfer Protocol — сетевой протокол передачи новостей). Мы изучим NNTP в этой книге, однако надо понимать, что это нечто вроде SMTP, они весьма похожи даже по стилю.

Протокол *gopher* используется системой Gopher, разработанной в университете штата Миннесота и получившей свое название от университетской спортивной команды «Golden Gophers» («Золотые суслики»). (Гоферами называют уроженцев штатов Миннесота, Арканзас и Флорида. Кроме того, на американском сленге это слово означает «добывать», «копать», «искать».) Система Gopher появилась в Интернете на несколько лет раньше Всемирной паутины. Концептуально они похожи: и та, и другая представляют собой схему поиска и получения информации, хранящейся на различных серверах, однако система Gopher поддерживала только тексты и не поддерживала изображений. Сейчас ее можно считать полностью устаревшей, используется она крайне редко.

Последние два протокола не занимаются имитацией получения веб-страниц, но они также полезны. Протокол *mailto* позволяет пользователю посылать электронную почту из веб-браузера. Например, в некоторых браузерах для этого нужно щелкнуть на кнопке OPEN и ввести URL-указатель, состоящий из слова *mailto:*, за которым следует почтовый адрес получателя. В ответ в большинстве

браузеров откроется специальная форма, содержащая поля для редактирования темы письма и других заголовков, а также окно для ввода текста самого письма.

С помощью протокола *telnet* можно установить в подключенном режиме соединение с удаленным компьютером. Он используется так же, как и программа Telnet, что неудивительно, так как большинство браузеров просто вызывают саму программу Telnet как вспомогательное приложение.

Итак, URL-указатели позволяют пользователям не только путешествовать по Всемирной паутине, но и работать с FTP-серверами, BBS, Gopher-серверами, электронной почтой и регистрироваться на удаленных серверах с помощью программы Telnet. Все эти ресурсы оказываются доступны при помощи всего одной программы — веб-браузера, что очень удобно. Если бы отцом этой идеи не был ученый-физик, она стала бы, вероятно, самой убедительной рекламой какой-нибудь компании, специализирующейся на выпуске программного обеспечения.

Несмотря на все перечисленные достоинства, все продолжающийся рост популярности Всемирной паутины выявил один врожденный недостаток URL-схемы. URL указывает на определенный хост. Часто запрашиваемые по сети страницы было бы лучше дублировать и хранить копии в удаленных концах сети, чтобы снизить сетевой трафик. Беда в том, что URL-указатели не предоставляют возможности для ссылки на страницу без указания ее точного адреса. Нельзя сказать: «Мне нужна страница abc, и мне все равно, где вы ее раздобудете». Для решения задачи репликации страниц проблемная группа проектирования Интернета IETF (Internet Engineering Task Force) работает над системой **URN** (Uniform Resource Name — универсальное имя ресурса). Универсальное имя ресурса URN можно считать обобщенным URL-указателем. В настоящее время этот вопрос находится в стадии исследования, хотя уже предложен синтаксис, описанный в RFC 2141.

Принцип отсутствия состояний и cookie-файлы

Как мы уже неоднократно повторяли, Всемирная паутина изначально не может находиться в каком-либо состоянии. Отсутствует понятия сеанса связи. Браузер клиента посылает запрос на сервер и получает в ответ файл. После этого сервер забывает о том, что он когда-либо видел этого клиента.

Вначале, когда Паутина использовалась исключительно для получения документов, предназначенных для широкой публики, такая модель была адекватна. Но по мере наращивания функциональности стали появляться различные проблемы. К примеру, на некоторых сайтах требуется регистрация пользователей; а иногда получение информации с сайта является платным. Возникает вопрос различения запросов от зарегистрированных пользователей и от всех остальных. Вторым примером является электронная коммерция. Как серверу собрать информацию о состоянии корзины пользователя, если тот время от времени пополняет ее содержимое? Третий пример — веб-порталы типа Yahoo. Пользователям предоставляется возможность настраивать вид начальной страницы (например, так, чтобы им сразу показывались последние новости о любимой спортивной команде или курсы ценных бумаг). Однако как сервер сможет корректно отобразить страницу, если он не знает, с каким пользователем он имеет дело?

На первый взгляд может показаться, что решение очевидно: сервер может различать пользователей по их IP-адресам. Однако эта идея не работает. Во-первых, многие пользователи работают на компьютерах с разделяемыми ресурсами (например, сотрудники компании), и с помощью IP-адреса можно идентифицировать лишь компьютер, а не пользователя. Во-вторых, что еще хуже, многие провайдеры используют NAT, поэтому все исходящие пакеты от тысяч клиентов имеют один и тот же IP-адрес.

Для решения этой проблемы фирмой Netscape был предложен сильно раскритикованный метод **cookie-файлов**. Такое название произошло от старинного программистского сленгового словечка. Программа вызывала процедуру и получала взамен нечто, что могло понадобиться впоследствии для выполнения какой-либо задачи. Это «нечто» и называлось cookie. В этом смысле файловый дескриптор в UNIX и дескриптор объектов в Windows можно рассматривать как cookie. Эти специальные маркеры позднее были формализованы в RFC 2109.

Когда пользователь запрашивает страницу, сервер может снабдить свой ответ дополнительной информацией, которая может включать в себя cookie, то есть маркер, представляющий собой маленький (до 4 Кбайт) файл (или строку). Браузеры сохраняют полученные маркеры в специальном каталоге на жестком диске, если только пользователь не отключил данную функцию. Итак, cookie — это файлы или строки, а не исполняемые программы. В принципе, в них может содержаться вирус, но поскольку они рассматриваются всего лишь как информационные данные, нет какой-либо официальной возможности для активации вирусов и нанесения ущерба системе. Однако у хакера всегда есть возможность, используя ошибки в браузере, заставить активироваться вирусы, содержащиеся в cookie.

В маркерах cookie может содержаться до пяти полей, как показано в табл. 7.10. Поле *Домен* содержит имя домена, с которого пришел маркер. Предполагается, что браузеры проверяют тот факт, что серверы не лгут относительно имен доменов. Каждый домен может хранить не более 20 маркеров, связанных с одним клиентом. Поле *Путь* содержит путь в структуре каталогов на сервере, указывающий те части дерева каталогов, которые могут использовать маркер. Часто в этом поле содержится знак «/», означающий, что доступно дерево целиком.

Таблица 7.10. Примеры cookie

Домен	Путь	Содержимое	Годен до	Защищенный
toms-casino.com	/	CustomerID=497793521	15-10-02 17:00	Да
joes-store.com	/	Cart=1-00501;1-07031;2-13721	11-10-02 14:22	Нет
aportal.com	/	Prefs=Stk;SUNW+ORCL;Spt:Jets	31-12-10 23:59	Нет
sneaky.com	/	UserID=362723910	31-12-12 23:59	Нет

Поле *Содержимое* имеет вид *имя* = *значение*. Как *имя*, так и *значение* могут быть совершенно произвольными, на усмотрение сервера. В этом поле хранится основная информация, которую несет в себе маркер.

Поле *Годен до* указывает срок годности маркера. Если это поле отсутствует, браузер отбрасывает cookie сразу после выхода из программы. Такой маркер называется **неустойчивым**. Если же указано время и дата, то о таком маркере говорят, что он **устойчивый**. Он хранится до тех пор, пока не выйдет срок годности. Время, указываемое в поле *Годен до*, — гринвичское. Чтобы удалить cookie с жесткого диска клиента, сервер просто посылает его заново, указывая вышедший срок годности.

Наконец, поле *Защищенный* может быть установлено для индикации того, что браузер может вернуть его только на защищенный сервер. Это свойство используется в электронной коммерции, банковском деле и других приложениях, в которых важна защита информации.

Итак, мы узнали о получении маркеров. Как же они используются? Непосредственно перед отправкой браузером запроса на получение страницы на какой-нибудь веб-сайт проверяется каталог с cookie. Ищутся маркеры, пришедшие с того домена, на который отправляется запрос. Все найденные маркеры отправляются вместе с запросом. Получив их, сервер может интерпретировать содержащуюся в них информацию так, как ему нужно.

Рассмотрим возможные применения cookie. В табл. 7.9 первый cookie был прислан доменом *toms-casino.com* и используется для идентификации клиента. Если через неделю тот же клиент возвращается на сайт, чтобы просадить очередную сумму денег, браузер отправляет cookie, так что сервер узнает старого знакомого. Вооружившись идентификатором пользователя, сервер может найти запись о нем в базе данных и использовать эту информацию для генерации соответствующей веб-страницы. В зависимости от азартности игрока ему может быть предложен покер, таблица результатов сегодняшних скачек или просто торговый автомат.

Второй cookie-маркер пришел с *joes-store.com*. Сценарий в этом случае заключается в том, что клиент бродит по магазину, выбирая себе покупки. Найдя что-нибудь привлекательное, он щелкает на значке товара. При этом сервер создает cookie-маркер, содержащий количество и идентификатор заказанного товара, и отсылает его клиенту. Клиент продолжает бродить по электронному магазину, и в ответ на каждый новый запрос страницы ему отсылается cookie. По мере накопления выбранных товаров дополняется информация в cookie. В таблице показано, что в корзине клиента содержатся три вида товаров, причем заказано два экземпляра товара третьего вида. Наконец, когда пользователь щелкает *ПЕРЕЙТИ К РАСЧЕТАМ*, cookie, содержащий теперь уже полную информацию о покупках, отсылается вместе с запросом на сервер. Таким образом, серверу точно известно, какие товары заказал клиент.

Третий cookie-маркер прибыл с веб-портала. Когда пользователь щелкает на ссылке на портал, браузер отсылает ему cookie, в котором говорится о том, что надо показать страницу, содержащую котировки акций Sun Microsysems и Oracle, а также результаты футбольного матча New York Jets. Так как максимальный размер cookie-файла равен 4 Кбайт, то остается еще много места для более детальной настройки страницы. Например, в нее можно включить сводку погоды, специальные предложения, заголовки статей в крупных газетах и т. п.

Cookie могут использоваться и для нужд самого сервера. Например, с их помощью можно отслеживать число различных посетителей сайта, узнавать, сколько страниц просмотрел каждый из них, и составлять по этим данным статистику. Когда на сервер приходит первый запрос от пользователя, вместе с ним, разумеется, не высылается никакой маркер. Поэтому сервер отсылает обратно cookie со значением счетчика, равным 1. Последующие переходы между страницами сайта уже будут сопровождаться отсылкой cookie. Всякий раз счетчик будет инкрементироваться и отсылаться пользователю. Таким образом, по счетчикам можно узнать, сколько пользователей покинуло сайт, просмотрев только первую страницу, сколько посетителей просматривают по две страницы, и т. д.

Cookie-файлы могут использоваться и не по прямому назначению. Теоретически, они могут отсылаться только на тот сайт, которым они были порождены, однако они могут попадать и в руки хакеров, использующих многочисленные ошибки браузеров. Поскольку некоторые сайты, посвященные электронной коммерции, указывают в cookie номера кредитных карт, нежелательное их использование может нанести серьезный ущерб.

Есть и противоположный вариант использования cookie — для незаметного сбора информации о сайтах, наиболее часто посещаемых данным клиентом. Это делается так. Рекламное агенство, скажем, «Черный Рекламщик», связывается с крупнейшими веб-сайтами и размещает на них рекламные баннеры своих корпоративных клиентов, за что сайту выплачиваются денежные взносы. Вместо того, чтобы предоставлять сайту GIF или JPEG с рекламой, ему дается URL, который следует поместить на всех страницах. Каждый из этих URL содержит уникальный идентификатор в виде имени файла, например:

http://www.sneaky.com/382674902342.gif

Когда пользователь впервые посещает страницу *P*, содержащую такую рекламу, браузер, как водится, принимает HTML-файл. Просматривая его, браузер находит ссылку на изображение на *www.sneaky.com*. Разумеется, он отправляет запрос на получение изображения. Вместе с GIF приходит cookie с уникальным идентификатором пользователя, 362723910 (см. табл. 7.9). «Черный Рекламщик» отмечает, таким образом, тот факт, что пользователь с таким идентификатором посетил страницу *P*. Это делается очень просто, так как ссылка на запрошенный файл (*382674902342.gif*) существует, на самом деле, только на странице *P*. Конечно, одна и та же реклама может располагаться на тысячах разных страниц, но каждая из них имеет свое имя файла. При этом за выпуск каждого экземпляра рекламная компания может взимать с заказчика небольшую сумму.

Затем пользователь может оказаться на другой странице, содержащей баннер от «Черного Рекламщика». Скачав HTML-файл с сервера, браузер видит ссылку на изображение с именем, скажем, http://www.sneaky.com/493654919923.gif и запрашивает его. Поскольку с домена *sneaky.com* уже был получен cookie, браузер отсылает его обратно с идентификатором пользователя. Так «Черный Рекламщик» (ЧР) узнает о том, что пользователь посетил вторую страницу с его рекламой.

Со временем ЧР может составить подробное описание пристрастий пользователя, при этом вовсе не обязательно, чтобы тот щелкал на баннерах. Конечно, остается неизвестным имя пользователя (хотя имеется IP-адрес, и этого может

оказаться достаточно для вычисления имени с помощью баз данных). Однако стоит пользователю указать свое имя на одном из сайтов, сотрудничающих с ЧР, как появляется возможность составить и продать целое веб-досье на пользователя. Продажа таких досье оказывается делом настолько прибыльным, что ЧР выгодно сотрудничать с максимально возможным количеством сайтов и собирать как можно больше информации. Самое коварное во всем этом то, что большинство пользователей даже не подозревают о том, что за ними ведется слежка, и даже считают себя в полной безопасности, поскольку никогда не щелкают ни на каких баннерах.

И если «Черный Рекламщик» хочет стать «Суперчерным Мегарекламщиком», его объявления не должны выглядеть как обычные классические баннерные ссылки. «Объявление» размером в 1 пиксел, сливающееся по цвету с задним фоном страницы (то есть невидимое), будет иметь ровно такой же эффект при слежении за пользователями: браузер будет запрашивать gif-изображение размером 1×1 пиксел и отправлять обратно cookie.

Для самоуспокоения некоторые пользователи настраивают свои браузеры так, чтобы они отвергали любые cookie. Однако это может породить проблемы при работе с «честными» сайтами, которым действительно необходимо обмениваться с пользователями cookie-маркерами. Для решения этой проблемы иногда устанавливают программы, занимающиеся поеданием cookie. Они анализируют все приходящие маркеры и принимают либо отвергают их в зависимости от выбора пользователя (например, задается список сайтов, которым можно доверять). Это дает пользователю возможность детального контроля принимаемых cookie-файлов. Современные браузеры, такие как Mozilla (www.mozilla.org), часто имеют встроенные средства пользовательского контроля cookie.

Статические веб-документы

Основная идея Всемирной паутины состоит в перемещении веб-страниц от сервера клиенту. Простейшие веб-страницы являются статическими, то есть это просто файлы, размещенные на каком-либо сервере и ожидающие востребования. В этом контексте даже видео может быть статической страницей, поскольку это всего лишь файл. В этом разделе мы подробно рассмотрим статические веб-страницы. В следующем разделе нам предстоит изучение динамического наполнения страниц.

HTML — язык разметки веб-страниц

Веб-страницы на сегодняшний день пишутся на языке **HTML** (HyperText Markup Language). С помощью HTML можно размещать на веб-страницах текст, графику, а также указатели на другие страницы. Он является языком разметки, то есть языком, описывающим способ форматирования документа. Термин «разметка» (markup) восходит к тем дням, когда технический редактор с помощью специальной разметки указывал типографу (это такой человек когда-то был), какой шрифт использовать для печати документа. Таким образом, языки разметки содержат подробные команды форматирования. Например, в языке HTML, коман-

да **означает начало участка текста, печатаемого полужирным шрифтом, а** означает конец такого участка. Преимущество языка разметки перед языком, не имеющим явных команд форматирования, заключается в том, что браузеры для отображения страниц, написанных на этом языке, программируются довольно просто: браузер должен понимать и выполнять содержащиеся в тексте команды разметки. Среди других популярных примеров языков разметки — языки TeX и troff.

С помощью встроенных стандартизированных команд разметки в HTML-файлах становится возможным читать и переформатировать любую веб-страницу веб-браузером. Способность изменять форматирование важно, так как должна быть возможность просматривать веб-страницу, созданную на экране с установленным разрешением 1600×1200 точек при 24 битах на точку, на экране с разрешением, например, 640×320 точек при 8 битах на точку.

Далее мы приведем краткий обзор языка HTML, просто чтобы дать о нем представление. Хотя, в принципе, можно создавать HTML-документы с помощью стандартных текстовых редакторов, и многие так и делают, также есть возможность использовать специальные HTML-редакторы, берущие на себя большую часть работы (за счет снижения возможностей пользователя детально контролировать получаемый результат).

Веб-страница состоит из заголовка и тела. Вся страница размещается между командами форматирования, называемыми в языке HTML **тегами**, <html> и </html>. Впрочем, большинство браузеров правильно отобразят страницу и в отсутствие этих тегов. Как видно из рис. 7.12, *а*, заголовок веб-страницы заключен в скобки тегов <head> и </head>, а тело располагается между тегами <body> и </body>. Команды внутри тегов называют **директивами**. Большинство HTML-тегов имеют такой формат, то есть <something> помечает начало чего-либо, а </something> — его конец. Большинство браузеров предоставляют возможность просмотра исходного HTML-кода веб-страниц (пункт меню View Source или нечто подобное).

Регистр символов в тегах не имеет значения. Например, <head> и <HEAD> означают одно и то же, однако новый стандарт требует использования исключительно строчных букв. Формат самого HTML-текста, то есть расположение строк и т. д., не имеет значения. Программы обработки HTML-текстов игнорируют лишние пробелы и переносы строк, так как они все равно форматируют текст так, чтобы он помещался в заданной области отображения. Соответственно для того чтобы исходные HTML-документы легче читались, в них можно добавлять произвольное количество знаков табуляции, пробелов и символов переноса строк. И наоборот, для разделения абзацев в тексте в исходный HTML-текст недостаточно вставить пустую строку, так как она просто игнорируется браузером. В этом случае необходимо явное использование специального тега.

Некоторые теги могут иметь (именованные) параметры, называемые **атрибутами**. Например:

```
<img src="abc" alt="foobar">
```

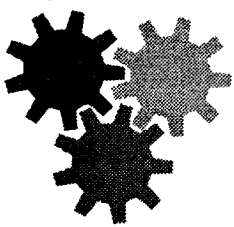
представляет собой тег <img> с атрибутом src, которому присвоено значение «abc», и атрибутом alt, которому присвоено значение «foobar». Для каждого тега стан-

дарт HTML устанавливает список допустимых атрибутов и их значение. Поскольку все атрибуты являются именованными, их порядок не имеет значения.

```
<HTML> <HEAD> <TITLE> Корпорация СООБЩЕСТВО ШТУЧЕК. </TITLE> </HEAD>
<BODY> <H1> Добро пожаловать на страницу компании СООБЩЕСТВО ШТУЧЕК. </H1>
<IMG SRC="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <BR>
Мы рады приветствовать вас на домашней странице корпорации <B> СООБЩЕСТВО ШТУЧЕК</B>
Мы надеемся, что <I> вы </I> найдете здесь всю необходимую вам информацию.
<P>Ниже приведены ссылки на информацию о нашей замечательной продукции.
Вы можете сделать заказ по Интернету, по телефону или по факсу. <HR>
<H2> Информация о продукции </H2>
<UL> <LI> <A HREF="http://widget.com/products/big"> Большущие штуковины </A>
<LI> <A HREF="http://widget.com/products/little"> Малюсенькие штучки </A>
</UL>
<H2> Номера телефонов </H2>
<UL> <LI> телефон: 1-800-WIDGETS
<LI> факс: 1-415-765-4321
</UL> </BODY> </HTML>
```

а

Добро пожаловать на страницу компании СООБЩЕСТВО ШТУЧЕК



Мы рады приввтствовать вас на домашнвй странице корпорации СООБЩЕСТВО ШТУЧЕК.
Мы надеемся, что вы найдетв здесь всю необходимую вам информацию.

Ниже приведены ссылки на информвцию о нашей замечательной продукции.
Вы можвте сделать заказ по Интврнету, по телефону или по факсу.

Информация о продукции

- Огромные штуковины
- Малюсенькие штучки

Номера талефонов

- 1-800-WIDGETS
- 1-415-785-4321

б

Рис. 7.12. Пример веб-страницы на HTML (*а*); форматированная страница (*б*)

Формально при написании HTML-документов должен использоваться набор символов Latin-1 международного стандарта ISO 8859-1, но для пользователей, чьи клавиатуры поддерживают только ASCII-символы, для ввода специальных символов, таких как, например, é, могут использоваться специальные управляющие последовательности символов. Эти последовательности должны начинаться со знака амперсанда и заканчиваться точкой с запятой. Например, è означает символ è, а é — символ é. Так как сами символы <, > и & оказываются зарезервированными, для их отображения в тексте также применяются управ-

ляющие последовательности: < (less than — знак «меньше»), > (greater than — знак «больше») и & (ampersand — амперсанд).

Главным пунктом заголовка является название страницы, располагающееся между тегами <title> и </title>. В него можно поместить также некоторую мета-информацию. Оно не отображается на странице, а используется некоторыми браузерами для того, чтобы помечать окно страницы.

Рассмотрим некоторые другие команды языка HTML, используемые в примере на рис. 7.12, и другие, приведенные в табл. 7.11. Заголовки в примере задаются тегами вида <h*n*>, где *n* — цифра от 1 до 6. <h1> является самым важным заголовком, <h6> — наименее важным. Как это отобразить на экране, зависит от браузера. Обычно заголовки с меньшими номерами отображаются более крупными шрифтами. Браузер может также выделять различные заголовки различными цветами. Обычно заголовки <h1> выводятся на экран крупным полужирным шрифтом и выделяются, по меньшей мере, одной пустой строкой над и под заголовком.

Таблица 7.11. Наиболее часто используемые HTML-теги. У некоторых из них могут быть дополнительные параметры

Тег	Описание
<html> ... </html>	Объявляет веб-страницу на языке HTML
<head> ... </head>	Определяет границы заголовка страницы
<title> ... </title>	Определяет границы неотображаемого названия страницы
<body> ... </body>	Определяет границы тела страницы
<h*n*> ... </h*n*>	Определяет границы заголовка уровня n
...	Маркирует блок текста, печатаемого полужирным шрифтом
...	Маркирует блок текста, печатаемого курсивом
...	Помечает начало и конец центрированного по горизонтали текста
...	Помечает начало и конец неупорядоченного списка
...	Помечает начало и конец упорядоченного списка
<menu> ... </menu>	Помечает границы меню
...	Маркирует начало и конец пункта меню
 	Разрыв (перевод строки)
<p>	Начало абзаца
<hr>	Горизонтальная линейка
<img src="...">	Загрузка изображения
 ... 	Определяет гиперссылку

Теги и обозначают, соответственно, полужирный шрифт (boldface) и курсив (italics). Если браузер не может отобразить полужирный шрифт или курсив, он должен применить какой-нибудь другой способ выделения, например, использовать другой цвет или инверсное отображение символов.

Язык HTML предоставляет несколько механизмов создания списков, включая вложенные списки. Тег <ul> (unordered list) начинает неупорядоченный спи-

сок. Отдельные пункты, помеченные исходном тексте тегом <li>, изображаются с маркером абзаца (обычно крупной черной точкой, •) перед собой. Тег <ol> (ordered list) означает начало упорядоченного списка. При его использовании абзацы, помеченные тегом <li>, автоматически нумеруются браузером. У списков, организованных при помощи тегов <ul> и <ol>, одинаковый синтаксис (за исключением открывающих и закрывающих тегов списков) и похожее поведение.

Теги <br>, и <hr> применяются для обозначения границ между различными участками текста. Точный формат может быть определен в *таблице стилей* (см. ниже), ассоциированной со страницей. Тег
 просто вставляет разрыв строки. Обычно браузеры не вставляют пустую строку после тега <br>. Тег , напротив, начинает новый абзац, перед которым может быть вставлена пустая строка и, возможно, добавлен отступ. (Тег отмечающий конец абзаца, существует, но на практике почти не используется. Большинство составителей HTML-страниц даже не знают о его существовании.) Наконец, тег <hr> прерывает строку и рисует на экране горизонтальную линию.

Язык HTML позволяет включать в веб-страницу изображения. Тег <img> указывает, что в данной позиции страницы должно быть загружено изображение. У этого тега может быть несколько параметров. Параметр src задает URL изображения. Стандартом HTML не определяются графические форматы. На практике все браузеры поддерживают файлы форматов GIF и JPEG. Браузеры могут поддерживать любые другие форматы, но эта свобода оказывается палкой о двух концах. Если пользователь привыкнет к браузеру, поддерживающему, скажем, формат файлов BMP, он может включить их в свои веб-страницы, а затем обнаружить, что остальные браузеры просто игнорируют всю его замечательную работу.

У тега <img> может быть еще несколько параметров. Параметр align управляет выравниванием изображения относительно текста. Он может принимать значения *top* (верх), *middle* (центр), *bottom* (низ). Параметр alt предоставляет текст, отображаемый вместо изображения, если пользователь запретил вывод изображений. Параметр ismap является флагом, указывающим, является ли данное изображение активной картой.

И наконец, мы подошли к гиперссылкам, использующим пару тегов <a> (anchor — якорь) и . У этого тега также могут быть различные параметры, из которых следует отметить href (гиперссылка, URL) и name (имя гиперссылки). Текст, располагающийся между тегами <a> и , отображается на экране. Если этот текст выбирается, браузер открывает страницу, на которую указывает гиперссылка. Между тегами <a> и можно также размещать изображение (тег <img>). В этом случае, если пользователь щелкнет на изображении, будет произведен переход по ссылке.

Для примера рассмотрим следующий фрагмент HTML-текста:

```
<a href=http://www.nasa.gov> Домашняя страница NASA </a>
```

При отображении страницы с этим фрагментом на экране появляется следующая строка:

Домашняя страница NASA

Если затем пользователь щелкнет мышью на этом тексте, браузер обратится по сети к указанному URL (http://www.nasa.gov) и попытается получить там веб-страницу и отобразить ее на экране.

В качестве второго примера рассмотрим следующую строку:

```
<a href=http://www.nasa.gov> <img src="shuttle.gif" alt=NASA"> </A>
```

При отображении этой страницы должно быть видно изображение (челночный воздушно-космический аппарат). Щелчок мышью на этом изображении будет иметь тот же результат, что и щелчок на подчеркнутом тексте в предыдущем примере. Если пользователь запретит автоматическое отображение изображений, вместо него будет показан текст «NASA».

Тег <a> может содержать параметр name, что позволяет создать гиперссылку посреди текста, на которую можно ссылаться из другого места этой же страницы. Например, некоторые веб-страницы начинаются с оглавления, состоящего из «локальных» гиперссылок. Щелчок мышью на пункте оглавления позволяет быстро переместиться в нужное место страницы.

HTML продолжает развиваться. В первых двух версиях не существовало таблиц, они были добавлены только в HTML 3.0. HTML-таблица состоит из нескольких **строк**, каждая из которых состоит из нескольких **ячеек**. Ячейка может содержать широкий спектр данных, включая текст, изображения и даже другие таблицы. Ячейки могут объединяться вместе, например, заголовок таблицы может охватывать несколько столбцов. Контроль составителей страниц над внешним видом таблиц ограничен. Последнее слово в таких вопросах, как выравнивание, стили рамок и границы ячеек, остается за браузерами.

Реализация таблицы на языке HTML показана в листинге 7.5, а ее возможное отображение браузером — на рис. 7.13. Этот пример демонстрирует несколько основных возможностей HTML-таблиц. Таблицы начинаются с тега <table>. Для описания основных свойств таблицы может быть предоставлена дополнительная информация.

Листинг 7.5. HTML-таблица

```
<html>
<head> <title> Пример страницы с таблицей </title> </head>
<body>
<table border=all rules=all>
<caption> Некоторые различия html версий </caption>
<col align=left>
<col align=center>
<col align=center>
<col align=center>
<tr> <th> Аспект <th> HTML 1.0 <th>HTML 2.0 <th> HTML 3.0 <th>HTML 4.0</tr>
<tr> <th> Гиперссылки <td> x <td> x <td> x<td> x </tr>
<tr> <th> Изображения <td> x <td> x <td> x<td> x</tr>
<tr> <th> Списки <td> x <td> x <td> x<td> x</tr>
<tr> <th> Активные карты и изображения <td> <td>x<td> x<td> x</tr>
<tr> <th> Формы <td> <td> x <td> x <td> x</tr>
<tr> <th> Математические выражения <td> <td> <td> x <td> x </tr>
<tr> <th> Панели инструментов <td> <td> <td> x <td> x </tr>
```

```
<tr> <th> Таблицы <td> <td> <td> x <td> x </tr>
<tr> <th> Доступность <td> <td> <td> <td> x </tr>
<tr> <th> Внедренные объекты <td> <td> <td> <td> <td> x </tr>
<tr> <th> Скрипты <td> <td> <td> <td> x </tr>
</table>
</body>
</html>
```

Некоторые различия версий HTML

Аспект	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0
Гиперссылки	X	X	X	X
Изображения	X	X	X	X
Списки	X	X	X	X
Активные карты и изображения		X	X	X
Формы		X	X	X
Математические выражения			X	X
Панели инструментов			X	X
Таблицы			X	X
Доступность				X
Внедренные объекты				X
Скрипты				X

Рис. 7.13. Возможное отображениа таблицы браузером

С помощью тега <caption> можно задать надпись над таблицей. Каждая строка таблицы начинается с тега <tr> (table row — строка таблицы). Отдельные ячейки помечаются тегом <th> (table header — заголовок таблицы) или <td> (table data — данные таблицы). Эти два тега позволяют создавать, соответственно, строки заголовков таблицы и обычные строки, что и показано в примере.

В таблицах также могут использоваться различные другие теги. С их помощью можно устанавливать параметры выравнивания содержимого ячеек по вертикали и горизонтали, задавать параметры границ ячеек, объединять ячейки в группы и многое другое.

HTML 4.0 отличается от предыдущих версий некоторыми новыми свойствами. Они включают в себя специальные методы доступа для людей с ограниченными возможностями, внедрение объектов (обобщение тега <img>, позволившее включать в состав страниц не только изображения, но и другие объекты), поддержку языков написания сценариев (скриптов), что дало толчок к развитию динамических страниц, и т. д.

Если веб-сайт достаточно велик по размерам и сложен настолько, что над ним работает целая группа программистов, желательно обеспечить более или менее схожий дизайн всех страниц. Эта проблема решается при помощи **таблиц стилей**. При их использовании отдельные страницы оформляются не реальными (физическими) стилями (например, курсивом или полужирным шрифтом), а логическими. Среди них могут быть, например, <dn> (Определение), (Слабое выделение), (Сильное выделение), <var> (Программная переменная). Логические стили определяются в таблицах стилей, ссылка на которые ставится в начале ко-

да каждой страницы. Таким образом можно придать всем страницам единый вид. Если веб-мастер решит изменить стиль и отображать его полужирным шрифтом величиной 18 пунктов радикального розового цвета вместо курсива величиной 14 пунктов синего цвета, ему необходимо будет лишь поменять одно определение в таблице стилей, чтобы изменения распространились на весь сайт. Таблицы стилей можно сравнить с файлами в языке C, включаемыми с помощью директивы #include, — там также изменение одного макроопределения влечет за собой изменение во всех программных файлах, использующих данный заголовочный файл.

Формы

Первая версия языка HTML фактически обеспечивала лишь одностороннюю связь. Пользователи могли получать страницы от поставщиков информации, но отправлять информацию обратно было довольно трудно. По мере того, как все больше коммерческих организаций начали использовать Всемирную паутину, потребность в двустороннем трафике стала возрастать. Так, многим компаниям потребовалась возможность принимать заказы на продукцию с помощью своих веб-страниц. Производители программного обеспечения хотели продавать по сети свои программы, для чего требовалась возможность заполнения регистрационных карточек. Компаниям, предоставляющим поиск информации в Паутине, было нужно, чтобы пользователи могли ввести ключевые слова поиска.

В результате в HTML 2.0 были включены **формы**. Формы могут содержать кнопки и поля для ввода текста, позволяющие пользователям делать выбор или вводить необходимую информацию, которую затем можно отсылать владельцу страницы. Для этой цели используется тег <input>. У него могут быть различные параметры, определяющие размер, назначение и другие свойства отображаемого окна. Наиболее часто используемыми формами являются пустые поля для ввода текста, флажки, переключатели, активные карты и кнопки подтверждения. В примере, приведенном в листинге 7.6, показаны некоторые из перечисленных форм. Отображение этой HTML-страницы браузером показано на рис. 7.14.

Листинг 7.6. HTML-текст для бланка заказа

```
<html>
<head> <title> Бланк заказа клиента awi </title> </head>
<body>
<h1> Бланк заказа штуковины </h1>
<form action="http://www.widget.com/cgi-bin/widgetorder" method=post>
Имя <input name="customer" size=60> <p>
Адрес <input name="address" size=58> <p>
Город <input name="city" size=21>
Штат <input name="state" size=4>
Страна <input name="country" size=10> <p>
№ кредитной карты <input name="cardno" size=10>
Срок действия <input name="expires" size=4>
m/c <input name="cc" type=radio value="mastercard">
visa <input name="cc" type=radio value="visacard"> <p>
Размер штуковины:  большой <input name="product" type=radio value="дорогая">
маленький <input name="product" type=radio value="дешевая">
```

```
Доставка  <input name="express" type=checkbox> <p>
<input type=submit value="отправить заказ"> <p>
Благодарим вас за то, что вы заказали у нас штуковины фирмы awi. Это правильный выбор!
</form>
</body>
</html>
```

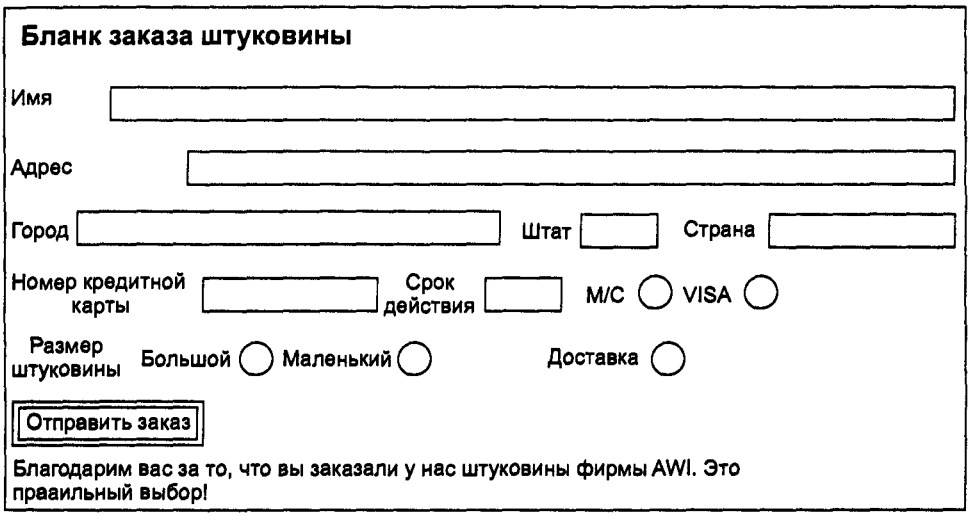


Рис. 7.14. Форматированная страница

Начнем изучение форм с этого примера. Как и все формы, она заключена между тегами <form> и </form>. Текст, не заключенный в теги, просто отображается. Внутри формы разрешено использование всех обычных тегов (например, <b>). В данной форме используются три типа окон для ввода данных.

Окно первого типа следует за текстом «Имя». Ширина этого окна 46 символов. Предполагается, что пользователь введет здесь свое имя, которое будет храниться в виде текстовой строки в переменной customer для последующей обработки. Тег <p> указывает браузеру, что последующий текст следует отображать с новой строки, даже если в текущей строке еще достаточно места. С помощью этого и других тегов автор страницы может управлять внешним видом бланка на экране.

В следующих окнах формы спрашивается адрес заказчика, то есть улица, город, штат и страна. Между этими полями не вставляются теги <p>, поэтому браузер по возможности пытается отобразить их все в одной строке. С точки зрения браузера, этот абзац представляет собой просто шесть отдельных элементов — три строки, перемежающиеся тремя окнами. Он отображает их друг за другом, слева направо, переходя на новую строку по мере необходимости. Таким образом, вполне возможно, что на экране с разрешением 1600×1200 точек все три строки и соответствующие им окна поместятся в одну строку, тогда как на экране с разрешением 1024×768 точек они будут разбиты на две строки. В худшем

случае слово «Страна» может оказаться в конце одной строки, а соответствующее окно ввода — в начале следующей строки.

В следующей строке у пользователя запрашивается номер кредитной карты и срок ее действия. Передавать номера кредитных карт по Интернету следует только в том случае, если приняты все соответствующие меры предосторожности. Более подробно этот аспект будет обсуждаться в главе 8.

Следом за датой истечения срока действия кредитной карты мы обнаруживаем новые для нас элементы управления — переключатели. Они используются, когда требуется выбрать только один вариант из нескольких. Они напоминают кнопки на автомагнитолах, служащие для быстрого доступа к заданным радиостанциям. Браузер отображает эти элементы управления таким образом, что пользователь может включать их щелчком мыши на них (или с помощью клавиатуры). Переключатели этого типа всегда объединяются в группы. При этом включение одного переключателя автоматически выключает все остальные переключатели этой группы. Внешний вид переключателя зависит от используемого графического интерфейса. В пункте бланка «Размер штуковины» также используются два переключателя. Группы переключателей определяются по значению параметра `name` (имя) тега `<input>`. Специальных скобок из тегов вроде `<radiobutton> ... </radiobutton>` для определения групп переключателей не предусмотрено.

Параметр `value` указывает, какая кнопка была нажата пользователем. В зависимости от того, какую кредитную карту выберет пользователь для своих расчетов, переменной `cc` будет присвоено значение текстовой строки «mastercard» или «visacard».

Следом за двумя наборами переключателей в бланке используется элемент управления типа `checkbox` (флажок). Он похож на переключатель предыдущего типа, так как тоже может находиться в одном из двух состояний (установлен/сброшен), но не объединяется в группы и включается и выключается щелчком мыши на нем независимо от других элементов управления того же типа. Например, заказывая пиццу на веб-странице компании Electropizza, пользователь может выбрать *и* сардины, *и* лук, *и* ананас (если у него крепкий желудок), но ему не удастся одновременно выбрать маленькую, среднюю и большую пиццу. Выбор приправы к пицце будет представлен в виде трех отдельных элементов управления типа `checkbox`, тогда как выбор размера пиццы будет реализован с помощью набора переключателей.

Если список, из которого предстоит сделать выбор пользователю, очень длинный, то использование переключателей несколько неудобно. В этом случае можно использовать теги `<select>` и `</select>` для определения списка альтернатив. При этом семантика переключателей сохраняется, если только не используется параметр `multiple`, превращающий список альтернатив в набор независимо устанавливаемых и сбрасываемых флажков. Некоторые браузеры отображают пункты списка между тегами `<select>` и `</select>` как всплывающее меню.

Итак, мы рассмотрели два встроенных типа тега `<input>`: `radio` и `checkbox`. На самом деле, мы уже познакомились и с третьим типом этого тега, то есть с типом `text`. Мы не заметили этого, потому что этот тип является типом по умолчанию,

поэтому параметр type = text можно не указывать. Еще два возможных значения параметра type: password и textarea. Окно password аналогично окну text, с той разницей, что при вводе текста в окне password символы не отображаются на экране. Окно textarea отличается от окна text тем, что может содержать несколько строк текста.

Возвращаясь к примеру на рис. 7.14, мы, наконец, добираемся до последнего использованного в этом примере элемента управления — кнопки submit (подтверждение). Когда пользователь щелкает мышью на этой кнопке, заполненный им бланк отсылается обратно на компьютер, на котором размещена эта веб-страница. Как и все остальные рассмотренные типы, submit является зарезервированным словом, интерпретируемым браузером. Строка параметра value (значение) в данном случае содержит надпись на кнопке. В принципе, все элементы управления, образуемые с помощью тега <input>, могут иметь параметр value. В окнах ввода текста содержимое параметра value отображается в окне редактирования, и пользователь может редактировать или удалить его. Элементы управления checkbox и radio также могут быть инициализированы с помощью специального служебного слова checked («выбрано». Дело в том, что value просто отображает текст, но не отображает предпочитаемый выбор.).

Когда пользователь нажимает кнопку submit, браузер упаковывает всю собранную информацию в одну большую строку и отправляет ее на сервер для обработки. Поля с данными разделяются амперсандами (&), а вместо пробелов ставятся знаки +. В нашем примере такая строка, отсылаемая на сервер, может выглядеть так, как показано ниже (вы видите две строки, а не одну, только из-за недостаточной ширины бумажного листа):

```
customer=John+Doe&address=100+Main+St.&city=White+Plains&state=NY&country=USA&cardno=1
234567890&expires=6/98&cc=mastercard&product=cheap&express=on
```

Это сообщение отправляется на сервер в виде одной текстовой строки. Если флажок элемента управления checkbox сброшен, соответствующая ему переменная опускается. Сервер сам решает, что ему делать с полученной строкой. Мы обсудим это позднее.

XML и XSL

Язык HTML — с формами или без оных — никак не определяет структуру веб-страниц. Он смешивает содержимое страницы и описание средств ее форматирования. По мере роста популярности электронной коммерции и других приложений появлялась все более очевидная необходимость в структурировании веб-страниц и отделении содержимого от форматирования. Например, поисковая программа, обещающая найти в Мировой паутине книгу или компакт-диск по самой выгодной цене, должна проанализировать множество страниц, находя нужное наименование и цену. Если страница написана на обычном HTML, такой программе будет очень тяжело определить, где указано название товара, а где — его цена.

По этой причине консорциум WWW (W3C) предложил расширение HTML, позволяющее структурировать страницы для облегчения их автоматической обработки. Для целей было создано два языка. Первый, **XML** (eXtensible Markup

Language — Расширяемый язык разметки веб-страниц), описывает структурированное содержимое страниц, а второй, **XSL** (eXtensible Style Language — расширяемый язык стилей), описывает форматирование независимо от содержимого. И о том, и о другом можно говорить очень долго, поэтому нам приходится ограничиться лишь поверхностным описанием идей, лежащих в основе этих языков.

Рассмотрим документ XML, представленный в листинге 7.7. В нем определяется структура book_list, представляющая собой список книг. Под каждую книгу отведено три поля: название, автор и год издания. Эти структуры чрезвычайно просты. Разрешается иметь структуры с повторяющимися полями (например, несколько полей с именами авторов), необязательными полями (например, название прилагающегося компакт-диска), а также альтернативные поля (например, URL магазина, если книга еще есть в продаже, и URL аукциона, если весь тираж уже распродан).

В приведенном примере каждое поле является неделимой сущностью, однако разрешается разделять поля на подполя. Например, поле, содержащее имя автора, может быть — для улучшения возможностей поиска и форматирования — организовано следующим образом:

```
<author>
<first_name> Эндрю </first_name>
<last_name> Таненбаум </last_name>
</author>
```

Итак, любое поле может иметь подполя неограниченной вложенности.

Код, представленный в листинге 7.7, делает лишь одно: определяет список из трех книг. Ничего не говорится о том, как должна выглядеть веб-страница на экране. Информация о форматировании страницы берется из другого файла, book_list.xsl, содержащего определения XSL. Реально данный файл представляет собой таблицу стилей, в которой оговаривается вид страницы. (Существуют и альтернативы таблицам стилей, позволяющие, например, преобразовывать XML в HTML, однако обсуждение этой темы выходит за рамки этой книги.)

Листинг 7.7. Простой пример на XML

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="book_list.xsl"?>

<book_list>
<book>
<title> Компьютерные сети, 4 изд. </title>
<author> Эндрю С. Таненбаум </author>
<year> 2003 </year>
</book>

<book>
<title> Современные операционные системы, 2 изд. </title>
<author> Эндрю С. Таненбаум </author>
<year> 2001 </year>
</book>

<book>
```

```
<title> Архитектура компьютера. 4 изд. </title>
<author> Эндрю С. Таненбаум </author>
<year> 1999 </year>
</book>

</book_list>
```

Пример XSL-файла для форматирования страницы из листинга 7.7 приведен в листинге 7.8. За некоторыми необходимыми объявлениями, включающими, например, URL используемого стандарта XSL, следуют теги, первыми из которых являются <html> и <body>. С этого начинается любая обычная веб-страница. Затем следует определение таблицы, включающее заголовки трех столбцов. Обратите внимание на то, что в дополнение к тегам <th> поставлены закрывающие теги </th>. Раньше нам было все равно, есть они или нет. Однако спецификации XML и XSL куда строже, чем HTML. Оговаривается, что страницы с синтаксическими ошибками должны отвергаться браузерами в любом случае, даже если они в состоянии понять, что имел в виду разработчик страницы. Браузер, отображающий синтаксически некорректный код XML или XSL, будет сам по себе признан некорректным при первом же тестировании на совместимость со стандартами. Однако браузерам разрешается выявлять ошибочные места. Такие драконовские меры нужны для борьбы с несметным числом небрежно написанных страниц, которые появились в Сети за последние годы.

Листинг 7.8. Таблица стилей на XSL

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl=http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">

<html>
<body>

<table border="2">
<tr>
<th>Название</th>
<th>Автор</th>
<th>Год</th>
</tr>

<xsl:for-each select="book_list/book">
<tr>
<td><xsl:value-of select="title"/> </td>
<td><xsl:value-of select="author"/> </td>
<td><xsl:value-of select="year"/> </td>
</tr>
</xsl:for-each>
</table>

</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Выражение

```
<xsl:for-each select="book_list/book">
```

аналогично подобному выражению на языке C. С его помощью запускается цикл (ограниченный тегами <xsl:for-each>). На каждую книгу приходится одна итерация этого цикла. И каждая итерация выдает пять строк: <tr>, название, автор, год и тег </tr>. По окончании цикла выводятся закрывающие теги </body> и </html>. Результат интерпретации браузером этой таблицы стилей такой же, как если бы это была обычная страница, содержащая таблицу. Однако благодаря такому формату анализирующая программа сможет по XML-файлу легко найти, например, книги, изданные после 2000 года. Надо отметить, что, хотя наш XSL-файл сдержит нечто вроде цикла, веб-страницы на XML и XSL все равно остаются статическими, поскольку они содержат лишь инструкции, указывающие браузеру, как отображать страницу. Тем же, в принципе, занимается и HTML. Разумеется, чтобы интерпретировать XML и XSL, браузер должен поддерживать эти языки. На сегодняшний день, впрочем, большинство браузеров имеют такую возможность. До сих пор не очень понятно, заменит ли XSL традиционные таблицы стилей.

Мы не показали этого в нашем примере, но XML позволяет разработчику веб-страницы определять структуры заранее в специальном файле. Такие файлы определений затем можно подключать для построения сложных страниц. Дополнительную информацию, касающуюся этого и многих других свойств XML и XSL, можно найти в любой из многочисленных книг, посвященных этой теме. Например, в (Livingston, 2000; Williamson, 2001).

Перед тем как закончить наш краткий рассказ об XML и XSL, будет нелишним прокомментировать идеологическую борьбу между консорциумом WWW и сообществом веб-дизайнеров. Изначальной целью HTML было определение именно *структуры* документа, а вовсе не его *внешнего вида*. Например, строка

```
<h1>Фотографии Натальи</h1>
```

сообщает браузеру о том, что следует выделить заголовок, однако ничего не говорит о его гарнитуре, размере или цвете. Все эти детали реализует браузер по своему усмотрению: у него есть преимущество, состоящее в том, что он знает свойства конкретного монитора (например, сколько на нем точек). Однако дизайнерам веб-страниц в какой-то момент захотелось получить тотальный контроль над видом создаваемых страниц. Тогда были добавлены новые теги, уточняющие, как должен выглядеть документ. Например,

```
<font face="Helvetica" size="24" color="red"> Фотографии Натальи </font>
```

Были добавлены также методы точного позиционирования элементов на экране. Проблема, присущая такому подходу, заключается в том, что такие страницы не обладают свойством переносимости. Они могут замечательно смотреться на браузере у создателя, однако на другом браузере, другой версии того же браузера или просто на экране с другим разрешением могут выглядеть совершеннейшей кашей. Одна из задач XML состояла в попытке вернуться к истокам, когда определялась только структура, а не внешний вид документа. Вместе с тем, XSL

позволяет управлять тем, как выглядят страницы. Оба языка, впрочем, порой используются не по назначению. Следует иметь это в виду.

XML можно использовать не только для описания веб-страниц. Все чаще он используется в качестве языка для связи между прикладными программами. В частности, **SOAP** (Simple Object Access Protocol — простой протокол доступа к объектам) предоставляет возможность выполнения удаленных вызовов процедур между приложениями способом, независимым от языка и системы. Клиент формирует запрос в виде сообщения XML и отсылает его на сервер по описываемому далее протоколу HTML. Сервер отсылает назад ответ, представляющий собой форматированное XML-сообщение. Таким образом могут общаться приложения, работающие на разнородных платформах.

XHTML — расширенный язык разметки гипертекста

К языку HTML постоянно предъявляются новые требования. Многие представители этой индустрии чувствуют, что в будущем большинство устройств, связанных со Всемирной паутиной, будут представлять собой не ПК, а беспроводные портативные устройства типа PDA. У таких мини-компьютеров нет столь большого объема памяти, чтобы работать с большими браузерами, обладающими сложной эвристикой, с помощью которой они пытаются отображать синтаксически некорректные страницы. Таким образом, следующей версией после HTML 4 должен стать язык, отличающийся крайне высокой требовательностью. Он называется не HTML 5, а **XHTML**, поскольку, по сути дела, представляет собой HTML 4, приведенный к стандарту XML. Под этим мы подразумеваем, что теги типа <h1> не имеют существенного значения. Чтобы добиться от такого тега того эффекта, который он производит в HTML 4, необходимо написать определение на XSL в отдельном файле. XHTML — это новый веб-стандарт, который рекомендуется использовать при создании любых веб-страниц для достижения максимальной переносимости на другие платформы и независимости отображения от браузера.

Между XHTML и HTML 4 существует шесть существенных и множество мелких различий. Во-первых, страницы и браузеры стандарта XHTML должны работать в строгом соответствии со стандартом. Низкопробные страницы уже отжили свой век. Это свойство унаследовано из XML.

Во-вторых, все теги и атрибуты должны быть написаны строчными буквами. Так, тег <HTML> будет считаться некорректным в XHTML. Необходимо писать <html>. Аналогично, некорректной записью считается такая: <img SRC="pic001.jpg"> Она содержит имя атрибута, написанное заглавными буквами, а это запрещено.

В-третьих, всегда должны присутствовать закрывающие теги, даже для </p>. Если у тега не может быть естественного закрывающего тега (например, <br>, <hr>, <img>), то перед закрывающей скобкой тега следует ставить косую черту. Например

```
<img src="pic001.jpg" />
```

В-четвертых, все значения атрибутов должны указываться в двойных кавычках. Вот пример неправильного использования тега:

```
<img src="pic001.jpg" height=500 />
```

Число 500 должно быть заключено в двойные кавычки, как и имя JPEG-файла.

В-пятых, свойство вложенности тегов должно использоваться корректно. В прошлом это не требовалось, важно было только получить ожидаемый результат. Раньше вполне легальным было написать:

```
<center> <b> Летние фотографии </center> </b>
```

В XHTML это запрещено. Закрывающие теги должны быть написаны строго в обратном порядке по отношению к открывающим тегам.

В-шестых, в каждом документе должен быть указан его тип. Мы имели возможность в этом убедиться на примере листинга 7.8. Все серьезные и мелкие изменения, которые происходят в стандартах, обсуждаются на сайте www.w3.org.

Динамические веб-документы

Все идеи, рассматривавшиеся до сих пор, соответствуют модели, показанной в листинге 6.1: клиент сообщает серверу имя файла, а тот в ответ возвращает файл. В первые годы существования Всемирной паутины все ее содержимое и в самом деле было статическим (просто файлы). Однако в последние годы в Сети появляется все больше динамических объектов, то есть таких, которые создаются по требованию, а не хранятся постоянно на диске. Автоматическое создание объектов может происходить как на стороне сервера, так и на стороне клиента. Рассмотрим оба случая по порядку.

Динамическая генерация содержимого веб-страниц на стороне сервера

Чтобы понять, зачем вообще нужна динамическая генерация веб-страниц на стороне сервера, рассмотрим использование форм, описанных ранее. Когда пользователь заполняет поля формы и нажимает кнопку *Submit* (Подтверждение), серверу отправляется сообщение, содержащее в себе данные, предоставленные пользователем. Это сообщение не содержит имя запрашиваемого файла. Требуется, чтобы оно было передано для обработки программе или скрипту. Обычно обработка подразумевает использование пользовательских данных для поиска по базе данных на серверном диске и создание HTML-страницы, содержимое которой зависит от результатов этого поиска. Затем страница отсылается клиенту. Например, в приложении для электронной коммерции нажатие кнопки *ПЕРЕЙТИ К РАСЧЕТАМ* приводит к тому, что браузер возвращает на сервер cookie-файл с содержимым корзины клиента. На сервере при этом должны запуститься определенные программы или скрипты, в задачу которых входят обработка cookie и генерация HTML-страницы. Отправляемая клиенту HTML-страница может содержать форму со списком товаров, положенных в корзину, и адрес доставки вместе с запросом подтверждения заказа и предложением выбрать одну из возможных форм оплаты. Этапы обработки информации, полученной из HTML-формы, показаны на рис. 7.15.

Традиционный способ работы с формами и другими видами интерактивных веб-страниц связан с использованием системы **CGI** (Common Gateway Interface —

общий шлюзовой интерфейс). Это стандартизованный интерфейс, позволяющий веб-серверам общаться с прикладными программами и скриптами, разрешающими вводить данные (например, в формы) и в ответ генерировать HTML-страницы. Обычно такие прикладные программы представляют собой скрипты, написанные на языке описания сценариев Perl, поскольку писать их проще и быстрее, чем программы (по крайней мере, если вы умеете программировать на Perl). Существует договоренность, в соответствии с которой эти скрипты должны размещаться в каталоге CGI-BIN, который доступен при помощи URL. Иногда вместо Perl используется другой язык написания скриптов, Python.

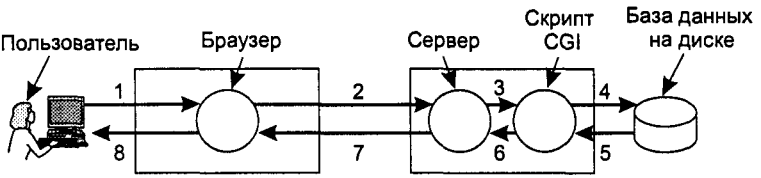


Рис. 7.15. Этапы обработки информации, полученной из формы

В качестве примера работы CGI рассмотрим случай, когда продукция компании «Великие Штуковины» приходит без гарантийного талона. Вместо этого клиенту предлагается зарегистрироваться в Интернете по адресу www.grwd.com. Там имеется ссылка:

Щелкните здесь для регистрации приобретения

Ссылка эта может указывать, например, на сценарий на языке Perl, расположенный по адресу www.grwd.com/cgi-bin/reg.perl. При запуске этого сценария без параметров обратно отсылается HTML-страница, содержащая регистрационную форму. Когда пользователь заполняет ее и нажимает кнопку *Submit*, скрипту передается сообщение, содержащее указанные им значения. Вид этого сообщения традиционен при работе с формами. Что происходит дальше, предугадать нетрудно. Perl-скрипт анализирует параметры, создает в базе данных запись о новом клиенте, отсылает назад HTML с регистрационным номером и телефоном службы поддержки. Понятно, что это не единственный способ обработки форм, однако он является наиболее распространенным. О создании CGI-скриптов и программировании на Perl написано много книг. Среди них стоит отметить (Hanegan, 2001; Lash, 2002; Meltzer и Michalski, 2001).

Динамическое создание веб-страниц на стороне сервера может быть реализовано не только с помощью CGI-скриптов. Существует еще один распространенный способ, который заключается во внедрении небольших скриптов в HTML-страницы. Они выполняются на сервере, в их задачу входит генерирование страницы. Популярным инструментом для написания таких скриптов является **PHP** (Hypertext Preprocessor — гипертекстовый препроцессор). При его использовании требуется, чтобы сервер понимал PHP (точно так же, как браузер должен понимать XML, чтобы интерпретировать страницы, написанные на одноименном языке). Обычно серверы предполагают, что у файлов страниц, написанных на PHP, расширение php, а не htm или html.

В листинге 7.9 приведен пример маленького скрипта на PHP; он должен работать на любом сервере, если на нем установлен гипертекстовый препроцессор. Он состоит из пр. вычного обрамления на HTML, а сам скрипт содержится внутри тега <?php ... ?>. Вся его работа заключается в создании страницы, сообщающей всю известную информацию о запустившем его браузере. Браузеры обычно отправляют кое-какие данные о себе вместе с запросами (и любыми прикладными cookie-файлами). Эти данные сохраняются в переменной HTTP_USER_AGENT. Если этот листинг сохранить в файле test.php в веб-каталоге компании ABCD, то пользователь, набрав URL www.abcd.com/test.php, сможет получить страницу, из которой он узнает, какие браузер, язык и операционную систему он использует.

Листинг 7.9. Пример страницы на HTML с внедренным PHP-скриптом

```
<html>
<body>

<h2>А я вот что про тебя знаю:</h2>
<?php echo $HTTP_USER_AGENT ?>

</body>
</html>
```

PHP особенно хорошо подходит для обработки форм — с его помощью она осуществляется даже проще, чем путем написания CGI-скрипта. Пример обработки формы показан в листинге 7.10, *а*. Здесь мы видим обычную HTML-страницу с формой. Непривычно выглядит только первая строка, в которой указывается, что скрипт action.php должен быть запущен для обработки параметров после заполнения пользователем формы и нажатия кнопки подтверждения. Форма в этом примере состоит из двух текстовых полей ввода, в одном из которых запрашивается имя клиента, а в другом — его возраст. По окончании работы пользователя с формой на сервер отсылается стандартная строка, пример которой мы уже видели ранее. Эта строка обрабатывается, из нее извлекаются значения переменных name и age. Затем начинает свою работу скрипт action.php, показанный в листинге 7.10, *б*. Он генерирует ответ. Работа скрипта заключается в исполнении php-команд. Если пользователь предоставил данные «Харриет» и «24», то ему будет прислан HTML-файл, код которого показан в листинге 7.10, *в*. Как видите, обработка форм с помощью PHP производится элементарно.

Несмотря на простоту использования, PHP — это мощный язык программирования, ориентированный на предоставление интерфейса между Всемирной паутиной и серверными базами данных. В PHP есть переменные, строки, массивы и большинство управляющих структур, присущих языку C, однако ввод-вывод гораздо мощнее, чем обычный printf. PHP имеет открытый исходный код и распространяется бесплатно.

Листинг 7.10. Веб-страница с формой (а); PHP-скрипт для обработки формы (б); результат работы PHP-скрипта при исходных данных «Харриет» и «24» соответственно (в)

```
<html>
<body>
<form action="action.php" method="post">
```

```
<p> Введите свое имя: <input type="text" name="name"> </p>
<p> Введите свой возраст: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

(а)

```
<html>
<body>
<h1> Ответ: </h1>
Привет, <?php echo $name; ?>!
Предсказываю: в следующем году тебе будет <?php echo $age+1; ?>
</body>
</html>
```

(б)

```
<html>
<body>
<h1> Ответ: </h1>
Привет, Харриет!
Предсказываю: в следующем году тебе будет 25
</html>
</body>
```

(в)

PHP был разработан специально под сервер Apache, который также обладает открытым исходным кодом и является самым распространенным веб-сервером в мире. Более подробную информацию по PHP можно найти в (Valade, 2002).

Итак, мы знаем уже два различных способа генерации динамических HTML-страниц: с помощью CGI-скриптов и внедрения PHP. Есть еще третий метод, называемый **JSP** (JavaServer Pages — страницы сервера Java). Он в целом схож с PHP и отличается только тем, что динамическая часть программируется на языке Java. Файлы страниц, написанных с помощью JSP, имеют одноименное расширение: .jsp. Еще один метод создания динамических страниц — **ASP** (Active Server Pages — активные серверные страницы). Это ответ Microsoft на PHP и JSP. В качестве языка динамического веб-программирования используется собственный язык написания скриптов, созданный Microsoft, — Visual Basic Script. Соответственно, файлы страниц, написанных с использованием этого метода, имеют расширение .asp. Вопрос выбора между PHP, JSP и ASP в основном политический (открытый код *Sun* против кода *Microsoft*). С точки зрения технологий все эти методы вполне сравнимы по возможностям.

Весь набор методов создания динамических страниц иногда называют **динамическим HTML** (DHTML).

Создание динамических веб-страниц на стороне клиента

Скрипты CGI, PHP, JSP и ASP решают вопросы обработки форм и взаимодействия с базами данных, расположенными на сервере. Они могут принимать входящую информацию из форм, осуществлять поиск по одной или нескольким базам

данных и в качестве результата генерировать HTML-страницы. Но ни один из этих методов не позволяет напрямую взаимодействовать с пользователем, например, реагировать на движения мышкой. Для этих целей необходимы скрипты, внедренные в HTML-страницы и выполняющиеся не на серверной, а на клиентской машине. Начиная с HTML 4.0, появилась возможность включать скрипты такого типа с помощью тега <script>. Наиболее популярный язык написания сценариев для клиентской стороны — это JavaScript. Его мы вкратце и рассмотрим далее.

Итак, JavaScript — это язык написания сценариев, использующий идеи, *крайне* отдаленно напоминающие язык программирования Java. Но JavaScript — это не Java по определению. Как и другие языки написания скриптов, он очень высокоуровневый. Так, одной строкой JavaScript можно создать диалоговое окно, войти в цикл ожидания пользовательского ввода и сохранить полученную строку в переменной. Столь высокий уровень языка идеально подходит для разработки интерактивных веб-страниц. С другой стороны, тот факт, что JavaScript не стандартизован и мутирует быстрее, чем мушка-дрозофила в рентгеновском луче, сильно усложняет написание платформонезависимых программ. Надо, впрочем, надеяться, что рано или поздно этот язык дойдет до более или менее устойчивого состояния.

Пример программы на JavaScript показан в листинге 7.11. Как и в листинге 7.10, *а*, программа создает форму с запросом имени и возраста пользователя и гениальным образом предсказывает, исходя из этих данных, каков будет возраст человека в следующем году. Тело скрипта почти такое же, как и в примере PHP. Основная разница состоит в объявлении кнопки *Submit* и определении присваивания в этом объявлении. Оператор присваивания сообщает браузеру о том, что в случае нажатия кнопки необходимо запустить скрипт response и передать ему форму в качестве параметра.

Совершенно по-новому здесь объявляется функция response. Объявление находится в заголовке HTML-файла, который обычно хранит информацию о заголовках, цвете фона и т. п. Функция извлекает из формы значение поля name и сохраняет его в виде строки в переменной person. Также извлекается значение поля age. Оно приводится к целочисленному типу с помощью функции eval, затем к значению добавляется 1, и результат сохраняется в years. После этого документ открывается для записи, в него записываются четыре строки (для этого используется метод writeln), и документ закрывается. Документ представляет собой HTML-страницу, как видно по многочисленным тегам HTML. Браузер выводит готовый документ на экран.

Листинг 7.11. Применение JavaScript для обработки формы

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
var person = test_form.name.value;
var years = eval(test_form.age.value) + 1;
document.open()
document.writeln("<html> <body>");
```

```
document.writeln("Привет, " + person + "!<br>");
document.writeln("Предсказываю: в следующем году тебе будет " + years +".");
document.writeln("</body> </html>");
document.close();
}
</script>
</head>

<body>
<form>
Введите свое имя: <input type="text" name="name">
<p>
Введите свой возраст: <input type="text" name="age">
<p>
<input type="button" value="Подтверждение" onclick="response(this.form)">
</form>
</body>
</html>
```

Важно понимать, что обрабатываются программы, показанные в листингах 7.10 и 7.11, совершенно по-разному, несмотря на их внешнее сходство. Что происходит в первом случае (листинг 7.10)? После того как пользователь нажимает кнопку *Submit*, браузер собирает всю введенную информацию в одну длинную строку и отправляет ее на тот сервер, с которого пришла страница. Сервер видит имя PHP-файла и запускает его. PHP-скрипт создает новую HTML-страницу, которая отсылается браузеру для отображения. Что касается второго случая (листинг 7.11), то после нажатия кнопки *Submit* браузер сам выполняет действия функции JavaScript, содержащейся на странице. Вся работа производится локально, внутри браузера. С сервером никакого взаимодействия не осуществляется. Как следствие, результат появляется практически мгновенно, тогда как при использовании PHP задержка прибытия страницы с результатом может составлять несколько секунд. Разница между скриптами, работающими на стороне сервере и на стороне клиента, показана на рис. 7.16. Показаны шаги, выполняемые в каждом случае. В обоих случаях все эти этапы производятся после вывода формы на экран. Шаг 1 состоит в приеме данных от пользователя. Затем следует их обработка, и вот здесь имеются значительные различия.

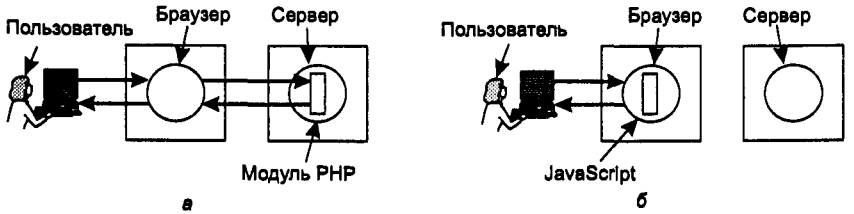


Рис. 7.16. PHP-окрипт на стороне сервера (*а*); оценарий на JavaScript на стороне клиента (*б*)

Эти различия вовсе не означают, что JavaScript «лучше», чем PHP. Просто у них различные сферы применения. PHP (а с ним неявно и JSP, и ASP) приме-

няется тогда, когда необходимо взаимодействовать с удаленной базой данных. JavaScript используют тогда, когда требуется взаимодействие с пользователем в пределах его компьютера. Разумеется, возможно (и довольно часто осуществляется) одновременное использование PHP и JavaScript, хотя они и не могут, например, обрабатывать одно и то же событие (типа нажатия кнопки) или производить одно и то же действие.

JavaScript — это полноценный язык программирования, ничуть не слабее по возможностям, чем C или Java. В нем есть понятия переменных, строк, массивов, объектов, функций и всех привычных управляющих структур. К тому же он обладает множеством полезных свойств, специфичных для веб-страниц, включая возможность управления окнами и фреймами, установки и получения cookie, работы с формами и гиперссылками. Пример программы на JavaScript с рекурсивной функцией приведен в листинге 7.12.

Листинг 7.12. Программа на JavaScript для подсчета и вывода факториалов

```
<html>
<head>
<script language="javascript" type="text/javascript">

function response(test_form) {
function factorial(n) {if (n == 0) return 1; else return n * factorial(n - 1);}
var r=eval(test_form.number.value); //r=введенный аргумент
document.myform.mytext.value = "Результаты:\n";
for (var i = 1;i<=r;i++)//Вывести одну строку от 1 до r
document.myform.mytext.value += (i + "!= "+factorial(i) + "\n");
}
</script>
</head>

<body>
<form name="myform">
Введите число: <input type="text" name="number">
<input type="button" value="Подсчет таблицы факториалов"
onclick="response(this.form)">
<p>
<textarea name="mytext" rows="25" cols="50"> </textarea>
</form>
</body>
</html>
```

С помощью JavaScript можно также отслеживать появление мыши над определенными объектами на странице. На многих веб-страницах с JavaScript можно заметить, что при наведении курсора мыши на какой-нибудь текст или изображение что-нибудь происходит. Часто при этом меняется изображение или вдруг выскакивает меню. Такое поведение легко программируется на JavaScript и несколько оживляет веб-страницы. Пример приведен в листинге 7.13.

Листинг 7.13. Интерактивная страница, отвечающая на движения мышью

```
<html>
<head>
<script language="javascript" type="text/javascript">
```

рам не приходится беспокоиться о потерянных, дублированных, слишком длинных сообщения и подтверждениях. Все это обеспечивается протоколом TCP.

В HTTP 1.0 после установки соединения посылался один запрос, на который приходил один ответ. После этого TCP-соединение разрывалось. В то время типичная веб-страница целиком состояла из HTML-текста, и такой способ взаимодействия был адекватным. Однако прошло несколько лет, и в странице оказалось множество значков, изображений и других украшений. Очевидно, что установка TCP-соединения для передачи одного значка нерациональна и слишком дорога.

Это соображение привело к созданию протокола HTTP 1.1, который поддерживал **устойчивые соединения**. Это означало, что появилась возможность установки TCP-соединения, отправки запроса, получения ответа, а затем передачи и приема дополнительных запросов и ответов. Таким образом, снизились накладные расходы, возникавшие при постоянных установках и разрывах соединения. Стало возможным также конвейеризировать запросы, то есть отправлять запрос 2 еще до прибытия ответа на запрос 1.

Методы

Несмотря на то что HTTP был разработан специально для использования в веб-технологиях, он был намеренно сделан более универсальным, чем это было необходимо, так как рассчитывался на будущее применение в объектно-ориентированных приложениях. По этой причине в дополнение к обычным запросам веб-страниц были разработаны специальные операции, называемые **методами**. Они обязаны своим существованием технологии SOAP. Каждый запрос состоит из одной или нескольких строк ASCII, причем первое слово является именем вызываемого метода. Встроенные методы перечислены в табл. 7.12. Помимо этих общих методов, у различных объектов могут быть также свои специфические методы. Имена методов чувствительны к регистру символов, то есть метод *GET* существует, а *get* — нет.

Таблица 7.12. Встроенные методы HTTP-запросов

Метод	Описание
GET	Запрос чтения веб-страницы
HEAD	Запрос чтения заголовка веб-страницы
PUT	Запрос сохранения веб-страницы
POST	Добавить к именованному ресурсу (например, к веб-странице)
DELETE	Удалить веб-страницу
TRACE	Отобразить входящий запрос
CONNECT	Зарезервирован для будущего использования
OPTIONS	Опрос определенных параметров

Метод *GET* запрашивает у сервера страницу (под которой в общем случае подразумевается объект, но на практике это обычно просто файл), закодирован-

ную согласно стандарту MIME. Большую часть запросов к серверу составляют именно запросы *GET*. Вот самая типичная форма *GET*:

```
GET filename HTTP/1.1,
```

где `filename` указывает на запрашиваемый ресурс (файл), а `1.1` — на используемую версию протокола.

Метод *HEAD* просто запрашивает заголовок сообщения, без самой страницы. С помощью этого метода можно узнать время последнего изменения страницы для сбора индексной информации или просто для проверки работоспособности данного URL.

Метод *PUT* является противоположностью метода *GET*: он не читает, а записывает страницу. Этот метод позволяет создать набор веб-страниц на удаленном сервере. Тело запроса содержит страницу. Она может быть кодирована с помощью MIME. В этом случае строки, следующие за командой *PUT*, могут включать различные заголовки, например, *Content-Type* или заголовки аутентификации, подтверждающие права абонента на запрашиваемую операцию.

Метод *POST* несколько напоминает метод *PUT*. Он также содержит URL, но вместо замены имеющихся данных новые данные «добавляются» (в неком общем смысле) к уже существующим. Это может быть публикация сообщения в конференции или добавление файла к электронной доске объявлений BBS. На практике ни *PUT*, ни *POST* широко не применяются.

Метод *DELETE*, что неудивительно, удаляет страницу. Как и в методе *PUT*, здесь особую роль могут играть аутентификация и разрешение на выполнение этой операции. Даже при наличии у пользователя разрешения на удаление страницы нет никакой гарантии, что метод *DELETE* удалит страницу, так как даже при согласии удаленного HTTP-сервера сам файл может оказаться защищенным от изменения или перемещения.

Метод TRACE предназначен для отладки. Он приказывает серверу отослать назад запрос. Этот метод особенно полезен, когда запросы обрабатываются некорректно и клиенту хочется узнать, что за запрос реально получает сервер.

Метод *CONNECT* в настоящее время не используется. Он зарезервирован для будущего применения.

Метод *OPTIONS* позволяет клиенту узнать у сервера о его свойствах или о свойствах какого-либо конкретного файла.

В ответ на каждый запрос от сервера поступает ответ, содержащий строку состояния, а также, возможно, дополнительную информацию (например, веб-страницу или ее часть). Строка состояния может содержать трехразрядный код состояния, сообщающий об успешном выполнении запроса или о причинах неудачи. Первый разряд предназначен для разделения всех ответов на пять основных групп, как показано в табл. 7.13. Коды, начинающиеся с 1 (1xx), на практике используются редко. Коды, начинающиеся с 2, означают, что запрос был обработан успешно и данные (если их запрашивали) отосланы. Коды 3xx сообщают клиенту о том, что нужно попытать счастья в другом месте — используя либо другой URL, либо свой собственный кэш.

Таблица 7.13. Группы кодов состояния, содержащиеся в ответах сервера

Код	Значение	Примеры
1xx	Информация	100 — сервер согласен обрабатывать запросы клиента
2xx	Успех	200 — запрос успешно обработан; 204 — содержимое отсутствует
3xx	Перенаправление	301 — страница перемещена; 304 — кэшированная страница все еще доступна
4xx	Ошибка клиента	403 — ошибка доступа; 404 — страница не найдена
5xx	Ошибка сервера	500 — внутренняя ошибка сервера; 503 — попробуйте еще раз позднее

Коды, начинающиеся с 4, означают, что запрос по какой-либо причине, связанной с клиентом, потерпел неудачу: например, была запрошена несуществующая страница или сам запрос был некорректен. Наконец, коды 5xx сообщают об ошибках сервера, возникших либо вследствие ошибки программы, либо из-за временной перегрузки.

Заголовки сообщений

За строкой запроса (например, содержащей название метода *GET*) могут следовать другие строки с дополнительной информацией. Они называются **заголовками запросов**. Эту информацию можно сравнить с параметрами, предоставляемыми при вызове процедуры. В свою очередь, ответы могут содержать **заголовки ответов**. Некоторые заголовки могут встречаться и там, и там. Наиболее важные из них перечислены в табл. 7.14.

Таблица 7.14. Некоторые заголовки сообщений протокола HTTP

Заголовок	Тип	Содержимое
User-Agent	Запрос	Информация о браузере и его платформе
Accept	Запрос	Тип страниц, поддерживаемых клиентом
Accept-Charset	Запрос	Поддерживаемые клиентом наборы символов
Accept-Encoding	Запрос	Поддерживаемые клиентом типы кодирования
Accept-Language	Запрос	Естественные языки, понимаемые клиентом
Host	Запрос	Имя DNS-сервера
Authorization	Запрос	Список персональных идентификаторов клиента
Cookie	Запрос	Отправка ранее принятого cookie-файла на сервер
Date	Запрос/Ответ	Дата и время отправки сообщения
Upgrade	Запрос/Ответ	Протокол, на который хочет переключиться отправитель
Server	Ответ	Информация о сервере
Content-Encoding	Ответ	Тип кодирования содержимого (например, gzip)
Content-Language	Ответ	Естественный язык, используемый на странице
Content-Length	Ответ	Размер страницы в байтах
Content-Type	Ответ	Тип MIME страницы

Заголовок	Тип	Содержимое
Last-Modified	Ответ	Время и дата внесения последних изменений в страницу
Location	Ответ	Команда клиенту на пересылку его запроса по другому адресу
Accept-Ranges	Ответ	Сервер готов принимать запросы на страницы указанного размера
Set-Cookie	Ответ	Сервер хочет, чтобы клиент сохранил cookie

Заголовок *User-Agent* позволяет клиенту информировать сервер о версии своего браузера, операционной системы или предоставлять другую информацию о себе. В листинге 7.9 мы видели, что сервер каким-то волшебным образом получал эти данные и мог при необходимости использовать их в PHP-скрипте. Как раз с помощью заголовка *User-Agent* клиент и сообщил серверу о себе.

Четыре заголовка, начинающиеся с *Accept*, сообщают серверу о типах информации, которые он готов принять (если их набор ограничен). Первый приведенный в таблице заголовок определяет типы MIME, которые будут корректно приняты клиентом (например, *text/html*). Заголовок *Accept-Charset* сообщает о том, какой набор символов клиент хотел бы видеть (например, ISO-8859 или Unicode-1-1). В заголовке *Accept-Encoding* речь идет о приемлемых методах сжатия (например, *gzip*). Наконец, *Accept-Language* сообщает, на каком языке клиент готов читать документы (например, на испанском). Если сервер имеет возможность выбирать из нескольких страниц, он подберет наиболее подходящий для клиента вариант в соответствии с полученной информацией. Если запрос удовлетворить невозможно, возвращается код ошибки, и запрос считается неудавшимся.

Заголовок *Host* описывает сервер. Его значение берется из URL. Этот заголовок обязателен. Почему? Потому что некоторые IP-адреса могут обслуживать несколько имен DNS одновременно, и серверу необходимо каким-то образом различать, кому передавать запрос.

Заголовок *Authorization* требуется в тех случаях, когда запрашивается защищенная страница. С его помощью клиент может подтвердить свои права на просмотр запрашиваемой страницы.

Несмотря на то, что cookie описываются в RFC 2109, а не в RFC 2616, для их описания существуют два заголовка. В частности, заголовок *Cookie* применяется клиентом при возвращении на сервер cookie-файла, который ранее был послан какой-либо машиной из домена сервера.

Заголовок *Date* может применяться как в запросах, так и в ответах. Он содержит время и дату отправки сообщения.

Заголовок *Upgrade* может использоваться для облегчения перехода на будущие (возможно, несовместимые с предыдущими) версии протокола HTTP. Он позволяет клиенту объявлять о поддерживаемых им протоколах, а серверу — объявлять о применяемых им протоколах.

А теперь мы подошли к заголовкам, которые может устанавливать только сервер при создании ответов на запросы. Первый из них, *Server*, позволяет серверу сообщать информацию о себе. При желании он может указать некоторые свои параметры.

Следующие четыре заголовка, начинающиеся с *Content-*, дают серверу возможность описать свойства посылаемой им страницы.

Заголовок *Last-modified* содержит дату и время внесения последних изменений в отправляемую страницу. Он играет важную роль при кэшировании страницы.

Заголовок *Location* вставляется сервером для информирования клиента о том, что стоит попробовать осуществить свой запрос повторно по другому URL. Такая ситуация может возникать при «переезде» страницы или тогда, когда несколько URL ссылаются на одну и ту же страницу (возможно, на «зеркало» страницы, расположенное на другом сервере). Этот трюк нередко применяется теми компаниями, главная веб-страница которых прописана в домене *.com*, однако клиенты перенаправляются с нее на национальные или региональные страницы, имеющие свои IP-адреса или написанные на более приемлемом для клиента языке.

Если страница очень велика по размеру, клиент может не захотеть принимать ее сразу целиком. Некоторые серверы могут принимать запросы, ограничивающие размеры страниц, отсылаемых за один раз. Если страница оказывается слишком большой, она будет разбита на более мелкие единицы и выслана в несколько приемов. Заголовок *Accept-Ranges* сообщает о том, что сервер готов поддерживать такие запросы частей страниц.

Set-cookie — это второй заголовок, относящийся к cookie-маркерам. Если этот заголовок установлен сервером, предполагается, что, увидев его, клиент сохранит у себя cookie и вернет его вместе со следующим запросом на сервер.

Пример использования HTTP

Поскольку HTTP является текстовым протоколом, взаимодействие с сервером посредством терминала (который в данном случае выступает как противоположность браузеру) можно организовать достаточно просто. Необходимо лишь установить TCP-соединение с портом 80 сервера. Читателю предоставляется возможность самому посмотреть, как работает этот сценарий (предпочтительнее запускать его в системе UNIX, поскольку некоторые другие системы могут не отображать статус соединения). Итак, последовательность команд такова:

```
telnet www.ietf.org 80 >log

GET /rfc.html HTTP/1.1
Host: www.ietf.org

close
```

Эта последовательность команд устанавливает telnet-соединение (то есть TCP-соединение) с портом 80 веб-сервера IETF, расположенного по адресу www.ietf.org. Результат сеанса связи записывается в файл log, который затем можно просмотреть. Далее следует команда *GET*. Указывается имя запрашиваемого файла и протокол передачи. Следом идет обязательная строка с заголовком *Host*. Пустая строка, которая находится за ней, также обязательна. Она сигнализирует серверу о том, что заголовки запросов закончились. Командой `close` (это команда программы telnet) соединение разрывается.

Файл журнала соединения, log, может быть просмотрен с помощью любого текстового редактора. Он должен начинаться примерно так, как показано в листинге 7.14, если только на сайте IETF за это время не произошли какие-нибудь изменения.

Листинг 7.14. Начало вывода файла www.ietf.org/rfc.html

```
Trying 4.17.168.6…
Connected to www.ietf.org
Escape character is '^]'.
HTTP/1.1 200 OK
Date: Wed, 08 May 2002 22:54:22 GMT
Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.5a
Last-Modified: Mon, 11 Sep 2000 13:56:29 GMT
ETag: "2a79d-c8b-39bce48d"
Accept-Ranges: bytes
Content-Length: 3211
Content-Type: text/html
X-Pad: предотвращает ошибки браузеров

<html>
<head>
<title>IETF RFC Page</title>

<script language="javascript">
function url() {
var x = document.form1.number.value
if (x.length == 1) {x = "000" + x}
if (x.length == 2) {x = "00" + x}
if (x.length == 3) {x = "0" + x}
document.form1.action = "/rfc/rfc" + x + ".txt"
document.form1.submit
}
</script>

</head>
```

Первые три строки в этом листинге созданы программой telnet, а не удаленным сайтом. А вот строка, начинающаяся с HTTP/1.1, — это уже ответ IETF, говорящий о том, что сервер желает общаться с вами при помощи протокола HTTP/1.1. Далее следует ряд заголовков и, наконец, само содержимое запрашиваемого файла. Мы уже видели все заголовки, кроме *ETag*, который является уникальным идентификатором страницы, связанным с кэшированием, и *X-Pad* — нестандартного заголовка, помогающего бороться с ошибками браузеров.

Повышение производительности

Популярность Всемирной паутины стала настоящей бедой для нее. Серверы, маршрутизаторы и каналы связи все чаще оказываются перегруженными. Многие уже стали называть WWW (World Wide Web) «Всемирным ожиданием» (World Wide Wait). Проблема нескончаемых задержек привела ученых к необхо-

димости разработки методов повышения производительности. Далее мы обсудим три из них: кэширование, репликацию серверов и сети с доставкой содержимого.

Кэширование

Довольно простым способом повышения производительности является сохранение ранее загружавшихся страниц на случай их повторного запроса. Этот метод особенно эффективен при работе с часто посещаемыми страницами (такими как www.yahoo.com или www.cnn.com). Сохранение веб-страниц «про запас» для последующего использования называется **кэшированием**. Обновление кэша является обычной процедурой для некоторого процесса, называемого сервером-посредником, или **прокси** . Чтобы иметь возможность использовать метод кэширования, браузер должен быть настроен на обращение к посреднику, а не к реальному серверу, на котором хранится страница. Если у сервера-посредника есть нужная страница, она сразу же возвращается пользователю. В противном случае ее придется получить с сервера, добавить в кэш для будущего использования и только после этого предоставить пользователю.

С кэшированием связаны два важных вопроса.

1. Кто должен заниматься кэшированием?
2. Сколько времени страницы должны храниться в кэше?

На первый вопрос есть несколько ответов. На отдельных персональных компьютерах часто имеется прокси, поэтому поиск ранее запрошенных страниц происходит быстро. В корпоративной ЛВС прокси-сервер обычно устанавливается на машине с разделяемыми ресурсами, и если один из клиентов данной ЛВС запросил страницу с сервера, то другой может получить ее уже из кэша сервера-посредника (прокси). Прокси-серверы часто устанавливают у себя провайдеры с целью повышения скорости доступа для всех своих клиентов. Нередко все эти кэши работают одновременно, поэтому запрос вначале отправляется на локальный прокси-сервер. Если там страница не обнаружена, запрос передается на прокси-сервер ЛВС. Не найдя у себя запрашиваемую страницу, последний обращается к прокси-серверу провайдера. На этом этапе страница уже должна быть получена в любом случае: либо она берется из кэша, либо приходит с веб-сервера. Схема с несколькими кэшами, работающими последовательно, называется **иерархическим кэшированием**. Возможная реализация этого метода показана на рис. 7.18.

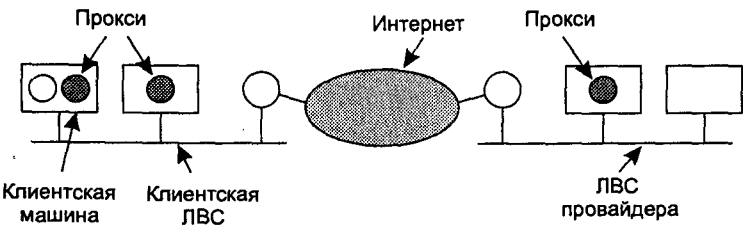


Рис. 7.18. Иерархическое кэширование с тремя серверами-посредниками

Вопрос о сроке хранения кэшированных страниц решается несколько хитрее. Некоторые страницы не кэшируются вообще. Это касается, например, страниц, содержащих списки 50 самых котируемых акций, цены на которые меняются каждую секунду. В случае кэширования пользователь получал бы *устаревшие* данные. С другой стороны, если на бирже в какой-то день торги не проводятся, эта страница может оставаться актуальной в течение нескольких часов или даже дней, до начала следующих торгов. Таким образом, необходимость в кэшировании каждой отдельно взятой страницы может сильно варьироваться с течением времени.

Ответ на вопрос, в какой момент удалять страницы из кэша, зависит от того, насколько свежими хочет видеть их пользователь (поскольку они сохраняются на диске, проблемы нехватки места обычно не возникают). Если сервер-посредник выбрасывает страницы из кэша слишком быстро, он вряд ли вернет устаревшую страницу, однако и эффективность такого кэша будет низкой. Если же хранить данные слишком долго, эффективность будет достигаться в основном за счет предоставления уже никому не нужной, устаревшей информации.

Решается этот философский вопрос с использованием двух подходов. Первый подразумевает эвристический анализ для принятия решения о сроке хранения страницы. Обычно он основывается на значении заголовка *Last-Modified* (см. табл. 7.13). Если страница подвергалась изменениям час назад, она будет храниться в кэше также в течение часа. Если же последние изменения были внесены в страницу год назад, очевидно, она содержит довольно стабильную информацию (например, список греческих и римских богов), и ее можно хранить в кэше в течение года, резонно предполагая, что за это время ничего на ней не изменится. Несмотря на то, что такой эвристический анализ работает весьма успешно, все же время от времени из кэша извлекаются устаревшие страницы.

Другой подход является более дорогим, но и более надежным в смысле исключения возможности хранения в кэше устаревших страниц. Используются особенности RFC 2616, имеющие отношение к управлению кэшем. Одной из самых полезных особенностей является наличие заголовка *If-Modified-Since*, который сервер-посредник может посылать веб-серверу. В нем указывается страница, состояние которой хочет выяснить прокси, а также время внесения последних изменений (значение заголовка *Last-Modified* на странице). Если страница не подвергалась изменениям, сервер отошлет обратно короткое сообщение *Not Modified* («Изменений нет», код 304, см. табл. 7.12). Это будет означать, что прокси может использовать хранящуюся в кэше страницу. Если же на странице произошли какие-либо изменения, сервер пришлет ее обновленную версию. Такой подход требует обмена запросами и ответами, но если страница еще не устарела, ответ сервера будет очень коротким.

Два указанных подхода можно комбинировать. В течение первого интервала времени ΔT после получения страницы прокси просто извлекает запрошенную страницу из кэша. По прошествии определенного промежутка времени прокси использует *If-Modified-Since* для проверки состояния страницы. Выбор ΔT подразумевает некоторый эвристический анализ, базирующийся на знании времени последних изменений на странице.

Динамические веб-страницы (например, созданные PHP-скриптом) не должны кэшироваться вообще никогда, так как их содержимое переменное по определению. Для этого, а также для некоторых других специальных случаев предусмотрен механизм, с помощью которого сервер может информировать все серверы-посредники на пути к клиенту о том, что не следует использовать текущую страницу без запроса ее актуальности. Этот же механизм может применяться для страниц, которые могут меняться довольно часто. В RFC 2616 определен также ряд других механизмов управления кэшированием.

Еще один подход к повышению производительности называется **упреждающим кэшированием**. Получая страницу с веб-сервера, прокси-сервер может проверить ее на наличие гиперссылок. Если таковые имеются, он может запросить и поместить в кэш страницы, на которые указывают гиперссылки, просто на тот случай, если они понадобятся пользователю. Этот метод может помочь уменьшить время доступа к последующим запросам, однако может привести к загрузке линий связи при передаче страниц, которые, возможно, так и не будут запрошены.

Понятно, что кэширование во Всемирной паутине реализуется далеко не тривиально. Эту тему можно было бы обсуждать еще долго. На самом деле, ей посвящены отдельные книги, например (Rabinovich и Spatscheck, 2002; Wessels, 2001). Однако нам пора двигаться дальше.

Репликация серверов

Кэширование — это технология повышения производительности, применяемая на стороне клиента. Существуют также методы, связанные со стороной сервера. Один из наиболее распространенных способов повышения производительности Сети заключается в репликации содержимого серверов на нескольких машинах, расположенных в разных точках земного шара. Такие серверы с одинаковым содержимым иногда называются **зеркалами**. Репликация часто применяется на корпоративных сайтах, где главная страница содержит несколько изображений и ссылок на региональные подразделения сайтов. Пользователь может выбрать сайт, находящийся на ближайшем сервере. Начиная с этого момента, все его взаимодействие с данным сайтом будет производиться через этот сервер.

Реплицированные сайты обычно отличаются статичностью. Компания решает, где расположить зеркала, решает вопрос с серверами в каждом из регионов и создает более или менее информативные страницы на каждом зеркале (возможно, убирая изображения падающих снежинок с сервера на Майами и фотографии пляжных лежаков с сервера в центре континента). Зеркала обычно сохраняются неизменными в течение многих месяцев или даже лет.

К сожалению, Всемирной паутине присущ феномен, известный под названием **внезапная давка**, когда тихий, неприметный, никому не известный сайт мгновенно оказывается в центре всеобщего внимания и интереса. Например, до 6 ноября 2000 года веб-сайт государственного секретаря штата Флорида, www.dos.state.fl.us, рассказывал немногочисленным посетителям о последних заседаниях и давал информацию о регистрации нотариальной деятельности во Флориде. Однако 7 ноября 2000 года, когда президентство в США вдруг стало целиком за-

висеть от нескольких тысяч спорных голосов в горстке округов Флориды, этот сайт внезапно стал одним из пяти наиболее посещаемых в мире. Надо ли говорить о том, что он просто не мог справиться с такой нагрузкой и практически задохнулся от наплыва посетителей.

Понадобился какой-то способ автоматической репликации веб-сайтов при угрозе массовой атаки посетителей. Причем количество зеркал в разных точках Земли должно удовлетворять спросу. Эти реплицированные сайты должны пережить волну высокого трафика, после чего большинство из них можно спокойно удалить с серверов. Для этого у сайта должна быть предварительная договоренность с какой-нибудь крупной хостинговой компанией о создании зеркал при необходимости и об оплате реально занимаемых в каждый момент времени ресурсов.

Более гибкая стратегия заключается в создании динамической постраничной репликации, зависящей от местоположения источника трафика. Некоторые исследования в этой области описаны в книгах (Pierre и др., 2001; Pierrre и др., 2002).

Сети доставки содержимого

Чем отличается капиталистический мир от всех иных формаций? Тем, что в нем всегда найдется человек, который извлечет выгоду даже из Всемирного ожидания. Как? Очень просто. Компании, называющие себя **CDN** (Content Delivery Networks — сети доставки содержимого), договариваются с поставщиками данных (например, музыкальными сайтами, газетами и другими компаниями, заинтересованными в быстрой доставке содержимого своих веб-сайтов) и за скромную плату предлагают им обеспечивать эффективную доставку их данных конечным пользователям. После подписания контракта компания передает CDN содержимое своего сайта для предварительной обработки (вкратце обсуждается далее) и последующего распространения.

После этого CDN договаривается с большим числом провайдеров, обещая хорошо заплатить за размещение в их ЛВС удаленно управляемой копии сайта с ценной для пользователя информацией. Для провайдера это не только способ получения дополнительного дохода, но и отличный ход в конкурентной борьбе: ведь клиенты этого провайдера смогут получать информацию CDN с очень хорошей скоростью. Провайдеры, отказывающиеся от предложений CDN, не только не получают «бесплатный сыр», но и проигрывают в качестве предоставляемых услуг по сравнению с провайдерами, сотрудничающими с сетями доставки содержимого. В таких условиях отказ от подписания контракта с CDN оказывается просто безумием. Крупнейшие сети доставки содержимого работают более чем с 10 000 серверами, разбросанными по всему миру.

Тысячекратное копирование содержимого, очевидно, существенно повышает производительность Сети. Тем не менее, вначале необходимо заставить систему работать, а для этого требуется способ переадресации запроса пользователя на ближайший сервер CDN. Желательно, чтобы его местоположение совпадало с провайдером. Эта переадресация должна происходить без изменения DNS или какой-либо другой части стандартной инфраструктуры Интернета. Далее в не-

сколько упрощенном виде описывается принцип работы Akamai — самой большой CDN.

Процесс начинается с передачи поставщиком информации своего веб-сайта в CDN. Каждая полученная страница проходит в сети доставки содержимого предварительную обработку. При этом существующие URL заменяются модифицированными. Идея, стоящая за этой стратегией, заключается в том, что обычно сайт состоит из некоторого числа маленьких страниц (обычного HTML-текста), на которых расположены ссылки на большие файлы (аудио- и видеозаписи, изображения). Модифицированные HTML-страницы остаются на своих прежних местах (на сервере поставщика информации), а на серверы CDN уходят только файлы больших размеров.

Рассмотрим работу этой схемы на примере веб-страницы фирмы «Пушистые фильмы» (листинг 7.15, *а*). После предварительной обработки она превращается в страницу, описанную в листинге 7.15, *б*, и размещается на сервере «Пушистых фильмов» по адресу www.furryvideo.com/index.html.

Когда пользователь набирает URL www.furryvideo.com, служба DNS возвращает IP-адрес веб-сайта фирмы, позволяя получить заглавную страницу сайта самым обычным образом — с сервера «Пушистых фильмов». Если же пользователь переходит куда-либо по гиперссылке с этой страницы, браузер через службу DNS обращается к серверу *cdn-server.com*. Затем на соответствующий IP-адрес браузером отправляется HTTP-запрос, в качестве ответа на который ожидается получение видеофайла в формате MPEG.

Однако этого не происходит по той простой причине, что сервер *cdn-server.com* не содержит никаких данных. А содержит их подставной HTTP-сервер сети доставки содержимого. По имени файла и названию сервера он определяет, какая страница запрашивается и кому из поставщиков информации она принадлежит. Кроме того, анализируется IP-адрес входящего запроса, и по базе данных определяется возможное местоположение пользователя. Вооружившись этой информацией, он устанавливает, какой из серверов CDN способен предоставить наилучшее качество обслуживания. Решение принять не так уж просто, поскольку близкое географическое расположение не обязательно означает близость в терминах сетевой топологии. Кроме того, даже оптимальный по расстоянию сервер в данный момент может быть слишком сильно загружен. Приняв решение, *cdn-server.com* отсылает ответ, возвращая код 301 и заголовок *Location*, содержащий URL файла, расположенного на одном из подведомственных серверов CDN. Предположим, что этот URL выглядит так: www.CDN-0420.com/furryvideo/bears.com. Браузер обрабатывает этот адрес стандартным образом, в результате чего пользователь получает MPEG-файл.

Листинг 7.15. Исходная веб-страница (а); та же страница после обработки CDN (б)

```
<html>
<head><title>Пушистые фильмы</title></head>
<body>
<h1> Список пушистых фильмов</h1>
<p>Бесплатные примеры:</p>
<a href="bears.mpg">Актуальные проблемы медведей</a><br>
```

```
<a href="bunnies.mpg">Забавные кролики</a><br>
<a href="mice.mpg">Мышки-норушки</a><br>
</body>
</html>
```

(а)

```
<html>
<head><title>Пушистые фильмы</title></head>
<body>
<h1> Список пушистых фильмов</h1>
<p>Бесплатные примеры:</p>
<a href="http://cdn-server.com/furryvideo/bears.mpg"> Актуальные проблемы
медведей</a><br>
<a href=" http://cdn-server.com/furryvideo/bunnies.mpg"> Забавные кролики</a><br>
<a href=" http://cdn-server.com/furryvideo/mice.mpg"> Мышки-норушки</a><br>
</body>
</html>
```

(б)

Все этапы описанного процесса показаны на рис. 7.19. На первом шаге определяется IP-адрес www.furryvideo.com. С сервера самым обычным образом загружается и выводится на экран HTML-страница. На ней размещены три гиперссылки на *cdn-server* (см. листинг 7.15, *б*). Пользователь выбирает, скажем, первую из них. Ищется ее DNS-адрес (шаг 5), который затем возвращается пользователю (шаг 6). При отправке запроса файла *bears.mpg* на *cdn-server* (шаг 7) клиенту возвращается просьба переадресовать свой запрос на сервер *CDN-0420.com* (шаг 8). Если он следует совету (шаг 9), ему выдается файл из кэша прокси-сервера (шаг 10). Вся эта система работает благодаря шагу 8, когда подставной HTTP-сервер перенаправляет запрос пользователя на сервер-посредник, расположенный максимально близко от клиента.

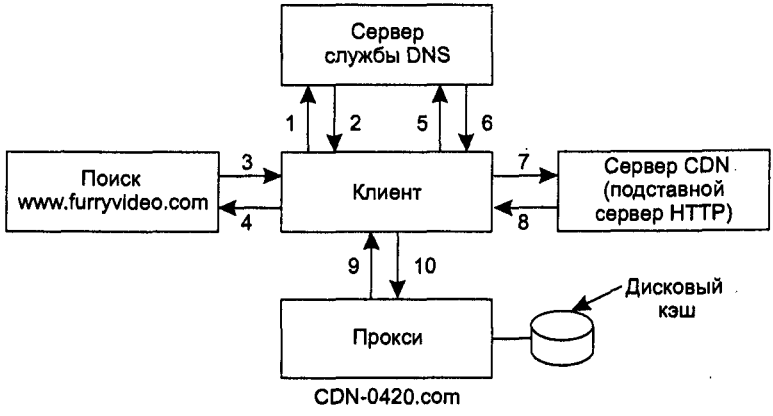


Рис. 7.19. Этапы поиска URL при использовании CDN

Сервер CDN, на который клиент обычно перенаправляется, чаще всего представляет собой прокси с кэшем большого размера, в который заранее загружают-

ся наиболее важные данные. Если, несмотря на эти меры, пользователь все же запрашивает файл, отсутствующий в кэше, он загружается с настоящего сервера поставщика информации и после этого сохраняется в кэше для последующего использования. Организация CDN-сервера в виде прокси, а не в виде точной копии исходного сервера позволяет сэкономить дисковое пространство, сократить время предварительной загрузки и улучшить другие показатели производительности.

Более полную информацию, касающуюся сетей доставки содержимого, можно найти в (Hill, 2002; Rabinovich и Spatcheck, 2002).

Беспроводная Паутина

В последнее время наблюдается все более широкий интерес к маленьким портативным устройствам, способным иметь доступ к Всемирной паутине с помощью беспроводных соединений. На самом деле, первые шаги в этом направлении уже сделаны. Несомненно, в ближайшие годы произойдет много изменений в этой области, однако все же стоит рассмотреть некоторые существующие ныне идеи, связанные с беспроводными технологиями в Web. Это поможет осознать, на какой стадии развития мы находимся сейчас и что может ждать нас впереди. Рассмотрим две технологии беспроводных глобальных веб-систем, успешно продвигающиеся на современном рынке: WAP и i-mode.

WAP — беспроводный протокол распространения данных через Интернет

Когда Интернет и мобильная связь стали общепринятыми явлениями, возникла идея объединения этих двух технологий в мобильном телефоне со встроенным экраном, с помощью которого пользователь смог бы получить доступ ко Всемирной паутине и электронной почте. Впервые эта идея была выдвинута консорциумом, основанным компаниями Nokia, Ericsson, Motorola и phone.com (бывшая Unwired Planet) и состоящим из сотен сотрудников. Система получила название **WAP** (Wireless Application Protocol — беспроводной прикладной протокол).

Устройством, поддерживающим WAP, может быть достаточно современный мобильный телефон, PDA или же ноутбук, не обладающий никакими голосовыми возможностями. Спецификация поддерживает все эти, а также другие устройства. Основная идея состоит в использовании уже существующей цифровой беспроводной инфраструктуры. Пользователи могут в буквальном смысле слова звонить на WAP-шлюз по беспроводному каналу и посылать ему запросы на веб-страницы. Шлюз сперва проверяет свой кэш на наличие в ней нужной страницы. Если она действительно там есть, страница, разумеется, отсылается абоненту; в противном случае он загружает ее себе через обычный Интернет. В сущности, это означает, что WAP 1.0 представляет собой систему с коммутацией каналов с очень высокой поминутной оплатой. Неудивительно, что большинство пользователей не пришли в восторг от возможности разглядывания веб-страниц на крошечном экране и поминутной оплаты. Поэтому технология WAP, мягко говоря, потерпела неудачу. Кроме указанных проблем, впрочем, были и другие.

Тем не менее, протокол WAP и его главный конкурент i-mode (описан далее) базируются на сходных технологиях, и не исключено, что WAP 2.0 еще ждет большой успех. Так как WAP 1.0 был первенцем среди технологий беспроводного Интернета, стоит рассказать о нем хотя бы вкратце.

WAP представляет собой стек протоколов доступа ко Всемирной паутине, оптимизированных под соединения с низкой пропускной способностью, использующие беспроводные устройства с медленными процессорами, небольшой памятью и маленьким экраном. Понятно, что эти требования значительно отличаются от случая настольных ПК. Эти отличия отразились на протоколах. Уровни стека протоколов показаны на рис. 7.20

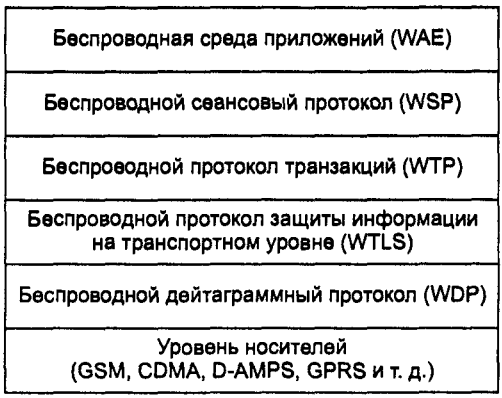


Рис. 7.20. Стек протоколов WAP

На нижнем уровне поддерживаются все существующие мобильные системы, включая GSM, D-AMPS и CDMA. Скорость передачи данных в WAP 1.0 составляет 9600 бит/с. Над уровнем носителей находится дейтаграммный протокол, **WDP** (Wireless Datagram Protocol — беспроводной дейтаграммный протокол). По сути дела, это UDP. Следом идет уровень обеспечения защиты информации, совершенно необходимый в сетях, где информация передается в эфире. Используемый здесь WTLS представляет собой часть Netscape SSL — системы, которую мы будем рассматривать в главе 8. Над ним располагает уровень транзакций, на котором происходит управление запросами и ответами. Возможны варианты разной степени надежности. Этот уровень заменяет TCP, который не используется в беспроводных соединениях по причине низкой эффективности. Затем идет сеансовый уровень, подобный HTTP/1.1, однако отличающийся от него некоторыми ограничениями и расширениями, соответствующими применению. Наконец, верхний уровень — это микробраузер (WAE).

Помимо высокой стоимости услуг есть еще один аспект, который, несомненно, повлиял на низкую популярность WAP: эта технология не использует HTML. Вместо этого на верхнем уровне (WAE) используется специальный язык разметки **WML** (Wireless Markup Language — язык разметки для беспроводного Интернета), являющийся одним из приложений XML. По этой причине, в принципе,

WAP-устройство может получать доступ только к страницам, преобразованным в WML. Понимание того, что это сильно ограничивает область применения, а следовательно, и значимость WAP, привело к разработке фильтров, конвертирующих HTML-страницы в WML-страницы «на лету». Вся эта система изображена на рис. 7.21.

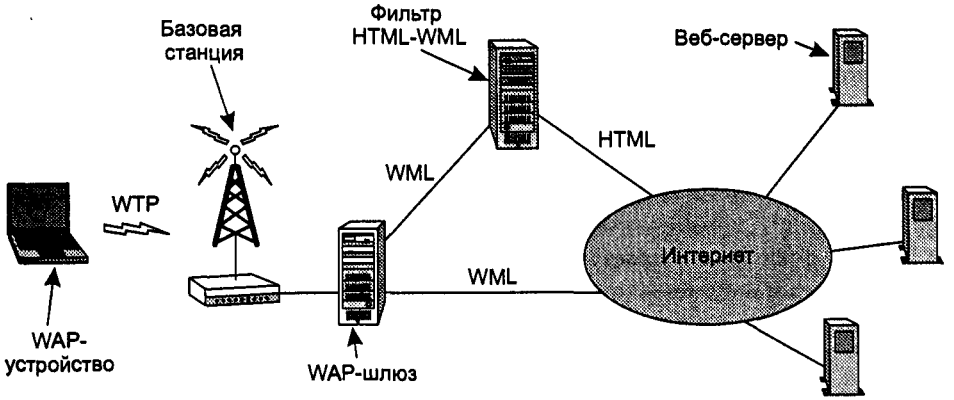


Рис. 7.21. Архитектура WAP

Надо отдать должное тому, что технология WAP, в общем-то, шла впереди своего времени. Когда она была впервые запущена, стандарт XML был почти неизвестен за пределами консорциума W3C, поэтому всезнающая пресса сделала далеко идущий вывод: ТЕХНОЛОГИЯ WAP НЕСОВМЕСТИМА С HTML. Более корректно было бы говорить о том, что ТЕХНОЛОГИЯ WAP УЖЕ ВНЕДРИЛА НОВЫЙ СТАНДАРТ HTML. Однако ущерб репутации был нанесен, и исправить положение оказалось так тяжело, что WAP 1.0 так и не стал популярным. Мы еще вернемся к этой технологии после рассмотрения ее главного соперника.

I-mode

В то время как многоотраслевой консорциум производителей телекоммуникационных систем и компьютерного оборудования выковывал новый открытый стандарт, использующий новейшую версию HTML, в Японии велись собственные разработки. Японка Мари Мацунага (Mari Matsunaga) изобрела альтернативный подход к беспроводным веб-технологиям и назвала его **i-mode** (information-mode — режим передачи информации). Она смогла убедить дочерние компании бывшей японской телефонной монополии в том, что ее подход является прогрессивным, и в феврале 1999 года NTT DoCoMo (буквально: Вездесущая японская телефонно-телеграфная компания) запустила новую услугу в Японии. Через три года у нее было уже более 35 миллионов абонентов по всей Японии, которые получили доступ к 40 000 специальных веб-сайтов i-mode. Телекоммуникационные компании по всему миру были удивлены огромным финансовым успехам, особенно в свете

того факта, что путь WAP оказался практически тупиковым. Что же это за система, как она работает? Далее мы ответим на эти вопросы.

Технология i-mode включает в себя три основных компонента: новую систему передачи данных, новый тип телефонных аппаратов и новый язык для создания веб-страниц. Система передачи состоит из двух отдельных сетей: существующей мобильной сети с коммутацией каналов (нечто вроде D-AMPS) и новой сети с коммутацией пакетов, специально разработанной для i-mode. В голосовом режиме используется традиционная сеть с коммутацией пакетов и поминутной оплатой. Что касается i-mode, то в этом режиме используется сеть с коммутацией пакетов, которая постоянно находится во включенном состоянии (как ADSL или кабельный Интернет), поэтому понятие повременной оплаты отсутствует. Взимается оплата, пропорциональная количеству посланных пакетов. В настоящее время использовать две сети одновременно невозможно.

Аппараты выглядят как мобильные телефоны с маленькими экранами. NNT DoCoMo постоянно рекламирует устройства i-mode, утверждая, что они являются более хорошими мобильными телефонами, чем беспроводные веб-терминалы, хотя на самом деле они как раз ими и являются. При этом пользователей, даже не знающих о том, что они имеют постоянное подключение к Интернету, большинство. По их мнению, принадлежащие им устройства i-mode представляют собой просто мобильные телефоны с расширенным набором функций. С такой моделью услуги i-mode вполне согласуется тот факт, что пользователь не имеет возможности программировать телефонный аппарат, хотя в них встроен некий аналог ПК образца 1995 года с Windows 95 или UNIX.

При включении телефона i-mode выводится список категорий официально предоставляемых услуг. Насчитывается более тысячи услуг, разбитых примерно на 20 категорий. Каждая услуга, представляющая собой маленький специализированный веб-сайт, поддерживается одной из независимых компаний. В основной пакет услуг входят такие категории, как электронная почта, новости, погода, спорт, игры, покупки, карты, гороскопы, развлечения, путешествия, путеводители, мелодии звонков, рецепты, азартные игры, доступ к банковским счетам, ставки акций. Такого рода услуги рассчитаны прежде всего на подростков и двадцатилетних молодых людей, которым обычно нравятся электронные безделушки, особенно если они имеют модную раскраску. Одно то, что более 40 компаний занимаются продажей такой «серьезной» продукции, как мелодии звонков, говорит о многом. Самой популярной услугой остается, разумеется, электронная почта, с помощью которой можно отправлять сообщения размером до 500 байт. Это, конечно, большой прогресс по сравнению с SMS (Short Message Service — Служба коротких сообщений), где размер сообщений не должен превышать 160 байт. Игры также популярны.

Существует более 40 000 веб-сайтов i-mode, однако для доступа к ним нужно набирать их URL, а не выбирать пункт меню. Можно сказать, что официальный список услуг — это некий интернет-портал, позволяющий выбрать один из сайтов с помощью пункта меню, без необходимости ввода URL.

NNT DoCoMo жестко контролирует официальные услуги. Чтобы попасть в список, услуга должна удовлетворять ряду известных критериев. Например,

«услуга не должна негативно влиять на общество», «в японско-английских словарях должно быть достаточно много слов», «услуги, предоставляющие мелодии для звонков, должны обновлять набор мелодий как можно чаще», а также «ни одна услуга не должна вести себя вызывающе или каким-либо мешать деятельности NNT DoCoMo» (Frengle, 2002). Остальные 40 000 сайтов могут делать что хотят, — это их личное дело.

Модель i-mode-бизнеса столь разительно отличается от привычного Интернета, что, наверное, стоит пояснить ее отдельно. Базовая ежемесячная абонентская плата составляет несколько долларов в месяц. Поскольку существует также плата за трафик, в базовый пакет услуг включено лишь небольшое число пакетов, которые можно передать бесплатно. Впрочем, пользователь может выбрать тариф с более высоким уровнем бесплатного трафика. Оплата в расчете на один пакет может резко снижаться при предоплате 10 Мбайт в месяц вместо 1 Мбайт. Дополнительный трафик можно оплачивать прямо через Интернет.

Для того чтобы пользоваться данной услугой, нужно стать ее абонентом. Это не слишком интеллектуальное действие: надо лишь щелкнуть на пункте меню и ввести свой PIN-код. Обычно ежемесячная плата за пользование большинством официальных услуг составляет $1–2 в месяц. NNT DoCoMo добавляет нужную сумму к счету за пользование телефоном. 91 % этой суммы уходит поставщику услуги, а 9 % телефонная компания оставляет себе. Если услуга, не являющаяся официальной, имеет миллион клиентов, ей необходимо каждый месяц рассылать миллион счетов, примерно по $1 каждый. Если же услуга входит в разряд официальных, все операции по счетам осуществляет NNT DoCoMo, производя, в частности, ежемесячное перечисление $910 000 на банковский счет услуги. Избавление от этих забот является хорошим стимулом для того, чтобы поставщики услуг становились официальными. Это выгодно и телефонной компании, поскольку сотрудничество с поставщиками официальных услуг ведет к повышению прибыли. Кроме того, доступ к официальной услуге осуществляется через меню, что на руку ее поставщику, поскольку пользователю найти такой сайт гораздо проще. В счет, предъявляемый абоненту, входит стоимость телефонных переговоров, плата за i-mode, услуги сайтов i-mode, а также за дополнительный трафик.

Несмотря на невероятный успех технологии i-mode в Японии, перспективы ее распространения в Европе и США до сих пор остаются неясными. Условия и предпосылки, существующие в это стране, во многом отличаются от западных.

Во-первых, большинство потенциальных клиентов на Западе (например, подростки, студенты колледжей, бизнесмены) имеют дома персональные компьютеры с большими мониторами и обеспечены доступом в Интернет со скоростью по крайней мере 56 Кбит/с, а чаще с гораздо более высокой. В Японии лишь немногие имеют дома персональные компьютеры с доступом в Интернет, частично из-за недостатка места для них, частично из-за заоблачных цен телефонных компаний на услуги местной связи (что-нибудь вроде $700 за установку линии и $1,50 за час местных разговоров). Для большинства японцев i-mode — это единственный способ доступа в Интернет.

Во-вторых, на Западе люди как-то не привыкли платить $1 в месяц за посещение сайта CNN, $1 за посещение Yahoo, еще $1 за Google, и т. д. И это еще не учитывая нескольких долларов за каждый загруженный мегабайт данных. Большинство провайдеров в западных странах взимают со своих клиентов фиксированную ежемесячную плату, не зависящую от реальных объемов использования линии. Надо отметить, что это является достойным ответом на требования пользователя.

В-третьих, большинство японцев пользуются i-mode по дороге в школу или на работу в метро или поезде. В Европе гораздо меньше людей, перемещающихся ежедневно на поезде, а в США таких почти нет. А использовать i-mode дома, имея под рукой компьютер с 17-дюймовым монитором, ADSL со скоростью 1 Мбит/с и огромным количеством свободных мегабайт на диске, как-то странно. Однако кто мог предвидеть, что мобильная связь станет такой популярной? Не исключено, что i-mode еще займет свою нишу на западном рынке телекоммуникаций.

Как мы уже говорили, аппараты i-mode используют уже имеющуюся сеть с коммутацией каналов для передачи речи и новую сеть с коммутацией пакетов для передачи данных. Последняя базируется на CDMA и передает 128-байтные пакеты со скоростью 9600 бит/с. Схема такой сети представлена на рис. 7.22. Телефон общается по беспроводному соединению со шлюзом преобразования протоколов при помощи протокола **LTP** (Lightweight Transport Protocol — упрощенный транспортный протокол). Этот шлюз, в свою очередь, общается с сервером i-mode по волоконно-оптическому соединению с высокой пропускной способностью. Наконец, сервер i-mode соединен со всеми доступными своему пользователю службами. Когда абонент выбирает один из элементов официального списка, отправляется запрос на сервер i-mode, в кэше которого хранится большинство страниц. Это помогает повысить производительность. Запросы страниц, отсутствующих в официальном списке, обходят сервер i-mode и попадают прямиком в Интернет.

В современный телефон обычно встраиваются процессор с частотой порядка 100 МГц, несколько мегабайт постоянной флэш-памяти, около 1 Мбайт оперативной памяти и небольшой экран. Для работы i-mode требуется экран размером, по крайней мере, 72×94 пиксела, однако некоторые высококачественные телефоны оснащены экранами размером 120×160 пикселов. При этом глубина цвета обычно составляет 8 бит (256 цветов). Этого, конечно, мало для отображения фотографий, однако вполне достаточно для векторных рисунков и несложной анимации. Поскольку мышь отсутствует, навигация по экрану осуществляется с помощью стрелок на клавиатуре.

Программная структура i-mode показана на рис. 7.23. На нижнем уровне расположена простая операционная система, управляющая оборудованием. Далее мы видим модуль реализации сетевой коммуникации, в котором применяется протокол LTP (собственный протокол NTT DoCoMo). Над ним расположен простой оконный менеджер, который поддерживает как текстовый режим, так и простой графический (GIF-файлы). Когда размер окна составляет в лучшем случае 120×160 пикселов, этот менеджер не особо перетруждается.

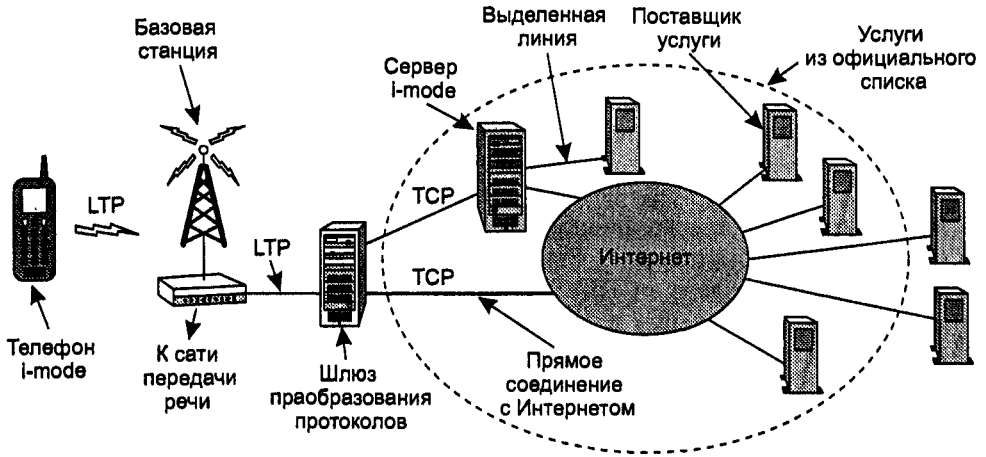


Рис. 7.22. Структура сети передачи данных i-mode и транспортные протоколы

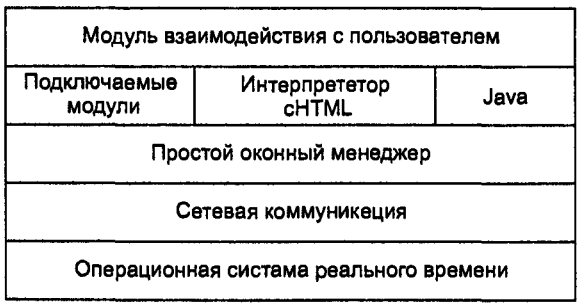


Рис. 7.23. Программная структура i-mode

Четвертый уровень содержит интерпретатор веб-страниц (то есть браузер). В i-mode используется не полная версия HTML, а только часть этого языка под названием **cHTML** (compact HTML – компактный HTML), имеющий много общего с HTML 1.0. Этот уровень также содержит подключаемые модули и вспомогательные приложения наподобие браузеров обычных ПК. Одним из стандартных вспомогательных приложений является интепретатор для слегка модифицированной версии JVM.

На самом верхнем уровне расположен модуль взаимодействия с пользователем, который, как следует из его названия, реализует взаимодействие «человек–машина».

Рассмотрим язык cHTML чуть более подробно. Как уже говорилось, это почти что HTML 1.0 с некоторыми сокращениями и в то же время добавлениями, позволившими применять его в мобильных телефонах. Он был направлен консорциуму W3C для стандартизации, однако тот не проявил к нему особого интереса, поэтому cHTML пока что так и остается патентованным (в противоположность стандартизованному) продуктом.

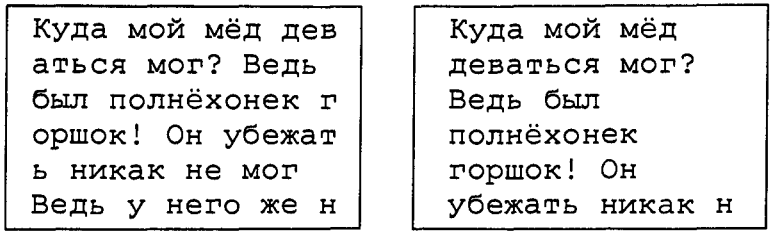
В cHTML разрешено использование большинства базовых тегов обычного HTML, включая <html>, <head>, <title>, <body>, <h*n*>, , <ul>, <ol>, <menu>, <li>, <br>, <p>, <hr>, <img>, <form> и <input>. Теги <b> и <i> запрещены.

Для связывания с другими страницами разрешено использование <a>, однако добавляется схема *tel*. В сущности, схема *tel* аналогична *mailto*. При выборе ссылки с *mailto* браузер открывает форму для редактирования и отправки электронной почты адресату, указанному в ссылке. Если же гиперссылка снабжена *tel*, браузер набирает указанный телефонный номер. Так, например, телефонная книжка может содержать простые изображения, связанные с каждым из абонентов. При выборе какого-либо из них браузер может набрать номер абонента. Телефонные URL обсуждаются в RFC 2806.

Браузер cHTML, разумеется, имеет ряд ограничений. Он не поддерживает JavaScript, фреймы, таблицы стилей, цвета фона и фоновые изображения. Также отсутствует поддержка изображений в формате JPEG, поскольку на их распаковку требуется слишком много времени. Java-апплеты разрешены, однако их размер (в настоящее время) ограничен 10 Кбайт, поскольку скорость беспроводных соединений весьма невысока.

Убрав некоторые теги HTML, NTT DoCoMo одновременно добавил некоторые свои. Так, тег <blink> заставляет текст мигать. Добавление этого элемента кажется шагом несколько непоследовательным, если учитывать, что <b> был убран лишь потому, что внешний вид страницы в данном случае большой роли не играет, а <blink> связан не с чем иным, как с внешним видом. Тем не менее, так было сделано. Тег <marquee> позволяет отображать текст на экране в виде бегущей строки (на манер бегущей строки с курсами ценных бумаг на бирже или рекламы на вокзале).

Еще одно новшество — атрибут *align* тега <br>. Это оказалось необходимо, поскольку на экране размером 6 строк по 16 символов велика вероятность того, что слово будет разорвано посередине (как на рис. 7.24, *а*). *Align* позволяет несколько снизить эту вероятность, форматируя текст так, как показано на рис. 7.24, *б*. Интересно, что текст на японском языке в принципе не может страдать от проблемы переноса на новую строку. При отображении иероглифов весь экран разбивается на ячейки размером 9×10 пикселов или 12×12 пикселов, в зависимости от шрифта. В каждой такой ячейке помещается ровно один иероглиф, который означает не меньше, чем слово на любом европейском языке. Разрывы строк между словами разрешены при отображении японского языка всегда.



а *б*

Рис. 7.24. Алан Милн сталкивается с проблемой экрана 16×6

Несмотря на то что в японском языке десятки тысяч иероглифов, компания NNT DoCoMo изобрела еще 166 собственных, называемых **emoji**. Они обладают высокой детализацией, а по сути аналогичны смайликам, представленным в табл. 7.2. Среди них есть символы астрологических знаков, пива, гамбургера, парка развлечений, символы дня рождения, мобильного телефона, собаки, кошки, Рождества, разбитого сердца, поцелуя, значки для различных настроений, спящая рожица и, конечно же, символ прекрасного чувства.

Еще один новый атрибут служит для того, чтобы пользователь мог выбрать пункт меню с помощью клавиатуры, что, несомненно, чрезвычайно важно для компьютера без мыши. Пример его использования в cHTML показан в листинге 7.16.

Листинг 7.16. Пример страницы на языке cHTML

```
<html>
<body>
<h1> Выберите:</h1>
<a href="messages.chtml" accesskey="1"> Проверить голосовую почту </a><br>
<a href="mail.chtml" accesskey="2"> Проверить E-mail </a><br>
<a href="games.chtml" accesskey="3"> Игра </a>
</body>
</html>
```

Хотя активность клиентской стороны несколько ограничена, сервер i-mode представляет собой полноценный компьютер, со всеми привычными штучками. Он поддерживает CGI, Perl, PHP, JSP, ASP и вообще все, что поддерживает любой нормальный веб-сервер.

Краткое сравнение первого поколения систем WAP и i-mode приведено в табл. 7.15. Некоторые аспекты сравнения кажутся не слишком значительными, однако зачастую они оказываются весьма важными. Например, у пятнадцатилетних подростков обычно нет кредитных карт, поэтому возможность включения оплаты покупок в электронных магазинах в телефонные счета радикально меняет их отношение к этой технологии. Более подробную информацию об i-mode можно узнать из книг (Frengle, 2002; Vacca, 2002).

Таблица 7.15. Сравнение первого поколения систем WAP и i-mode

Свойство	WAP	i-mode
Определение	Стек протоколов	Услуга
Устройства	Мобильный телефон, PDA, ноутбук	Мобильный телефон
Доступ	Наборный (dial-up)	Постоянное подключение
Базовая сетевая технология	Коммутация каналоа	Две: коммутация каналов и коммутация пакетов
Скорость передачи данных	9600 бит/с	9600 бит/с
Экран	Монохромный	Цветной
Язык разметки	WML (Приложение XML)	cHTML
Язык написания сценариев	WMLscript	Отсутствует
Оплата	Поминутная	За трафик

Свойство	WAP	i-mode
Оплата покупок	Кредитной картой	Включение в телефонный счет
Пиктограммы	Отсутствуют	Присутствуют
Стандартизация	Открытый стандарт форума WAP	Собственная разработка NNT DoCoMo
Зона использования	Европа, Япония	Япония
Типичный пользователь	Бизнесмен	Девушка

Второе поколение беспроводных веб-технологий

Технология WAP 1.0, базировавшаяся на известных международных стандартах, должна была стать мощным инструментом для деловых людей, которым приходится часто совершать поездки. Эта идея провалилась. Технология i-mode была создана в качестве игрушки для японских подростков и базировалась на стандартах, совершенно не известных за пределами Японии. И она оказалась чрезвычайно популярной. Что произошло после этого? Обе стороны извлекли уроки из первого опыта создания беспроводных веб-систем. Консорциум WAP убедился в том, что содержимое, как ни странно, имеет значение, и малое число сайтов, написанных на совместимом с системой языке разметки, губительно. Компания NNT DoCoMo осознала, что закрытая система, слишком тесно связанная с крошечными телефонами и японским образом жизни, не является оптимальной в качестве экспортного продукта. Разработчики обеих систем пришли к выводу о том, что необходим открытый стабильный универсальный язык разметки. Он позволил бы представлять множество сайтов в совместимом формате. Наконец-то производители поняли, что войны форматов не способствуют успеху в бизнесе.

Вот-вот должно появиться второе поколение обеих беспроводных веб-технологий. Версия WAP 2.0 появилась первой, поэтому мы рассмотрим ее в качестве примера. Кое-что было унаследовано из WAP 1.0, поскольку некоторые идеи были весьма хороши. Во-первых, WAP может работать на базе разных сетей. В первом поколении использовалась коммутация каналов, однако всегда существовала и остается возможность использования коммутации пакетов. Второе поколение в большей мере обращается к коммутации пакетов. Взять для примера хотя бы GPRS. Во-вторых, WAP всегда поддерживал и продолжает поддерживать широкий спектр устройств, от мобильных телефонов до мощных портативных компьютеров.

WAP 2.0 обладает некоторыми новыми чертами:

1. Пассивная модель приема сообщений наряду с активной моделью опроса сообщений.
2. Поддержка приложений с интегрированной телефонией.
3. Поддержка мультимедийных сообщений.
4. Наличие 264 пиктограмм.
5. Интерфейс с устройствами долговременного хранения информации.
6. Поддержка модулей, подключаемых к браузеру.

Активная модель опроса давно известна: клиент запрашивает страницу и получает ее. Что касается пассивной модели, то она подразумевает получение данных без запроса (например, клиент может получать непрерывный поток сообщений о ставках на ценные бумаги или состоянии трафика).

Технологии передачи данных и речи постепенно сливаются воедино, и WAP 2.0 реализует многие из них. Мы уже имели возможность убедиться в том, что это реально, на примере i-mode, где гиперссылка может представлять собой текст (или изображение), связанный с телефонным номером. Наряду с электронной почтой и телефонией имеется поддержка мультимедийных сообщений.

Невероятная популярность японских смайликов, emoji, не могла оставить равнодушным консорциум WAP, и в WAP 2.0 появились 264 новые пиктограммы. Они разбиты на категории, среди которых есть следующие: животные, приборы, одежда, эмоции, кулинария, человек, пол, карты, музыка, производство, спорт, время, инструменты, транспорт, оружие и погода. Получилось довольно забавно: стандарт описывает только названия пиктограмм (например, «сонная рожица», «крепкие объятия»); сами же картинки не приводятся. Наверное, причиной тому стала боязнь оскорбить представителей какой-нибудь культуры своим пониманием того, как выглядит «сонный человек» или «крепкие объятия». В i-mode такой проблемы не существовало, поскольку система была рассчитана только на одну страну и одну культуру.

Предоставление интерфейса с устройствами долговременного хранения данных не означает, что отныне все телефоны стандарта WAP 2.0 будут снабжаться большим жестким диском. Флэш-память тоже является устройством хранения информации. Например, беспроводная WAP-камера может использовать ее для временного сохранения изображений перед размещением лучших снимков в Интернете.

Наконец, теперь можно расширять возможности браузера за счет подключаемых модулей. Появилась поддержка языков написания скриптов.

Имеются также некоторые технические отличия WAP 2.0. Два наиболее серьезных из них связаны со стеком протоколов и языком разметки. WAP 2.0 продолжает поддерживать как старый стек протоколов WAP (см. рис. 7.20), так и стандартный стек интернет-протоколов TCP и HTTP/1.1. В целях некоторого упрощения кода в TCP были внесены четыре небольших изменения (не угрожающие совместимости): 1) используется фиксированное окно размером 64 Кбайи; 2) отсутствует затяжной запуск; 3) максимальное значение MTU равно 1500 байт; 4) слегка изменен алгоритм пересылки данных. TLS является стандартизованным IETF протоколом транспортного уровня с защитой информации; мы будем рассматривать его в главе 8. Многие устройства, судя по всему, будут поддерживать оба стека протоколов, показанных на рис. 7.25.

Еще одно техническое отличие WAP 2.0 от WAP 1.0 связано с языком разметки. WAP 2.0 поддерживает базовый язык XHTML Basic, предназначенный для малых беспроводных устройств. Так как японская телефонная компания NNT DoCoMo согласилась использовать это же подмножество, веб-дизайнеры могут пользоваться этим форматом, не заботясь о том, чтобы их страницы рабо-

тали как в фиксированном Интернете, так и на всех беспроводных устройствах. Эти решения должны положить конец войнам форматов, которые были настоящей угрозой развитию беспроводных веб-технологий.



Рис. 7.25. WAP 2.0 поддерживает два стека протоколов

Теперь, наверное, уместно сказать несколько слов об HTML Basic. Он предназначен для мобильных телефонов, телевизоров, PDA, торговых автоматов, пейджеров, автомобильных компьютеров, электронных игр и даже часов. В связи с этим в нем отсутствует поддержка каких бы то ни было таблиц стилей, скриптов, фреймов. Однако большинство стандартных тегов реализовано. Они сгруппированы в 11 модулей. Некоторые являются обязательными, некоторые — нет. Все они определены в XML. Модули и некоторые примеры тегов перечислены в табл. 7.16. Мы не стали перечислять все теги: более полную информацию можно найти на сайте www.w3.org.

Таблица 7.16. Модули и теги XHTML Basic

Модуль	Необходимость	Функция	Примеры тегов
Структура	Да	Структуре документа	`body, head, html, title`
Текст	Да	Информация	`br, code, dfn, em, hn, kbd, p, strong`
Гипертекст	Да	Гиперссылки	`a`
Списки	Да	Титульный список	`dl, dt, dd, ol, ul, li`
Формы	Нет	Заполнение форм	`form, input, label, option, textarea`
Таблицы	Нет	Прямоугольные таблицы	`caption, table, td, th, tr`
Изображения	Нет	Размещение изображений	`img`
Объекты	Нет	Апплеты, керты и т. д.	`object, param`

продолжение

Таблица 7.16 (*продолжение*)

Модуль	Необходимость	Функция	Примеры тегов
Метаинформация	Нет	Дополнительная информация	meta
Ссылка	Нет	Аналогично <a>	link
База	Нет	Точка отсчета URL	base

Несмотря на принятое соглашение об использовании XHTML Basic, над WAP и i-mode нависает угроза под кодовым названием *802.11*. Предполагается, что второе поколение беспроводных веб-технологий будет работать на скорости 384 Кбит/с. Это куда лучше, чем 9600 бит/с, однако все же несколько медленнее, чем 11 или 54 Мбит/с (стандарт 802.11). Разумеется, сети 802.11 установлены далеко не везде, однако появляется все больше ресторанов, отелей, магазинов, компаний, аэропортов, автобусных станций, музеев, университетов, больниц и других организаций, устанавливающих у себя базовые станции для своих работников и посетителей. Уже очень скоро настанет момент, когда в городской местности появится новая мода: пройти пару кварталов и зайти в кафе, чтобы выпить чашечку кофе и проверить электронную почту. В различных заведениях и конторах рядом с логотипами принимаемых к оплате кредитных карт уже пора помещать логотип 802.11. Это, несомненно, привлечет клиентов. На картах города (которые, кстати, можно загрузить из Интернета) появятся зеленые и красные зоны: охваченные и не охваченные базовыми станциями 802.11 соответственно. И люди будут перемещаться из одной зоны покрытия в другую, как кочевники, бредущие по пустыне от оазиса к оазису.

Тем не менее, если в ресторанах базовые станции появятся довольно скоро, в сельской местности фермеры, скорее всего, не будут торопиться устанавливать 802.11. Поэтому зоны действия таких сетей будут неоднородными и ограниченными, в основном, центрами городов (из-за небольшого радиуса действия базовых станций — в лучшем случае несколько сотен метров). Это может привести к появлению двухрежимных беспроводных устройств, работающих с сетью 802.11, а при отсутствии оной — с WAP.

Мультимедиа

Беспроводные веб-технологии — это, конечно, замечательное изобретение последних лет, однако далеко не единственное. Для многих не что иное, как мультимедиа, служит чашей святого Грааля сетевых технологий. Слово «мультимедиа» возбуждает и физиков, и лириков, и разработчиков, и коммерсантов. Одни видят в мультимедиа бесконечный источник интересных технических проблем, связанных, например, с доставкой (интерактивного) видео по заказу, другие — не меньший источник прибыли. Поскольку для передачи мультимедийных данных требуется очень высокая пропускная способность, довольно трудно заставить систему работать даже поверх стационарных соединений. Что касается беспроводных технологий, то добиться даже VHS-качества пока что не удается, и на решение этой

проблемы потребуется, как минимум, несколько лет. Мы будем рассматривать далее методы передачи мультимедийных данных по проводным сетям.

Буквально мультимедиа означает использование двух или более средств аудиовизуальной информации. Если бы издатель этой книги захотел присоединиться к всеобщей моде пускания пыли в глаза, он мог бы разрекламировать ее как использующую мультимедийные технологии. В конце концов, в ней действительно используются два средства информации: текст и графика (рисунки). Тем не менее, когда употребляется это слово, обычно имеются в виду некие информационные сущности, *длящиеся во времени*, то есть проигрываемые в течение определенного интервала времени, обычно во взаимодействии с пользователем. На практике чаще всего это аудио и видео, то есть звук плюс движущееся изображение.

Несмотря на такое довольно понятное определение, многие, говоря «мультимедиа», имеют в виду только чисто звуковую информацию, например интернет-телефонию или интернет-радио. Строго говоря, они не правы. Более удачным термином в данном случае является словосочетание **потоковая информация** (streaming media), однако мы, поддавшись стадному инстинкту, все же будем именовать аудиоданные, передающиеся в реальном масштабе времени, «мультимедиа». В следующих разделах мы узнаем, как компьютер обрабатывает звук и видео и как он сжимает такого рода данные. Затем мы рассмотрим некоторые сетевые технологии, связанные с мультимедиа. Довольно понятно и в хорошем объеме (три тома) взаимодействие сетевых и мультимедийных технологий описано в (Steinmetz и Nahrstedt, 2002; Steinmetz и Nahrstedt, 2003a; Steinmetz и Nahrstedt, 2003b).

Основы цифровой обработки звука

Звуковая волна представляет собой одномерную акустическую волну (волну давления). Когда такая волна достигает уха, барабанная перепонка начинает вибрировать, вызывая вибрацию тонких костей внутреннего уха, в результате чего в мозг по нерву посылается пульсирующий сигнал. Эта пульсация воспринимается слушателем как звук. Подобным образом, когда акустическая волна воздействует на микрофон, им формируется электрический сигнал, представляющий собой амплитуду звука как функцию времени. Представление, хранение, обработка и передача подобных аудиосигналов — именно эти вопросы рассматриваются при изучении мультимедийных систем.

Человеческое ухо способно слышать сигналы в диапазоне частот от 20 до 20 000 Гц, хотя некоторые животные, например собаки, могут слышать и более высокие частоты. Громкость, воспринимаемая ухом, изменяется логарифмически по отношению к частоте, поэтому сила звука обычно измеряется в логарифмах отношения амплитуд. Единицей измерения служит **децибел (дБ)**:

$$1 \text{ дБ} = 20 \log_{10}(A/B).$$

Если принять нижний порог слышимости (давление около 0,0003 дин/см^2, что равно $3 \cdot 10^{-5}$ Па) для синусоидальной волны частотой 1 кГц за 0 дБ,

то громкость обычного разговора будет соответствовать 50 дБ, а болевой порог наступит при силе звука около 120 дБ, что соответствует отношению амплитуд, равному 1 миллиону.

Человеческое ухо удивительно чувствительно к изменениям звука, длящимся всего несколько миллисекунд. Глаз, напротив, не в состоянии заметить такие кратковременные изменения. Таким образом, флуктуация (джиттер) в несколько миллисекунд при передаче мультимедиа влияет в большей степени на качество звука, чем на качество изображения.

Звуковые волны можно преобразовывать в цифровую форму при помощи **аналого-цифрового преобразователя (АЦП)**. На вход АЦП подается электрическое напряжение, а на выходе формируется двоичное число. На рис. 7.26, *а* показан пример синусоидальной волны. Чтобы представить этот сигнал в цифровом виде, мы можем измерять значения сигнала (отсчеты) через равные интервалы времени ΔT, как показано на рис. 7.26, *б*. Если звуковая волна не является чисто синусоидальной, а представляет собой сумму нескольких синусоидальных волн и самая высокая частота ее составляющих равна f, тогда, согласно теореме Найквиста (см. главу 2), для последующего восстановления сигнала достаточно измерять значения сигнала с частотой дискретизации $2f$. Производить замеры сигнала с большей частотой нет смысла, так как более высокие частоты отсутствуют в сигнале.

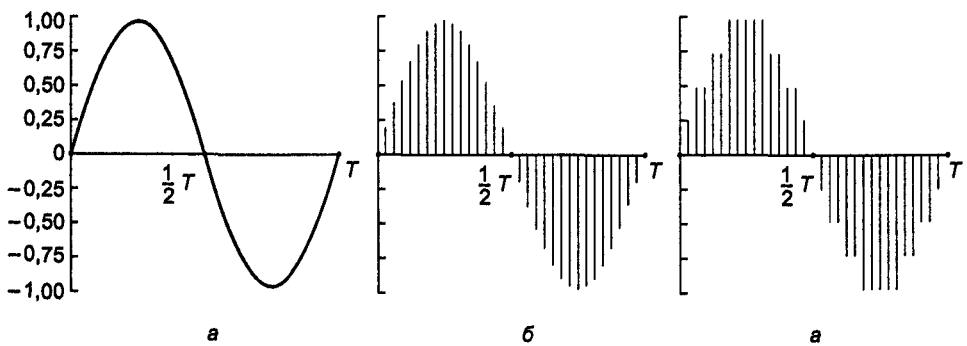


Рис. 7.26. Синусоидальная волна (*а*); дискретизация синусоидальной волны (*б*); квантование отсчетов 4 битами (*в*)

Оцифрованные отсчеты (сэмплы) никогда не бывают точными. Например, отсчеты на рис. 7.26, *в* могут принимать только 9 значений — от −1,00 до +1,00 с шагом 0,25. При 8-битовом квантовании каждый отсчет может принимать одно из 256 различных значений. При 16 битах на отсчет можно кодировать сигнал с еще более высокой точностью, так как каждому значению сигнала можно сопоставить одно из 65 536 различных значений. Ошибка, возникающая в результате неточного соответствия квантованного сигнала, способного принимать конечное число значений, исходному сигналу, называется **шумом квантования**. При недостаточном количестве битов, которыми представляется каждый отсчет сигнала, этот шум может быть настолько велик, что будет различим на слух как искажение исходного сигнала или как посторонние шумы.

Двумя хорошо известными примерами использования цифрового звука являются телефон (если применяются новые цифровые АТС) и аудио-компакт-диски. В кодово-импульсной модуляции, применяемой в телефонной системе, используются восьмибитовые отсчеты, замеряемые 8000 раз в секунду. В Северной Америке и Японии семью битами кодируются данные, а восьмой бит является служебным; в европейских же системах все 8 бит отводятся для данных. Таким образом, скорость передачи данных составляет 56 000 или 64 000 бит/с. При частоте дискретизации в 8 кГц частотные составляющие сигнала выше 4 кГц теряются.

Аудио-компакт-диски содержат звуковой сигнал, оцифрованный с частотой дискретизации 44 100 Гц, в результате чего они могут хранить звуки с частотами до 22 кГц, что воспринимается как достаточно качественный звук людьми, но считается весьма низким качеством среди собак, ценящих хорошую музыку. Каждому отсчету выделяется 16 бит, его значение пропорционально амплитуде сигнала. Обратите внимание на то, что 16-битовый отсчет может принимать всего 65 536 различных значений, хотя измерения показывают, что динамический диапазон человеческого уха составляет около 1 миллиона значений. Таким образом, использование 16 бит на отсчет дает некоторый шум квантования (хотя полный динамический диапазон и не охвачен, качество звучания компакт-дисков обычно не вызывает нареканий). При 44 100 отсчетах в секунду по 16 бит каждый аудио-компакт-диску требуется пропускная способность в 705,6 Кбит/с для монофонического сигнала и 1,411 Мбит/с — для стереофонического. Хотя это и меньше, чем требуется для передачи видеосигнала (см. далее), передача звука в таком (несжатом) формате в реальном времени займет канал T1 почти целиком.

Цифровой звук легко обрабатывается компьютерным программным обеспечением. Существуют десятки программ для персональных компьютеров, позволяющие пользователям записывать, воспроизводить, редактировать, микшировать и хранить звук. Сегодня вся профессиональная звукозапись и редактирование звука осуществляются в цифровом виде.

Музыка представляет собой лишь частный случай звука, хотя и очень важный. Помимо музыки, другим важным частным случаем мультимедиа является передача речи. Ее достаточно осуществлять в диапазоне частот от 600 до 6000 Гц. Речь состоит из гласных и согласных звуков, обладающих разными свойствами. Гласные звуки производятся при открытом голосовом тракте, при этом воздух, проходя через гортань, резонирует с частотой, определяемой размерами и формой голосовой системы и положением языка и челюсти говорящего. Гласные звуки являются почти периодичными с длительностью периода около 30 мс. Согласные звуки производятся при частично блокированном голосовом тракте. Эти звуки имеют не столь регулярную структуру, как гласные.

Некоторые системы воспроизведения и передачи речи используют модели голосовой системы для сведения речи к небольшому набору параметров (например, размерам и формам различных полостей) вместо того, чтобы просто дискретизировать звуковой сигнал речи. Однако рассмотрение устройства этих голосовых кодеров выходит за рамки данной книги.

Сжатие звука

Итак, как мы уже знаем, для передачи звука с качеством аудио-компакт-дисков требуется пропускная способность, равная 1,411 Мбит/с. Понятно, что для практической передачи подобных данных через Интернет требуется значительное сжатие. Для этого были разработаны различные алгоритмы сжатия оцифрованного звука. Одним из самых популярных форматов является аудио-MPEG, имеющий три уровня (разновидности). Самым известным и качественным является **MP3** (MPEG layer 3 — MPEG 3-го уровня). В Интернете можно найти огромное количество записей в MP3, не все из которых на самом деле являются легальными. Это привело к множеству судебных разбирательств, инициированных ущемленными в своих законных правах артистами и обладателями авторских прав. MP3 — это часть стандарта MPEG, предназначенного для сжатия видеосигнала. Методы сжатия движущихся изображений мы рассмотрим позднее в этой главе, а сейчас обратимся к сжатию звука.

Существуют две концепции сжатия звука. При **кодировании формы сигналов** сигнал раскладывается на компоненты при помощи преобразования Фурье. На рис. 2.1, *а* показан пример в виде временной функции и амплитуд, получающихся в результате ее разложения в ряд Фурье. Амплитуда каждого компонента кодируется с минимальными искажениями. Задачей является максимально аккуратная передача формы сигнала с минимально возможной затратой битов.

Другая концепция называется **перцепционным кодированием**. Она основана на некоторых недостатках слухового аппарата человека, позволяющих шифровать сигнал таким образом, что слушатель не ощутит никакой разницы по сравнению с настоящим сигналом, хотя на осциллографе эта разница будет весьма заметна. Наука, на которой базируется перцепционное кодирование, называется **психоакустикой**. Она изучает восприятие звука человеком. Формат MP3 использует перцепционное кодирование.

Ключевым свойством перцепционного кодирования является то, что одни звуки могут **маскировать** другие. Представьте себе, что теплым летним вечером вы медитируете на лужайке, слушая живой концерт для флейты с оркестром. Затем, откуда ни возьмись, появляется бригада рабочих с отбойными молотками в руках, которая начинает вскрывать асфальт на близлежащей улице. Расслышать флейту, к сожалению, уже никто не в состоянии. Нежные звуки, издаваемые ею, подверглись *маскированию* звуками отбойных молотков. Если рассматривать ситуацию с точки зрения передачи данных, то в этот момент достаточно кодировать лишь диапазон частот, в котором работают отбойные молотки, — все равно флейту за этим грохотом не слышно. Способность громких звуков определенного диапазона частот «прятать» более тихие звуки других диапазонов (которые были бы слышны при отсутствии громких звуков) называется **частотным маскированием**. На самом деле, даже после того как рабочие выключат отбойные молотки, слушатели не будут слышать флейту в течение некоторого небольшого периода времени. Это связано с тем, что при появлении очень громкого звука коэффициент усиления человеческого уха резко снизился, и после прекращения работы отбойных молотков требуется время для его возвращения в нормальное состояние. Этот эффект называется **временным маскированием**.

Чтобы перейти от качественного описания этих эффектов к количественным, представим себе проведение некого эксперимента 1. Человек, находящийся в тихом помещении, надевает наушники, соединенные со звуковой картой компьютера. Компьютер генерирует звук (чистую синусоидальную звуковую волну) с частотой 100 Гц, сила которого постепенно возрастает. Испытуемый должен нажать клавишу на клавиатуре, как только он услышит звук. Компьютер запоминает силу звука, при которой была нажата клавиша, и повторяет эксперимент на частотах 200 Гц, 300 Гц и т. д., доходя до верхнего предела слышимых частот. Эксперимент необходимо провести над большим количеством испытуемых. На рис. 7.27, *а* показан график с логарифмическим масштабом на обеих осях, показывающий усредненную зависимость порога слышимости от частоты звука. Наиболее очевидный вывод, который можно сделать при взгляде на эту кривую, состоит в том, что нет никакой необходимости когда бы то ни было кодировать частоты, амплитуда которых ниже порога слышимости. Например, если сила звука на частоте 100 Гц равна 20 дБ, этот звук можно не кодировать, и качество звучания при этом не ухудшится, так как уровень 20 дБ при 100 Гц находится ниже порога слышимости (рис. 7.27, *а*).

Теперь рассмотрим эксперимент 2. Пусть компьютер повторяет действия эксперимента 1, но на этот раз на каждую тестовую частоту будет накладываться синусоидальная звуковая волна постоянной амплитуды с частотой, скажем, 150 Гц. Мы обнаружим, что порог слышимости для частот, расположенных вблизи 150 Гц, резко возрастает. Это отражено на графике на рис. 7.27, *б*.

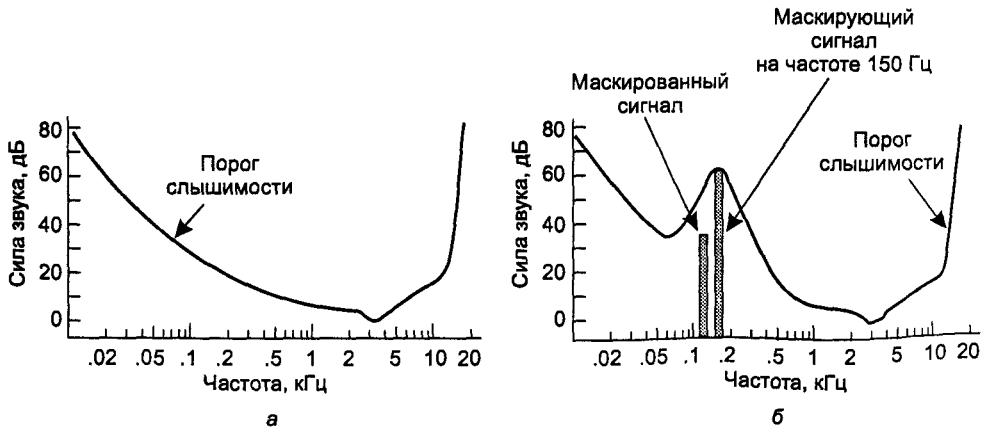


Рис. 7.27. Порог слышимости как функция частоты (*а*); эффект маскирования (*б*)

Из последнего наблюдения можно сделать следующий вывод: зная, какие сигналы маскируются более мощными сигналами на близлежащих частотах, мы можем пренебречь соответствующими частотами и не кодировать их, экономя тем самым биты. Из рис. 7.27, *б* очевидно, что сигналом с частотой 125 Гц можно полностью пренебречь, и никто не заметит разницы. Знание свойств временного маскирования позволяет даже после прекращения звучания громкого сигнала

в каком бы то ни было частотном диапазоне в течение некоторого времени (пока ухо настраивается на меньшую мощность звука) продолжать пренебрегать кодированием этой частоты. Суть алгоритма MP3 состоит в разложении сигнала в ряд Фурье для получения силы звука на каждой из частот с последующей передачей исключительно немаскированных частот, кодируемых минимально возможным числом бит.

Теперь, зная основной принцип, мы можем рассмотреть, как производится само кодирование. Сжатие звука выполняется путем замеров формы сигналов, производимых с частотой 32 000, 44 100 или 48 000 раз в секунду. Замеры могут сниматься по одному или двум каналам в одной из четырех комбинаций:

1. Монофонический звук (один входной поток).
2. Двойной монофонический звук (например, звуковая дорожка на английском и японском).
3. Разъединенное стерео (каждый канал сжимается отдельно).
4. Объединенное стерео (учитывается межканальная избыточность сигнала).

Для начала выбирается желаемая выходная битовая скорость. С помощью алгоритма MP3 можно сжать записанную на компакт-диск стереофоническую запись рок-н-ролла до 96 Кбит/с с потерей качества, едва заметной даже для фанатов рок-н-ролла, не лишенных слуха. Если мы хотим «перегнать в MP3» фортепианный концерт, нам понадобится битовая скорость по крайней мере 128 Кбит/с. Чем обусловлена такая разница? Дело в том, что соотношение сигнал/шум в рок-н-ролле гораздо выше, чем в фортепианном концерте (только в техническом смысле, разумеется). Можно, впрочем, выбрать меньшую битовую скорость и получить более низкое качество воспроизведения.

После этого отсчеты обрабатываются группами по 1152 (что занимает около 26 мс). Каждая группа предварительно проходит через 32 цифровых фильтра, выделяющих 32 частотных диапазона. Одновременно входной сигнал заводится в психоакустическую модель для определения маскирующих частот. Затем каждый из 32 частотных диапазонов преобразуется с целью получения более точного спектрального разрешения.

Следующим шагом является распределение имеющегося запаса бит между частотными диапазонами. При этом большее число бит отводится под диапазон с наибольшей немаскированной спектральной мощностью, меньшее — под немаскируемые диапазоны с меньшей спектральной мощностью, и совсем не отводятся биты под маскируемые диапазоны. Наконец, битовые последовательности шифруются с помощью кода Хаффмана (Huffman), который присваивает короткие коды числам, появляющимся наиболее часто, и длинные — появляющимся редко.

На самом деле, эта тема далеко не исчерпана. Существуют методы шумоподавления, сглаживания сигналов, использования межканальной избыточности (при наличии такой возможности), однако все это, к сожалению, невозможно охватить в рамках нашей книги. Более формально изложенные математические основы этих процессов даются в книге (Pan, 1995).

Потоковое аудио

Перейдем от технологии оцифровки звука к трем сетевым приложениям, использующим ее. Первое, что мы рассмотрим, будет потоковое аудио, то есть прослушивание звукозаписей через Интернет. Это иногда называется «музыкой по заказу». Затем мы познакомимся с интернет-радио и технологией передачи речи поверх IP.

В Интернете есть множество музыкальных веб-сайтов, на многих из которых представлены списки песен, которые пользователь может прослушать, щелкнув на названии. Подобные услуги на некоторых сайтах бесплатны (например, таким образом молодая группа может рекламировать свою деятельность и искать своего слушателя); иногда за прослушивание взимается плата, хотя при этом чаще всего есть возможность бесплатно ознакомиться с композицией (например, первые 15 секунд записи). Наиболее распространенный способ реализации такой системы «музыки по заказу» показан на рис. 7.28.

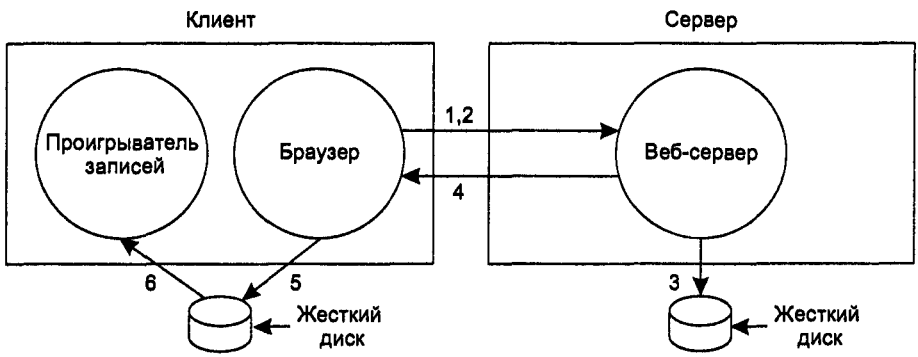


Рис. 7.28. Простейший способ раализации технологии «музыки по заказу» на ваб-сайте

Процесс начинается, когда пользователь щелкает на названии песни. Вначале вступает в дело браузер. Первый шаг заключается в установке TCP-соединения с веб-сервером, на который указывает гиперссылка. На втором шаге ему направляется HTTP-запрос *GET*, содержащий заявку на получение файла. Шаги 3 и 4 выполняются сервером: он получает песню (которая представляет собой обычный файл в формате MP3 или каком-нибудь другом) с диска и отсылает ее браузеру. Если размер файла превышает объем памяти сервера, он может извлекать и отправлять его поблочно.

По MIME-типу файла (например, *audio/mp3*) или по расширению браузер определяет способ его воспроизведения. Обычно запускается вспомогательное приложение, ассоциированное с данным типом файлов, например, RealOne Player, Windows Media Player или Winamp. Обычно браузер общается со вспомогательным приложением, записывая информацию, предназначенную для воспроизведения, во временный файл. Поэтому до начала проигрывания весь музыкальный файл будет записан во временный файл на диске (шаг 5). После этого будет запущен проигрыватель, которому браузер передаст имя временного файла. Шаг 6

заключается в поблочном считывании данных из файла и непосредственном воспроизведении музыки.

В принципе, такой подход совершенно корректен, и музыку пользователь услышит. Единственная серьезная проблема заключается в том, что вся запись должна быть предварительно передана по сети. Если музыкальный файл занимает 4 Мбайт (типичный размер аудиофайла с песней в формате MP3) и передается при помощи модема со скоростью 56 Кбит/с, пользователь будет наслаждаться тишиной в течение почти 10 минут, прежде чем услышит музыку. Далеко не все меломаны приходят от этого в восторг. К тому же не стоит забывать, что после окончания данной песни следующую можно будет услышать также только через 10 минут.

Как обойти эту проблему, не внося изменений в работу браузера? Музыкальные сайты пришли к следующей схеме. Файл, связанный гиперссылкой с названием песни, на самом деле является не музыкальным, а **метафайлом**. Метафайл обычно очень короткий, в типичной ситуации он состоит всего лишь из одной текстовой строки, которая выглядит примерно так:

```
rtsp://joes-audio-server/song-0025.mp3
```

Получив такой однострочный файл, браузер записывает его во временный файл, запускает в качестве вспомогательного приложения проигрыватель и передает ему, как обычно, имя временного файла. Проигрыватель видит, что временный файл содержит URL. Он соединяется с сервером *joes-audio-server* и запрашивает песню. Вы, должно быть, уже заметили, что браузер не принимает участия в этом процессе.

В большинстве случаев сервер, указанный в метафайле, не совпадает с веб-сервером, содержащим ссылку на метафайл. Более того, обычно это даже не HTTP-сервер, а специализированный мультимедийный сервер. В приведенном примере этот сервер использует протокол **RTSP**, это становится понятно при взгляде на ссылку — она начинается с названия схемы, *rtsp*. RTSP описан в RFC 2326.

Проигрыватель мультимедиа решает следующие 4 основные задачи:

1. Управление интерфейсом пользователя.
2. Обработка ошибок передачи.
3. Распаковка сжатых аудиоданных.
4. Устранение флуктуации (джиттера).

Большинство современных программ для воспроизведения мультимедиа имеют привлекательный интерфейс. Часто внешний вид панелей управления (**оболочек**) можно менять. Как мы уже сказали, в задачи проигрывателя входит общение с пользователем.

Вторая его задача — обработка ошибок. При передаче музыки в реальном времени редко используется протокол TCP, так как ошибки и повторные передачи могут приводить к недопустимо долгим паузам. Вместо этого передача осуществляется по протоколу типа RTP, который мы изучали в главе 6. Подобно большинству протоколов реального времени, RTP работает поверх UDP, то есть

пакеты могут теряться по пути. Проблему потерянных пакетов решает проигрыватель.

Иногда для упрощения обработки ошибок при передаче музыки применяется принцип чередования. Например, пакет может содержать 220 стереоотсчетов, каждый из которых содержит пару 16-разрядных чисел. Этого вполне достаточно для 5 мс звучания музыки. Однако протокол может посылать в одном пакете все нечетные отсчеты для 10 мс звучания, а в следующем — все четные отсчеты для того же интервала. Таким образом, потеря одного пакета не приведет к возникновению паузы длиной 5 мс, вместо этого будет потерян каждый второй отсчет 10-миллисекундного интервала. Эта утрата может быть восполнена, если проигрыватель произведет интерполяцию, используя предыдущий и следующий отсчеты (относительно каждого потерянного). Таким образом, примерные значения будут восстановлены.

Принцип чередования, используемый для восстановления ошибок, показан на рис. 7.29. Здесь видно, что каждый пакет содержит чередующиеся отсчеты для 10-миллисекундного интервала. В результате потери пакета 3 (см. рисунок) не возникает паузы, а только лишь на некоторое время снижается частота следования отсчетов. Утерянные значения путем интерполяции могут быть восстановлены; это обеспечит непрерывность звучания. Данная конкретная схема работает только с несжатыми отсчетами, однако она показывает, как при помощи хитрого алгоритма кодирования превращать потерянные пакеты не в раздражающие паузы, а в непрерывные интервалы с пониженным качеством. Между тем в RFC 3119 описан метод, позволяющий применять аналогичную процедуру к сжатым аудиоданным.

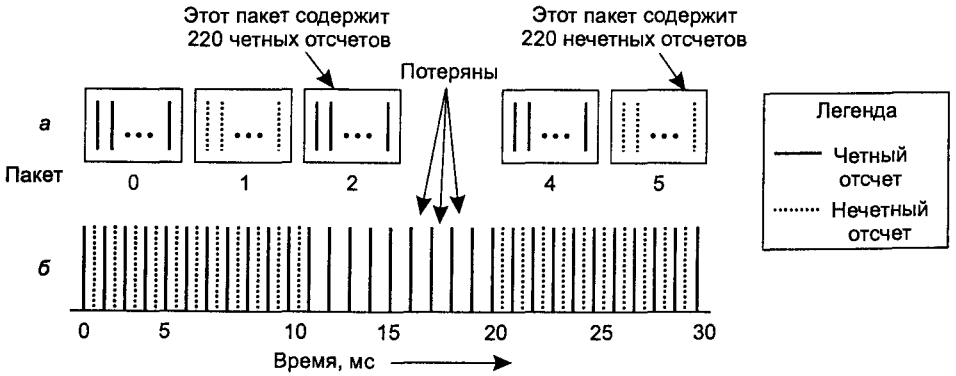


Рис. 7.29. Когда пакеты содержат чередующиеся отсчеты, потеря пакета уменьшает частоту следования отсчетов, но не приводит к возникновению паузы

Третья задача программы-проигрывателя заключается в декодировании сжатых аудиоданных. Несмотря на высокую требуемую интенсивность вычислений, применяемый метод довольно очевиден.

Четвертая задача — устранение флуктуации (джиттера), бича всех систем реального времени. Все системы воспроизведения потокового аудио перед началом

проигрывания буферизируют около 10–15 с звучания, как показано на рис. 7.30. В идеале — сервер должен продолжать заполнять буфер со скоростью, необходимой для проигрывателя, однако в реальности это может и не происходить, поэтому всегда полезно иметь постоянную обратную связь.

Существуют два способа постоянного поддержания буфера в заполненном состоянии. При использовании **выталкивающего сервера** (pull server) проигрыватель отправляет запросы на получение нового блока всегда, когда в буфере есть свободное место. Цель этого метода состоит в том, чтобы загружать в буфер как можно больше данных.

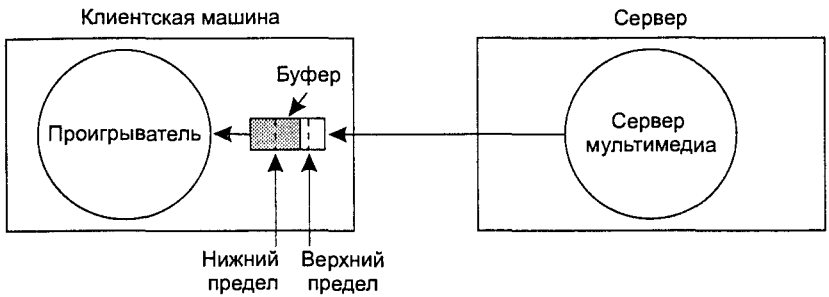


Рис. 7.30. Проигрыватель буферизирует входные данные, принимаемые с мультимедийного сервера, и воспроизводит содержимое буфера, а не данные, поступающие напрямую из сети

Недостатком выталкивающего сервера является наличие необязательных запросов данных. Серверу известно, что он отослал весь файл, так почему же проигрыватель продолжает что-то запрашивать? По этой причине данный метод используется редко.

При использовании **проталкивающего сервера** проигрыватель отправляет запрос *PLAY* (Воспроизведение), и сервер просто отправляет ему данные. При этом возможны два варианта: либо сервер мультимедиа работает со скоростью воспроизведения, либо он работает с более высокой скоростью. В обоих случаях некоторое количество данных буферизируется перед началом проигрывания. Если сервер работает со скоростью воспроизведения, то посылаемые им данные добавляются в конец буфера, а проигрыватель извлекает данные из начала буфера. Пока все работает без сбоев, объем данных, хранимых в буфере, остается постоянным во времени. Это очень простая схема — здесь не требуется пересылать управляющие сообщения в каких-либо направлениях.

Еще один вариант схемы проталкивания основан на том, что сервер выдает данные быстрее, чем реально требуется проигрывателю. Преимуществом является то, что при отсутствии гарантий стабильной скорости сервер имеет возможность наверстать упущенное из-за задержки время. Однако есть опасность переполнения буфера, возникающая вследствие того, что данные «потребляются» медленнее, чем выдаются (а это, в свою очередь, необходимо для устранения возможных пауз при воспроизведении).

Решением этой проблемы является установка проигрывателем **нижнего** и **верхнего пределов** заполнения буфера. Суть в том, что сервер выдает данные

лишь до тех пор, пока буфер не заполнится до верхнего предела. После этого проигрыватель просит сервер приостановить передачу. Поскольку данные будут продолжать прибывать в течение времени передачи запроса приостановки, расстояние между верхним пределом и концом буфера должно быть больше некоторого X — количества данных, которые успеют передаться за этот промежуток времени. X соответствует задержкам в канале, возникающим вследствие ограниченной пропускной способности. После приостановки сервера буфер начнет опустошаться. Когда количество данных в нем достигнет нижнего предела, проигрыватель попросит сервер мультимедиа возобновить передачу. Нижний предел должен быть установлен таким образом, чтобы буфер не опустошался полностью.

Для работы по методу проталкивающего сервера проигрыватель должен осуществлять удаленное управление сервером. Это обеспечивается протоколом RTSP, предоставляющим соответствующий механизм управления. Он определен в RFC 2326. Протокол не предназначен для управления потоком данных, для этого обычно применяется RTP. Основные команды RTSP приведены в табл. 7.17.

Таблица 7.17. Команды RTSP, посылаемые проигрывателем на сервер

Команда	Действие сарвера
DESCRIBE	Перечисляет параметры мультимедийных данных
SETUP	Устанавливает логическое соединение между проигрывателем и сервером
PLAY	Начинает отправлять данные клиенту
RECORD	Начинает прием данных от клиента
PAUSE	Приостанавливает передачу данных
TEARDOWN	Удаляет логическое соединение

Интернет-радио

Как только стало возможным передавать потоковое аудио через Интернет, коммерческие радиостанции задумались о том, как бы организовать вещание в Сети (параллельно с вещанием в эфире). Вскоре маленькие радиостанции в колледжах стали передавать сигнал через Интернет. *Студенты* колледжей организовали собственные радиостанции. Действительно, технологии сейчас находятся на том уровне, когда практически каждый может основать собственную радиостанцию. интернет-радио — это совсем молодая область, находящаяся в стадии развития, но о ней стоит сказать пару слов.

Есть два основных подхода к организации радиовещания в Интернете. Первый подразумевает, что передачи предварительно записываются и сохраняются на диске. Слушатели могут получить доступ к архивам радиостанции, выбрать интересующую передачу и загрузить ее себе для прослушивания. В общем-то, это ничем не отличается от потокового аудио, которое мы только что обсуждали. Также возможно сохранять передачи сразу же после их выхода в прямом эфире. В этом случае архив состоит из передач, которые звучали, скажем, полчаса назад или еще меньше. Преимуществом является то, что технически это очень просто

организовать, — используются все те же методы потокового аудио; при этом слушатели могут выбрать любую передачу из архива.

Совсем другой подход связан с *радиовещанием* через Интернет. Некоторые станции организуют параллельное вещание — в эфире и в Сети. Однако появляется все больше станций, работающих исключительно через Интернет. Некоторые технологии, применяемые для передачи потокового аудио, подходят и для живого вещания, однако есть некоторые серьезные различия.

Похожи эти технологии тем, что и там, и там требуется буферизация на стороне пользователя, позволяющая снизить флуктуацию (джиттер). Буферизация 10–15 с звучания до начала проигрывания позволяет сделать вещание непрерывным даже в условиях довольно заметной флуктуации (джиттера) в сети. До тех пор, пока пакеты прибывают раньше, чем они реально нужны, не имеет никакого значения, когда именно они прибывают.

Одно из ключевых *отличий* состоит в том, что потоковое аудио можно выдавать со скоростью, превышающей скорость воспроизведения, поскольку приемник может остановить процесс, когда буфер заполняется до верхнего предела. В принципе, за счет этого появляется время на передачу потерянных пакетов, хотя практически это свойство редко используется. Что касается живого радиовещания, здесь скорость выдачи информации всегда точно соответствует скорости ее создания и воспроизведения.

Еще одно отличие состоит в том, что аудитория радиостанции может исчисляться сотнями или тысячами слушателей, тогда как потоковое аудио рассчитано на двухточечный обмен информацией. В таких условиях, очевидно, интернет-радио может передавать широковещательный сигнал с помощью протоколов RTP/RTSP. Это наиболее эффективный способ работы.

Однако на сегодняшний день интернет-радио работает по-другому. Реально происходит вот что: пользователь устанавливает TCP-соединение с радиостанцией и принимает данные посредством протокола TCP. Конечно, это порождает ряд проблем, таких как остановка передачи при заполнении окна, потеря пакетов с последующей повторной передачей и т. д.

Почему же вместо широковещания по RTP применяется однонаправленная передача по TCP? Есть три причины этого. Во-первых, лишь немногие провайдеры поддерживают широковещание, этот метод передачи используется очень редко. Во-вторых, протокол RTP гораздо менее известен, нежели TCP, а многие радиостанции слишком малы, чтобы иметь в штате профессиональных компьютерщиков. Гораздо проще использовать понятный и популярный протокол TCP, который поддерживается большинством программных продуктов. В-третьих, многие любят слушать радио на работе, то есть за границей брандмауэра. Большинство сетевых администраторов настраивают брандмауэры таким образом, чтобы защитить локальную сеть от нежелательного проникновения в нее извне. Обычно разрешается установка TCP-соединений с удаленного порта 25 (SMTP для электронной почты), прием UDP-пакетов с удаленного порта 53 (DNS), а также установка TCP-соединений с портом 80 (HTTP для Всемирной паутины). Почти все прочие возможности, включая RTP, могут быть заблокированы. Таким образом, единственный способ передать радиосигнал через брандмауэр — это заставить

веб-сайт притвориться HTTP-сервером (по крайней мере, для брандмауэра) и, соответственно, использовать HTTP-серверы, которые общаются по TCP. Такие суровые меры, обеспечивая лишь минимальную защиту информации, зачастую резко снижают эффективность мультимедийных приложений.

Поскольку интернет-радио — это новая среда передачи данных, войны форматов идут полным ходом. RealAudio, Windows Media Audio и MP3 ведут достаточно агрессивную конкуренцию на этом рынке, борясь за право быть доминирующим форматом радиовещания в Интернете. Сейчас появился еще один формат — Vorbis, который технически похож на MP3, но является открытым и не использует патентованные методы, на которые опирается MP3.

Типичная интернет-радиостанция представляет собой веб-сайт, на котором выложены расписание передач, информация о ведущих и множество рекламы. Обычно можно найти один или несколько логотипов, указывающих на поддерживаемые аудиоформаты (или просто надпись «ПРОСЛУШАТЬ», если поддерживается только один формат). Значки с этими логотипами являются гиперссылками на метафайлы, о которых говорилось ранее.

Когда пользователь щелкает на одном из значков, пересылается короткий метафайл. Браузер, используя MIME-тип или расширение файла, определяет подходящее вспомогательное приложение (то есть проигрыватель). Метафайл записывается во временный файл, затем открывается программа-проигрыватель, которой передается имя временного файла. Видя содержащийся в нем URL (обычно со схемой *http* или *rtsp*, что позволяет обойти накладываемые брандмауэром ограничения и одновременно удовлетворить потребности популярных мультимедийных приложений), проигрыватель связывается с сервером и начинает работать как радиоприемник. Кстати говоря, аудиоданные передаются в виде одного потока, поэтому работа по *http* возможна, но только для радио: передавать видео, для которого характерно наличие по крайней мере двух потоков, с помощью *http* не удается — нужно что-нибудь типа *rtsp*.

Еще одной интересной особенностью интернет-радио является то, что практически все желающие, даже студенты, могут организовать собственную радиостанцию. Основные компоненты, необходимые для этого, изображены на рис. 7.31. Базой является обычный персональный компьютер со звуковой картой и микрофоном. Что касается программного обеспечения, то понадобится проигрыватель типа Winamp или Freeamp с подключаемым модулем для записи звука и кодеком выбранного формата (например, MP3 или Vorbis).

Поток аудиоданных, создаваемый станцией, отправляется на большой сервер мультимедиа в Интернете, который занимается распространением этого потока между множеством TCP-соединений. Сервер обычно работает с большим количеством маленьких радиостанций. Ведется список обслуживаемых радиостанций и предоставляется информация о том, какие из них в данных момент вещают. Потенциальные слушатели соединяются с этим сервером, выбирают станцию и получают данные по TCP. Существуют как коммерческие программы, включающие в себя все необходимые компоненты, так и открытые программные средства, такие как icecast. Разумеется, есть серверы, занимающиеся платной поддержкой радиостанций.

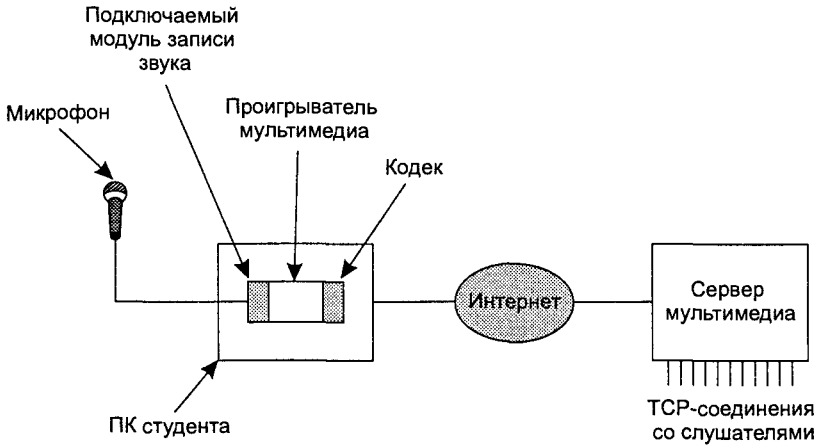


Рис. 7.31. Студенческая радиостанция

Передача речи поверх IP

Когда-то в стародавние времена общественная коммутируемая телефонная сеть была основным средством передачи речи; изредка она использовалась и для передачи данных. Однако с годами объем передаваемых данных все возрастал, и к 1999 году объемы данных и речи в телефонной сети уравнялись (и то, и другое можно измерить количеством бит в секунду, поскольку в глубинах телефонной системы используется цифровое PCM-кодирование). К 2002 году объем информационного трафика стал на порядок больше объема речевого трафика, и его рост (экспоненциальный!) продолжается. Между тем объем речевого трафика сохраняется практически неизменным (прирост составляет около 5 % в год).

В результате этого многие сетевые операторы, использующие системы с коммутацией пакетов, внезапно оказались заинтересованы в передаче речевых данных по сетям передачи данных. Требуемое для этого увеличение пропускной способности минимально, так как масштабы сетей с коммутацией пакетов позволяют передавать информационный трафик. Тем не менее, расходы среднего потребителя на телефонные переговоры могут превышать расходы на Интернет, поэтому сетевые операторы увидели в интернет-телефонии источник больших дополнительных доходов, не требующий прокладки новых кабелей. Так родилась **система передачи речи поверх IP**, или **интернет-телефония**.

H.323

С самого начала всем было понятно, что если каждый производитель станет изобретать собственный стек протоколов, система никогда работать не будет. Во избежание возникновения этой проблемы заинтересованные стороны объединились под покровительством Международного союза телекоммуникаций (ITU) и начали разработку единого стандарта. В 1996 году ITU выпустил рекомендации с индексом H.323 под заголовком «Видеотелефонные системы и оборудование

локальных вычислительных сетей, не предоставляющих гарантированное качество обслуживания». Такое название могло родиться только в телефонной индустрии. Данные рекомендации были пересмотрены в 1998 году, и новый вариант H.323 стал основой построения первых глобальных систем интернет-телефонии.

H.323 скорее дает общее представление об архитектуре систем интернет-телефонии, нежели описывает некий конкретный протокол. В документе можно найти множество ссылок на различные специализированные протоколы кодирования речи, установки соединения, передачи сигналов, данных и т. п., однако их описание не приводится. Общая модель изображена на рис. 7.32. В центре находится **шлюз**, соединяющий Интернет с телефонной сетью. Он поддерживает протокол H.323 со стороны Интернета и протоколы коммутируемой телефонной сети общего пользования с «телефонной» стороны. Устройства коммуникации называются **терминалами**. В локальной вычислительной сети может быть **машина-вратарь**, управляющая конечными узлами, находящимися под ее юрисдикцией (в ее **зоне**).

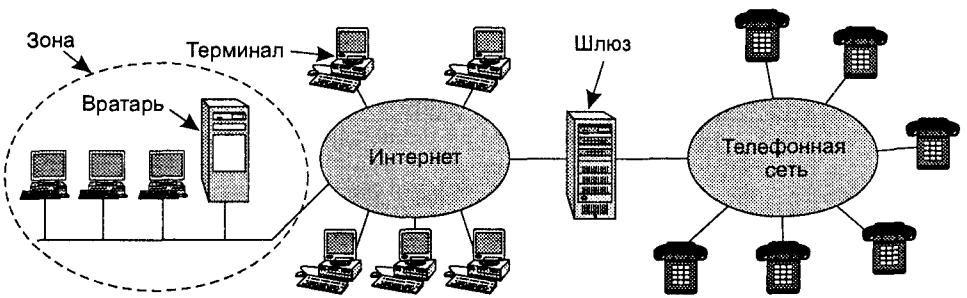


Рис. 7.32. Модель архитектуры H.323 для интернет-телефонии

Работу телефонной сети обеспечивает множество протоколов. Во-первых, необходим протокол кодирования и декодирования речи. Система PCM, которую мы изучали в главе 2, определена в рекомендациях ITU **G.711**. С ее помощью один голосовой канал кодируется 8-битными отсчетами с частотой 8000 раз в секунду. В результате получается 64-килобитный несжатый поток речевых данных. Все системы H.323 обязаны поддерживать G.711. Тем не менее, разрешена (но не является обязательной) поддержка и других протоколов кодирования речи. Они используют иные алгоритмы сжатия и приводят к несколько отличающемуся компромиссу между качеством и использованием пропускной способности. Например, в **G.723.1** берутся блоки по 240 отсчетов (30 мс речи) и используется кодирование с предсказанием, снижающее размер блоков до 24 или 20 байт. На выходе этого алгоритма получается поток со скоростью 6,4 или 5,3 Кбит/с (сжатие в 10 или 12 раз соответственно). Разумеется, качество звучания при этом гораздо ниже. Могут быть реализованы и другие алгоритмы кодирования.

Поскольку разрешено использование нескольких алгоритмов сжатия, необходим отдельный протокол, который позволил бы терминалам договориться об использовании одного из этих протоколов. Такой протокол называется **H.245**. Он

позволяет согласовать также другие параметры соединения, например битовую скорость. RTCP требуется для управления каналами RTP. Кроме того, нужны протоколы для установления и разрыва соединений, обеспечения тонального вызова, генерирования звуков звонков и других стандартных функций телефонной системы. Используется стандарт ITU **Q.931**. Терминалам нужен протокол для ведения переговоров с машиной-вратарем (если такая присутствует в локальной сети). Для этого в системе работает протокол **H.225**. Канал между ПК и вратарем, которым этот протокол управляет, называется каналом **RAS** (Registration/Admission/Status — Регистрация/Доступ/Статус). Он позволяет терминалам, кроме всего прочего, входить в зону и покидать ее, запрашивать и освобождать пропускную способность, обновлять данные о состоянии. Наконец, нужен протокол для непосредственной передачи данных. На этом участке работает RTP. Как обычно, управляется он RTCP. Иерархия всех этих протоколов показана на рис. 7.33.

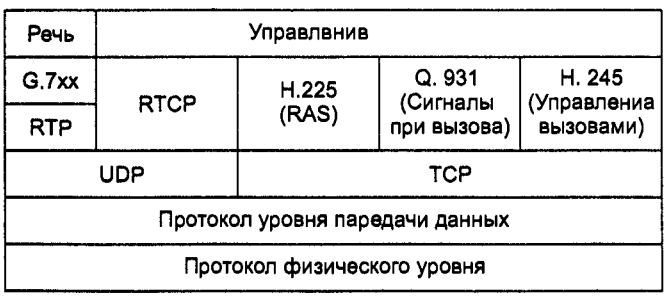


Рис. 7.33. Ствк протоколов H.323

Чтобы понять, как эти протоколы взаимодействуют друг с другом, рассмотрим случай ПК, являющегося терминалом локальной сети (с вратарем) и звонящего на удаленный телефон. Вначале компьютеру нужно найти вратаря, поэтому он рассылает широковещательным образом специальный UDP-пакет через порт 1718. Из ответа вратаря ПК узнает его IP-адрес. Теперь компьютер должен зарегистрироваться у вратаря. Для этого он посылает ему сообщение RAS в пакете UDP. После регистрации компьютер обращается к вратарю с просьбой (сообщение доступа RAS) о резервировании пропускной способности. Только после выделения этого ресурса можно начинать установку соединения. Предварительное резервирование пропускной способности позволяет вратарю ограничить число соединений, устанавливаемых на исходящей линии, что, в свою очередь, служит для обеспечения необходимого качества обслуживания.

Теперь ПК устанавливает TCP-соединение с вратарем, чтобы осуществить телефонный звонок. При установлении телефонного соединения используются традиционные протоколы телефонной сети, ориентированные на соединение. Поэтому требуется протокол TCP. С другой стороны, в телефонной системе нет никаких RAS, которые позволяли бы телефонным аппаратам заявлять о своем присутствии, поэтому разработчики H.323 могли применять как UDP, так и TCP

для передачи сообщений RAS, и они выбрали протокол с наименьшими накладными расходами — UDP.

Теперь, когда терминалу уже выделена пропускная способность, он может послать по TCP-соединению сообщение *SETUP* (стандарт Q.931). В нем указывается номер вызываемого абонента (или IP-адрес и порт, если вызывается удаленный компьютер). Вратарь отвечает Q.931-сообщением *CALL PROCEDING*, подтверждая тем самым факт корректного приема запроса. Затем вратарь пересылает сообщение *SETUP* на шлюз.

Шлюз, который является с одной стороны компьютером, а с другой — телефонным коммутатором, осуществляет обычный звонок на обычный телефон. Оконечная телефонная станция вызываемого абонента выполняет свою обычную работу (у абонента звенит звонок), а кроме этого отсылает обратно Q.931-сообщение *ALERT*, извещая ПК о том, что началась серия звонков. Когда абонент поднимает трубку, оконечная телефонная станция отправляет сообщение *CONNECT*, сообщая компьютеру о том, что соединение установлено.

После установки соединения вратарь перестает принимать участие в этом процессе, хотя шлюз, конечно, продолжает работать, обеспечивая двустороннюю связь. Пакеты идут в обход вратаря и направляются напрямую по IP-адресу шлюза. Эту ситуацию можно сравнить с обычным каналом между двумя сторонами. Это действительно просто соединение физического уровня, по которому передаются биты, и все. Ни одна из сторон не в курсе того, что представляет собой противоположная сторона.

Для переговоров о предпочитаемых параметрах соединения используется протокол H.245. При этом используется специальный управляющий канал H.245, который всегда открыт. Каждая из сторон начинает с объявления своих возможностей. Например, может сообщаться о поддержке видео (H.323 может поддерживать видео), конференц-связи, используемых кодеках и т. п. После того как каждая из сторон узнает возможности противоположной стороны, организуются два однонаправленных канала, с которыми связываются определенные кодеки и которым присваиваются определенные параметры. Поскольку на каждой из сторон может быть установлено разное оборудование, вполне возможна ситуация, когда каждый из однонаправленных каналов использует свой кодек. По достижении договоренности по всем вопросам можно начинать передачу данных (по протоколу RTP). Управление производится RTCP, контролирующим перегрузку. Если передаются видеоданные, RTCP занимается синхронизацией звукового и видеоряда. На рис. 7.34 показаны различные виды логических каналов. После того, как на одной из сторон вешают трубку, по каналу Q.931 передается сигнал окончания связи.

После разрыва соединения вызывающий ПК должен снова связаться с вратарем и послать ему сообщение RAS с запросом освобождения зарезервированной пропускной способности. Впрочем, вместо этого он может осуществить новый звонок.

Мы до сих пор ничего не говорили о качестве обслуживания, а ведь на самом деле это довольно существенный аспект успешной передачи речи поверх IP. Дело в том, что QoS не входит в область рассмотрения H.323. Если сеть, по которой

передаются данные, способна обеспечить стабильное соединение без флуктуации (джиттера) между ПК (например, с использованием методов, обсуждавшихся в главе 5) и шлюзом, значит, нам повезло и качество обслуживания будет хорошим. В противном случае качество будет, увы, плохое. В телефонной части системы используется PCM-кодирование, исключающее флуктуацию (джиттер).

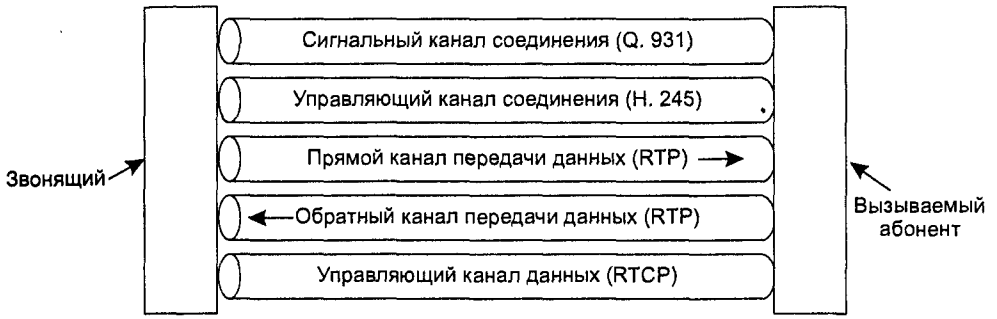


Рис. 7.34. Логические каналы между звонящим и вызываемым абонентами во время разговора

SIP — протокол запуска соединения

Стандарт H.323 был разработан ITU. В Интернет-сообществе многим он показался типичным продуктом телефонной компании: громоздким, сложным и недостаточно гибким. Было решено организовать специальный комитет IETF для создания более простой и гибкой системы передачи речи поверх IP. Основным результатом деятельности этого комитета стал протокол **SIP** (Session Initiation Protocol — протокол запуска соединения), описанный в RFC 3261. Протокол оговаривает способ установки телефонных соединений через Интернет, технологию организации систем для видеоконференций и способы создания других мультимедийных приложений. В отличие от H.323, представляющего собой целый набор протоколов, SIP — это единый модуль, способный взаимодействовать с разнообразными интернет-приложениями. Например, номера телефонов определяются в виде URL, то есть на веб-страницах можно размещать гиперссылки, щелкнув на которых, пользователь сможет установить телефонное соединение (примерно так же схема *mailto* позволяет написать электронное письмо и отправить его по указанному в ссылке адресу).

Протокол SIP позволяет устанавливать и двухсторонние соединения (то есть обычные телефонные соединения), и многосторонние (когда каждый из участников может как слушать собеседников, так и говорить), и широковещательные (когда один из участников говорит, а остальные могут только слушать). Во время сеанса связи могут передаваться аудио-, видео- или другие данные. Эта возможность используется, например, при организации сетевых игр с большим количеством участников в реальном времени. SIP занимается только установкой, управлением и разрывом соединений. Для передачи данных используются другие протоколы, например, RTP/RTCP. SIP — это протокол прикладного уровня, работающий поверх TCP или UDP.

Протокол SIP предоставляет разнообразные услуги, включая поиск вызываемого абонента (который может в данный момент быть далеко от своего домашнего компьютера), определение его возможностей, поддержку механизмов установки и разрыва телефонного соединения. В простейшем случае SIP устанавливает сеанс связи между компьютерами звонящего и вызываемого абонентов. Именно этот случай мы сейчас и рассмотрим.

Телефонные номера в SIP представляются в виде URL со схемой *sip*. Например, *sip:ilse@cs.university.edu* свяжет вас с пользователем по имени Ilse, хост которого определяется DNS-именем *cs.university.edu*. SIP URL могут содержать также адреса формата IPv4, IPv6 или реальные номера телефонов.

Протокол SIP является текстовым, он построен по модели HTTP. Одна из сторон посылает ASCII-сообщение, в котором первая строка содержит имя метода, а ниже следуют дополнительные строки, содержащие заголовки для передачи параметров. Многие заголовки взяты из стандарта MIME, что позволяет SIP взаимодействовать с существующими интернет-приложениями. Шесть методов, определяемых базовой спецификацией, перечислены в табл. 7.18.

Таблица 7.18. Методы SIP, определяемые базовой спецификацией

Метод	Описание
INVITE	Запрос запуска сеанса связи
ACK	Подтверждение запуска сеанса
BYE	Запрос окончания сеанса
OPTIONS	Опрос возможностей хоста
CANCEL	Отмена запроса
REGISTER	Информирование сервера переадресации о текущем местоположении пользователя

Для установки сеанса связи звонящий должен либо создать TCP-соединение с вызываемым абонентом и послать по нему сообщение *INVITE*, либо послать это же сообщение в UDP-пакете. В обоих случаях заголовки, содержащиеся во второй и всех последующих строках, описывают структуру тела сообщения, содержащего информацию о возможностях звонящего, типах мультимедиа и форматах. Если вызываемый абонент принимает звонок, он посылает в качестве ответа трехразрядный код результата типа HTTP (группы этих кодов перечислены в табл. 7.13, код 200 означает прием вызова). Следом за строкой с кодом результата вызываемый абонент может также сообщить данные о своих возможностях, типах мультимедиа и форматах.

Соединение устанавливается путем тройного рукопожатия, звонящий высылает *ACK* как для окончания работы протокола, так и для подтверждения приема кода 200.

Любая из сторон может послать запрос окончания сеанса связи, для этого используется метод *BYE*. Сеанс считается законченным после получения подтверждения от противоположной стороны.

Метод *OPTIONS* применяется для опроса возможностей машины. Обычно это делается перед запуском сеанса связи для того чтобы определить, поддержива-

ется ли тип сеанса, на который рассчитывает вызывающая сторона (например, передача голоса поверх IP).

Метод *REGISTER* относится к возможности протокола SIP разыскивать пользователя и соединяться с ним, даже если его нет дома. Сообщение, содержащее данный метод, отправляется на поисковый сервер SIP, хранящий данные о том, кто где находится в данный момент. Впоследствии с помощью этого сервера можно попробовать найти абонента. Операция переадресации, используемая при этом, показана на рис. 7.35. На этом рисунке мы видим, что звонящий отправляет сообщение *INVITE* на прокси-сервер. Это делает возможную переадресацию незаметной. Прокси пытается разыскать абонента и посылает *INVITE* по найденному адресу. Дальнейшее общение представляет собой коммутацию последовательности сообщений при «тройном рукопожатии». Сообщения *LOOKUP* и *REPLY* не входят в протокол SIP; на этой стадии может использоваться любой подходящий протокол в зависимости от типа поискового сервера.



Рис. 7.35. Использование прокси и серверов переадресации в протоколе SIP

SIP обладает также множеством других свойств, которые мы здесь не стали описывать подробно. Среди них есть функции ожидания вызова, отображения звонка, шифрования и идентификации звонящего. Кроме того, есть возможность звонить с компьютера на обычный телефон, если есть доступ к соответствующему шлюзу между Интернетом и телефонной системой.

Сравнительный анализ H.323 и SIP

H.323 и SIP во многом схожи, однако между ними есть и некоторые различия. Оба стандарта поддерживают как двухстороннюю, так и многостороннюю связь. Оконечным оборудованием могут служить как компьютеры, так и обычные телефоны. И там, и там стороны предварительно договариваются о параметрах, возможно шифрование данных и используются протоколы RTP/RTCP. Сводная сравнительная табл. 7.19 показывает все сходства и различия.

Несмотря на схожий набор свойств и характеристик, протоколы разительно отличаются друг от друга концепцией и философией. H.323 — это типичный тяжеловесный стандарт, характерный для телефонной индустрии. Он описывает целый стек протоколов и очень точно указывает, что разрешено, а что запрещено. Такой подход приводит к хорошо определенным протоколам на каждом уровне,

тем самым упрощается задача взаимодействия сетей. Однако платой за это оказывается большой, сложный и жесткий стандарт, тяжело адаптируемый к приложениям, которые появятся в будущем.

SIP, наоборот, представляет собой типичный интернет-протокол, работа которого основана на обмене короткими текстовыми строками. Это небольшой модуль, который хорошо взаимодействует с другими протоколами Интернета, однако несколько хуже согласуется с существующими сигнальными протоколами телефонной системы. Поскольку модель системы передачи данных поверх IP, предложенная IETF, использует модульный принцип, она оказывается достаточно гибкой и может легко адаптироваться к новым приложениям. Недостаток этого протокола связан с возможными проблемами межсетевого взаимодействия. Впрочем, специально для исключения этого недостатка часто проводятся встречи разработчиков и тестирование совместной работы различных сетей.

Таблица 7.19. Сравнение H.323 и SIP

Аспект	H.323	SIP
Разработчик	ITU	IETF
Совместимость с телефонной системой	Полная	В большой мере
Совместимость с Интернетом	Отсутствует	Присутствует
Архитектура	Монолитная	Модульная
Завершенность	Полный стек протоколов	SIP обеспечивает лишь установку соединения
Переговоры относительно параметров	Ведутся обеими сторонами	Ведутся обеими сторонами
Сигналы при вызове	Q.931 поверх TCP	SIP поверх TCP или UDP
Формат сообщений	Двоичный	ASCII
Передача мультимедийных данных	RTP/RTCP	RTP/RTCP
Многосторонняя связь	Есть	Есть
Мультимедийные конференции	Возможны	Невозможны
Адресация	Номер телефона или хоста	URL
Разрыв связи	Явный или разрыв TCP-соединения	Явный или по тайм-ауту
Постоянный обмен сообщениями	Нет	Есть
Шифрование данных	Есть	Есть
Объем описания стандарта	1400 страниц	250 страниц
Реализация	Громоздкая и сложная	Умеренная по объему
Статус	Широко распространен	Перспективен

Интернет-телефония — это новая перспективная область. Неудивительно поэтому, что уже выпущено несколько книг, посвященных этой теме. Среди них

стоит упомянуть (Collins, 2001; Davidson и Peters, 2000; Kumar и др., 2001; Wright, 2001). В журнале *Internet Computing* за май-июнь 2001 можно найти несколько статей, посвященных передаче речи поверх IP.

Видео

Итак, мы достаточно долго обсуждали услуги, предоставляемые компьютерными сетями человеческому уху. Пора обратиться к глазам (нет-нет, за этим разделом не следует раздел, посвященный носу). Сетчатка глаза человека обладает инерционными свойствами, то есть яркое изображение, быстро появившееся на сетчатке, остается на ней несколько миллисекунд, прежде чем угаснуть. Если последовательности одинаковых изображений появляются и исчезают с частотой около 50 Гц, глаз человека не замечает, что он смотрит на дискретные изображения. Все видео (то есть телевизионные) системы используют этот принцип для создания движущихся изображений.

Аналоговые системы

Чтобы понять, как работают видеосистемы, лучше всего начать с рассмотрения простого старомодного черно-белого телевидения. Для представления двухмерного изображения в виде одномерной зависимости электрического напряжения от времени камера быстро сканирует электронным лучом изображение, разбивая его на горизонтальные линии и запоминая по мере продвижения интенсивность света. Закончив сканирование **кадра**, луч возвращается в исходную точку. Зависимость интенсивности света от времени — это та функция, значения которой распространяются в виде телевизионного сигнала. Телевизионные приемники для восстановления изображения повторяют процесс сканирования. Путь, который проходит электронный луч в передающей камере и принимающей телевизионной трубке, показан на рис. 7.36. (Что касается камер с ПЗС-матрицами (с зарядовой связью), то их принцип работы основан на интегрировании, а не на сканировании, однако некоторые камеры и все мониторы все же занимаются именно сканированием.)

В разных странах приняты различные стандарты, описывающие точные параметры сканирования В системе, принятой в Северной и Южной Америке, а также Японии, экран разбивается на 525 горизонтальных линий развертки, соотношение горизонтального и вертикального размеров экрана составляет 4:3, кадры передаются с частотой 30 кадров в секунду. Европейская система PAL/SECAM подразумевает разбиение кадра на 625 линий, размеры экрана у нее также 4:3, а частота кадров составляет 25 кадров в секунду. В обеих системах самые верхние и самые нижние линии кадра не показываются (это связано с попыткой адаптировать прямоугольную картинку к круглой форме электронно-лучевой трубки). На экране телевизоров показываются только 483 из 525 линий развертки для системы NTSC и 576 из 625 — для системы PAL/SECAM. Во время обратного хода луч выключается, и этот интервал времени многие телевизионные станции (особенно в Европе) используют для передачи телетекста (текстовых страниц, содержащих новости, прогнозов погоды, биржевых сводок и т. п.).

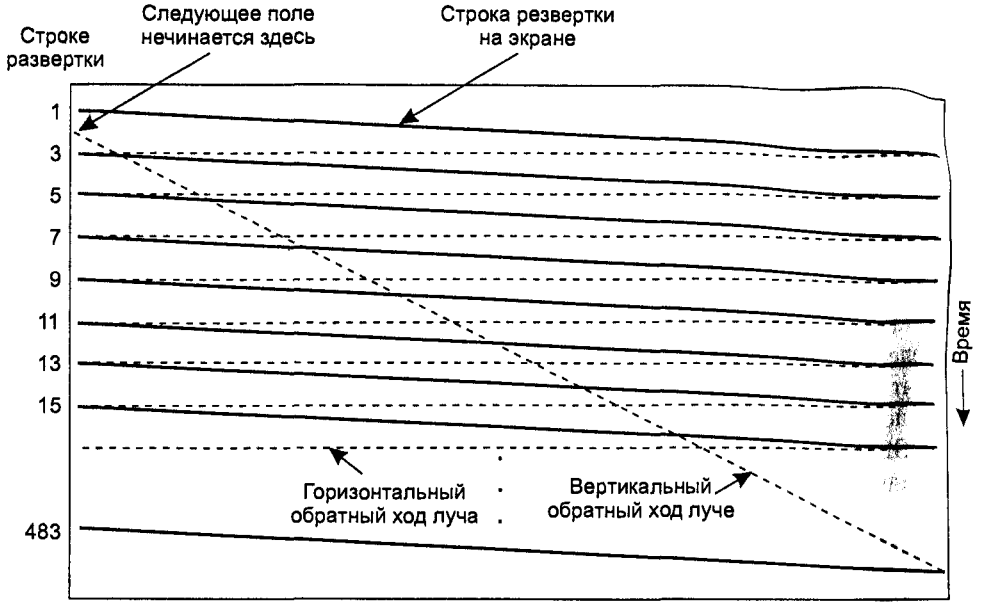


Рис. 7.36. Путь сканирующего луча в видео- и телесистемах NTSC

Хотя частоты в 25 кадров в секунду достаточно, чтобы передать плавное движение, при такой частоте кадровой развертки многие зрители (особенно пожилые) заметят мигание изображения, связанное с тем, что сетчатка глаза успеет восстановиться, прежде чем появится новый кадр. Увеличение частоты кадров потребовало бы увеличения объемов хранимой и передаваемой информации. Вместо этого был выбран другой подход. Линии развертки показываются на экране телевизора не подряд, а через одну: сначала все нечетные, а затем все четные. Каждый такой полукадр называют **полем**. Эксперименты показали, что хотя люди замечают мерцание при частоте 25 кадров в секунду, при частоте 50 полей в секунду оно уже не заметно. Этот метод называется **чересстрочной разверткой**. Телевидение или видеосистема, в которой не используется чересстрочная развертка, называется **поступательным**. Обратите внимание на то, что движущиеся изображения показываются со скоростью 24 кадра в секунду, но каждый кадр полностью виден в течение 1/24 секунды.

В цветном видео используется тот же принцип развертки, что и в монохромном (черно-белом), с той разницей, что вместо одного луча изображение представляется синхронно двигающимися тремя лучами: красным, зеленым и синим (RGB — red, green, blue). Комбинации этих трех цветов оказывается достаточно для передачи любого цвета благодаря особенностям устройства человеческого глаза. При передаче по каналу связи эти три цветовых сигнала объединяются в единый **смешанный** сигнал.

При изобретении цветного телевидения разные страны решили выбрать различные методы, что привело к созданию систем, до сих пор остающихся несо-

вместимыми. (Не следует путать различные методы кодирования цвета с различными стандартами записи видеосигнала, такими как VHS, Betamax и P2000.) Во всех странах правительство требовало от разработчиков цветного телевидения, чтобы программы, передаваемые в цвете, могли также приниматься на черно-белых телевизорах. Поэтому простейшая схема, представляющая собой просто раздельное кодирование RGB-сигналов, была неприемлема. Кроме того, такая схема не является самой эффективной.

Первая система цветного телевидения была стандартизована в США Национальным комитетом по телевизионным стандартам, **NTSC** (National Television Standards Committee). Стандарт получил название по сокращенному имени комитета. Спустя несколько лет в Европе были разработаны свои системы цветного телевидения. К этому времени технология стала уже значительно совершеннее, что привело к созданию систем с более высокой помехозащищенностью и улучшенной цветопередачей. Во Франции и Восточной Европе был принят стандарт **SECAM** (SEquence de Couleurs Avec Memoire — последовательные цвета с запоминанием), а в остальной части Европы — стандарт **PAL** (Phase Alternation Line — построчное изменение фазы). Различие в качестве передачи цвета между системами NTSC и PAL/SECAM послужило причиной появления шутки, согласно которой NTSC следует расшифровывать как «никогда не повторяющийся цвет» (Never Twice the Same Color).

Для совместимости с черно-белыми телевизионными приемниками во всех трех системах сигналы RGB линейно объединяются в сигнал **яркости** и два сигнала **цветности**. Однако в каждой из этих систем эти сигналы формируются с использованием различных коэффициентов. Экспериментально доказано, что человеческий глаз обладает гораздо более высокой чувствительностью к изменению сигнала яркости, чем к изменению сигнала цветности. В результате сигнал цветности не обязательно кодировать так же точно, как сигнал яркости. Поэтому было решено передавать сигнал яркости в том же формате, что и черно-белый сигнал, а два узкополосных сигнала цветности — отдельно на более высокой частоте. Телевизоры обычно снабжаются регуляторами яркости и насыщенности цвета. Знакомство с сигналами яркости и цветности важно для понимания принципов действия алгоритмов видеосжатия.

В последние годы появился значительный интерес к телевидению высокой четкости **HDTV** (High Definition TeleVision). Системы HDTV отличаются от обычных систем телевидения примерно двукратным увеличением числа строк развертки. США, Европа и Япония уже успели создать собственные, как всегда совершенно не совместимые между собой, системы HDTV. Базовые принципы HDTV, касающиеся сканирования, яркости и цветности, подобны уже существующим системам. Однако в этих системах в основном используется формат экрана 16:9 вместо 4:3, что лучше соответствует стандарту, принятому в кинематографе для фильмов, снятых на ленте шириной 35 мм и имеющих формат 3:2.

Цифровые видеосистемы

Простейшая форма представления цифрового видео заключается в последовательности кадров, состоящих из прямоугольной сетки элементов рисунка (**пиксе-**

лов). Если каждый пиксел представлять одним битом, то мы получим бинарное изображение, состоящее только из черного и белого цветов, без оттенков серого. Качество такого изображения можно оценить, если послать фотографию по факсу: оно будет ужасным. (Попробуйте сделать так, если у вас есть возможность. Можно, кстати, вместо этого сделать фотокопию цветной фотографии на копировальном аппарате, который не умеет осуществлять преобразование в растровый формат.)

Следующий шаг заключается в использовании 8 бит на пиксел, тогда можно получить, например, 256 различных оттенков серого цвета, что достаточно для черно-белого телевидения высокого качества. В цветном телевидении хорошие системы используют по 8 бит на каждый из трех цветов RGB несмотря на то, что почти все системы передают смешанный сигнал. Хотя числом, состоящим из 24 бит, можно обозначить около 16 миллионов цветовых оттенков, человеческий глаз не в состоянии даже различить такое огромное количество цветовых оттенков, не говоря уж о больших количествах. Цифровые цветные изображения создаются тремя сканирующими лучами, по одному на цвет. Геометрия сканирования кадра полностью совпадает с аналоговой системой, показанной на рис. 7.36, с той разницей, что непрерывные линии развертки заменяются аккуратными рядами дискретных пикселов.

Для сглаживания при передаче движения, как и в аналоговом видео, в цифровом видео необходимо отображать по меньшей мере 25 кадров в секунду. Однако, поскольку качественные мониторы обычно сами сканируют по несколько раз изображения, хранящиеся в их памяти, с частотой 75 и более кадров в секунду, проблема мерцания изображения решается на уровне монитора сама собой, и чередование строк в цифровом видео не требуется.

Другими словами, плавность движущегося изображения определяется количеством *различающихся* изображений в секунду, тогда как мерцание зависит от частоты перерисовки экрана. Не следует путать эти два параметра. Неподвижное изображение, рисуемое с частотой 20 кадров в секунду, не будет дергаться, но будет мерцать, поскольку возбуждение сетчатки глаза успеет угаснуть, прежде чем появится следующий кадр. Фильм, в котором выводится 20 различных кадров в секунду, каждый из которых повторяется по четыре раза, не будет мерцать, но будет заметно отсутствие плавности движений.

Важность этих двух параметров станет ясна, если мы попробуем оценить пропускную способность, необходимую для передачи цифрового видеосигнала по сети. Во всех современных компьютерных мониторах используется соотношение размеров экрана 4:3, поэтому в них вполне могут использоваться недорогие серийные электронно-лучевые трубки, предназначенные для телевизоров. Стандартные используемые разрешения экрана составляют 1024×768, 1280×960 и 1600×1200. Для показа цифрового видео даже на экране с самым низким из этих разрешений при 24 битах на пиксел и 25 кадрах в секунду потребуется передавать поток данных со скоростью 472 Мбит/с. С этим может справиться лишь канал SONET OC-12, однако подведение таких линий в каждый дом пока что не стоит на повестке дня. Удвоение частоты, позволяющее избавиться от мерцания, выглядит еще менее привлекательно. Оптимальное решение состоит в том, что-

бы передавать 25 кадров в секунду, и позволить компьютеру самому хранить эти кадры и отображать их с удвоенной частотой. В обычном широковещательном телевидении такая стратегия почти не используется, так как у обычных телевизоров нет памяти. Кроме того, для хранения аналоговых сигналов в ОЗУ необходимо сначала преобразовать их в цифровую форму, что потребует дополнительного оборудования. Таким образом, чередование строк применяется в обычном широковещательном телевидении, но оно не нужно в цифровом видео.

Сжатие видеоданных

Итак, теперь должно быть очевидно, что о передаче мультимедийной информации в несжатом виде не может быть и речи. Есть лишь одна надежда: на повсеместное сжатие данных. К счастью, за несколько последних десятилетий было разработано множество методов сжатия, делающих возможной передачу мультимедийной информации. В данном разделе мы рассмотрим некоторые методы сжатия движущихся изображений.

Для всех систем сжатия требуется два алгоритма: один для компрессии данных источником информации, а другой — для распаковки ее получателем. В литературе эти алгоритмы называют, соответственно, алгоритмами **кодирования** и **декодирования**. Мы также будем пользоваться здесь этой терминологией.

Эти алгоритмы обладают определенной асимметрией, которую следует хорошо понять. Во-первых, во многих приложениях мультимедийный документ, например фильм, кодируется всего один раз, при его создании и помещении на мультимедийный сервер, но декодируется тысячи раз при просмотре пользователями. Эта асимметрия означает, что алгоритм кодирования может быть довольно медленным и требовать дорогого оборудования, тогда как алгоритм декодирования должен быть быстрым и должен работать даже на дешевом оборудовании. В конце концов, оператор мультимедийного сервера может позволить себе арендовать на пару недель параллельный суперкомпьютер, чтобы закодировать всю свою видеотеку, но нельзя требовать от клиентов, чтобы они арендовали суперкомпьютер на 2 часа для просмотра видеофильма. В большинстве современных систем сжатия высокая скорость декодирования является основной задачей, достигаемой даже ценой большой сложности и медленности алгоритма кодирования.

С другой стороны, для мультимедийных данных реального времени, например видеоконференций, медленное кодирование неприемлемо. Кодирование здесь должно происходить «на лету», в режиме реального времени. Поэтому в системах мультимедиа реального времени применяются другие алгоритмы или их параметры, чем при хранении фильмов на дисках. Чаще всего используется существенно меньшее сжатие.

Еще один аспект асимметрии состоит в том, что процесс кодирования/декодирования не обязан быть обратимым. Это значит следующее. При сжатии обычного файла, его передаче и декомпрессии получатель обязан получить копию, совпадающую с оригиналом с точностью до бита. При передаче мультимедиа аб-

солютная точность не требуется. Вполне допустимы небольшие отклонения видеосигнала от оригинала после его кодирования и декодирования. Система, в которой декодированный сигнал не точно соответствует кодированному оригиналу, называется системой **с потерями**. Если же выходной сигнал идентичен входному, то такая система называется системой **без потерь**. Системы с потерями являются важными, так как, допуская небольшие потери информации, можно достичь очень большого коэффициента сжатия.

Стандарт JPEG

Видеосигнал представляет собой обычную последовательность изображений (сопровождаемую звуком). Если мы сможем найти хороший алгоритм сжатия неподвижного изображения, нам с не меньшим успехом удастся сжать видеоданные. Уже существуют достаточно хорошие алгоритмы сжатия изображений, поэтому мы начнем изучение принципов сжатия видеоданных именно с этого. Стандарт **JPEG** для сжатия неподвижных изображений с непрерывно меняющимся цветом (например, фотографий) был разработан группой экспертов в области фотографии JPEG (Joint Photography Experts Group — Объединенная группа экспертов по машинной обработке фотоизображений). Эта группа работала под совместным покровительством Международного союза телекоммуникаций ITU, Международной организации по стандартизации ISO и еще одной организации, занимающейся стандартизацией, — IEC (International Electrotechnical Commission — Международная электротехническая комиссия). Стандарт JPEG является очень важным для мультимедиа, так как в первом приближении мультимедийный стандарт для движущихся изображений, MPEG, представляет собой просто кодирование каждого кадра отдельно алгоритмом JPEG плюс некоторые дополнительные процедуры межкадрового сжатия и обнаружения движения. Стандарт JPEG определен как международный стандарт 10918.

У метода сжатия JPEG есть четыре режима и множество параметров. Он больше напоминает номенклатуру товаров в небольшом магазине, чем просто один алгоритм. Для нас сейчас важен только последовательный режим с потерями, показанный на рис. 7.37. Кроме того, мы сконцентрируемся лишь на использовании JPEG для кодирования 24-битового RGB-видеоизображения и опустим некоторые незначительные детали.



Рис. 7.37. Работа алгоритма JPEG в последовательном режиме с потерями

Первый этап кодирования изображения алгоритмом JPEG представляет собой подготовку блока. Рассмотрим частный случай кодирования 24-битового RGB-видеоизображения размером 640×480 пикселов, показанного на рис. 7.38, *а*. Поскольку разделение на яркость и цветность позволяет сильнее сжать изображение,

мы сосчитаем сначала яркость Y и два значения цветности I и Q (по стандарту NTSC) в соответствии со следующими формулами:

$$Y = 0{,}30\ R + 0{,}59\ G + 0{,}11\ B;$$

$$I = 0{,}60\ R - 0{,}28\ G - 0{,}32\ B;$$

$$Q = 0{,}21\ R - 0{,}52\ G + 0{,}31\ B.$$

В стандарте PAL значения цветности называются U и V, и коэффициенты используются другие, но идея та же самая. Стандарт SECAM отличается и от NTSC, и от PAL.

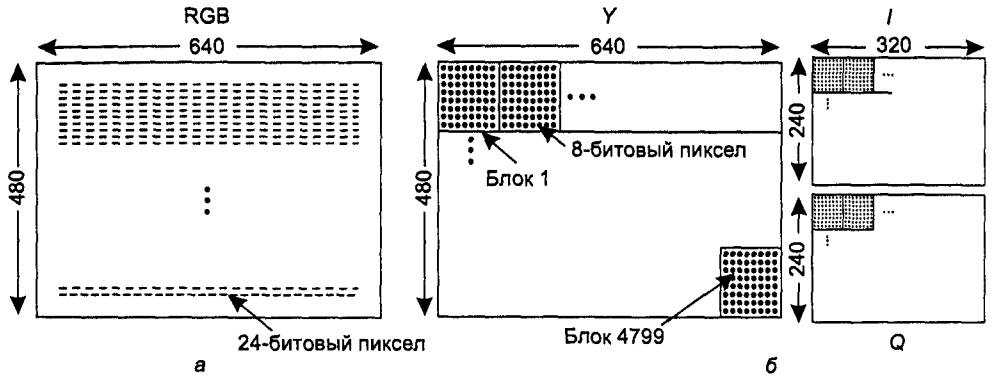


Рис. 7.38. Исходные данные в формате RGB (*а*); после подготовки блока (*б*)

Для значений Y, I и Q строятся отдельные матрицы с элементами, значения которых лежат в диапазоне от 0 до 255. Затем значения цветности I и Q усредняются по квадратным блокам из четырех пикселов, что уменьшает размеры матриц цветности в 4 раза до размера 320×240. Это сжатие является преобразованием с потерями, но человеческий глаз его почти не замечает, так как чувствительность глаза к яркости значительно выше, чем к цветности. Тем не менее, уже на этом этапе общий объем данных уменьшается вдвое. Затем из каждого элемента вычитается число 128, чтобы переместить число 0 в середину диапазона. Наконец, каждая матрица разбивается на квадраты по 8×8 пикселов. Таким образом, матрица Y состоит из 4800 квадратных блоков, а матрицы I и Q содержат по 1200 блоков каждая, как показано на рис. 7.38, *б*.

На втором этапе кодирования изображения алгоритмом JPEG к каждому из 7200 квадратных блоков отдельно применяется **дискретное косинусное преобразование**. На выходе получается 7200 матриц 8×8 коэффициентов дискретного косинусного преобразования (ДКП). Элемент (0, 0) такой матрицы представляет собой среднее значение блока. Остальные элементы содержат информацию о спектральной интенсивности каждой пространственной частоты. В теории дискретное косинусное преобразование является преобразованием без потерь (обратимым), но на практике использование чисел с плавающей точкой и трансцендентных функций приводит к ошибкам округления, в результате чего часть информации теряется. Обычно элементы матрицы ДКП-коэффициентов быстро убывают с расстоянием от элемента (0, 0), как показано на рис. 7.39.

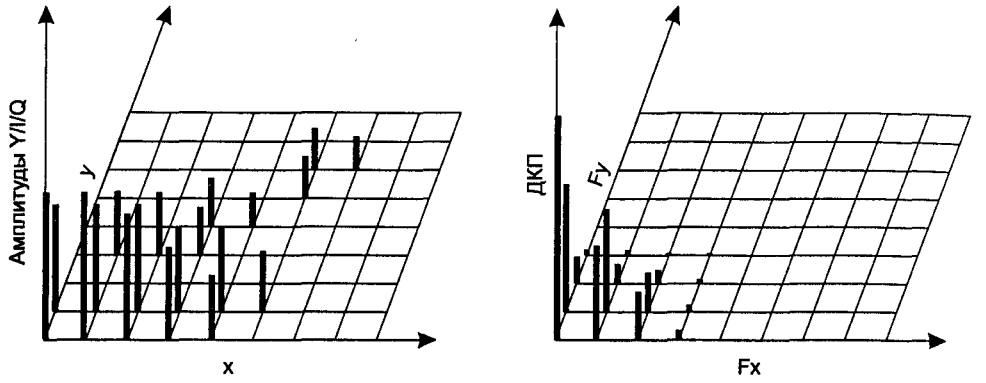


Рис. 7.39. Один блок матрицы Y (*а*); ДКП-коэффициенты (*б*)

По завершении дискретного косинусного преобразования алгоритм JPEG переходит к этапу 3, называемому **квантованием**, в котором наименее важные ДКП-коэффициенты удаляются. Это преобразование (с потерями) выполняется делением всех коэффициентов ДКП-матрицы на табличные весовые коэффициенты. Если все весовые коэффициенты равны 1, то это преобразование ничего не меняет. Однако при резком росте весовых коэффициентов по мере удаления от элемента матрицы (0, 0) более высокие пространственные частоты быстро теряются.

ДКП-коэффициенты

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Квантованные коэффициенты

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Таблица квантования

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Рис. 7.40. Квантование ДКП-коэффициентов

Пример этого этапа работы алгоритма показан на рис. 7.40. Здесь мы видим исходную ДКП-матрицу, таблицу квантования и результат деления каждого

ДКП-элемента на соответствующий элемент таблицы квантования. Значения в таблице квантования не являются частью стандарта JPEG. Каждое приложение должно содержать свою таблицу весовых коэффициентов, что позволяет ему контролировать соотношение потерь и коэффициента сжатия.

На четвертом этапе значение, содержащееся в элементе (0, 0) каждого блока (в левом верхнем углу квадрата), заменяется его отклонением относительно значения в предыдущем блоке. Так как эти значения представляют собой усредненные величины своих блоков, они должны меняться медленно, следовательно, полученные в результате разности должны быть невелики. Для остальных значений разности не вычисляются. Значения элементов (0, 0) называются DC-компонентами, а остальные элементы — AC-компонентами.

На пятом этапе 64 элемента блока выстраиваются в ряд, к которому применяется кодирование длин серий. Чтобы сконцентрировать нули в конце ряда, сканирование блока выполняется зигзагом, как показано на рис. 7.41. В нашем примере в конце блока группируется 38 нулей. Эта строка нулей заменяется просто числом 38. В этой замене и состоит результат работы метода, называемого **кодированием длин серий**.

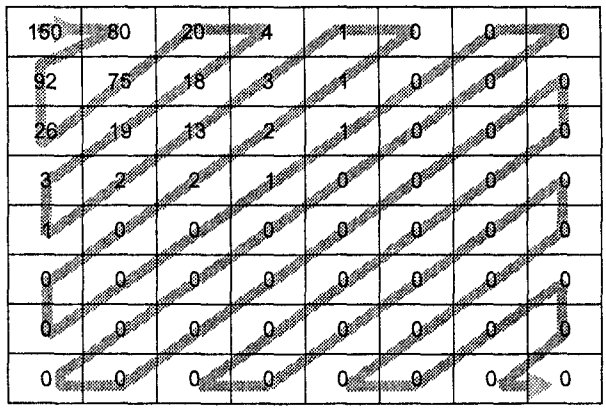


Рис. 7.41. Порядок передачи квантованных значений

Теперь у нас есть список чисел, представляющий изображение (в трансформированном виде). На шестом этапе применяется код Хаффмана (Huffman), присваивающий часто встречающимся последовательностям короткие коды, а редко встречающимся — более длинные коды.

Алгоритм JPEG может показаться сложным, но это только потому, что он и в самом деле является таковым. Тем не менее, он широко применяется, так как позволяет сжимать фотографии в 20 и более раз. Для декодирования сжатого изображения нужно выполнить все те же операции в обратном порядке. Алгоритм JPEG почти симметричен: декодирование занимает столько же времени, сколько и кодирование. Это справедливо далеко не для всех алгоритмов, как мы увидим позже.

Стандарт MPEG

Наконец мы подходим к ключевому вопросу мультимедиа — стандартам **MPEG** (Motion Pictures Experts Group — экспертная группа по вопросам движущегося изображения). Эти стандарты, ставшие международными в 1993 году, описывают основные алгоритмы, используемые для сжатия видеофильмов. Поскольку фильмы содержат как изображение, так и звук, алгоритмы MPEG занимаются сжатием и того, и другого. Принципы сжатия аудиоданных и неподвижных изображений мы рассмотрели ранее, поэтому этот раздел мы посвятим обсуждению алгоритмов сжатия видео.

Первым законченным стандартом стал стандарт MPEG-1 (Международный Стандарт 11172). Его целью было создание выходного потока данных качества бытового видеомагнитофона (352×240 для NTSC) на скорости 1,2 Мбит/с. Мы уже видели ранее, что несжатый поток видеоинформации может составлять 472 Мбит/с (для экрана размером 1024×768). Для экрана 352×240 потребуется в 9,3 раза меньший поток данных, то есть 50,7 Мбит/с. Тем не менее и это значение намного превышает 1,2 Мбит/с, так что данную задачу никак нельзя назвать тривиальной. Видеофильм в формате MPEG-1 можно передавать по витой паре на умеренные расстояния. Кроме того, фильмы в этом формате можно хранить на компакт-дисках.

Следом появился стандарт MPEG-2 (Международный стандарт 13818), разрабатывавшийся для сжатия видеофильмов качества широковещания до скорости потока от 4 до 6 Мбит/с, что позволяло передавать этот фильм в цифровом виде по стандартному телевизионному каналу NTSC или PAL. Позднее стандарт MPEG-2 был расширен для поддержки видеосигнала более высокого разрешения, включая HDTV. Сейчас он очень популярен, поскольку составляет основу DVD и цифрового спутникового телевидения.

Основные принципы стандартов MPEG-1 и MPEG-2 одинаковы. В первом приближении стандарт MPEG-2 является расширением стандарта MPEG-1 с дополнительными возможностями, форматами кадров и вариантами кодирования. Сначала мы рассмотрим MPEG-1, а затем MPEG-2.

Алгоритм MPEG-1 состоит из трех частей: аудио- и видеошифраторов и системы, объединяющей эти два потока данных, как показано на рис. 7.42. Системы кодирования аудио и видео работают независимо, в результате чего возникает проблема синхронизации этих потоков в приемнике. Для решения этой проблемы используются системные часы, работающие на частоте 90 кГц. Показания этих часов хранятся в 33-битных счетчиках, что позволяет указывать время в интервале 24 часов. Эти отметки времени вносятся в закодированный результат и передаются получателю, который может использовать их для синхронизации изображения и звука.

Перейдем к рассмотрению сжатия видеоинформации с помощью алгоритма MPEG-1. В видеофильмах имеется избыточность двух типов: пространственная и временная. Алгоритм сжатия MPEG-1 использует оба типа. Чтобы использовать пространственную избыточность, можно просто кодировать каждый кадр отдельно алгоритмом JPEG. Иногда такой подход применяется, особенно если нужен прямой доступ к каждому кадру, например, при редактировании видеоизображе-

ния. Этот режим сжатия позволяет снизить поток данных до скорости от 8 до 10 Мбит/с.

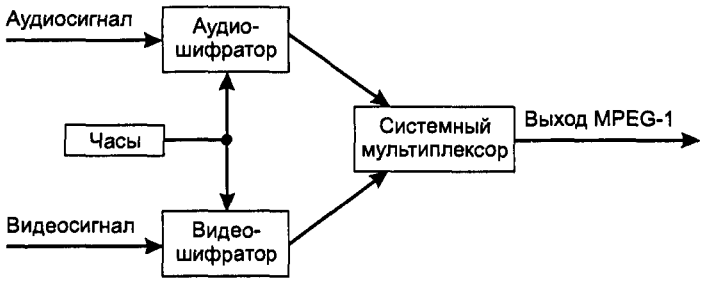


Рис. 7.42. Синхронизация аудио- и видеопотоков в MPEG-1

Дополнительного сжатия можно достичь, используя преимущество того факта, что идущие подряд кадры часто бывают почти идентичны. Может показаться, что этот эффект не столь уж велик, так как в большинстве фильмов сцены часто меняются — в среднем каждые 3–4 с.. Тем не менее даже последовательность в 75 очень близких кадров позволяет применить значительно большее сжатие, нежели с помощью компрессии отдельных кадров алгоритмом JPEG.

В сценах, в которых камера и задний план неподвижны и один или два актера медленно двигаются, почти все пикселы в соседних кадрах будут идентичны. В этом случае простое вычитание каждого кадра из предыдущего и обработка разности алгоритмом JPEG даст достаточно хороший результат. Однако этот метод плохо подходит к сценам, в которых камера поворачивается или наезжает на снимаемый объект. Для таких сцен необходим какой-то способ компенсировать это движение камеры. В этом и состоит основное отличие MPEG от JPEG.

Выходной поток MPEG-1 состоит из кадров четырех следующих типов:

1. I (Intracoded — автономные) — независимые неподвижные изображения, кодированные алгоритмом JPEG.
2. P (Predictive — предсказывающие) — содержат разностную информацию относительно предыдущего кадра.
3. B (Bidirectional — двунаправленные) — содержат изменения относительно предыдущего и последующего кадров.
4. D (DC-coded — усредненные) — содержат усредненное по блоку значение, используемые для быстрой перемотки вперед.

I-кадры представляют собой обычные неподвижные изображения, кодированные алгоритмом JPEG с использованием полного разрешения яркости и половинного разрешения для обоих сигналов цветности. I-кадры должны периодически появляться в выходном потоке по трем причинам. Во-первых, должна быть возможность просмотра фильма не с самого начала. Зритель, пропустивший первый кадр, не сможет декодировать все последующие кадры, если все кадры будут зависеть от предыдущих и I-кадры не будут время от времени включаться в поток. Во-вторых, дальнейшее декодирование фильма станет невозможным в слу-

чае ошибки при передаче какого-либо кадра. В-третьих, наличие таких кадров существенно упростит индикацию во время быстрой перемотки вперед или назад. По этим причинам I-кадры включаются в выходной поток примерно один-два раза в секунду.

P-кадры, напротив, представляют собой разность между соседними кадрами. Они основаны на идее **макроблоков**, покрывающих 16×16 пикселов в пространстве яркости и 8×8 пикселов в пространстве цветности. При кодировании макроблока в предыдущем кадре ищется наиболее близкий к нему макроблок.

Пример использования макроблоков показан на рис. 7.43. Здесь мы видим три последовательных кадра с одинаковым задним планом, различающихся только положением человека. Макроблоки, содержащие задний план, будут полностью совпадать друг с другом, но макроблоки, содержащие человека, будут смещаться на некоторую величину. Именно для этих макроблоков алгоритм должен найти максимально похожие макроблоки из предыдущего кадра.

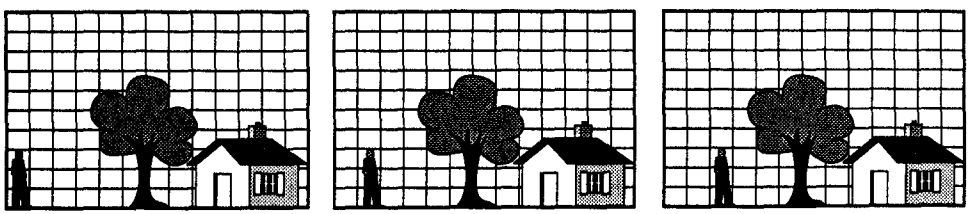


Рис. 7.43. Три последовательных кадра

Стандарт MPEG-1 не указывает, как искать, насколько далеко искать или насколько хорошо должен подходить похожий макроблок. Все эти вопросы оставлены на усмотрение разработчика программы, реализующей стандарт MPEG-1. Например, макроблок можно искать в предыдущем кадре в текущей позиции с заданными смещениями по горизонтали и вертикали. Для каждой позиции может вычисляться количество совпадений матрицы яркости. Позиция с наибольшим значением совпадений будет объявляться победительницей при условии, что это значение превосходит некое пороговое значение. В противном случае макроблок будет считаться не найденным. Возможны, конечно, и значительно более сложные алгоритмы.

Макроблок, для которого найден похожий на него макроблок в предыдущем кадре, кодируется в виде разности значений яркости и цветности. Затем матрицы разностей подвергаются дискретному косинусному преобразованию, квантованию, кодированию длин серий и кодированию Хаффмана так же, как это делает алгоритм JPEG. В выходном потоке макроблок представляется в виде вектора сдвига (насколько далеко сдвинулся макроблок по горизонтали и вертикали от положения в предыдущем кадре), за которым следует список чисел, кодированных по Хаффману. Если же в предыдущем кадре не нашлось подходящего макроблока, текущее значение кодируется алгоритмом JPEG, как в I-кадре.

Очевидно, что этот алгоритм обладает сильной асимметричностью. В принципе, программа может искать соответствие макроблоку в любой позиции предыдущего кадра. Такой подход позволит минимизировать выходной поток MPEG-1 за счет очень медленного кодирования. Следовательно, такой метод применим для одноразового кодирования при создании фильмотеки, но не годится для видеоконференций реального времени.

Аналогично разработчики каждой реализации могут самостоятельно определять алгоритм сравнения макроблоков. Эта свобода позволяет разработчикам соревноваться в качестве и скорости своих алгоритмов при полной совместимости создаваемых алгоритмами потоков MPEG-1. Независимо от алгоритма поиска конечный выходной поток будет состоять либо из текущего блока, закодированного по стандарту JPEG, либо из закодированной в соответствии с тем же стандартом JPEG разности текущего и одного из предыдущего кадров, расстояние до которого должно быть указано.

Декодирование потока MPEG-1 выполняется довольно просто. Декодирование I-кадров аналогично декодированию JPEG-изображений. Чтобы декодировать P-кадр, декодер должен сохранить в буфере предыдущий кадр, а затем во втором буфере построить новый кадр из макроблоков как в абсолютной, так и в относительной кодировке. Кадры собираются макроблок за макроблоком.

B-кадры аналогичны P-кадрам — с той лишь разницей, что позволяют привязывать макроблок либо к предыдущему, либо к следующему кадру. Такая дополнительная свобода позволяет достичь хорошей компенсации движения. Для декодирования B-кадров необходимо удерживать в памяти сразу три кадра: предыдущий, текущий и следующий. Хотя B-кадры позволяют добиться наибольшего сжатия, они поддерживаются не всеми реализациями.

D-кадры используются только для индикации изображения с низким разрешением при быстрой перемотке фильма. Выполнение обычного декодирования видеоформата MPEG-1 уже само по себе требует достаточной мощности процессора. Выполнять же это декодирование в десять раз быстрее при поиске практически нереально. Каждый D-кадр представляет собой просто среднее значение блока, без дальнейшего кодирования, что упрощает его отображение в режиме реального времени.

Перейдем теперь к стандарту MPEG-2. В основном, этот стандарт схож со стандартом MPEG-1, но он не поддерживает D-кадры. Кроме того, в нем применяется дискретное косинусное преобразование матрицы размером не 8×8, а 10×10, что увеличивает число ДКП-коэффициентов в полтора раза, результатом чего является и более высокое качество. Поскольку стандарт MPEG-2 предназначается как для широковещательного телевидения, так и для DVD, в нем имеется поддержка и поступательного, и чересстрочного отображения (в отличие от MPEG-1, поддерживающего только поступательное отображение). У этих двух стандартов имеются также и другие различия, правда, не столь значительные.

В отличие от стандарта MPEG-1, поддерживающего только одно разрешение, стандарт MPEG-2 поддерживает четыре: низкое (352×240), основное (720×480), высокое-1440 (1440×1152) и высокое (1920×1080). Низкое разрешение предназначено для бытовых видеомагнитофонов и совместимости со стандартом MPEG-1.

Основное является нормой для широковещания в режиме NTSC. Остальные два режима предназначены для HDTV. Для обеспечения высокого качества изображения MPEG-2 обычно работает со скоростью 4–8 Мбит/с.

Видео по заказу

Видео по заказу иногда сравнивают с электронным видеопрокатом. Пользователь (клиент) выбирает из большого списка доступных видеофильмов один и берет его, чтобы просмотреть дома. В случае видео по заказу выбор производится не выходя из дома, с помощью пульта дистанционного управления телевизора, а заказанный фильм начинается немедленно. В пункт проката идти не нужно. Надо ли говорить, что реализация видео по заказу несколько сложнее его описания. В данном разделе мы познакомимся с основными идеями и их реализацией.

Действительно ли видео по заказу подобно видеопрокату или оно больше напоминает выбор фильма в системе кабельного телевидения, состоящей из 500 или 5000 каналов? От ответа на этот вопрос зависят важные технические решения. В частности, пользователи видеопроката привыкли к тому, что можно остановить просмотр, совершить прогулку на кухню или в ванную комнату, а затем возобновить просмотр с того места, на котором они остановили фильм. У телезрителей же кнопки ПАУЗА нет.

Если видео по заказу собирается успешно конкурировать с пунктами проката видеокассет, возможно, нужно позволить пользователям останавливать, запускать и перематывать видеофильмы. Чтобы обеспечить пользователю такую возможность, каждому зрителю нужно передавать отдельную копию фильма.

С другой стороны, если рассматривать видео по заказу скорее как обычное телевидение с заранее составленным расписанием, тогда возможен другой подход, при котором популярный фильм передается сразу по нескольким каналам с интервалом в 10 минут. Пользователь, желающий посмотреть этот фильм, должен подождать начала фильма несколько минут. Хотя при такой реализации приостановка и возобновление просмотра невозможны, пользователь, вернувшийся к телевизору после короткого перерыва, может переключиться на один из соседних каналов и найти тот же фильм, но идущий с отставанием на 10 или 20 минут. Такую схему называют **«почти видео по заказу»**. Ее реализация обходится значительно дешевле, так как хотя для передачи одного фильма и используется полтора десятка параллельных каналов, но предполагается, что этот фильм одновременно смотрят тысячи зрителей. Различие между видео по заказу и этой схемой примерно такое же, как между личным и общественным транспортом.

Просмотр видео по заказу представляет собой всего лишь одну из большого набора новых услуг, которые станут возможными, как только сети с большой пропускной способностью получат широкое распространение. Общая модель показана на рис. 7.44. В центре системы мы видим глобальную сетевую магистраль (национальную или интернациональную) с высокой пропускной способностью. С ней соединены тысячи локальных распределительных сетей, таких как сети кабельного телевидения или телефонные сети. Локальные распределительные сети доходят до домов пользователей, в которых они заканчиваются телевизион-

ными приставками (английское название: set-top box — коробочка, которую кладут на телевизор), являющимися, по сути, мощными специализированными персональными компьютерами.

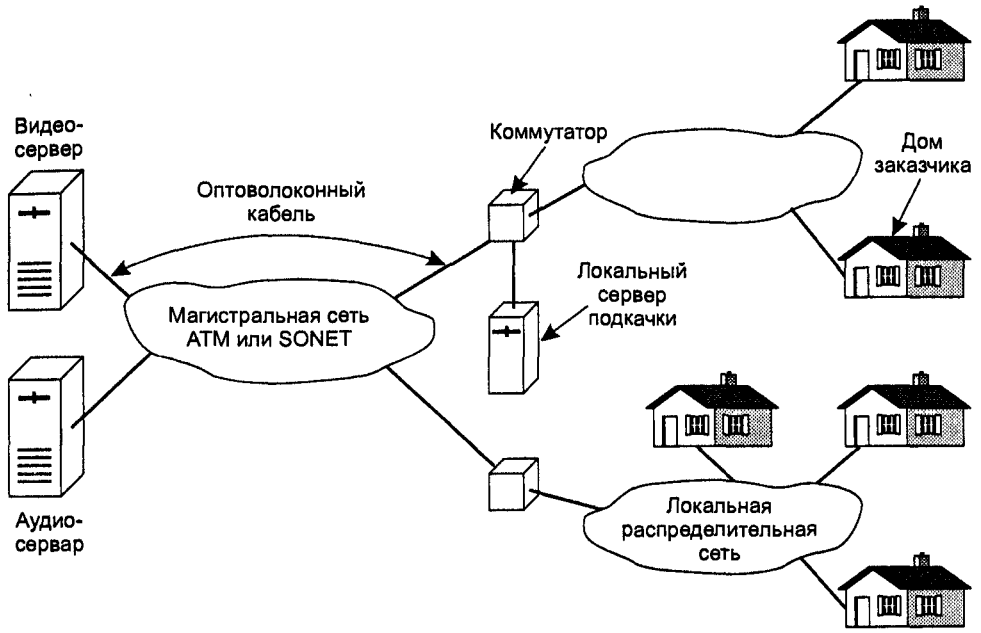


Рис. 7.44. Общая модель системы видео по заказу

С магистралью с помощью высокоскоростных оптоволоконных кабелей соединены тысячи поставщиков информации. Некоторые из них будут предоставлять видеофильмы и аудиокомпакт-диски за плату, другие будут специализироваться на таких услугах, как интернет-магазины (с возможностью прокрутить тарелку с супом и изменить масштаб списка ингредиентов или с демонстрацией видеоролика, объясняющего, как пользоваться газонокосилкой с бензиновым мотором). Без сомнения, быстро станут доступными бесчисленные возможности, такие как спорт, новости, сериалы, доступ к Всемирной паутине и т. д.

В систему также входят локальные серверы подкачки, позволяющие снижать нагрузку на главные магистрали в часы пик. Вопросы стыковки отдельных узлов этой системы и вопросы собственности на них в настоящее время являются предметами жарких споров. Далее мы рассмотрим устройство основных частей системы: видеосерверов и распределительных сетей.

Видеосерверы

Для предоставления видео по заказу необходимы специализированные **видеосерверы**, способные хранить и передавать одновременно большое количество фильмов. Общее число когда-либо снятых фильмов было оценено в 65 000 (Minoli, 1995). В сжатом с помощью алгоритма MPEG-2 виде обычный фильм занимает

около 4 Гбайт, таким образом, для хранения 65 000 фильмов потребуется около 260 Тбайт. Добавьте к этому все когда-либо сделанные старые телевизионные программы, спортивные передачи, новости, рекламу и т. д., и станет ясно, что мы имеем дело с проблемой хранения данных в промышленных масштабах.

Дешевле всего хранить большие объемы информации на магнитной ленте. Так было раньше, возможно, так будет и в дальнейшем. На кассету типа Ultrium можно записать 200 Гбайт (50 фильмов) по цене около $1–2 за фильм. Уже сейчас можно приобрести большие механические кассетные серверы, хранящие тысячи кассет, снабженные автоматическими манипуляторами, способными менять кассеты в магнитофоне. Основными проблемами таких систем остаются время доступа (особенно к 50-му фильму на кассете), скорость передачи и ограниченное количество магнитофонов (для одновременного показа n фильмов нужно n магнитофонов).

К счастью, опыт работы с пунктами видеопроката, библиотеками и другими подобными организациями показывает, что не все фильмы (книги) одинаково популярны. Экспериментально доказано, что если в пункте проката есть N фильмов, то доля заявок на конкретный фильм, стоящий на k-м месте в списке популярности, примерно равна C/k. Здесь C — это число, дополняющее сумму долей до 1, а именно:

$$C = 1/(1 + 1/2 + 1/3 + 1/4 + 1/5 + ... + 1/N).$$

Таким образом, например, самый популярный фильм берут примерно в семь раз чаще, чем седьмой в списке популярности. Такая зависимость называется законом Ципфа (Zipf, 1949).

Тот факт, что одни фильмы значительно популярнее других, наводит на идею иерархической организации хранилища фильмов, показанной на рис. 7.45. В изображенной пирамиде производительность возрастает при движении вверх.



Рис. 7.45. Иерархия хранилища видеосервера

Фильмы можно хранить и на компакт-дисках. Современные диски DVD позволяют хранить 4,7 Гбайт данных, чего вполне достаточно для записи одного фильма, однако следующее поколение дисков будет иметь примерно удвоенную емкость. Хотя по сравнению с жесткими магнитными дисками время поиска на оптическом диске на порядок выше (50 мс вместо 5 мс), их низкая цена и высокая надежность делает автомат с тысячами сменных DVD хорошей альтернативой магнитной ленте для более часто используемых фильмов.

На следующем уровне в этой иерархии находятся жесткие магнитные диски. Они обладают малым временем доступа (5 мс), высокой скоростью передачи

(320 Мбайт/с для SCSI 320) и значительной емкостью (свыше 100 Гбайт), что делает их вполне подходящими для хранения фильмов, которые действительно смотрят зрители (в отличие от тех, что хранятся на тот случай, если кто-либо захочет их посмотреть). Основной недостаток накопителей на жестких дисках заключается в их слишком высокой цене, чтобы хранить на них редко просматриваемые фильмы.

На вершине пирамиды, показанной на рис. 7.45, находится ОЗУ. Оперативная память является самым быстрым запоминающим устройством, но в то же время и самым дорогим. Оперативная память лучше всего подходит для хранения фильмов, посылаемых в данный момент различным получателям одновременно (например, настоящее видео по заказу 100 пользователям, начавшим просмотр в различное время). Когда цены на микросхемы памяти упадут до 50 долларов за гигабайт, 4 Гбайт ОЗУ, необходимые для хранения одного фильма, будут стоить 200 долларов, а 100 фильмов, находящихся в памяти, потребуют ОЗУ на 20 000 долларов (400 Гбайт). То есть видеосервер, имеющий в наличии 100 фильмов, скоро сможет позволить себе хранить их все в ОЗУ. А если у видеосервера всего 100 клиентов, которые день за днем смотрят одни и те же 20 фильмов, это уже не только реальное, но и выгодное решение.

Поскольку видеосервер представляет собой в действительности просто огромное устройство ввода-вывода, работающее в режиме реального времени, его аппаратная и программная архитектура должна отличаться от используемой в PC или рабочей станции UNIX. Аппаратная архитектура типичного видеосервера показана на рис. 7.46. Сервер состоит из одного или нескольких высокопроизводительных процессоров, у каждого из которых есть своя локальная память, память общего доступа, большой кэш ОЗУ для хранения популярных фильмов, множества различных накопителей для хранения фильмов, а также сетевое оборудование, например, оптического интерфейса с магистралью SONET или ATM с каналом OC-12 или выше. Все эти подсистемы соединены сверхскоростной шиной (по меньшей мере 1 Гбайт/с).

Познакомимся теперь с программным обеспечением видеосервера. Процессоры принимают запросы пользователей, находят нужные фильмы, перемещают данные между устройствами, выписывают счета клиентам, а также выполняют множество других операций. Для некоторых из этих функций фактор времени не является критичным, но для большинства он критичен, поэтому на некоторых (если не на всех) центральных процессорах должна работать операционная система реального времени, такая как микроядро реального времени. Такие системы обычно разбивают задание на отдельные небольшие задачи, для каждой из которых известен срок ее окончания. Затем программа, следящая за расписанием выполнения задач, может запустить, например, алгоритм выполнения задач в порядке сроков выполнения или алгоритм монотонной скорости (Liu и Layland, 1973).

Программное обеспечение, работающее в центральных процессорах, также определяет тип интерфейса, предоставляемого сервером своим клиентам (серверам подкачки и телевизионным приставкам). Наиболее популярными являются два интерфейса. Один из них представляет собой традиционную файловую систему, в которой клиент может открывать, читать, писать и закрывать файлы. Помимо

сложностей, связанных с иерархией хранения и с соображениями реального времени, в таком интерфейсе клиенту придется еще заниматься и вопросами файловой системы, например, UNIX.

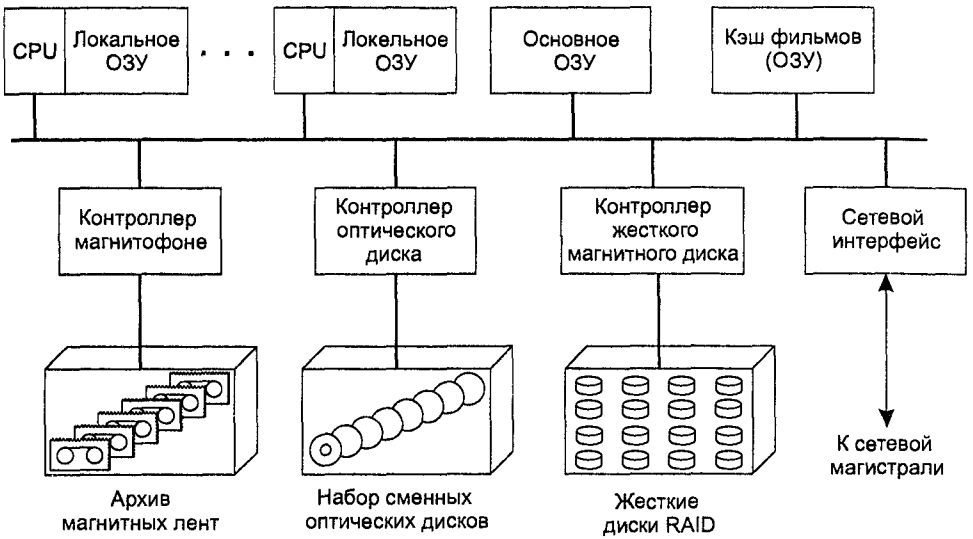


Рис. 7.46. Аппаратная архитектура типичного видеосервера

В основе второго типа интерфейса лежит модель видеомагнитофона. Сервер получает команды открытия, начала или приостановки воспроизведения, а также команды быстрой перемотки в обе стороны. Основное отличие этого интерфейса от модели UNIX состоит в том, что, получив команду PLAY (воспроизведение), сервер начинает выдавать данные с постоянной скоростью без необходимости передачи новых команд.

Центральной частью программного обеспечения видеосервера является программа управления дисками. Она выполняет две основные задачи: копирование фильмов на жесткий диск с оптического диска и обработка дисковых запросов для нескольких выходных потоков. Вопрос размещения фильма очень важен, так как он сильно влияет на производительность.

Возможны два способа организации дискового хранения: дисковая ферма и дисковый массив. **Дисковая ферма** подразумевает хранение на каждом диске нескольких фильмов целиком. Каждый фильм должен дублироваться на двух или даже более дисках, чтобы обеспечить высокую надежность и производительность. **Дисковый массив**, или набор **RAID** (Redundant Array of Independent Disks — матрица независимых дисковых накопителей с избыточностью), является формой организации хранения, в которой фильм хранится сразу на нескольких дисках, например, блок 0 на диске 0, блок 1 на диске 1 и так далее циклически. Такая организация хранения называется **разбиением на полосы**.

Разбитый на полосы массив диска обладает рядом преимуществ перед дисковой фермой. Во-первых, все n дисков могут работать параллельно, увеличивая

производительность системы в n раз. Во-вторых, такой набор можно сделать отказоустойчивым, если к каждой группе дисков добавить еще один диск, на котором хранить поблочную сумму по модулю два всех данных набора. Наконец, такой способ автоматически решает вопрос балансировки нагрузки. (Все популярные фильмы не смогут оказаться на одном диске). С другой стороны, организация дисковых массивов является более сложной, чем дисковая ферма, и чрезвычайно чувствительной к множественным отказам. Она также плохо подходит для таких операций видеомагнитофона как быстрая перемотка назад или вперед.

Другая задача дискового программного обеспечения состоит в обслуживании выходных потоков реального времени и удовлетворении соответствующих временных требований. Еще несколько лет назад это было связано с необходимостью реализации сложных алгоритмов планирования, однако по мере снижения цен на модули памяти стал возможен более простой подход. Для передачи каждого видеопотока необходимо хранить в буфере ОЗУ отрезок фильма длиной, скажем, 10 с (5 Мбайт). Буфер заполняется путем выполнения дисковой операции и освобождается сетевым процессом. Имея 500 Мбайт оперативной памяти, мы сможем одновременно буферизировать 100 потоков. Конечно, дисковая подсистема должна обладать устойчивой скоростью передачи данных 50 Мбайт/с для нормального заполнения буферов, однако матрица RAID, составленная из высокопроизводительных дисков типа SCSI, запросто может справиться с этой задачей.

Распределительная сеть

Распределительная сеть представляет собой набор коммутаторов и линий связи между источником данных и их получателями. Как было показано на рис. 7.44, она состоит из магистрали, соединенной с локальными распределительными сетями. Обычно магистраль является коммутируемой, а локальная распределительная сеть — нет.

Основным требованием, предъявляемым к магистрали, является высокая пропускная способность. Раньше вторым требованием был низкий уровень флуктуации (джиттера), то есть неравномерности трафика, однако теперь, когда даже самые дешевые ПК получили возможность буферизировать 10 с высококачественного видео в формате MPEG-2, это требование перестало быть актуальным.

Локальное распределение представляет собой чрезвычайно неупорядоченное, хаотическое явление — это связано с тем, что различные компании пытаются предоставлять услуги в различных регионах по различным сетям. Телефонные компании, компании кабельного телевидения, а также совершенно новые организации пытаются первыми застолбить участок. В результате наблюдается быстрое распространение новых технологий. В Японии некоторые компании, занимающиеся канализационными трубами, утверждают, что именно они обладают преимуществом, так как принадлежащие им трубы большого диаметра подведены в каждый дом (через них действительно прокладывается оптоволоконный кабель, однако нужно очень аккуратно следить за местами его вывода из труб). В настоящее время уже используется множество схем локального распределения видео по заказу, из которых четырьмя основными являются ADSL, FTTC, FTTH и HFC. Мы рассмотрим их все по очереди.

Система ADSL (Asymmetric Digital Subscriber Line — асимметричная цифровая абонентская линия) была первой попыткой телефонных компаний поучаствовать в деле локального распределения. Мы изучали ADSL в главе 2 и сейчас не будем повторять этот материал. Идея состояла в том, что практически каждый дом в США, Европе и Японии уже оснащен медными витыми парами (для аналоговой телефонной связи). Если бы эти провода можно было использовать для видео по заказу, то телефонные компании могли бы сорвать большой куш.

Проблема заключалась в том, что по этим проводам невозможно передать на их обычную длину в 10 км даже поток данных MPEG-1, не говоря уже о MPEG-2. Для передачи полноцветного видео с плавными движениями и высоким разрешением необходима скорость 4–8 Мбит/с в зависимости от требуемого качества. Таких скоростей с помощью ADSL достичь невозможно (вернее, возможно, но только при передаче на очень малые расстояния).

Второй вариант распределительной сети был также разработан телефонными компаниями. Он получил название **FTTC** (Fiber To The Curb — оптоволоконный кабель к распределительной коробке). В данной модели телефонная компания прокладывает оптический кабель от каждой конечной телефонной станции до устройства, расположенного по соседству с абонентами и называющегося блоком **ONU** (Optical Network Unit — блок оптической сети). К блоку ONU может быть подключено около 16 медных витых пар. Поскольку эти витые пары короткие, по ним можно передавать дуплексные потоки T1 или T2 и, соответственно, фильмы в форматах MPEG-1 или MPEG-2. Более того, благодаря симметричности схемы, FTTC может поддерживать видеоконференции.

Третье решение, предложенное телефонными компаниями, состоит в прокладке оптоволоконного кабеля в каждый дом. Эта схема называется **FTTH** (Fiber To The Home — оптоволоконный кабель к дому). При этом каждый абонент получит возможность пользоваться каналом связи OC-1, OC-3 или даже выше. Такое решение очень дорого и вряд ли получит широкое распространение в ближайшие годы, но очевидно, что возможности этого варианта практически безграничны. На рис. 7.31 мы видели схему, используя которую, каждый желающий может содержать собственную радиостанцию. А как вам идея о том, чтобы каждый член семьи был главой собственной телестудии? Все перечисленные системы— ADSL, FTTC и FTTH— представляют собой локальные распределительные сети с топологией «точка—точка», что неудивительно, учитывая то, как создавалась современная телефонная система.

Полностью противоположный подход реализуется в настоящее время поставщиками кабельного телевидения. Он называется **HFC** (Hybrid Fiber Coax — гибрид оптоволоконного и коаксиального кабелей) и показан на рис. 2.41, *а*. Суть подхода в следующем. Современные коаксиальные кабели с полосой пропускания от 300 до 450 МГц будут заменены кабелями с полосой в 750 МГц, что повысит их пропускную способность с 50–75 каналов шириной в 6 МГц до 125 таких каналов. 75 из 125 каналов по-прежнему будут использоваться для передачи аналогового телевизионного сигнала.

В 50 новых каналах будет использоваться квадратурная амплитудная модуляция QAM-256, в результате чего по каждому каналу можно будет передавать

данные со скоростью около 40 Мбит/с, что в сумме составит 2 Гбит/с. Каждым кабелем предполагается охватить около 500 домов, Таким образом, каждому дому может быть предоставлен выделенный канал с пропускной способностью 4 Мбит/с, который можно будет использовать для передачи MPEG-2.

При всех достоинствах данного подхода, для его реализации потребуется замена всех имеющихся кабелей на новые, установка новых концевых адаптеров и удаление всех односторонних усилителей — в общем, замена всей системы кабельного телевидения. Таким образом, общий объем новой инфраструктуры здесь сопоставим с тем, что требуется телефонным компаниям для организации FTTC. В обоих случаях поставщик локальных сетевых услуг должен прокладывать волоконно-оптический кабель довольно близко к домам заказчиков. Кроме того, в обоих случаях оптоволоконный кабель заканчивается оптоэлектрическим преобразователем. Разница этих двух систем в том, что в FTTC на последнем участке используется выделенная витая пара, соединяющая абонента с распределительной коробкой соединением «точка—точка», тогда как в HFC последний сегмент распределительной сети представляет собой общий коаксиальный кабель. Технически эти две системы различаются не столь сильно, как это часто утверждают их сторонники.

Тем не менее, на одно различие следует обратить внимание. Система HFC использует общий носитель без коммутации и маршрутизации. Любая информация, проходящая по кабелю, может быть прочитана любым абонентом, подключенным к этому кабелю, без каких-либо хлопот. Система FTTC, являясь полностью коммутируемой, не обладает этим свойством. В результате операторы системы HFC требуют от видеосерверов способности шифровать потоки, чтобы клиенты, не заплатившие за просмотр фильма, не могли смотреть его. Операторам системы FTTC шифрование не нужно, так как оно увеличивает сложность системы, снижает ее производительность и не повышает защищенность системы. Нужно ли шифрование с точки зрения компании-владельца видеосервера? Телефонная компания, управляющая сервером, может решить намеренно отказаться от шифрования видеофильмов, якобы по соображениям эффективности, но на самом деле, чтобы экономически подорвать своих конкурентов, предоставляющих аналогичные услуги с помощью системы HFC.

Во всех подобных распределительных сетях в каждой локальной области обслуживания обычно используется один или несколько серверов подкачки. Они представляют собой уменьшенные версии видеосерверов, о которых рассказывалось ранее. Большим достоинством этих локальных серверов является то, что они берут на себя некоторую часть магистральной нагрузки.

На локальные серверы можно заранее копировать фильмы. Если пользователи отошлют заявки на фильмы, которые они хотели бы просмотреть, заранее, такие фильмы могут копироваться на локальные серверы в периоды минимальной загрузки линий. Можно ввести гибкие тарифы на заблаговременные заказы фильмов по аналогии с бронированием авиабилетов. Так, например, можно давать 27-процентную скидку при заблаговременном заказе фильмов (от 24 до 72 часов) для просмотра во вторник или в четверг до 18:00 или после 23:00. Фильмы,

заказанные в первое воскресенье месяца до 8:00 для просмотра в среду после полудня в день, дата которого является простым числом, получают 43-процентную скидку, и т. д.

Система MBone

В то время как столько различных промышленных предприятий составляют и публикуют свои грандиозные планы на будущее, Интернет-сообщество спокойно занимается реализацией своей собственной мультимедийной системы, называющейся **MBone** (Multicast Backbone — широковещательная магистраль). В данном разделе мы познакомимся с этой системой и с принципом ее работы.

Система MBone представляет собой телевещание по Интернету. В отличие от видео по заказу, в котором упор делается на заказе и просмотре заранее сжатых и хранящихся на сервере видеофильмов, система MBone используется для широковещательного распространения по Интернету передач, выходящих в прямом эфире, в цифровой форме. Эта система существует с 1992 года. Она использовалась для организации многочисленных научных телеконференций, особенно встреч IETF (Internet Engineering Task Force — проблемная группа проектирования Интернета), а также для освещения различных значимых событий науки, например, запусков космических челноков. По системе MBone были проведены трансляции концерта Rolling Stones и некоторых эпизодов Каннского кинофестиваля. Следует ли это событие считать значимым научным событием — вопрос спорный.

Технически система MBone представляет собой виртуальную сеть поверх сети Интернет. Она состоит из соединенных туннелями островов, на которых возможна многоадресная рассылка, как показано на рис. 7.47. На рисунке сеть MBone состоит из шести островов, от *A* до *F*, соединенных семью туннелями. Каждый остров (обычно это локальная сеть или группа соединенных друг с другом локальных сетей) поддерживает многоадресную рассылку на аппаратном уровне. По туннелям, соединяющим острова, рассылаются MBone-пакеты. Когда-нибудь в будущем, когда все маршрутизаторы будут напрямую поддерживать многоадресную рассылку, такая оверлейная структура станет ненужной, но пока что именно она обеспечивает работу системы.

На каждом острове находится один или несколько специальных **многоадресных маршрутизаторов**. Некоторые из них являются обычными маршрутизаторами, но большая часть представляет собой просто рабочие станции UNIX, на которых на пользовательском уровне работает специальное программное обеспечение. Многоадресные маршрутизаторы логически соединены туннелями. MBone-пакеты инкапсулируются в IP-пакеты и рассылаются как обычные одноадресные пакеты по IP-адресу многоадресного маршрутизатора.

Туннели настраиваются вручную. Обычно туннель проходит по пути, для которого существует физическое соединение, хотя это не требуется. Если вдруг физический путь, по которому проходит туннель, прервется, многоадресные маршрутизаторы, использующие этот туннель, даже не заметят этого, так как Интернет автоматически изменит маршрут всего IP-трафика между ними.

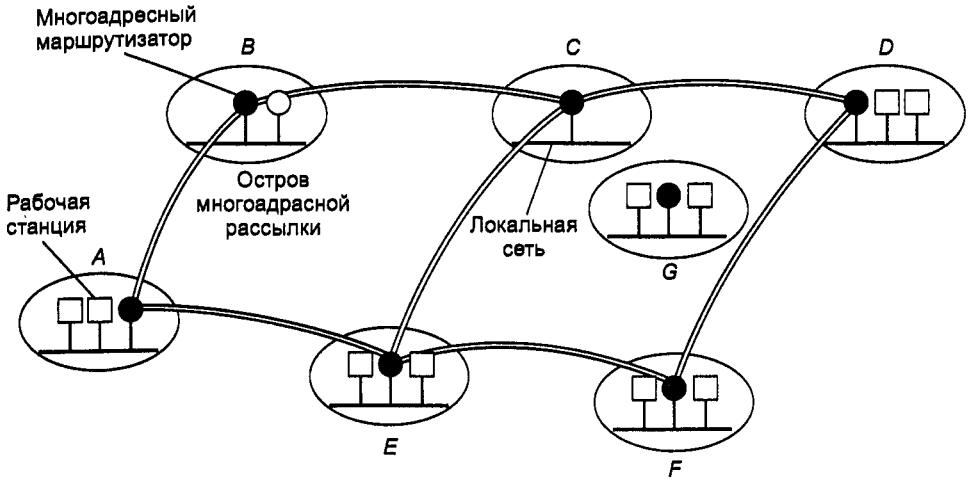


Рис. 7.47. Структура системы MBone

При появлении нового острова, желающего присоединиться к системе MBone, как, например, остров *G* на рис. 7.47, его администратор посылает по списку рассылки системы MBone сообщение, в котором объявляет о создании нового острова. После этого администраторы соседних сайтов связываются с ним для установки туннелей. Иногда при этом заново прокладываются уже существующие туннели, так как появление нового острова позволяет оптимизировать топологию системы. Физически туннели не существуют. Они определяются таблицами многоадресных маршрутизаторов и могут добавляться, удаляться или перемещаться при помощи простого изменения содержимого таблиц. В системе MBone обычно у каждой страны есть магистраль, с которой соединены региональные острова. Пара туннелей системы пересекает Атлантический и Тихий океаны, что делает ее глобальной системой.

Таким образом, в любой момент времени система MBone состоит из островов и туннелей, независимо от числа используемых в данный момент групповых адресов. Эта ситуация очень напоминает нормальную (физическую) подсеть, поэтому к ней применимы нормальные алгоритмы маршрутизации. Соответственно, в системе MBone изначально использовался алгоритм маршрутизации **DVMRP** (Distance Vector Multicast Routing Protocol — протокол дистанционной маршрутизации сообщений с использованием векторной многоканальной трансляции), основанный на дистанционно-векторном алгоритме Беллмана—Форда (Bellman—Ford). Например, остров *C* может быть связан с островом *A* либо через остров *B*, либо через остров *E* (или, возможно, через остров *D*). Алгоритм делает свой выбор на основании сообщаемых ему узлами значений своей удаленности от острова *A* и своей удаленности от других островов. Подобным образом каждый остров может определить лучший маршрут до всех остальных островов. Однако, как мы скоро увидим, эти маршруты необязательно используются именно таким образом.

Познакомимся теперь с тем, как осуществляется многоадресная рассылка. Для этого источник должен сначала получить групповой адрес класса D, действующий подобно частоте радиостанции или номеру канала. Адреса класса D зарезервированы для программ, ищущих в базе данных свободные групповые адреса. Одновременно может производиться несколько операций многоадресной рассылки, и хост может «настроиться» на интересующую его передачу, прослушивая соответствующие групповые адреса.

Периодически каждый многоадресный маршрутизатор посылает широковещательный IGMP-пакет, сфера распространения которого ограничена пределами его острова. В этом пакете содержится предложение сообщить, кого какой канал интересует. Хосты, заинтересованные в получении (или в продолжении получения) какого-либо одного или нескольких каналов, посылают в ответ свои IGMP-пакеты. Чтобы избежать перегрузки локальной сети, эти ответы посылаются не сразу, а располагаются во времени особым образом. Чтобы не тратить понапрасну пропускную способность локальных сетей, каждый многоадресный маршрутизатор хранит таблицу каналов, которые он должен направлять в свою локальную сеть.

Когда источник аудио- или видеопотока создает новый пакет, он распространяет его с помощью аппаратно реализованной многоадресной рассылки внутри своего острова. Затем этот пакет подбирается локальным многоадресным маршрутизатором, который копирует его по всем туннелям, с которыми он соединен.

Получив такой пакет по туннелю, каждый многоадресный маршрутизатор проверяет, прибыл ли этот пакет по оптимальному маршруту, то есть по маршруту, который, как указано в таблице маршрутизатора, следует использовать для данного источника (как если бы это был пункт назначения). Если пакет прибыл по наилучшему маршруту, он копируется маршрутизатором во все его туннели. В противном случае пакет игнорируется. Если, например, в таблицах маршрутизатора *C* (рис. 7.47) говорится, что маршрут к острову *A* должен пролегать через остров *B*, — тогда если многоадресный пакет от острова *A* прибудет на остров *C* через остров *B*, то он будет скопирован на острова *D* и *E*. Если же такой пакет от острова *A* прибудет на остров *C* через остров *E*, то он будет проигнорирован. Этот алгоритм представляет собой просто алгоритм пересылки в обратном направлении, рассматривавшийся в главе 5. Хотя он и не совершенен, он достаточно хорош и прост в реализации.

Помимо использования алгоритма пересылки в обратном направлении, позволяющего избежать перегрузки Интернета, для ограничения сферы распространения многоадресных пакетов также используется IP-поле *Time to live* (время жизни). Каждый пакет начинает свой путь с определенным значением этого поля (определяемым источником). Каждому туннелю присваивается весовой коэффициент. Пакет пропускается сквозь туннель, только если он обладает достаточным весом. В противном случае пакет отвергается. Например, туннелям, пересекающим океаны, обычно назначаются весовой коэффициент, равный 128, поэтому область распространения всех пакетов, время жизни которых не превы-

шает 127, ограничивается одним континентом. После прохождения пакета по туннелю значение его поля *Time to live* уменьшается на вес туннеля.

Много внимания было уделено улучшению алгоритма маршрутизации. Одно из предложений основывалось на идее дистанционно-векторной маршрутизации. В этом случае MBone-сайты группируются в области, а алгоритм становится иерархическим (Thyagarajan и Deering, 1995).

Другое предложение состоит в использовании вместо дистанционно-векторной маршрутизации модифицированной формы маршрутизации с учетом состояния линий. В частности рабочая группа Интернета IETF занимается изменением алгоритма маршрутизации OSPF (Open Shortest Path First — первоочередное открытие кратчайших маршрутов), чтобы сделать его более подходящим для многоадресной рассылки в пределах автономной системы. Новый алгоритм получил название **MOSPF** (многоадресный алгоритм OSPF) (Moy, 1994). В дополнение к обычной информации, используемой для выбора маршрута, в этом алгоритме также строится полная карта всех островов многоадресной рассылки и всех туннелей. Зная полную топологию системы, несложно рассчитать лучший маршрут между любыми двумя островами по имеющимся туннелям. Например, можно воспользоваться алгоритмом Дейкстры (Dijkstra).

Вторая область исследований охватывает маршрутизацию внутри автономной системы. Этот алгоритм называется **PIM** (Protocol Independent Multicast — не зависящая от протокола многоадресная рассылка). Он разрабатывается другой рабочей группой IETF. Создано две версии алгоритма PIM, применяющихся в зависимости от плотности островов (почти все желают смотреть видео или, наоборот, почти никто ничего не хочет смотреть). Вместо того чтобы создавать оверлейную топологию, как это делается в алгоритмах DVMPR и MOSPF, в обеих версиях алгоритма PIM используются стандартные таблицы одноадресной маршрутизации.

В плотном варианте алгоритма PIM (PIM-DM) идея состоит в отсечении бесполезных путей. Когда многоадресный пакет прибывает по «неправильному» туннелю, обратно посылается специальный отсекающий пакет, предлагающий отправителю прекратить отправлять по этому туннелю пакеты данному адресату. Если же пакет прибывает по «правильному» туннелю, он копируется во все остальные еще не отсеченные туннели. Если все остальные туннели маршрутизатора отсечены, а в его области этот канал никто смотреть не желает, маршрутизатор сам посылает отсекающий пакет по «правильному» туннелю. Таким образом, многоадресная рассылка автоматически адаптируется к спросу на видеоданные.

Вариант алгоритма PIM для редко расположенных островов (PIM-SM), описанный в RFC 2363, действует по-другому. Идея этого варианта заключается в том, чтобы не забивать Интернет излишней многоадресной рассылкой из-за, скажем, трех человек из университета Беркли, желающих устроить небольшую видеоконференцию с помощью IP-адресов класса D. В этом случае алгоритм PIM создает так называемые точки встречи. Каждый источник посылает в эти точки свои пакеты. Любой сайт, желающий присоединиться к видеоконференции, просит установить туннель с точкой встречи. Таким образом, в этом варианте алгоритма PIM трафик переносится при помощи обычной одноадресной рассылки.

Популярность варианта PIM-SM возрастает, и система MBone все чаще прибегает к его использованию. Соответственно, алгоритм MOSPF встречается все реже и реже. С другой стороны, сама система MBone переживает период стагнации и, возможно, никогда не достигнет процветания.

Несмотря на этот грустный вывод относительно MBone, мультимедиа в целом представляет собой захватывающую и быстро меняющуюся область. Ежедневно появляется информация о создании новых технологий и приложений. Многоадресная рассылка и качество обслуживания становятся все более популярны, их обсуждению посвящена книга (Striegel и Manimaran, 2002). Еще одной интересной темой является многоадресная рассылка по беспроводным сетям (Gossain и др., 2002). Вообще же область знаний, касающаяся многоадресной рассылки и всего, что с этим связано, вероятно, будет волновать умы человечества еще долгие годы.

Резюме

Именование доменов в Интернете реализуется при помощи иерархической схемы, называемой службой имен доменов (DNS). В системе DNS на верхнем уровне находятся популярные родовые домены, включая *com*, *edu* и около двухсот национальных доменов. DNS реализована в виде распределенной базы данных, серверы которой расположены по всему миру. В ней хранятся записи с IP-адресами, адресами почтовых обменников и прочей информацией. Обратившись к DNS-серверу, процесс может преобразовать имя домена Интернета в IP-адрес, требующийся для общения с доменом.

Электронная почта — это одно из самых популярных приложений Интернета. Ею пользуются все, начиная от детей младшего школьного возраста и заканчивая стариками преклонных годов. Большинство систем электронной почты соответствуют стандартам, описанным в RFC 2821 и 2822. Сообщения, пересылаемые по e-mail, содержат ASCII-заголовки, определяющие свойства самого сообщения. Можно пересылать данные разных типов, указывая эти типы в MIME-заголовках. Передача писем осуществляется по протоколу SMTP, устанавливающему TCP-соединение между хостом-источником и хостом-приемником. Почта передается напрямую по этому TCP-соединению.

Еще одним безумно популярным приложением Интернета является Всемирная паутина (WWW). Она представляет собой систему связанных между собой гипертекстовых документов. Изначально каждый документ был страницей, написанной на HTML и содержащей гиперссылки на другие страницы. Сегодня при написании страниц все чаще используется обобщенный язык XML. Кроме того, немалая часть содержимого документов генерируется динамически при помощи скриптов, работающих как на стороне сервера (PHP, JSP и ASP), так и на стороне клиента (JavaScript). Браузер выводит документы на экран, устанавливая TCP-соединение с сервером, запрашивая у него страницу и разрывая после этого соединение. В сообщениях с такими запросами содержится множество заголовков, позволяющих сообщить дополнительную информацию. Для повыше-

ния производительности Всемирной паутины применяются кэширование, репликация и сети доставки содержимого.

На горизонте Интернета начинают появляться беспроводные веб-системы. Первыми такими системами являются WAP и i-mode. Для них обеих характерны наличие маленького экрана мобильного телефона и низкая пропускная способность, однако следующее поколение этих систем, наверное, будет более мощным.

Мультимедиа — это еще одна звезда, восходящая на сетевом небосклоне. Эта область включает приложения, занимающиеся оцифровкой звука и видеоизображений и их передачей по сетям. Для передачи звука требуется относительно низкая пропускная способность, благодаря этому данный вид мультимедиа более распространен в сетях. Потоковое аудио, интернет-радио, передача речи поверх IP — все эти приложения сегодня реально работают. Кроме того, постоянно появляются новые приложения. Видео по заказу — это перспективная область, к которой сейчас проявляется большой интерес. Наконец, MBone представляет собой экспериментальную систему всемирного цифрового телевещания в Интернете.

Вопросы

1. Многие коммерческие компьютеры имеют три разных и в то же время абсолютно уникальных идентификатора. Как они выглядят?
2. Основываясь на информации, приведенной в листинге 7.1, определите, к какому классу сети принадлежит хост *little-sister.cs.vu.nl*, A, B или C?
3. В листинге 7.1 после слова *rowboat* не поставлена точка. Почему?
4. Попробуйте угадать, что означает смайлик `:-X` (иногда изображаемый как `:-#`).
5. DNS использует UDP вместо TCP. Если DNS-пакет теряется, он автоматически не восстанавливается. Приводит ли это к возникновению проблем, и если да, то как они решаются?
6. Кроме того, что UDP теряет пакеты, на них еще и накладывается ограничение по длине, причем оно может быть довольно строгим: 576 байт. Что произойдет, если длина искомого имени DNS превысит это число? Можно ли будет послать его в двух пакетах?
7. Может ли компьютер иметь одно имя DNS и несколько IP-адресов? Как такое может быть?
8. Может ли компьютер иметь два имени DNS в разных доменах верхнего уровня? Если да, приведите правдоподобный пример. Если нет, объясните, почему это невозможно.
9. Число компаний, имеющих собственный веб-сайт, в последнее время сильно возросло. В результате в домене *com* существуют сайты тысяч фирм, что приводит к сильной нагрузке на сервер, обслуживающий этот домен верхнего уровня. Предложите способ решения этой проблемы без изменения схемы именования (то есть без изобретения нового домена верхнего уровня). Возможно, ваше решение потребует внесения изменений в клиентские программы.

10. Некоторые системы электронной почты поддерживают поле *Content-Return:*. В нем указывается, нужно ли возвращать содержимое письма в том случае, если оно не будет доставлено получателю. Это поле входит в состав конверта или заголовка письма?
11. Системы электронной почты хранят адресные книги e-mail, с помощью которых пользователь может найти нужный адрес. Для поиска по таким книгам имена адресатов должны быть разбиты на стандартные компоненты (например, имя, фамилия). Обсудите некоторые проблемы, которые следует решить, чтобы можно было разработать соответствующий международный стандарт.
12. Адрес e-mail состоит из имени пользователя, знака @ и имени домена DNS с записью MX. В качестве имени пользователя может указываться реальное имя человека, фамилия, инициалы или любые другие идентификаторы. Допустим, причиной потери многих писем, приходящих в адрес большой компании, является то, что авторы писем не знают точные имена пользователей. Существует ли возможность решения этой проблемы без изменения DNS? Если да, предложите свой вариант и объясните принцип его работы. Если нет, объясните, почему.
13. Имеется двоичный файл длиной 3072 байта. Каков будет его размер после кодирования с помощью системы base64? Пара символов CR+LF вставляется через каждые 80 байт, а также в конце сообщения.
14. Рассмотрите схему кодирования MIME quoted-printable. Укажите проблемы, которые мы не затронули в тексте, и предложите способ их решения.
15. Назовите пять типов MIME, не указанных в тексте. Информацию можно взять из настроек браузера или из Интернета.
16. Предположим, вы хотите переслать другу MP3-файл, однако провайдер, услугами которого пользуется ваш друг, ограничивает максимальный размер входящей почты до 1 Мбайт, а файл занимает 4 Мбайт. Можно ли решить поставленную задачу, используя RFC 822 и MIME?
17. Предположим, некто устанавливает каникулярного демона, после чего посылает сообщение и сразу же выходит из системы. К сожалению, получатель сообщения уже с неделю находится в отпуске и на его машине также установлен каникулярный демон. Что произойдет? Будут ли каникулярные демоны переписываться без конца, пока кто-нибудь не вернется из отпуска и не прервет их диалог?
18. В любом стандарте, таком как RFC 822, должно быть описание точной грамматики — это требуется для межсетевого взаимодействия. Даже самые простые элементы должны быть четко определены. Например, в заголовках SMTP допустимы пробелы между символами. Приведите *два* правдоподобных альтернативных определения этих пробелов.
19. Каникулярный демон является частью пользовательского агента или агента передачи сообщений? Понятно, что он настраивается с помощью пользовательского агента, но какая часть системы занимается реальной отправкой автоматических ответов? Поясните свой ответ.

20. Протокол POP3 позволяет пользователям запрашивать и загружать почту из удаленного почтового ящика. Означает ли это, что внутренний формат почтовых ящиков должен быть стандартизован, чтобы любые клиентские программы, использующие POP3, могли обратиться к почтовому ящику на любом сервере? Аргументируйте свой ответ.
21. С точки зрения провайдера POP3 и IMAP отличаются друг от друга довольно сильно. Пользователи POP3 обычно опустошают почтовые ящики ежедневно. Пользователи IMAP хранят свою почту на сервере неопределенно долго. Представьте, что провайдер хочет посоветоваться с вами, решая, какие протоколы ему поддерживать. Какие соображения вы выскажете в ответ?
22. Какие протоколы использует Webmail: POP3, IMAP или ни тот, ни другой? Если какой-то из этих двух, то почему выбран именно он? Если ни тот, ни другой, то к какому из них ближе по духу реально используемый протокол?
23. При пересылке веб-страницы предваряются заголовками MIME. Зачем?
24. Когда бывают нужны внешние программы просмотра? Как браузер узнает, какую из этих программ использовать?
25. Возможна ли ситуация, при которой щелчок пользователя на одной и той же ссылке с одним и тем же MIME-типом в Internet Explorer и в Netscape приводит к запуску совершенно разных вспомогательных приложений? Ответ поясните.
26. Многопотоковый веб-сервер организован так, как показано на рис. 7.9. На прием запроса и поиск в кэше уходит 500 мкс. В половине случаев файл обнаруживается в кэше и немедленно возвращается. В другой половине случаев модуль блокируется на 9 мс, в течение которых ставится в очередь и обрабатывается дисковый запрос. Сколько модулей должен поддерживать сервер, чтобы процессор постоянно находился в работе (предполагается, что диск не является узким местом системы)?
27. Стандартный URL со схемой *http* подразумевает, что веб-сервер прослушивает порт 80. Тем не менее, веб-сервер может прослушивать и другой порт. Предложите синтаксис URL, который позволил бы обращаться к серверу, прослушивающему нестандартный порт.
28. Хотя об этом и не было сказано в тексте, существует альтернативный вариант записи URL, использующий вместо имени DNS IP-адрес. Пример такого URL может выглядеть так: http://192.31.231.66/index.html. Как браузер узнает, что следует вслед за схемой: имя DNS или IP-адрес?
29. Представьте, что сотрудник факультета компьютерных наук Стэнфордского университета написал новую программу, которую он хочет распространять по FTP. Он помещает программу newprog.c в каталог ftp/pub/freebies. Как будет выглядеть URL этой программы?
30. В соответствии с табл. 7.9 www.aportal.com хранит список предпочтений клиента в виде cookie. Недостаток такого решения: размер cookie ограничен 4 Кбайт, и если нужно сохранить много данных о пользователе (например, о том, что он хочет видеть на странице множество биржевых сводок, новостей спортив-

ных команд, типов новых историй, погоду сразу во многих городах, специальные предложения по разным категориям товаров и т. д.), то вскоре может быть достигнут 4-килобайтный порог этих описаний. Предложите альтернативный способ хранения данных о пользователе, в котором эта проблема не возникала бы.

31. Некий «Банк для лентяев» хочет организовать специальную онлайновую банковскую систему для своих ленивых клиентов. После регистрации в системе и идентификации с помощью пароля пользователь получает cookie-файл, содержащий идентификационный номер клиента. Таким образом, ему не приходится всякий раз при входе в систему повторять ввод своих идентификационных данных. Как вам такая идея? Будет ли она работать? Насколько вообще хороша такая идея?
32. В листинге на рис 7.12 в теге `<IMG>` устанавливается значение параметра *ALT*. При каких условиях браузер использует его и как?
33. Как в языке HTML с изображением ассоциируется гиперссылка? Покажите на примере.
34. Напишите тег `<A>`, необходимый для того, чтобы связать строку «ACM» с гиперссылкой на адрес http://www/acm.org.
35. Создайте форму на языке HTML для новой компании Interburger, принимающей заказы на гамбургеры по Интернету. Бланк заказа должен содержать имя заказчика, его адрес, размер гамбургера (гигантский или огромный) и наличие сыра. Оплата гамбургеров производится наличными после доставки, поэтому информация о кредитной карте не требуется.
36. Создайте форму, предлагающую пользователю ввести два числа. После нажатия кнопки «Подтверждение» сервер должен вернуть сумму введенных чисел. Напишите PHP-скрипт для серверной части.
37. Для каждого из перечисленных случаев укажите: 1) возможно ли и 2) лучше ли использовать PHP-скрипт или JavaScript и почему:

    1) календарь на любой месяц, начиная с сентября 1752 года;

    2) расписание рейсов из Амстердама в Нью-Йорк;

    3) вывод полинома с коэффициентами, введенными пользователем.
38. Напишите программу на JavaScript, принимающую на входе целочисленные значения, превышающие 2, и сообщающую, является ли введенное число простым. В JavaScript синтаксис выражений `if` и `while` совпадает с аналогичными выражениями в C и Java. Оператор выполнения арифметических действий по произвольному модулю: `%`. Если понадобится найти квадратный корень числа x, воспользуйтесь функцией `Math.sqrt(x)`.
39. HTML-страница состоит из следующего кода:

```
<html><body>
<a href="www.info-source.com/welcome.html">Информация</a>
</body></html>
```

    Когда пользователь щелкает на гиперссылке, открывается TCP-соединение и на сервер отправляется некоторая последовательность строк. Перечислите эти строки.

40. Заголовок *If-Modified-Since* может использоваться для проверки актуальности страницы, хранящейся в кэше. Соответствующие запросы могут посылаться на страницы, содержащие изображения, звуки, видео и т. д., а также на обычные страницы на HTML. Как вы думаете, эффективность этого метода будет выше для изображений JPEG или для страниц HTML? Хорошенько подумайте над значением слова «эффективность» и после этого объясните свой ответ.
41. В день очень важного спортивного события (скажем, финала международного чемпионата по популярному виду спорта) огромное количество посетителей стремятся попасть на официальный веб-сайт мероприятия. Схожа ли эта ситуация внезапного роста трафика с выборами во Флориде в 2000 году и почему?
42. Имеет ли смысл отдельному провайдеру функционировать в качестве сети доставки содержимого? Если да, то как должна работать система? Если нет, то чем плоха такая идея?
43. При каких обстоятельствах мысль об использовании сети доставки содержимого является неудачной?
44. Беспроводные веб-терминалы обладают низкой пропускной способностью, что приводит к необходимости эффективного кодирования. Предложите схему эффективной передачи текста на английском языке по беспроводным линиям связи на WAP-устройство. Можно предположить, что терминал обладает постоянной памятью объемом несколько мегабайт и процессором средней мощности. *Подсказка:* вспомните, как передается текст на японском языке, где каждый символ представляет собой слово.
45. Компакт-диск вмещает 650 Мбайт данных. Используется ли сжатие для аудиокомпакт-дисков? Аргументируйте свой ответ.
46. На рис. 7.26, *в* показано, что шум квантования возникает из-за использования 4-битных отсчетов для представления 9 уровней сигнала. Первый отсчет (в нуле) является точным, а остальные — нет. Чему равна относительная погрешность для отсчетов, взятых в моменты времени, равные 1/32, 2/32 и 3/32 периода?
47. Можно ли использовать психоакустическую модель для уменьшения требуемой пропускной способности для систем интернет-телефонии? Если да, то каковы условия, при которых эта модель будет работать (если они вообще есть)? Если нет, то почему?
48. Сервер аудиопотока расположен на удалении от проигрывателя, дающем задержку 50 мс в одном направлении. Он выдает данные со скоростью 1 Мбит/с. Если проигрыватель содержит буфер объемом 1 Мбайт, то что можно сказать о расположении нижнего и верхнего пределов заполнения этого буфера?
49. Алгоритм чередования, показанный на рис. 7.29, хорош тем, что потеря пакета не приводит к возникновению паузы в звучании. Тем не менее, при использовании этого алгоритма в интернет-телефонии выявляется некий недостаток. Какой?

50. Возникают ли при передаче речи поверх IP те же проблемы с брандмауэрами, что и при передаче потокового аудио? Ответ обсудите.
51. Какая скорость требуется для передачи несжатого цветного изображения с размером 800×600 с 8 битами на пиксел при 40 кадрах в секунду?
52. Может ли ошибка в одном бите в кадре MPEG повредить более одного кадра? Аргументируйте свой ответ.
53. Рассмотрим пример видеосервера, обслуживающего 100 000 клиентов. Каждый клиент смотрит два фильма в месяц. Предположим, что половину всех фильмов начинают просматривать в 20:00. Сколько фильмов одновременно должен передавать сервер в этот период? Если для передачи каждого фильма требуется 4 Мбит/с, сколько соединений типа OC-12 нужно для соединения сервера с сетью?
54. Предположим, что закон Ципфа соблюдается для доступа к видеосерверу, на котором хранится 10 000 фильмов. Допустим, сервер хранит 1000 самых популярных фильмов на магнитных дисках, а остальные 9000 фильмов — на оптических дисках. Какая часть запросов будет адресована магнитным дискам? Напишите небольшую программу, вычисляющую данное значение численно.
55. Некие нехорошие люди зарегистрировали имена доменов, которые незначительно отличаются от всемирно известных, таких как www.microsoft.com и которые пользователь может посетить, просто случайно опечатавшись при наборе адреса. Приведите пример по крайней мере пяти таких доменов.
56. Существует множество сайтов, зарегистрированных под именами www.слово.com, где *слово* — это обычное слово английского языка. Для каждой из приведенных категорий составьте список из пяти веб-сайтов и вкратце опишите их суть (например, www.cosmos.com — это сайт, посвященный проблемам космоса). Вот список категорий: животные, продукты питания, предметы быта, части тела. Что касается последней категории, просьба указывать объекты, расположенные выше талии.
57. Разработайте несколько собственных значков emoji, используя битовую карту размером 12×12. Попытайтесь изобразить «ее парня», «его подружку», профессора и политика.
58. Напишите POP3-сервер, работающий со следующими командами: USER, PASS, LIST, RETR, DELE и QUIT.
59. Перепишите листинг 6.1, превратив его в реальный веб-сервер, использующий команду GET для работы с протоколом HTTP 1.1. Он должен реагировать на сообщение *Host*. Сервер должен кэшировать файлы, которые были недавно запрошены с диска, и обслуживать запросы, по возможности выдавая файлы из кэша.

Глава 8
Безопасность в сетях

- Криптография
- Алгоритмы с симметричным криптографическим ключом
- Алгоритмы с открытым ключом
- Цифровые подписи
- Управление открытыми ключами
- Защита соединений
- Протоколы аутентификации
- Конфиденциальность электронной переписки
- Защита информации во Всемирной паутине
- Социальный аспект
- Резюме
- Вопросы

В первые десятилетия своего существования компьютерные сети использовались, в первую очередь, университетскими исследователями для обмена электронной почтой и сотрудниками корпораций для совместного использования принтеров. В таких условиях вопросы безопасности не привлекали большого внимания. Однако теперь, когда миллионы обычных граждан пользуются сетями для управления своими банковскими счетами, заполнения налоговых деклараций, приобретают товары в интернет-магазинах, проблема сетевой безопасности становится все более актуальной. В этой главе мы рассмотрим вопросы безопасности сетей с различных точек зрения, укажем на подводные камни и обсудим различные алгоритмы и протоколы, позволяющие повысить безопасность сетей.

Тема безопасности включает в себя множество вопросов, связанных с различными человеческими грехами. В простейшем виде службы безопасности гарантируют,

что любопытные личности не смогут читать, или, что еще хуже, изменять сообщения, предназначающиеся другим получателям. Службы безопасности пресекают попытки получения доступа к удаленным службам теми пользователями, которые не имеют на это прав. Кроме того, система безопасности позволяет определить, написано ли сообщение «Оплатите счета до пятницы» тем отправителем, чье имя в нем указано, или же это фальсификация. Кроме того, системы безопасности решают проблемы, связанные с перехватом и повторным воспроизведением сообщений и с людьми, пытающимися отрицать, что они посылали данные сообщения.

Большинство проблем безопасности возникает из-за злоумышленников, пытающихся извлечь какую-либо пользу для себя или причинить вред другим. Несколько наиболее распространенных типов нарушителей перечислено в табл. 8.1. Из этого списка должно быть ясно, что задача обеспечения безопасности сетей включает в себя значительно больше, нежели просто устранение программных ошибок. Часто стоит задача перехитрить умного, убежденного и иногда хорошо финансированного противника. Также очевидно, что меры, способные остановить случайного нарушителя, мало повлияют на серьезного преступника. Статистика, собираемая полицией, говорит о том, что большинство атак предпринимается не извне (людьми, прослушивающими линии связи), а изнутри — завистливыми или недовольными чем-либо людьми. Следовательно, системы безопасности должны учитывать и этот факт.

Таблица 8.1. Типы нарушителей и цели их действий

Нарушитель	Цель
Студент	Прочитать из любопытства чужие письма
Хакер	Проверить на прочность чужую систему безопасности; украсть данные
Торговый агент	Притвориться представителем всей Европы, а не только Андорры
Бизнесмен	Разведать стратегические маркетинговые планы конкурента
Уволенный сотрудник	Отомстить фирме за увольнение
Бухгалтер	Украсть деньги компании
Биржевой брокер	Не выполнить обещание, данное клиенту по электронной почте
Аферист	Украсть номера кредитных карт для продажи
Шпион	Узнать военные или производственные секреты противника
Террорист	Украсть секреты производства бактериологического оружия

В первом приближении проблемы безопасности сетей могут быть разделены на четыре пересекающиеся области: секретность, аутентификация, обеспечение строгого выполнения обязательств и обеспечение целостности. Секретность (конфиденциальность) означает предотвращение попадания информации в руки неавторизованных пользователей. Именно это обычно приходит в голову при упоминании безопасности сетей. Аутентификация позволяет определить, с кем вы разговариваете, прежде чем предоставить собеседнику доступ к секретной информации или вступить с ним в деловые отношения. Проблема обеспечения строгого

выполнения обязательств имеет дело с подписями. Как доказать, что ваш клиент действительно прислал электронной почтой заказ на десять миллионов винтиков с левосторонней резьбой по 89 центов за штуку, если впоследствии он утверждает, что цена была 69 центов? Наконец, как можно быть уверенным, что принятое вами сообщение не подделано и не модифицировано по пути злоумышленником?

Все эти аспекты (секретность, аутентификация, обеспечение строгого выполнения обязательств и обеспечение целостности) встречаются и в традиционных системах, но с некоторыми существенными отличиями. Секретность и целостность достигаются с помощью заказных писем и хранения документов в несгораемых сейфах. Сегодня ограбить почтовый поезд значительно сложнее, чем во времена Джесси Джеймса.

Кроме того, обычно несложно отличить на глаз оригинальный бумажный документ от фотокопии. В качестве проверки попробуйте сделать фотокопию настоящего банковского чека. В понедельник попытайтесь обналичить настоящий чек в вашем банке. А во вторник попробуйте сделать то же самое с фотокопией. Проследите за разницей в поведении банковского служащего. Увы, но оригинал и копия электронных чеков неотличимы друг от друга. Понадобится некоторое время, прежде чем банки привыкнут к этому.

Люди опознают друг друга по лицам, голосам и почеркам. Доказательства подлинности бумажных документов обеспечиваются подписями на печатных бланках, печатями, рельефными знаками и т. д. Подделка обычно может быть обнаружена специалистами по почерку, бумаге и чернилам. При работе с электронными документами все это недоступно. Очевидно, требуются другие решения.

Прежде чем перейти к обсуждению сути самих решений, имеет смысл потратить несколько минут и попытаться определить, к какому уровню стека протоколов относится система сетевой безопасности. Вероятно, какое-то одно место для нее найти сложно. Каждый уровень должен внести свой вклад. На физическом уровне с подслушиванием можно бороться за счет помещения передающих кабелей в герметичные трубы, наполненные аргоном под высоким давлением. Любая попытка просверлить трубу приведет к утечке части газа из трубы, в результате давление снизится, и это послужит сигналом тревоги. Подобная техника применяется в некоторых военных системах.

На уровне передачи данных пакеты, передаваемые по двухточечной линии, могут кодироваться при передаче в линию и декодироваться при приеме. Все детали этого могут быть известны только уровню передачи данных, причем более высокие уровни могут даже не догадываться о том, что там происходит. Однако такое решение перестает работать в том случае, если пакету нужно преодолеть несколько маршрутизаторов, поскольку при этом пакет придется расшифровывать на каждом маршрутизаторе, что сделает его беззащитным перед атаками внутри маршрутизатора. Кроме того, такой метод не позволит защищать отдельные сеансы, требующие защиты (например, осуществление покупок в интернет-магазинах), и при этом не защищать остальные. Тем не менее, этот метод, называемый **шифрованием в канале связи**, легко может быть добавлен к любой сети и часто бывает полезен.

На сетевом уровне могут быть установлены брандмауэры, позволяющие отвергать подозрительные пакеты, приходящие извне. К этому же уровню относится IP-защита.

На транспортном уровне можно зашифровать соединения целиком, от одного конца до другого. Максимальную защиту может обеспечить только такое сквозное шифрование.

Наконец, проблемы аутентификации и обеспечения строго выполнения обязательств могут решаться только на прикладном уровне.

Итак, очевидно, что безопасность в сетях — это вопрос, охватывающий все уровни протоколов, именно поэтому ему посвящена отдельная глава.

Несмотря на то, что эта глава большая, важная и содержит множество технических описаний, в настоящий момент вопрос безопасности в сетях может показаться несколько неуместным. Ведь хорошо известно, что большинство «дыр» в системах безопасности возникают из-за неумелых действий персонала, невнимательного отношения к процедурам защиты информации или недобросовестности самих сотрудников защищаемого объекта. Доля проблем, возникающих из-за преступников-интеллектуалов, прослушивающих линии связи и расшифровывающих полученные данные, сравнительно мала. Посудите сами: человек может прийти в совершенно произвольное отделение банка с найденной на улице магнитной кредитной карточкой, посетовать на то, что он забыл свой PIN-код, а бумажку, на которой он написан, съел, и ему тотчас «напомнят» секретный шифр (чего только не сделаешь ради добрых отношений с клиентами). Ни одна криптографическая система в мире здесь не поможет. В этом смысле книга Росса Андерсона (Anderson, 2001) действительно ошеломляет, так как в ней приводятся сотни примеров того, как системы безопасности в самых различных производственных областях не справлялись со своей задачей. И причинами этих неудач были, мягко говоря, неряшливость в ведении дел или простое пренебрежение элементарными правилами безопасности. Тем не менее, мы оптимистично надеемся на то, что электронная коммерция со временем станет более популярной, компании станут более тщательно производить операции, связанные с защитой информации, и техническая сторона вопроса все же будет превалировать над фактором человеческой невнимательности.

На всех уровнях, за исключением физического, защита информации базируется на принципах криптографии. Поэтому мы начнем изучение систем безопасности с детального рассмотрения основ криптографии. В разделе «Криптография» мы изучим базовые принципы. Далее до раздела «Управление открытыми ключами» будут рассмотрены некоторые классические алгоритмы и структуры данных, применяемые в криптографии. После этого мы свяжем теорию с практикой и посмотрим, как все эти концепции применяются в компьютерных сетях. В конце этой главы будут приведены некоторые мысли, касающиеся технологии и социальных вопросов.

Прежде чем приступить к изложению, позвольте назвать вопросы, о которых *не* пойдет речь в этой главе. Подбирая материал, мы старались сконцентрировать внимание на вопросах, связанных именно с компьютерными сетями, а не с опе-

рационными системами или приложениями (хотя провести четкую грань зачастую бывает довольно трудно). Например, ничего не говорится об авторизации пользователя с использованием биометрии, защите паролей, атак, связанных с переполнением буферов, вирусах типа «троянский конь», получении доступа путем обмана, логических бомбах, вирусах, червях и т. п. Обо всем этом очень много говорится в главе 9 книги «*Современные операционные системы*» (Таненбаум, 2001). Все желающие узнать о вопросах системной защиты могут обратиться к этой книге.

Итак, в путь.

Криптография

Слово **криптография** происходит от греческих слов, означающих «скрытное письмо». У криптографии долгая и красочная история, насчитывающая несколько тысяч лет. В данном разделе мы всего лишь кратко упомянем некоторые отдельные моменты в качестве введения к последующей информации. Желающим ознакомиться с полной историей криптографии рекомендуется (Kahn, 1995). Для получения всестороннего представления о текущем положении дел см. (Kaufman и др., 2002). С математическими аспектами криптографии можно познакомиться, прочитав книгу (Stinson, 2002). Менее формальным (с математической точки зрения) языком ведется изложение в (Burnett и Paine, 2001).

С профессиональной точки зрения понятия «шифр» и «код» отличаются друг от друга. **Шифр** представляет собой посимвольное или побитовое преобразование, не зависящее от лингвистической структуры сообщения. **Код**, напротив, заменяет целое слово другим словом или символом. Коды в настоящее время не используются, хотя история у них, конечно, славная. Наилучшим считается код, использовавшийся американскими войсками в Тихом океане во время второй Мировой войны. Просто-напросто для ведения секретных переговоров использовались носители языка индейцев навахо, словами из которого обозначались военные термины. Например, слово *чаи-дагахи-найл-цайди* (буквально — убийца черепах) означало противотанковое оружие. Язык навахо тоновый (для различения смысла используется повышение или понижение тона), весьма сложный, не имеет письменной формы. Но самое большое его достоинство заключалось в том, что ни один японец не имел о нем ни малейшего представления.

В сентябре 1945 года *Уния Сан-Диего* так описывала этот код: «В течение трех лет, где бы ни высаживались военные моряки, уши японцев различали лишь странный булькающий шум, перемежающийся с другими звуками. Все это напоминало клич тибетского монаха или звук опустошаемой бутылки с горячей водой». Японцы так и не смогли взломать этот код, и многие носители языка индейцев навахо были удостоены высоких воинских наград за отличную службу и смелость. Тот факт, что США смогли расшифровать японский код, а японцы так и не узнали язык навахо, сыграл важную роль в американской победе в Тихом океане.

Основы криптографии

Исторически использовали и развивали искусство криптографии представители четырех профессий: военные, дипломатический корпус, люди, ведущие дневники, и любовники. Из них наиболее важную роль в развитии этой области сыграли военные. В военных организациях секретные сообщения традиционно отдавались для зашифровки и передачи плохо оплачиваемым шифровальщикам. Сам объем сообщений не позволял выполнить эту работу небольшим количеством элитных специалистов.

До появления компьютеров одним из основных сдерживающих факторов в криптографии была неспособность шифровальщика выполнить необходимые преобразования — зачастую на поле боя, с помощью несложного оборудования. Кроме того, довольно сложной задачей было быстрое переключение с одного криптографического метода на другой, так как для этого требовалось переобучение большого количества людей. Тем не менее опасность того, что шифровальщик может быть захвачен противником, заставила развить способность к постоянной смене криптографических методов. Модель процесса шифрования—дешифрации показана на рис. 8.1.

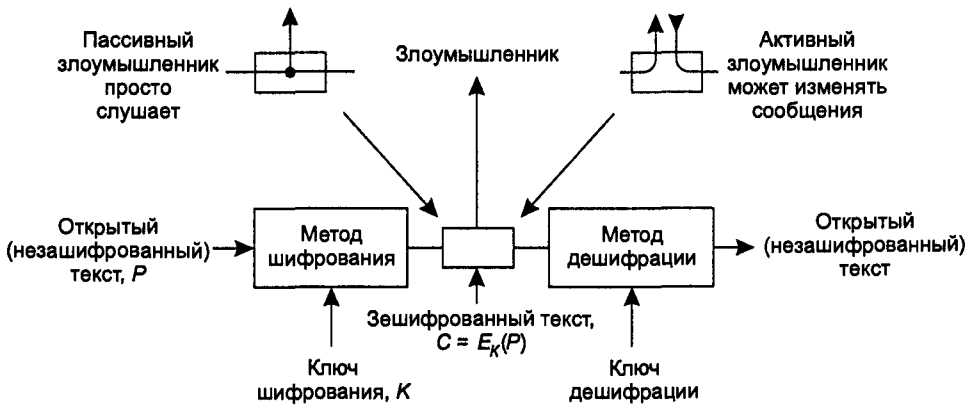


Рис. 8.1. Модель процесса шифрования—дешифрации

Сообщения, подлежащие зашифровке, называемые **открытым текстом**, преобразуются с помощью функции, вторым входным параметром которой является **ключ**. Результат процесса шифрования, называемый **зашифрованным текстом**, передается обычно по радио или через связного. Предполагается, что противник или **злоумышленник** слышит и аккуратно копирует весь зашифрованный текст. Однако в отличие от получателя, которому предназначается данное сообщение, злоумышленник не знает ключа дешифрации, и поэтому расшифровка сообщения представляет для него большие трудности, а порой она просто невозможна. Иногда злоумышленник может не только прослушивать канал связи (пассивный злоумышленник), но способен также записывать сообщения и воспроизводить их позднее, вставлять свои сообщения или модифицировать оригинальные со-

общения, прежде чем они достигнут получателя (активный злоумышленник). Искусство взлома шифров называется **криптоанализом**. Искусства изобретать шифры (криптография) и взламывать их (криптоанализ) называются вместе **криптологией**.

Обычно для обозначения открытого текста, зашифрованного текста и ключей полезно использовать специальную нотацию. Мы будем использовать формулу $C = E_K(P)$, обозначающую, что при зашифровке открытого текста P с помощью ключа K получается зашифрованный текст C. Аналогично формула $P = D_K(C)$ означает расшифровку зашифрованного текста C для восстановления открытого текста. Из этих двух формул следует, что

$$D_K(E_K(P)) = P.$$

Такая нотация предполагает, что E и D являются просто математическими функциями. Они в действительности таковыми и являются. Единственная хитрость состоит в том, что обе эти функции имеют по два параметра, один из которых (ключ) мы написали не в виде аргумента, а в виде нижнего индекса, чтобы отличать его от сообщения.

Основное правило криптографии состоит в предположении, что криптоаналитику (взломщику кода) известен используемый метод шифрования. Другими словами, злоумышленник точно знает, как работает метод шифрования E на рис. 8.1. Из-за огромных усилий, необходимых для разработки, тестирования и установки нового метода, каждый раз, когда старый метод оказывался или считался скомпрометированным, хранить секрет шифрования просто непрактично. А предположение, что метод остается секретным, когда это уже не так, могло бы причинить еще больший вред.

Здесь на помощь приходит ключ шифрования. Ключ состоит из относительно короткой строки, определяющей один из огромного количества вариантов результата шифрования. В отличие от самого метода шифрования, который может изменяться только раз в несколько лет, ключ можно менять так часто, как это нужно. Таким образом, наша базовая модель представляет собой постоянный и известный общий метод, в котором в качестве параметра используется секретный и легко изменяемый ключ. Идея, заключающаяся в предположении о том, что криптоаналитику известен метод, и краеугольным камнем секретности является эксклюзивный ключ, называется **принципом Керкгофа**. Его в 1883 году впервые высказал фламандский военный шифровальщик Аугуст Керкгоф (Auguste Kerckhoff, 1883). Таким образом, принцип Керкгофа гласит:

Алгоритмы шифрования общедоступны; секретны только ключи.

Секретности алгоритма не стоит придавать большого значения. Попытка сохранить алгоритм в тайне, называемая в торговле **безопасностью за счет неясности** (security by obscurity), обречена на провал. К тому же, опубликовав свой алгоритм, разработчик получает бесплатную консультацию от большого количества ученых-криптоаналитиков, горящих желанием взломать новую систему и тем самым продемонстрировать свои ум и ученость. Если никто не смог взломать алгоритм за пять лет с момента его опубликования, то, по-видимому, этот алгоритм достаточно прочен.

Поскольку реально в тайне хранится только ключ, основной вопрос заключается в его длине. Рассмотрим простой кодовый замок. Его основной принцип состоит в том, что вы последовательно вводите десятичные цифры. Все это знают, но ключ хранится в секрете. Ключ длиной в две цифры образует 100 вариантов. Ключ длиной в три цифры означает 1000 вариантов, а при длине ключа в шесть цифр число комбинаций достигает миллиона. Чем длиннее ключ, тем выше **показатель трудозатрат** взломщика кода. При увеличении длины ключа показатель трудозатрат для взлома системы путем простого перебора значений ключа растет экспоненциально. Секретность передаваемого сообщения обеспечивается мощным (но все же открытым) алгоритмом и длинным ключом. Чтобы не дать прочитать свою электронную почту младшему брату, достаточно 64-разрядного ключа. В коммерческих системах имеет смысл использовать 128-битный код. Чтобы защитить ваши тексты от правительств развитых государств, потребуются ключи длиной по меньшей мере в 256 бит.

С точки зрения криптоаналитика задача криптоанализа имеет три принципиальных варианта. Во-первых, у криптоаналитика может быть некоторое количество зашифрованного текста без соответствующего открытого текста. Задачки, в которых в качестве исходных данных имеется в наличии **только зашифрованный текст**, часто печатаются в различных газетах в разделе ребусов. Во-вторых, у криптоаналитика может оказаться некоторое количество зашифрованного текста и соответствующего ему открытого текста. В этом случае мы имеем дело с **проблемой известного открытого текста**. Наконец, когда у криптоаналитика есть возможность зашифровать любой кусок открытого текста по своему выбору, мы получаем третий вариант проблемы дешифрации, то есть **проблему произвольного открытого текста**. Если бы криптоаналитикам было позволено задавать вопросы типа: «Как будет выглядеть зашифрованное ABCDEFGHJKL?», задачки из газет решались бы очень легко.

Новички в деле криптографии часто полагают, что шифр является достаточно надежным, если он может выдержать атаку первого типа (только зашифрованный текст). Такое предположение весьма наивно. Во многих случаях криптоаналитик может угадать часть зашифрованного текста. Например, первое, что говорят многие компьютеры при входе в систему, это LOGIN:. После того как криптоаналитик получит несколько соответствующих друг другу пар кусков зашифрованного и открытого текста, его работа становится значительно легче. Для обеспечения секретности нужна предусмотрительность криптографа, который должен гарантировать, что система не будет взломана, даже если его оппонент сможет закодировать несколько участков открытого текста по выбору.

Исторически методы шифрования разделились на две категории: метод подстановки и метод перестановки. Мы кратко рассмотрим их в качестве введения в современную криптографию.

Метод подстановки

В шифрах, основанных на **методе подстановки**, каждый символ или группа символов заменяется другим символом или группой символов. Одним из древней-

ших шифров является приписываемый Юлию Цезарю (Julius Caesar) **шифр Цезаря**. Этот шифр заменяет все буквы алфавита на другие с помощью циклического сдвига на три позиции. Так, буква *a* становится буквой *D*, *b* становится *E*, *c* превращается в *F*, ..., а *z* — в *C*. Например, слово *attack* превращается в *DWWDFN*. В наших примерах открытый текст будет обозначаться строчными символами, а зашифрованный текст — прописными.

Некоторое обобщение шифра Цезаря представляет собой сдвиг алфавита не на три символа, а на произвольное число *k* символов. В этом случае *k* становится ключом к общему методу циклически сдвигаемых алфавитов. Шифр Цезаря, возможно, и сумел обмануть жителей Помпеи, но с тех пор ему более уже никого не удалось ввести в заблуждение.

Следующее усовершенствование состоит в установлении соответствия каждому встречающемуся в открытом тексте символу другого символа. Например,

открытый текст: a b c d e f g h i j k l m n o p q r s t u w x y z
зашифрованный текст: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

Такая система называется **моноалфавитной подстановкой**, ключом к которой является 26-символьная строка, соответствующая полному алфавиту. В нашем примере слово *attack* будет выглядеть, как *QZZQEA*.

На первый взгляд такая система может показаться надежной, так как даже если криптоаналитику известна общая система, он не знает, какой из $26! \approx 4 \cdot 10^{26}$ вариантов ключа применить. В отличие от шифра Цезаря, применение метода простого перебора в данном случае весьма сомнительно. Даже при затратах 1 нс на проверку одного варианта ключа, чтобы перепробовать все ключи, компьютеру понадобится около 10^{10} лет.

Тем не менее, подобный шифр легко взламывается даже при наличии довольно небольших кусков зашифрованного текста. Для атаки шифра может быть использовано преимущество статистических характеристик естественных языков. Например, в английском языке, буква *e* встречается в тексте чаще всего. Следом за ней по частоте использования идут буквы *t*, *o*, *a*, *n*, *i* и т. д. Наиболее часто встречающимися комбинациями из двух символов (**биграммами**) являются *th*, *in*, *er*, *re* и *an*. Наиболее часто встречающимися комбинациями из трех символов, или **триграммами**, являются *the*, *ing*, *and* и *ion*.

Криптоаналитик, пытающийся взломать моноалфавитный шифр, начнет с того, что сосчитает относительные частоты всех символов алфавита. Затем он может попытаться заменить наиболее часто встречающийся символ буквой *e*, а следующий по частоте — буквой *t*. Затем он посмотрит на триграммы и попытается найти что-либо похожее на *tXe*, после чего он сможет предположить, что *X* — это *h*. Аналогично, если последовательность *thYt* встречается достаточно часто, то, вероятно, *Y* обозначает символ *a*. Обладая этой информацией, криптоаналитик может поискать часто встречающуюся триграмму вида *aZW*, что, скорее всего, означает *and*. Продолжая в том же духе, угадывая буквы, биграммы, триграммы и зная, какие последовательности символов являются наиболее вероятными, криптоаналитик побуквенно восстанавливает исходный текст.

Другой метод заключается в угадывании сразу целого слова или фразы. Например, рассмотрим следующий зашифрованный текст, полученный от бухгалтерской фирмы (разбитый на блоки по пять символов):

CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ QJSGS TJQZZ MNQJS VLNSX VSZJU JDSTS JQUUS JUBXJ DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW

В сообщении бухгалтерской фирмы, скорее всего, должно встречаться слово *financial* (финансовый). Используя тот факт, что в этом слове буква *i* встречается дважды, разделенная четырьмя другими буквами, мы будем искать в зашифрованном тексте повторяющиеся символы, отстоящие друг от друга на это расстояние. В результате мы найдем 12 таких мест в тексте в позициях 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 и 82. Однако только в двух случаях, в позициях 31 и 42, следующий символ (соответствующий букве *n* в открытом тексте) повторяется в соответствующем месте. Из этих двух вариантов символ *a* будет иметь правильное расположение только для позиции 31. Таким образом, теперь нам известно, что слово *financial* начинается в позиции 30. Далее можно продолжать, применяя лингвистическую статистику английского языка и угадывая целые слова.

Метод перестановки

Шифры, основанные на методе подстановки, сохраняют порядок символов, но подменяют их. Шифры, использующие **метод перестановки**, меняют порядок следования символов, но не изменяют сами символы. На рис. 8.2 показан простой перестановочный шифр с колоночной перестановкой. Ключом к шифру служит слово или фраза, не содержащая повторяющихся букв. В данном примере в качестве ключа используется слово MEGABUCK. Цель ключа — пронумеровать колонки. Первой колонкой становится колонка под буквой, расположенной ближе всего к началу алфавита, и т. д. Открытый текст записывается горизонтально в строках. Шифрованный текст читается по колонкам, начиная с колонки с младшей ключевой буквой.

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	e	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	e	b	c	d

Открытый текст

pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

Зашифрованный текст

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

Рис. 8.2. Перестановочный шифр

Чтобы взломать перестановочный шифр, криптоаналитик должен вначале понять, что он имеет дело именно с перестановочным шифром. Если взглянуть на частоту символов *E*, *T*, *A*, *O*, *I*, *N* и т. д., легко заметить, что их частоты соответствуют нормальным частотам открытого текста. В таком случае очевидно, что этот шифр является перестановочным, так как каждая буква в таком шифре представляет сама себя.

Затем нужно угадать число колонок. Во многих случаях по контексту сообщения можно угадать слово или фразу. Например, предположим, что криптоаналитик подозревает, что где-то в сообщении должно встретиться словосочетание *milliondollars*. Обратите внимание, что в результате того, что эти слова присутствуют в исходном тексте, в шифрованном тексте встречаются биграммы *MO*, *IL*, *LL*, *LA*, *IR* и *OS*. Символ *O* следует за символом *M* (то есть они стоят рядом по вертикали в колонке 4), так как они разделены в предполагаемой фразе дистанцией, равной длине ключа. Если бы использовался ключ длиной семь, тогда вместо перечисленных выше биграмм встречались бы следующие: *MD*, *IO*, *LL*, *LL*, *IA*, *OR* и *NS*. Таким образом, для каждой длины ключа в шифрованном тексте образуется различный набор биграмм. Перебрав различные варианты, криптоаналитик часто довольно легко может определить длину ключа.

Остается узнать только порядок колонок. Если число колонок k невелико, можно перебрать все $k(k-1)$ возможных комбинаций пар соседних колонок, сравнивая частоты образующихся биграмм со статистическими характеристиками английского языка. Пара с лучшим соответствием считается правильно позиционированной. Затем все оставшиеся колонки по очереди проверяются в сочетании с уже найденной парой. Колонка, в которой биграммы и триграммы дают максимальное совпадение со статистикой, предполагается правильной. Весь процесс повторяется, пока не будет восстановлен порядок всех колонок. Есть шанс, что на данном этапе текст уже будет распознаваемым (например, если вместо слова *million* мы увидим *milloin*, то сразу станет ясно, где сделана ошибка).

Некоторые перестановочные шифры принимают блок фиксированной длины на входе и выдают блок фиксированной длины на выходе. Такие шифры полностью определяются списком, сообщающим порядок, в котором символы попадают в выходной блок. Например, шифр на рис. 8.2 можно рассматривать в виде шифра с 64-символьным блоком. Его выход описывается последовательностью чисел 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, ..., 62. Другими словами, четвертая входная буква, *a*, первой появится на выходе, за ней последует двенадцатая, *f*, и т. д.

Одноразовые блокноты

Разработать шифр, который невозможно взломать, на самом деле весьма просто. Методика для этого известна уже несколько десятилетий. В качестве ключа выбирается произвольная битовая строка, длина которой совпадает с длиной исходного текста. Открытый текст также преобразуется в последовательность двоичных разрядов, например, с помощью стандартной кодировки ASCII. Наконец, эти две строки поразрядно складываются по модулю 2 (операция «исключающее ИЛИ», XOR). Полученный в результате зашифрованный текст сломать невоз-

можно, поскольку в достаточно большом отрывке любая буква, биграмма или триграмма будет равновероятной. Этот метод, известный как **одноразовый блокнот**, теоретически является панацеей от любых атак, независимо от вычислительных мощностей, которыми обладает или будет когда-либо в будущем обладать взломщик. Объясняется этот невероятный факт с помощью теории информации: дело в том, что в зашифрованном сообщении не содержится никакой информации для взломщика, поскольку любой открытый текст является равновероятным кандидатом.

Пример практического использования одноразового блокнота показан на рис. 8.3. Для начала фраза «I love you.» («Я люблю тебя.») преобразуется в 7-битный ASCII-код. Затем выбирается одноразовая последовательность, *Последовательность 1*, которая складывается по модулю 2 с сообщением. В результате получается некий шифр. Для того чтобы его разгадать, криптоаналитику придется перебрать все возможные одноразовые последовательности, всякий раз проверяя, каким получается открытый текст. Например, если попробовать расшифровать послание с помощью *Последовательности 2* (рис. 8.3), получится текст «Elvis lives» («Элвис жив»). Обсуждение правдоподобности этого утверждения выходит за рамки данной книги, однако, как мы видим, исходное сообщение разгадать не удалось, но при этом получилась вполне нормальная с точки зрения английской грамматики фраза. На самом деле, для любой последовательности из 11 символов в кодировке ASCII найдется одноразовый блокнот для ее генерации. Именно это мы имеем в виду, говоря, что в зашифрованном тексте не содержится никакой информации: из него можно извлечь любое сообщение подходящей длины.

```
Сообщение 1:
1001001 0100001 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110
Последовательность 1:
1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011
Зашифрованный текст:
0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101

Последовательность 2:
1011110 0000111 1101000 1010011 1010111 0100110 1000111 0111010 1001110 1110110 1110110
Открытое сообщение 2
1000101 1101100 1110110 1101001 1110011 0100000 1101100 1101001 1110110 1100101 1110011
```

Рис. 8.3. Использование одноразового блокнота для шифрования сообщений и возможность получения произвольного открытого сообщения из зашифрованного путем подстановки другой ключевой последовательности

Одноразовые ключи теоретически являются очень мощным инструментом, однако у них есть ряд практических недостатков. Во-первых, такой длинный ключ невозможно запомнить, поэтому и отправитель, и получатель должны носить с собой письменную копию ключа. Если есть опасность того, что одна из этих копий может быть захвачена неприятелем, хранение письменных копий оказывается весьма нежелательным. Кроме того, полный объем данных, которые могут быть переданы, ограничен размером доступного ключа. Если шпиону повезет и он добудет большое количество информации, может выясниться, что он не сможет пе-

редать все эти сведения в центр, так как ему не хватит длины ключа. Еще одна проблема заключается в чувствительности данного метода к потерянным или вставленным символам. Если отправитель или получатель потеряют синхронизацию, все данные, начиная с этого места, будут испорчены.

С появлением компьютеров метод одноразового блокнота может получить практическое применение. Ключ можно хранить на специальном диске DVD, содержащем несколько гигабит информации. Он даже не вызовет особых подозрений, если его перевозить в коробке от видеокомпакт-диска и в начале даже записать несколько минут фильма. Конечно, в сетях со скоростями передачи данных, исчисляющимися гигабитами, необходимость менять компакт-диск через каждые 30 секунд быстро утомит. Получается, что DVD с одноразовым ключом нужно вручить лично в руки будущему получателю секретного сообщения, а такой подход резко снижает вероятность практического использования метода одноразового блокнота.

Квантовая криптография

Интересно, что решение проблемы передачи по сети одноразовой последовательности пришло из довольно далекой науки — квантовой механики. Эта область остается экспериментальной, однако многообещающей. Если удастся усовершенствовать этот метод, все задачи криптографии можно будет решать с помощью одноразовых последовательностей, ведь это самый надежный способ защиты информации. Далее мы вкратце опишем суть технологии, называемой **квантовой криптографией**. В частности, мы рассмотрим протокол **BB84**, названный так в честь его создателей и года, в котором его описание было впервые опубликовано (Bennet и Brassard, 1984).

Допустим, пользователь по имени Алиса хочет передать одноразовую последовательность другому пользователю, Бобу. Алиса и Боб называются **принципалами**, это главные герои в нашей истории. Например, Боб может быть банкиром, а Алиса — лицом, желающим вступить в деловые отношения с ним. В последнее время практически во всех материалах, касающихся криптографии, имена «Алиса» и «Боб» традиционно используются для обозначения принципалов. Шифровальщики вообще обожают разного рода традиции. Если бы мы стали описывать тут взаимоотношения Андрея и Беллы вместо Алисы и Боба, ни одному слову в этой главе никто бы не поверил (стало бы понятно, что автор, на самом деле, далек от криптографии). А так, может быть, поверят. Поэтому пусть Алиса и Боб будут героями нашей книги.

Итак, если Алисе и Бобу удастся установить единую одноразовую последовательность, их переговоры будут полностью конфиденциальными. Задача стоит следующим образом: как им обменяться секретным ключом, не передавая друг другу DVD? Мы можем предположить, что оба пользователя находятся на разных концах одного оптоволоконного кабеля, по которому они могут передавать и принимать световые импульсы. Тем не менее, нашлась бесстрашная шпионка Труди, которой удалось установить на пути этого кабеля активное подслушивающее устройство. Она может считывать сигналы, идущие в обоих направлениях. Кроме того, она может посылать как в одну, так и в другую сторону фальши-

вые сообщения. Ситуация для Алисы и Боба, казалось бы, безнадежная. Но тут на помощь приходит квантовая криптография, и мы сейчас увидим, что у наших героев появляется лучик надежды.

Квантовая криптография базируется на том факте, что свет передается маленькими порциями, называемыми **фотонами** и обладающими некоторыми необычными свойствами. Кроме того, пропуская свет через поляризационный фильтр, можно добиться его поляризации. Этот факт известен, прежде всего, тем, кто носит солнцезащитные очки, а также фотографам. Световой луч (то есть поток фотонов), проходя через такой фильтр, поляризуется в направлении оси фильтра (например, вертикально). Если после этого пропустить луч через второй фильтр, интенсивность света на выходе будет пропорциональна квадрату косинуса угла между осями фильтров. Если оси расположить перпендикулярно, фотоны через фильтры проникнуть не смогут. Абсолютная ориентация осей в пространстве значения не имеет — важно только их взаимное расположение.

Чтобы сгенерировать одноразовый блокнот, Алисе понадобятся два набора поляризационных фильтров. Первый набор состоит из вертикального и горизонтального фильтров. Это называется **прямолинейным базисом**. Базис — это просто система координат. Второй набор фильтров отличается от первого только тем, что он повернут на 45°, то есть один фильтр можно представить в виде линии, идущей из нижнего левого угла в верхний правый, а другой — из верхнего левого в нижний правый угол. Это называется **диагональным базисом**. Итак, у Алисы есть два набора фильтров, и она может поставить любой из них на пути светового луча. Но четыре стеклышка-поляризатора — это нечто в стиле сказки про Алису в стране чудес. В реальности имеется кристалл, одна из четырех возможных поляризаций которого изменяется электронным способом с огромной скоростью. У Боба есть такое же устройство, как у Алисы. Тот факт, что у Алисы и Боба есть по два базиса, играет важную роль в квантовой криптографии.

В каждом базисе Алиса обозначает одно из направлений нулем, а другое, соответственно, — единицей. В примере, показанном далее, мы предполагаем, что она выбрала вертикальное направление в качестве нулевого, а горизонтальное — в качестве единичного. Независимо от этого направлению «нижний левый — верхний правый» присваивается значение 0, а направлению «верхний левый — нижний правый» — 1. Эта информация передается Бобу в виде открытого текста.

Теперь Алиса берет одноразовый блокнот, построенный, например, генератором случайных чисел (этот генератор сам по себе, кстати, представляет собой довольно сложный предмет), и передает его Бобу. Передача производится поразрядно, для каждого бита один из двух базисов выбирается случайным образом. Чтобы передать бит, фотонная пушка испускает один фотон, поляризованный таким образом, чтобы он мог пройти через базис, выбранный для этого бита. Например, базисы могут выбираться в такой последовательности: диагональный, прямолинейный, прямолинейный, диагональный, прямолинейный и т. д. Чтобы передать одноразовый блокнот, состоящий из последовательности 1001110010100110, с помощью этих базисов, посылаются фотоны, показанные на рис. 8.4, *а*. Для данного одноразового блокнота и соответствующей ему последовательности базисов единственным образом определяется поляризация, применяемая для каждого

бита. Биты, посылаемые одним фотоном за единицу времени, называются **квантобитами**.

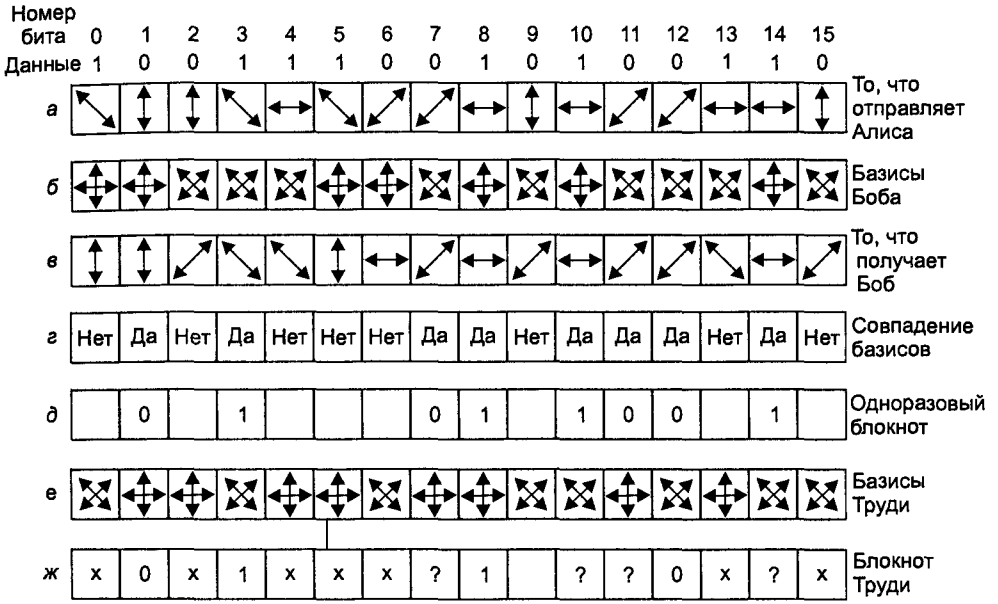


Рис. 8.4. Пример квантовой криптографии

Боб не знает, какой базис надо использовать, поэтому он использует базисы в случайном порядке для каждого прибывающего фотона, как показано на рис. 8.4, *б*. Если для данного фотона базис выбран правильно, Боб получит правильный бит. В противном случае значение бита будет случайным, так как фотон, проходя через поляризатор, повернутый на 45° относительно его собственной поляризации, с равными вероятностями попадет на направление, соответствующее единице или нулю. Это свойство фотонов является одним из основных во всей квантовой механике. Таким образом, некоторые биты будут получены правильно, некоторые — нет, но Боб не может распознать, какие из них являются правильными. Получаемая Бобом информация показана на рис. 8.4, *в*.

Так как же Боб узнает, какие базисы были подставлены правильно, а какие — нет? Для этого он открытым текстом сообщает Алисе, какие базисы он использовал при приеме каждого бита, а она в ответ сообщает (также открытым текстом), какие из базисов были подобраны Бобом корректно. Это показано на рис. 8.4, *г*. Владея этой информацией, Алиса и Боб могут составить битовую строку корректных предположений, что показано на рис. 8.4, *д*. В среднем длина этой строки будет равна половине полной длины исходной строки, однако, так как обеим сторонам это известно, они могут использовать строку корректных предположений в качестве одноразового блокнота. Все, что Алисе надо сделать, это передать битовую строку, длина которой немного превышает удвоенную длину одноразового блокнота. Проблема решена.

Но погодите, мы же забыли про Труди! Предполагается, что ей очень хочется узнать, о чем говорит Алиса, поэтому она внедряет в линию передачи свой детектор и передатчик. К сожалению (для Труди), она тоже не знает, через какой базис пропускать каждый фотон. Лучшее, что она может сделать, это выбирать базисы случайным образом, как Боб. Пример того, как она это делает, показан на рис. 8.4, *е*. Когда Боб открытым текстом сообщает Алисе, какие базисы он использовал, а та отвечает ему, какие из них были правильные, Труди, как и Боб, узнает, какие биты она угадала, а какие — нет. Как видно из рисунка, базисы Труди совпали с базисами Алисы в позициях 0, 1, 2, 3, 4, 6, 8, 12 и 13. Однако из ответа Алисы (рис. 8.4, *г*) ей становится известно, что в одноразовый блокнот входят только биты 1, 3, 7, 8, 10, 11, 12 и 14. Четыре из этих битов (1, 3, 8 и 12) были угаданы правильно, Труди их запоминает. Остальные биты (7, 10, 11 и 14) не были угаданы, и их значения остаются для Труди неизвестными. Таким образом, Бобу известен одноразовый блокнот, начинающийся с последовательности 01011001 (рис. 8.4, *д*), а все, что досталось Труди, — это обрывок 01?1??0? (рис. 8.4, *ж*).

Конечно, Алиса и Боб осознают, что Труди пытается захватить часть их одноразового блокнота, поэтому они стараются уменьшить количество той информации, которая ей может достаться. Для этого они могут произвести некоторые действия над последовательностью. Например, одноразовый блокнот можно поделить на блоки по 1024 бита и возвести каждый из блоков в квадрат, получая, таким образом, числа длиной 2048 бит. Затем можно использовать конкатенацию 2048-битных чисел в качестве одноразового блокнота. Имея лишь часть битовой строки, Труди никогда не сможет проделать эти преобразования. Действия над исходным одноразовым блокнотом, уменьшающие долю информации, получаемой Труди, называются **усилением секретности**. На практике вместо поблочного возведения в квадрат применяются сложные преобразования, в которых каждый выходной бит зависит от каждого входного.

Бедная Труди. У нее не только нет идей по поводу того, как выглядит одноразовый блокнот, но она к тому же понимает, что ее присутствие не является ни для кого секретом. Как-никак, ей приходится передавать каждый принятый бит Бобу, чтобы он думал, будто разговаривает с Алисой. Беда в том, что лучшее, что может сделать Труди, это передавать каждый принятый квантобит с использованием поляризации, с помощью которой он был принят ею. При этом примерно в половине случаев поляризация будет неправильной, что приведет к появлению множества ошибок в одноразовом блокноте Боба.

Когда, наконец, Алиса начинает отправлять данные, она шифрует их, используя сложный код с упреждающей коррекцией ошибок. С точки зрения Боба, ошибка в одном бите одноразовой последовательности — это то же самое, что ошибка передачи данных, произошедшая в одном бите. В любом случае, результат состоит в получении неправильного значения бита. С помощью упреждающей коррекции ошибок, возможно, удастся восстановить потерянную информацию, но можно, кроме того, сосчитать количество ошибок. Если оно намного превышает ожидания (связанные с вероятностью возникновения ошибок оборудования), становится очевидно, что линию прослушивает Труди, и Боб может

принять необходимые меры (например, сказать Алисе, чтобы она переключилась на радиоканал, вызвать полицию и т. д.). Если бы Труди могла скопировать фотон и обрабатывать свою копию, а исходный фотон пересылать Бобу в неизменном виде, у нее был бы шанс остаться незамеченной, однако на сегодняшний день отсутствуют методы клонирования фотонов. Но даже если Труди удастся получить копию фотона, значения квантовой криптографии это ни в коей мере не умалит.

Квантовая криптография может работать при наличии оптоволоконных каналов длиной 60 км, однако требуется сложное и дорогое оборудование. Несмотря на это сама идея весьма многообещающая. Более подробно о квантовой криптографии можно узнать из книги (Mullins, 2002).

Два фундаментальных принципа криптографии

Хотя на следующих страницах мы рассмотрим много различных криптографических систем, в основе их всех лежат два принципа, очень важных для понимания.

Избыточность

Первый принцип гласит, что все зашифрованные сообщения должны содержать определенную избыточность, то есть информацию, не требующуюся для понимания сообщения. Поясним это на примере. Представим себе компанию «Домосед», торгующую по почтовым заказам продуктами 60 000 наименований. Радуясь, что им удалось так экономно распорядиться ресурсами, программисты компании «Домосед» решили, что весь бланк заказа будет состоять из 16 байт имени клиента, за которым последует 3-байтовое поле товара (1 байт для обозначения количества и 2 байта для идентификатора товара). Последние три байта было решено закодировать с помощью очень длинного ключа, известного только клиенту и компании «Домосед».

На первый взгляд, такая схема может показаться надежной, в частности, потому, что пассивные злоумышленники не смогут расшифровать сообщения. К сожалению, в этой схеме имеется критический недостаток, полностью ее обесценивающий. Предположим, какой-нибудь недавно уволенный сотрудник хочет отомстить компании «Домосед» за увольнение. Перед самым уходом ему удается забрать с собой часть списка клиентов. За ночь он составляет программу, посылающую фиктивные заказы с настоящими именами клиентов. Поскольку списка ключей у него нет, он просто помещает в последние три байта случайные числа и посылает сотни заказов компании «Домосед».

Когда эти сообщения прибывают, компьютер компании «Домосед» по имени клиента находит ключ для дешифрации сообщения. К несчастью для компании «Домосед», почти все 3-байтовые сообщения могут восприниматься как достоверные, поэтому компьютер начинает печатать заявки на доставку товаров. Хотя может показаться странным, если клиент заказывает 837 сидений для детских качелей или 540 песочниц, однако вполне возможно, что клиент собирается заняться строительством детских игровых площадок. Таким образом, активный

злоумышленник (уволенный сотрудник) способен доставить очень много неприятностей, даже не вникая в смысл сообщений, посылаемых его компьютером.

Эта проблема может быть решена при помощи добавления избыточной информации к каждому сообщению. Например, если добавить к трем шифруемым байтам еще девять, например, нулевых, уволенный сотрудник уже не сможет сформировать серьезный поток достоверно выглядящих сообщений. Мораль этой истории в том, что все сообщения должны содержать достаточное количество избыточной информации, чтобы активный злоумышленник не смог выдать случайный мусор за настоящие сообщения.

Однако добавление избыточной информации облегчает работу криптоаналитика по взлому шифра. Предположим, что конкуренция в бизнесе почтовых заказов чрезвычайно высока и что главному конкуренту компании «Домосед», фирме «Лежебока», очень хочется узнать, сколько песочниц в месяц продает компания «Домосед». Для этого «лежебоки» подключились к телефонной линии «домоседов». В исходной схеме с 3-байтовыми номерами криптоанализ был почти невозможен, поскольку, предположив значение ключа, криптоаналитик не мог проверить правильность своей догадки. Как-никак, почти все сообщения были технически корректны. С введением новой 12-байтовой схемы криптоаналитик легко сможет отличить допустимое сообщение от недопустимого. Итак:

Криптографический принцип номер 1: Сообщения должны содержать избыточные данные.

Другими словами, при расшифровке сообщения получатель должен иметь возможность проверить его подлинность путем анализа и, возможно, выполнения несложных вычислений. Избыточность требуется для того, чтобы можно было противостоять попыткам активных злоумышленников обмануть получателя фальшивыми сообщениями, содержащими мусор. Вместе с тем, добавление избыточной информации облегчает пассивным злоумышленникам задачу взлома системы, так что здесь есть определенное противоречие. Кроме того, избыточные данные никогда не должны принимать форму последовательности нулей в начале или конце сообщения, так как при зашифровке подобных сообщений некоторые алгоритмы дают более предсказуемые результаты, что облегчает работу криптоаналитиков. Многочлен циклического избыточного кода (CRC) значительно лучше подойдет для этих целей, чем просто ряд нулей, поскольку получатель сможет проверить его корректность, и это несколько усложняет работу криптоаналитика. Гораздо удобнее использовать криптографическую хэш-функцию, речь о которой пойдет ниже.

Возвращаясь к квантовой криптографии, хочется сказать о том, какую роль в этой технологии играет избыточность. Из-за того, что Труди перехватывает фотоны, некоторые биты одноразового блокнота, получаемого Бобом, будут иметь неправильные значения. Бобу требуется некоторая избыточность информации, содержащейся во входящих сообщениях, для определения наличия ошибок. Одной из грубых форм избыточности можно считать повторение передачи одного и того же сообщения. Если две пришедшие копии оказываются не идентичными, Боб узнает, что либо канал сильно зашумлен, либо кто-то перехватывает данные.

Конечно, отправлять все по два раза — это слишком. Гораздо лучше использовать код Хэмминга или Рида—Соломона для определения и коррекции ошибок. Однако должно быть понятно, что для того чтобы отличать настоящие сообщения от поддельных, необходима некоторая избыточность. Это особенно важно, если присутствует активный злоумышленник.

Ограниченный срок годности

Второй принцип криптографии состоит в том, что необходимо иметь методы, позволяющие удостовериться в том, что пришедшее сообщение свежее, то есть было послано совсем недавно. Эта мера направлена на борьбу с активными злоумышленниками, воспроизводящими перехваченные ими старые сообщения. Если не принять подобных мер, наш уволенный может подсоединиться к телефонной линии компании «Домосед» и просто повторять посланные ранее настоящие сообщения. Итак, сформулируем утверждение:

Криптографический принцип номер 2: Необходим способ борьбы с повторной отправкой посланных ранее сообщений.

Одной из подобных мер является включение в каждое сообщение временного штампа, действительного, скажем, только в течение 10 с. Получатель может просто хранить принятые сообщения в течение 10 с, отсеивая дубликаты. Сообщения возрастом более 10 с просто игнорируются как устаревшие. Другие меры защиты от дубликатов будут обсуждаться позже.

Алгоритмы с симметричным криптографическим ключом

В современной криптографии применяются те же основные идеи, что и в традиционной криптографии, то есть перестановка и подстановка, но акценты расставляются иначе. В традиционной криптографии применялись простые алгоритмы. Сегодня верно обратное: целью является создание настолько сложного и запутанного алгоритма шифрования, что даже если криптоаналитику попадут в руки целые горы зашифрованного текста, он не сможет извлечь из этого никакой пользы.

Первым классом алгоритмов шифрования, который мы изучим, будет класс **алгоритмов с симметричным ключом**. Он получил такое название благодаря тому, что для шифрации и дешифрации сообщений применяется один и тот же ключ. На рис. 8.1 показан пример использования алгоритма с симметричным ключом. В частности, мы подробно рассмотрим **блочные шифры**, которые принимают на входе n-битные блоки открытого текста и преобразуют их с использованием ключа в n-битный шифр.

Криптографические алгоритмы могут быть реализованы как аппаратно (что повышает скорость их работы), так и программно (для повышения гибкости). Несмотря на то, что большая часть наших размышлений касается алгоритмов и протоколов, не зависящих от конкретной реализации, нам кажется полезным рассмотрение принципов построения шифровальной аппаратуры. Подстановки

и перестановки могут быть реализованы при помощи простых электрических цепей. На рис. 8.5 показано устройство, называющееся **P-блоком** (литера P означает permutation — перестановка) и используемое для перестановки восьми входных разрядов. Если пронумеровать входные биты сверху вниз (01234567), выход этого конкретного P-блока будет выглядеть как 36071245. При помощи соответствующего внутреннего устройства P-блока (распайки проводов) можно заставить его выполнять любую операцию перестановки практически со скоростью света, так как никакие вычисления в нем не производятся, а просто-напросто передается сигнал со входа на выход. Такое решение соответствует принципу Керкгофа: взломщик знает, что используется метод перестановки битов. Однако он не знает ключа, заключающегося в порядке перестановок.

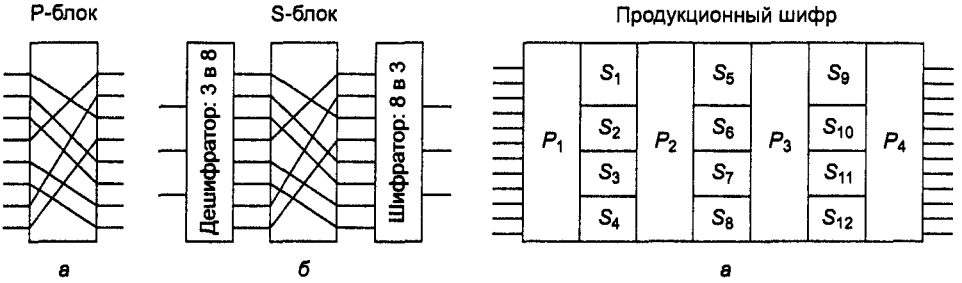


Рис. 8.5. Основные элементы продукционных шифров: P-блок (*а*); S-блок (*б*); продукционный шифр (*в*)

Подстановки (то есть замещения) выполняются **S-блоками** (S означает substitution — подстановка, замена), как показано на рис. 8.5, *б*. В данном примере на вход подается 3-битный открытый текст, а на выходе появляется 3-битный зашифрованный текст. Для каждого входного сигнала выбирается одна из восьми выходных линий декодера путем установки ее в 1. Все остальные линии устанавливаются в 0. Затем эти восемь линий проходят через P-блок, представляющий собой вторую ступень S-блока. Третья ступень производит обратное кодирование одной из восьми линий в 3-битовое двоичное число. Такое устройство заменяет восьмеричные числа 01234567 на 24506713 соответственно. То есть 0 заменяется числом 2, 1 — числом 4 и т. д. Опять же, при соответствующей распайке проводов P-блока внутри S-блока можно реализовать любой вариант подстановки. К тому же такое устройство может быть встроено в аппаратуру и работать на огромных скоростях, поскольку шифраторы и дешифраторы вносят лишь одну или две вентильных задержки (менее 1 нс), а время распространения сигнала внутри P-блока может быть менее 1 пс.

Настоящая сила этих элементов становится очевидна, если сформировать каскад из этих устройств, как показано на рис. 8.5, *в*. Получившееся в результате устройство называется **продукционным шифром**. В данном примере на первом этапе (P_1) 12 входных линий меняются местами. Теоретически, вторая ступень могла бы быть S-блоком, отображающим одно 12-разрядное число в другое 12-разрядное число. Однако такое устройство должно содержать в средней ста-

дии $2^{12} = 4096$ перекрещенных проводов. Вместо этого вход разбивается на четыре группы по 3 разряда, с каждой из которых операция замены выполняется независимо. Хотя такой метод представляет собой лишь частный случай общего решения, его мощь достаточно высока. Выход продукционного шифра можно сделать сложной функцией входа, используя достаточно большое количество дополнительных ступеней.

Продукционные шифры, работающие с k-битными входами и производящие k-битные последовательности, широко распространены. Обычно значение k колеблется от 64 до 256.

Стандарт шифрования данных DES

В январе 1977 году правительство Соединенных Штатов приняло продукционный шифр, разработанный фирмой IBM, в качестве официального стандарта для несекретных сведений. Этот шифр, получивший название **DES** (Data Encryption Standard — стандарт шифрования данных), получил широкое распространение в промышленности для защиты информации. В своем исходном виде он уже больше не является надежным, но в модифицированном виде все еще полезен. Сейчас мы объясним принципы его работы.

Схема DES-шифра показана на рис. 8.6, *а*. Открытый текст шифруется блоками по 64 бита, в результате чего на выходе получаются 64-битные блоки зашифрованного текста. Алгоритм, использующий 56-разрядный ключ, состоит из 19 отдельных этапов. На первом этапе выполняется независимая перестановка 64 разрядов открытого текста. Последний представляет собой обратную перестановку. Предпоследний этап меняет местами левые и правые 32 разряда. Остальные 16 этапов функционально идентичны, но управляются разными функциями входного ключа. Алгоритм был разработан так, чтобы дешифрация выполнялась тем же ключом, что и шифрование. Это обеспечивает соответствие алгоритма принципу симметричных ключей. Этапы при расшифровке просто выполняются в обратном порядке.

Операция, выполняемая на одном из промежуточных этапов, показана на рис. 8.6, *б*. На каждом этапе из двух порций по 32 разряда на входе формируются две порции по 32 разряда на выходе. Правая половина входа просто копируется в левые разряды выхода. Правые 32 выходных разряда представляют собой сумму по модулю 2 левой части входа и функции правой части входа и ключа данного этапа K_i. Вся сложность шифра заключается в этой функции.

Функция состоит из четырех последовательно выполняемых шагов. Сначала из 32 разрядов правой части R_{i-1} с помощью фиксированной перестановки и дублирования формируется 48-разрядное число E. На втором шаге число E и ключ K_i складываются по модулю 2. Затем выход разделяется на восемь групп по шесть разрядов, каждая из которых преобразуется независимым S-блоком в 4-разрядные группы. Наконец, эти $8 \cdot 4$ разряда пропускаются через P-блок.

На каждом из 16 этапов используются различные функции исходного ключа. Перед началом работы алгоритма к ключу применяется 56-разрядная перестановка. Перед каждым этапом ключ разделяется на две группы по 28 разрядов, каж-

дая из которых вращается влево на число разрядов, зависящее от номера этапа. Ключ K_i получается из результата этой операции при помощи еще одной перестановки 56 разрядов. На каждом этапе из 56 разрядов ключа выбираются 48 разрядов, которые также переставляются местами.

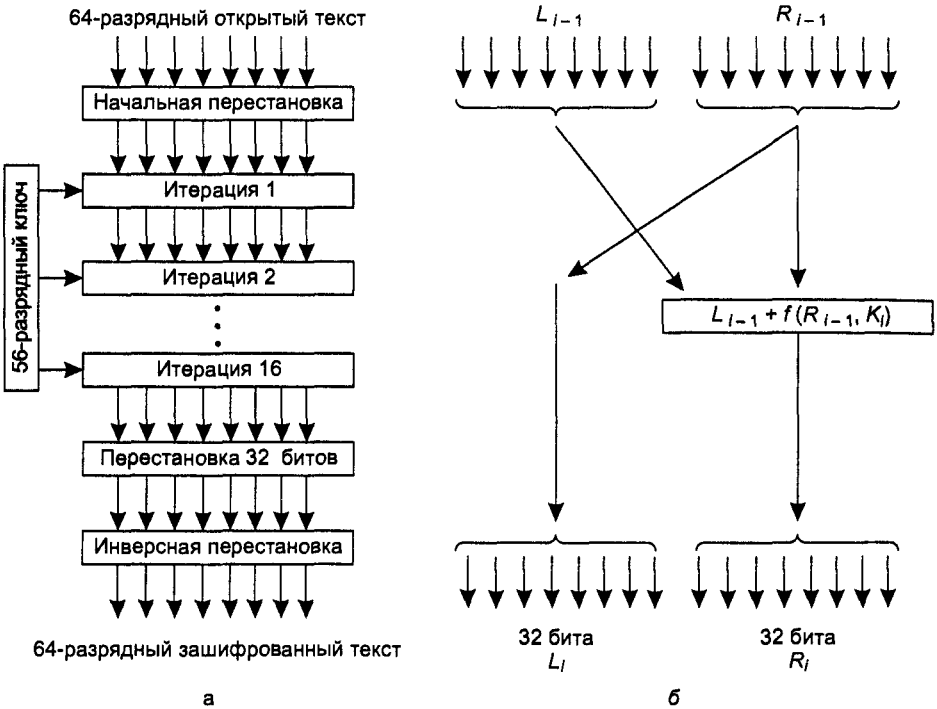


Рис. 8.6. Стандарт шифрования данных DES: общий вид (*а*); детализация одного из этапов (*б*)

Иногда для повышения надежности DES используется метод, называемый **побелкой**. Он заключается в том, что перед передачей шифра каждый блок открытого текста складывается по модулю 2 с произвольным 64-битным ключом, затем отправляется в устройство DES, после чего получившийся шифр складывается по модулю 2 со вторым 64-битным ключом. На приемном конце побелка легко устраняется путем выполнения обратных операций (это возможно, если у получателя есть два побелочных ключа). Поскольку применение этого метода увеличивает длину ключа, полный перебор в пространстве значений ключа становится еще более длительным. Обратите внимание: для побелки каждого блока применяется один и тот же ключ (то есть для каждого сообщения имеется только один побелочный ключ).

Стандарт шифрования данных DES был полон противоречий с самого момента его создания. Он основывался на шифре Люцифер (Lucifer), разработанном и запатентованном корпорацией IBM, с той разницей, что IBM использовала 128-разрядный, а не 56-разрядный ключ. Когда федеральное правительство

Соединенных Штатов пожелало стандартизировать какой-то шифр для несекретного применения, оно «пригласило» IBM на «обсуждение» этого вопроса с Агентством национальной безопасности, NSA (National Security Agency), являющимся самым крупным в мире работодателем в области математики и криптоанализа. Агентство национальной безопасности США настолько секретно, что существует даже такая популярная шутка:

Вопрос: Что означает аббревиатура NSA?
Ответ: No Such Agency — такого агентства нет.

После этих обсуждений корпорация IBM уменьшила длину ключа со 128 до 56 бит и решила держать в секрете процедуру разработки стандарта DES. Многие полагали, что длина ключа была уменьшена, чтобы гарантировать, что NSA сможет взломать DES, но организациям с более низким финансированием это будет не по силам. Вероятно, цель засекречивания проекта состояла в сокрытии потайного хода, позволяющего Агентству национальной безопасности еще легче взламывать шифр DES. Когда сотрудник этого управления предложил Институту инженеров по электротехнике и электронике (IEEE) отменить планирующуюся конференцию по криптографии, ощущения комфорта это не прибавило. Агентство национальной безопасности всегда препятствовало всему.

В 1977 году ученые Стэнфордского университета, занимающиеся исследованиями в области криптографии, Диффи (Diffie) и Хеллман (Hellman), разработали машину для взлома кода DES и оценили стоимость ее создания в 20 млн долларов. По небольшому участку открытого текста и соответствующего ему зашифрованному тексту эта машина путем полного перебора 2^{56} вариантов за один день могла найти 56-разрядный ключ. На сегодняшний день такая машина могла бы стоить около 1 млн долларов.

Тройное шифрование с помощью DES

Уже в 1979 году корпорация IBM поняла, что ключ стандарта DES слишком короток, и разработала метод, позволяющий существенно увеличить его надежность с помощью тройного шифрования (Tuchman, 1979). Выбранный метод, ставший с тех пор Международным стандартом 8732, показан на рис. 8.7. Здесь используются два ключа и три этапа. На первом этапе открытый текст зашифровывается (блок Encryption на рисунке) обычным DES ключом K_1. На втором этапе DES работает в режиме дешифрации (блок Decryption), используя ключ K_2. Наконец, выполняется еще одна операция шифрования с ключом K_1.

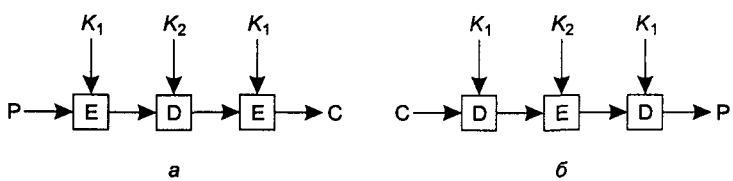


Рис. 8.7. Тройное шифрование с помощью DES (*а*); дешифрация (*б*)

Сразу возникают два вопроса. Во-первых, почему используются только два ключа, а не три? Во-вторых, почему используется последовательность операций EDE (Шифрация Дешифрация Шифрация), а не EEE (Шифрация Шифрация Шифрация)? Причина использования всего двух ключей в том, что даже самые параноидальные шифровальщики в мире считают, что в настоящее время ключа длиной 112 бит вполне достаточно для коммерческих приложений (хотя в мире криптографии паранойя считается достоинством, а не болезнью). Переход на 168 разрядов повлечет лишь дополнительные расходы по хранению и транспортировке дополнительного ключа, а реальной пользы принесет мало.

Использование последовательности шифрования, дешифрации и снова шифрования объясняется обратной совместимостью с существующими DES-системами с одним ключом. Обе функции, шифрования и дешифрации, устанавливают соответствия между наборами 64-разрядных чисел. С точки зрения криптографии обе функции одинаково надежны. Однако использование EDE вместо EEE позволяет компьютеру, применяющему тройное шифрование, общаться с компьютером, применяющим обычное одиночное шифрование, просто установив $K_1 = K_2$. Таким образом, тройное шифрование можно легко включать в виде дополнительного режима работы, что не представляет интереса для ученых криптографов, но довольно важно для корпорации IBM и ее клиентов.

Улучшенный стандарт шифрования AES

В какой-то момент стало понятно, что ресурс DES (даже с тройным шифрованием) уже приближается к концу. Тогда Национальный институт стандартов и технологий (NIST) — агентство Министерства торговли, занимающееся разработкой стандартов для Федерального правительства США, — решило, что правительству нужен новый криптографический стандарт для несекретных данных. NIST ясно осознавал все противоречия, связанные с DES, и прекрасно понимал, что как только будет объявлено о создании нового стандарта, все, кто хоть что-то смыслит в криптографической политике, по умолчанию будут предполагать, что и здесь имеется лазейка, с помощью которой Агентство национальной безопасности с легкостью сможет расшифровывать любую информацию. На таких условиях вряд ли кто-то согласится применять у себя новую технологию, и она, скорее всего, так и умрет в безвестности.

Исходя из этих предпосылок, институт стандартов и технологий применил неожиданный для правительственного бюрократического аппарата подход: он решил просто спонсировать криптографический конкурс. В январе 1997 года ученые со всего мира были приглашены для представления своих разработок, касающихся нового стандарта, который назвали **AES** (Advanced Encryption Standard — улучшенный стандарт шифрования). Требования, предъявляемые к разработкам, были таковы:

1. Алгоритм должен использовать симметричный блочный шифр.
2. Все детали разработки должны быть общедоступны.
3. Должны поддерживаться длины ключей 128, 192 и 256 бит.

4. Должна быть возможна как программная, так и аппаратная реализация.
5. Алгоритм должен быть общедоступным или базирующимся на не дискредитировавших себя понятиях.

Было рассмотрено 15 серьезных предложений. На общедоступных конференциях разработчики представляли свои проекты, а оппоненты должны были приложить максимум усилий для поиска возможных недостатков в каждом из проектов. В августе 1998 года Институтом стандартов и технологий были выбраны пятеро финалистов. Выбор основывался в основном на таких аспектах, как обеспечиваемая безопасность, эффективность, простота, гибкость, а также требования к памяти (это важно для встроенных систем). Был проведен еще ряд конференций, на которых было высказано множество критических замечаний. На последней конференции было проведено независимое голосование. Его результаты выглядели следующим образом:

1. Rijndael (Джон Домен (John Daemen) и Винсент Раймен (Vincent Rijmen), 86 голосов).
2. Serpent (Росс Андерсон (Ross Anderson), Эли Бихам (Eli Biham) и Ларс Кнудсен (Lars Knudsen), 59 голосов).
3. Twofish (команда, возглавляемая Брюсом Шнайером (Bruce Schneier), 31 голос).
4. RC6 (компания RSA Laboratories, 23 голоса).
5. MARS (корпорация IBM, 13 голосов).

В октябре 2000 года NIST объявил о том, что он также голосует за Rijndael, и уже в ноябре 2001 года Rijndael становится стандартом правительства США, опубликованным как Федеральный стандарт обработки информации, FIPS 197. Благодаря полной открытости конкурса, а также благодаря техническим возможностям Rijndael и тому факту, что выигравшая конкурс команда состояла из двух молодых бельгийских шифровальщиков (которые вряд ли стали бы сотрудничать с NSA, предоставляя какие-то лазейки), ожидается, что Rijndael станет доминирующим мировым криптографическим стандартом, по крайней мере, на ближайшее десятилетие. Название Rijndael (произносится примерно как Райндол) представляет собой сокращение фамилий авторов: Раймен + Домен.

Rhindael поддерживает длины ключей и размеры блоков от 128 до 256 бит с шагом в 32 бита. Длины ключей и блоков могут выбираться независимо друг от друга. Тем не менее, стандарт AES говорит о том, что размер блока должен быть равен 128 битам, а длина ключа — 128, 192 или 256 бит. Однако вряд ли кто-то будет использовать 192-битные ключи, поэтому фактически AES применяется в двух вариантах: со 128-битными блоками и 128-битными ключами, а также со 128-битными блоками и 256-битными ключами.

Далее приводится описание алгоритма, и там мы рассматриваем только один случай — 128/128, поскольку именно это, скорее всего, станет нормой для коммерческих приложений. 128-битный ключ означает, что размер пространства его значений равен $2^{128} \approx 3 \cdot 10^{38}$. Даже если Агентству национальной безопасности удастся собрать машину на миллионе параллельных процессоров, каждый из ко-

торых будет способен вычислять один ключ в пикосекунду, на перебор всех значений потребуется около 10^{10} лет. К тому времени Солнце уже давно потухнет, и нашим далеким потомкам придется читать распечатку со значением ключа при свете свечи.

Rijndael

С математической точки зрения, метод Rijndael основывается на теории полей Галуа, благодаря чему можно строго доказать некоторые его свойства, касающиеся секретности. Тем не менее, можно рассматривать его и с точки зрения кода программы на языке C, не вдаваясь в математические подробности.

Как и в DES, в Rijndael применяются замены и перестановки. И там, и там используются несколько итераций, их число зависит от размера ключа и блока и равно 10 для 128-разрядного ключа и 128-битных блоков; для максимального размера ключа и блоков число итераций равно 14. Однако, в отличие от DES, все операции выполняются над целыми байтами, что позволяет создавать эффективные реализации как в аппаратном, так и в программном исполнении. Схематичный алгоритм метода Rijndael приведен в листинге 8.1.

Листинг 8.1. Схематичный алгоритм метода Rijndael

```
#define LENGTH 16/* Число байтов в блоке данных или ключе */
#define NROWS 4/* Число строк в массиве state */
#define NCOLS 4/* Число столбцов в массиве state */
#define ROUNDS 10/* Число итераций */
typedef unsigned char byte/8-разрядное целое без знака */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
int r;/* Счетчик цикла */
byte state[NROWS][NCOLS];/* Текущее состояние */
struct{byte k[NROWS][NCOLS];} rk[ROUNDS + 1];/* Ключи итерации */

expand_key(key,rk);/* Сформировать ключи итерации */
copy_plaintext_to_text(state, plaintext);/* Инициализация текущего состояния */
xor_roundkey_into_state(state, rk[0]);/* Сложить по модулю 2 ключ с текущим состоянием
*/

for(r=1; r<=ROUNDS; r++) {
substitute(state);/* Пропустить каждый байт через S-блок */
rotate_rows(state);/* Повернуть строку i на i байт */
if(r < ROUNDS) mix_columns(state);/* Смешивающая функция */
xor_roundkey_into_state(state, rk[r]);/* Сложить по модулю 2 ключ с текущим состоянием
*/
}
copy_state_to_ciphertext(ciphertext, state);/* Вернуть результат */
}
```

У функции `rijndael` три аргумента: `plaintext` — массив размером 16 байт, содержащий входные данные, `ciphertext` — массив размером 16 байт, в который будет возвращен шифр, а также `key` — 16-разрядный ключ. В процессе вычислений текущее состояние данных сохраняется в байтовом массиве `state`, размер которо-

го равен NROWS × NCOLS. Для 128-битных блоков данных размер этого массива равен 4×4 байта. В 16 байтах целиком уместится один блок.

Массив state изначально содержит открытый текст и модифицируется на каждом этапе вычислений. На некоторых этапах выполняется побайтовая подстановка. На других этапах — перестановка байтов внутри массива. Могут выполняться и другие преобразования. В конечном итоге содержимое state представляет собой зашифрованный текст, который и возвращается в качестве результата функции.

Алгоритм начинается с распространения ключа по 11 массивам одинакового размера, представляющим состояние (state). Эти массивы хранятся в rk — массиве структур, содержащих массивы состояний. Одна из этих структур будет использована в начале вычислений, а остальные 10 — во время 10 итераций (по одной на итерацию). Вычисление ключа для каждой итерации производится довольно сложным образом, и мы не будем рассматривать детали этого процесса. Достаточно сказать, что для этого осуществляются циклические повороты и суммирования по модулю 2 различных групп разрядов ключа. Подробности можно узнать в (Daemen и Rijmen, 2002).

Следующий шаг состоит в копировании открытого текста в массив state для того, чтобы его можно было обрабатывать во время последующих итераций. Копируется текст в колонки по 4 байта: первые 4 байта попадают в колонку 0, вторые — в колонку 1 и т. д. И колонки, и строки нумеруются с нуля, а итерации — с единицы. Процесс создания 12 байтовых массивов размером 4×4 показан на рис. 8.8.

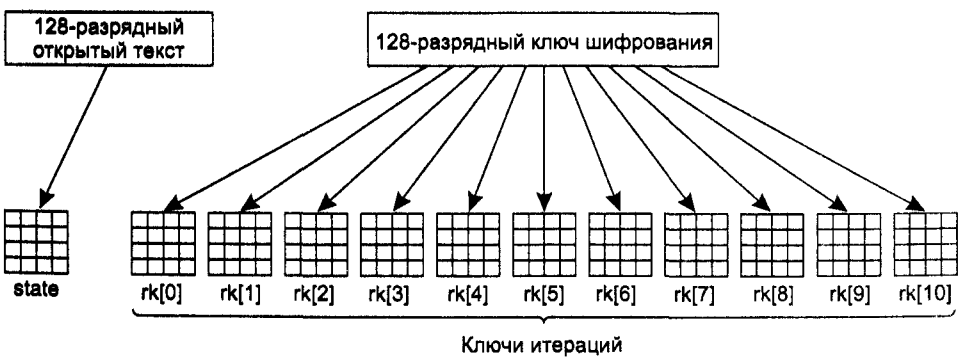


Рис. 8.8. Создание массивов state и rk

Перед началом основного цикла вычислений производится еще одно действие: rk[0] поразрядно складывается по модулю 2 с массивом state. Другими словами, каждый из 16 байт в массиве state заменяется суммой по модулю 2 от него самого и соответствующего байта в rk[0].

Только после этого начинается главное развлечение. В цикле проводятся 10 итераций, в каждой из которых массив state подвергается преобразованию. Каждый раунд (итерация) состоит из четырех шагов. На шаге 1 в state производится посимвольная подстановка. Каждый байт по очереди используется в качестве индекса для S-блока, заменяющего его значение на соответствующую запись S-бло-

ка. На этом шаге получается прямой моноалфавитный подстановочный шифр. В отличие от DES, где используются несколько S-блоков, в Rijndael S-блок всего один.

На шаге 2 каждая из четырех строк поворачивается влево. Строка 0 поворачивается на 0 байт (то есть не изменяется), строка 1 — на 1 байт, строка 2 — на 2 байта, а строка 3 — на 3 байта. Смысл заключается в разбрасывании данных вокруг блока. Это аналогично перестановкам, показанным на рис. 8.5.

На шаге 3 происходит независимое перемешивание всех колонок. Делается это с помощью операции матричного умножения, в результате которого каждая новая колонка оказывается равной произведению старой колонки на постоянную матрицу. При этом умножение выполняется с использованием конечного поля Галуа, $GF(2^8)$. Несмотря на то, что все это кажется довольно сложным, алгоритм устроен так, что каждый элемент матрицы вычисляется посредством всего лишь двух обращений к таблице и трех суммирований по модулю 2 (Daemen и Rijmen, 2002, приложение E).

Наконец, на шаге 4 ключ данной итерации складывается по модулю 2 с массивом `state`.

Благодаря обратимости всех действий расшифровка может быть выполнена с помощью такого же алгоритма, но с обратным порядком следования всех шагов. Однако есть одна хитрость, которая позволяет заниматься расшифровкой, используя алгоритм шифрования с измененными таблицами.

Данный алгоритм обладает не только очень высокой защищенностью, но и очень высокой скоростью. Хорошая программная реализация на машине с частотой 2 ГГц может шифровать данные со скоростью 700 Мбит/с. Такой скорости достаточно для шифрации видео в формате MPEG-2 в реальном масштабе времени. Аппаратные реализации работают еще быстрее.

Режимы шифрования

Несмотря на всю свою сложность, AES (или DES, или любой другой блочный код) представляет собой, по сути дела, моноалфавитный подстановочный шифр с большими длинами символов (в AES используются 128-битные символы, в DES — 64-битные). Для любого отрывка открытого текста шифр при прогоне через один и тот же шифрующий блок будет получаться всегда одинаковым. Скажем, если вы будете 100 раз пытаться зашифровать текст *abcdefgh*, задавая всякий раз один и тот же ключ для алгоритма DES, вы получите 100 одинаковых копий шифра. Взломщик может попытаться использовать этот недостаток при попытке расшифровки текста.

Режим электронного шифроблокнота

Чтобы понять, каким образом это свойство моноалфавитного подстановочного шифра может быть использовано для частичного взлома шифра, мы рассмотрим (тройное) кодирование по стандарту DES только лишь потому, что изображать 64-разрядные блоки проще, чем 128-разрядные. Надо иметь при этом в виду, что AES сталкивается с теми же самыми проблемами. Самый очевидный способ ко-

дирования длинного сообщения заключается в разбиении его на отдельные блоки по 8 байт (64 бита) с последующим кодированием этих блоков по очереди одним и тем же ключом. Последний блок при необходимости можно дополнить до 64 бит. Такой метод называется **режимом электронного шифроблокнота**, по аналогии со старомодными шифроблокнотами, содержавшими слова и соответствующие им шифры (обычно это были пятизначные десятичные числа).

На рис. 8.9 показано начало компьютерного файла, в котором перечислены поощрительные премии сотрудникам компании. Этот файл состоит из последовательных 32-разрядных записей, по одной на сотрудника, следующего формата: 16 байт на имя, 8 байт на должность и 8 байт на премию. Каждый из шестнадцати 8-байтовых блоков (пронумерованных от 0 до 15) кодируется шифром DES.

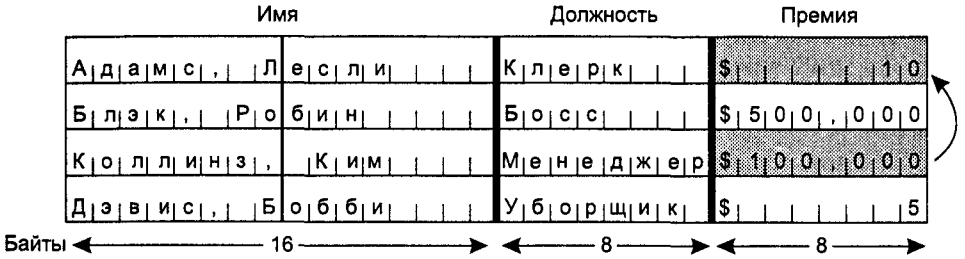


Рис. 8.9. Открытый текст файла, зашифрованного в виде 16 DES-блоков

Лесли только что поругалась с боссом и не рассчитывает на большую премию. Ким, напротив, — любимица босса, что всем известно. Лесли может получить доступ к файлу после того как он будет зашифрован, но до того как он будет отослан в банк. Может ли Лесли исправить ситуацию, имея доступ только к зашифрованному файлу?

В данном случае это очень просто. Все, что нужно сделать Лесли, — это скопировать зашифрованный блок 12 (содержащий премию Ким) и заменить им блок 4 (содержащий премию Лесли). Даже не зная содержимого блока 12, Лесли может рассчитывать на значительно более веселое Рождество. (Скопировать зашифрованный блок 8 тоже можно, но вероятность, что это будет обнаружено, выше; кроме того, Лесли, в общем-то, не жадная).

Режим сцепления блоков шифра

Чтобы противостоять атакам подобного типа, все блочные шифры можно модернизировать таким образом, чтобы замена одного блока вызывала повреждение других блоков открытого текста после их расшифровки, превращая эти блоки (начиная с модифицированного места) в мусор. Один из таких способов — **сцепление блоков шифра**. При этом методе, показанном на рис. 8.10, каждый блок открытого текста перед зашифровкой складывается по модулю 2 с предыдущим уже зашифрованным блоком. При этом одинаковым блокам открытого текста уже не соответствуют одинаковые блоки зашифрованного текста. Таким образом, шифр перестает быть большим моноалфавитным подстановочным шифром. Пер-

вый блок складывается по модулю 2 со случайным **вектором инициализации**, IV (Initialization Vector), передаваемым вместе с зашифрованным текстом.

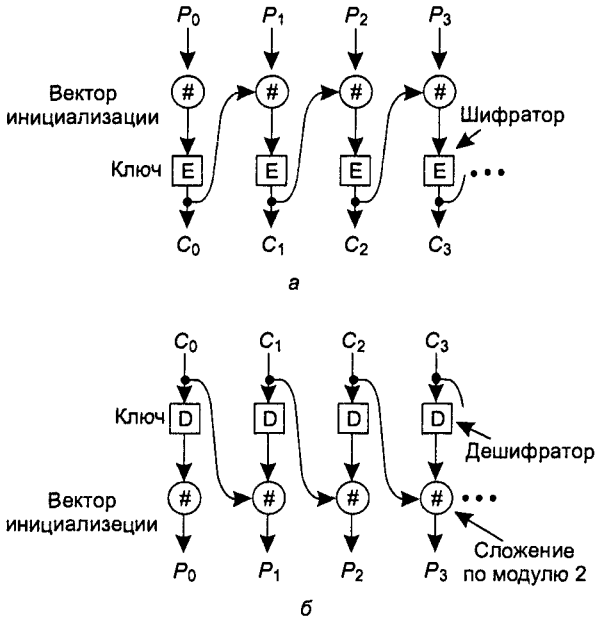


Рис. 8.10. Сцепление зашифрованных блоков: шифрование (*а*); дешифрация (*б*)

Рассмотрим работу сцепления блоков шифра на примере рис. 8.10. Начнем с вычисления $C_0 = E(P_0 \text{ XOR } IV)$. Затем мы вычислим $C_1 = E(P_1 \text{ XOR } C_0)$ и т. д. Расшифровка производится по формуле $P_0 = IV \text{ XOR } D(C_0)$, и т. д. Обратите внимание на то, что блок i является функцией всех блоков открытого текста с 0 по $i - 1$, поэтому один и тот же исходный блок текста преобразуется в разные зашифрованные блоки в зависимости от их расположения. При использовании такого способа шифрования преобразование, произведенное Лесли, приведет к появлению двух блоков чепухи, начиная с поля премии Лесли. Для сообразительного офицера службы безопасности эта странность может послужить подсказкой в последующем расследовании.

Сцепление блоков шифра также обладает тем достоинством, что усложняет криптоанализ, так как одни и те же блоки открытого текста преобразуются в разные зашифрованные блоки. Именно по этой причине и применяется описанный метод.

Режим шифрованной обратной связи

Однако у метода сцепления блоков шифра есть и недостаток, заключающийся в том, что прежде чем может начаться шифрование или дешифрация, должен появиться целый 64-битовый блок данных. Для пользователей интерактивных терминалов, набирающих строки короче восьми символов и ждущих ответа, такой

метод не подходит. Для побайтового шифрования может применяться **режим шифрованной обратной связи** с использованием (тройного) DES, как показано на рис. 8.11. Для стандарта AES идея остается той же самой, только используется 128-разрядный сдвиговый регистр. На рисунке мы видим состояние шифрующей машины после того, как байты с 0 по 9 уже зашифрованы и посланы. Когда прибывает десятый байт открытого текста, как показано на рис. 8.11, *а*, алгоритм DES обрабатывает 64-разрядный сдвиговый регистр, чтобы произвести 64-разрядный зашифрованный блок. Самый левый байт этого зашифрованного текста извлекается и складывается по модулю 2 с P_{10}. Этот байт передается по линии. Затем сдвиговый регистр сдвигается влево на 8 разрядов. При этом байт C_2 извлекается с левого конца регистра, а байт C_{10} вставляется в него на освободившееся место справа от C_9. Обратите внимание на то, что содержимое сдвигового регистра зависит от всей предыстории открытого текста, так что повторяющиеся фрагменты исходного текста будут кодироваться каждый раз по-разному. Как и для метода сцепленных блоков шифра, для начала шифрования этим методом требуется вектор инициализации.

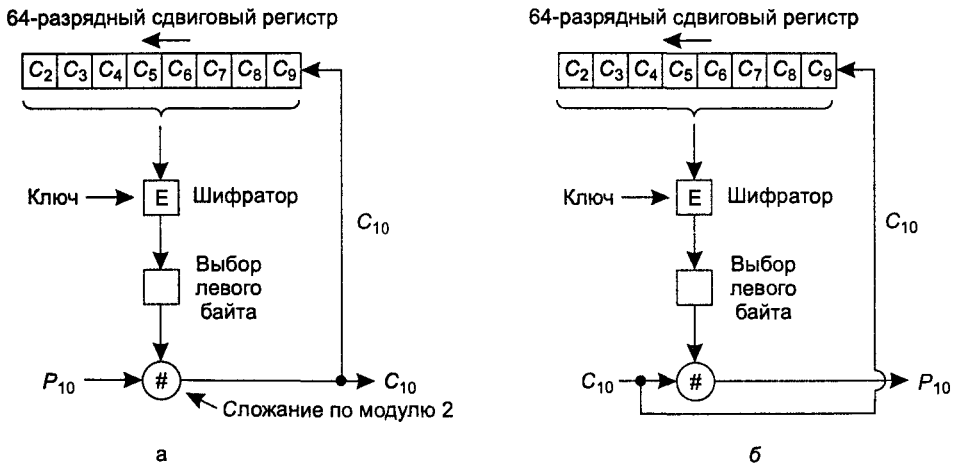


Рис. 8.11. Режим шифрованной обратной связи (*а*); обратная связь по выходу (*б*)

При использовании режима шифрованной обратной связи дешифрация аналогична шифрованию. В частности, содержимое сдвигового регистра *шифруется*, а не *дешифруется*, поэтому байт, который складывается по модулю 2 с C_{10} для получения P_{10}, равен тому байту, который складывается по модулю 2 с P_{10} для получения C_{10}. Пока содержимое двух сдвиговых регистров идентично, дешифрация выполняется корректно. Это показано на рис. 8.11, *б*.

Проблемы с режимом шифрованной обратной связи начинаются, когда при передаче шифрованного таким способом текста один бит случайно инвертируется. При этом испорченными окажутся 8 байтов, расшифровываемые в то время, когда поврежденный байт находится в сдвиговом регистре. Когда поврежденный байт покинет сдвиговый регистр, на выходе опять станет расшифровывать-

ся правильный открытый текст. Таким образом, результат инверсии одного бита оказывается относительно локализованным и не портит всего остатка сообщения.

Режим группового шифра

Тем не менее, существуют приложения, в которых один испорченный при передаче бит приводит к порче 64 бит открытого текста, а это многовато. Для таких приложений существует четвертый вариант, называемый **режимом группового** (потокового) **шифра**. Суть его заключается в том, что выходной блок получается шифрацией вектора инициализации с использованием ключа. Затем этот выходной блок снова шифруется с использованием ключа, в результате чего получается второй выходной блок. Для получения третьего блока шифруется второй блок, и т. д. Последовательность (произвольной длины) выходных блоков, называемая **ключевым потоком**, воспринимается как одноразовый блокнот и складывается по модулю 2 с открытым текстом. В результате получается шифрованный текст, как показано на рис. 8.12, *а*. Обратите внимание: вектор инициализации используется только на первом шаге. После этого шифруются выходные блоки. Кроме того, ключевой поток не зависит от данных, поэтому он в случае необходимости может быть вычислен заранее и совершенно не чувствителен к ошибкам передачи. Процесс дешифрации показан на рис. 8.12, *б*.

Дешифрация осуществляется путем генерации точно такого же ключевого потока на принимающей стороне. Поскольку он зависит только от вектора инициализации и ключа, ошибки передачи шифрованного текста на него не влияют. Таким образом, ошибка в одном бите передаваемого шифра приводит к ошибке только одного бита расшифрованного текста.

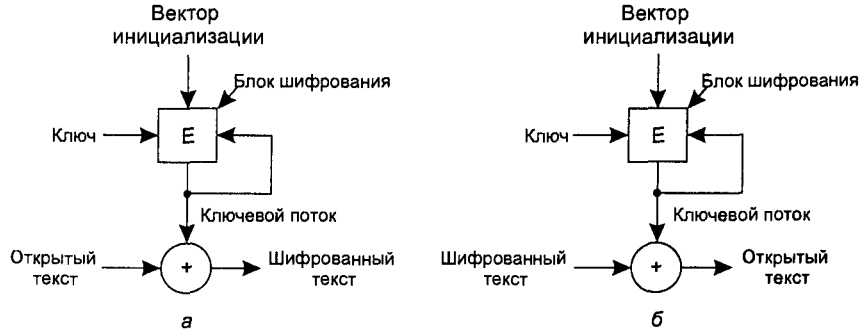


Рис. 8.12. Групповой шифр: шифрование (*а*); дешифрация (*б*)

Важно никогда не использовать одну и ту же пару ключ — вектор инициализации в одном и том же групповом шифре, поскольку при этом всякий раз будет получаться одинаковый ключевой поток. Повторное использование ключевого потока может привести к неприятному эффекту **взлома шифра при помощи многократного использования ключевого потока**. Допустим, блок открытого текста P_0 шифруется с помощью ключевого потока, в результате чего получается сумма

по модулю 2 P_0 и K_0. Затем берется второй блок открытого текста, Q_0, и шифруется тем же ключевым потоком (получаем Q_0 XOR K_0). Криптоаналитик, перехвативший оба блока шифрованного текста, может просто сложить их вместе по модулю 2 и получить в результате P_0 XOR Q_0, убирая тем самым ключ. Теперь у него есть сумма по модулю 2 двух блоков открытого текста. Если один из них известен (или его можно угадать), найти второй — не проблема. В любом случае, взломать сумму по модулю 2 двух блоков открытого текста можно, используя статистические свойства сообщения. Скажем, если передается английский текст, то наиболее часто встречающейся буквой в потоке будет «е», и т. д. Короче говоря, имея сумму по модулю 2 двух частей открытого текста, взломщик с высокой вероятностью сможет вычислить обе части.

Режим счетчика

Все режимы, кроме электронного шифроблокнота, обладают одним и тем же неприятным свойством: доступ к произвольной части зашифрованных данных невозможен. Допустим, например, что файл передается по сети и затем сохраняется на диске в зашифрованном виде. Так иногда делают, если принимающий компьютер представляет собой ноутбук, который может быть украден. Хранить все важные данные в зашифрованном виде действительно полезно: риск утечки секретной информации в случае попадания аппаратуры в руки «нехорошим дядям» резко снижается.

Однако доступ к дискам зачастую бывает необходимо осуществлять в произвольном порядке. Особенно это касается файлов баз данных. Если файл зашифрован в режиме сцепления блоков, придется вначале дешифровать все блоки, предшествующие нужному. Согласитесь, такой способ работы несколько неудобен. По этой причине был введен еще один режим шифрования — **режим счетчика**. Он показан на рис. 8.13. Здесь открытый текст не шифруется напрямую. Вместо этого шифруется вектор инициализации плюс некоторая константа, а уже получающийся в результате шифр складывается по модулю 2 с открытым текстом. Сдвигаясь на 1 по вектору инициализации при шифрации каждого нового блока, можно легко получить способ дешифрации любого места файла. При этом нет необходимости расшифровывать все предшествующие блоки.

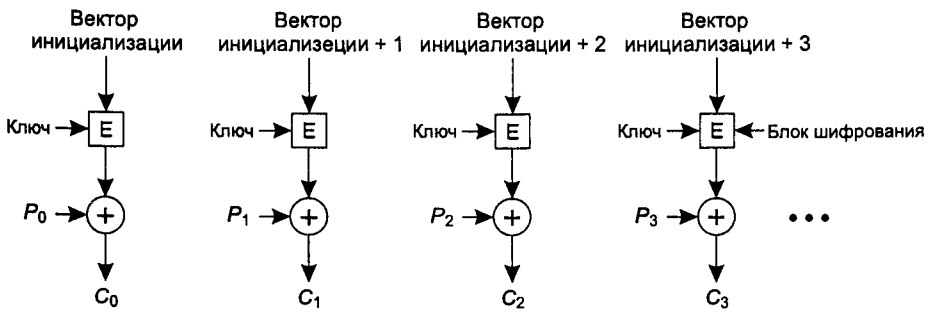


Рис. 8.13. Шифрование в режиме счетчика

Несмотря на то что режим счетчика весьма полезен, у него есть один существенный недостаток, который стоит упомянуть. Допустим, уже использовавшийся однажды ключ *K* будет использован повторно (с другим открытым текстом, но с тем же вектором инициализации), и взломщик захватит весь шифрованный текст, который был послан в обоих случаях. Ключевые потоки остаются неизменными, в итоге возникает риск взлома за счет повторного использования ключевого потока (тот же эффект, на который мы уже указывали, обсуждая режим группового шифра). Все, что криптоаналитику остается сделать, это сложить по модулю 2 два перехваченных сообщения. Тем самым он полностью снимет какую бы то ни было криптографическую защиту, и в его распоряжении окажется сумма по модулю 2 двух блоков открытого текста. Этот недостаток вовсе не означает, что режим счетчика в целом неудачен. Это говорит лишь о том, что как ключи, так и векторы инициализации должны выбираться независимо и случайным образом. Даже если один и тот же ключ случайно попадется дважды, от перехвата информацию может спасти отличающийся вектор инициализации.

Другие шифры

DES и Rijndael — это самые известные криптографические алгоритмы с симметричными ключами. Тем не менее, стоит отметить, что в природе существует еще много других шифров с симметричными ключами. Некоторые из них встраиваются в различные программно-аппаратные продукты. Наиболее распространенные из них перечислены в табл. 8.2.

Таблица 8.2. Некоторые распространенные криптографические алгоритмы с симметричными ключами

Название	Автор	Длина ключа	Комментарии
Blowish	Брюс Шнайер (Bruce Schneier)	1–448 бит	Старый и медленный
DES	IBM	56 бит	Слишком слабый для современных систем
IDEA	Массей (Massey) и Ксюэйа (Xuejia)	128 бит	Хороший, но запатентованный
RC4	Рональд Ривест (Ronald Rivest)	1–2048 бит	Внимание: есть слабые ключи
RC5	Рональд Ривест (Ronald Rivest)	128–256 бит	Хороший, но запатентованный
Rijndael	Домен (Daemen) и Раймен (Rijmen)	128–256 бит	Лучший
Serpent	Андерсон (Anderson), Байхэм (Biham) и Кнудсен (Knudsen)	128–256 бит	Очень сильный
Тройной DES	IBM	168 бит	На втором месте после Rijndael
Twofish	Брюс Шнайер (Bruce Schneier)	128–256 бит	Очень сильный; широко распространен

Криптоанализ

Прежде чем закончить разговор об использовании симметричных ключей в криптографии, необходимо хотя бы упомянуть о четырех направлениях развития криптоанализа. Первый подход называется **дифференциальным криптоанализом** (Biham и Shamir, 1993). Он может использоваться для взлома любых блочных шифров. Для начала анализируется пара блоков открытого текста, различающихся лишь небольшим числом бит. При этом внимательно анализируется происходящее при каждой внутренней итерации во время шифрации. Во многих случаях можно заметить, что одни битовые последовательности встречаются чаще других, и это соображение используется для взлома, основанного на теории вероятностей.

Второй подход, который следует обозначить, называется **линейным криптоанализом** (Matsui, 1994). С его помощью можно взломать DES только с 2^{43} известными открытыми текстовыми блоками. Принцип работы основан на суммировании по модулю 2 некоторых бит открытого текста и изучении результатов для шаблонных последовательностей. Если повторять эту процедуру много раз, половина бит будет иметь нулевые значения, половина — единичные. Тем не менее, довольно часто это соотношение изменяется в ту или иную сторону. Это отклонение, каким бы малым оно ни было, может использоваться для снижения показателя трудозатрат криптоаналитика. Более подробную информацию см. в материалах автора этого метода, Мацуи (Matsui).

Третье направление развития связано с анализом потребляемой электроэнергии для вычисления секретного ключа. Обычно в компьютерах напряжение 3 В соответствует логической единице, а 0 В — логическому нулю. Таким образом, обработка единицы требует большего потребления электроэнергии, чем обработка нуля. Если криптографический алгоритм состоит из цикла, в котором разряды ключа обрабатываются поочередно, взломщик, заменив главные системные n-гигагерцевые часы медленными (например, с частотой 100 Гц) и повесив «крокодилы» на ножки питания и заземления центрального процессора, может с большой точностью отслеживать мощность, потребляемую каждой машинной инструкцией. По этим данным узнать секретный ключ оказывается на удивление просто. Этот метод криптоанализа можно победить лишь аккуратным кодированием алгоритма на языке Ассемблера таким образом, чтобы энергопотребление не зависело ни от общего ключа, ни от ключей каждой итерации.

Четвертый подход основан на временном анализе. Криптографические алгоритмы содержат большое количество условных операторов (`if`), тестирующих биты итерационных ключей. Если части `then` и `else` выполняются за различное время, то, замедлив системные часы и измерив длительность всех шагов, можно вычислить ключи итераций. По этим ключам обычно можно вычислить и общий ключ. Анализ энергопотребления и временной анализ могут применяться и одновременно, что позволяет упростить задачу криптоанализа. Несмотря на то, что анализы энергозатрат и времени выполнения операций могут показаться несколько экзотическими, на самом деле они представляют собой мощные методы, способные взломать любой шифр, если только он не имеет специальной защиты.

Алгоритмы с открытым ключом

Исторически процесс передачи ключа всегда был слабым звеном почти во всех системах шифрования. Независимо от того, насколько прочна была сама криптосистема, если нарушитель мог украсть ключ, система становилась бесполезной. До 1976 года все криптологи исходили из предпосылки, что ключ дешифрации должен быть идентичен ключу шифрования (или один может легко получиться из другого). В то же время, ключи должны были быть у всех пользователей системы. Таким образом, казалось, что эта проблема неустранима: ключи должны быть защищены от кражи, и в то же время их нужно распространять среди пользователей, поэтому их нельзя просто хранить в банковском сейфе.

В 1976 году два исследователя из Стэнфордского университета, Диффи (Diffie) и Хеллман (Hellman), предложили радикально новую криптосистему, в которой ключ шифрования и ключ дешифрации были различными, кроме того, ключ дешифрации нельзя было получить из ключа шифрования. Предложенные ими алгоритм шифрования E и алгоритм дешифрации D (оба параметризованные ключом) должны были удовлетворять следующим трем требованиям:

1. $D(E(P)) = P$.
2. Крайне сложно вывести D из E.
3. E нельзя взломать при помощи произвольного открытого текста.

Первое требование состоит в том, что если применить алгоритм дешифрации D к зашифрованному сообщению $E(P)$, то мы опять получим открытый текст P. Без этого авторизованный получатель просто не сможет расшифровать сообщение. Второе требование говорит само за себя. Третье требование необходимо, потому что, как мы скоро увидим, злоумышленники могут экспериментировать с алгоритмом столько, сколько пожелают. При таких условиях нет никаких причин, по которым ключ шифрования нельзя было бы сделать общедоступным.

Этот метод работает следующим образом. Некто, например Алиса, желая получать секретные сообщения, сначала формирует два алгоритма, удовлетворяющие перечисленным выше требованиям. Затем алгоритм шифрования и его ключ открыто объявляются, отсюда название — **шифрование с открытым ключом**. Это можно сделать, разместив открытый ключ, например, на домашней страничке Алисы. Для обозначения алгоритма шифрования, параметризованного открытым ключом Алисы, мы будем использовать запись E_A. По аналогии (секретный) алгоритм дешифрации, параметризованный персональным ключом Алисы, мы будем обозначать D_A. Боб делает то же самое, открыто объявляя E_B, но храня в тайне D_B.

Теперь посмотрим, сможем ли мы решить проблему установки надежного канала между Алисой и Бобом, которые ранее никогда не встречались. Оба ключа шифрования Алисы и Боба, E_A и E_B, являются открытыми. (Вообще, все пользователи сети могут, становясь пользователями, опубликовать свои ключи шифрования.) Теперь Алиса берет свое первое сообщение P, вычисляет $E_B(P)$ и посылает его Бобу. Боб расшифровывает его с помощью своего секретного ключа D_B, то есть вычисляет $D_B(E_B(P)) = P$. Больше никто не может прочитать это зашифро-

ванное сообщение $E_B(P)$, так как предполагается, что система шифрования достаточно надежна, а получить ключ D_B на основании известного ключа E_B очень трудно. Посылая ответ, Боб передает $E_A(R)$. Таким образом, Алиса и Боб получают надежный секретный канал связи.

Обратите внимание на используемую здесь терминологию. Шифрование с открытым ключом предполагает у каждого пользователя наличие двух ключей — открытого ключа, используемого всеми для шифрования сообщений, посылаемых этому пользователю, и закрытого ключа, требующегося пользователю для дешифрации приходящих к нему сообщений. Мы будем и далее называть эти ключи *открытым* и *закрытым*, чтобы отличать их от *секретных* ключей, используемых для шифрования и дешифрации в обычной криптографии с симметричным ключом.

Алгоритм RSA

Единственная загвоздка состоит в том, чтобы найти алгоритмы, удовлетворяющие всем трем требованиям. Поскольку преимущества шифрования с открытым ключом очевидны, многие исследователи неустанно работали над созданием подобных алгоритмов, и некоторые из них уже опубликованы. Один хороший метод был разработан группой исследователей Массачусетского технологического института (Rivest и др., 1978). Он назван по начальным буквам фамилий трех разработчиков: **RSA** (Rivest, Shamir, Adleman). Этот метод вот уже четверть века выдерживает попытки взлома и считается очень прочным. На его основе построены многие практические системы безопасности. Главный недостаток RSA заключается в том, что для обеспечения достаточного уровня защищенности требуется ключ длиной, по крайней мере, 1024 бита (против 128 бит в алгоритмах с симметричными ключами). Из-за этого алгоритм работает довольно медленно.

В основе метода RSA лежат некоторые принципы теории чисел. Опишем в общих чертах, как пользоваться этим методом. Подробности см. в соответствующих источниках.

1. Выберем два больших простых числа p и q (обычно длиной 1024 бита).
2. Сосчитаем $n = pq$ и $z = (p - 1)(q - 1)$.
3. Выберем число d, являющееся взаимно простым с числом z.
4. Найдем такое число e, что остаток от деления произведения ed на число z равен 1.

Вычислив заранее эти параметры, можно начинать шифрование. Сначала разобьем весь открытый текст (рассматриваемый в качестве битовой строки) на блоки так, чтобы каждое сообщение P попадало в интервал $0 < P < n$. Это несложно сделать, если разбить открытый текст на блоки по k бит, где k — максимальное целое число, для которого $2^k < n$.

Чтобы зашифровать сообщение P, вычислим $C = P^e \pmod{n}$. Чтобы расшифровать C, сосчитаем $P = C^d \pmod{n}$. Можно доказать, что для всех значений P в указанном диапазоне функции шифрования и дешифрации являются взаимно обратными. Чтобы выполнить шифрование, нужны e и n. Для дешифрации тре-

буются d и n. Таким образом, открытый ключ состоит из пары (e, n), а закрытый ключ — из пары (d, n).

Надежность метода обеспечивается сложностью нахождения множителей больших чисел. Если бы криптоаналитик мог разложить на множители (открытое) число n, он мог бы тогда найти значения p и q, а следовательно, и число z. После этого числа e и d можно найти при помощи алгоритма Евклида. К счастью, математики пытались решить проблему разложения на множители больших чисел по меньшей мере 300 лет, и накопленный опыт позволяет предположить, что эта проблема чрезвычайно трудна.

Ривест (Rivest) с коллегами утверждает, что для разложения на множители числа из 500 цифр необходимо 10^{25} лет, если применять грубую силу. Предполагается, что задействованы лучший известный алгоритм и компьютер, выполняющий одну инструкцию за 1 мкс. Даже при сохранении экспоненциального роста скоростей компьютеров потребуются века, чтобы найти множители числа из 500 цифр, а к этому времени наши потомки могут просто выбрать еще большие p и q.

Тривиальный учебный пример работы алгоритма RSA приведен на рис. 8.14. Для этого примера мы выбрали $p = 3$, а $q = 11$, что дает значения $n = 33$, а $z = 20$. Число d можно выбрать равным 7, так как числа 20 и 7 не имеют общих делителей. При таком выборе значение e можно найти, решив уравнение $7e = 1 \pmod{20}$, откуда следует, что $e = 3$. Зашифрованный текст C получается из открытого сообщения P по формуле $C = P^3 \pmod{33}$. Получатель расшифровывает сообщение по формуле $P = C^7 \pmod{33}$. В качестве примера на рисунке показано шифрование слова «SUZANNE».

Открытый текст (P)		Зашифрованный текст (C)			После дешифрации	
Символ	Число	P^3	$P^3 \pmod{33}$	C^7	$C^7 \pmod{33}$	Символ
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	1	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	5	E
		Вычисление отправителя		Вычисление получателя		

Рис. 8.14. Пример работы алгоритма RSA

Поскольку выбранные для данного примера простые числа так малы, P должно быть менее 33, поэтому каждый блок открытого текста может содержать лишь одну букву. В результате получается моноалфавитный подстановочный шифр, что не очень впечатляет. Если бы мы вместо этого выбрали числа p и q порядка 2^{512}, тогда число n было бы около 2^{1024}. В этом случае каждый блок мог бы содержать до 1024 бит, или 128 восьмиразрядных символов, против 8 символов шифра DES или 16 AES.

Следует отметить, что использование алгоритма RSA в описанном ранее виде аналогично использованию симметричного алгоритма в режиме ECB (Electronic Code Book — электронный шифроблокнот), в котором одинаковые блоки на входе преобразуются в одинаковые блоки на выходе. Таким образом, для шифрования данных требуется сцепление блоков в каком-либо виде. Однако на практике алгоритм RSA с открытым ключом используется только для передачи одноразового секретного ключа, после чего применяется какой-нибудь алгоритм с симметричным ключом типа AES или тройного DES. Система RSA слишком медленная, чтобы шифровать большие объемы данных, однако она широко применяется для распространения ключей.

Другие алгоритмы с открытым ключом

Хотя алгоритм RSA получил широкое распространение, он ни в коей мере не является единственным известным алгоритмом с открытым ключом. Первым алгоритмом с открытым ключом стал «алгоритм ранца» (Merkle и Hellman, 1978). Его идея состоит в том, что имеется большое количество объектов различного веса. Владелец этих объектов кодирует сообщение, выбирая подмножество объектов и помещая их в ранец. Общий вес объектов в рюкзаке известен всем, как и список всех возможных объектов. Список объектов, находящихся в рюкзаке, хранится в секрете. При определенных дополнительных ограничениях, задача определения возможного списка объектов по известному общему весу считалась неразрешимой для вычисления, то есть считалось, что решение можно найти только полным перебором различных сочетаний предметов списка. Поэтому она была положена в основу алгоритма с открытым ключом.

Изобретатель алгоритма Ральф Меркле (Ralph Merkle) был настолько уверен в надежности своего алгоритма, что предложил 100 долларов любому, кто сумеет его взломать. Ади Шамир (Adi Shamir), «S» в группе RSA, мгновенно взломал его и получил награду. Это не смутило Меркле. Он усилил алгоритм и предложил за его взлом уже 1000 долларов. Рон Ривест (Ron Rivest), «R» в RSA, тут же взломал улучшенную версию алгоритма и получил награду. Меркле не рискнул предложить 10 000 долларов за следующую версию, поэтому «A», Леонарду Эйдлману (Leonard Adleman), не повезло. Несмотря на то, что алгоритм ранца был в очередной раз исправлен, он не считается надежным и редко используется.

Другие схемы с открытым ключом основаны на сложности вычисления дискретных логарифмов. Алгоритмы, использующие этот принцип, были разработаны Эль-Гамалем (El Gamal, 1985) и Шнорром (Schnorr, 1991).

Существуют и некоторые другие методы, например, основанные на эллиптических кривых (Menezes и Vanstone, 1993). Однако две основные категории составляют алгоритмы, основанные на сложности нахождения делителей больших чисел и вычислений дискретных логарифмов. Эти задачи считаются особенно сложными, так как математики уже много лет пытаются их решить без особых успехов.

Цифровые подписи

Подлинность различных бумажных документов (юридических, финансовых и др.) определяется наличием или отсутствием авторизованной рукописной подписи. Фотокопии документами не считаются. Чтобы системы компьютерных сообщений могли заменить физическое перемещение документов, написанных чернилами на бумаге, нужно решить проблему подписи.

Проблема разработки замены рукописной подписи довольно сложна. По существу, требуется система, с помощью которой одна сторона могла бы послать другой стороне «подписанное» сообщение так, чтобы:

- получатель мог проверить объявленную личность отправителя;
- отправитель не мог позднее отрицать содержимое сообщения;
- получатель не мог позднее изменить подписанное сообщение.

Первое требование существенно, например, для финансовых систем. Когда компьютер клиента заказывает компьютеру банка купить тонну золота, банковский компьютер должен быть уверен, что компьютер, пославший заказ, действительно принадлежит компании, со счета которой следует снять денежную сумму. Другими словами, банк должен иметь возможность установить подлинность клиента (а клиент — подлинность банка).

Второе требование необходимо для защиты банка от мошенничества. Предположим, что банк покупает тонну золота, и сразу после этого цена на золото резко падает. Бесчестный клиент может подать в суд на банк, заявляя, что он никогда не посылал заказа на покупку золота. Банк может показать сообщение в суде, но клиент будет отрицать, что посылал его. Таким образом, должно быть обеспечено требование **невозможности отречения от данных ранее обязательств**. Методы составления цифровых подписей, которые мы изучим далее, предназначены в том числе и для этого.

Третье требование нужно для защиты клиента в случае, если цена на золото после его покупки банком взлетает вверх и банк пытается создать подписанное сообщение, в котором клиент просит купить не одну тонну золота, а один слиток. При таком сценарии банк, совершив мошенничество, забирает оставшуюся часть золота себе.

Подписи с симметричным ключом

Один из методов реализации цифровых подписей состоит в создании некоего центрального авторитетного органа, которому все доверяют, — назовем его, например, Большим Братом (Big Brother, BB). Затем каждый пользователь выбирает секретный ключ и лично относит его в офис Большого Брата. Таким образом, например, секретный ключ Алисы, K_A, известен только Алисе и Большому Брату.

Когда Алиса хочет послать открытым текстом своему банкиру Бобу подписанное сообщение P, она формирует сообщение, зашифрованное K_A (ключом

Алисы), $K_A(B, R_A, t, P)$, где B — идентификатор Боба, R_A — случайное число, выбранное Алисой, t — временной штамп, подтверждающий свежесть сообщения. Затем она посылает его Большому Брату, как показано на рис. 8.15. Большой Брат видит, что это сообщение от Алисы, расшифровывает его и посылает Бобу. Сообщение, посылаемое Бобу, содержит открытый текст сообщения Алисы и подпись Большого Брата $K_{BB}(A, t, P)$. Получив подписанное сообщение, Боб может выполнять заказ Алисы.

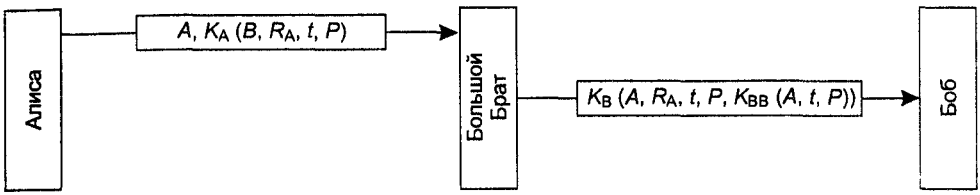


Рис. 8.15. Цифровая подпись Большого Брата

Что случится, если позднее Алиса станет отрицать отправку этого сообщения? Естественно, в случае возникновения подобных конфликтов конфликтующие стороны первым делом подают друг на друга в суд (по крайней мере, в Соединенных Штатах). Наконец, когда дело попадает в суд, Алиса энергично утверждает, что она не отправляла Бобу обсуждаемое. Судья спрашивает Боба, почему он уверен, что данное сообщение пришло именно от Алисы, а не от злоумышленницы Труди. Боб сначала заявляет, что Большой Брат не принял бы сообщения от Алисы, если бы оно не было зашифровано ее ключом K_A, — у злоумышленника просто нет возможности послать Большому Брату фальшивое сообщение от Алисы.

Затем Боб элегантным жестом демонстрирует суду сообщение A: $K_{BB}(A, t, P)$. Боб заявляет, что это сообщение подписано Большим Братом, что доказывает, что Алиса послала сообщение P Бобу. Затем судья просит Большого Брата (которому все доверяют) проверить подпись под этим сообщением. Когда Большой Брат подтверждает, что Боб говорит правду, судья решает дело в пользу Боба. Дело закрыто.

Теоретически проблема с этим протоколом цифровых подписей, который показан на рис. 8.15, может возникнуть, если злоумышленник повторно воспроизведет оба сообщения. Для минимизации вероятности этого используются временные штампы. Кроме того, Боб может просмотреть все недавние сообщения, проверяя, не встречалось ли в них такое же R_A. В этом случае сообщение считается дубликатом и просто игнорируется. Очень старые сообщения, обнаруживаемые по значению временного штампа, также игнорируются. Для защиты от мгновенной атаки повторным воспроизведением Боб просто проверяет значение случайного числа R_A, содержащегося в каждом приходящем сообщении, запоминая все такие числа, полученные за последний час. Если в течение часа такое значение получено еще не было, Боб может быть уверен, что пришедшее сообщение является новым заказом.

Подписи с открытым ключом

Главная проблема, связанная с применением шифрования с симметричным ключом для цифровых подписей, состоит в том, что все должны согласиться доверять Большому Брату. Кроме того, Большой Брат получает возможность читать все подписываемые им сообщения. Наиболее логичными кандидатами на управление сервером Большого Брата являются правительство, банки или нотариальные бюро. Однако эти организации вызывают доверие не у всех граждан. Таким образом, было бы гораздо лучше, если бы для получения подписи на электронном документе не требовалась авторитетная доверительная организация.

К счастью, здесь может помочь шифрование с открытым ключом. Предположим, что алгоритмы шифрования и дешифрации с открытым ключом, помимо обычного свойства $D(E(P)) = P$, обладают свойством $E(D(P)) = P$. Таким свойством, например, обладает алгоритм RSA, поэтому такое предположение не является голословным. В этом случае Алиса может послать Бобу подписанное открытое сообщение P, переслав ему $E_B(D_A(P))$. Обратите внимание на то, что Алиса знает свой собственный (закрытый) ключ дешифрации D_A, так же как и открытый ключ Боба E_B, так что сформировать такое сообщение ей по силам.

Получив это сообщение, Боб расшифровывает его как обычно, используя свой закрытый ключ D_B и получая в результате $D_A(P)$, как показано на рис. 8.16. Он сохраняет этот зашифрованный текст в надежном месте, после чего расшифровывает его открытым ключом шифрования Алисы E_A, получая открытый текст.

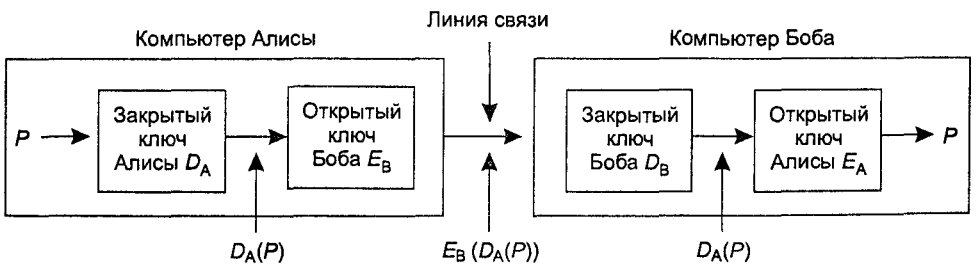


Рис. 8.16. Цифровая подпись, полученная при помощи шифрования с открытым ключом

Чтобы понять, как работает цифровая подпись в данном случае, предположим, что Алиса впоследствии отрицает, что посылала Бобу сообщение P. Когда дело доходит до суда, Боб предъявляет суду P и $D_A(P)$. Судья легко может убедиться, что у Боба есть действительное сообщение, зашифрованное ключом D_A, просто применив к нему ключ E_A. Боб не знает закрытого ключа Алисы, следовательно, получить зашифрованное этим ключом сообщение он мог только от Алисы. Сидя в тюрьме за лжесвидетельство и мошенничество, Алиса сможет заняться разработкой новых интересных алгоритмов с открытым ключом.

Хотя схема использования шифрования с открытым ключом довольно элегантна, она обладает серьезными недостатками, связанными, правда, скорее не с самим алгоритмом, а со средой, в которой ему приходится работать. Во-первых, Боб может доказать, что это сообщение было послано Алисой, только пока

ключ D_A остается секретным. Если Алиса раскроет свой секретный ключ, этот аргумент перестанет быть убедительным, так как послать сообщение мог кто угодно, включая самого Боба.

Проблема может возникнуть, если Боб, например, является биржевым брокером Алисы. Алиса заказывает Бобу купить некоторое количество акций. Сразу после этого цена акций резко падает. Чтобы отречься от своего сообщения, посланного Бобу, Алиса заявляет в полицию, что ее дом был обворован, а компьютер вместе с секретным ключом украден. В зависимости от законов ее страны или штата она может быть признана или не признана ответственной перед законом, особенно если она заявляет, что обнаружила, что ее квартира взломана, только через несколько часов после возвращения с работы.

Другая проблема данной схемы цифровой подписи возникает в случае, если Алиса решит сменить свой ключ. Подобное действие абсолютно законно, более того, рекомендуется периодически менять ключ, чтобы гарантировать его высокую надежность. В этом случае, если дело дойдет до судебного разбирательства, судья попытается применить к подписи $D_A(P)$ *текущий* ключ E_A и обнаружит, что в результате не получается сообщение P. При этом Боб будет выглядеть довольно глупо.

В принципе, для цифровых подписей можно использовать любой алгоритм с открытым ключом. Алгоритм RSA, фактически, стал промышленным стандартом. Он применяется во многих программах, предназначенных для обеспечения безопасности. Однако в 1991 г. Национальный институт стандартов и технологий США NIST (National Institute of Standards and Technology) предложил использовать для нового **стандарта цифровой подписи DSS** (Digital Signature Standard) вариант алгоритма с открытым ключом Эль-Гамаля, основанный не на трудности разложения больших чисел на множители, а на сложности вычисления дискретных алгоритмов.

Как обычно, попытка правительства навязать новые криптографические стандарты вызвала много шума. Стандарт DSS критиковали за то, что он:

- слишком засекречен (протокол, использующий алгоритм Эль-Гамаля, разрабатывался Агентством национальной безопасности США);
- слишком медленный (от 10 до 40 раз медленнее алгоритма RSA для проверки подписей);
- слишком новый (алгоритм Эль-Гамаля еще не был достаточно тщательно проверен);
- слишком ненадежен (фиксированный 512-разрядный ключ).

При последующей переработке четвертый пункт претензий стал спорным, так как было разрешено использовать ключи длиной до 1024 разрядов. Однако первые два пункта актуальны и по сей день.

Профили сообщений

Многие методы цифровых подписей критикуются за то, что в них совмещаются две различные функции: аутентификация и секретность. Довольно часто требует-

ся только аутентификация. К тому же, например, получить лицензию на экспорт обычно проще, если система обеспечивает только аутентификацию, но не секретность. Далее будет описана схема аутентификации, не требующая шифрования всего сообщения.

Эта схема основана на идее необратимой хэш-функции, которая принимает на входе участок открытого текста произвольной длины и по нему вычисляет строку битов фиксированной длины. У этой хэш-функции, часто называемой **профилем сообщения** (message digest, MD), есть четыре следующих важных свойства:

1. По заданному открытому тексту P легко сосчитать значение хэш-функции $MD(P)$.
2. По цифровой подписи $MD(P)$ практически невозможно определить значение открытого текста P.
3. Для данного P практически невозможно подобрать такой P', чтобы выполнялось равенство $MD(P') = MD(P)$.
4. Изменение даже одного бита входной последовательности приводит к очень непохожему результату.

Чтобы удовлетворять требованию 3, результат хэш-функции должен быть длиной, по крайней мере, в 128 бит, желательно даже больше. Чтобы удовлетворять требованию 4, хэш-функция должна искажать входные значения очень сильно. Этим данный метод напоминает алгоритмы с симметричными ключами, которые мы рассматривали ранее.

Профиль сообщения по части открытого текста вычисляется значительно быстрее, чем шифруется все сообщение с помощью алгоритма с открытым ключом. Поэтому профили сообщений могут использоваться для ускорения работы алгоритмов цифровых подписей. Чтобы понять, как все это работает, рассмотрим снова протокол передачи цифровой подписи, показанный на рис. 8.15. Вместо того чтобы посылать открытый текст P вместе с $K_{BB}(A, t, P)$, Большой Брат теперь вычисляет профиль сообщения $MD(P)$, применяя функцию хэширования MD к открытому тексту P. Затем он помещает $K_{BB}(A, t, MD(P))$ как пятый элемент в список, который зашифровывает ключом K_B, и отправляет его Бобу вместо $K_{BB}(A, t, P)$.

В случае возникновения спора Боб может предъявить на суде как открытый текст P, так и $K_{BB}(A, t, MD(P))$. По просьбе судьи Большой Брат расшифровывает $K_{BB}(A, t, MD(P))$, в результате чего суду предъявляются также цифровая подпись $MD(P)$, подлинность которой гарантируется Большим Братом, и сам открытый текст P, подлинность которого суд должен выяснить. Поскольку практически невозможно создать другой открытый текст, соответствующий данной цифровой подписи, суд убеждается в том, что Боб говорит правду. Использование профиля сообщения экономит время шифрования и затраты на транспортировку и хранение.

Профиль сообщения может также применяться для гарантии сохранности сообщения при передаче его по сети в системах шифрования с открытым ключом, как показано на рис. 8.17. Здесь Алиса сначала вычисляет профиль сообщения

для своего открытого текста. Затем она подписывает профиль сообщения и посылает зашифрованный профиль сообщения и открытый текст Бобу. Если злоумышленник попытается подменить по дороге открытый текст *P*, Боб обнаружит это, сосчитав значение профиля сообщения *MD*(*P*).

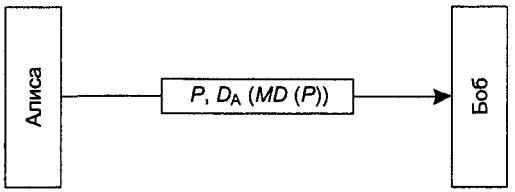


Рис. 8.17. Цифровая подпись с использованием профиля сообщения

MD5

Было предложено несколько вариантов функций, вычисляющих профиль сообщения. Самое широкое распространение получили алгоритмы MD5 (Rivest, 1992) и SHA (NIST, 1993). Алгоритм **MD5** (Message Digest 5 — профиль сообщения 5) представляет собой пятую версию хэш-функций, разработанных Рональдом Ривестом (Ronald Rivest). Он перемешивает входные биты достаточно сложным образом, так что каждый выходной бит зависит от каждого входного бита. Сначала сообщение дополняется до длины 448 бит по модулю 512. Затем к нему добавляется исходная длина сообщения, рассматриваемая как 64-разрядное число, в результате чего получается блок битов, длина которого кратна 512. Последний шаг подготовки к вычислениям инициализирует 128-разрядный буфер, задавая его содержимое равным некоему фиксированному значению.

Затем начинаются вычисления. На каждом этапе берется 512-разрядный блок входного текста и тщательно перемешивается со 128-разрядным буфером. Для пущей навористости в кастрюлю также кидается содержимое таблицы синусов. Именно синусы используются не потому, что их результат более случаен, чем результат других генераторов случайных чисел (в которых часто также применяются тригонометрические функции), а чтобы избежать каких бы то ни было подозрений в создании потайной лазейки, через которую потом разработчик (или заказчик) мог бы войти. Отказ корпорации IBM раскрыть принципы устройства S-блоков, применяемых в стандарте шифрования DES, привел к появлению большого количества слухов и домыслов о потайных ходах. Каждый входной блок обрабатывается за четыре итерации. Процесс продолжается, пока не будут обработаны все входные блоки. Содержимое 128-разрядного буфера и образует профиль сообщения.

MD5 появился около десяти лет назад, и за это время было предпринято множество атак на этот алгоритм. Были обнаружены некоторые слабые места, однако существуют определенные внутренние процедуры, позволяющие защититься от взлома. И все же, если упадут и эти последние барьеры, в один прекрасный день MD5 может оказаться ненадежным. Несмотря на это на момент написания книги этот алгоритм еще держится на плаву.

SHA-1

Второй широко применяемой функцией вычисления профиля сообщения является **SHA** (Secure Hash Algorithm — надежный алгоритм хэширования), разработанный Агентством национальной безопасности США (NSA) и получивший благословение национального института стандартов и технологий NIST (выразившееся в федеральном стандарте FIPS 180-1). Как и MD5, алгоритм SHA обрабатывает входные данные 512-битовыми блоками, но, в отличие от MD5, он формирует 160-разрядный профиль сообщения. Типичный случай отправки Алисой несекретного, но подписанного сообщения Бобу показан на рис. 8.18. Открытый текст обрабатывается алгоритмом SHA-1, на выходе получается 160-битный хэш SHA-1. Он подписывается Алисой (закрытым ключом RSA) и отправляется вместе с открытым текстом Бобу.

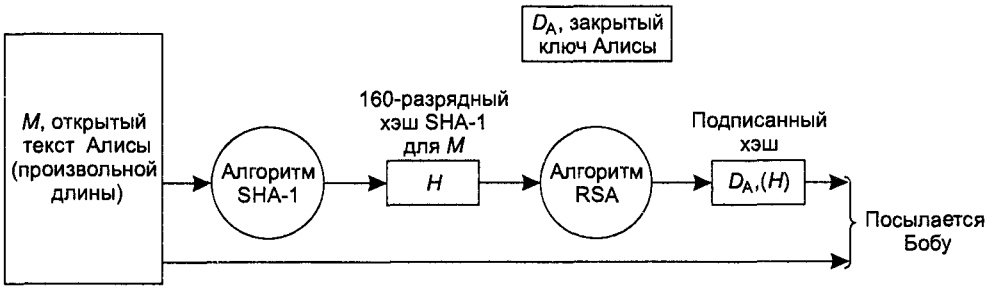


Рис. 8.18. Применение SHA-1 и RSA для создания подписей несекретных сообщений

При получении сообщения Боб сам вычисляет хэш-функцию с помощью алгоритма SHA-1 и применяет открытый ключ Алисы к подписанному хэшу для того, чтобы получить исходный хэш, *H*. Если они совпадают, сообщение считается корректным. Так как Труди не может, перехватив сообщение, изменить его таким образом, чтобы значение *H* совпадало с контрольным, Боб легко узнает обо всех подменах, которые совершила Труди. Для сообщений, чья неприкосновенность существенна, а секретность не имеет значения, часто применяется схема, показанная на рис. 8.18. При относительно небольших затратах на вычисления она гарантирует, что любые изменения, внесенные на пути следования сообщения, будут с высокой вероятностью выявлены.

Давайте теперь вкратце рассмотрим, как работает SHA-1. Для начала алгоритм SHA-1 также дополняет сообщение единичным битом в конце, за которым следует такое количество нулевых бит, чтобы в итоге получилось общее число битов, кратное 512. Затем 64-разрядное число, содержащее длину сообщения (до битового дополнения), логически складывается (операция ИЛИ) с 64 младшими битами. На рис. 8.19, *а* показано сообщение с дополнением, расположенным справа, потому что английский текст и рисунки читаются слева направо (то есть правая граница рисунка воспринимается как его конец, а левая — как начало). Применительно к вычислительной технике такое расположение соответствует обратному порядку хранения байтов (сначала передается самый значимый, старший бит).

Такая реализация присуща, например, SPARC. Однако вне зависимости от используемой техники SHA-1 вставляет битовое дополнение в конец сообщения.

Во время выполнения вычислений SHA-1 работает с пятью 32-битными переменными (H_0 ... H_4), в которых накапливается значение хэш-функции. Они показаны на рис. 8.19, *б*. Их начальные значения — это постоянные величины, определенные стандартом.

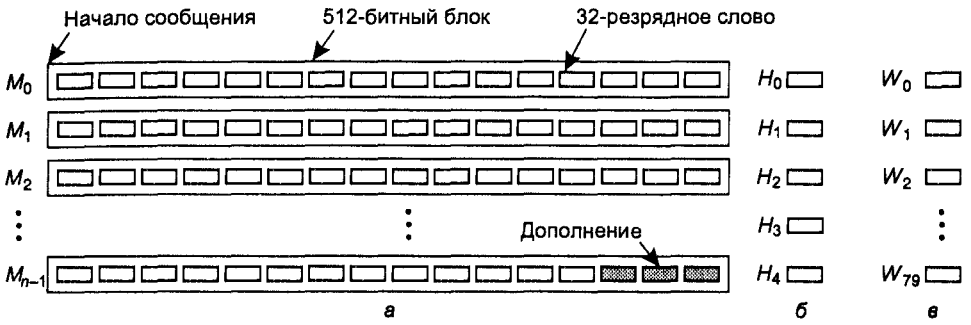


Рис. 8.19. Сообщение, дополненное до размера, кратного 512 битам (*а*); выходные переменные (*б*); массив слов (*в*)

Затем поочередно обрабатываются блоки с M_0 по M_{n-1}. Для текущего блока 16 слов сначала копируются в начало вспомогательного массива W размером 80 слов, как показано на рис. 8.19, *в*. 64 оставшихся слова вычисляются с использованием следующей формулы:

$$W_i = S^1(W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \qquad (16 \le i \le 79),$$

где $S^b(W)$ представляет собой поворот 32-разрядного слова W на b бит. Теперь по значениям H_0 ... H_4 инициализируются переменные от A до E.

Сами вычисления на псевдо-С можно записать таким образом:

```
for(i=0;i<80;i++){
temp = S5(A) + f1(B, C, D) + E + W1 + K1;
E=D; D=C; C=S30(B); B=A; A=temp;
}
```

где постоянные K_i определяются стандартом. Смешивающие функции f_i задаются следующим образом:

$$f_i(B, C, D) = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D) \qquad (0 \le i \le 19),$$

$$f_i(B, C, D) = B \text{ XOR } C \text{ XOR } D \qquad (20 \le i \le 39),$$

$$f_i(B, C, D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) \qquad (40 \le i \le 59),$$

$$f_i(B, C, D) = B \text{ XOR } C \text{ XOR } D \qquad (60 \le i \le 79).$$

После 80 итераций цикла значения переменных A ... E добавляются к H_0 ... H_4 соответственно.

После обработки первого 512-битного блока начинается обработка следующего. Массив W инициализируется заново с помощью нового блока, однако H сохраняется неизменным. По окончании этого блока обрабатывается следующий, и так далее, пока все 512-разрядные блоки сообщения не попадут в эту кастрюлю. После обработки последнего блока пять 32-разрядных слов в массиве H выводятся в качестве 160-битного значения криптографической хэш-функции. Полный код SHA-1 приведен в RFC 3174.

В настоящее время идет работа над новыми версиями SHA-1 с 256-, 384- и 512-разрядными значениями хэш-функций.

Задача о днях рождения

В мире шифров многое оказывается совсем не таким, каким кажется на первый взгляд. Можно, например, предполагать, что для ниспровержения профиля сообщения, состоящего из m разрядов, потребуется порядка 2^m операций. На самом деле, часто оказывается достаточно $2^{m/2}$ операций, если использовать метод, основанный на **задаче о днях рождения**, опубликованный в ставшей классической книге (Yuval, 1979).

В основе идеи этого метода лежит задача, часто приводимая в качестве примера профессорами математики на курсах по теории вероятности. Вопрос: сколько студентов должно находиться в классе, чтобы вероятность появления двух человек с совпадающими днями рождения превысила 1/2? Большинство студентов обычно ожидают, что ответ будет значительно больше 100. На самом же деле, теория вероятности утверждает, что это число равно 23. Не вдаваясь в тонкости анализа этой проблемы, дадим интуитивно понятное объяснение: из 23 человек мы можем сформировать $(23 \cdot 22)/2 = 253$ различных пары, у каждой из которой дни рождения могут совпасть с вероятностью 1/365. Теперь этот ответ уже не кажется таким удивительным.

В более общем случае, если имеется некое соответствие между n входами (людьми, сообщениями и т. д.) и k возможными выходами (днями рождения, профилями сообщений и т. д.), мы имеем $n(n-1)/2$ входных пар. Если $n(n-1)/2 > k$, то вероятность того, что будет хотя бы одно совпадение выхода при различных входах, довольно велика. Таким образом, вероятность существования двух сообщений с одинаковыми профилями велика уже при $n > \sqrt{k}$. Это означает, что 64-разрядный профиль сообщения можно с большой вероятностью взломать (то есть найти два различных сообщения с одинаковым профилем), перебрав 2^{32} сообщений.

Рассмотрим практический пример. На кафедре компьютерных наук Государственного университета появились вакансия и два кандидата на эту должность, Том и Дик. Том работает на факультете на два года дольше Дика, поэтому его кандидатура будет рассматриваться первой. Если он получит эту должность, значит, Дику не повезло. Том знает, что заведующая кафедрой Мэрилин высоко ценит его работу, поэтому он просит ее написать для него рекомендательное письмо декану факультета, который будет решать дело Тома. После отправки все письма становятся конфиденциальными.

Мэрилин просит написать это письмо декану свою секретаршу Элен, подчеркивая, что она хотела бы видеть в этом письме. Когда письмо готово, Мэрилин просматривает его, подписывает 64-разрядной подписью и посылает декану. Позднее Элен может послать это письмо электронной почтой.

К несчастью для Тома, у Элен роман с Диком, и она хочет обмануть Тома. Поэтому она пишет следующее письмо с 32 вариантами в квадратных скобках.

Уважаемый господин декан,

Это [*письмо* | *обращение*] отражает мое [*искреннее* | *откровенное*] мнение о проф. Томе Уилсоне, являющемся [*кандидатом* | *претендентом*] на профессорскую должность [*в настоящее время* | *в этом году*]. Я [*знакома* | *работала*] с проф. Уилсоном в течение [*почти* | *около*] шести лет. Он является [*выдающимся* | *блестящим*] исследователем, обладающим [*большим талантом* | *большими возможностями*] и известным [*во всем мире* | *не только в нашей стране*] своим [*серьезным* | *созидательным*] подходом к [*большому числу* | *широкому спектру*] [*сложных* | *перспективных*] вопросов.

Он также является [*высоко* | *весьма*] [*уважаемым* | *ценимым*] [*преподавателем* | *педагогом*]. Его студенты дают его [*занятиям* | *лекциям*] [*самые высокие* | *высочайшие*] оценки. Он самый [*популярный* | *любимый*] [*преподаватель* | *учитель*] [*нашей кафедры* | *нашего университета*].

[*Кроме* | *Помимо*] того, [*гранты* | *контракты*] проф. Уилсона [*существенно* | *значительно*] пополнили [*фонды* | *финансовые запасы*] нашей кафедры. Эти [*денежные* | *финансовые*] средства [*позволили нам* | *дали возможность*] [*выполнить* | *осуществить*] [*много* | *ряд*] [*важных* | *специальных*] программ, [*таких как* | *среди которых*], государственная программа 2000 года. Без этих средств было бы невозможным продолжение этой программы, такой [*важной* | *значительной*] для нас. Я настойчиво рекомендую вам предоставить ему эту должность.

К несчастью для Тома, закончив печатать это письмо, Элен тут же принимается за второе:

Уважаемый господин декан,

Это [*письмо* | *обращение*] отражает мое [*искреннее* | *откровенное*] [*мнение* | *суждение*] о проф. Томе Уилсоне, являющемся [*кандидатом* | *претендентом*] на профессорскую должность в [*настоящее время* | *этом году*]. Я [*знакома* | *работала*] с проф. Уилсоном в течение [*почти* | *около*] шести лет. Он является [*слабым* | *недостаточно талантливым*] [*исследователем* | *ученым*], почти не известным в той области науки, которой он занимается. В его работах практически не заметно понимания [*ключевых* | *главных*] [*проблем* | *вопросов*] современности.

[*Более* | *Кроме*] того, он также не является сколько-нибудь [*уважаемым* | *ценимым*] [*преподавателем* | *педагогом*]. Его студенты дают его [*занятиям* | *лекциям*] [*самые низкие* | *негативные*] оценки. Он самый непопулярный [*преподаватель* | *учитель*] нашей кафедры, [*славящийся* | *печально известный*] своей [*привычкой* | *склонностью*] [*высмеивать* |*ставить в неудобное положение*] студентов, осмелившихся задавать вопросы на его [*лекциях* | *занятиях*].

[*Кроме* | *Помимо*] того, [*гранты* | *контракты*] проф. Уилсона [*почти* | *практически*] не пополняют [*фондов* | *финансовых запасов*] нашей кафедры. Если не удастся быстро найти новый источник финансирования, [*мы будем вынуждены* |

нам придется] [*закрыть* | *прекратить*] [*много* | *ряд*] [*важных* | *специальных*] программ, [*таких как* | *среди которых*] государственная программа 2000 года. К сожалению, при таких [*условиях* | *обстоятельствах*] я не могу [*предлагать* | *рекомендовать*] его вам на эту должность.

Затем Элен заставляет свой компьютер сосчитать 2^{32} профиля сообщения для каждого варианта обоих писем, что занимает всю ночь. Есть шансы, что один профиль первого письма совпадет с одним из профилей второго письма. Если нет, она может добавить еще несколько вариантов слов и выражений в каждое письмо и попытаться еще раз за выходные. Предположим, что ей удалось найти такое совпадение. Назовем положительный отзыв письмом *A*, а отрицательный — письмом *B*.

Элен отправляет письмо *A* электронной почтой Мэрилин на утверждение. Мэрилин, конечно, утверждает письмо, вычисляет 64-разрядный профиль сообщения, подписывает профиль и посылает по почте подписанный профиль декану. Независимо Элен посылает декану письмо *B* (вместо письма *A*, которое следует отправить на самом деле).

Получив письмо и подписанный профиль, декан запускает алгоритм вычисления профиля сообщений для письма *B*, видит, что его профиль совпадает с тем, что ему прислала Мэрилин, и увольняет Тома. Он даже не может себе представить, что Элен удалось создать два письма с одинаковыми профилями сообщений и отправить ему совсем не тот вариант, который читала и подписала Мэрилин. (Вариант с хэппи-эндом, столь любимым американцами: Элен сообщает Дику о своих проделках. Потрясенный Дик порывает с ней. Элен в ярости бежит сознаваться во всем Мэрилин. Мэрилин звонит декану. В конце концов, Том получает профессуру.) При использовании алгоритма MD5 подобная атака шифра невозможна, так как даже если компьютер сможет вычислять по 1 млрд профилей в секунду, потребуется около 500 лет, чтобы перебрать по 2^{64} варианта для обоих писем, что все равно не даст 100-процентной гарантии совпадения. Конечно, если заставить параллельно работать 5000 компьютеров, вместо 500 лет потребуется 5 недель. В этом смысле SHA-1 подходит лучше (так как работает дольше).

Управление открытыми ключами

Криптография с использованием открытых ключей позволяет передавать секретные данные, не обладая общим ключом, а также создавать электронные подписи сообщений без необходимости привлечения третьей, доверительной стороны. Наконец, подписанные профили сообщений позволяют легко проверять целостность и аутентичность полученных данных.

Однако мы как-то слишком ловко обошли один вопрос: если Алиса и Боб друг друга не знают, как они смогут обменяться открытыми ключами перед началом общения? Вот, казалось бы, очевидное решение: выложить открытый ключ на веб-сайте. Однако так делать нельзя, и вот почему: представим себе, что Алиса хочет найти открытый ключ Боба на его веб-сайте. Каким образом она это

делает? Набирает URL сайта. Браузер ищет DNS-адрес домашней страницы Боба и посылает запрос *GET*, как показано на рис. 8.20. К сожалению, Труди в этот момент перехватывает запрос и посылает Алисе фальшивую страницу. В качестве таковой может выступать, к примеру, копия страницы Боба, на которой вместо его открытого ключа выложен открытый ключ Труди. После того как Алиса зашифрует свое первое сообщение с помощью E_T, Труди расшифрует его, прочтет, зашифрует с помощью открытого ключа Боба и перешлет сообщение Бобу, который даже не подозревает обо всех этих перипетиях. Но гораздо хуже то, что Труди может изменять сообщения перед повторной шифрацией и отправкой Бобу. Очевидно, нужен некий механизм секретного обмена открытыми ключами.

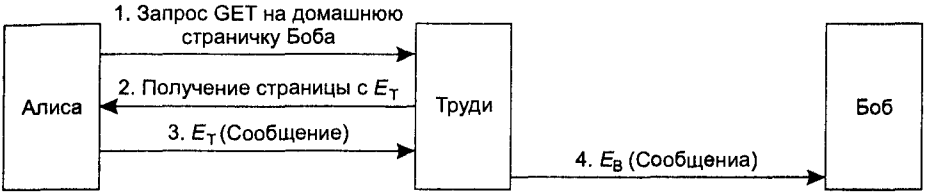


Рис. 8.20. Способ вторжения в систему с открытыми ключами

Сертификаты

Первая попытка организации защищенного обмена ключами может состоять в создании круглосуточного интернет-центра распространения ключей по требованию. Один из множества недостатков такого решения заключается в том, что данную систему не удастся масштабировать, и сам этот центр очень скоро станет узким местом. А если он не выдержит нагрузки, вся интернет-безопасность в один момент сойдет на нет.

По этой причине было придумано новое решение, не требующее, чтобы центр распространения ключей был доступен постоянно. В общем-то, даже не требуется, чтобы он вообще работал в подключенном (онлайновом) режиме. Вместо этого на него возлагается обязанность сертификации открытых ключей, принадлежащих как физическим, так и юридическим лицам. Организация, занимающаяся сертификацией открытых ключей, в настоящее время называется **Управлением сертификации** (CA — Certification Authority).

В качестве примера рассмотрим такую ситуацию: Боб хочет разрешить Алисе и другим лицам устанавливать с ним защищенные соединения. Он приходит в Управление сертификации со своим открытым ключом, а также паспортом или другим удостоверением личности и просит зарегистрировать ключ. Управление выдает ему сертификат, напоминающий тот, что показан на рис. 8.21, и подписывает хэш SHA-1 своим закрытым ключом. Затем Боб оплачивает услуги Управления и получает дискету с сертификатом и подписанным хэшем.

Основная задача сертификата состоит в связывании открытого ключа с именем принципала (физического или юридического лица). Сертификаты сами по себе никак не защищаются и не хранятся в тайне. Например, Боб может выло-

жить его на свой сайт и поставить ссылку: «Здесь можно посмотреть на сертификат моего открытого ключа». Перейдя по этой ссылке, пользователь увидит и сертификат, и блок с подписью (подписанный хэш SHA-1 сертификата).

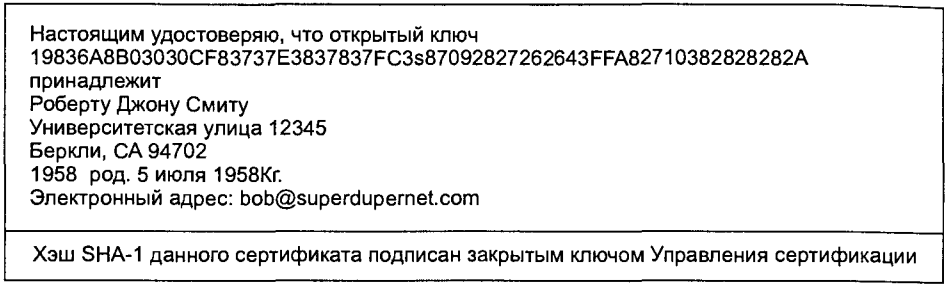


Рис. 8.21. Пример сертификата и подписанного хэша

Давайте теперь снова взглянем на сценарий, показанный на рис. 8.20. Что может сделать Труди, перехватив запрос страницы Боба, посланный Алисой? Она может выложить на странице свой собственный сертификат и блок с подписью на подложной странице, однако Алиса, читая этот сертификат, сразу догадается, что она разговаривает не с Бобом: в нем его имени просто нет. Труди может изменить домашнюю страницу Боба «на лету», заменив его открытый ключ своим собственным. И все же, проверив сертификат алгоритмом SHA-1, она получит значение хэш-функции, не согласующееся с тем, которое будет вычислено по открытому ключу Управления сертификации и блоку подписи. Так как Труди не имеет доступа к закрытому ключу Управления сертификации, она никак не может сгенерировать блок подписи, содержащий хэш модифицированной страницы с выложенным на ней открытым ключом Труди. Таким образом, Алиса может не беспокоиться о подлинности ключа Боба. И вот, как мы и обещали, при такой схеме не требуется, чтобы Управление сертификации постоянно работало в подключенном режиме. Тем самым ликвидируется потенциально узкое место системы.

Сертификат может связывать открытый ключ не только с принципалом, но и с **атрибутом**. Например, сертификат может содержать такую информацию: «Данный открытый ключ принадлежит лицу старше 18 лет». Этим можно подтвердить статус принципала или убедить окружающих в том, что ему разрешен доступ к некоторым специфическим данным, на которые накладываются возрастные ограничения. При этом имя принципала может и не раскрываться. Обычно владелец сертификата отсылает его на веб-сайт, принципалу или тому процессу, который обеспокоен возрастом клиента. В ответ генерируется случайное число, шифруемое с помощью открытого ключа (считываемого с сертификата). Если клиент сможет расшифровать его и отослать обратно, это будет служить подтверждением того, что он действительно обладает указанными в сертификате характеристиками. Еще это случайное число может быть использовано в качестве сеансового ключа для будущего соединения.

Сертификат может содержать атрибут еще в одном случае: если речь идет об объектно-ориентированной распределенной системе. Каждый объект обычно обладает некоторым набором методов. Владелец объекта может предоставлять каждому клиенту сертификат с указанием тех методов, которыми он может пользоваться. Этот список в виде поразрядной карты отображения информации (битовой карты) можно связать с открытым ключом, используя подписанный сертификат. Опять же, если владелец сертификата сможет подтвердить факт обладания соответствующим закрытым ключом, он сможет воспользоваться методами, указанными в списке. Особенностью такого использования сертификатов является отсутствие необходимости указывать имя владельца. Это бывает полезно в ситуациях, когда важна конфиденциальность.

X.509

Если бы все желающие подписать что-нибудь обращались в Управление идентификации, обслуживание клиентов с разного рода документами, требующими подписи, вскоре стало бы проблемой. Во избежание этого был разработан и утвержден организацией ITU специальный стандарт сертификатов. Он называется **X.509** и широко применяется в Интернете. В свет, начиная с 1988 года, вышло уже три версии стандарта, и мы будем рассматривать новейшую из них — третью.

На стандарт X.509 сильное влияние оказал мир OSI, в связи с чем в нем появились некоторые неприятные вещи, как, например, определенный принцип кодирования и именования. Удивительно, что проблемная группа по развитию Интернета, IETF, согласилась с данной концепцией, особенно учитывая то, что практически во всех областях, начиная с машинных адресов и заканчивая транспортными протоколами и форматами электронных писем, IETF всегда игнорировала OSI и пыталась сделать что-нибудь более толковое. IETF-версия X.509 описана в RFC 3280.

По сути, X.509 — это способ описания сертификатов. Основные поля сертификата перечислены в табл 8.3. Из описаний, приведенных в правой колонке, должно быть понятно, что для чего служит поле. За дополнительной информацией обращайтесь к RFC 2459.

Например, если Боб работает в отделе ссуд банка Money Bank, его X.500-адрес будет выглядеть так:

```
/C=US/O=MoneyBank/OU=Loan/CN=Bob/
```

где `C` — страна, `O` — организация, `OU` — отдел организации, `CN` — имя. Управление сертификации и другие сущности именуются похожим образом. Существенная проблема с системой именования X.500 заключается в том, что если Алиса пытается соединиться с *bob@moneybank.com* и имеет данные сертификата с именем в этом формате, для нее вовсе не очевидно, что этот сертификат имеет отношение именно к тому Бобу, который ей нужен. К счастью, начиная с третьей версии, разрешено использование имен DNS вместо X.500, поэтому данная проблема может счастливо разрешиться.

Сертификаты шифруются с использованием **системы записи абстрактного синтаксиса 1 (ASN** — Abstract Syntax Notation) OSI. Ее можно рассматривать

как нечто подобное структуре в языке C, за тем исключением, что эта запись очень странная и многословная. Более подробную информацию можно найти в (Ford и Baum, 2000).

Таблица 8.3. Основные поля сертификата в стандарте X.509

Поле	Значение
Version	Версия X.509
Serial number	Это число вместе с названием Управления сертификации однозначно идентифицирует сертификат
Signature algorithm	Алгоритм генерации подписи сертификата
Issuer	X.500-имя Управления
Validity period	Начало и конец периода годности
Subject name	Сущность, ключ которой сертифицируется
Public key	Открытый ключ сущности и идентификатор использующего его алгоритма
Issuer ID	Необязательный идентификатор, единственным образом определяющий эмитента (создателя) сертификата
Subject ID	Необязательный идентификатор, единственным образом определяющий владельца сертификата
Extensions	Различные возможные расширения
Signature	Подпись сертификата (генерируется с помощью закрытого ключа Управления сертификации)

Инфраструктуры систем с открытыми ключами

Понятно, что одного Управления сертификации на весь мир недостаточно. Оно бы быстро перестало функционировать из-за огромной нагрузки, да еще и стало бы эпицентром всех проблем, связанных с безопасностью сетей. Возможно, следует создать целый набор таких Управлений, использующих один и тот же закрытый ключ для подписания сертификатов, под покровительством одной и той же организации. Однако, хотя это и решит проблему распределения нагрузки, возникнет новый вопрос, связанный с *утечкой ключа*. Если по всему миру будут разбросаны десятки серверов, хранящих закрытый ключ Управления сертификации, велик шанс, что рано или поздно этот ключ будет украден или пропадет каким-то иным образом. Если ключ будет рассекречен, всю мировую инфраструктуру электронной безопасности можно будет похоронить. Вместе с тем, наличие всего одного центрального Управления сертификации — это тоже риск.

Далее, какая организация будет заведовать Управлением? Довольно трудно представить себе какую-либо законную структуру с большим кредитом доверия мирового масштаба. В некоторых странах предпочтительно, чтобы это было какое-нибудь правительственное учреждение, а где-то — наоборот, чтобы это было чем угодно, но не правительством.

По этим причинам был разработан альтернативный способ сертификации открытых ключей. Он известен под общим названием **PKI** (Public Key Infrastructure — инфраструктура систем с открытыми ключами). В этом разделе мы

рассмотрим только общие принципы действия PKI, поскольку было внесено множество предложений по ее модификации и некоторые детали могут со временем измениться.

PKI состоит из множества компонентов, среди которых Управления сертификации, сами сертификаты, а также каталоги. Инфраструктура систем с открытыми ключами предоставляет возможность структурной организации этих компонентов и определяет стандарты, касающиеся различных документов и протоколов. Одним из простейших видов PKI является иерархия Управлений, представленная на рис. 8.22. На рисунке представлены три уровня, однако реально их может быть как больше, так и меньше. Управление сертификации верхнего уровня (root) мы будем называть Центральным управлением (ЦУ). Центральное управление сертифицирует управления второго уровня — назовем их Региональными отделами (РО), — так как они могут обслуживать некоторый географический регион, например, страну или континент. Этот термин не стандартизован. Названия для уровней иерархии вообще не оговариваются стандартом. Региональные отделы, в свою очередь, занимаются легализацией реальных Управлений сертификации (УС), эмитирующих сертификаты стандарта X.509 для физических и юридических лиц. При легализации Центральным управлением нового Регионального отдела последнему выдается сертификат, подтверждающий его признание. Он содержит открытый ключ нового РО и подпись ЦУ. Аналогичным образом РО взаимодействуют с Управлениями сертификации: выдают и подписывают сертификаты, содержащие открытые ключи УС и признающие легальность деятельности.

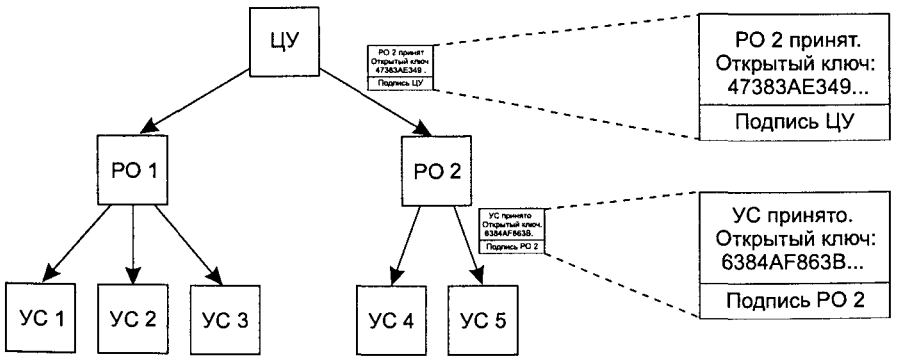


Рис. 8.22. Иерархия PKI (*а*); цепочка сертификатов (*б*)

Итак, наш PKI работает следующим образом. Допустим, Алисе нужен открытый ключ Боба, чтобы она могла с ним пообщаться. Она ищет и находит содержащий его сертификат, подписанный УС 5. Однако Алиса никогда ничего не слышала про УС 5. Этим «Управлением» может оказаться, на самом деле, десятилетняя дочка Боба. Алиса может отправиться в УС 5 и попросить подтвердить легитимность. Управление в ответ может показать сертификат, полученный от

РО 2 и содержащий открытый ключ УС 5. Теперь, вооружившись открытым ключом УС 5, Алиса может удостовериться в том, что сертификат Боба действительно подписан УС 5, а значит, является легальным.

Если только РО 2 не является двенадцатилетним сыном Боба. Если Алисе вдруг придет в голову такая мысль, она может запросить подтверждение легитимности РО 2. Ответом будет служить сертификат, подписанный Центральным управлением и содержащий открытый ключ РО 2. Вот теперь Алиса может не сомневаться, что она получила открытый ключ Боба, а не кого-то другого.

А если Алиса хочет узнать открытый ключ ЦУ? Как это сделать? Загадка. Предполагается, что открытый ключ ЦУ знают все. Например, он может быть «зашит» внутрь ее браузера.

Но наш Боб — добряк, он не хочет доставлять Алисе лишние хлопоты. Он понимает, что она захочет проверить легитимность УС 5 и РО 2, поэтому он сам собирает соответствующие сертификаты и отправляет их ей вместе со своим. Теперь, зная открытый ключ ЦУ, Алиса может проверить по цепочке все интересующие ее организации. Ей не придется никого беспокоить для подтверждения. Поскольку все сертификаты подписаны, она может запросто уличить любые попытки подлога. Цепочка сертификатов, восходящая к ЦУ, иногда называется **доверительной цепочкой** или **путем сертификата**. Описанный метод широко применяется на практике.

Конечно, остается проблема определения владельца ЦУ. Следует иметь не одно Центральное управление, а несколько, причем связать с каждым из них свою иерархию региональных отделов и управлений сертификации. На самом деле, в современные браузеры действительно «зашиваются» открытые ключи более 100 центральных управлений, иногда называемые **доверительными якорями**. Как видите, можно избежать проблемы одного всемирного учреждения, занимающегося сертификацией.

Однако встает вопрос, какие доверительные якоря производители браузеров могут считать надежными, а какие — нет. Все, на самом деле, сводится к тому, насколько конечный пользователь доверяет разработчику браузера, насколько он уверен в том, что решения генерируются грамотно и доверительные якоря не принимаются от всех желающих за умеренную плату. Большинство браузеров обеспечивают возможность проверки ключей ЦУ (обычно это делается с помощью сертификатов, подписанных им) и удаления подозрительных ключей.

Каталоги

Инфраструктура систем с открытыми ключами должна решать еще один вопрос. Он касается места хранения сертификатов (и цепочек, ведущих к какому-нибудь доверительному якорю). Можно заставить всех пользователей хранить свои сертификаты у себя. Это безопасно (так как невозможно подделать подписанные сертификаты незаметно), но не слишком удобно. В качестве каталога для сертификатов было предложено использовать DNS. Прежде чем соединиться с Бобом, Алисе, видимо, все равно придется узнать с помощью службы имен доменов (DNS) его IP-адрес. Так почему бы не заставить DNS возвращать вместе с IP-адресом всю цепочку сертификатов?

Кому-то это кажется выходом из положения, однако некоторые считают, что лучше иметь специализированные серверы с каталогами для хранения сертификатов X.509. Такие каталоги могли бы с помощью имен X.500 обеспечивать возможность поиска. Например, теоретически можно представить себе услугу сервера каталогов, позволяющую получать ответы на запросы типа «Дайте мне полный список всех людей по имени Алиса, работающих в отделе продаж в любом месте США или Канады». Хранить такую информацию можно, например, при помощи LDAP.

Аннулирование

Реальный мир полон разного рода сертификатов, среди которых, например, паспорта, водительские удостоверения. Иногда зти сертификаты необходимо аннулировать (например, водительское удостоверение надо аннулировать за езду в нетрезвом состоянии). Та же проблема возникает и в мире цифровых технологий: лицо, предоставившее сертификат, может отозвать его за нарушение противоположной стороной каких-либо условий. Это необходимо делать и тогда, когда закрытый ключ, в сущности, перестал быть защищенным или, что еще хуже, ключ УС потерял кредит доверия. Таким образом, инфраструктура систем с открытыми ключами должна как-то обеспечивать процедуру аннулирования.

Первым шагом в этом направлении является принуждение всех УС к периодическому выпуску **списка аннулированных сертификатов** (CRL — Certificate Revocation List). В нем перечисляются порядковые номера всех аннулированных сертификатов. Поскольку в сертификатах содержится дата окончания срока годности, в CRL следует включать номера только тех из них, срок годности которых еще *не* истек. По истечении срока годности сертификаты перестают быть действительными автоматически, поэтому нужно различать случаи аннулирования «по старости» и по другим причинам. В любом случае их использование необходимо запрещать.

К сожалению, возникновение списков аннулированных сертификатов означает, что лицо, собирающееся использовать сертификат, должно вначале убедиться в том, что его нет в этом списке. В противном случае от использования надо отказаться. Тем не менее, сертификат мог быть аннулирован тотчас же после выпуска самого свежего варианта черного списка. Получается, что единственный надежный способ — это узнать о состоянии сертификата непосредственно у УС. Причем эти запросы придется посылать при каждом использовании сертификата, так как нет никакой гарантии, что его аннулирование не произошло несколько секунд назад.

Еще больше усложняет ситуацию то, что аннулированный сертификат иногда требуется восстанавливать. Например, если причиной отзыва была неуплата каких-нибудь взносов, после внесения необходимой суммы не остается никаких причин, которые не позволяли бы восстановить сертификат. Обработка ситуаций аннулирования и восстановления сводят на нет такое ценное свойство сертификатов, как возможность их использования без помощи УС.

Где хранить списки аннулированных сертификатов? Было бы здорово хранить их там же, где и сами сертификаты. Одна из стратегий подразумевает, что

УС периодически выпускает «черные» списки и заставляет вносить обновления в каталоги (удаляя отозванные сертификаты). Если для хранения сертификатов каталоги не используются, можно кэшировать их в разных удобных местах в сети. Поскольку «черный» список сам по себе является подписанным документом, любые попытки подлога тотчас будут замечены.

Если сертификаты имеют большие сроки годности, списки аннулированных сертификатов также будут довольно длинными. Например, количество отозванных кредитных карточек со сроком годности 5 лет будет гораздо больше списка отозванных трехмесячных карточек. Стандартным способом борьбы с длинными списками является довольно редкий выпуск самих списков и частый — обновлений к ним. Кроме всего прочего, это помогает снизить необходимую для распространения списков пропускную способность.

Защита соединений

Мы закончили изучение прикладных инструментов. Были описаны большинство применяемых методов и протоколов. Оставшаяся часть главы будет посвящена применению этих методов на практике для обеспечения безопасности сетей. Кроме того, будут высказаны некоторые мысли относительно социального аспекта этого вопроса.

Следующие четыре раздела посвящены безопасности соединений, то есть тому, как секретно и без риска подмены данных передавать биты от пункта отправления до пункта назначения, а также тому, как не пускать на линию посторонние биты. Это ни в коем случае не полный список проблем сетевой безопасности, однако перечисленные вопросы являются одними из самых важных.

IPsec

Проблемная группа IETF в течение многих лет мирилась с отсутствием безопасности в Интернете. Обеспечить ее было действительно непросто, и прежде всего потому, что разгорелись жаркие споры вокруг того, какую часть Интернета следует, собственно, защищать. Большинство экспертов по вопросам безопасности уверены в том, что по-настоящему надежная система должна выполнять сквозную шифрацию и сквозное обеспечение целостности данных (то есть все это должно быть сделано на прикладном уровне). Это означает, что процесс-источник шифрует и/или ставит защиту целостности данных и отправляет их процессу-приемнику, который, соответственно, дешифрует данные и проверяет их целостность. Тогда можно будет заметить любые попытки взлома (даже на уровне операционной системы на любой из сторон). Беда такого подхода в том, что для обеспечения безопасности требуется вносить изменения во все приложения. Это означает, что необходимо «спустить» шифрацию на транспортный уровень или организовать новый специализированный подуровень между прикладным и транспортным уровнями. Он должен быть сквозным, но в то же время не требующим внесения изменений в приложения.

Противоположная точка зрения состоит в том, что пользователи все равно не осознают необходимости применения мер безопасности и просто не способны корректно использовать все предоставленные возможности. При этом никто не захочет каким-либо образом изменять существующие программы, поэтому сетевой уровень должен выполнять проверку подлинности и/или шифровать сообщения незаметно для пользователя. Долгие годы сражений привели к победе этой точки зрения: был разработан стандарт безопасности, ориентированный на сетевой уровень. Одним из аргументов было то, что шифрование на сетевом уровне, с одной стороны, не помешает тем пользователям, которые серьезно относятся к безопасности, и а с другой — в некоторой степени убережет беспечных пользователей.

Результатом всех этих дискуссий было создание стандарта **IPsec** (IP security — IP-безопасность), описанного в RFC 2401, 2402, 2406 и др. Не всем пользователям требуется шифрация соединений (выполнение соответствующих процедур может занимать существенную часть вычислительных ресурсов). Однако вместо того чтобы делать шифрацию необязательной, пользователю предлагается в случае необходимости выбирать пустой алгоритм. В RFC 2410 расписываются такие достоинства пустого алгоритма, как простота, легкость реализации и высокая скорость.

IPsec служит основой для множества услуг, алгоритмов и модулей. Причиной наличия множества услуг является то, что далеко не все хотят постоянно платить за все возможные услуги, поэтому нужные сервисы предоставляются порционно. Основные услуги таковы: секретность, целостность данных, защита от взлома методом повторения сообщений (когда жулик повторяет подслушанный разговор). Все это основано на криптографии с симметричными ключами, поскольку здесь критична высокая производительность.

Для чего нужен целый набор алгоритмов? Дело в том, что считающийся сегодня надежным алгоритм завтра может быть сломан. Если сделать IPsec независимым от конкретного алгоритма, стандарт выживет даже в случае взлома одного из алгоритмов.

Для чего нужны разные модули? Для того чтобы можно было защищать и одно TCP-соединение, и весь трафик между парой хостов, и весь трафик между парой защищенных хостов, и т. д.

Несколько удивительно, что IPsec ориентирован на соединение, несмотря на его присутствие на уровне IP. На самом деле, это не так странно, как кажется. Ведь безопасность можно обеспечить только созданием ключа и использованием его в течение какого-то времени. А это, по сути дела, разновидность соединения. К тому же все соединения погашают расходы на их установление за счет передачи большого количества пакетов. «Соединение» в контексте IPsec называется **защищающей связью** (security connection). Защищающая связь — это симплексное соединение между двумя конечными точками, с которым связан специальный идентификатор защиты. Если требуется передача защищенных данных в обоих направлениях, понадобятся две защищающие связи. Идентификаторы защиты передаются в пакетах, следующих по этим надежным соединениям, и использу-

ются по прибытии защищенных пакетов для поиска ключей и другой важной информации.

Технически IPsec состоит из двух основных частей. Первая описывает два новых заголовка, которые можно добавлять к пакету для передачи идентификатора защиты, данных контроля целостности и другой информации. Вторая часть, **ISAKMP** (Internet Security and Key Management Protocol — интернет-безопасность и протокол управления ключами), предназначена для создания ключей. Мы не станем вдаваться в подробности устройства ISAKMP, потому что, во-первых, это очень сложная тема и, во-вторых, основной протокол **IKE** (Internet Key Exchange — обмен ключами в Интернете) работает очень некорректно и требует замены (Perlman и Kaufman, 2000).

IPsec может работать в двух режимах. В **транспортном режиме** заголовок IPsec вставляется сразу за заголовком IP. Поле *Protocol* заголовка IP изменяется таким образом, чтобы было понятно, что далее следует заголовок IPsec (перед заголовком TCP). В заголовке IPsec содержится информация, касающаяся безопасности, — в частности, идентификатор защищающей связи, новый порядковый номер и, возможно, проверка целостности поля полезной нагрузки.

В **режиме туннелирования** весь IP-пакет вместе с заголовком вставляется внутрь нового IP-пакета с совершенно новым заголовком. Этот режим хорош тогда, когда туннель заканчивается где-нибудь вне конечного пункта. В некоторых случаях концом туннеля является шлюз, обеспечивающий безопасность, например, корпоративный брандмауэр. В этом режиме брандмауэр вставляет и извлекает пакеты, проходящие через него в разные стороны. При такой организации машины ЛВС компании гарантированно будут обслужены по стандарту IPsec. Об этом совершенно не приходится беспокоиться: все заботы берет на себя брандмауэр.

Еще режим туннелирования полезен, когда несколько TCP-соединений объединяются вместе и обрабатываются в виде единого шифрованного потока, поскольку в данном случае взломщик не может узнать, кто и кому передает пакеты, а также в каком количестве. А ведь иногда даже объем трафика, передаваемого одним лицом другому, является ценной информацией. Например, если во время военного кризиса трафик между Пентагоном и Белым домом резко снижается и при этом так же резко растет трафик между Пентагоном и какой-нибудь военной базой в Колорадо, перехватчик может сделать из этого далеко идущие выводы.

Изучение структуры потока по проходящим пакетам называется **анализом трафика**. Если используется туннелирование, такой анализ становится задачей весьма сложной. Недостаток режима туннелирования заключается в том, что приходится расширять заголовок IP-пакетов, за счет чего заметно возрастает суммарный размер пакетов. В транспортном режиме размер пакетов изменяется незначительно.

Первый из новых заголовков называется **заголовком идентификации** (AH — Authentication Header). С его помощью проверяется целостность данных и выполняется защита от взлома путем повторной передачи. Однако он не имеет никакого отношения к секретности (то есть шифрации данных). Применение AH в транспортном режиме показано на рис. 8.23. В стандарте IPv4 он располагается

между заголовком IP (вместе со всеми необязательными полями) и заголовком TCP. В IPv6 это просто еще один дополнительный заголовок. Так он и воспринимается. Формат AH действительно очень близок к формату дополнительного заголовка IPv6. К полезной нагрузке иногда добавляют заполнение, чтобы достичь определенной длины, необходимой алгоритму идентификации. Это показано на рисунке.

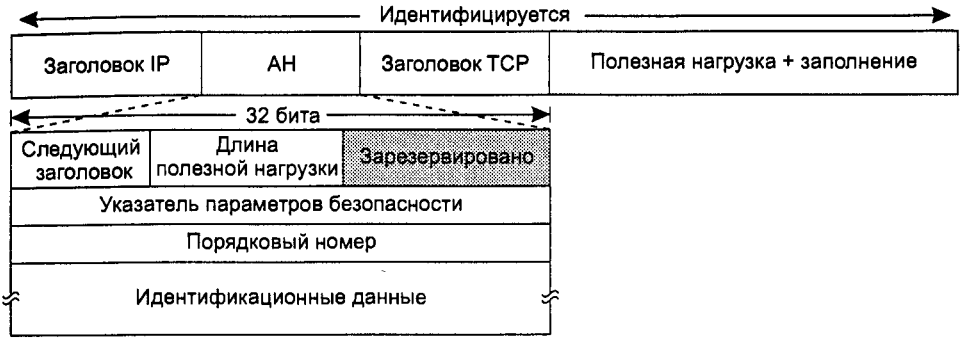


Рис. 8.23. Заголовок идентификации IPsec в транспортном режиме для IPv4

Рассмотрим заголовок AH. Поле *Следующий заголовок* хранит предыдущее значение, которое в поле *Протокол* заголовка IP ранее было заменено на 51, чтобы показать, что далее следует заголовок AH. Обычно здесь встречается код для TCP (6). Поле *Длина полезной нагрузки* хранит количество 32-разрядных слов заголовка AH минус 2.

Поле *Указатель параметров безопасности* — это идентификатор соединения. Он вставляется отправителем и ссылается на конкретную запись в базе данных у получателя. В этой записи содержится общий ключ и другая информация данного соединения. Если бы этот протокол был придуман ITU, а не IETF, это поле, скорее всего, называлось бы *Номером виртуального канала*.

Поле *Порядковый номер* применяется для нумерации всех пакетов, посылаемых по защищенной связи. Все пакеты получают уникальные номера, даже если они посылаются повторно. Имеется в виду, что повторно передаваемый пакет имеет номер, отличный от номера оригинального пакета (даже если порядковый номер TCP тот же самый). Это поле служит для предотвращения взлома путем повторной передачи. Порядковые номера никогда не повторяются. Если же окажутся использованными все 2^{32} номера, для продолжения общения устанавливается новая защищающая связь.

Наконец, поле переменной длины *Данные идентификации* содержит цифровую подпись, вычисляемую относительно полезной нагрузки. При установке защищающей связи стороны договариваются об используемом алгоритме генерации подписей. Чаще всего здесь не применяется шифрование с открытыми ключами, так как все известные алгоритмы этого типа работают слишком медленно, а пакеты необходимо обрабатывать с очень большой скоростью. Протокол IPsec основан на шифровании с симметричными ключами, поэтому перед уста-

новкой защищающей связи отправитель и получатель должны договориться о значении общего ключа, применяемого при вычислении подписи. Один из простейших способов заключается в вычислении хэш-функции для пакета и общего ключа. Отдельно общий ключ, конечно, не передается. Подобная схема называется **HMAC** (Hashed Message Authentication Code — код идентификации хэшированного сообщения). Вычисление этого кода выполняется гораздо быстрее, чем последовательный запуск SHA-1 и RSA.

Заголовок AH не позволяет шифровать данные. Его основная польза выявляется, когда важна проверка целостности, но не нужна секретность. Стоит отметить, что при проверке целостности при помощи AH охватываются некоторые поля заголовка IP, в частности, те из них, которые не изменяются при прохождении пакета от маршрутизатора к маршрутизатору. Поле *Время жизни*, например, меняется при каждой пересылке через маршрутизатор, поэтому его нельзя охватить при проверке целостности. Однако IP-адрес источника охватывается, тем самым предотвращается возможность его подмены взломщиком.

Альтернативой заголовку IPsec служит заголовок **ESP** (Encapsulating Security Payload — инкапсулированная защищенная полезная нагрузка). Как показано на рис. 8.24, этот заголовок может применяться как в транспортном режиме, так и в режиме туннелирования.

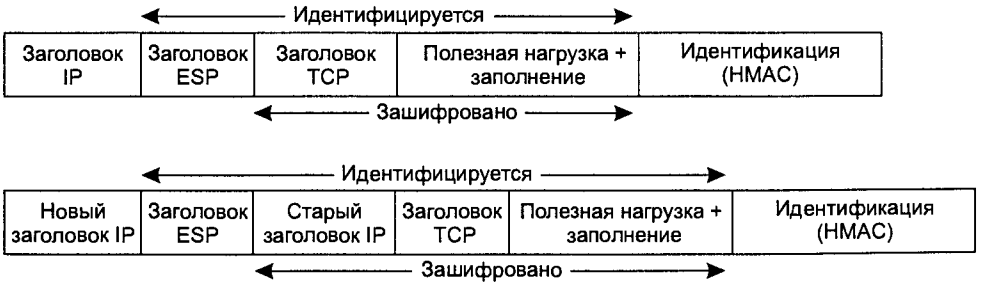


Рис. 8.24. ESP в транспортном режиме (*а*); ESP в режиме туннелирования (*б*)

Заголовок ESP состоит из двух 32-разрядных слов: *Указателя параметров безопасности* и *Порядкового номера*. Мы их уже встречали в заголовке AH. Третье слово, которое обычно следует за ними, однако технически не является частью заголовка, — это *Вектор инициализации* (если только не применяется пустой алгоритм шифрования, тогда это поле опускается).

ESP, как и AH, обеспечивает проверку целостности при помощи HMAC, однако вместо того, чтобы включать хэш в заголовок, его вставляют после поля полезной нагрузки. Это видно на рис. 8.24. Такое расположение полей дает преимущество при аппаратной реализации метода. Оно заключается в том, что HMAC может подсчитываться во время передачи битов полезной нагрузки по сети и добавляться к ним в конец. Именно поэтому в Ethernet и других стандартах локальных сетей циклический контроль избыточности вставляется в концевик, а не в заголовок. При применении заголовка AH пакет приходится буферизировать и вычислять подпись, только после его можно отправлять. Это потенциаль-

но приводит к уменьшению числа пакетов, которые можно передать за единицу времени.

Казалось бы, если ESP умеет делать все то же самое, что и AH, и даже больше, причем он еще и гораздо эффективнее, тогда зачем мучиться с AH? Причины этого в основном исторические. Изначально заголовок AH обеспечивал только проверку целостности, а ESP — только секретность. Позднее ESP научили использовать для проверки целостности, но разработчики AH не хотели, чтобы он канул в Лету после всей той работы, которую они проделали. Единственный аргумент в пользу AH заключается в том, что с его помощью можно частично проверять заголовок IP, чего не умеет ESP. И все же это аргумент довольно слабый. Еще один сомнительный аргумент состоит в том, что система, поддерживающая AH, но не поддерживающая ESP, возможно, будет иметь меньше проблем при получении лицензии на экспорт, поскольку этот заголовок не имеет отношения к секретности и шифрованию. Похоже, что AH со временем все-таки исчезнет с горизонта.

Брандмауэры

Возможность соединять любые компьютеры друг с другом в некоторых случаях является достоинством, а в других, наоборот, недостатком. Возможность бродить по Интернету доставляет много радости домашним пользователям. Менеджерам отдела безопасности корпораций эта возможность кажется кошмаром. Большинство компаний располагает огромными объемами конфиденциальной информации, размещенной на компьютерах, подключенных к сети, — коммерческие тайны, планы развития производства, рыночные стратегии, аналитические отчеты финансового состояния и т. д. Раскрытие этих сведений перед конкурентами может иметь ужасные последствия.

Помимо опасности утечки информации наружу, имеется опасность проникновения вредной информации, такой как вирусы, черви и прочей цифровой заразы, способной взламывать секреты, уничтожать ценные данные, на борьбу с которой уходит масса времени сетевых администраторов. Часто эту инфекцию заносят беззаботные сотрудники, желающие поиграть в новую модную компьютерную игру.

Таким образом, требуются специальные средства, удерживающие «доброкачественную» информацию внутри, а «вредную» — снаружи. Один из способов состоит в применении IPsec. Этот метод защищает данные при их пересылке. Однако шифрование не спасает от вирусов и хакеров, способных проникнуть в локальную сеть. Помочь защитить сети от нежелательного проникновения снаружи может установка брандмауэров, к рассмотрению которых мы сейчас обратимся.

Брандмауэры представляют собой современную реализацию средневекового принципа обеспечения безопасности. Они напоминают ров, вырытый вокруг замка. Суть конструкции заключается в том, что все входящие и выходящие из замка должны проходить по одному подъемному мосту, где полиция ввода-вывода сможет проверить их личность. Тот же принцип может быть применен и в сетях:

у компании может быть несколько локальных сетей, соединенных произвольным образом, но весь внешний трафик должен проходить через электронный подъемный мост (брандмауэр), как показано на рис. 8.25.

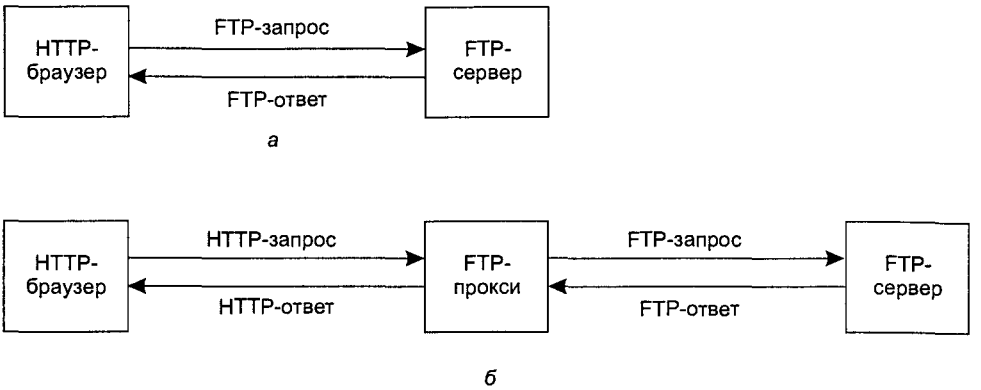


Рис. 8.25. Брандмауэр, состоящий из двух пакетных фильтров и шлюза прикладного уровня

Брандмауэр в данной конфигурации состоит из двух компонентов: двух маршрутизаторов, фильтрующих пакеты, и шлюза прикладного уровня. Существуют также и более простые конструкции, но преимущество такой разработки состоит в том, что каждый пакет, желающий войти или выйти, должен пройти через два фильтра и один шлюз прикладного уровня. Других путей нет. Читатели, полагающие, что достаточно одного контрольно-пропускного пункта, видимо, давно не летали международными авиалиниями.

Каждый **пакетный фильтр** представляет собой стандартный маршрутизатор с расширенными функциями, позволяющими анализировать входящие и выходящие пакеты. Пакеты, удовлетворяющие определенным критериям, пропускаются сквозь фильтр. Не сумевшие пройти проверку пакеты удаляются.

Показанный на рис. 8.25 пакетный фильтр внутренней локальной сети проверяет выходящие пакеты, а пакетный фильтр внешней локальной сети проверяет входящие пакеты. Пакеты, преодолевшие первый барьер, проходят к шлюзу прикладного уровня для дальнейшего исследования. Размещение двух фильтров в разных локальных сетях позволяет гарантировать, что ни один пакет не попадет из одной сети в другую, не пройдя через шлюз прикладного уровня. Обходного пути вокруг него нет.

Пакетные фильтры обычно управляются таблицами, настраиваемыми системным администратором. В этих таблицах перечислены допустимые и блокируемые отправители и получатели, а также правила, описывающие действия над исходящими и входящими пакетами.

В общем случае настроек TCP/IP информация о получателе или отправителе состоит из IP-адреса и номера порта. Номера портов определяют требуемую службу. Например, порт 23 используется для программы Telnet, порт 79 — для Finger, а порт 119 — для новостей сети USENET. Компания может заблокировать все входящие пакеты для комбинаций всех IP-адресов с одним из этих но-

меров портов. Таким образом, никто посторонний не сможет войти в сеть через Telnet или просмотреть список текущих пользователей сети с помощью программы Finger. Кроме того, компания может таким образом не допустить, чтобы ее сотрудники весь день читали новости USENET.

Блокирование исходящих пакетов сложнее. Несмотря на то что названия большинства сайтов чаще всего соответствуют стандартным соглашениям об именах, никто не обязывает их придерживаться. Кроме того, для некоторых важных служб, таких как FTP (File Transfer Protocol — протокол передачи файлов), номера портов назначаются динамически. Более того, хотя блокирование TCP-соединения является непростым делом, блокировать UDP-пакеты еще тяжелее, так как почти ничего нельзя сказать заранее о том, что они собираются делать. Многие пакетные фильтры по этой причине просто запрещают UDP-трафик совсем.

Вторая составляющая механизма брандмауэра представляет собой **шлюз прикладного уровня**. Вместо того чтобы просто разглядывать пакеты, этот шлюз работает на прикладном уровне. Например, может быть установлен почтовый шлюз, просматривающий каждое входящее и выходящее сообщение. Каждое сообщение пропускается или отвергается в зависимости от содержимого полей заголовков, размера сообщения и даже содержимого (например, шлюз, работающий на военном объекте, может реагировать особым образом на ключевые слова вроде «атомная» или «бомба»).

Может быть одновременно установлено несколько шлюзов для специфических приложений, тем не менее, осторожные организации нередко разрешают обмен электронной почтой и даже WWW, однако запрещают все остальное как слишком рискованное. В сочетании с шифрованием и фильтрацией пакетов подобные меры обеспечивают некоторый уровень безопасности ценой некоторого неудобства.

Даже в случае идеально настроенного брандмауэра остается множество проблем, связанных с безопасностью. Например, если входящие пакеты пропускаются только со стороны конкретных сетей (например, со стороны ЛВС дочерней фирмы компании), взломщик, находящийся вне зоны действия брандмауэра, может просто фальсифицировать адрес отправителя и тем самым преодолеть барьер. Если же нечестный сотрудник компании решит переслать секретную документацию, он может зашифровать ее или вообще сфотографировать, и тогда эти данные смогут проникнуть через любые лингвистические анализаторы. Мы даже не обсуждаем тот факт, что в 70 % случаев мошенники находятся в зоне действия брандмауэра. Очень часто ими являются недовольные сотрудники (Schneier, 2000).

К тому же, существует целый класс атак, с которыми не способны справиться никакие брандмауэры. Идея, лежащая в основе брандмауэров, заключается в том, чтобы не давать взломщикам проникнуть в систему, а секретным данным — уходить наружу. К сожалению, в мире есть много людей, которые не могут найти себе лучшего занятия, нежели препятствовать работоспособности сайтов. Они отправляют вполне легитимные сообщения до тех пор, пока сайт не перестанет функционировать из-за чрезмерной нагрузки. Например, такое хулиганство может заключаться в рассылке пакетов *SYN* для установки соединений. Сайт выде-

лит часть таблицы под это соединение и пошлет в ответ пакеты *SYN + ACK*. Если взломщик не ответит, табличная запись будет продолжать оставаться зарезервированной в течение нескольких секунд до наступления тайм-аута. Если одновременно посылаются тысячи запросов на соединение, никакие запросы от честных граждан просто не пробьются к серверу, так все ячейки таблицы окажутся заняты. Атаки, целью которых является нарушение деятельности объекта, а не получение секретных данных, называются атаками типа **DoS** (Denial of Service — отказ в обслуживании (запроса) — сравните с сокращением QoS — качество обслуживания). Обычно адрес отправителя в пакетах с запросами фальсифицирован, поэтому найти вандала не так просто.

Существует и более жестокий вариант такой атаки. Если сетевому хулигану уже удалось взломать несколько сотен компьютеров, расположенных по всему миру, он может приказать им всем забивать запросами один и тот же сервер. Тем самым не только повышается «убойная сила», но и уменьшаются шансы на обнаружение негодяя, так как пакеты приходят с самых разных компьютеров, ничем плохим себя ранее не зарекомендовавших. Этот тип атаки носит название **DDoS** (Distributed Denial of Service — распределенный отказ в обслуживании). С этой напастью бороться трудно. Даже если атакуемая машина сможет быстро распознать поддельный запрос, на его обработку и отвержение потребуется некоторое время, в течение которого придут другие запросы, и в итоге центральный процессор будет постоянно занят их обработкой.

Виртуальные частные сети

Многие компании владеют множеством подразделений, расположенных в разных городах, иногда даже в разных странах. До появления общедоступных сетей передачи данных обычным делом было арендовать выделенную телефонную линию для организации связи между некоторыми или всеми парами подразделений. В некоторых компаниях такой подход применяется до сих пор. Сеть, состоящая из компьютеров, принадлежащих компании, и выделенных телефонных линий, называется **частной сетью**. Пример частной сети, соединяющей три подразделения, показан на рис. 8.26, *а*.

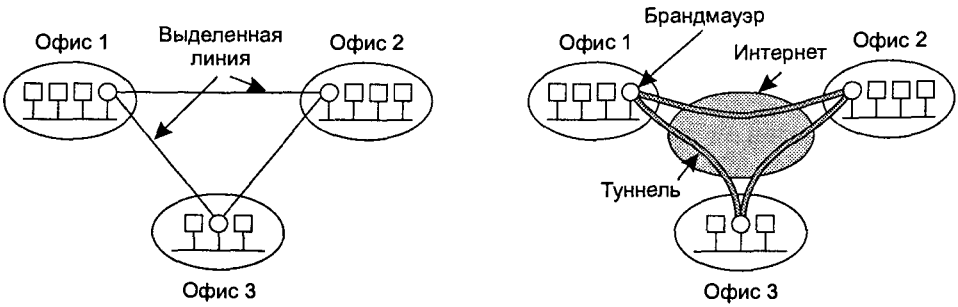


Рис. 8.26. Частная сеть на основе выделенной линии (*а*); виртуальная частная сеть (*б*)

Частные сети работают хорошо и обладают высокой защищенностью. Если бы были доступны только выделенные линии, то отсутствовала бы проблема утечки трафика, и взломщикам пришлось бы физически подключаться к линиям, чтобы перехватить данные, а это не так просто. Беда в том, что стоимость аренды одного выделенного канала T1 составляет тысячи долларов (в месяц!), а аренда линии T3 во много раз дороже. Когда появились общедоступные сети передачи данных, у компаний возникло естественное желание воспользоваться ими для передачи данных (а может, и голоса). При этом, правда, не хотелось терять свойства защищенности, присущие частной сети.

Это соображение вскоре привело к изобретению **виртуальных частных сетей** (VPN — Virtual Private Networks), которые являются оверлейными сетями, работающими поверх обычных общедоступных сетей, но обладающими свойствами частных сетей. Они называются «виртуальными», потому что такие сети — это почти иллюзия; аналогичным образом виртуальные каналы — это не реальные каналы, а виртуальная память — это не реальная память.

Хотя виртуальные частные сети могут строиться на основе ATM (или сетей с коммутацией кадров — frame relay), все более популярным становится организация VPN прямо в Интернете. При этом обычно каждый офис оборудуется брандмауэром и создаются интернет-туннели между всеми парами офисов, как показано на рис. 8.26, *б*. Если IPsec работает в режиме туннелирования, можно собрать весь трафик между любыми двумя парами офисов в один надежный поток и установить защищающую связь, обеспечив тем самым контроль целостности, секретности и даже определенный иммунитет против анализа трафика.

При запуске системы каждая пара брандмауэров должна договориться о параметрах защищающей связи, таких как набор услуг, режимов, алгоритмов и ключей. Во многие брандмауэры встроен специальный инструментарий для работы с виртуальными частными сетями, но можно построить систему и на обычных маршрутизаторах. Тем не менее, поскольку брандмауэры — это почти неотъемлемая часть систем сетевой безопасности, вполне естественно начинать и заканчивать туннели именно на брандмауэрах, проводя четкую границу между компанией и Интернетом. Таким образом, наиболее распространенная комбинация подразумевает наличие брандмауэров, виртуальных частных сетей и IPsec с ESP в режиме туннелирования.

После установки защищающей связи начинается передача данных. С точки зрения маршрутизатора, работающего в Интернете, пакет, проходящий по туннелю VPN, — это самый обычный пакет. Единственное, что его отличает от остальных, это наличие заголовка IPsec после заголовка IP. Но поскольку дополнительные заголовки на процесс пересылки никак не влияют, маршрутизаторы не сильно беспокоит заголовок IPsec.

Основное преимущество такой организации виртуальной частной сети состоит в том, что она совершенно прозрачна для всего пользовательского ПО. Установкой и управлением защищающих связей занимаются брандмауэры. Единственный человек, которому есть дело до настройки сети, — это системный администратор, который обязан сконфигурировать и поддерживать брандмауэры. Для всех остальных виртуальная частная сеть мало чем отличается от частной

сети на основе выделенной линии. Более подробно про VPN написано в (Brown, 1999; Izzo, 2000).

Безопасность в беспроводных сетях

Оказывается, удивительно просто создать систему, которая логически полностью надежна, то есть состоит из VPN и брандмауэров, и при этом на практике протекает, как решето. Такая ситуация может возникнуть, если в сети есть беспроводные машины, передающие данные с помощью радиосигнала, проходящего прямо над брандмауэром в обе стороны. Радиус действия сетей типа 802.11 может составлять несколько сотен метров, поэтому шпион, желающий перехватить информацию, может просто приехать на автостоянку перед зданием фирмы, оставить в машине ноутбук с приемопередатчиком 802.11, записывающим все, что слышно в эфире, и пойти гулять по городу. К вечеру на жестком диске он обнаружит массу ценной информации. Теоретически такого происходить не должно. Правда, теоретически же ограбления банков тоже не должны происходить.

За многие проблемы безопасности стоит сказать спасибо производителям беспроводных базовых станций (точек доступа), пытающихся сделать свою продукцию дружественной по отношению к пользователю. Обычно, если пользователь вынимает свое устройство из сумки и вставляет в розетку, оно сразу начинает работать, и практически всегда все окружающие в зоне действия радиопередатчика смогут услышать любые секреты, о которых он проболтается. Если же затем это устройство подключить к Ethernet, весь трафик, проходящий по локальной сети, может быть перехвачен ноутбуком, стоящим в припаркованной неподалеку машине. Беспроводная связь — это мечта шпиона, ставшая реальностью: информация сама идет в руки, только успевай ее ловить. Очевидно, что вопрос безопасности в беспроводных сетях стоит куда острее, чем в проводных. В этом разделе мы рассмотрим некоторые методы, позволяющие в какой-то мере обезопасить системы такого рода. Дополнительную информацию можно найти в (Nichols и Lekkas, 2002).

Безопасность в сетях 802.11

Стандарт 802.11 описывает протокол безопасности уровня передачи данных под названием **WEP** (Wired Equivalent Privacy — секретность, эквивалентная проводным сетям), предназначенный для того, чтобы обезопасить беспроводные ЛВС так же надежно, как и проводные. По умолчанию в проводных сетях вопрос безопасности отсутствует как таковой, поэтому добиться этой цели несложно, и WEP, как мы увидим далее, справляется со своей задачей.

При наличии системы безопасности в сети 802.11 каждая станция имеет общий закрытый ключ с базовой станцией. Метод распространения ключей стандартом не оговаривается. Скажем, они могут быть прошиты в устройствах или программах производителем. Ими можно обменяться заранее по проводной сети. Наконец, либо базовая станция, либо пользовательская машина может случайным образом выбирать ключ и отсылать его противоположной стороне, предварительно зашифровав при помощи открытого ключа этой стороны. После уста-

новки ключи могут оставаться неизменными в течение нескольких месяцев или даже лет.

Шифрация при помощи WEP использует потоковый шифр, основанный на алгоритме RC4. Создателем RC4 был Роналд Ривест (Ronald Rivest). Этот алгоритм хранился в тайне до тех пор, пока в 1994 году он не просочился в Интернет. Как мы уже упоминали, практически нереально сохранить какой-либо алгоритм в тайне, даже с такой скромной целью, как соблюдение закона об интеллектуальной собственности (случай RC4), не говоря уже о том, чтобы сохранить его в тайне от взломщиков (а такой задачи создатели RC4 перед собой даже не ставили). В WEP RC4 генерирует потоковый шифр, который суммируется по модулю 2 с открытым текстом, в результате чего получается шифрованный текст.

Полезная нагрузка каждого пакета шифруется с использованием метода, показанного на рис. 8.27. Вначале проверяется контрольная сумма (по многочлену CRC-32), которая добавляется к полезной нагрузке. Так формируется открытый текст, передаваемый алгоритму шифрования. Этот открытый текст складывается по модулю 2 с отрезком ключевого потока, равного ему по размеру. Результатом этих преобразований является зашифрованный текст. Вектор инициализации, необходимый для запуска RC4, передается вместе с шифром. После получения пакета приемник извлекает из него зашифрованные данные (полезную нагрузку), создает ключевой поток из общего закрытого ключа и только что принятого вектора инициализации, затем ключевой поток суммируется по модулю 2 с полезной нагрузкой, что позволяет восстановить открытый текст. Наконец, можно проверить контрольную сумму, чтобы убедиться в подлинности принятой информации.

На первый взгляд, такой подход кажется довольно убедительным, однако метод его взлома уже опубликован (Borisov и др., 2001). Далее будут подведены итоги этого. Во-первых, как ни странно, очень многие используют одинаковые общие ключи для всех пользователей, из-за этого все пользователи могут запросто читать весь трафик друг друга. Это, конечно, подход вполне эквивалентный подходу, принятому вEthernet, однако не слишком безопасный.

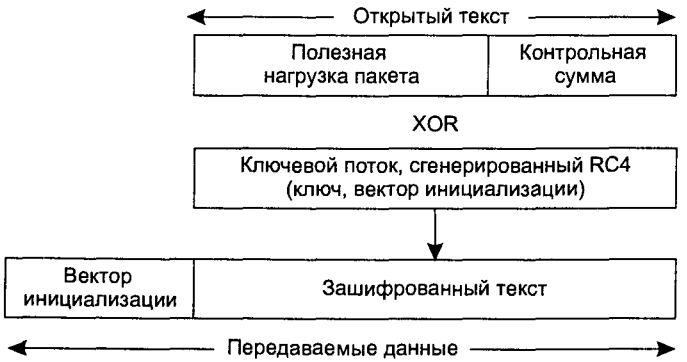


Рис. 8.27. Шифрация пакета с использованием WEP

Но даже если всем пользователям раздать разные ключи, WEP все равно может быть взломан. Так как ключи не изменяются в течение больших периодов времени, стандарт WEP рекомендует (но не обязывает) изменять вектор инициализации при передаче каждого пакета во избежание атак посредством повторного использования (мы обсуждали это в разделе «Режимы шифрования»). К сожалению, многие сетевые карты стандарта 802.11 для ноутбуков сбрасывают вектор инициализации в 0, когда ее вставляют в разъем, и увеличивают на единичку с каждым пересылаемым пакетом. Так как сетевые карты вставляются и вынимаются весьма часто, малые числа, выступающие в качестве векторов инициализации, — обычное дело. Если Труди удастся собрать несколько пакетов, посланных одним и тем же пользователем, с одинаковыми значениями вектора инициализации (который сам по себе посылается открытым текстом вместе с пакетом), она сможет вычислить сумму по модулю 2 двух блоков открытого текста, и, возможно, у нее получится взломать шифр.

Даже если сетевая карта 802.11 будет подбирать значения вектора инициализации для каждого пакета случайным образом, все равно благодаря ограниченной длине вектора (24 разряда) после передачи 2^{24} пакетов векторы начнут повторяться. Хуже того, при случайном методе подбора значений ожидаемое число пакетов, которые можно послать, не опасаясь повторного выпадения того же значения вектора инициализации, равно примерно 5000. Это связано с «задачей о днях рождения», которую мы обсуждали в разделе «Задача о днях рождения». Итак, прослушивая линию в течение нескольких минут, Труди почти наверняка захватит два пакета с одинаковыми векторами инициализации и ключами. Складывая по модулю эти два пакета, она получит сумму (по модулю 2, разумеется) открытых текстов. А уж эту битовую последовательность можно атаковать самыми разными способами с целью восстановления исходных данных. Приложив некоторые усилия, можно подобрать ключевой поток для данного вектора инициализации. Продолжив свои исследования, Труди сможет составить целый словарь ключевых потоков для разных векторов. Взломав вектор инициализации, можно расшифровывать все проходящие по сети пакеты.

Далее при случайном выборе значений векторов Труди достаточно определить рабочую пару (вектор, ключевой поток), чтобы начать генерировать собственные пакеты произвольного содержания. Это может сильно помешать нормальному обмену данными. Теоретически, принимающая сторона может заметить, что слишком много пакетов имеют одинаковые значения векторов инициализации, но ведь WEP этого не запрещает, и к тому же, все равно никто это не проверяет, на самом-то деле.

Наконец, проверка при помощи CRC — это тоже довольно наивный метод, так как Труди может изменить полезную нагрузку так, чтобы она соответствовала циклическому коду по избыточности, для этого даже не придется расшифровывать само сообщение. Короче говоря, сломать защиту 802.11 очень несложно, а ведь мы перечислили далеко не все методы, обнаруженные Борисовым и др.

В августе 2001, спустя месяц после опубликования работы Борисова и др., был обнародован еще один документ, еще больше низвергающий WEP (Fluhrer и др., 2001). В нем отмечается слабость самого алгоритма RC4. Флурер (Fluhrer)

и его соавторы обнаружили, что очень многие ключи обладают одним неприятным свойством: некоторые разряды ключа можно извлечь, анализируя ключевой поток. Если несколько раз повторить попытки атаки шифра, в конце концов удастся извлечь ключ целиком. Для этого даже не понадобится совершать большие усилия. Впрочем, свои теоретические выводы Флурер не пытался применить, взламывая какие-нибудь сети 802.11.

Вместе с тем, когда один студент-практикант и двое ученых из компании AT&T Labs узнали о работе Флурера, они решили испробовать описанный метод на практике (Stubblefield и др., 2002). За неделю был взломан 128-разрядный ключ, использовавшийся в сети 802.11. Причем, большая часть недели ушла на поиски самой дешевой сетевой платы 802.11, получение разрешения на ее приобретение, а также на ее установку и тестирование. Программирование заняло два часа.

После объявления ими результатов своей деятельности телекомпания CNN затеяла историю под названием «Крутой хакер взламывает систему безопасности беспроводных сетей», в которой некоторые гуру этой индустрии пытались высмеять результаты эксперимента. Мол, результаты работы Флурера делают эксперимент слишком тривиальным. С технической точки зрения, это действительно так, но суть не в этом, а в том, что объединенными усилиями двух команд стандарты WEP и 802.11 были просто низвергнуты.

7 сентября 2001 года институт IEEE представил свой ответ на падение стандарта WEP в виде небольшого бюллетеня. В нем названы шесть позиций, которые резюмированы далее:

1. Мы предупреждали, что уровень безопасности, обеспечиваемый WEP, не выше, чем в Ethernet.
2. Гораздо опаснее просто забыть обеспечить безопасность.
3. Надо попробовать разработать какую-нибудь другую систему безопасности (например, на транспортном уровне).
4. Следующая версия, 802.11i, будет, несомненно, обладать более надежной защитой.
5. В будущем сертификация будет подразумевать обязательное использование версии 802.11i.
6. Мы постараемся решить, что делать до того, как появится 802.11i.

Мы более или менее детально рассмотрели эту историю, чтобы читатель мог осознать, что обеспечение безопасности — непростая задача даже для профессионалов.

Безопасность в системах Bluetooth

Радиус действия систем Bluetooth значительно короче, чем сетей 802.11, поэтому взломщику не удастся произвести атаку, оставив ноутбук в припаркованной рядом со зданием машине, однако вопрос безопасности важен и тут. Например, предположим, что компьютер Алисы оборудован беспроводной клавиатурой стандарта Bluetooth. Если не установить систему защиты, то Труди, находясь за стен-

кой, в соседнем офисе, сможет без труда прочесть все, что набирает Алиса, включая исходящую почту. Можно захватить все, что передается на беспроводной принтер, если расположиться неподалеку от него (включая входящую почту и конфиденциальные бумаги). К счастью, в Bluetooth есть рабочая схема защиты, нарушающая планы всевозможных личностей типа Труди. Далее мы опишем основные черты этой схемы.

Система защиты Bluetooth может работать в трех режимах, начиная от полного бездействия и заканчивая тотальной шифрацией данных и контролем целостности. Как и в случае с 802.11, если система защиты отключена (по умолчанию это именно так), о какой-либо безопасности говорить не приходится. Большинство пользователей не включают защиту до тех пор, пока не грянет гром. Можно привести сельскохозяйственный пример такого подхода: ворота конюшни закрывают только после исчезновения лошади.

Bluetooth обеспечивает безопасность на нескольких уровнях. На физическом уровне для этого применяются скачкообразные изменения частот, но поскольку любое устройство, появляющееся в микросети, должно узнать последовательность скачков частоты, эта последовательность, очевидно, не является секретной. Настоящая защита информации начинает проявляться тогда, когда вновь прибывшее подчиненное устройство пытается запросить канал для связи с управляющим устройством. Предполагается, что оба устройства совместно используют предварительно установленный закрытый ключ. В некоторых случаях он прошивается в обоих устройствах (например, в гарнитуре и мобильном телефоне, продающихся вместе). В других случаях в одном из устройств (например, в гарнитуре) ключ прошит, а в сопряженное устройство (например, мобильный телефон) пользователь должен ввести ключ вручную в виде десятичного числа. Общие ключи такого типа называются **отмычками**.

Перед установкой канала подчиненное и управляющее устройства должны выяснить, владеют ли они отмычками. В случае положительного ответа им необходимо договориться о том, каким будет канал: шифрованным, с контролем целостности или и таким, и таким. Затем выбирается 128-разрядный ключ сеанса, некоторые биты которого могут быть сделаны общедоступными. Такое послабление сделано в целях соответствия системы ограничениям, введенным правительствами разных стран и запрещающим экспорт или использование ключей, длина которых больше той, что способно взломать правительство.

Шифрация выполняется с применением потокового шифра $\mathbf{E}_0$, контроль целостности — с применением **SAFER+**. И тот, и другой представляют собой традиционные блочные шифры с симметричными ключами. SAFER+ пытались использовать в AES, однако очень быстро отказались от этой мысли, так как он работал гораздо медленнее других. Работа над Bluetooth завершилась еще до того, как был выбран шифр AES; в противном случае, вероятно, использовался бы алгоритм Rijndael.

Процесс шифрации с использованием ключевого потока показан на рис. 8.12. На нем видно, что открытый текст суммируется по модулю 2 с ключевым потоком. В результате получается шифрованный текст. К сожалению, алгоритм E_0, как и RC4, чрезвычайно слаб (Jacobsson и Wetzel, 2001). Несмотря на то, что на

момент написания книги он еще не взломан, его сходство с шифром A5/1, чей провал угрожает безопасности всего GSM-трафика, наводит на грустные мысли (Biryukov и др., 2000). Многим (в том числе и автору) кажется удивительным тот факт, что в игре «кошки-мышки» между шифровальщиками и криптоаналитиками так часто побеждают последние.

Еще одна проблема безопасности, связанная с Bluetooth, состоит в том, что система идентифицирует только устройства, а не пользователей. Это приводит к тому, что вор, укравший устройство Bluetooth, получит доступ к финансовым и другим счетам жертвы. Тем не менее, система безопасности в Bluetooth реализована и на верхних уровнях, поэтому даже в случае взлома защиты на уровне передачи данных некоторые шансы еще остаются, особенно если приложение для выполнения транзакции требует ввода PIN-кода вручную с помощью какой-нибудь разновидности клавиатуры.

Безопасность в WAP 2.0

Надо признать, что форум разработчиков WAP извлек уроки из нестандартного стека протоколов, придуманного для WAP 1.0. В отличие от первой версии, WAP 2.0 характеризуется стандартными протоколами на всех уровнях. Это касается и вопросов безопасности. Базируясь на IP, он полностью поддерживает все возможности IPsec на сетевом уровне. На транспортном уровне TCP-соединения можно защитить TLS — стандартом IETF, который мы изучим далее в этой главе. На более высоких уровнях применяется идентификация клиентов в соответствии с RFC 2617. Криптографическая библиотека прикладного уровня обеспечивает контроль целостности и обнаружение ложной информации. В конечном итоге, так как WAP 2.0 базируется на известных стандартах, есть шанс, что услуги защиты, в частности, секретность, идентификация и обнаружение ложной информации, будут реализованы значительно лучше, чем в 802.11 и Bluetooth.

Протоколы аутентификации

Аутентификация (или идентификация) — это метод, с помощью которого процесс удостоверяется в том, что его собеседник является именно тем, за кого он себя выдает. Проверка подлинности удаленного процесса при активных злонамеренных попытках проникновения представляет собой удивительно сложную задачу и требует сложных протоколов, основанных на криптографии. В данном разделе мы познакомимся с несколькими протоколами аутентификации, применяемыми в незащищенных компьютерных сетях.

Следует отметить, что понятия аутентификации и авторизации иногда путают. Аутентификация связана с вопросом подлинности вашего собеседника. Авторизация имеет дело с разрешениями. Например, клиентский процесс обращается к файловому серверу и говорит: «Я процесс Скотта, и я хочу удалить файл cookbook.old». Файл-сервер должен решить следующие два вопроса:

1. Действительно ли это процесс Скотта (аутентификация)?
2. Имеет ли Скотт право удалять файл cookbook.old (авторизация)?

Только после того, как на оба вопроса будет получен недвусмысленный утвердительный ответ, может быть выполнено запрашиваемое действие. Ключевым является первый вопрос. После того как сервер узнает, с кем он разговаривает, для проверки прав доступа потребуется лишь просмотреть содержимое локальных таблиц или баз данных. По этой причине в данном разделе мы уделим особое внимание вопросу аутентификации.

Общая схема, используемая всеми протоколами аутентификации, состоит из следующих действий. Алиса желает установить защищенное соединение с Бобом или считающимся надежным **Центром распространения ключей**. Затем в разных направлениях посылаются еще несколько сообщений. По мере их передачи хулиганка по имени Труди может перехватить, изменить и снова воспроизвести эти сообщения, чтобы обмануть Алису и Боба или просто сорвать сделку.

Тем не менее, когда протокол завершит свою работу, Алиса должна быть уверена, что разговаривает с Бобом, а Боб — что разговаривает с Алисой. Кроме того, в большинстве протоколов собеседники также установят секретный **ключ сеанса**, которым будут пользоваться для последующего обмена информацией. На практике весь обмен данными шифруется с помощью одного из алгоритмов с секретным ключом (AES или тройной DES), так как их производительность намного выше производительности алгоритмов с открытым ключом. Тем не менее, алгоритмы с открытым ключом широко применяются в протоколах аутентификации и для определения ключа сеанса.

Цель использования нового, случайно выбираемого ключа сеанса для каждого нового соединения состоит в минимизации трафика, посылаемого с использованием закрытых и открытых ключей пользователя, уменьшении количества шифрованного текста, который может достаться злоумышленнику, а также минимизации вреда, причиняемого в случае, если процесс даст сбой и дамп ядра попадет в чужие руки. Поэтому после установки соединения в процессе должен храниться только один временный ключ сеанса. Все постоянные ключи должны быть тщательно стерты.

Аутентификация, основанная на общем секретном ключе

В нашем первом протоколе аутентификации мы уже предполагали, что у Алисы и Боба есть общий секретный ключ K_{AB}. Об этом секретном ключе можно договориться при личной встрече или по телефону, но, в любом случае, не по сети.

В основе этого протокола лежит принцип, применяемый во многих протоколах аутентификации: одна сторона посылает другой случайное число, другая сторона преобразует его особым образом и возвращает результат. Такие протоколы называются протоколами типа **оклик—отзыв**. В этом и последующих протоколах аутентификации будут использоваться следующие условные обозначения:

A и B — Алиса и Боб;
R_i — оклик, где индекс означает его отправителя;
K_i — ключи, где индекс означает владельца ключа;
K_S — ключ сеанса.

Последовательность сообщений нашего первого протокола аутентификации с общим ключом показана на рис. 8.28. В первом сообщении Алиса посылает свое удостоверение личности, A, Бобу тем способом, который ему понятен. Боб, конечно, не знает, пришло ли это сообщение от Алисы или от злоумышленника, поэтому он выбирает большое случайное число R_B и посылает его в качестве оклика «Алисе» открытым текстом (сообщение 2). Затем Алиса шифрует это сообщение секретным ключом, общим для нее и Боба, и отправляет шифрованный текст $K_{AB}(R_B)$ в сообщении 3. Когда Боб видит это сообщение, он понимает, что оно пришло от Алисы, так как злоумышленник не должен знать ключа K_{AB} и поэтому не смог бы сформировать такое сообщение. Более того, поскольку оклик R_B выбирался случайно в большом пространстве чисел (например, 128-разрядных случайных чисел), очень маловероятно, чтобы злоумышленник мог уже видеть этот оклик и ответ на него в предыдущих сеансах.

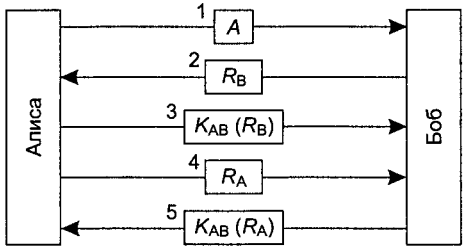


Рис. 8.28. Двусторонняя аутентификация при помощи протокола оклик—отзыв

К этому моменту Боб уверен, что говорит с Алисой, однако Алиса еще пока не уверена ни в чем. Злоумышленник мог перехватить сообщение 1 и послать обратно оклик R_B. Возможно, Боба уже нет в живых. Далее протокол работает симметрично: Алиса посылает оклик, а Боб отвечает на него. Теперь уже обе стороны уверены, что говорят именно с тем, с кем собирались. После этого они могут установить временный ключ сеанса K_S, который можно переслать друг другу, закодировав его все тем же общим ключом K_{AB}.

Количество сообщений в этом протоколе можно сократить, объединив в каждом сообщении ответ на предыдущее сообщение с новым окликом, как показано на рис. 8.29. Здесь Алиса сама в первом же сообщении посылает Бобу оклик. Отвечая на него, Боб помещает в то же сообщение свой оклик. Таким образом, вместо пяти сообщений понадобилось всего три.

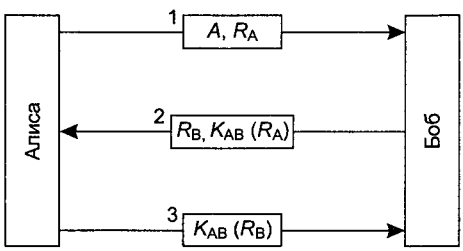


Рис. 8.29. Укороченный двусторонний протокол аутентификации

Лучше ли этот протокол, чем предыдущий? С одной стороны, да: он короче. Но, к сожалению, пользоваться таким протоколом не рекомендуется. При некоторых обстоятельствах злоумышленник может атаковать этот протокол способом, известным под названием **зеркальная атака**. В частности, Труди может взломать его, если ей будет позволено одновременно открыть несколько сеансов связи с Бобом. Такое вполне возможно, если, скажем, Боб — это банк, позволяющий устанавливать несколько одновременных соединений с банкоматами.

Схема зеркальной атаки показана на рис. 8.30. Она начинается с того, что Труди, объявляя себя Алисой, посылает оклик R_T. Боб, как обычно, отвечает своим собственным окликом R_B. Теперь, казалось бы, Труди в тупике. Что ей делать? Она ведь не знает $K_{AB}(R_B)$.

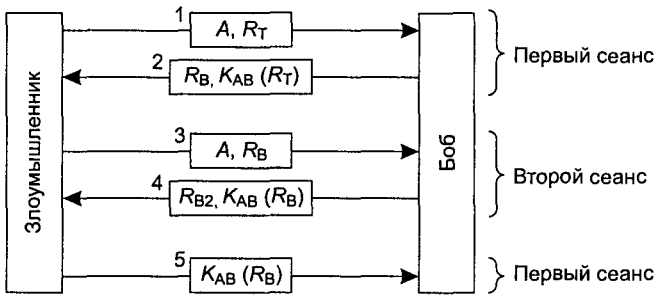


Рис. 8.30. Зеркальная атака

Злоумышленник может открыть второй сеанс сообщением 3 и подать в качестве оклика Бобу оклик самого Боба, взятый из второго сообщения. Боб спокойно шифрует его и посылает обратно $K_{AB}(R_B)$ в сообщении 4. Теперь у Труди есть необходимая информация, поэтому она завершает первый сеанс и прерывает второй. Боб теперь уверен, что злоумышленник — это Алиса, поэтому предоставляет Труди доступ к банковским счетам Алисы и позволяет перевести деньги с ее текущего счета на секретный счет в Швейцарском банке без каких-либо колебаний.

Мораль этой истории такова:

Разработать корректный протокол аутентификации сложнее, чем это может показаться.

Приведем четыре общих правила, которые часто оказываются полезными:

1. Инициатор сеанса должен подтверждать свою личность прежде, чем это сделает отвечающая сторона. В этом случае злоумышленник не сможет получить ценной для него информации, прежде чем подтвердит свою личность.
2. Следует использовать два раздельных общих секретных ключа: один для инициатора сеанса, а другой для отвечающего, K_{AB} и K'_{AB}.
3. Инициатор и отвечающий должны выбирать оклики из различных непересекающихся наборов. Например, инициатор должен пользоваться четными номерами, а отвечающий — нечетными.
4. Протокол должен уметь противостоять атакам, при которых запускается второй параллельный сеанс, информация для которого извлекается при помощи первого сеанса.

Если нарушается хотя бы одно из этих правил, протокол оказывается уязвимым. В приведенном примере были нарушены все четыре правила, что привело к разрушительным последствиям.

Вернемся к ситуации, показанной на рис. 8.28. Можно ли с уверенностью сказать, что этот протокол не подвержен зеркальным атакам? Это зависит от различных факторов. Ситуация с этим очень шаткая. Труди удалось справиться с нашим протоколом, используя зеркальную атаку, потому что он позволял запустить параллельный сеанс с Бобом и ввести его в заблуждение, передав ему его собственный оклик. А что произойдет, если вместо живой Алисы, сидящей за компьютером, стоит обычный компьютер общего назначения, принимающий параллельные сеансы связи? Посмотрим, что Труди сможет сделать.

Чтобы понять, каким образом Труди взламывает протокол, обратимся к рис. 8.31. Алиса объявляет свои идентификационные данные в сообщении 1. Труди это сообщение перехватывает и запускает собственный сеанс, посылая сообщение 2 и прикидываясь Бобом. Здесь мы, как и раньше, изобразили серыми квадратиками сообщения второго сеанса. Алиса отвечает на сообщение 2 так: «Ты представляешься Бобом? Это необходимо подтвердить в сообщении 3». Здесь Труди заходит в тупик: она не может подтвердить, что она — это Боб.

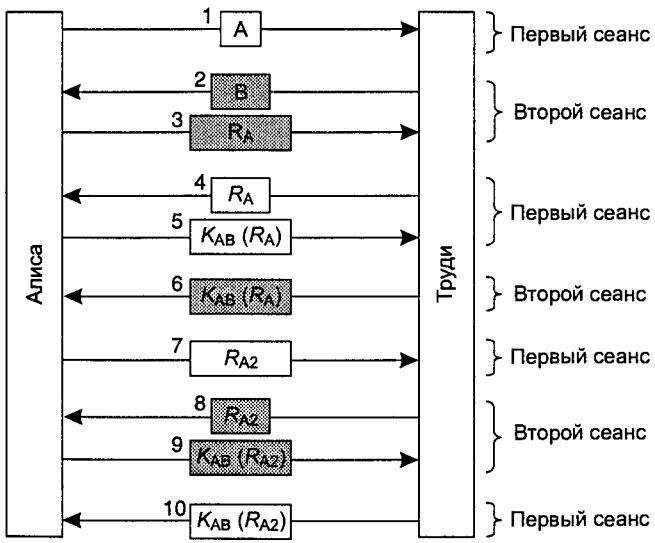


Рис. 8.31. Зеркальная атака протокола, показанного на рис. 8.28

Что же теперь Труди может сделать? Она возвращается к первому сеансу, где как раз наступает ее очередь отправки оклика. При этом отправляется R_A, полученный в сообщении 3. Алиса любезно отвечает на это в сообщении 5, предоставляя тем самым Труди информацию, необходимую ей для создания сообщения 6 в сеансе 2. Труди может теперь выбирать сеанс, так как она корректно ответила на оклик Алисы во втором сеансе. Сеанс 1 можно закрыть, переправить в сеансе 2 какое-нибудь старое число и получить в итоге заверенный сеанс связи с Алисой.

Однако Труди просто невыносима, и она доказывает это своим дальнейшим поведением. Вместо того чтобы отправить какое-нибудь старое число для завершения регистрации сеанса 2, она ждет, пока Алиса пошлет сообщение 7 (ее оклик для сеанса 1). Конечно, Труди понятия не имеет, как ответить на это, поэтому она вновь проводит зеркальную атаку, отправляя R_{A2} в качестве сообщения 8. Алиса очень кстати шифрует R_{A2} в сообщении 9. Труди переключается на сеанс 1 и отправляет Алисе то число, какое ей хочется, в сообщении 10. Откуда она берет это число? Очевидно, из сообщения 9, пришедшего от Алисы. С этого момента Труди может гордиться тем, что у нее есть два полностью заверенных сеанса связи с Алисой.

Эта атака приводит к несколько иному результату, нежели протокол с тремя сообщениями, показанный на рис. 8.30. На этот раз Труди удается установить сразу два заверенных соединения с Алисой. В предыдущем примере одно заверенное соединение было установлено с Бобом. Опять же, если бы протокол удовлетворял всем четырем перечисленным требованиям, атака успеха бы не имела. Детальное обсуждение различных типов атак и методов противодействия им приведено в (Bird и др., 1993). Там также описана методика построения протоколов, корректность которых можно строго доказать. Однако даже простейший из таких протоколов достаточно сложен, поэтому сейчас мы обратимся к другому классу (вполне корректных) протоколов.

Итак, новый протокол аутентификации показан на рис. 8.32 (Bird и др., 1993). Тут мы видим тот же самый HMAC, который мы уже обсуждали при изучении IPsec. Для начала Алиса посылает Бобу отметку времени R_A в виде сообщения 1. Боб при ответе выбирает собственную отметку времени, R_B, и высылает ее вместе с HMAC. HMAC формирует структуру данных, состоящую из временных отметок Алисы и Боба, их идентификаторов, а также общего закрытого ключа K_{AB}. Затем вся эта структура с помощью хэш-функции (например, SHA-1) помещается в HMAC. После приема сообщения 2 Алиса становится счастливым обладателем R_A (это значение выбрано ею же), R_B, полученного в виде открытого текста, двух идентификаторов и закрытого ключа K_{AB}, известного и так. Имея все эти данные, она может вычислить HMAC самостоятельно. Если он согласуется с HMAC, содержащимся в сообщении, она убеждается, что говорит с Бобом, поскольку Труди не знает K_{AB} и, следовательно, не может угадать HMAC, который следует отослать. В ответе Алисы Бобу содержится HMAC, состоящий из двух временных отметок.

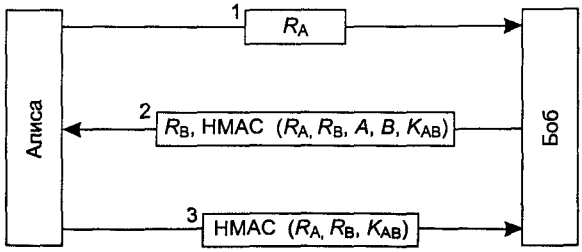


Рис. 8.32. Аутентификация с применением HMAC

Вопрос: может ли Труди как-нибудь взломать такой протокол? Нет, потому что она не может заставить ни одну из сторон шифровать выбранное ею самой значение или применять к нему хэш-функцию, как это было в ситуации на рис. 8.30. Оба HMAC включают в себя значения, выбранные отправителем. Труди не способна их контролировать каким-либо образом.

Использование HMAC — это далеко не единственное, что можно сделать. Альтернативная схема, которая применяется довольно часто, заключается в шифровании элементов данных последовательно с помощью сцепления блоков шифра.

Установка общего ключа: протокол обмена ключами Диффи—Хеллмана

Итак, мы предположили, что у Алисы и Боба есть общий секретный ключ. Предположим теперь, что у них его нет (поскольку до сих пор не разработана универсальная инфраструктура PKI создания подписей и распространения сертификатов). Как им получить такой ключ? Алиса может позвонить Бобу и передать ему ключ по телефону, но он, возможно, спросит: «Как вы докажете, что вы — Алиса, а не злоумышленник?» Они могут попытаться организовать встречу, на которую каждый придет с паспортом, водительскими правами и тремя кредитными картами, но, будучи занятыми людьми, они, возможно, не смогут найти устраивающую обоих дату встречи в течение нескольких месяцев. К счастью, существует способ для совершенно незнакомых людей установить общий секретный ключ средь бела дня, даже если злоумышленник старательно записывает каждое сообщение.

Протокол, позволяющий не встречавшимся ранее людям устанавливать общий секретный ключ, называется **протоколом обмена ключами Диффи—Хеллмана** (Diffie и Hellman, 1976) и работает следующим образом. Алиса и Боб договариваются о двух больших простых числах, n и g, где $(n-1)/2$ также является простым числом, кроме того, на число g накладываются некоторые дополнительные условия. Эти числа могут быть открытыми, поэтому каждый из них может просто выбрать n и g и открыто сообщить о них другому. Затем Алиса выбирает большое (например, 512-разрядное) число x и держит его в секрете. Аналогично, Боб выбирает большое секретное число y.

Алиса начинает протокол обмена ключами с того, что посылает Бобу сообщение, содержащее $(n, g, g^x \bmod n)$, как показано на рис. 8.33. Боб отвечает Алисе сообщением, содержащим $g^y \bmod n$. Теперь Алиса берет число, присланное ей Бобом, и возводит его в степень x, получая $(g^y \bmod n)^x$. Боб выполняет подобные вычисления и получает $(g^x \bmod n)^y$. В соответствии с законами арифметики оба вычисления должны быть равны $g^{xy} \bmod n$. Таким образом, у Алисы и Боба есть общий секретный ключ $g^{xy} \bmod n$.

Конечно, злоумышленник видел оба сообщения. Ему известны значения n и g из первого сообщения. Если бы ему удалось вычислить значения x и y, ему бы удалось получить секретный ключ. Беда в том, что, зная $g^x \bmod n$ и n, найти значение x очень трудно. На сегодняшний день неизвестен алгоритм вычисления дискретного логарифма модуля очень большого простого числа.

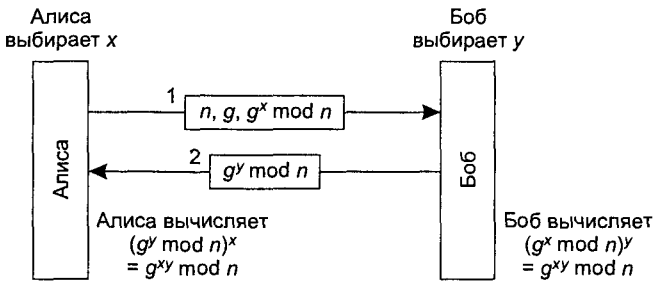


Рис. 8.33. Обмен ключами Диффи—Хеллмана

Для примера возьмем (совершенно нереальные) значения $n = 47$ и $g = 3$. Алиса выбирает значение $x = 8$, а Боб выбирает $y = 10$. Оба эти числа хранятся в секрете. Сообщение Алисы Бобу содержит числа (47, 3, 28), так как $3^8 \bmod 47 = 28$. Боб отвечает Алисе числом 17. Алиса вычисляет $17^8 \bmod 47$ и получает 4. Боб вычисляет $28^{10} \bmod 47$ и получает также 4. Таким образом, независимо друг от друга Алиса и Боб определили, что значение секретного ключа равно 4. Злоумышленнику придется решить уравнение $3^x \bmod 47 = 28$, что можно сделать путем полного перебора для таких небольших чисел, но только не для чисел длиной в несколько сотен бит.

Несмотря на всю элегантность алгоритма Диффи—Хеллмана, имеется одна проблема: когда Боб получит три числа (47, 3, 28), как он сможет удостовериться в том, что они посланы Алисой, а не злоумышленником? Способа узнать это не существует. К сожалению, злоумышленник может воспользоваться этим, чтобы обмануть Алису и Боба, как показано на рис. 8.34. Здесь, пока Алиса с Бобом выбирают значения x и y, злоумышленник выбирает свое случайное число z. Алиса посылает Бобу сообщение 1. Злоумышленник перехватывает его и отправляет вместо него Бобу сообщение 2, используя правильные значения n и g (которые посылались открытым текстом), но со своим значением z вместо x. Он также посылает обратно Алисе сообщение 3. Позднее Боб отправляет Алисе сообщение 4, которое злоумышленник снова перехватывает и хранит у себя.

Теперь все занимаются вычислением остатков от деления. Алиса вычисляет значение секретного ключа: $g^{xz} \bmod n$. Те же самые вычисления производит злоумышленник (для общения с Алисой). Боб вычисляет $g^{yz} \bmod n$, что также делает и злоумышленник (для общения с Бобом). Каждое сообщение, посылаемое Алисой в шифрованном сеансе, перехватывается злоумышленником, сохраняется, изменяется, если это нужно, и отправляется (по желанию злоумышленника) Бобу. То же самое происходит и в обратном направлении. Злоумышленник видит все сообщения и может изменять их по своему усмотрению, в то время как Алиса и Боб полагают, что у них имеется защищенный канал для связи друг с другом. Подобные действия злоумышленника называются атакой типа **«пожарная цепочка»**, поскольку слегка напоминают старинных пожарных, передававших друг другу по цепочке ведра с водой. Еще одно название этой атаки — **«человек посередине»**.

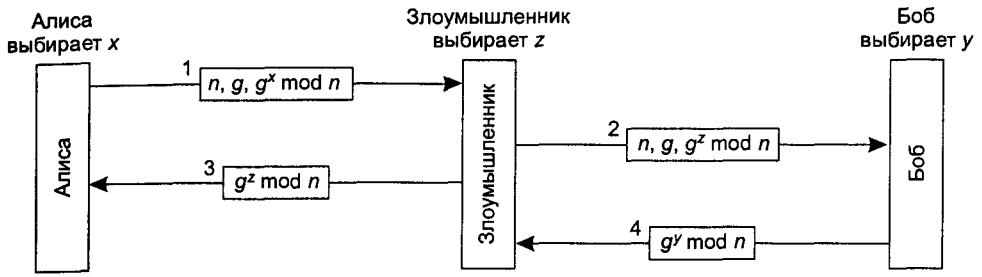


Рис. 8.34. Атака типа «пожарная цепочка»

Аутентификация с помощью центра распространения ключей

Итак, установка общего секретного ключа с незнакомцем почти удалась. С другой стороны, вероятно, не следовало вообще этим заниматься. Чтобы общаться с m людьми, вам понадобится хранить n ключей. Для людей, чей круг общения широк, хранение ключей может превратиться в серьезную проблему, особенно если все эти ключи придется хранить на отдельных пластиковых картах.

Другой подход состоит в организации доверительного центра распространения ключей (KDC, key distribution center). При такой схеме у каждого пользователя всего один ключ, общий с KDC-центром. Операции с ключами аутентификации и сеансовыми ключами проходят через KDC-центр. Простейший протокол аутентификации с помощью центра распространения ключей, включающий две стороны и доверенный KDC-центр, изображен на рис. 8.35.

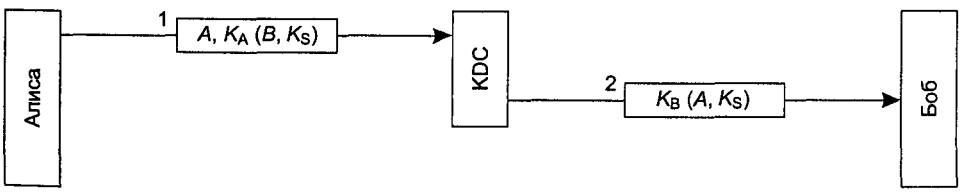


Рис. 8.35. Первая попытка протокола аутентификации с помощью KDC-центра

Идея, лежащая в основе протокола, проста: Алиса выбирает ключ сеанса, K_S, и заявляет KDC-центру, что она желает поговорить с Бобом при помощи ключа K_S. Это сообщение шифруется секретным ключом K_A, которым совместно владеют только Алиса и центр распространения ключей. Центр распространения ключей расшифровывает это сообщение и извлекает из него идентификатор личности Боба и ключ сеанса. Затем он формирует новое сообщение, содержащее идентификатор личности Алисы и ключ сеанса, и посылает его Бобу. Это сообщение зашифровывается ключом K_B – секретным ключом, общим для Боба и центра распространения ключей. Расшифровав это сообщение, Боб узнает, что Алиса желает с ним поговорить и какой ключ она хочет использовать.

Аутентификация в данном случае происходит сама собой. KDC знает, что сообщение 1 пришло от Алисы, так как больше никто не может зашифровать его секретным ключом Алисы. Аналогично, Боб знает, что сообщение 2 пришло от KDC, так как кроме него их общий секретный ключ никому не известен.

К сожалению, этот протокол содержит серьезную ошибку. Труди нужны деньги, поэтому она придумывает некую легальную услугу, которую она могла бы выполнить для Алисы. Затем Труди делает Алисе заманчивое предложение и получает эту работу. Выполнив ее, Труди вежливо предлагает Алисе оплатить услуги, переведя деньги на ее банковский счет. Чтобы оплатить работу, Алиса устанавливает ключ сеанса со своим банкиром Бобом. Затем она посылает Бобу сообщение с просьбой перевести деньги на счет Труди.

Тем временем Труди возвращается к своим темным делам. Она копирует сообщение 2 (см. рис. 8.35) и запрос на перевод денег, следующий за ним. Затем она воспроизводит оба сообщения для Боба. Боб получает их и думает: «Должно быть, Алиса снова наняла Труди. Похоже, она хорошо справляется с работой». Боб перечисляет еще столько же денег со счета Алисы на счет Труди. Получив пятидесятый запрос на перевод денег, Боб выбегает из офиса, чтобы найти Труди и предложить ей большую ссуду, чтобы она могла расширить свой чрезвычайно успешный бизнес. Подобная проблема получила название **атаки повторным воспроизведением**.

Существует несколько решений этой проблемы. Первое решение состоит в помещении в каждое сообщение временного штампа. Все устаревшие сообщения просто игнорируются. Беда здесь в том, что системные часы в сети синхронизировать с большой степенью точности невозможно, поэтому должен существовать какой-то срок годности временного штампа. Труди может обмануть протокол, послав повторное сообщение во время этого интервала.

Второе решение заключается в помещении в сообщение уникального порядкового номера, обычно называемого **нонсом** (nonce — данный случай, данное время). Каждая сторона должна запоминать все предыдущие нонсы и отвергать любое сообщение, содержащее использованный ранее нонс. Однако нонсы должны храниться вечно, иначе Труди попытается воспроизвести сообщение пятилетней давности. Кроме того, если машина потеряет список нонсов в результате сбоя, она снова станет уязвимой к атакам повторным воспроизведением. Можно комбинировать временные штампы и нонсы, чтобы ограничить срок хранения нонсов, но так или иначе, протокол должен быть значительно усложнен.

Более сложный метод аутентификации состоит в использовании многостороннего протокола оклик—отзыв. Хорошо известным примером такого протокола является **протокол аутентификации Нидхэма—Шрёдера** (Needham—Schroeder, 1978), один из вариантов которого показан на рис. 8.36.

Работа протокола начинается с того, что Алиса сообщает KDC-центру, что она желает поговорить с Бобом. Это сообщение содержит в качестве нонса большое случайное число R_A. Центр распространения ключей посылает обратно сообщение 2, содержащее случайное число Алисы, ключ сеанса и так называемый билет, который она может послать Бобу. Цель посылки случайного числа R_A состоит в том, чтобы убедить Алису в том, что сообщение 2 является свежим, а не

повторно воспроизведенным. Идентификатор Боба также помещается в сообщение 2 на случай, если злоумышленник (Труди) вздумает заменить его идентификатор на свой в сообщении 1, так чтобы KDC-центр зашифровал билет в конце сообщения 2 ключом K_T (ключ Труди) вместо K_B. Билет, зашифрованный ключом K_B, помещается внутри зашифрованного сообщения, чтобы злоумышленник не смог заменить его чем-либо другим, пока сообщение 2 добирается до Алисы.

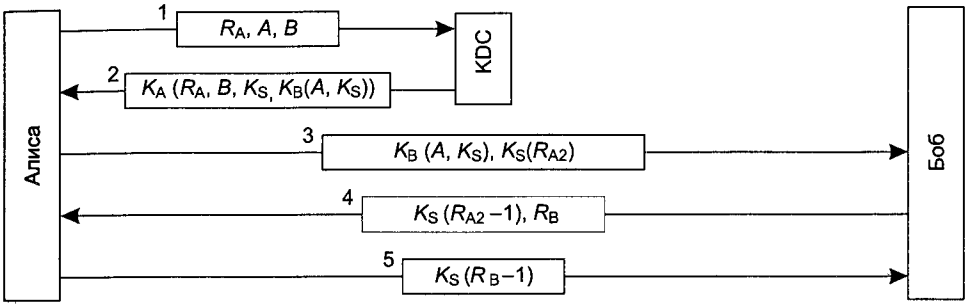


Рис. 8.36. Протокол аутентификации Нидхэма—Шрёдера

Затем Алиса посылает билет Бобу вместе с новым случайным числом R_{A2}, зашифрованным ключом сеанса K_S. В сообщении 4 Боб посылает обратно $K_S(R_{A2}-1)$, чтобы доказать Алисе, что она разговаривает с настоящим Бобом. Отсылать обратно просто $K_S(R_{A2})$ бессмысленно, так как это число могло быть украдено злоумышленником из сообщения 3.

Получив сообщение 4, Алиса убеждается, что разговаривает с Бобом и что до сих пор не было использовано повторных сообщений. Между отправкой случайного числа R_{A2} и получением ответа на него в виде $K_S(R_{A2}-1)$ проходит довольно короткий промежуток времени. Цель сообщения 5 — убедить Боба, что он действительно разговаривает с Алисой и что в этом сеансе связи также отсутствуют повторно воспроизведенные данные. Возможность атаки с помощью повторного воспроизведения ранее записанной информации исключается этим протоколом благодаря тому, что каждая сторона формирует оклик другой стороны и получает на него отзыв.

Несмотря на всю кажущуюся солидность протокола, в нем, тем не менее, имеется небольшое слабое место. Если злоумышленнику удастся каким-либо способом раздобыть старый ключ сеанса K_S, он сможет инициировать новый сеанс с Бобом, повторно воспроизведя сообщение 3 с использованием скомпрометированного ключа, и выдать себя за Алису (Denning и Sacco, 1981). На этот раз злоумышленник может украсть деньги со счета Алисы, даже не выполнив никаких услуг.

Позднее Нидхэм и Шрёдер опубликовали протокол, решающий эту проблему (Needham и Shroeder, 1987). В том же выпуске того же журнала Отуэй (Otway) и Рис (Rees) также опубликовали протокол, решающий эту проблему более коротким путем. На рис. 8.37 показан слегка видоизмененный протокол Отуэя—Риса.

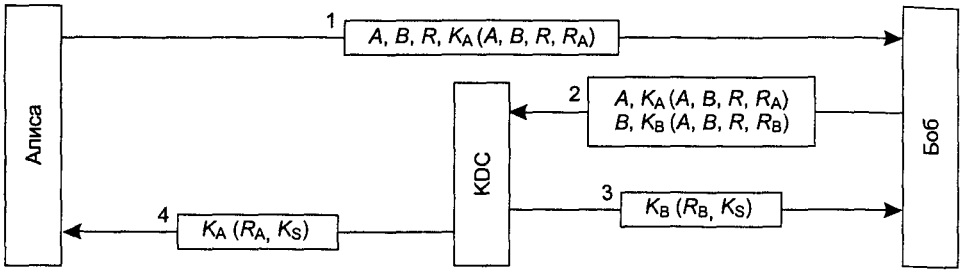


Рис. 8.37. Протокол аутентификации Отуэя—Риса (слегка упрощенный)

В протоколе Отуэя—Риса Алиса начинает с формирования пары случайных номеров: R, который будет использоваться в качестве общего идентификатора, и R_A, который Алиса будет использовать в качестве оклика Боба. Получив это сообщение, Боб формирует новое сообщение из зашифрованной части сообщения Алисы и аналогичной собственной части. Обе части сообщения, зашифрованные ключами K_A и K_B, идентифицируют Алису и Боба, содержат общий идентификатор и оклики.

Центр распространения ключей проверяет, совпадают ли общие идентификаторы R в обеих частях сообщения. Они могут не совпадать, если злоумышленник подменил R в сообщении 1 или заменил часть сообщения 2. Если оба общих идентификатора R совпадают, KDC-центр считает сообщение, полученное от Боба, достоверным. Затем он формирует ключ сеанса K_S и отправляет его Алисе и Бобу, зашифровав ключ сеанса ключами Алисы и Боба. Каждое сообщение также содержит случайное число получателя в доказательство того, что эти сообщения посланы KDC-центром, а не злоумышленником. К этому моменту Алиса и Боб обладают одним и тем же ключом сеанса и могут начать обмен информацией. После первого же обмена данными они увидят, что обладают одинаковыми копиями ключа сеанса K_S, на чем процесс аутентификации можно будет считать завершенным.

Аутентификация при помощи протокола Kerberos

Во многих реально работающих системах применяется протокол аутентификации **Kerberos**, основанный на одном из вариантов протокола Нидхэма—Шрёдера. Он назван по имени трехглавого пса греческих мифов Кербера (чаще называемого Цербером благодаря латинскому написанию. — *Примеч. перев.*), охранявшего выход из Аида. Кербер пропускал в Аид всякого, но не выпускал оттуда никого. Протокол Kerberos был разработан в Массачусетском технологическом институте для обеспечения пользователям рабочих станций надежного доступа к сетевым ресурсам. Его основное отличие от протокола Нидхэма—Шрёдера состоит в предположении о довольно хорошей синхронизации всех часов в сети. Было разработано несколько последовательных версий протокола. Версия V4 наиболее широко применяется в промышленности, поэтому она будет здесь описана. Затем

будет сказано несколько слов о следующей версии, V5. Дополнительную информацию см. в (Steiner и др., 1988).

В работе протокола Kerberos, помимо рабочей (клиентской) станции Алисы, принимают участие еще три сервера:

- сервер аутентификации (AS, Authentication Server): проверяет личность пользователей при входе в сеть;
- сервер выдачи билетов (TGS, Ticket Granting Server): выдает «билеты, подтверждающие подлинность»;
- Боб, то есть сервер, предоставляющий услуги Алисе.

Сервер аутентификации AS аналогичен центру распространения ключей KDC в том, что у него есть общий секретный пароль для каждого пользователя. Работа сервера выдачи билетов TGS состоит в выдаче свидетельств, убеждающих другие серверы в том, что владелец билета действительно является тем, за кого он себя выдает.

Чтобы начать сеанс, Алиса усаживается за клавиатуру произвольной общедоступной рабочей станции и вводит свое имя. Рабочая станция посылает введенное имя открытым текстом на сервер аутентификации, как показано на рис. 8.38. Сервер аутентификации AS возвращает рабочей станции Алисы ключ сеанса и билет $K_{TGS}(A, K_S)$ для сервера выдачи билетов TGS. Эти данные упаковываются вместе и зашифровываются секретным ключом Алисы так, чтобы только Алиса могла их расшифровать. Только после получения сообщения 2 рабочая станция запрашивает пароль Алисы. С помощью этого пароля формируется ключ K_A, которым расшифровывается сообщение 2, и из него извлекаются ключ сеанса и билет для получения доступа к серверу выдачи билетов TGS. После расшифровки рабочая станция сразу же уничтожает хранящийся в ее памяти пароль. Если вместо Алисы на рабочей станции попытается зарегистрироваться Труди, введенный ею пароль окажется неверным, что будет обнаружено рабочей станцией, так как стандартная часть сообщения 2 окажется неверной.

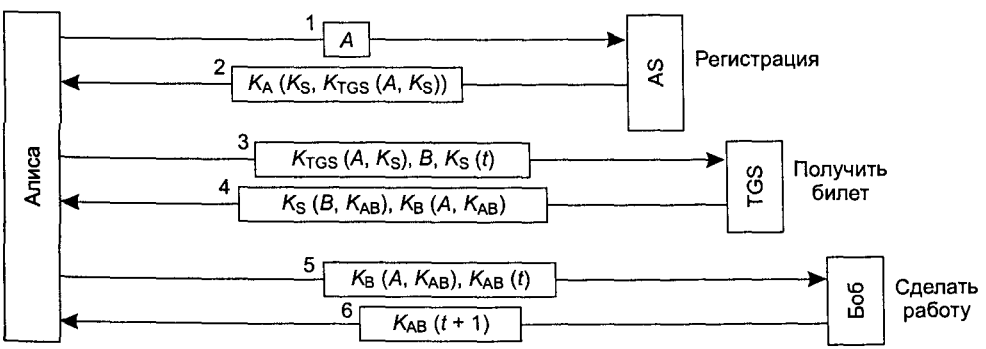


Рис. 8.38. Работа протокола Kerberos V4

После регистрации в сети Алиса может сообщить рабочей станции, что она хочет вступить в контакт с файловым сервером, то есть Бобом. При этом рабочая

станция посылает серверу выдачи билетов сообщение 3 с просьбой выдать билет для общения с Бобом. Ключевым элементом этого запроса является билет $K_{TGS}(A, K_S)$, который зашифрован секретным ключом TGS-сервера и используется для подтверждения личности отправителя. Сервер выдачи билетов отвечает созданием ключа сеанса K_{AB}, которым будут пользоваться Алиса и Боб. Он отправляет Алисе две версии этого ключа. Один ключ зашифрован ключом сеанса K_S, поэтому Алиса может его прочитать. Второй ключ шифруется ключом Боба K_B, что позволяет Бобу его прочитать.

Злоумышленник может скопировать сообщение 3 и попытаться использовать его снова, но ему помешает временной штамп t, отправляемый вместе с этим сообщением. Злоумышленник не может заменить этот временной штамп на более новый, так как не знает ключа сеанса K_S, которым пользуется Алиса для общения с сервером выдачи билетов. Даже если злоумышленник очень быстро повторит сообщение 3, все равно, единственное, что он получит в ответ, это сообщение 4, которое он не смог расшифровать в первый раз и не сможет расшифровать и во второй раз.

После этого Алиса может послать Бобу ключ K_{AB} для установки сеанса с Бобом. Эти сообщения также содержат временные штампы. Сообщение 6, получаемое в ответ, подтверждает, что Алиса говорит именно с Бобом, а не со злоумышленником.

Наконец, после этой серии обмена сообщениями Алиса сможет обмениваться с Бобом данными, используя ключ сеанса K_{AB}. Если после этого Алиса решит, что ей необходим другой сервер, например Кэрол (Carol, C), она может просто послать серверу выдачи ключей сообщение, аналогичное третьему, заменив в нем B на C (то есть идентификатор Боба на идентификатор Кэрол). TGC-сервер мгновенно ответит сообщением, содержащим билет, зашифрованный ключом K_C. Этот билет Алиса пошлет Кэрол, для которой он будет служить гарантией подлинности Алисы.

Достоинство этого протокола состоит в том, что теперь Алиса может получать защищенный доступ к любому серверу сети, и в то же время ее пароль ни разу не передавался по сети. В действительности он только на несколько миллисекунд появлялся в ее рабочей станции. Однако обратите вниманиена то, что каждый сервер выполняет свою собственную процедуру авторизации. Когда Алиса предъявляет свой билет Бобу, это всего лишь подтверждает Бобу подлинность предъявителя билета. К чему же Алиса может получить доступ на сервере, решает Боб.

Поскольку разработчики системы Kerberos не рассчитывали, что весь мир станет доверять одному единственному серверу аутентификации, они обеспечили существование нескольких **областей**, каждая из которых имеет свой собственный сервер аутентификации и сервер выдачи билетов. Чтобы получить билет для сервера, расположенного в удаленной области, Алиса должна запросить у своего TGS-сервера билет, который будет принят TGS-сервером удаленной области. Если удаленный TGS-сервер зарегистрировался на локальном TGS-сервере (так же, как это делают локальные серверы), локальный TGS-сервер выдаст Алисе билет, действительный на удаленном TGS-сервере. После этого она может получить у удаленного TGS-сервера билеты к серверам данной удаленной области.

Обратите внимание на то, что для того чтобы две стороны, расположенные в различных областях, могли установить друг с другом защищенный сеанс связи, каждая из сторон должна доверять TGS-серверу другой стороны.

Протокол Kerberos V5 сложнее четвертой версии и подразумевает большее количество накладных расходов. Кроме того, он использует язык OSI ASN.1 для описания типов данных. Претерпели небольшие изменения и протоколы. Помимо этого в системе Kerberos V5 время жизни билетов более длительное, билеты могут обновляться и даже датироваться задним числом. Также, по крайней мере в теории, пятая версия системы Kerberos не является зависимой от стандарта DES, как V4, и поддерживает различные области.

Аутентификация с помощью шифрования с открытым ключом

Взаимная аутентификация также может выполняться с помощью шифрования с открытым ключом. Для начала Алисе нужно получить открытый ключ Боба. Если инфраструктура PKI реализована на основе сервера каталогов, выдающего сертификаты на открытые ключи, Алиса может потребовать сертификат Боба, что показано в виде сообщения 1 на рис. 8.39. Ответ, содержащийся в сообщении 2, — это сертификат X.509 с открытым ключом Боба. Проверив корректность подписи, Алиса может отправить Бобу сообщение со своим идентификатором и нонсом.

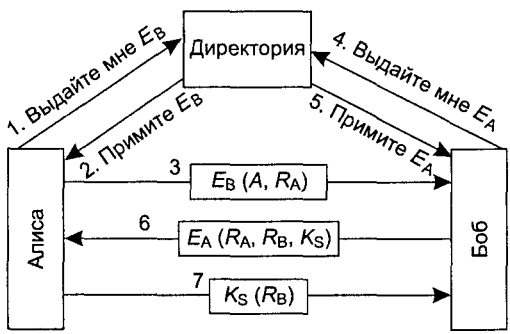


Рис. 8.39. Взаимная идентификация с помощью открытого ключа

Когда Боб получает это сообщение, он не знает, пришло ли оно от Алисы или от злоумышленника, но он делает вид, что все в порядке, и просит сервер каталогов выдать ему открытый ключ Алисы (сообщение 4). Вскоре он его получает (в сообщении 5). Затем он отсылает Алисе сообщение, содержащее случайное число Алисы R_A, свой собственный нонс R_B и предлагаемый ключ сеанса K_S. Все это сообщение зашифровывается открытым ключом Алисы.

Алиса расшифровывает полученное сообщение 6 своим закрытым ключом. Она видит в нем свое случайное число, R_A, и очень этому рада: это подтверждает, что сообщение пришло от Боба, так как у злоумышленника не должно быть способа определить значение этого числа. Кроме того, случайное число R_A свиде-

тельствует о свежести этого сообщения. Алиса соглашается на установку сеанса, отправляя сообщение 7. Когда Боб видит свое случайное число R_B, зашифрованное ключом сеанса, который он сам же сформировал, он понимает, что Алиса получила сообщение 6 и проверила значение R_A.

Может ли злоумышленник каким-либо образом обмануть этот протокол? Он может сфабриковать сообщение 3 и спровоцировать Боба на проверку Алисы, но Алиса увидит число R_A, которого она не посылала, и не станет продолжать. Злоумышленник не сможет убедительно подделать сообщение 7, так как ему не известны значения оклика R_B или ключа K_S, и он не может определить их, не имея закрытого ключа Алисы. Так что ему не везет.

Конфиденциальность электронной переписки

При пересылке между двумя удаленными пользователями сообщение обычно преодолевает по пути десяток других машин. Любая из них может читать и записывать проходящую через нее почту. Конфиденциальности не существует, что бы ни думали об этом многие пользователи. Тем не менее, многие пользователи желали бы иметь возможность посылать электронную почту так, чтобы ее мог прочитать только тот, для кого она предназначается, и никто другой: ни шеф, ни хакеры, ни даже правительство. Эта потребность стимулировала применение некоторыми группами и отдельными разработчиками криптографических принципов к электронной почте. В следующих разделах мы познакомимся с широко распространенной системой защиты электронной почты PGP, а также дадим общее представление о двух других: PEM и S/MIME. Дополнительную информацию см. в (Kaufman и др., 2002; Schneier, 1995).

PGP — довольно неплохая конфиденциальность

Наш первый пример, система **PGP** (Pretty Good Privacy — довольно хорошая конфиденциальность), создана всего одним человеком, Филом Циммерманом (Phil Zimmermann, 1995a, 1995b). Циммерман является сторонником безопасности в сетях, и его девиз таков: «Если конфиденциальность объявлена вне закона, значит, пользоваться ею будут только нарушители закона». Выпущенная в 1991 году система PGP представляет собой полный пакет для электронной почты, обеспечивающий конфиденциальность, аутентификацию, цифровые подписи и сжатие. Все это делается в легкой и удобной форме. Более того, полный пакет, включающий все исходные тексты программ, свободно распространяется через Интернет. Благодаря своему качеству, цене (нулевой) и простоте установки на различных платформах, включая UNIX, Linux, Windows и Mac OS, в настоящее время система PGP получила широкое распространение.

PGP кодирует данные с помощью блочного шифра **IDEA** (International Data Encryption Algorithm — международный алгоритм шифрования данных), ис-

пользующего 128-разрядные ключи. Он был изобретен в Швейцарии в те времена, когда DES уже считался устаревшим, а AES еще не был придуман. Концептуально IDEA похож на DES/AES: производится смешивание разрядов в серии, однако детали реализации функций отличают его от DES и AES. Управление ключами происходит с помощью RSA, а для задач обеспечения целостности данных применяется MD5. Все эти методы мы обсуждали ранее.

История создания системы PGP весьма запутана с первого дня ее существования (Levy, 1993). Поскольку она свободно распространялась через Интернет, правительство США объявило, что Циммерман нарушил закон, запрещающий экспорт военного имущества. Следствие по этому делу длилось пять лет, однако в один прекрасный день прекратилось по двум основным причинам. Во-первых, Циммерман собственноручно не выкладывал PGP в Интернете, и адвокат аргументировал позицию защиты тем, что обвиняемый *сам* никогда не занимался экспортом чего бы то ни было (кроме того, еще надо доказать, что создание сайта равносильно экспорту). Во-вторых, правительство вдруг осознало, что выигрыш дела означал бы, что любой веб-сайт, содержащий загружаемые программы, связанные с секретностью, подпадает под действие закона о торговле такими предметами, как танки, подводные лодки, военные самолеты и ядерное оружие. Этим можно было бы добиться лишь одного: бурного протеста общественности. Это не решение проблемы.

Честно говоря, законы, касающиеся экспорта, кажутся несколько диковатыми. Правительство решило, что размещение программы на веб-странице можно приравнять к нелегальному экспорту, и надоедало Циммерману целых 5 лет. С другой стороны, если кто-то опубликует в книге полный исходный код PGP на языке C (крупным шрифтом, да еще и с контрольной суммой в конце каждой страницы, что облегчит сканирование) и затем займется экспортом этой книги, правительство и глазом не моргнет: книги по закону не являются военным имуществом. *Оружие сильней пера в законе дяди Сэма.*

Еще одна проблема, с которой внезапно столкнулась PGP, была связана с посягательством на патентные права. Владелец патента на RSA, корпорация RSA Security, сослалась на то, что использование метода RSA в PGP является посягательством на патент. Эта проблема разрешилась в версиях начиная с 2.6. Забавно, что вместо RSA в PGP стали применять IDEA, что поначалу тоже вызывало некоторые вопросы.

Так как PGP — это система с открытым исходным кодом, появилось множество модификаций, созданных различными группами и отдельными заинтересованными лицами. Некоторые из них пытались каким-то образом обойти законы об экспорте оружия, другие старались избежать применения запатентованных алгоритмов, а третьи работали над превращением PGP в коммерческий продукт с закрытым исходным кодом. Несмотря на то, что законы об экспорте оружия несколько смягчились (тем не менее, продукцию, использующую AES, до сих пор нельзя экспортировать за пределы США), а срок действия патента RSA закончился в сентябре 2000 года, следствием всех этих проблем стало появление и распространение нескольких несовместимых версий PGP, имеющих разные названия. Далее обсуждается классический вариант PGP, он же является самым старым

и простым. Еще одна популярная версия, Open PGP, описана в RFC 2440. Можно отметить еще GNU Privacy Guard.

В системе PGP намеренно используются уже существующие криптографические алгоритмы, а не изобретаются новые. Все они прошли тщательную проверку ведущими криптоаналитиками мира, и история создания этих алгоритмов не запятнана участием каких-либо государственных организаций, пытающихся их ослабить. Последнее качество является особенно большим преимуществом для всех, кто склонен не доверять правительству.

Система PGP поддерживает сжатие текста, секретность и цифровые подписи, а также предоставляет исчерпывающие средства управления ключами. Как ни странно, не поддерживаются средства электронной почты. Она больше всего похожа на препроцессор, берущий на входе открытый текст и создающий на выходе шифр base64. Разумеется, эти выходные данные можно отправить по электронной почте. Некоторые реализации на последнем шаге обращаются к пользовательскому агенту, чтобы упростить задачу реальной отправки сообщения.

Чтобы понять, как работает система PGP, рассмотрим пример на рис. 8.40. Алиса хочет надежным способом послать Бобу открытым текстом подписанное сообщение *P*. У Алисы и у Боба есть закрытый (D_X) и открытый (D_Y) RSA-ключи. Предположим, что каждому из них известен открытый ключ другого. Способы передачи ключей мы рассмотрим позднее.

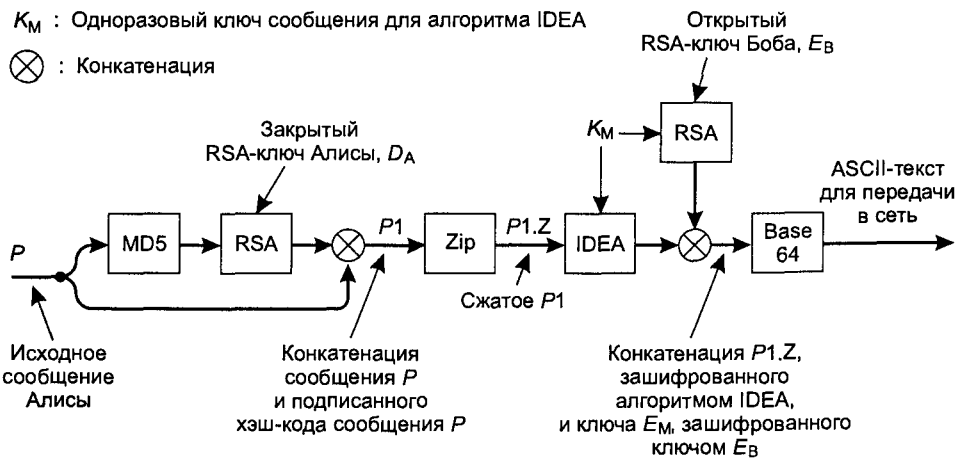


Рис. 8.40. Использование системы PGP для передачи сообщения

Алиса начинает с того, что запускает на своем компьютере программу PGP. Программа PGP сначала хэширует ее сообщение *P* с помощью алгоритма MD5, а затем шифрует полученный хэш-код при помощи ее закрытого RSA-ключа D_A. Получив это сообщение, Боб может расшифровать хэш-код открытым ключом Алисы и убедиться в его правильности. Даже если какой-либо злоумышленник мог получить хэш на этой стадии и расшифровать его известным открытым ключом Алисы, сила алгоритма MD5 гарантирует невозможность создания другого сообщения с тем же хэш-кодом (из-за трудоемкости вычислений).

Затем зашифрованный хэш-код и оригинальное сообщение объединяются в единое сообщение $P1$, которое сжимается с помощью программы ZIP, использующей алгоритм Зива—Лемпеля (Ziv—Lempel, 1977). Будем называть результат этого этапа $P1.Z$.

Затем программа PGP предлагает Алисе ввести случайную текстовую строку. При формировании 128-разрядного ключа сообщения K_M для алгоритма IDEA учитываются как содержимое, так и скорость ввода. (В PGP-литературе этот ключ назван сеансовым, что является неправильным употреблением термина, так как никакого сеанса нет.) Затем $P1.Z$ шифруется алгоритмом IDEA с помощью ключа K_M в режиме шифрованной обратной связи. Кроме того, ключ K_M шифруется открытым ключом Боба, E_B. Эти два компонента объединяются и преобразуются в кодировку base64, о которой уже рассказывалось в главе 7, когда мы говорили о стандартах MIME. Получающееся в результате сообщение содержит только буквы, цифры и символы +, * и =, что означает, что это сообщение может быть помещено в тело письма стандарта RFC 822, и можно надеяться, что оно прибудет к получателю без изменений.

Получив сообщение, Боб выполняет обратное преобразование Base64 и расшифровывает IDEA-ключ своим закрытым RSA-ключом. С помощью IDEA-ключа он расшифровывает сообщение и получает $P1.Z$. Распаковав zip-файл, Боб отделяет зашифрованный хэш-код от открытого текста и расшифровывает его открытым ключом Алисы. Если в результате обработки открытого текста алгоритмом MD5 получается тот же самый хэш-код, это означает, что сообщение P действительно пришло от Алисы.

Следует отметить, что алгоритм RSA используется здесь только в двух местах: для зашифровки 128-разрядного MD5-хэша и 128-разрядного IDEA-ключа. Алгоритм RSA медленный, но ему нужно зашифровать всего лишь 256 бит, что совсем немного. Более того, все эти 256 бит в высшей степени случайны, поэтому злоумышленнику придется очень сильно попотеть, чтобы угадать ключ. Основное шифрование выполняется алгоритмом IDEA, который на порядок быстрее, чем RSA. Итак, система PGP обеспечивает секретность, сжатие и цифровую подпись, и делает это намного эффективнее, чем схема, показанная на рис. 8.16.

Система PGP поддерживает четыре длины ключа RSA. Пользователь может самостоятельно выбирать нужную длину. Предлагаются следующие варианты длины:

1. Несерьезная (384 бит): шифр может быть взломан сегодня же организациями с большим бюджетом.
2. Коммерческая (512 бит): возможно, шифр смогут взломать организации из трех букв.
3. Военная (1024 бит): никто на Земле не сможет взломать этот шифр.
4. Межпланетная (2048 бит): никто во всей вселенной не сможет взломать шифр.

Поскольку алгоритм RSA используется только для двух небольших вычислений, всем следует всегда применять ключи межпланетного варианта, длиной 2048 бит.

Формат PGP-сообщения показан на рис. 8.41. Сообщение состоит из трех частей: области ключа, области подписи и области сообщения. Область ключа, помимо самого IDEA-ключа, содержит также идентификатор ключа, так как пользователям разрешено иметь несколько открытых ключей.

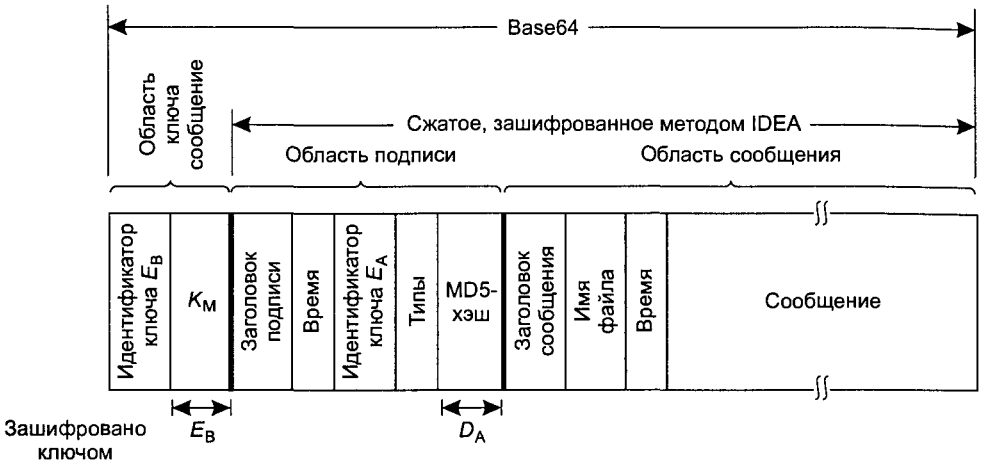


Рис. 8.41. PGP-сообщение

Область подписи содержит заголовок, который нас сейчас не интересует. За заголовком следует временной штамп, идентификатор открытого ключа отправителя, с помощью которого получатель сможет расшифровать хэш-код, используемый в качестве подписи. Следом идет идентификатор использованных алгоритмов шифрования и хэширования (чтобы можно было пользоваться, например, MD6 или RSA2, когда они будут разработаны). Последним в области подписи располагается сам зашифрованный хэш-код.

Часть сообщения также содержит заголовок, имя файла по умолчанию, на случай, если получатель будет сохранять принятое сообщение на диске, временной штамп создания сообщения и, наконец, само сообщение.

Работе с ключами в системе PGP было уделено особенное внимание, так как это ахиллесова пята всех систем защиты. У каждого пользователя локально находится две структуры данных: набор закрытых ключей и набор открытых ключей (эти наборы иногда называют связками). **Связка закрытых ключей** содержит несколько личных пар ключей, состоящих из закрытого и открытого ключей. Несколько пар ключей поддерживаются, чтобы позволить пользователям периодически их менять, когда возникают опасения, что тот или иной ключ скомпрометирован. При этом для смены ключа не требуется принимать каких-либо экстренных мер по передаче нового ключа. У каждой пары ключей есть связанный с ней идентификатор, так что отправителю нужно всего лишь сообщить получателю, которым открытым ключом был зашифрован ключ сообщения. Идентификатор сообщения состоит из младших 64 разрядов открытого ключа. За отсутствие конфликтов между идентификаторами ключей отвечают сами пользователи.

Закрытые ключи на диске зашифрованы специальным паролем (произвольной длины), защищающем их от кражи.

Связка открытых ключей содержит открытые ключи корреспондентов пользователей. Они нужны для зашифровки ключей сообщений, связанных с каждым сообщением. Каждая запись набора открытых ключей содержит не только ключ, но также его 64-разрядный идентификатор и отметку, указывающую степень доверия пользователя этому ключу.

Степень доверия ключу зависит, например, от способа его получения. Предположим, что открытые ключи расположены на электронных досках объявлений (BBS). Злоумышленник может атаковать доску объявлений и подменить размещенный там открытый ключ Боба своим ключом. Когда Алиса попытается воспользоваться подмененным ключом, злоумышленник сможет применить к Бобу атаку типа «человек посередине».

Чтобы предотвратить подобные атаки или хотя бы минимизировать их ущерб, Алисе необходимо знать, насколько она может доверять открытому ключу Боба, хранящемуся в ее наборе открытых ключей. Если Боб лично дал ей дискету с ключом, она может поставить такому ключу максимальную степень доверия. В этом и заключается децентрализованный, контролируемый пользователем подход к управлению открытыми ключами, отличающий PGP от централизованной схемы PKI.

Однако на практике открытые ключи часто получают, опрашивая доверенный сервер ключей. По этой причине после стандартизации X.509 система PGP стала поддерживать сертификаты наряду с традиционным для PGP механизмом связки открытых ключей. Все современные версии PGP имеют поддержку X.509.

PEM — почта повышенной секретности

В противоположность системе PGP, целиком созданной одним человеком, система **PEM** (Privacy Enhanced Mail — почта повышенной секретности) является официальным стандартом Интернета и описана в четырех RFC: с RFC 1421 по RFC 1424. Система PEM обладает примерно тем же набором функций, что и система PGP: секретность и аутентификация для систем электронной почты стандарта RFC 822. Тем не менее, она имеет несколько отличий от системы PGP в методах и технологии.

Сообщения, посылаемые с помощью системы PEM, сначала преобразуются в каноническую форму, удовлетворяющую особому набору правил, касающихся использования пробелов, табуляторов, символов возврата каретки и перевода строки. Затем с помощью алгоритма MD2 или MD5 вычисляется хэш-код сообщения. Потом конкатенация хэш-кода и сообщения шифруются при помощи алгоритма DES. В свете обсуждавшихся недостатков 56-разрядного ключа выбор именно этой системы шифрования, несомненно, вызывает большие подозрения. Затем зашифрованное сообщение преобразуется в кодировку base64 и передается получателю.

Как и в PGP, каждое сообщение шифруется одноразовым ключом, передаваемым вместе с сообщением. Ключ может быть защищен либо алгоритмом RSA, либо тройным применением системы DES в режиме EDE.

Управление ключами в системе PEM более структурированное, чем в PGP. Ключи сертифицируются по стандарту X.509 управлениями сертификации, организованными в виде жесткой иерархии с единым центром управления на ее вершине. Преимущество этой схемы в том, что сертификаты могут отзываться, для чего центр управления периодически публикует «черные списки».

С системой PEM связана только одна небольшая проблема: никто ею никогда не пользовался. Это связано, прежде всего, с политикой: как решить, кто и при каких условиях должен стать центром управления? Недостатка в кандидатах никогда не ощущалось, однако многие боятся доверять безопасность всей системы какой-либо одной компании. Наиболее серьезным кандидатом представлялась корпорация RSA Security, но она собиралась взимать плату за каждый выданный сертификат. Такая идея многим не понравилась. В частности, правительству США всегда было разрешено безвозмездно пользоваться американскими патентами, да и компании за пределами США привыкли бесплатно пользоваться алгоритмом RSA (разработчики забыли запатентовать его за пределами США). Ни те, ни другие не воодушевились внезапной необходимостью оплаты услуг RSA Security, которые они всегда получали бесплатно. В итоге центр управления так и не был выбран, а система PEM потерпела неудачу.

S/MIME

Следующим изобретением IETF в области обеспечения конфиденциальности электронной почты стала система под названием **S/MIME** (Secure/MIME — защищенный MIME). Она описывается в RFC с 2632 по 2643. Подобно PEM, она обеспечивает аутентификацию, целостность данных, секретность и проверку подлинности информации. Обладает неплохой гибкостью, поддерживает разнообразные криптографические алгоритмы. По названию можно догадаться, что S/MIME тесно связана с MIME в том смысле, что позволяет защищать любые типы сообщений. Определено множество новых заголовков MIME, например, для цифровых подписей.

Группа IETF определенно извлекла какие-то уроки из опыта PEM. В S/MIME нет жесткой иерархии сертификатов, отсутствует единый центр управления. Вместо этого пользователи могут работать с набором доверительных якорей. До тех пор, пока сертификат может быть проверен по доверительному якорю, он считается корректным. Система S/MIME использует стандартные алгоритмы и протоколы, которые мы уже рассматривали, поэтому на этом мы закончим ее обсуждение. Более подробную информацию вы найдете в RFC.

Защита информации во Всемирной паутине

Мы только что закончили изучение двух важных областей, в которых требуется защита информации, — соединения и электронная почта. Можно сказать, что это были аперитив и суп. Теперь же мы приступаем к главному блюду: защите ин-

формации во Всемирной паутине. Именно в WWW работает большинство злоумышленников, делая свое грязное дело. В следующих разделах будут рассмотрены некоторые проблемы, относящиеся к безопасности в Паутине.

Эту тему можно разделить на три части. Первая связана с безопасным именованием объектов и ресурсов. Вторая — с установлением аутентифицированных соединений. Третья — с тем, что случается, когда веб-сайт отправляет клиенту исполняемый код. После перечисления возможных опасностей мы рассмотрим все эти вопросы.

Возможные опасности

Практически каждую неделю газеты публикуют статьи о проблемах безопасности во Всемирной паутине. Ситуация складывается действительно довольно мрачная. Посмотрим на некоторые примеры того, что уже имело место. Во-первых, мы помним, как домашние страницы многочисленных организаций самых разных масштабов подвергались атакам хакеров и заменялись подложными страницами. (Термин «хакер» (hacker) приобрел значение «взломщик» благодаря журналистам, которые мало что понимали в компьютерном мире, но попытались воспользоваться профессиональным жаргоном программистов. На самом же деле изначально хакерами называли великих программистов. Взломщиков же мы и называем взломщиками (cracker).) В списке сайтов, которые удалось взломать, находятся такие, как Yahoo!, сайт Вооруженных сил США, ЦРУ, НАСА, а также New York Times. В большинстве случаев взломщики просто заменяли оригиналы на свои странички с каким-нибудь смешным (обычно издевательским) текстом, и уже через несколько часов сайты удавалось восстановить.

Однако были и гораздо более серьезные атаки. Многие сайты были сломаны за счет искусственно созданной чрезмерной нагрузки (атака типа «отказ в обслуживании», DoS), с которой заведомо не может справиться сервер. Зачастую такие нападения совершались сразу с нескольких машин, которые взломщику уже удалось сломать и заставить против воли участвовать в преступлении («распределенный DoS», DDoS). Такие атаки настолько распространены, что уже перестали быть новостью. Тем не менее, ущерб от них исчисляется тысячами долларов.

В 1999 году шведский взломщик проник на сайт Hotmail (корпорации Microsoft) и создал зеркало, на котором все желающие могли ввести имя любого пользователя этого сайта и прочесть всю его текущую почту и почтовые архивы.

А один русский 19-летний взломщик по имени Максим смог украсть с сайта, посвященного электронной коммерции, номера 300 000 кредитных карт. Затем он обратился к их владельцам и сообщил, что если они не заплатят ему 100 000 долларов, он опубликует номера кредиток в Интернете. Они не поддались на провокацию, и тогда он действительно опубликовал номера кредитных карт, что нанесло серьезный ущерб невинным жертвам.

23-летний студент из Калифорнии послал по электронной почте в агентство новостей фальшивый пресс-релиз, в котором сообщалось об огромных убытках корпорации Emulex и об уходе в отставку ее генерального директора. Спустя не-

сколько часов биржевые цены на акции Emulex снизились на 60 %, в результате чего их держатели лишились более 2 миллиардов долларов. Злоумышленник заработал около четверти миллиона долларов, продав акции незадолго до своего ложного заявления. Хотя в данном случае взлом не произошел непосредственно во Всемирной паутине, понятно, что объявление подобного рода, размещенное на сайте компании, привело бы к такому же эффекту.

К сожалению, одно перечисление таких примеров могло бы занять несколько страниц. И теперь нам пора обратиться к технической стороне дела. Более подробную информацию, касающуюся проблем безопасности всех видов, см. в (Anderson, 2001; Garfinkel и Spafford, 2002; Schneier, 2000). Поиск в Интернете также даст неплохие результаты.

Безопасное именование ресурсов

Начнем с чего-нибудь очень простого. Допустим, Алиса хочет посетить веб-сайт Боба. Она набирает в браузере URL, и через несколько секунд появляется страничка. Но в самом ли деле эта страничка создана Бобом? Может, да, а может, нет. Не исключено, что Труди снова принялась за свои шуточки. Например, она могла перехватить исходящие от Алисы пакеты и изучить их. Найдя запрос *GET* на получение страницы Боба, Труди могла сама зайти на эту страницу, изменить ее и отослать Алисе. Алиса не заметила бы ровным счетом ничего. Хуже того, Труди могла изменить цены на акции на более низкие, сделав тем самым предложение Боба очень привлекательным. Вероятность того, что теперь Алиса вышлет номер своей кредитной карты «Бобу» (с целью приобрести акции по выгодной цене), резко повысилась.

Одним из недостатков схемы «человек посередине» является то, что Труди должна быть в состоянии перехватывать исходящий трафик Алисы и подделывать свой исходящий трафик. На практике она должна прослушивать телефонную линию либо Боба, либо Алисы (поскольку прослушивание оптоволоконного кабеля — задача непростая). Это, конечно, возможно, но Труди не только умна и хитра, но и ленива. Она знает более простые способы обмануть Алису.

Обман DNS

Допустим, Труди может взломать систему DNS (например, ту ее часть, которая хранится в кэше DNS у провайдера Алисы) и заменить IP-адрес Боба (например, 36.1.2.3) своим IP-адресом (например, 42.9.9.9). Тогда можно провести атаку. То, как все должно работать в нормальной ситуации, показано на рис. 8.42, *а*: 1) Алиса запрашивает у службы DNS IP-адрес Боба; 2) получает его; 3) она запрашивает домашнюю страничку Боба; 4) получает ее. После того как Труди заменяет IP-адрес Боба на свой собственный, мы получаем ситуацию, показанную на рис. 8.42, *б*. Алиса ищет IP-адрес Боба, а получает вместо него IP-адрес злоумышленницы Труди, поэтому весь трафик Алисы, предназначенный для Боба, приходит, на самом деле, Труди. Та может организовать атаку типа «человек посередине», не мучаясь с установкой «крокодилов» на телефонной линии Алисы. Вместо этого она может заменить всего одну запись на сервере имен DNS. Это, согласитесь, более просто.

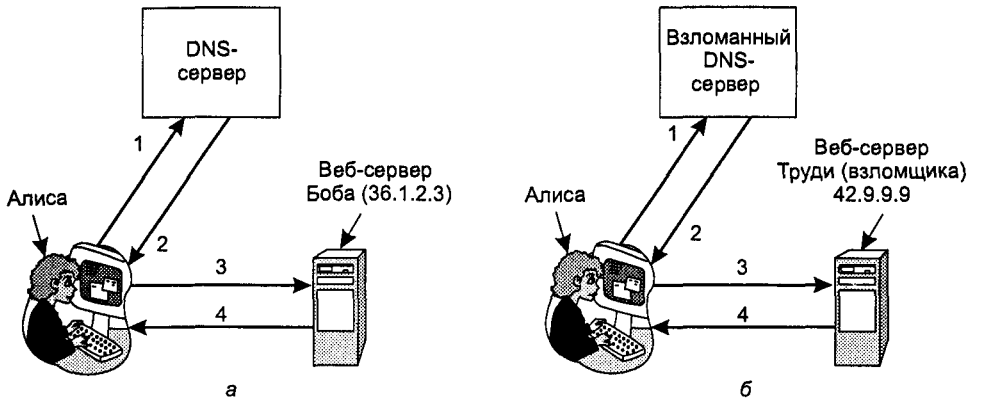


Рис. 8.42. Нормальная ситуация (*а*); атака со взломом DNS и изменением записи, относящейся к Бобу (*б*)

Как Труди удалось обмануть DNS? А это оказалось не таким уж сложным делом. Если не вдаваться в подробности, можно описать процесс так: Труди обманным путем заставляет DNS-сервер провайдера Алисы послать запрос для поиска адреса Боба. К несчастью, так как DNS использует UDP, сервер не может узнать, кто является реальным отправителем ответа. Труди использует это свойство, фальсифицируя ожидаемый ответ и тем самым занося неверные сведения об IP-адресе Боба в кэш DNS-сервера. Для простоты мы будем предполагать, что провайдер Алисы изначально не имеет сведений о веб-сайте Боба, bob.com. Если же такие сведения есть, злоумышленник может выждать, пока срок действия записи истечет, и попробовать еще раз (либо применить другие хитрости).

Труди начинает свою атаку с того, что посылает провайдеру Алисы запрос на поиск IP-адреса bob.com. Так как соответствующая запись отсутствует, сервер, в свою очередь, опрашивает сервер домена верхнего уровня (.com). Но Труди опережает этот сервер и посылает ложный ответ, в котором сообщается, что IP-адрес bob.com якобы 42.9.9.9. Как мы знаем, в реальности это адрес Труди. Так как этот ответ приходит первым, данные из него заносятся в кэш сервера провайдера, а настоящий ответ, если он приходит позже, отвергается. Установка ложного IP-адреса называется **обманом DNS**. А кэш, в котором хранится заведомо ложный IP-адрес, называется **отравленным кэшем**.

Надо сказать, что на практике все не так просто. Во-первых, провайдер Алисы все-таки проверяет наличие в ответе правильного адреса сервера верхнего уровня. Но Труди может написать в соответствующем поле что угодно и преодолеть эту преграду. Учитывая то, что адреса серверов верхнего уровня общедоступны, сделать это несложно.

Во-вторых, для того чтобы DNS-сервер мог понять, какому запросу соответствует ответ, во все запросы добавляются порядковые номера. Чтобы обмануть

провайдера Алисы, Труди должна знать текущий порядковый номер. Самый простой способ узнать его — это зарегистрировать собственный домен, например, trudy-the-intruder.com.

Предположим, что IP-адрес этого домена также 42.9.9.9. Труди создает DNS-сервер для этого домена: dns.trudy-the-intruder.com. Его IP-адрес тот же самый (42.9.9.9), поскольку оба домена расположены на одном и том же компьютере. Теперь надо заставить провайдера Алисы поинтересоваться DNS-сервером Труди. Сделать это несложно. Требуется лишь запросить, например, foobar.trudy-the-intruder.com, и серверу провайдера Алисы придется опросить сервер верхнего уровня, .com, и узнать у него, кто обслуживает новый домен Труди.

И вот теперь, когда запись dns.trudy-the-intruder.com занесена в кэш провайдера, можно спокойно начинать атаку. Труди запрашивает у провайдера Алисы www.trudy-the-intruder.com, а тот в ответ посылает на DNS-сервер Труди соответствующий запрос. Вот в этом-то запросе и содержится нужный злоумышленнице порядковый номер. Теперь Труди должна действовать без промедления: она ищет с помощью провайдера Алисы Боба и тут же отвечает на собственный вопрос, посылая фальшивку: «Адрес bob.com: 42.9.9.9». Этот подделанный ответ несет в себе порядковый номер на единицу больше только что полученного. За время атаки она может послать еще одну фальшивку, с номером, на два больше полученного, а также еще около дюжины таких «ответов» с увеличивающимися номерами. Задача одного из них нам уже ясна. Остальные никому не нужны, их просто выкинут. После прибытия фальшивого ответа на запрос Алисы он будет помещен в кэш; к тому времени, когда доберется настоящий ответ, он будет отвергнут, так как сервер уже ничего не ожидает.

И вот Алиса ищет IP-адрес bob.com и узнает, что он равен 42.9.9.9. Как мы знаем, это адрес Труди, которая провела успешную атаку типа «человек посередине», не выходя из своей комнаты. Последовательность предпринятых ею шагов показана на рис. 8.43. К сожалению, это еще и не единственный способ обмануть DNS. Этих способов действительно много.

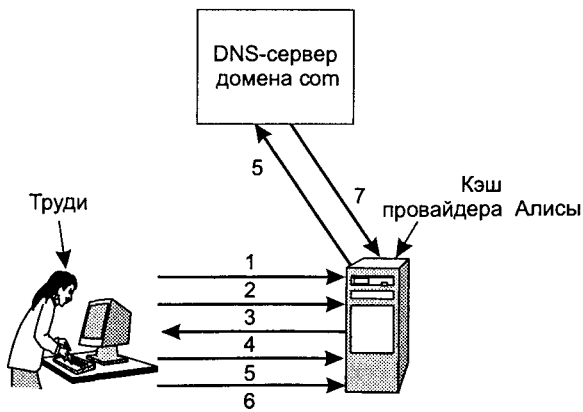


Рис. 8.43. Обман провайдера Алисы

Защита DNS

Избежать атак описанного типа можно, заставив DNS-серверы использовать случайные идентификаторы в своих запросах вместо того, чтобы просто инкрементировать их. К сожалению, заткнув одну «дыру», мы обнаруживаем другую. Настоящая проблема в том, что служба DNS разрабатывалась в те времена, когда Интернет был чисто исследовательской сетью, работавшей в нескольких сотнях университетов, и ни Алиса, ни Боб, ни Труди на этот праздник жизни приглашены не были. Вопрос защиты информации тогда еще не стоял; задачей-максимум было заставить Интернет функционировать. Но с годами среда, в которой приходилось выживать Интернету, сильно изменилась, поэтому в 1994 году IETF основала рабочую группу, задачей которой было защитить DNS. Этот проект известен под названием **DNSsec (Защита DNS)**; результаты работы группы опубликованы в RFC 2535. К сожалению, DNSsec до сих пор не удается развернуть в больших масштабах, поэтому многие DNS-серверы продолжают подвергаться нападениям злоумышленников.

Концептуально система DNSsec очень проста. Она основана на шифровании с открытыми ключами. Каждая зона DNS (в терминах рис. 7.2) обладает парой ключей, а именно, открытым и закрытым. Вся информация, отправляемая DNS-сервером, подписывается с помощью закрытого ключа зоны отправителя, поэтому аутентичность может быть запросто проверена принимающей стороной.

DNSsec предоставляет три основные услуги:

1. Подтверждение места отправления данных.
2. Распространение открытых ключей.
3. Аутентификацию транзакций и запросов.

Самой главной является первая услуга, с ее помощью проверяется то, что пришедшие данные были подтверждены их отправителем. Вторая услуга полезна для безопасного хранения и извлечения открытых ключей. Третья позволяет защититься от атак повторного воспроизведения и обмана сервера. Обратите внимание: секретность здесь не обеспечивается, этой задачи нет, поскольку вся информация DNS считается открытой. Так как процесс введения DNSsec в строй, скорее всего, будет продолжаться в течение нескольких лет, важно предоставить возможность общения между собой серверам, снабженным системой защиты и не снабженным ею. Это неявно подразумевает, что протокол изменять нельзя. Рассмотрим теперь эту систему чуть более детально.

Записи DNS группируются в наборы, называемые **RRSet** (Resource Record Set — набор записей ресурсов). В набор входят все записи с одинаковыми именами, классами и типами. Скажем, в наборе может быть несколько записей *A*, если имя DNS соответствует первичному и вторичному IP-адресам. Наборы расширяются за счет некоторых новых типов записей (обсуждаются далее). Каждый RRSet хэшируется (например, с использованием MD-5 или SHA-1). Хэш подписывается при помощи закрытого ключа зоны (например, по алгоритму RSA). Единицей передаваемой клиентам информации является подписанный RRSet. Получив его, клиент может проверить, действительно ли для генерации подписи

был взят закрытый ключ зоны отправителя. Если подпись корректна, данные принимаются. Так как каждый RRSet содержит собственную подпись, наборы можно кэшировать где угодно, даже на не слишком надежных серверах, не опасаясь за их судьбу.

Система DNSec вводит несколько новых типов записей. Первая из них — это запись *KEY*. В ней хранятся открытый ключ зоны, пользователя, хоста или другого принципала, криптографический алгоритм генерации подписи, наименование протокола передачи и еще несколько бит. Открытый ключ хранится в незащищенном виде. Сертификаты X.509 не используются из-за их громоздкости. В поле алгоритма рекомендуемое значение, соответствующее MD5/RSA, равно 1, для других комбинаций используются другие значения. Поле протокола может указывать на использование IPsec или другого протокола защиты соединений (если таковой вообще применяется).

Второй новый тип записей — *SIG*. В такой записи содержится подписанный хэш, сформированный в соответствии с алгоритмом, указанным в *KEY*. Подпись охватывает все записи RRSet, включая все записи *KEY*, однако не включая саму себя. Здесь также содержатся время начала и конца действия подписи, имя владельца подписи и некоторая дополнительная информация.

Система DNSsec устроена так, что закрытый ключ зоны может храниться в автономном режиме. Один или два раза в день содержимое базы данных зоны можно вручную переносить (например, записав компакт-диск) на машину, работающую в автономном режиме и хранящую закрытый ключ. Там можно сгенерировать подписи для всех наборов, и полученные таким образом записи *SIG* можно снова записать на компакт-диск и перенести на главный сервер. Таким образом, закрытый ключ можно хранить на компакт-диске, запертом в сейфе и вынимаемом только для того, чтобы подписать на автономной машине ежедневное обновление наборов типа RRSet. По окончании генерации подписей все копии ключа удаляются из памяти, а диск и компакт-диск возвращаются в сейф. Эта процедура превращает электронную защиту информации в физическую, что гораздо понятнее пользователям.

Метод предварительного подписания наборов значительно ускоряет процесс обработки запросов, так как отпадает необходимость в шифровании «на лету». Платой за это является большой объем дискового пространства, необходимого для хранения всех ключей и подписей в базах данных DNS. Из-за этого некоторые записи увеличиваются в размере десятикратно.

Получив подписанный RRSet, клиент должен применить открытый ключ зоны для расшифровки хэша, затем вычислить хэш самостоятельно и сравнить два значения. В случае их соответствия данные считаются корректными. Тем не менее, эта процедура не решает вопрос получения клиентом открытого ключа зоны. Одним из способов является запрос ключа у надежного сервера и передача его по защищенному соединению (например, при помощи IPsec).

Однако на практике предполагается, что у клиентов уже есть открытые ключи всех доменов верхнего уровня. Если Алиса пожелает посетить сайт Боба, она запросит у службы DNS набор RRSet для bob.com, в котором будет содержаться IP-адрес и запись *KEY* с открытым ключом Боба. RRSet будет подписан доменом

Защита DNS

Избежать атак описанного типа можно, заставив DNS-серверы использовать случайные идентификаторы в своих запросах вместо того, чтобы просто инкрементировать их. К сожалению, заткнув одну «дыру», мы обнаруживаем другую. Настоящая проблема в том, что служба DNS разрабатывалась в те времена, когда Интернет был чисто исследовательской сетью, работавшей в нескольких сотнях университетов, и ни Алиса, ни Боб, ни Труди на этот праздник жизни приглашены не были. Вопрос защиты информации тогда еще не стоял; задачей-максимум было заставить Интернет функционировать. Но с годами среда, в которой приходилось выживать Интернету, сильно изменилась, поэтому в 1994 году IETF основала рабочую группу, задачей которой было защитить DNS. Этот проект известен под названием **DNSsec (Защита DNS)**; результаты работы группы опубликованы в RFC 2535. К сожалению, DNSsec до сих пор не удается развернуть в больших масштабах, поэтому многие DNS-серверы продолжают подвергаться нападениям злоумышленников.

Концептуально система DNSsec очень проста. Она основана на шифровании с открытыми ключами. Каждая зона DNS (в терминах рис. 7.2) обладает парой ключей, а именно, открытым и закрытым. Вся информация, отправляемая DNS-сервером, подписывается с помощью закрытого ключа зоны отправителя, поэтому аутентичность может быть запросто проверена принимающей стороной.

DNSsec предоставляет три основные услуги:

1. Подтверждение места отправления данных.
2. Распространение открытых ключей.
3. Аутентификацию транзакций и запросов.

Самой главной является первая услуга, с ее помощью проверяется то, что пришедшие данные были подтверждены их отправителем. Вторая услуга полезна для безопасного хранения и извлечения открытых ключей. Третья позволяет защититься от атак повторного воспроизведения и обмана сервера. Обратите внимание: секретность здесь не обеспечивается, этой задачи нет, поскольку вся информация DNS считается открытой. Так как процесс введения DNSsec в строй, скорее всего, будет продолжаться в течение нескольких лет, важно предоставить возможность общения между собой серверам, снабженным системой защиты и не снабженным ею. Это неявно подразумевает, что протокол изменять нельзя. Рассмотрим теперь эту систему чуть более детально.

Записи DNS группируются в наборы, называемые **RRSet** (Resource Record Set — набор записей ресурсов). В набор входят все записи с одинаковыми именами, классами и типами. Скажем, в наборе может быть несколько записей *A*, если имя DNS соответствует первичному и вторичному IP-адресам. Наборы расширяются за счет некоторых новых типов записей (обсуждаются далее). Каждый RRSet хэшируется (например, с использованием MD-5 или SHA-1). Хэш подписывается при помощи закрытого ключа зоны (например, по алгоритму RSA). Единицей передаваемой клиентам информации является подписанный RRSet. Получив его, клиент может проверить, действительно ли для генерации подписи

был взят закрытый ключ зоны отправителя. Если подпись корректна, данные принимаются. Так как каждый RRSet содержит собственную подпись, наборы можно кэшировать где угодно, даже на не слишком надежных серверах, не опасаясь за их судьбу.

Система DNSec вводит несколько новых типов записей. Первая из них — это запись *KEY*. В ней хранятся открытый ключ зоны, пользователя, хоста или другого принципала, криптографический алгоритм генерации подписи, наименование протокола передачи и еще несколько бит. Открытый ключ хранится в незащищенном виде. Сертификаты X.509 не используются из-за их громоздкости. В поле алгоритма рекомендуемое значение, соответствующее MD5/RSA, равно 1, для других комбинаций используются другие значения. Поле протокола может указывать на использование IPsec или другого протокола защиты соединений (если таковой вообще применяется).

Второй новый тип записей — *SIG*. В такой записи содержится подписанный хэш, сформированный в соответствии с алгоритмом, указанным в *KEY*. Подпись охватывает все записи RRSet, включая все записи *KEY*, однако не включая саму себя. Здесь также содержатся время начала и конца действия подписи, имя владельца подписи и некоторая дополнительная информация.

Система DNSsec устроена так, что закрытый ключ зоны может храниться в автономном режиме. Один или два раза в день содержимое базы данных зоны можно вручную переносить (например, записав компакт-диск) на машину, работающую в автономном режиме и хранящую закрытый ключ. Там можно сгенерировать подписи для всех наборов, и полученные таким образом записи *SIG* можно снова записать на компакт-диск и перенести на главный сервер. Таким образом, закрытый ключ можно хранить на компакт-диске, запертом в сейфе и вынимаемом только для того, чтобы подписать на автономной машине ежедневное обновление наборов типа RRSet. По окончании генерации подписей все копии ключа удаляются из памяти, а диск и компакт-диск возвращаются в сейф. Эта процедура превращает электронную защиту информации в физическую, что гораздо понятнее пользователям.

Метод предварительного подписания наборов значительно ускоряет процесс обработки запросов, так как отпадает необходимость в шифровании «на лету». Платой за это является большой объем дискового пространства, необходимого для хранения всех ключей и подписей в базах данных DNS. Из-за этого некоторые записи увеличиваются в размере десятикратно.

Получив подписанный RRSet, клиент должен применить открытый ключ зоны для расшифровки хэша, затем вычислить хэш самостоятельно и сравнить два значения. В случае их соответствия данные считаются корректными. Тем не менее, эта процедура не решает вопрос получения клиентом открытого ключа зоны. Одним из способов является запрос ключа у надежного сервера и передача его по защищенному соединению (например, при помощи IPsec).

Однако на практике предполагается, что у клиентов уже есть открытые ключи всех доменов верхнего уровня. Если Алиса пожелает посетить сайт Боба, она запросит у службы DNS набор RRSet для bob.com, в котором будет содержаться IP-адрес и запись *KEY* с открытым ключом Боба. RRSet будет подписан доменом

верхнего уровня (com), поэтому Алиса запросто сможет проверить подлинность набора. Пример содержимого набора RRSet приведен в табл. 8.4.

Таблица 8.4. Пример набора RRSet для bob.com. Запись KEY содержит открытый ключ Боба. Запись SIG — это хэш A и KEY, подписанный сервером домена верхнего уровня (com) для проверки их аутентичности

Имя домена	Время жизни	Класс	Тип	Значение
bob.com	86400	IN	A	36.1.2.3
bob.com	86400	IN	KEY	3682793A7B73F731029CE2737D...
bob.com	86400	IN	SIG	86947503A8B848F5272E53930C...

Теперь, вооружившись заверенной копией открытого ключа Боба, Алиса может узнать у DNS-сервера Боба IP-адрес www.bob.com. Этот RRSet будет подписан закрытым ключом Боба, поэтому Алиса сможет проверить подлинность подписи возвращенного Бобом набора. Если злоумышленнику каким-то образом удастся внедрить фальшивый RRSet в один из кэшей, Алиса заметит это, так как запись *SIG* будет неправильной.

Тем не менее, DNSsec также предоставляет криптографический механизм для связывания ответов с соответствующими запросами, предотвращающий атаки типа той, что показана на рис. 8.43. Эта (необязательная) мера заключается в добавлении к ответу хэша запроса, подписанного закрытым ключом опрашиваемого. Поскольку Труди неизвестен закрытый ключ сервера верхнего уровня (домена com), она не сможет подделать ответ этого сервера на запрос провайдера Алисы. Она, конечно, может опередить настоящий ответ, но фальшивка будет замечена по неправильной подписи хэшированного запроса.

DNSsec поддерживает и некоторые другие типы записей. Так, для хранения сертификатов (например, стандарта X.509) можно использовать запись *CERT*. Зачем нужна такая запись? Дело в том, что есть желающие превратить DNS в инфраструктуру PKI. Случится это на самом деле или нет, пока еще неизвестно. На этом мы заканчиваем обсуждение DNSsec. Более подробную информацию вы найдете в RFC 2535.

Самозаверяющиеся имена

Защита DNS — это не единственный способ защиты имен. Совершенно другой подход применяется в **защищенной файловой системе** (Mazi res и др., 1999). Авторами этого проекта была создана надежная, масштабируемая файловая система мирового масштаба, не требующая внесения изменений в структуру DNS, не использующая сертификаты и не подразумевающая существование инфраструктуры PKI. В этом разделе мы покажем, как эти идеи можно применить во Всемирной паутине. Соответственно, мы будем пользоваться веб-терминологией, а не понятиями файловых систем (именно последними оперирует документация, описывающая защищенную файловую систему). Однако во избежание недоразумений мы считаем своим долгом предупредить, что эта схема *может быть* применена в веб-технологиях для обеспечения безопасности, однако в данное время

она еще не используется, а для ее внедрения потребуется внести серьезные изменения в программное обеспечение.

Мы начнем с предположения о том, что каждый веб-сервер имеет два ключа: открытый и закрытый. Суть идеи состоит в том, чтобы каждый URL содержал хэш имени сервера и открытого ключа. Например, на рис. 8.44 мы видим URL фотографии Боба. Он начинается с традиционного http://, за которым следует имя сервера (www.bob.com). Далее ставится двоеточие, а за ним — 32-символьный хэш. В конце мы видим обычное имя файла. Если исключить хэш, получится вполне обычный URL. Вместе с хэшем он образует **самозаверяющийся URL**.

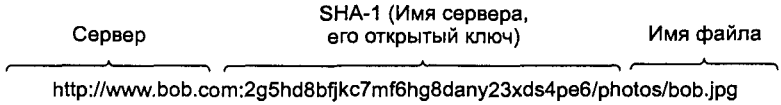


Рис. 8.44. Самозаверяющийся URL, содержащий хэш имени сервера и его открытого ключа

Сразу же возникает вопрос: для чего служит хэш? Он вычисляется конкатенацией (объединением) DNS-имени сервера с его открытым ключом и прогоном получающейся строки через функцию SHA-1, в результате чего получается 160-разрядный хэш. В данной схеме значение хэш-функции представлено последовательностью из 32 цифр и букв нижнего регистра. Из указанного алфавита во избежание путаницы исключены буквы «l» и «o» и цифры «1» и «0». В распоряжении остаются 32 символа, которые можно кодировать 5-разрядными битовыми последовательностями. В результате мы как раз и получаем $32 \cdot 5 = 160$ бит хэша SHA-1. На самом деле, вовсе не обязательно использовать хэш-функцию. Вместо этого можно вставлять и сам открытый ключ. Преимущество применения хэша заключается в том, что по сравнению с незашифрованным ключом уменьшается длина имени ресурса.

В простейшем (но наименее удобном) варианте Алисе, чтобы посмотреть фотографию Боба, придется набрать всю строку, показанную на рис. 8.44. Браузер отправляет на веб-сайт Боба просьбу прислать открытый ключ. По прибытии этого ключа браузер объединяет имя сервера с ключом и вычисляет хэш-функцию. Если результат согласуется с 32-символьным хэшем из защищенного URL, то у браузера не остается сомнений в том, что у него есть подлинный ключ Боба. В конце концов, даже если Труди перехватит запрос и подделает ответ, ей не удастся подобрать открытый ключ для получения корректного хэша благодаря свойствам алгоритма SHA-1. Любые попытки подлога будут сразу же выявлены. Что касается открытого ключа Боба, его можно поместить в кэш для будущего использования.

Теперь Алисе необходимо проверить соответствующий закрытый ключ Боба. Она формирует сообщение, содержащее AES-ключ сеанса, нонс и временной штамп. Затем она шифрует это сообщение с помощью открытого ключа Боба и отправляет его ему. Так как подходящий закрытый ключ есть только у Боба, только он сможет расшифровать сообщение и отослать назад нонс, зашифрованный ключом AES. Принятый Алисой корректный нонс убеждает ее в том, что она действительно разговаривает с Бобом. Теперь и у Алисы, и у Боба есть AES-

ключ сеанса, которым они будут пользоваться при дальнейшем обмене запросами и ответами.

Алисе достаточно один раз загрузить фотографию Боба (или любую веб-страницу). После этого она может поставить на данный ресурс закладку, и ей больше не придется вручную набирать всю эту огромную строку URL. Более того, указатели URL, расположенные на веб-страницах, также могут быть самозаверяющимися, и ими можно пользоваться обычным образом (просто щелкая на них мышкой). Однако такие указатели гарантируют, что вы получите настоящую, а не поддельную информацию. Еще один способ избежать ручного набора длинных строк URL — использовать для их получения защищенное соединение с надежным сервером или вписывать их в сертификаты X.509, подписанные управлением сертификации.

Другой путь получения самозаверяющихся URL заключается в установлении соединения с надежной поисковой машиной. Вначале придется набрать длинную строку, но затем можно будет воспользоваться описанным ранее протоколом, приводящим к защищенному, аутентифицированному соединению с надежной поисковой машиной. Результатом поиска будет страница (с электронной подписью), содержащая список нужных самозаверяющихся URL, на которых можно щелкать вместо того, чтобы вводить вручную в строке адреса в браузере.

Давайте теперь посмотрим, как этот подход противостоит обманным действиям Труди. Если ей удастся «отравить» кэш провайдера Алисы, запрос последней будет против ее воли направлен не к Бобу, а к злоумышленнице. Но наш протокол требует, чтобы получатель начального сообщения (то есть Труди) вернул открытый ключ, способный породить корректный кэш. Если Труди вернет открытый ключ Боба, Алиса не заметит подлога, но зашифрует свое следующее сообщение ключом Боба. Труди это сообщение получит, но расшифровать его не сможет. Следовательно, она не сможет расшифровать ни AES-ключ, ни нонс. В общем, худшее, что в этом случае можно сделать путем обмана DNS, это осуществить атаку типа DoS (отказ в обслуживании).

SSL — протокол защищенных сокетов

Ну что ж, защита имен ресурсов — это неплохое начало, однако этим не обеспечивается в полной мере безопасность во Всемирной паутине. Следующий шаг состоит в установлении безопасных соединений. Сейчас мы рассмотрим, как это делается.

Когда веб-технологии были впервые представлены широкой публике, они использовались для распространения статических страниц. Однако уже давным-давно некоторые компании задумались об использовании Паутины для выполнения финансовых транзакций, таких как покупка товаров по кредитным картам, онлайновые банковские операции, электронная торговля ценными бумагами. Для таких приложений требовалась организация защищенных соединений. В 1995 году тогдашний лидер среди производителей браузеров, корпорация Netscape Communications, в ответ на это представила систему безопасности под названием **SSL** (Secure Sockets Layer — протокол защищенных сокетов). Соответствую-

щее программное обеспечение, как и сам протокол, в наше время используется очень широко (в том числе и программой Internet Explorer), поэтому стоит рассмотреть SSL более детально.

Итак, SSL создает защищенное соединение между двумя сокетами, позволяющее:

- клиенту и серверу договориться об используемых параметрах;
- клиенту и серверу произвести взаимную аутентификацию;
- организовать тайное общение;
- обеспечить защиту целостности данных.

Все перечисленные пункты нам уже знакомы, поэтому мы не будем их комментировать.

Расположение SSL в структуре обычного стека протоколов показано на рис. 8.45. По сути дела, между прикладным и транспортным уровнями появляется новый уровень, принимающий запросы от браузера и отсылающий их по TCP для передачи серверу. После установки защищенного соединения основная задача SSL заключается в поддержке сжатия и шифрования. Если поверх SSL используется HTTP, этот вариант называется **HTTPS** (Secure HTTP — защищенный HTTP) несмотря на то, что это обычный протокол HTTP. Впрочем, возможно и отличие: скажем, доступ может осуществляться через новый порт (443) вместо стандартного (80). Кстати говоря, область применения SSL не ограничивается исключительно веб-браузерами, но это наиболее распространенное применение.



Рис. 8.45. Уровни (и протоколы), используемые обычным домашним браузером с SSL

Существует несколько версий протокола SSL. Далее мы будем обсуждать только версию 3, так она распространена наиболее широко. SSL поддерживает множество разных алгоритмов и может обладать разными дополнительными функциями, среди которых наличие или отсутствие сжатия, тот или иной алгоритм шифрования, а также некоторые вещи, связанные с ограничениями экспорта в криптографии. Последнее, в основном, предназначено для того, чтобы можно было

удостовериться, что оба конца соединения находятся в США. Иногда длину ключа ограничивают 40 битами, что шифровальщики воспринимают как своего рода шутку. Однако Netscape должен был ввести это ограничение, чтобы получить лицензию на экспорт от правительства США.

SSL состоит из двух субпротоколов, один из которых предназначен для установления защищенного соединения, а второй — для использования этого соединения. Начнем с рассмотрения вопроса установления соединения. Работа субпротокола, занимающегося этим, показана на рис. 8.46. Все начинается с сообщения 1, в котором Алиса посылает Бобу запрос на установку соединения. В нем указываются версия SSL, а также предпочтения Алисы относительно сжатия и алгоритмов шифрования. Также в нем содержится нонс R_A, который будет использован впоследствии.

Теперь наступает очередь Боба. В сообщении 2 он выбирает один из алгоритмов, поддерживаемых Алисой, и посылает собственный нонс R_B. В сообщении 3 он отсылает сертификат со своим открытым ключом. Если сертификат не подписан какой-нибудь уважаемой организацией, он также отправляет цепочку сертификатов, по которым Алиса может удостовериться в том, что сертификату Боба действительно можно доверять. Все браузеры, включая тот, что установлен у Алисы, изначально снабжаются примерно сотней открытых ключей, поэтому если среди присланных Бобом сертификатов встретится один из этих ключей, Алиса сможет по нему восстановить ключ Боба и проверить его. В этот момент Боб может прислать и другие сообщения (например, запрос на получение сетификата Алисы с ее открытым ключом). После окончания выполнения своей части протокола Боб посылает сообщение 4, в котором говорит, что настала очередь Алисы.

Алиса в ответ выбирает 384-разрядный **подготовительный ключ** и посылает его Бобу, зашифровав предварительно своим открытым ключом (сообщение 5). Настоящий ключ сеанса вычисляется при помощи подготовительного ключа и нонсов обеих сторон. Это довольно сложная процедура. После получения сообщения 5 и Алиса, и Боб могут вычислить ключ сеанса. Для этого Алиса просит Боба переключиться на новый шифр (сообщение 6), а также сообщает о том, что она считает субпротокол установления соединения оконченным (сообщение 7). Боб соглашается с ней (сообщения 8 и 9).

Однако несмотря на то, что Алиса знает, кто такой Боб, последний Алису не знает (если только у нее нет открытого ключа и сертификата к нему, что довольно необычно для обычного физического лица). Поэтому первым сообщением для Алисы запросто может оказаться просьба пройти регистрацию, используя полученные ранее имя пользователя и пароль. Впрочем, протокол регистрации в системе не выходит за область полномочий SSL. Так или иначе, по окончании этой серии запросов-подтверждений может начинаться передача данных.

Как уже говорилось, SSL поддерживает разнообразные криптографические алгоритмы. Наиболее сильный из них использует для шифрации тройной DES с тремя отдельными ключами и SHA-1 для обеспечения целостности данных. Такое сочетание алгоритмов работает довольно медленно, поэтому применяется в основном при выполнении банковских операций и в других приложениях, в которых

требуется высокий уровень защиты. В обычных приложениях электронной коммерции для шифрации применяется 128-разрядный ключ, а для аутентификации — MD5. В качестве исходных данных RC4 передается 128-разрядный ключ, который разрастается во много раз при работе алгоритма. Это внутреннее число используется для создания ключевого потока. Последний суммируется по модулю 2 с открытым текстом, в результате чего получается обычный потоковый шифр, как было показано на рис. 8.12. Экспортные версии алгоритма также работают с алгоритмом RC4 и 128-разрядным ключом, однако 88 из этих разрядов делаются открытыми, что позволяет довольно быстро взломать шифр.

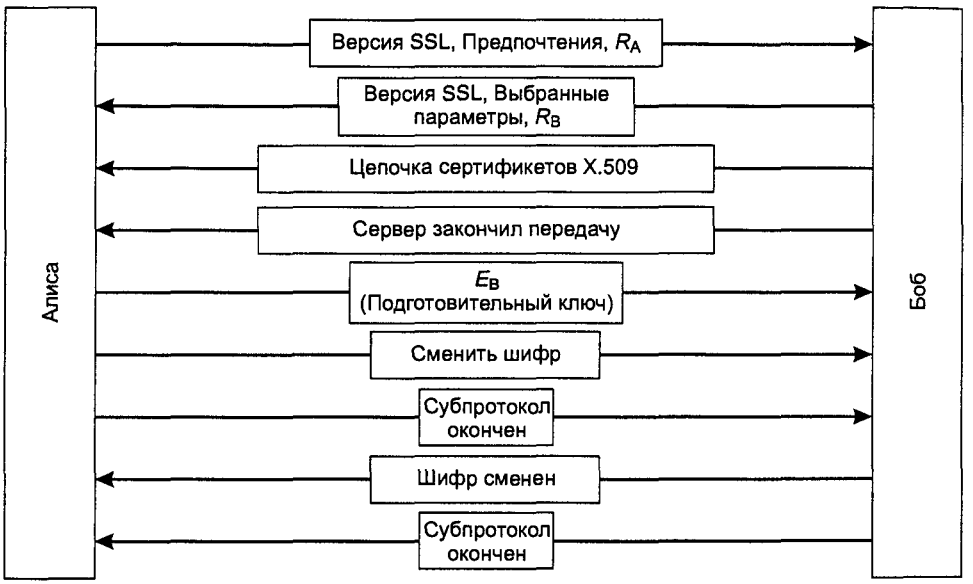


Рис. 8.46. Упрощенный вариант субпротокола SSL установления соединения

Для реальной передачи данных используется второй субпротокол, показанный на рис. 8.47. Сообщения, поступающие от браузера, разбиваются на единицы данных размером до 16 Кбайт. Если сжатие включено, каждая из этих единиц независимо сжимается. Затем по двум нонсам вычисляется закрытый ключ, подготовительный ключ объединяется со сжатым текстом и результат хэшируется по согласованному алгоритму (чаще всего MD5). Хэш добавляется к каждому фрагменту в виде MAC (Message Authetication Code — код аутентификации сообщения). Этот сжатый фрагмент вместе с MAC кодируется согласованным алгоритмом с симметричным ключом (обычно это суммирование по модулю 2 с ключевым потоком RC4). Наконец, присоединяется заголовок фрагмента, и фрагмент передается по TCP-соединению.

Следует остерегаться следующего подводного камня: уже говорилось о том, что RC4 имеет некоторые слабые ключи, которые довольно просто взламываются, поэтому SSL с RC4 — это довольно шаткая основа (Fluhrer и др., 2001). Браузеры, позволяющие пользователю выбирать тот или иной шифр, лучше всего

настраивать на постоянное использование тройного алгоритма DES со 168-разрядными ключами и SHA-1 невзирая на то, что такая комбинация работает еще медленнее, чем RC4 + MD5.

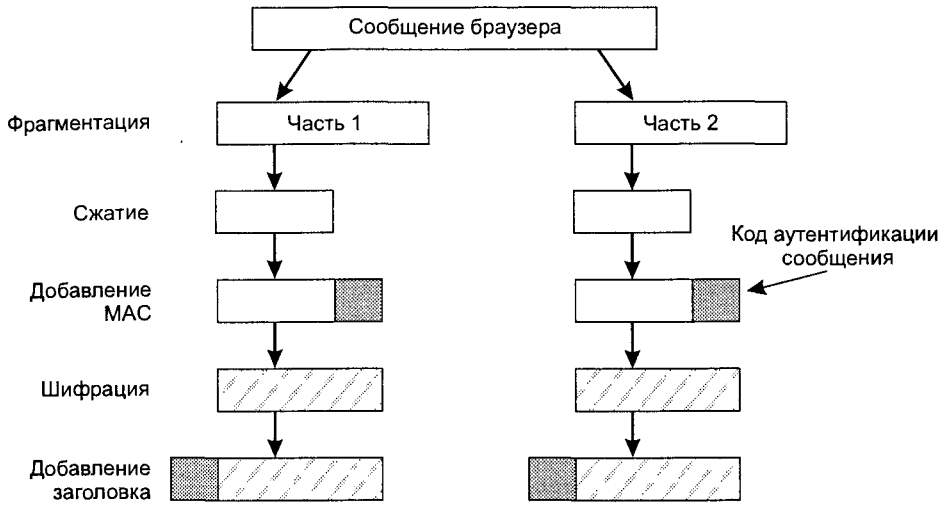


Рис. 8.47. Передача данных при использовании SSL

С SSL связана еще одна проблема: у принципалов может не быть сертификатов, а даже если они есть, далеко не всегда производится проверка соответствия ключей и сертификатов.

В 1996 году корпорация Netscape Communications направила SSL на стандартизацию в IETF. Результатом стал стандарт **TLS** (Transport Layer Security — защита транспортного уровня). Он описан в RFC 2246.

В SSL при создании стандарта TLS было внесено не так уж много изменений, однако их оказалось достаточно для того, чтобы SSL версии 3 и TLS стали несовместимыми. Например, в целях усиления ключа был изменен способ вычисления ключа сеанса по подготовительному ключу и нонсам. TLS также известен как SSL версии 3.1. Первая реализация появилась в 1999 году, однако до сих пор не очень понятно, заменит ли TLS SSL, даже учитывая то, что TLS несколько надежнее. Проблема с ключами RC4, между прочим, никуда не исчезла.

Защита переносимых программ

Именование ресурсов и соединения — это две области, которые, несомненно, тесно связаны с защитой информации во Всемирной паутине. Однако существуют и другие, не менее важные вопросы, связанные с той же темой. Поначалу веб-страницы представляли собой полностью статические HTML-файлы и не содержали исполняемый код. Теперь же на веб-страницах очень часто встречаются небольшие программы: Java-апплеты, управляющие элементы ActiveX, скрипты JavaScript. Загрузка и выполнение таких **переносимых программ**, очевидно, связаны

с большим риском возникновения массовых атак. Были разработаны различные методы, направленные на минимизацию этого риска. Далее мы обозначим некоторые вопросы, связанные с проблемами защиты переносимых программ.

Защита Java-апплетов

Java-апплеты — это небольшие программы на языке Java, откомпилированные в машинный язык со стековой организацией под названием **JVM** (Java Virtual Machine — виртуальная машина Java). Такие программы могут размещаться на веб-странице и загружаться вместе с ней. После загрузки страницы апплеты обрабатываются интерпретатором JVM в браузере, как показано на рис. 8.48.

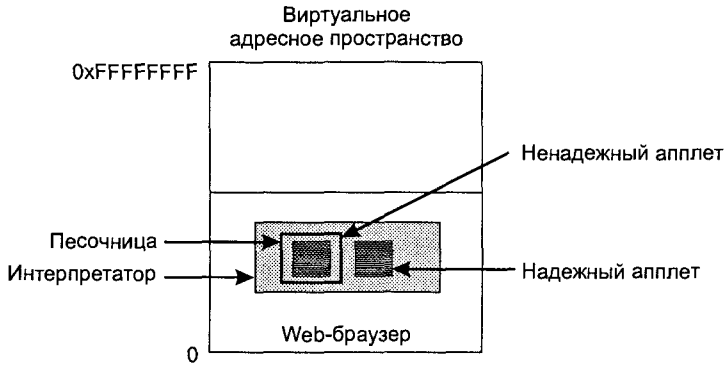


Рис. 8.48. Апплеты могут интерпретироваться веб-браузером

Преимущество интерпретируемого кода перед компилируемым состоит в том, что перед исполнением изучается каждая инструкция. Это дает интерпретатору возможность проверить состоятельность адреса инструкции. Кроме того, системные вызовы также перехватываются и интерпретируются. Как именно они обрабатываются, зависит от политики защиты информации. Например, если апплет надежный (например, он был создан на локальном диске), его системные вызовы могут обрабатываться без дополнительных проверок. Если же апплет не может считаться надежным (например, он был загружен из Интернета), его можно поместить в так называемую **песочницу**, регулирующую его поведение и пресекающую его попытки использовать системные ресурсы.

Если апплет пытается захватить системный ресурс, вызов передается монитору безопасности, который может разрешить или запретить данное действие. Монитор исследует вызов с точки зрения локальной политики защиты информации и затем принимает нужное решение. Таким образом, можно предоставить апплетам доступ к некоторым (но не ко всем) ресурсам. К сожалению, в реальной жизни такая модель работает плохо, в ней постоянно возникают ошибки.

ActiveX

Управляющие элементы ActiveX — это двоичные программы, рассчитанные на процессор Pentium, которые можно внедрять в веб-страницы. Когда на странице

встречается такая программа, производится проверка необходимости ее выполнения, и в случае положительного ответа она запускается. Эти программы не интерпретируются и не помещаются в песочницы, поэтому они обладают такими же возможностями, как обычные пользовательские программы, и, в принципе, могут нанести большой вред. Таким образом, вся защита информации в данном случае сводится к вопросу о том, стоит ли запускать управляющий элемент.

Для принятия таких решений корпорацией Microsoft был выбран метод, базирующийся на **подписях кода**. Суть в том, что каждый элемент ActiveX снабжается цифровой подписью, а именно хэшем кода, подписанным его создателем с использованием открытого ключа. Когда браузер встречает управляющий элемент, он вначале проверяет правильность подписи, убеждаясь в том, что код не был заменен по дороге. Если подпись корректна, браузер проверяет по своим внутренним таблицам, можно ли доверять создателю программы. Возможно, про самого создателя ничего не известно, но существует цепочка заверений, ведущая к какому-либо известному своей надежностью разработчику. Если создатель надежный, программа выполняется, в противном случае игнорируется. Система, созданная Microsoft для проверки управляющих элементов ActiveX, называется **Authenticode**.

Полезно противопоставлять друг другу подходы Java и ActiveX. В первом случае не производятся никакие попытки установить авторство апплета.

Вместо этого используется интерпретатор, который запрещает апплету совершать определенные нежелательные действия. Что касается метода подписания кода, то в этом случае, напротив, поведение программы во время ее выполнения никак не отслеживается. Если она была получена из проверенного источника и по дороге не была изменена, она просто запускается. Проверка самого кода не осуществляется. Если программист намеренно написал код, форматирующий жесткий диск и стирающий флэш-память компьютера, и при этом он считается проверенным и надежным программистом, то код выполнится и выведет из строя компьютер (если только в браузере не отключены управляющие элементы ActiveX).

Многие считают, что доверять неизвестным производителям программного обеспечения несколько легкомысленно. Чтобы доказать это, один программист из Сиэтла основал свою компанию и добился получения сертификата надежности, что не так уж сложно. После этого он написал управляющий элемент ActiveX, который всего-навсего выключал компьютер. Он распространил свою программу весьма широко, и она выключила не одну тысячу компьютеров. Впрочем, машины после этого можно было запросто включить заново, поэтому никакого ущерба такой управляющий элемент нанести не мог. Цель проекта состояла в том, чтобы указать миру на наличие проблемы. Официальная реакция выразилась в отзыве сертификата для данного конкретного управляющего элемента, и на этом инцидент был исчерпан. Но проблема-то решена не была, и нечистые на руку программисты могли продолжать использовать эту дыру в защите (Garfinkel и Spafford, 2002). Поскольку нет никакой возможности проследить за деятельностью всех компаний, пишущих переносимые программы, вскоре метод подписания кода может представлять собой довольно серьезную угрозу.

JavaScript

В JavaScript вообще отсутствует какая-либо официальная модель системы защиты информации, зато существует длинная история неудачных попыток ее внедрения. Каждый производитель пытается придумать что-нибудь свое. Например, в Netscape Navigator версии 2 было реализовано нечто подобное Java-модели, а уже в четвертой версии прослеживаются черты модели подписей кода.

Суть проблемы в том, что чужеродной программе разрешается выполнять какие-то действия. Это может привести к непредсказуемым последствиям. С точки зрения безопасности это то же самое, что позвать в гости вора и пытаться внимательно следить за тем, чтобы он не проник из кухни в гостиную. Если произойдет что-нибудь неожиданное, а вы в этот момент отвлечетесь, может случиться что угодно. Аргументом в защиту переносимых программ служит то, что с их помощью легко реализуются флэш-графика и быстрое взаимодействие с пользователем. Создатели веб-сайтов обычно считают, что это гораздо важнее, чем защита информации, особенно когда дело касается какого-нибудь чужого компьютера.

Вирусы

Вирусы — это своеобразная форма переносимого кода. Только в отличие от приведенных выше примеров, запускать такие программы никто не хочет. Основное отличие вирусов от обычных переносимых программ заключается в том, что они воспроизводят сами себя. Когда в систему проникает вирус (с веб-страницы, во вложении электронного письма или как-то еще), для начала он заражает исполняемые программы, хранящиеся на диске. При запуске какой-либо из этих программ управление передается вирусу, а тот обычно пытается распространить свое действие еще и на другие машины, например, рассылая самого себя по электронной почте всем адресатам из адресной книги жертвы. Некоторые вирусы заражают загрузочный сектор жесткого диска, поэтому вирус активируется при загрузке машин.

Вирусы в какой-то момент стали представлять собой крупномасштабную проблему для Интернета и принесли многомиллиардные убытки. Какого-либо простого решения проблемы не существует. Возможно, положение может спасти создание нового поколения операционных систем, базирующихся на защищенных микроядрах, и сплоченность пользователей, процессов и ресурсов.

Социальный аспект

Интернет и технологии защиты информации — это те области, в которых очень тесно сплелись социальные вопросы, государственная политика и технологии. Далее мы кратко рассмотрим три проблемы: конфиденциальность, свободу слова и авторские права. Совершенно очевидно, что в рамках этой книги мы сможем дать лишь поверхностное описание этой темы. Более подробную информацию следует искать в (Anderson, 2001; Garfinkel и Spafford, 2002; Schneier, 2000). Многие материалы можно прочитать в Интернете. Достаточно лишь набрать в поис-

ковой машине «конфиденциальность» (privacy — для получения информации на английском языке), «цензура» (censorship) или «авторские права» (copyright).

Конфиденциальность

Имеют ли люди право на секреты? Хороший вопрос. Четвертая поправка к конституции США запрещает правительственным организациям без особой нужды интересоваться намерениями граждан, их жильем и личными бумагами. Ограничен перечень обстоятельств, при которых этот запрет может быть нарушен. Таким образом, вопрос конфиденциальности стоит на повестке дня уже более 200 лет, по крайней мере, в США.

Что изменилось за последнее десятилетие? Правительство получило возможность с невиданной легкостью шпионить за гражданами, а граждане — с не меньшей легкостью предотвращать шпионаж. В XVIII веке для получения доступа к личным бумагам гражданина требовалось выслать к нему в имение полицейского, которому нужно было в любую непогоду доскакать на коне, претерпевая всевозможные лишения, которые нередки в долгом пути, — и все это для того, чтобы прочесть один чужой листок бумаги. В наши дни телефонные компании и поставщики услуг Интернета обеспечивают всех, кто может предъявить соответствующий ордер, подслушивающими устройствами. С их помощью задача полицейских сильно облегчается, к тому же нет риска выпасть из седла, заснув в пути.

Тем не менее, использование криптографии в значительной степени меняет дело. Любой желающий может озаботиться загрузкой и установкой PGP, генерированием хорошо защищенного, надежного ключа, и в результате он получит уверенность в том, что никто во Вселенной не сможет прочесть его электронную почту, независимо от наличия у него ордера на обыск. Правительства прекрасно это понимают, и им, разумеется, это сильно не нравится. В реальности конфиденциальность означает, что уполномоченным органам очень трудно следить за преступниками всех мастей, а также за журналистами и политическими оппонентами. Неудивительно, что многие правительства запрещают использование и экспорт криптографии. Во Франции, к примеру, до 1999 года любая негосударственная криптография была просто запрещена, если только государству не предоставлялись все используемые ключи.

Франция в этом деле не была одинока. В апреле 1993 года правительство США объявило о своем желании создать аппаратный криптопроцессор (clipper chip) и сделать его стандартным для применения в любых сетевых коммуникациях. Таким образом, как было заявлено, граждане получат гарантированную конфиденциальность. Вместе с тем, упоминалось о том, что правительство будет иметь возможность расшифровывать весь трафик таких криптопроцессоров при помощи специальной технологии, позволяющей правительству получать доступ ко всем ключам. Однако были даны обещания использовать эту возможность только при наличии соответствующей санкции. Понятно, что такое заявление вызвало большой фурор: сторонники конфиденциальности осуждали весь план от начала до конца, а чиновники, выступающие в поддержку этого начинания, были

восхищены предложением правительства. Тем не менее, правительство почему-то сдало позиции и отказалось от собственной идеи.

Огромное количество материалов, посвященных конфиденциальности цифровой информации, доступно на веб-сайте фонда Electronic Frontier (www.eff.org).

Анонимные рассылки

PGP, SSL и другие технологии позволяют устанавливать между двумя сторонами защищенные, аутентифицированные соединения, не подверженные вмешательству третьих сторон. Однако иногда конфиденциальность лучше всего обеспечивается как раз *отсутствием* аутентификации, то есть, по сути дела, установлением анонимных соединений. Анонимность востребована как при передаче сообщений между двумя пользователями, так и в сетевых телеконференциях.

Рассмотрим некоторые примеры. Во-первых, политические диссиденты, живущие при авторитарном режиме, могут захотеть во избежание репрессий общаться анонимно. Во-вторых, различные нарушения во многих коммерческих, образовательных, правительственных и других организациях зачастую выявляются не без помощи доносчиков, которые желают оставаться неизвестными. В-третьих, приверженцы нетрадиционных (а значит, как правило, порицаемых) социальных политических или религиозных убеждений видят одну из немногих возможностей общения в телеконференциях (или электронной почте), где они могут скрывать свои истинные имена. В-четвертых, многие предпочитают обсуждать алкоголизм, душевные заболевания, сексуальные проблемы, проблемы жестокого обращения с детьми или отношение к преследуемым меньшинствам в телеконференциях, где они могут оставаться анонимными. Кроме того, конечно, существует масса иных примеров.

Рассмотрим один конкретный пример. В 1990-х годах некоторые критики одной нетрадиционной религиозной секты опубликовали свои взгляды в конференции USENET с помощью **анонимной рассылки**. Сервер позволял пользователям создавать псевдонимы и посылать на него электронные письма, которые затем рассылались от имени выбранного псевдонима. В итоге не было возможности понять, кто является настоящим автором письма. Некоторые из этих статей были разоблачениями, в состав которых, по мнению представителей секты, входили коммерческие тайны и документы, защищенные авторским правом. В ответ на эти разоблачения секта подала в суд, жалуясь на раскрытие коммерческих тайн и нарушение закона об авторском праве. И то, и другое в том округе, где находился сервер, считалось преступлением. Последовал суд, и владельцев сервера вынудили раскрыть истинные имена тех, кто скрывался под псевдонимами и писал разоблачения (кстати, это был не первый прецедент, связанный с недовольством церкви раскрытием ее тайн: Уильям Тиндэйл (William Tyndale) был в 1536 году сожжен на столбе за перевод Библии на английский).

Значительная часть Интернет-сообщества была сильно возмущена таким грубым нарушением принципов конфиденциальности. Все были согласны и с тем, что владелец анонимной рассылки, хранившей таблицу соответствия настоящих электронных адресов и псевдонимов (это было названо анонимной рассылкой первого типа), был не прав. Этот случай стимулировал развитие анонимных рассылок, которые могли бы противостоять таким атакам со стороны суда.

Рассылки нового типа, часто называемые **шифрованными панковскими рассылками** (cypherpunk remailer), работают следующим образом. Пользователь создает электронное письмо с обычными заголовками RFC 822 (разумеется, отсутствует *From*), шифрует его открытым ключом рассылки и отправляет на сервер. Там от него отрезаются заголовки RFC 822, содержимое расшифровывается, и сообщение рассылается подписчикам. В рассылке нет никаких учетных записей, не ведутся никакие журналы, поэтому даже в случае конфискации сервера никаких следов прошедших через него писем обнаружено не будет.

Многие пользователи, которые особенно сильно озабочены проблемой собственной анонимности, прогоняют свои сообщения через цепочки анонимных рассылок, как показано на рис. 8.49. В данном примере Алиса хочет послать Бобу действительно очень-очень анонимное поздравление с днем св. Валентина. Для этого она использует три анонимные рассылки. Она сочиняет письмо M и вставляет заголовок, содержащий адрес электронной почты Боба. Затем все это сообщение шифруется открытым ключом рассылки 3, E_3 (показано горизонтальной штриховкой). К этому прибавляется заголовок с электронным адресом рассылки 3 (передается открытым текстом). В итоге получается сообщение, показанное между рассылками 2 и 3 на рисунке.

На этом история сообщения не заканчивается. Оно шифруется открытым ключом рассылки 2, E_2 (показано вертикальной штриховкой), и дополняется открытым заголовком, содержащим электронный адрес рассылки 2. Получившееся в итоге сообщение показано на рисунке между рассылками 1 и 2. Затем Алиса шифрует свое сообщение открытым ключом рассылки 1, E_1, добавляет адрес этой рассылки и, наконец, отправляет его. Конечное состояние сообщения показано на рисунке справа от Алисы.

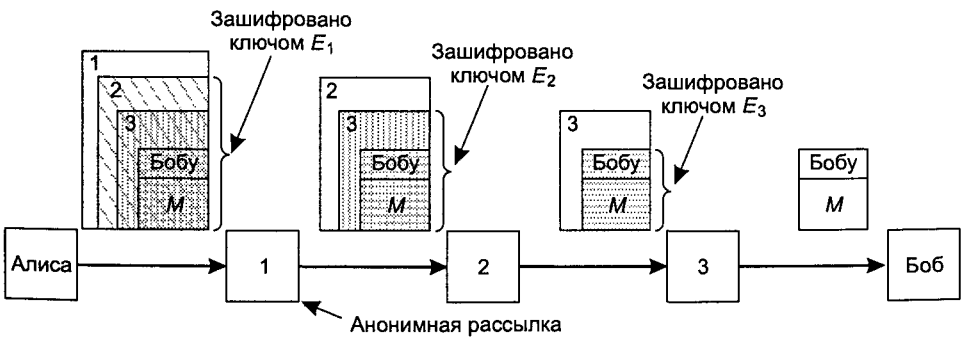


Рис. 8.49. Использование трех анонимных рассылок для передачи письма Алисы Бобу

Когда письмо Алисы достигает рассылки 1, от него отрезается внешний заголовок. Тело сообщения расшифровывается и пересылается в рассылку 2. Аналогичные шаги производятся и на двух других серверах.

Несмотря на то, что восстановить путь конечного сообщения до Алисы и так очень сложно, многие рассылки принимают еще и дополнительные меры предосторожности. Например, они могут задерживать сообщения на какой-то случай-

ный промежуток времени, добавлять или удалять всякий мусор в конце сообщения, переставлять сообщения местами, в общем, делать все возможное для того, чтобы запутать тех, кто отслеживает трафик и пытается понять, кто является автором того или иного сообщения, прошедшего через анонимную рассылку. Описание системы, реализующей в настоящее время описанные ранее идеи, можно найти в (Mazi res и Kaashoek, 1998).

Сфера применения принципов анонимности не ограничивается одной электронной почтой. Существуют также услуги, позволяющие анонимно просматривать интернет-материалы. Пользователь может настроить свой бразуер на использование такой услуги в качестве прокси. После этого все HTTP-запросы направляются по адресу, принадлежащему этому сервису, который производит запрос страницы за пользователя. Если на сервере, обеспечивающем анонимность, не хранятся журналы активности, никто не сможет определить, кто на самом деле запрашивал страницу.

Свобода слова

Конфиденциальность связана с проблемой сокрытия от посторонних глаз информации, не подлежащей обнародованию. Вторым ключевым социальным аспектом является, несомненно, свобода слова и ее противоположность — цензура. В этом случае правящие органы пытаются ограничить спектр информации, которую граждане могут читать и публиковать. Всемирная паутина с ее миллионами страниц — это настоящий рай для цензуры. В зависимости от типа и идеологии режима, в список запрещенных к просмотру материалов могут помещаться страницы, содержащие что-либо из перечисленного ниже:

1. Материалы, которые нельзя показывать детям и подросткам.
2. Материалы, пропагандирующие ненависть к каким-либо этническим, религиозным, сексуальным или другим группам.
3. Информацию о демократии и демократических ценностях.
4. Описания исторических событий, не совпадающие с официальной версией.
5. Руководства по взлому разного рода замков, созданию оружия, шифрованию сообщений и т. д.

Плохой, негодный сайт проще всего запретить к просмотру.

Иногда результаты такой политики оказываются неожиданными. Например, некоторые публичные библиотеки установили у себя веб-фильтры, не пропускающие порнографические сайты и таким образом делающие содержимое Паутины безопасным для просмотра детьми. Фильтры перед выводом страниц сверяют адреса со своими «черными списками», к тому же проверяют их на наличие бранных слов. Однажды в округе Лаудаун, штат Вирджиния, фильтр заблокировал поиск информации по раку молочной железы, так как запрос содержал слово «breast» (женская грудь, молочная железа). Клиент возбудил дело против правительства округа. В то же время, был и другой случай: в Ливерморе, штат Калифорния, один родитель подал в суд на публичную библиотеку за то, что там

не был установлен фильтр и он застал своего 12-летнего сына за просмотром порнографического сайта. Так что же библиотеке делать?

Многие никак не могут понять, что Всемирная паутина — действительно всемирная. Она охватывает весь земной шар. В разных странах существуют разные взгляды на то, что должно, а чего не должно быть в Сети. Например, в ноябре 2000 года французский суд постановил, что Yahoo, корпорация, находящаяся в Калифорнии, должна запретить доступ к аукциону памятных вещей нацистов для французских пользователей, так как обладание такой информацией идет вразрез с французским законодательством. Yahoo апеллировала к суду США, который решил спор в пользу корпорации, однако в целом проблема того, законы какой страны должны применяться в Интернете, остается актуальной.

Попробуйте представить себе, что случится, если какой-нибудь суд штата Юта вынесет решение о том, что Франция должна заблокировать сайты, посвященные винам, так как распространение подобной информации нарушает строгие законы штата, касающиеся алкогольной продукции? Точно так же Китай может сделать табуированными все сайты, на которых рассказывается про демократию, так как это не в интересах Поднебесной. Должны ли иранские законы о религии применяться в либеральной Швеции? Может ли Саудовская Аравия заблокировать сайты, защищающие права женщин? Становится понятно, что эта проблема подобна ящику Пандоры и способна породить множество вопросов.

Ценное замечание высказывает Джон Гилмор (John Gilmore): «Сеть воспринимает цензуру как разрушенный участок дороги и идет в обход». Конкретная реализация этой мысли называется **службой вечности** (Anderson, 1996). Ее цель — гарантировать, что однажды опубликованные материалы не исчезнут и не будут переписаны заново, как было принято в Советском Союзе во времена Сталина. Пользователь службы вечности должен лишь указать, в течение какого срока следует обеспечивать сохранность информации, заплатить пропорциональную сроку и объемам информации сумму и загрузить данные на сервер. После этого никто, включая самого пользователя, не сможет удалить или отредактировать размещенные на сервере службы вечности материалы.

Как такую услугу реализовать на практике? Проще всего организовать одноранговую систему, в которой документы будут размещаться на десятках серверов-участников проекта, каждый из которых будет получать свою долю вознаграждения, что послужит стимулом для их вступления в проект. Серверы должны располагаться в самых разных местах и под разной юрисдикцией, что обеспечит максимальную устойчивость системы. Списки 10 выбранных случайным образом серверов следует хранить в тайне в разных местах, чтобы в случае неудачи, произошедшей с одним из них, могли выжить другие. Любые государственные органы, помешанные на уничтожении неугодной информации, никогда не смогут быть до конца уверенными в том, что они нашли все копии. Кроме того, систему можно сделать самовосстанавливающейся, в том смысле, что в случае прихода известия об уничтожении каких-то экземпляров документов держатели остальных копий попытаются найти новые места хранения для замены выбывших из строя.

Служба вечности была первой попыткой противостояния цензуре в Сети. С тех пор было высказано много различных идей на эту тему, и некоторые из них даже нашли свое воплощение. Были добавлены новые возможности, такие как шифрование, анонимность, отказоустойчивость. Зачастую документы разбивают на несколько фрагментов и хранят их на нескольких серверах. Среди таких систем можно упомянуть Freenet (Clarke и др., 2002), PASIS (Wylie и др., 2000) и Publius (Waldman и др., 2000). Еще одна разработка описана в (Сержантов, 2002).

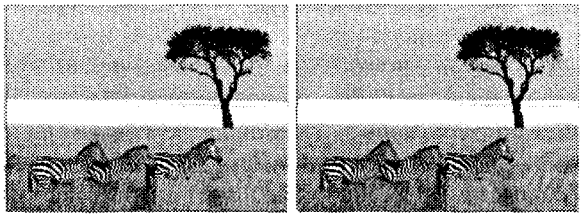
Все больше и больше стран пытаются контролировать экспорт таких неосязаемых вещей, как веб-сайты, программное обеспечение, научные документы, электронная почта, телефонные службы помощи и т. п. Даже в Великобритании, славящейся своими вековыми традициями поддержки свободы слова, появляются строгие законы, которые, к примеру, определяют техническую дискуссию между британским профессором и иностранным студентом в Кембриджском университете как предмет экспорта, подлежащий государственному лицензированию (Anderson, 2002). Очевидно, что такая политика крайне противоречива.

Стеганография

В странах, где цензура применяется особенно широко, всегда существуют диссиденты, использующие свои методы обхода этой цензуры. Криптография, конечно, позволяет (не всегда вполне законно) посылать секретные сообщения так, чтобы никто не смог узнать их смысл, однако если государство считает Алису своим врагом, один тот факт, что она общается с Бобом, может и его поставить в положение врага государства. Таким образом политики, обычно не слишком хорошо владеющие математикой, понимают и применяют принцип транзитивности. Выручить могут анонимные рассылки, но и их местное правительство может запретить, и тогда для отправки сообщения за границу понадобится экспортная лицензия. Таким образом, анонимные рассылки — это тоже не панацея. Однако Всемирная паутина всегда найдет выход из положения.

Люди, которым требуется секретное общение, зачастую пытаются скрыть сам факт общения. Наука, занимающаяся сокрытием сообщений, называется **стеганографией** (не путать со стенографией!), от греческого слова, которое можно перевести как «защищенное письмо». Сами древние греки первыми и начали использовать этот метод. Геродот описывал своеобразный метод секретного общения военачальников: посыльному брили волосы на голове, на затылке рисовали татуировку с секретным сообщением и ждали, пока волосы снова отрастут, после чего отправляли посыльного в путь. Современные технологии базируются на той же концепции, разве что пропускная способность стала выше, а задержки — ниже.

В качестве примера рассмотрим рис. 8.50, *а*. На этой фотографии, сделанной автором в Кении, изображены три зебры и акация. Однако она привлекательна не только с эстетической точки зрения. Дело в том, что рис. 8.50, *б* включает в себя внедренный полный текст пяти самых известных пьес Шекспира: *«Гамлет»*, *«Король Лир»*, *«Макбет»*, *«Венецианский купец»* и *«Юлий Цезарь»* — все тексты вместе занимают более 700 Кбайт.



а *б*

Рис. 8.50. Три зебры и дерево (*а*); три зебры, дерево и полный текст пяти пьес Вильяма Шекспира (*б*)

Как же это сделано? Стеганографический канал работает следующим образом. Размер исходного изображения равен 1024×768 пикселов. Каждый пиксел представлен тремя 8-битными числами, каждое из которых отвечает за свою составляющую цвета (существуют красный, зеленый и синий каналы интенсивности цвета). Результирующий цвет получается линейной суперпозицией интенсивностей трех цветов. При методе стеганографического шифрования младший бит каждого из трех указанных чисел заменяется нужным значением секретного канала. Таким образом, в каждом пикселе помещаются три бита секретной информации: один в канале интенсивности красного цвета, один — зеленого и один — синего. В нашем изображении поместится $1024 \cdot 768 \cdot 3$ бит, или 294 912 байт секретной информации.

Полный текст пяти пьес Шекспира с небольшим предисловием занимает 734 891 байт. Текст можно сжать до 274 Кбайт с помощью любого архиватора. Затем зашифровать эти сжатые данные с использованием алгоритма IDEA, после чего разместить данные в младших битах значений интенсивностей цветов. Как видите (вернее, это как раз остается невидимым), присутствие текстовой информации совершенно незаметно. На большой, полноцветной фотографии это также незаметно. Глаз человека просто не в состоянии отличить оттенки, выраженные 21-разрядным числом, от 24-разрядных оттенков.

Черно-белые фотографии, приведенные на рис. 8.50, возможно, дают неполное представление о возможностях стеганографии. Более эффектный вариант рис. 8.50, *б* продемонстрирован в полноцветной версии, которую можно найти на веб-сайте книги.

Для нужд незаметного обмена информацией диссиденты могут создать вебсайт, заполненный политкорректными фотографиями (например, Великого Вождя, местных спортивных команд, теле- и кинозвезд и т. п.). Разумеется, эти фотографии могут содержать стеганографические сообщения. Учитывая то, что эти сообщения перед помещением в графический файл сжимаются и шифруются, даже если кто-то и заподозрит их присутствие «внутри» фотографий, отличить их от белого шума, а тем более расшифровать их смысл будет очень и очень тяжело. Конечно, фотографии для стеганографической обработки должны быть отсканированы или сняты на цифровую камеру самостоятельно: если просто взять картинку из Интернета и записать в нее секретное сообщение, выявить «второе дно» простым побитовым сравнением будет гораздо проще.

Понятно, что стеганографические сообщения можно записывать не только в графические файлы. Очень хорошо для этих целей подходят звуковые файлы. А уж в видеофайлы можно записать действительно внушительные объемы данных. Надо сказать, что даже форматирование текста и расположение тегов в HTML-файле может нести определенную скрытую информацию.

Мы рассмотрели стеганографию в контексте проблемы свободы слова, однако у этой технологии есть масса иных применений. Очень часто авторы электронных изображений вшивают в файлы секретные сообщения, которые в случае необходимости смогут подтвердить их авторские права. Если кто-нибудь захочет украсть удачную фотографию и разместит ее на своем сайте без указания авторства, законный владелец с помощью стеганографической подписи сможет доказать в суде свою правоту. Такие подписи иногда называют **водяными знаками**. Эта тема обсуждается в (Piva и др., 2002).

Подробное рассмотрение проблем стеганографии можно найти в (Artz, 2001; Johnson и Jajoda, 1998; Katzenbeisser и Petitcolas, 2000; Wayner, 2002).

Защита авторских прав

Конфиденциальность и цензура — это те области, в которых сталкиваются технологические аспекты и общественные интересы. Третьей такой областью является защита авторских прав. **Авторское право** гарантирует свободу распоряжения **интеллектуальной собственностью** ее *создателям*, которыми могут быть, например, писатели, художники, композиторы, музыканты, фотографы, кинорежиссеры, хореографы и т. д. Авторское право выдается на определенный срок, который обычно равен сроку жизни автора плюс 50 лет (75 лет — в случае корпоративного авторского права). По окончании этого срока интеллектуальная собственность становится достоянием общества, и каждый волен распоряжаться ею, как хочет. Так, например, проект «Gutenberg» (www.promo.net/pg) разместил в Сети тысячи произведений, давно уже ставших общественным достоянием (работы Шекспира, Диккенса, Марка Твена). По просьбе Голливуда в 1998 году Конгресс США разрешил продлить срок действия авторских прав еще на 20 лет, утверждая, что без принятия этой меры больше никто ничего не станет создавать. В то же время, патенты на изобретения действуют в течение всего лишь 20 лет, и никто не жалуется — люди продолжают совершать открытия и делать изобретения.

Вопрос о защите авторских прав вышел на первый план, когда у службы Napster, незаконным образом распространявшей музыку, внезапно оказалось 50 миллионов пользователей. Несмотря на то, что записи системой Napster никуда не копировались, судом было предъявлено обвинение в хранении центральной базы данных с информацией о том, какие у кого из пользователей имеются записи. Таким образом, система побуждала пользователей совершать преступные действия, нарушающие закон об авторских правах. В общем-то, никто не жалуется на то, что идея авторских прав плоха (хотя многим не нравится то, что компании обладают намного большими привилегиями в этом плане, чем простые граждане), однако следующее поколение технологий распространения музыкальных записей уже поднимает вопросы этического характера.

Например, рассмотрим равноранговую сеть, члены которой занимаются вполне законным обменом файлами (записями, являющимися общественным достоянием, домашними видеозаписями, религиозными трактатами (не составляющими коммерческую тайну церквей) и т. д.). Возможно, какая-то небольшая часть этих файлов защищена законом об авторских правах. Допустим, все участники проекта находятся на постоянном соединении (пользуются ADSL или кабельным Интернетом). На каждой машине хранится список того, что есть на жестком диске, а также список всех машин сети. В поисках какого-либо файла можно обратиться к произвольно выбранной машине и запросить у нее список имеющихся материалов. Если нужная информация не находится, можно попробовать опросить все остальные машины из списка, а также машины из списков, хранящихся у других. Компьютеры здесь сильно облегчают задачу поиска некоторых редких материалов. Найдя нужный файл, пользователь просто копирует его себе.

Если найденная работа оказывается защищенной законом об авторских правах, пользователь, сам того не желая, становится нарушителем этого закона (тут еще, конечно же, играет роль то, в какой стране происходит дело и, соответственно, какие законы следует применять в каждом конкретном случае). А виноват ли поставщик информации? Является ли преступлением хранение у себя на жестком диске записей, за которые были заплачены деньги и которые были вполне легально загружены из Интернета, при условии, что к диску могут иметь доступ посторонние лица? Если в вашу лачугу, никогда не знавшую замков и засовов, проникает вор с ноутбуком и сканером, снимает копию с книги, защищенной законом об авторских правах, и уходит восвояси, разве *вы* виноваты в этом преступлении, разве *вы* должны защищать чужие авторские права?

Однако есть еще одна проблема, связанная с авторскими правами. Ведется ожесточенная борьба между Голливудом и компьютерной индустрией. Голливуд требует усилить защиту интеллектуальной собственности, а компьютерщики говорят, что они не обязаны сторожить голливудские ценности. В октябре 1998 года Конгресс принял **Акт об авторских правах в цифровых технологиях** (DCMA — Digital Millenium Copyright Act), в котором говорится о том, что взлом механизма защиты, присутствующего в работе, защищенной законом об авторских правах, а также сообщение другим технологии взлома является преступлением. Аналогичный законодательный акт был принят и в Евросоюзе. С одной стороны, все как-то забыли подумать о том, что для пиратов с Дальнего Востока такие акты указом не являются, а с другой стороны, многие считают, что новый закон нарушил баланс между интересами владельцев авторских прав и общественными интересами.

Взять хотя бы такой пример. В сентябре 2000 года консорциум, связанный с музыкальной индустрией, озабоченный созданием надежной онлайновой системы продажи аудиозаписей, организовал конкурс, пригласив всех желающих попробовать взломать систему (это действительно очень важный этап, необходимый при создании любой новой системы защиты). Группа ученых из различных университетов под руководством профессора Эдварда Фельтена (Edward Felten) из Принстона, специализирующихся в области защиты информации, приняли вызов и сломали систему. Затем была написана статья, описывающая выводы, сде-

ланные в ходе исследования. Она была направлена на конференцию USENIX, посвященную проблемам защиты информации. Доклад был рассмотрен и принят на соответствующем уровне. Однако незадолго до конференции Фельтен получил уведомление от Ассоциации звукозаписи о том, что эта организация в случае опубликования статьи подаст на авторов в суд за нарушение акта DCMA.

Ученым ничего не оставалось делать, как послать запрос в федеральный судебный орган, пытаясь выяснить, является ли еще легальным опубликование научных статей, касающихся защиты информации. Опасаясь, что дальнейшее развитие событий будет отнюдь не в пользу звукозаписывающей индустрии, ее представители сняли свои претензии к Фельтену, и на этом инцидент был исчерпан. Несомненно, причиной недовольства звукозаписывающей индустрии была слабость предложенной ими системы. Получилось, что сначала пригласили людей, чтобы те попытались взломать защиту, а когда некоторым это реально удалось, на них чуть было не подали за это в суд. После того, как все конфликты были улажены, статья все-таки была напечатана (Craver и др., 2001). Почти очевидно, что таких конфронтаций впереди еще много.

С обсуждаемой темой тесно связана **доктрина законного использования**, ставшая результатом судебных решений во многих странах. Эта доктрина состоит в том, что покупатели продукции, защищенной законом об авторских правах, имеют сильно ограниченные права на копирование этой продукции, включая даже использование ее частей для научных целей (например, в качестве обучающего материала в школах и колледжах) и создание архивных резервных копий на тот случай, если что-нибудь случится с исходным носителем. Как проверить, является ли использование продукции законным? Показатели таковы: 1) коммерческое использование; 2) количество процентов скопированных данных; 3) влияние копирования на объем продаж. Так как DMCA и аналогичные законы, принятые Евросоюзом, запрещают взлом систем защиты от копирования, такие законы заодно запрещают легальное добросовестное использование. По сути, DMCA отбирает у потребителей историческое право активно поддерживать продавцов, у которых они приобрели продукцию. Провал этой идеи неизбежен.

Еще одно явление, затмевающее собой по уровню смещения баланса между обладателями авторских прав и потребителями даже DMCA, — это Альянс надежных вычислительных платформ (TCPA — Trusted Computing Platform Alliance). Разрабатывается этот проект совместными усилиями Intel и Microsoft. Идея состоит в том, чтобы процессор и операционная система зорко наблюдали за действиями пользователя (например, за воспроизведением скопированной незаконным образом звукозаписи) и запрещали совершать нежелательные действия. Такая система могла бы даже позволить обладателям авторских прав удаленно манипулировать персональными компьютерами пользователей, изменяя при необходимости определенные правила. Несомненно, общественный резонанс будет огромен. Конечно, это очень здорово, что индустрия наконец обратила внимание на проблемы защиты информации, однако нельзя не заметить с прискорбием, что большинство усилий направлено на усиление законов об авторских правах, а не на борьбу с вирусами, взломщиками, мошенниками и другими проблемами, волнующими большинство пользователей.

Короче говоря, авторам законов и юристам теперь предстоит в течение долгих лет пытаться урегулировать взаимоотношения владельцев авторских прав и потребителей. Киберпространство ничем не отличается от социума: и там, и там постоянно сталкиваются интересы различных групп, что приводит к ожесточенной борьбе и судебным разбирательствам, результатом которых рано или поздно становится нахождение какого-то компромисса. По крайней мере, стоит на это надеяться, как стоит надеяться на относительное затишье до появления новой противоречивой технологии.

Резюме

Криптография представляет собой инструмент, используемый для обеспечения конфиденциальности информации, проверки ее целостности и аутентичности. Все современные криптографические системы основаны на принципе Керкгофа, гласящем, что алгоритмы должны быть доступны всем желающим, а ключи — храниться в тайне. Многие алгоритмы при шифровании текста выполняют сложные преобразования, включающие в себя замены и перестановки. Тем не менее, если удастся реализовать на практике принципы квантовой криптографии, то с помощью одноразовых блокнотов можно будет создать действительно надежные криптосистемы.

Все криптографические алгоритмы можно разделить на два типа: с симметричными ключами и с открытыми ключами. Алгоритмы с симметричными ключами при шифрации искажают значения битов в последовательности итераций, параметризованных ключом. Наиболее популярные алгоритмы этого типа — тройной DES и Rijndael (AES). Алгоритмы с симметричным ключом могут работать в режиме электронного шифроблокнота, сцепления блоков шифра, потокового шифра и др.

Алгоритмы с открытыми ключами отличаются тем, что для шифрования и дешифрации используются разные ключи, причем ключ дешифрации невозможно вычислить по ключу шифрования. Эти свойства позволяют делать ключ открытым. Чаще всего применяется алгоритм RSA, основанный на сложности разложения больших чисел на простые сомножители.

Официальные, коммерческие и другие документы необходимо подписывать. Существуют различные методы генерации цифровых подписей, основанные на алгоритмах как с симметричными, так и с открытыми ключами. Обычно при помощи алгоритмов типа MD5 или SHA-1 вычисляется хэш-функция сообщения, которое необходимо подписать, и подпись ставится не на само сообщение, а на значение хэш-функции.

Управление открытыми ключами можно реализовать при помощи сертификатов, представляющих собой документы, связывающие между собой открытые ключи и обладающих ими принципалов. Сертификаты подписываются надежной организацией или кем-либо, способным предъявить цепочку сертификатов, ведущих в итоге к этой надежной организации. Начальное звено этой цепочки (центральное управление) должно быть известно заранее, для этого в браузеры вшиваются номера многих сертификатов центральных управлений.

Эти криптографические методы позволяют защитить сетевой трафик. На сетевом уровне для этого работает система IPsec, занимающаяся шифрацией пакетов, передающихся между хостами. Брандмауэры могут фильтровать как входящий, так и исходящий трафик, анализируя используемые протоколы и порты. Виртуальные частные сети могут стимулировать повышение степени защиты информации в старых сетях, построенных на основе выделенных линий. Наконец, в беспроводных сетях требуется довольно серьезная защита информации, и этого не может обеспечить WEP 802.11. Остается надеяться на улучшение ситуации с появлением 802.11i.

При установлении сеанса связи между двумя сторонами они должны идентифицировать себя и, возможно, определить общий ключ сеанса. Для этого существуют различные протоколы аутентификации, среди которых есть те, которые подразумевают наличие третьей, надежной стороны, а также протоколы Диффи—Хеллмана, «Керберос» и протоколы с использованием открытых ключей.

Защита информации в электронной почте может быть достигнута применением различных комбинаций тех методов, которые мы изучили в этой главе. PGP, например, сжимает сообщение, а потом шифрует его с помощью IDEA. Ключ IDEA шифруется при помощи открытого ключа получателя. Кроме того, вычисляется хэш-функция сообщения. Проверка целостности может быть выполнена за счет того, что на хэш перед отправкой ставится подпись.

Безопасность во Всемирной паутине — это тоже важная проблема, которая начинается с безопасного именования ресурсов. DNSsec позволяет предотвращать обманы DNS-серверов, а также создавать самозаверяющиеся имена. Большинство сайтов, посвященных электронной коммерции, для установления надежных аутентифицированных сеансов между клиентом и сервером используют SSL. Для борьбы с возможными проблемами, связанными с переносимыми программами, разработаны разные методы, среди которых помещение загружаемого кода в изолированную среду («песочницу») и подписание кодов.

В Интернете возникает множество вопросов, в которых технологические проблемы переплетаются с государственной политикой. Мы рассмотрели лишь некоторые из них: конфиденциальность, свободу слова и авторские права.

Вопросы

1. Расшифруйте следующее сообщение, составленное с помощью моноалфавитного шифра. Открытый текст, состоящий только из букв, представляет собой хорошо известный отрывок из поэмы Льюиса Кэррола.

```
kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm
ur mfudm zhx mftnm zhx mdzythc pzq ur ezsszcdm zhx gthcm
zhx pfa kfd mdz tm sutythc fuk zhx pfdkfdi ntcm fzld pthcm
sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk
rui mubd ur om zid uok ur sidzkf zhx zyy ur om zid rzk
hu foiia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk
```

2. Взломайте следующий колоночный перестановочный шифр. Открытый текст взят из популярной книги о компьютерах, поэтому слово «computer» может

встретиться с большой вероятностью. Открытый текст состоит из одних букв (без пробелов). Зашифрованный текст разбит на блоки по пять символов для удобства чтения.

```
aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo gorch crsoc nntii imiha oofpa
gsivt tpsit lbolr otoex
```

3. Подберите 77-разрядный одноразовый блокнот, с помощью которого из шифра, показанного на рис. 8.3, можно получить текст «Donald Duck».
4. При использовании квантовой криптографии требуется наличие фотонной пушки, способной при необходимости испускать одиночный фотон, соответствующий одному биту. Подсчитайте, сколько фотонов переносит 1 бит по 100-гигабитной оптоволоконной линии связи. Длина фотона предполагается равной длине волны, то есть 1 мкм (для нужд данной задачи). Скорость света в оптоволокне равна 20 см/нс.
5. Если злоумышленнику удастся перехватить и регенерировать фотоны при использовании квантовой криптографии, некоторые значения будут приняты неверно, а значит, появятся ошибки и в одноразовом блокноте, принимаемом Бобом. Какая (в среднем) часть блокнота, принятого Бобом, будет содержать ошибки?
6. Фундаментальный принцип криптографии гласит, что все сообщения должны быть избыточными. Но мы знаем, что избыточность позволяет злоумышленнику понять, правильно ли он угадал секретный ключ. Рассмотрите два типа избыточности. В первом типе изначальные n бит открытого текста содержат известную последовательность. Во втором результирующие n бит сообщения содержат хэш. Эквивалентны ли эти типы избыточности с точки зрения безопасности? Ответ поясните.
7. На рис. 8.5 S-блоки и P-блоки сменяют друг друга. Сделано ли это из эстетических соображений, или такая организация шифратора дает более надежный код, чем в случае, когда сначала идут все P-блоки, а затем все S-блоки?
8. Разработайте атаку шифра DES, основываясь на информации, что открытый текст состоит исключительно из прописных ASCII-символов, пробелов, запятых, двоеточий, символов «точка с запятой», «перевод строки» и «возврат каретки». О битах четности, содержащихся в открытом тексте, ничего не известно.
9. В тексте было подсчитано, что машине для взлома шифра, снабженной миллиардом процессоров, способных обрабатывать 1 ключ за 1 пс, понадобится всего лишь 10^{10} лет для взлома 128-битной версии AES. Тем не менее, на сегодняшний день машины могут иметь не более 1024 процессоров и обрабатывать ключ за 1 мс, так что даже для достижения приведенных ранее результатов необходимо увеличить производительность в 10^{15} раз. Если закон Мура, утверждающий, что вычислительные мощности компьютеров удваиваются каждые 18 месяцев, будет продолжать соблюдаться всегда, сколько лет придется ждать указанного прироста производительности?
10. Ключи AES могут иметь длину 256 бит. Сколько вариантов ключей доступно в таком режиме? Сможете ли вы найти в какой-нибудь науке — например,

физике, химии или астрономии — понятия, оперирующие подобными величинами? Найдите в Интернете материалы, посвященные большим числам. Сделайте выводы из этого исследования.

11. Предположим, что сообщение было зашифровано с помощью шифра DES в режиме сцепления блоков. В одном из блоков зашифрованного текста при передаче один бит изменился с 0 на 1. Какая часть текста будет при этом повреждена?
12. Снова рассмотрим шифрование со сцеплением блоков. На этот раз между блоками зашифрованного текста при передаче вставился один нулевой бит. Какая часть текста будет повреждена в этом случае?
13. Сравните режим шифрования со сцеплением блоков с режимом шифрованной обратной связи с точки зрения количества операций шифрования, необходимых для передачи большого файла. Какой режим более эффективен и насколько?
14. Используя алгоритм открытого ключа RSA, при $a = 1$, $b = 2$ и т. д.;

    1) перечислите пять допустимых значений d для $p = 7$ и $q = 11$;

    2) найдите e, если $p = 13$, $q = 31$ и $d = 7$;

    3) для $p = 5$, $q = 11$ и $d = 27$ найдите e и зашифруйте «abcdefghij».
15. Предположим, что Мария внезапно обнаруживает, что ее закрытый ключ RSA (d 1, n 1) совпадает с открытым ключом RSA (e 2, n 2) другого пользователя, Франчески. Другими словами, d 1 = e 2 и n 1 = n 2. Следует ли Марии задуматься о смене открытого и закрытого ключей? Ответ поясните.
16. Рассмотрите режим счетчика, показанный на рис. 8.13, но со значением вектора инициализации, равным 0. Угрожает ли использование нуля надежности шифра в целом?
17. Протокол генерации подписей, показанный на рис. 8.15, обладает одним недостатком. Если с компьютером Боба что-нибудь случится, содержимое оперативной памяти будет утеряно. Какие проблемы это может вызвать, и как с ними бороться?
18. На рис. 8.17 мы видим, как Алиса передает Бобу подписанное сообщение. Если Труди заменит P, Боб заметит это. А что будет, если Труди заменит и P, и подпись?
19. Цифровые подписи имеют один недостаток: их могут использовать лентяи. После составления договора в электронной коммерции обычно требуется, чтобы клиент подписал его своим хэшем SHA-1. Если пользователь не озаботится проверкой соответствия хэша и контракта, он имеет шанс случайно подписать другой договор. Допустим, вездесущая и бессмертная мафия решила сделать на этом деньги. Бандиты создают платный веб-сайт (содержащий, например, порнографию, азартные игры и т. п.) и спрашивают у новых клиентов номера кредитных карт. Затем пользователю отсылается договор, в котором говорится, что он желает воспользоваться услугами и оплатить их с помощью кредитной карты. Договор следует подписать, однако владельцы

сайта знают, что вряд ли кто-то станет сверять хэш с контрактом. Покажите, каким образом злоумышленники смогут купить бриллианты в легальном ювелирном интернет-магазине за счет ничего не подозревающих клиентов.

20. В математическом классе 20 учащихся. Какова вероятность того, что, по крайней мере, у двух из них совпадают дни рождения? Допустим, что ни у кого из них день рождения не приходится на 29 февраля, поэтому общее число рассматриваемых вариантов дней рождения равно 365.
21. После того как Элен созналась Мэрилин в своем обмане в деле с предоставлением должности Тому, Мэрилин решила устранить проблему, записывая содержимое своих сообщений на диктофон, с тем, чтобы ее новая секретарша просто набирала соответствующий текст. Затем Мэрилин собирается изучать набранные сообщения на своем терминале, чтобы проверить, что они содержат точные ее слова. Может ли новая секретарша по-прежнему воспользоваться атакой, основанной на задаче о днях рождения, чтобы фальсифицировать сообщение, и если да, то как? *Подсказка*: может.
22. Рассмотрите пример неудачной попытки получения Алисой открытого ключа Боба (см. рис. 8.20). Допустим, они уже установили общий закрытый ключ, однако Алиса все же хочет узнать открытый ключ Боба. Существует ли в данной ситуации безопасный способ его получения? Если да, то какой?
23. Алиса желает общаться с Бобом, используя шифрование с открытым ключом. Она устанавливает соединение с кем-то, кто, по ее предположению, является Бобом. Она спрашивает у него открытый ключ, и тот отсылает его вместе с сертификатом X.509, подписанным Центральным управлением. У Алисы уже есть открытый ключ ЦУ. Что должна теперь сделать Алиса, чтобы удостовериться, что она действительно общается с Бобом? Допустим, Бобу безразлично, с кем он общается (то есть под именем Боба мы здесь понимаем, например, какую-нибудь общедоступную службу).
24. Допустим, система использует PKI, базирующийся на древовидной иерархии Управлений сертификации. Алиса желает пообщаться с Бобом и после установления соединения получает от него сертификат, подписанный УС *X*. Допустим, Алиса никогда не слышала про *X*. Что ей нужно сделать, чтобы удостовериться, что на той стороне канала связи находится именно Боб?
25. Может ли IPsec с заголовком AH использоваться в транспортном режиме, если одна из машин находится за NAT-блоком? Ответ поясните.
26. В чем заключается одно из преимуществ использования HMAC над использованием RSA при создании подписей хэшей SHA-1?
27. Назовите одну причину, по которой брандмауэры следует настраивать на анализ входящего трафика. Теперь назовите одну причину, по которой брандмауэры следует настраивать на анализ исходящего трафика. Как вы думаете, насколько успешен такой анализ?
28. Формат WEP-пакета показан на рис. 8.27. Допустим, используется 32-битная контрольная сумма, вычисляемая путем суммирования по модулю 2 всех 32-битных слов заголовка. Предположим, что проблема RC4 решилась его за-

меной на потоковый шифр без изъянов, а вектор инициализации расширен до 128 бит. Может ли злоумышленник каким-то образом незаметно заниматься шпионажем или изменять сообщения?

29. Допустим, организация установила виртуальную частную сеть для обеспечения безопасной связи между своими сайтами в Интернете. Имеет ли смысл Джиму, сотруднику этой организации, для общения с Мэри, сотрудницей той же организации, применять шифрование или какие-то еще механизмы защиты информации?
30. Измените одно сообщение в протоколе, изображенном на рис. 8.30, так, чтобы протокол стал устойчивым к зеркальной атаке. Объясните суть ваших изменений.
31. Для установки секретного ключа между Алисой и Бобом используется алгоритм Диффи—Хеллмана. Алиса посылает Бобу (719, 3, 191). Боб отвечает (543). Секретное число Алисы $x = 16$. Чему равен секретный ключ?
32. Если Алиса никогда не встречалась с Бобом, если они не пользовались общим ключом, не имеют сертификатов, они все равно могут установить общий закрытый ключ при помощи алгоритма Диффи—Хеллмана. Объясните, почему так тяжело бороться с атакой типа «человек посередине».
33. Почему в протоколе, изображенном на рис. 8.35, *A* посылается открытым текстом вместе с зашифрованным ключом сеанса?
34. В протоколе, изображенном на рис. 8.35, мы отмечали, что начинать каждое сообщение с 32 нулевых битов рискованно с точки зрения безопасности. Предположим, что каждое сообщение начинается со случайного числа или второго секретного ключа, известного только пользователю и центру распространения ключей. Защищает ли это от атаки методом известного открытого текста?
35. В протоколе Нидхэма—Шрёдера (Needham—Schroeder) Алиса формирует два оклика, R_A и R_{A2}. Не достаточно ли будет использовать один оклик?
36. Предположим, что организация для аутентификации применяет метод Kerberos. Как в терминах безопасности и работоспособности повлияет на систему выход из строя сервера аутентификации или сервера выдачи билетов?
37. В протоколе аутентификации с открытым ключом, показанном на рис. 8.39, в сообщении 7 случайное число R_B зашифровано ключом K_S. Необходимо ли это шифрование или допустимо отправить это число обратно открытым текстом? Ответ поясните.
38. У кассовых аппаратов, использующих магнитные карты и PIN-коды, есть существенный недостаток: кассир-злоумышленник может модифицировать считывающее устройство своего кассового аппарата, чтобы считать и сохранить всю информацию, хранящуюся на карте, включая PIN-код, с целью послать дополнительные (поддельные) транзакции в будущем. В следующем поколении кассовых терминалов будут использоваться карты с полноценным центральным процессором, клавиатурой и небольшим дисплеем. Разработайте для этой системы протокол, который не сможет взломать кассир-злоумышленник.

39. Назовите *две* причины, по которым PGP использует сжатие сообщений.
40. Предположим, что повсеместно в Интернете используется PGP. Можно ли в этом случае послать PGP-сообщение по произвольному интернет-адресу и быть полностью уверенным в том, что на приемной стороне оно будет корректно расшифровано? Обсудите свой ответ.
41. На атаке, показанной на рис. 8.43, пропущен один шаг. Он не является необходимым для того, чтобы атака была удачной, однако его наличие может помочь уменьшить возможные подозрения. Итак, что же пропущено?
42. Однажды было предложено в целях противостояния атакам DNS-серверов использовать предсказание идентификаторов (то есть ввести случайные, а не последовательные идентификаторы). Обсудите это предложение с точки зрения защиты информации.
43. Протокол передачи данных SSL подразумевает наличие двух нонсов и подготовительного ключа. Какую роль играют нонсы (если они вообще играют какую-либо роль)?
44. Изображение на рис. 8.50, *б* содержит ASCII-текст пяти пьес Шекспира (хотя по фотографии этого не скажешь). Можно ли спрятать музыку, а не текст, между зебрами? Если да, то как и сколько? Если нет, то почему?
45. Алиса была постоянным пользователем анонимной рассылки первого типа. Она могла помещать сколько угодно сообщений в свою любимую конференцию *alt.fanclub.alice*, но все знали, что их автором является Алиса, так как все письма были подписаны одним и тем же псевдонимом. Если предполагать, что рассылка работает правильно, у Труди нет возможности писать от имени Алисы. После того как все анонимные рассылки первого типа были закрыты, Алиса перешла на шифрованные панковские рассылки и начала обсуждать новую тему в своей конференции. Предложите способ защиты в новых условиях от Труди, пытающейся писать письма от чужого имени.
46. Найдите в Интернете описание какого-нибудь интересного случая, касающегося конфиденциальности, и напишите небольшой отчет на эту тему.
47. Найдите в Интернете описание очередного случая противостояния закона об авторских правах добросовестному использованию продукции и напишите небольшой отчет о том, что вы узнали.
48. Напишите программу, шифрующую входные данные путем суммирования по модулю 2 с ключевым потоком. Найдите или напишите сами хороший генератор случайных чисел для создания ключевого потока. Программа должна работать подобно фильтру, принимая на входе открытый текст и выдавая на выходе на стандартное устройство вывода шифр (и наоборот). У программы должен быть один параметр: ключ, запускающий генератор случайных чисел.
49. Напишите процедуру для вычисления хэша SHA-1 блока данных. У процедуры должно быть два параметра: указатель на входной буфер и указатель на 20-байтовый выходной буфер. Чтобы найти спецификацию SHA-1, поищите в Интернете FIPS 180-1, в этом документе содержится полное описание.

Глава 9
Библиография

- Литература для дальнейшего чтения
- Алфавитный список литературы

Мы закончили изучение компьютерных сетей, но все это — только начало. Многие интересные темы не были рассмотрены во всей полноте и со всеми подробностями, а многие вопросы были вообще опущены из-за отсутствия места. Эта глава содержит список дополнительной литературы для читателей, желающих продолжить изучение компьютерных сетей.

Литература для дальнейшего чтения

Существует большое количество книг, касающихся всех аспектов компьютерных сетей и распределенных систем. Среди журналов, часто публикующих статьи по этой теме, стоит выделить следующие три: *IEEE Transactions on Communications*, *IEEE Journal on Selected Areas in Communications* и *Computer Communication Review*. Многие другие журналы также иногда публикуют статьи по теме компьютерных сетей.

Кроме того, Институт инженеров по электротехнике и электронике IEEE выпускает еще три журнала: *IEEE Internet Computing*, *IEEE Network Magazine* и *IEEE Communications Magazine* — в них содержатся обзоры, учебные статьи и информация об исследованиях, связанные с компьютерными сетями. Первые два в основном посвящены архитектуре, стандартам и программному обеспечению, тогда как журнал *IEEE Communications Magazine* большей частью занят освещением технологий коммуникаций (оптоволоконной, спутниковой связи и т. д.).

Помимо этого ежегодно или раз в два года проводятся несколько конференций, которые освещаются в многочисленных статьях, посвященных сетям и распределенным системам, в частности, ежегодная конференция *SIGCOMM, The*

International Conference on Distributed Computer Systems и *The Symposium on Operating Systems Principles.*

Далее мы перечислим дополнительную литературу, сгруппированную по главам этой книги. Большинство книг представляют собой самоучители либо обзоры. Иногда даются ссылки на главы из руководств.

Введение и неспециализированная литература

Bi и др., «Wireless Mobile Communications at the Start of the 21st Century» Новый век, новые технологии. Звучит здорово. Приводится история беспроводной связи, а также рассматриваются основные вопросы, включая стандарты, применения, Интернет и технологии.

Comer, «The Internet Book» Сюда стоит заглянуть всем, кто ищет простое и понятное описание Интернета. В этой книге в доступной даже для новичка форме рассказывается об истории, развитии, технологиях, протоколах и службах Интернета. Тем не менее, эта книга будет интересна и более подготовленным читателям благодаря большому количеству содержащегося в ней материала.

Garber, «Will 3G Really Be the Next Big Wireless Technology?» Предполагается, что третье поколение мобильных телефонов будет сочетать в себе возможности передачи как голоса, так и данных со скоростью до 2 Мбит/с. Движение в эту сторону уже началось. В этой статье, которая читается очень легко, дан обзор возможностей, подводных камней, технологий, политических и экономических аспектов применения широкополосных беспроводных коммуникаций.

IEEE Internet Computing, Jan.-Feb. 2000 В первом выпуске нового тысячелетия журнал *IEEE Internet Computing* представил то, что и ожидалось: размышления людей, участвовавших в создании Интернета, о том, каким он будет в новом веке. В обсуждении участвуют такие эксперты, как Поль Бэрэн (Paul Baran), Лоуренс Робертс (Lawrence Roberts), Леонард Кляйнрок (Leonard Kleinrock), Стефан Крокер (Stephen Crocker), Дэнни Коэн (Danny Cohen), Боб Меткалф (Bob Metcalfe), Билл Гейтс (Bill Gates), Билли Джой (Billy Joy) и др. Советую подождать лет 500 и только потом перечитать эти предсказания.

Kipnis, «Beating the System: Abuses of the Standards Adoption Process» Комитеты по стандартизации пытаются работать максимально добросовестно и независимо от разработчиков, но, к сожалению, некоторые компании пытаются нарушить эту систему. Например, уже не раз случалось так, что компания помогает в разработке стандарта, а после его утверждения заявляет, что стандарт основывается на принадлежащем ей патенте и что вопросы выдачи лицензий и цен на них компания будет решать сама. Данный материал будет полезен тем, кто хочет узнать о нелицеприятной стороне стандартизации.

Kyas и Crawford, «ATM Networks» Стандарт ATM был когда-то разрекламирован как сетевой протокол будущего, и он до сих пор остается весьма значимым в телефонной системе. Эта книга отражает нынешнее состояние дел ATM, содержит детальное описание протоколов ATM и их интеграции с сетями на базе IP.

Kwok, «A Vision for Residential Broadband Service» Если вам интересно, что думала корпорация Microsoft об организации видео по заказу в 1995 году, эта статья для вас. Спустя пять лет эта точка зрения безнадежно устарела. Назначение этой статьи состоит в том, чтобы показать, что даже высокообразованные и подкованные эксперты иногда не способны предвидеть развитие ситуации даже на пять лет вперед. Это должно послужить хорошим уроком для всех нас.

Naughton, «A Brief History of the Future» Кто же все-таки изобрел Интернет? Многие хотят, чтобы их включили в число изобретателей. И некоторые из них по праву этого заслуживают. Правдивая история Интернета, рассказанная остроумно и завораживающе, перемежающаяся историческими анекдотами (например, о том, как компания AT&T многократно разъясняла всем и искренне верила сама, что у цифровых коммуникаций нет будущего) — вот суть этой книги.

Perkins, «Mobile Networking in the Internet» Здесь вы найдете качественное описание всех уровней протоколов мобильных сетей, от физического до транспортного, включая такие вопросы как защита информации и специализированные сети.

Teger и Waks, «End-User Perspectives on Home Networking» Домашние сети не похожи на корпоративные. Отличаются и приложения (в домашних сетях более интенсивно используется мультимедиа), и оборудование, и сами пользователи, которые обычно мало искушены в технических вопросах и не понимают, что делать при возникновении каких бы то ни было неполадок. Более подробную информацию вы найдете в этой книге.

Varshney и Vetter, «Emerging Mobile and Wireless Networks» Еще одна книга, посвященная основам беспроводной связи. Здесь рассматриваются беспроводные ЛВС, местные линии связи, спутниковые сети, а также некоторые программные продукты и приложения.

Wetteroth, «OSI Reference Model for Telecommunications» Несмотря на то, что сами протоколы OSI в чистом виде сейчас не используются, семиуровневая модель стала очень известной. В этой книге не только подробно рассказывается об OSI, но и описывается связь этой модели с телефонными (то есть *не* компьютерными) сетями. Показано, как обычная телефония и другие голосовые протоколы вписываются в стек сетевых протоколов.

Физический уровень

Abramson, «Internet Access Using VSATs» В развитых странах маленькие наземные станции широко применяются как для организации телефонной связи в сельской местности, так и для корпоративного доступа в Интернет. Однако природа трафика в этих двух случаях очень разная, соответственно, нужны разные протоколы. В этой статье изобретатель системы ALOHA описывает методы выделения каналов, которые могут использоваться в системах VSAT.

Alkhatib и др., «Wireless Data Networks: Reaching the Extra Mile» Небольшое введение в технологии (включая расширенный спектр) и терминологию беспроводных сетей, которое можно использовать в качестве самоучителя.

Azzam и Ransom, «Broadband Access Technologies» В качестве технологий сетевого доступа здесь рассматриваются телефонные системы, оптоволоконные сети, ADSL, кабельные сети, спутники и даже линии электропередач. Разговор идет также о домашних сетях, службах, производительности и стандартах. В конце книги приведены биографии наиболее значимых телекоммуникационных и сетевых компаний, однако, учитывая быстро меняющуюся конъюнктуру этого рынка, можно предположить, что эта последняя глава устареет быстрее, чем остальные.

Bellamy, «Digital Telephony» В этом солидном издании содержится все, что вы когда-либо хотели узнать о телефонной системе, и даже более того. Особый интерес представляют главы, посвященные передаче данных и мультиплексированию, цифровой коммутации, волоконной оптике, мобильной телефонии и DSL.

Berezdivin и др., «Next-Generation Wireless Communications Concepts and Technologies» Авторы этой книги опередили всех остальных, так как рассказывают о четвертом поколении беспроводных сетей. Ожидается, что эти сети будут повсеместно предоставлять услуги IP, а значит, и доступ в Интернет, обеспечивая высокую пропускную способность и отличное качество обслуживания. Этого можно добиться грамотным распределением спектра, динамическим управлением ресурсами, адаптивным обслуживанием. Все это звучит несколько утопически сейчас, однако не менее утопически звучали слова о мобильной телефонии в 1995 году.

Dutta-Roy, «An Overview of Cable Modem Technology and Market Perspectives» Кабельное телевидение уже давно перестало быть службой передачи телевизионного сигнала и превратилось в сложную распределенную систему, совмещающую телевидение, Интернет и телефонию. Эти изменения заметно повлияли на инфраструктуру кабельных систем. Статья посвящена обсуждению кабельных участков сетей связи, стандартам, а также маркетинговым аспектам, особый упор делается на DOCSIS.

Farserotu и Prasad, «A Survey of Future Broadband Multimedia Sattelite Systems, Issues, and Trends» В небе над нами летает множество спутников передачи данных, среди которых Astrolink, Cyberstar, Spaceway, Skybridge, Teledisc и iSky. Немало спутниковых программ в настоящее время еще только разрабатываются. В них применяются различные технологии, включая методы «трубы» и коммутацию спутников. В предлагаемом вашему вниманию материале дается обзор различных спутниковых систем и технологий.

Hu и Li, «Sattelite-Based Internet: A Tutorial» Доступ в Интернет через спутник отличается от доступа с помощью наземных линий связи. Здесь должны учитываться не только задержки, но и маршрутизация, а также коммутация. Авторы рассматривают некоторые проблемы применения спутниковых систем для доступа в Интернет.

Joel, «Telecommunications and the IEEE Communications Society» Здесь в очень сжатой, но удивительно понятной форме описывается история телекоммуникаций, начиная с телеграфа и заканчивая сетями стандарта 802.11. Вы найдете разделы, посвященные радио, телефонии, аналоговой и цифровой коммутации, подводным кабелям, цифровой передаче данных, ATM, телевизионному вещанию, спутникам, кабельному ТВ, оптическим линиям связи, мобильным телефонам, пакетной коммутации, ARPANET и, конечно же, Интернету.

Metcalfe, «Computer/Network Interface Design: Lessons from Arpanet & Ethernet» Хотя инженеры занимаются созданием сетей уже несколько десятков лет, иногда задаешься вопросом, научились ли они чему-либо за это время? В этой статье разработчик сетей Ethernet рассказывает, как построить сетевой интерфейс и что с ним делать после этого. Автор откровенно сообщает, где он ошибался, а где был прав.

Palais, «Fiber Optic Communications», 3-е издание Обычно книги по оптоволоконной технологии предназначаются для специалистов, но эта книга написана более доступным языком. В этой книге рассказывается про волноводы, источники света, детекторы света, соединительные муфты, модуляцию, шум и др.

Pandya, «Emerging Mobile and Personal Communication Systems» Эта статья представляет собой краткое и забавное введение в портативные системы связи. Одна из девяти страниц статьи содержит список из 70 сокращений, используемых на остальных восьми страницах.

Sarikaya, «Packet Mode in Wireless Networks: Overview of Transition to Third Generation» Суть сотовых систем третьего поколения состоит в беспроводной передаче данных. Обзор того, как передача данных осуществляется в сетях второго поколения и какие изменения ждут нас с приходом третьего поколения, изложен в этой книге. Среди обсуждаемых тем GPRS, IS-95B, WCDMA и CDMA2000.

Уровень передачи данных

Carlson, PPP Design, Implementation and Debugging», 2-е издание Если вы хотите узнать о подробностях реализации протоколов, входящих в PPP, включая CCP (Сжатие) и ECP (Шифрование), эта книга послужит хорошим подспорьем. Много внимания, в частности, уделяется одной из популярных реализаций PPP, а именно ANU PPP-2.3.

Gravano, «Introduction to Error Control Codes» Ошибки проникают практически во все цифровые системы связи, поэтому было разработано много типов кодов обнаружения и исправления ошибок. В этой книге рассказывается о наиболее важных из них, начиная от простых линейных кодов Хэмминга и заканчивая кодами, в основе которых лежит теория полей Галуа, а также сверточными кодами. Разумеется, автор оперирует лишь минимально необходимыми алгебраическими понятиями, но и это может показаться слишком сложным человеку, не искушенному в специальных разделах высшей математики.

Holtzman, «Design and Validation of Computer Protocols» Читателям, интересующимся более формальным аспектами протоколов передачи данных (и подобным им), следует прочитать эту книгу. Здесь подробно описываются спецификация, моделирование и тестирование таких протоколов.

Peterson и Davie, «Computer Networks: A System Approach» В главе 2 этой книги содержится материал, касающийся проблем уровня передачи данных, включая формирование кадров, обнаружение ошибок, протоколы с ожиданием, протоколы скользящего окна и локальные сети IEEE 802.

Stallings, «Data and Computer Communications» В главе 7 описывается уровень передачи данных и связанные с ним вопросы управления потоком и обнаружения ошибок, а также базовые протоколы этого уровня, включая протокол с ожиданием и возвратом на *n*. Описаны и протоколы типа HDLC.

Подуровень управления доступом к носителю

Bhagwat, «Bluetooth: Technology for Short-Range Wireless Apps» Простое и понятное введение в систему Bluetooth. Основные протоколы и профили, радиосвязь, пикосети, связи, а также основы различных протоколов — все это составляет книгу.

Bisdikian, «An Overview of the Bluetooth Wireless Technology» Как и представленный выше материал, это хорошая отправная точка для тех, кто хочет узнать больше о системе Bluetooth. Кроме всего прочего, рассматриваются пикосети, стек протоколов и профили.

Crow и др., «IEEE 802.11 Wireless Local Area Networks» Введение в технологии и протоколы 802.11, написанное довольно простым языком. Упор делается на подуровень управления доступом к носителю (MAC-подуровень). Обсуждается как распределенное, так и централизованное управление. В конце приводится анализ производительности 802.11 при различных условиях.

Eklund и др., «IEEE Standard 802.16: A Technical Overview of the Wireless MAN Air Interface for Broadband Wireless Access» Беспроводные местные линии связи, стандартизованные IEEE в 2002 году (стандарт 802.16), могут совершить настоящую революцию на рынке услуг телефонной связи, позволив провести широкополосные линии в каждый дом. В данном обзоре автор поясняет основные технологические вопросы, связанные с этим стандартом.

Kapp, «802.11: Leaving the Wire Behind» Краткое введение в 802.11 включает в себя описание основ, протоколов и наиболее значимых стандартов.

Kleinrock, «On Some Principles of Nomadic Computing and Multi-Access Communications» Беспроводный доступ по общему разделяемому каналу — это более сложный вопрос, чем разделение обычного проводного канала. Кроме всего прочего, возникают проблемы динамических топологий, маршрутизации и управления питанием. В этой статье описываются эти и другие вопросы доступа к каналу со стороны мобильных беспроводных устройств.

Miller и Cummins, «LAN Technologies Explained» Хотите узнать больше о технологиях, применяемых в локальных сетях? В этой книге описано большинство из них, включая FDDI и маркерное кольцо и, разумеется, Ethernet. Сети первых двух типов сейчас устанавливают довольно редко, однако эти технологии широко применяются в уже существующих сетях, а кольцевая организация и сейчас очень популярна (взять хотя бы SONET).

Perlman, «Interconnections», 2-е издание Это заслуживающая доверия и в то же время увлекательно написанная книга о мостах, маршрутизаторах и маршрутизации в целом. Автор книги сам участвовал в разработке алгоритмов, применяемых в мосте связующего дерева в сетях стандарта IEEE 802, так что он, несомненно, является одним из ведущих экспертов в области различных аспектов сетевых технологий.

Webb, «Broadband Fixed Wireless Access» Здесь вы найдете ответы на вопросы, как и зачем функционируют стационарные беспроводные широкополосные сети. В разделе «Зачем?» приводится следующий аргумент: людям надоедает иметь разные домашний и рабочий электронные адреса, отдельные номера домашнего телефона, рабочего и мобильного, а также центра диалогового обмена сообщениями, да к тому же еще один-два номера факса. Хочется иметь единую интегрированную систему, которая работала бы везде. В разделе, посвященном технологиям («Как?»), упор делается на физический уровень, здесь вы найдете сравнения TDD и FDD, адаптивной и постоянной модуляций, а также большого числа носителей.

Сетевой уровень

Bhatti и Crowcroft, «QoS Sensitive Flows: Issues in IP Packet Handling» Один из способов достижения высокого качества обслуживания в сетях заключается в тщательном составлении расписания отправки пакетов с каждого маршрутизатора. В этой статье более или менее детально рассматриваются алгоритмы планирования и смежные вопросы.

Chakrabarti, «QoS Issues in Ad Hoc Wireless Networks» Маршрутизация в специализированных сетях, составленных из ноутбуков, оказавшихся рядом, достаточно сложна, даже если не задумываться о качестве обслуживания. Однако людям все же важно качество обслуживания, поэтому на данный вопрос приходится обращать внимание. В этой статье обсуждаются природа специализированных сетей и некоторые проблемы маршрутизации и качества обслуживания.

Comer, «Internetworking with TCP/IP», том 1, 4-е издание Автор представил наиболее полный труд о наборе протоколов TCP/IP. Главы с 4-й по 11-ю посвящены протоколу IP и родственным протоколам сетевого уровня. В остальных главах, также заслуживающих внимания, описываются в основном более высокие уровни.

Hiutema, «Routing in the Internet» Если вы хотите знать абсолютно все о маршрутизации в Интернете, то эта книга для вас. В ней детально описываются алгоритмы как с произносимыми названиями (например, RIP, CIDR и MBONE), так

и с непроизносимыми (OSPF, IGRP, EGP и BGP). В книге рассказывается также о новых возможностях, таких как многоадресная рассылка, мобильный IP-протокол и резервирование от источника.

Malhotra, «IP Routing» Детальное описание IP-маршрутизации. Среди рассматриваемых протоколов — RIP, RIP-2, IGRP, EIGRP, OSPF и BGP-4.

Metz, «Differentiated Services» Гарантии качества обслуживания важны для многих мультимедийных приложений. Интегрированные и дифференцированные услуги представляют собой два возможных подхода к обеспечению качества обслуживания. Здесь обсуждается и то, и другое, однако упор делается на дифференцированное обслуживание.

Metz, «IP routers: New Tool for Gigabit Networking» Большинство литературы, посвященной темам, обсуждаемым в главе 5 нашей книги, связано с алгоритмами маршрутизации. Здесь же рассказывается о том, как реально работают маршрутизаторы. Они прошли в своем развитии долгий путь от рабочих станций общего назначения до узкоспециализированных устройств маршрутизации. Если вы хотите познакомиться с маршрутизаторами поближе, советуем вам для начала прочесть эту статью.

Nemeth и др., «UNIX System Administration Handbook» Надо сказать, что глава 13 этой книги представляет собой наиболее практическое руководство по сравнению со всеми остальными ссылками. Здесь не описываются абстрактные концепции, а даются практические советы о том, что делать в той или иной ситуации, возникающей при управлении реальной сетью.

Perkins, «Mobile Networking through Mobile IP» Мобильные вычислительные устройства становятся все популярнее, а с ними и протокол Mobile IP. Этот краткий самоучитель представляет собой хорошее введение в курс дела.

Perlman, «Interconnections: Bridges and Routers», 2-е издание В главах с 12-й по 15-ю автор описывает многочисленные аспекты разработки алгоритмов одноадресной и групповой рассылки, как для глобальных, так и для локальных сетей, и их реализацию в различных протоколах. Однако интереснее всего читать главу 18, в которой автор делится своими личными впечатлениями о работе с сетевыми протоколами. Эта глава и информативна, и весьма забавна.

Puzmanova, «Routing and Switching: Time of Convergence?» В конце 1990-х некоторые производители оборудования стали называть коммутаторами все подряд, а менеджеры многих крупных сетей стали говорить, что они «переходят с маршрутизаторов на коммутаторы». Как следует из названия книги («Маршрутизация и коммутация: пора сближения?»), автор пытается угадать будущее обоих типов устройств и проанализировать реальные возможности их совмещения.

Royer и Toh, «A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks» Специализированный алгоритм маршрутизации AODV, который мы рассматривали в главе 5 нашей книги, разумеется, далеко не единственный. Среди прочих стоит назвать DSDV, CGSR, WRP, DSR, TORA, ABR, DRP

и SRP. Все они рассматриваются и сравниваются между собой в этой работе. Кажется, уже сейчас вам должен быть очевиден первый шаг, необходимый для разработки любого специализированного алгоритма маршрутизации: придумывание какого-нибудь ужасного трех- или четырехбуквенного сокращения.

Stevens, «TCP/IP Illustrated», том 1 Главы с 3-й по 10-ю содержат доступное описание протокола IP и родственных ему протоколов (ARP, RARP и ICMP), дополненное примерами.

Streigel и Manimaran, «A Survey of QoS Multicasting Issues» Многоадресная рассылка и качество обслуживания — это те основные проблемы, которые возникают в условиях растущей популярности радио и телевидения в Интернете. Что касается данного обзора, здесь авторы обсуждают то, как в алгоритмах маршрутизации следует учитывать оба этих вопроса.

Yang и Reddy, «A Taxonomy for Congestion Control Algorithms in Packet Switching Networks» Авторы систематизировали алгоритмы борьбы с перегрузкой. Основными категориями являются алгоритмы без обратной связи с управлением от источника и от приемника, а также алгоритмы с явной и неявной обратной связью. В книге классифицируются и описываются 23 существующих алгоритма.

Транспортный уровень

Comer, Internetworking with TCP/IP, том 1, 4-е издание Как уже говорилось ранее, автор написал наиболее полный труд о наборе протоколов TCP/IP. Глава 12 посвящена протоколу UDP, а глава 13 — протоколу TCP.

Hall и Cerf, «Internet Core Protocols: The Definitive Guide» Если вы предпочитаете получать информацию из первых рук и вам хочется узнать побольше о TCP, эта книга для вас. Один из авторов, Cerf, как-никак был одним из разработчиков TCP. В главе 7 вы найдете качественное описание TCP, показывающее, как интерпретировать информацию, получаемую в результате анализа протокола и с помощью инструментария управления сетью. В остальных главах рассказывается про UDP, IGMP, ICMP и ARP.

Kurose и Ross, «Computer Networking: A Top-Down Approach Featuring the Internet» Глава 3 этой книги посвящена транспортному уровню и содержит много информации про UDP и TCP. Также обсуждаются протоколы с ожиданием и возвратом на *n*, описанные в главе 3 нашей книги.

Mogul, «IP Network Performance» Несмотря на свое название, эта статья скорее о производительности протокола TCP и производительности сети вообще. Книга содержит большое количество полезных указаний и практических советов.

Peterson и Davie, «Computer Networks: A System Approach» Глава 5 посвящена UDP, TCP и некоторым смежным протоколам. Также вкратце рассказывается о производительности сетей.

Stevens, «TCP/IP Illustrated», том 1 Главы с 17-й по 24-ю содержат доступное описание протокола TCP, дополненное примерами.

Прикладной уровень

Begholz, «Extending Your Markup: An XML Tutorial» Короткое и понятное введение в XML для начинающих.

Cardellini и др., «The State-of-the-Art in Locally Dustributed Web-Server Systems» По мере роста популярности Всемирной паутины некоторым сайтам требуются все большие масштабы серверных ферм для обработки трафика. Одной из тяжелейших задач построения серверных ферм является распределение нагрузки между машинами. В данной работе очень подробно обсуждается эта проблема.

Berners-Lee и др., «The World Wide Web» Взгляд на Паутину и на пути ее развития со стороны человека, который ее придумал, и его коллег по CERN. Статья посвящена архитектуре Паутины, унифицированным указателям (URL), протоколу HTTP, языку HTML, а также перспективам на будущее. Приводится сравнение с другими распределенными информационными средами.

Choudbury и др., «Copyright Protection for Electronic Publishing on Computer Networks» Об алгоритмах шифрования написано огромное количество книг и статей. Однако мало где можно найти сведения о том, как использовать эти алгоритмы, чтобы предотвратить дальнейшее распространение пользователями документов, которые им разрешено расшифровывать. В этой статье описывается несколько способов, полезных для защиты авторских прав в эру электроники.

Collins, «Carrier Grade Voice over IP» Если вы уже прочитали работу Varshney и др. и теперь хотите узнать все подробности передачи голоса поверх IP с использованием H.323, вам следует обратить внимание на этот труд. Несмотря на то, что книга толстая и подробная, по сути своей она является самоучителем и даже, в общем-то, не требует предварительных знаний о телефонных системах.

Davidson, «A Web Chaching Primer» По мере роста Всемирной паутины кэширование становится все более критичным вопросом в деле обеспечения хорошей производительности. Краткое введение, описывающее веб-кэширование, составляет суть этой работы.

Krishnamurthy и Redfox, «Web Protocols and Practice» Тяжело найти более понятную книгу, которая при этом охватывала бы все аспекты Всемирной паутины. Здесь, разумеется, рассказывается о клиентах, серверах, прокси и кэшировании. Однако вряд ли вы могли бы предположить, что есть в этой книге разделы, посвященные трафику и его измерению в Паутине, а также текущим исследованиям и направлениям развития Веб.

Rabinovich и Spatscheck, «Web Caching and Replication» Здесь весьма просто объясняются принципы кэширования и репликации во Всемирной паутине. Очень подробно описываются прокси, кэширование, сети доставки содержимого, выбор сервера и многое другое.

Shahabi и др., «Yima: A Second-Generation Continuous Media Server» Серверы мультимедиа представляют собой сложные системы, которым приходится заниматься планированием загрузки ЦП, размещением файлов на диске, синхронизацией потоков и многим другим. Со временем люди учатся реализовывать их все лучше и лучше. В этой работе представлен обзор архитектуры одной из недавно появившихся систем.

Tittel и др., «Mastering XHTML» Две книги, описывающие новый стандарт языка описания веб-страниц, собраны под одной обложкой. В начале находится текст, описывающий XHTML, в основном в сравнении с обычным HTML. Далее следует удобный справочник по тегам, кодам и спецсимволам, использующимся в XHTML 1.0.

Varshney и др., «Voice over IP» Как работает система передачи голоса поверх IP? Заменит ли она обычную коммутируемую телефонную сеть общего доступа? Об этом вы узнаете, прочитав эту книгу.

Безопасность в сетях

Anderson, R., «Why Cryptosystems Fail» Согласно Андерсону, безопасность банковских систем очень плоха, но не потому, что хитроумные злоумышленники взламывают шифр DES на своих персональных компьютерах. Настоящие проблемы варьируются от нечестных банковских служащих (изменяющих почтовый адрес клиента на свой, чтобы перехватить пластиковую карту и номер PIN) до ошибок в программах (всем клиентам выдается один и тот же номер PIN). Особенно интересна реакция банков на случаи ошибок: наши системы безупречны, так что все подобные случаи происходят в результате ошибок или мошенничества самих клиентов.

Anderson, «Security Engineering» Можно сказать, что это 600-страничная версия описанной ранее книги того же автора. В ней описываются более технические вещи, нежели в *«Secrets and Lies»*, однако менее технические, чем в *«Network Security»* (см. далее). Вначале даются основы методов защиты информации, затем следуют целые главы, посвященные разным приложениям, включая банковские системы, системы управления атомной энергетикой, безопасную печать, биометрию, физическую защиту, электронные войны, защиту в телекоммуникациях, электронную коммерцию и защиту авторских прав. Третья часть книги посвящена политике, управлению и оценке систем.

Brands, «Rethinking Public Key Infrastructures and Digital Certificates» Это больше, чем просто хорошее введение в технологии цифровых сертификатов, — это еще и мощная пропагандистская работа. Автор уверен, что нынешние бумажные системы верификации изжили себя и являются неэффективными, и утверждает, что цифровые сертификаты очень хорошо подходят для таких приложений, как цифровые голосования, управление предоставлением прав и даже в качестве замены бумажных денег. Кроме того, он предупреждает, что без PKI и шифрования Интернет может стать огромной системой слежения.

Kaufman и др., «Network Security», 2-е издание Эту заслуживающую доверия и часто остроумную книгу следует читать в первую очередь, если вас интересует дополнительная информация о техническом аспекте алгоритмов и протоколов безопасности сетей. В ней подробно освещаются алгоритмы и протоколы с секретным и открытым ключом, хэширование сообщений, протокол Kerberos, PKI, IPsec, SSL/TLS и электронная почта. Все темы снабжены примерами. Глава 26, посвященная фольклору на тему защиты информации, — это настоящий шедевр. В деле обеспечения безопасности важны все детали. Если вы собираетесь разработать действительно полезную систему защиты, эта глава подскажет вам в виде примеров из реальной жизни много интересного.

Pohlmann, «Firewall Systems» Брандмауэры — это начало и конец системы защиты от взломщиков. В этой книге поясняется, как работают брандмауэры и чем они занимаются. Начинается описание с простейших программных систем, предназначенных для защиты отдельных ПК, и заканчивается сложными аппаратами, располагающимися между частными сетями и Интернетом.

Schneier, «Applied Cryptography», 2-е издание Этот монументальный сборник является самым страшным кошмаром Агентства Национальной Безопасности США: под одной обложкой собраны описания всех известных криптографических алгоритмов. Мало того, большинство алгоритмов приводятся в этой книге в виде действующих программ на языке C. Также в книге содержится более 1600 ссылок на криптографическую литературу. Если вы *всерьез* решили сохранить содержимое ваших файлов в секрете, прочитайте эту книгу.

Schneier, «Secrets and Lies» Прочитав «*Applied Cryptography*» от корки до корки, вы станете знатоком криптографических алгоритмов. Если же вы после этого не менее внимательно прочтете «*Secrets and Lies*» (что займет уже гораздо меньше времени), то поймете, что одними алгоритмами дело не ограничивается. Слабость большинства систем защиты связана не с плохими алгоритмами или слишком короткими ключами, а с пороками в окружающей эти системы среде. Приведено бессчетное множество примеров возможных угроз, атак, защит от них, контратак и т. д. Эта книга является прекрасным нетехническим описанием систем безопасности, рассматривающим проблему в самом широком смысле.

Skoudis, «Counter Hack» Как остановить взломщика? Надо думать так же, как он. В этой книге показан взгляд на сеть со стороны взломщика; автор утверждает, что защита информации должна быть одной из функций всей сетевой системы в целом, она не должна додумываться и прикручиваться в виде специальной технологии к уже существующим сетям. Рассматриваются почти все типы наиболее распространенных атак, включая «социальный инжиниринг» — тип атаки, рассчитанный на незнание пользователем систем электронной безопасности.

Алфавитный список литературы

Abramson, N.: «Internet Access Using VSATs», IEEE Commun. Magazine, vol. 38, pp. 60–68, July 2000.

Abramson, N.: «Development of the ALOHANET», IEEE Trans, on Information Theory, vol. IT-31, pp. 119–123, March 1985.

Adams, M., and Dulchinos, D.: «OpenCable», IEEE Commun. Magazine, vol. 39, pp. 98–105, June 2001.

Alkhatib, H. S., Bailey, C., Gerla, M., and McRae, J.: «Wireless Data Networks: Reaching the Extra Mile», Computer, vol. 30, pp. 59–62, Dec. 1997.

Anderson, R. J.: «Free Speech Online and Office», Computer, vol. 25, pp. 28–30, June 2002.

Anderson, R. J.: «Security Engineering», New York: Wiley, 2001.

Anderson, R. J.: «The Eternity Service», Proc. First Int'l Conf. on Theory and Appl. of Cryptology, CTU Publishing House, 1996.

Anderson, R. J.: «Why Cryptosystems Fail», Commun. of the ACM, vol. 37, pp. 32–40, Nov. 1994.

Artz, D.: «Digital Steganography», IEEE Internet Computing, vol. 5, pp. 75–80, 2001.

Azzam, A. A., and Ransom, N.: «Broadband Access Technologies», New York: McGraw-Hill, 1999.

Bakne, A., and Badrinath, B. R.: «I-TCP: Indirect TCP for Mobile Hosts», Proc. 15th Int'l Conf. on Distr. Computer Systems, IEEE, pp. 136–143, 1995.

Balakrishnan, H., Seshan, S., and Katz, R. H.: «Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks», Proc. ACM Mobile Computing and Networking Conf., ACM, pp. 2–11, 1995.

Ballardie, T., Francis, P., and Crowcroft, J.: «Core Based Trees (CBT)», Proc. SIGCOMM '93 Conf., ACM, pp. 85–95, 1993.

Barakat, C., Altman, E., and Dabbous, W.: «On TCP Performance in a Heterogeneous Network: A Survey», IEEE Commun. Magazine, vol. 38, pp. 40–46, Jan. 2000.

Bellamy, J.: Digital Telephony, 3rd ed., New Ytfrk: Wiley, 2000.

Bellman, R. E.: Dynamic Programming, Princeton, NJ: Princeton University Press, 1957.

Belsnes, D.: «Flow Control in the Packet Switching Networks», Communications Networks, Uxbridge, England: Online, pp. 349–361, 1975.

Bennet, C. H., and Brassard, G.: «Quantum Cryptography: Public Key Distribution and Coin Tossing», Int'l Conf. on Computer Systems and Signal Processing, pp. 175–179, 1984.

Berezdivin, R., Breinig, R., and Topp, R.: «Next-Generation Wireless Communication Concepts and Technologies», IEEE Commun. Magazine, vol. 40, pp. 108–116, March 2002.

Berghel, H. L.: «Cyber Privacy in the New Millennium», Computer, vol. 34, pp. 132–134, Jan. 2001.

Bergholz, A.: «Extending Your Markup: An XML Tutorial», IEEE Internet Computing, vol. 4, pp. 74–79, July-Aug. 2JX)0.

Berners-Lee, T., Cailliau, A., Loutonen, A., Nielsen, H. F., and Secret, A.: «The World Wide Web», Commun. of the ACM, vol. 37, pp. 76–82, Aug. 1994.

Bertsekas, D., and Gallager, R.: «Data Networks», 2nd ed., Englewood Cliffs, NJ: Prentice Hall, 1992.

Bhagwat, P.: «Bluetooth: Technology for Short-Range Wireless Apps», IEEE Internet Computing, vol. 5, pp. 96–103, May-June 2001.

Bharghavan, V., Demers, A., Shenker, S., and Zhang, L.: «MACAW: A Media Access Protocol for Wireless LANs», Proc. SIGCOMM '94 Conf., ACM, pp. 212–225, 1994.

Bhatti, S. N., and Crowcroft, J.: «QoS Sensitive Flows: Issues in IP Packet Handling», IEEE Internet Computing, vol. 4, pp. 48–57, July-Aug. 2000.

Bi, Q., Zysman, G. I., and Menkes, H.: «Wireless Mobile Communications at the Start of the 21st Century», IEEE Commun. Magazine, vol. 39, pp. 110–116, Jan, 2001.

Biham, E., and Shamir, A.: «Differential Cryptanalysis of the Data Encryption Standard», Proc. 17th Ann. Int'l Cryptology Conf., Berlin: Springer-Verlag LNCS 1294, pp. 513–525, 1997.

Bird, R., Gopal, I., Herzberg, A., Janson, P. A., Kutten, S., Molva, R, and Yung, M.: «Systematic Design of a Family of Attack-Resistant Authentication Protocols», IEEE J. on Selected Areas in Commun., vol. 11, pp. 679–693, June 1993.

Birrell, A. D., and Nelson, B. J.: «Implementing Remote Procedure Calls», ACM Trans, on Computer Systems, vol. 2, pp. 39–59, Feb. 1984.

Biryukov, A., Shamir, A., and Wagner, D.: «Real Time Cryptanalysis of A5/1 on a PC», Proc. Seventh Int'l Workshop on Fast Software Encryption, Berlin: Springer-Verlag LNCS 1978, 2000.

Bisdikian, C.: «An Overview of the Bluetooth Wireless Technology», IEEE Commun. Magazine, vol. 39, pp. 86–94, Dec. 2001.

Blaze, M.: «Protocol Failure in the Escrowed Encryption Standard», Proc. Second ACM Conf. on Computer and Commun. Security, ACM, pp. 59–67, 1994.

Blaze, M., and Bellovin, S.: «Tapping on My Network Door», Commun. of the ACM, vol. 43, p. 136, Oct. 2000.

Bogineni, K., Sivalingam, K. M., and Dowd, P. W.: «Low-Complexity Multiple Access Protocols for Wavelength-Division Multiplexed Photonic Networks», IEEE Journal on Selected Areas in Commun., vol. 11, pp. 590–604, May 1993.

Bolcskei, H., Paulraj, A. J., Hari, K. V. S., and Nabar, R. U.: «Fixed Broadband Wireless Access: State of the Art, Challenges, and Future Directions», IEEE Commun. Magazine, vol. 39, pp. 100–108, Jan. 2001.

Borisov, N., Goldberg, I., and Wagner, D.: «Intercepting Mobile Communications: The Insecurity of 802.11», Seventh Int'l Conf. on Mobile Computing and Networking, ACM, pp. 180–188, 2001.

Brands, S.: «Rethinking Public Key Infrastructures and Digital Certificates», Cambridge, MA: M.I.T. Press, 2000.

Bray, J., and Sturman, C. F.: «Bluetooth 1.1: Connect without Cables», 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2002.

Breyer, R., and Riley, S.: «Switched, Fast, and Gigabit Ethernet», Indianapolis, IN: New Riders, 1999.

Brown, S.: «Implementing Virtual Private Networks», New York: McGraw-Hill, 1999.

Brown, L., Kwan, M., Pieprzyk, J., and Seberry, J.: «Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI», ASIACRYPT '91 Abstracts, pp. 25–30, 1991.

Burnett, S., and Paine, S.: «RSA Security's Official Guide to Cryptography», Berkeley, CA: Osborne/McGraw-Hill, 2001.

Capetanakis, J. I.: «Tree Algorithms for Packet Broadcast Channels», IEEE Trans, on Information Theory, vol. IT-25, pp. 505–515, Sept. 1979.

Cardellini, V., Casalicchio, E., Colajanni, M., and Yu, P. S.: «The State-of-the-Art in Locally Distributed Web-Server Systems», ACM Computing Surveys, vol. 34, pp. 263–311, June 2002.

Carlson, J.: «PPP Design, Implementation and Debugging», 2nd ed., Boston: Addison-Wesley, 2001.

Cere, V., and Kahn, R.: «A Protocol for Packet Network Interconnection», IEEE Trans, on Commun., vol. COM-22, pp. 637–648, May 1974.

Chakrabarti, S.: «QoS Issues in Ad Hoc Wireless Networks», IEEE Commun. Magazine, vol. 39, pp. 142–148, Feb. 2001.

Chase, J. S., Gallatin, A. J., and Yocum, K. G.: «End System Optimizations for High-Speed TCP», IEEE Commun. Magazine, vol. 39, pp. 68–75, April 2001.

Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R.: «Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks», ACM Wireless Networks, vol. 8, Sept. 2002.

Chen, K.-C.: «Medium Access Control of Wireless LANs for Mobile Computing», IEEE Network Magazine, vol. 8, pp. 50–63, Sept.-Oct. 1994.

Choudbury, A. K., Maxemchuk, N. F., Paul, S., and Schulzrinne, H. G.: «Copyright Protection for Electronic Publishing on Computer Networks», IEEE Network Magazine, vol. 9, pp. 12–20, May-June 1995.

Chu, Y., Rao, S. G., and Zhang, H.: «A Case for End System Multicast», Proc. Int'l Conf. on Measurements and Modeling of Computer Syst., ACM, pp. 1–12, 2000.

Clark, D. D.: «The Design Philosophy of the DARPA Internet Protocols», Proc. SIGCOMM '88 Conf., ACM, pp. 106–114, 1988.

Clark, D. D.: «Window and Acknowledgement Strategy in TCP», RFC 813, July 1982.

Clark, D. D., Da Vie, B. S., Farber, D. J., Gopal, I. S., Kadaba, B. K., Sincoskie, W. D., Smith, J. M., and Tennenhouse, D. L.: «The Aurora Gigabit Testbed», Computer Networks and ISDN Systems, vol. 25, pp. 599–621, Jan. 1993.

Clark, D. D., Jacobson, V., Romkey, J., and Sal Wen, H.: «An Analysis of TCP Processing Overhead», IEEE Commun. Magazine, vol. 27, pp. 23–29, June 1989.

Clark, D. D., Lambert, M., and Zhang, L.: «NETBLT: A High Throughput Transport Protocol», Proc. SIGCOMM '87 Conf., ACM, pp. 353–359, 1987.

Clarke, A. C.: «Extra-Terrestrial Relays», Wireless World, 1945.

Clarke, I., Miller, S. G., Hong, T. W., Sandberg, O., and Wiley, B.: «Protecting Free Expression Online with Freenet», IEEE Internet Computing, vol. 6, pp. 40–49, Jan.-Feb. 2002.

Collins, D.: «Carrier Grade Voice over IP», New York: McGraw-Hill, 2001.

Collins, D., and Smith, C.: «3G Wireless Networks», New York: McGraw-Hill, 2001.

Comer, D. E.: «The Internet Book», Englewood Cliffs, NJ: Prentice Hall, 1995.

Comer, D. E.: «Internetworking with TCP/IP», vol. 1, 4th ed., Englewood Cliffs, NJ: Prentice Hall, 2000.

Costa, L. H. M. K., Fdida, S., and Duarte, O. C. M. B.: «Hop by Hop Multicast Routing Protocol», Proc. 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Commun., ACM, pp. 249–259, 2001.

Craver, S. A., Wu, M., Liu, B., Stubblefield, A., Swartzlander, B., Wallach, D. W., Dean, D., and Felten, E. W.: «Reading Between the Lines: Lessons from the SDMI Challenge», Proc. 10th USENIX Security Symp., USENIX, 2001.

Crespo, P. M., Honig, M. L., and Salehi, J. A.: «Spread-Time Code-Division Multiple Access», IEEE Trans, on Commun., vol. 43, pp. 2139–2148, June 1995.

Crow, B. P., Widjaja, I, Kim, J. G., and Sakai, P. T.: «IEEE 802.11 Wireless Local Area Networks», IEEE Commun. Magazine, vol. 35, pp. 116–126, Sept. 1997.

Crowcroft, J., Wang, Z., Smith, A., and Adams, J.: «A Rough Comparison of the IETF and ATM Service Models», IEEE Network Magazine, vol. 9, pp. 12–16, Nov.-Dec. 1995.

Dabek, F., Brunskill, E., Kaashoek, M. F., Karger, D., Morris, R., Stoica, R., and Balakrishnan, H.: «Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service», Proc. 8th Workshop on Hot Topics in Operating Systems, IEEE, pp. 71–76, 2001a.

Dabek, F., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I.: «Wide-Area Cooperative Storage with CFS», Proc. 18th Symp. on Operating Systems Prin., ACM, pp. 202–215, 2001b.

Daemen, J., and Rijmen, V.: «The Design of Rijndael», Berlin: Springer-Verlag, 2002.

Danthine, A. A. S.: «Protocol Representation with Finite-State Models», IEEE Trans, on Commun., vol. COM-28, pp. 632–643, April 1980.

Davidson, J., and Peters, J.: «Voice over IP Fundamentals», Indianapolis, IN: Cisco Press, 2000.

Davie, B., and Rekhter, Y.: «MPLS Technology and Applications», San Francisco: Morgan Kaufmann, 2000.

Davis, P. T., and McGuffin, C. R.: «Wireless Local Area Networks», New York: McGraw-Hill, 1995.

Davison, B. D.: «A Web Caching Primer», IEEE Internet Computing, vol. 5, pp. 38–45, July-Aug. 2001.

Day, J. D.: «The (Un)Revised OSI Reference Model», Computer Commun. Rev., vol. 25, pp. 39–55, Oct. 1995.

Day, J. D., and Zimmermann, H.: «The OSI Reference Model», Proc. of the IEEE, vol. 71, pp. 1334–1340, Dec. 1983.

De Vriendt, J., Laine, P., Lerouge, C, and Xu, X.: «Mobile Network Evolution: A Revolution on the Move», IEEE Commun. Magazine, vol. 40, pp. 104–111, April 2002.

Deering, S. E.: «SIP: Simple Internet Protocol», IEEE Network Magazine, vol. 7, pp. 16–28, May-June 1993.

Demers, A., Keshav, S., and Shenker, S.: «Analysis and Simulation of a Fair Queue-ing Algorithm», Internetwork: Research and Experience, vol. 1, pp. 3–26, Sept. 1990.

Denning, D. E., and Sacco, G. M.: «Timestamps in Key Distribution Protocols», Commun. of the ACM, vol. 24, pp. 533–536, Aug. 1981.

Diffie, W., and Hellman, M. E.: «Exhaustive Cryptanalysis of the NBS Data Encryption Standard», Computer, vol. 10, pp. 74–84, June 1977.

Diffie, W., and Hellman, M. E.: «New Directions in Cryptography», IEEE Trans, on Information Theory, vol. IT-22, pp. 644–654, Nov. 1976.

Dijkstra, E. W.: «A Note on Two Problems in Connexion with Graphs», Numer. Math., vol. 1, pp. 269–271, Oct. 1959.

Dobrowski, G., and Grise, D.: «ATM and SONET Basics», Fuquay-Varina, NC: APDG Telecom Books, 2001.

Donaldson, G., and Jones, D.: «Cable Television Broadband Network Architectures», IEEE Commun. Magazine, vol. 39, pp. 122–126, June 2001.

Dorfman, R.: «Detection of Defective Members of a Large Population», Annals Math. Statistics, vol. 14, pp. 436–440, 1943.

Doufexi, A., Armour, S., Butler, M., Nix, A., Bull, D., McGeehan, J., and Karlsson, P.: «A Comparison of the HIPERLAN/2 and IEEE 802.11 A Wireless LAN Standards», IEEE Commun. Magazine, vol. 40, pp. 172–180, May 2002.

Durand, A.: «Deploying IPv6», IEEE Internet Computing, vol. 5, pp. 79–81, Jan.-Feb. 2001.

Dutcher, B.: «The NAT Handbook», New York: Wiley, 2001.

Dutta-Roy, A.: «An Overview of Cable Modem Technology and Market Perspectives», IEEE Commun. Magazine, vol. 39, pp. 81–88, June 2001.

Easttom, C.: «Learn JavaScript», Ashburton, U.K.: Wordware Publishing, 2001.

El Gamal, T.: «A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms», IEEE Trans, on Information Theory, vol. IT-31, pp. 469–472, July 1985.

Elhanany, I., Kahane, M., and Sadot, D.: «Packet Scheduling in Next-Generation Multiterabit Networks», Computer, vol. 34, pp. 104–106, April 2001.

Elmirghani, J. M. H., and Mouftah, H. T.: «Technologies and Architectures for Scalable Dynamic Dense WDM Networks», IEEE Commun. Magazine, vol. 38, pp. 58–66, Feb. 2000.

Farserotu, J., and Prasad, R.: «A Survey of Future Broadband Multimedia Satellite Systems, Issues, and Trends», IEEE Commun. Magazine, vol. 38, pp. 128–133, June 2000.

Fiorini, D., Chiani, M., Tralli, V., and Salati., C.: «Can we Trust HDLC?», Computer Commun. Rev., vol. 24, pp. 61–80, Oct. 1994.

Floyd, S., and Jacobson, V.: «Random Early Detection for Congestion Avoidance», IEEE/ACM Trans, on Networking, vol. 1, pp. 397–413, Aug. 1993.

Fluhrer, S., Mantin, I., and Shamir, A.: «Weakness in the Key Scheduling Algorithm of RC4», Proc. Eighth Ann. Workshop on Selected Areas in Cryptography, 2001.

Ford, L. R., Jr., and Fulkerson, D. R.: «Flows in Networks», Princeton, NJ: Princeton University Press, 1962.

Ford, W., and Baum, M. S.: «Secure Electronic Commerce», Upper Saddle River, NJ: Prentice Hall, 2000.

Forman, G. H., and Zahorjan, J.: «The Challenges of Mobile Computing», Computer, vol. 27, pp. 38–47, April 1994.

Francis, P.: «A Near-Term Architecture for Deploying Pip», IEEE Network Magazine, vol. 7, pp. 30–37, May-June 1993.

Fraser, A. G.: «Towards a Universal Data Transport System», in Advances in Local Area Networks, Kummerle, K., Tobagi, F., and Limb, J.O. (Eds.), New York: IEEE Press, 1987.

Frengle, N.: «I-Mode: A Primer», New York: Hungry Minds, 2002.

Gadecki, C., and Heckert, C.: «ATM for Dummies», New York: Hungry Minds, 1997.

Gareer, L.: «Will 3G Really Be the Next Big Wireless Technology?», Computer, vol. 35, pp. 26–32, Jan. 2002.

Garfinkel, S., with Spafford, G.: «Web Security, Privacy, and Commerce», Sebastopol, CA: O'Reilly, 2002.

Geier, J.: «Wireless LANs», 2nd ed., Indianapolis, IN: Sams, 2002.

Gevros, P., Crowcroft, J., Kirstein, P., and Bhatti, S.: «Congestion Control Mechanisms and the Best Effort Service Model», IEEE Network Magazine, vol. 15, pp. 16–25, May-June 2001.

Ghani, N., and Dixit, S.: «TCP/IP Enhancements for Satellite Networks», IEEE Commun. Magazine, vol. 37, pp. 64–72, 1999.

Ginsburg, D.: «ATM: Solutions for Enterprise Networking», Boston: Addison-Wesley, 1996.

Goodman, D. J.: «The Wireless Internet: Promises and Challenges», Computer, vol. 33, pp. 36–41, July 2000.

Goralski, W. J.: «Optical Networking and WDM», New York: McGraw-Hill, 2001.

Goralski, W. J.: «SONET», 2nd ed., New York: McGraw-Hill, 2000.

Goralski, W. J.: «Introduction to ATM Networking», New York: McGraw-Hill, 1995.

Gossain, H., De Morais Cordeiro, and Agrawal, D. P.: «Multicast: Wired to Wireless», IEEE Commun. Mag., vol. 40, pp. 116–123, June 2002.

Gravano, S.: «Introduction to Error Control fades», Oxford, U.K.: Oxford University Press, 2001.

Guo, Y., and Chaskar, H.: «Class-Based Quality of Service over Air Interfaces in 4G Mobile Networks», IEEE Commun. Magazine, vol. 40, pp. 132–137, March 2002.

Haartsen, J.: «The Bluetooth Radio System», IEEE Personal Commun. Magazine, vol. 7, pp. 28–36, Feb. 2000.

Hac, A.: «Wireless and Cellular Architecture and Services», IEEE Commun. Magazine, vol. 33, pp. 98–104, Nov. 1995.

Hac, A., and Guo, L.: «A Scalable Mobile Host Protocol for the Internet», Int'l J. of Network Mgmt, vol. 10, pp. 115–134, May-June, 2000.

Hall, E., and Cerf, V.: «Internet Core Protocols: The Definitive Guide», Sebastopol, CA: O'Reilly, 2000.

Hamming, R. W.: «Error Detecting and Error Correcting Codes», Bell System Tech. J., vol. 29, pp. 147–160, April 1950.

Hanegan, K.: «Custom CGI Scripting with Perl», New York: Wiley, 2001.

Harris, A.: «JavaScript Programming for the Absolute Beginner», Premier Press, 2001.

Harte, L., Kellogg, S., Dreher, R., and Schaffnit, T.: «The Comprehensive Guide to Wireless Technology», Fuquay-Varina, NC: APDG Publishing, 2000.

Harte, L., Levine, R., and Kikta, R.: «3G Wireless Demystified», New York: McGraw-Hill, 2002.

Hawley, G. T.: «Historical Perspectives on the U.S. Telephone System», IEEE Commun. Magazine, vol. 29, pp. 24–28, March 1991.

Hecht, J.: «Understanding Fiber Optics», Upper Saddle River, NJ: Prentice Hall, 2001.

Heegard, C., Coffey, J. T., Gummadi, S., Murphy, P. A., Provencio, R., Rossin, E. J., Schrum, S., and Shoemaker, M. B.: «High-Performance Wireless Ethernet», IEEE Commun. Magazine, vol. 39, pp. 64–73, Nov. 2001.

Held, G.: «The Complete Modem Reference», 2nd ed., New York: Wiley, 1994.

Hellman, M. E.: «A Cryptanalytic Time-Memory Tradeoff», IEEE Trans, on Information Theory, vol. IT-26, pp. 401–406, July 1980.

Hills, A.: «Large-Scale Wireless LAN Design», IEEE Commun. Magazine, vol. 39, pp. 98–104, Nov. 2001.

Holzmann, G.J.: «Design and Validation of Computer Protocols», Englewood Cliffs, NJ: Prentice Hall, 1991.

Hu, Y., and Li, V. O. K.: «Satellite-Based Internet Access», IEEE Commun. Magazine, vol. 39, pp. 155–162, March 2001.

Hu, Y.-C., and Johnson, D. B.: «Implicit Source Routes for On-Demand Ad Hoc Network Routing», Proc. ACM Int'l Symp. on Mobile Ad Hoc Networking & Computing, ACM, pp. 1–10, 2001.

Huang, V., and Zhuang, W.: «QoS-Oriented Access Control for 4G Mobile Multimedia CDMA Communications», IEEE Commun. Magazine, vol. 40, pp. 118–125, March 2002.

Huber, J. F., Weiler, D., and Brand, H.: «UMTS, the Mobile Multimedia Vision for IMT-2000: A Focus on Standardization», IEEE Commun. Magazine, vol. 38, pp. 129–136, Sept. 2000.

Hui, J.: «A Broadband Packet Switch for Multi-rate Services», Proc. Int'l Conf. on Com-mun., IEEE, pp. 782–788, 1987.

Huitema, C.: «Routing in the Internet, Englewood Cliffs», NJ: Prentice Hall, 1995.

Hull, S.: «Content Delivery Networks», Berkeley, CA: Osborne/McGraw-Hill, 2002.

Humblet, P. A., Ramaswami, R., and Sivarajan, K. N.: «An Efficient Communication Protocol for High-Speed Packet-Switched Multichannel Networks», Proc. SIGCOMM '92 Conf., ACM, pp. 2–13, 1992.

Hunter, D. K., and Andonovic, I.: «Approaches to Optical Internet Packet Switching», IEEE Commun. Magazine, vol. 38, pp. 116–122, Sept. 2000.

Huston, G.: «TCP in a Wireless World», IEEE Internet Computing, vol. 5, pp. 82–84, March-April, 2001.

Ibe, O. C.: «Essentials of ATM Networks and Services», Boston: Addison-Wesley, 1997.

Irmer, T.: «Shaping Future Telecommunications: The Challenge of Global Standardization», IEEE Commun. Magazine, vol. 32, pp. 20–28, Jan. 1994.

Izzo, P.: «Gigabit Networks», New York: Wiley, 2000.

Jacobson, V.: «Congestion Avoidance and Control», Proc. SIGCOMM '88 Conf., ACM, pp. 314–329, 1988.

Jain, R.: «Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey», Computer Networks and ISDN Systems, vol. 27, Nov. 1995.

Jain, R.: «FDDI Handbook — High-Speed Networking Using Fiber and Other Media», Boston: Addison-Wesley, 1994.

Jain, R.: «Congestion Control in Computer Networks: Issues and Trends», IEEE Network Magazine, vol. 4, pp. 24–30, May-June 1990.

Jakobsson, M., and Wetzel, S.: «Security Weaknesses in Bluetooth», Topics in Cryptology: CT-RSA 2001, Berlin: Springer-Verlag LNCS 2020, pp. 176–191, 2001.

Joel, A.: «Telecommunications and the IEEE Communications Society», IEEE Commun. Magazine, 50th Anniversary Issue, pp. 6–14 and 162–167, May 2002.

Johansson, P., Kazantzidis, M., Kapoor, R., and Gerla, M.: «Bluetooth: An Enabler for Personal Area Networking», IEEE Network Magazine, vol. 15, pp. 28–37, Sept.-Oct. 2001.

Johnson, D. B.: «Scalable Support for Transparent Mobile Host Internetworking», Wireless Networks, vol. 1, pp. 311–321, Oct. 1995.

Johnson, H. W.: «Fast Ethernet — Dawn of a New Network», Englewood Cliffs, NJ: Prentice Hall, 1996.

Johnson, N. F., and Jajoda, S.: «Exploring Steganography: Seeing the Unseen», Computer, vol. 31, pp. 26–34, Feb. 1998.

Kahn, D.: «Cryptology Goes Public», IEEE Commun. Magazine, vol. 18, pp. 19–28, March 1980.

Kahn, D.: «The Codebreakers», 2nd ed., New York: Macmillan, 1995.

Kamoun, F., and Kleinrock, L.: «Stochastic Performance Evaluation of Hierarchical Routing for Large Networks», Computer Networks, vol. 3, pp. 337–353, Nov. 1979.

Kapp, S.: «802.11: Leaving the Wire Behind», IEEE Internet Computing, vol. 6, pp. 82–85, Jan.-Feb. 2002.

Karn, P.: «MACA — A New Channel Access Protocol for Packet Radio», ARRL/CRRL Amateur Radio Ninth Computer Networking Conf., pp. 134–140, 1990.

Kartalopoulos, S.: «Introduction to DWDM Technology: Data in a Rainbow», New York, NY: IEEE Communications Society, 1999.

Kasera, S. K., Hjalmtysson, G., Towlsey, D. F., and Kurose, J. F.: «Scalable Reliable Multicast Using Multiple Multicast Channels», IEEE/ACM Trans, on Networking, vol. 8, pp. 294–310, 2000.

Katz, D., and Ford, P. S.: «TUBA: Replacing IP with CLNP», IEEE Network Magazine, vol. 7, pp. 38–47, May-June 1993.

Katzenbeisser, S., and Petitcolas, F. A. P.: «Information Hiding Techniques for Steganography and Digital Watermarking», London, Artech House, 2000.

Kaufman, C., Perlman, R., and Speciner, M.: «Network Security», 2nd ed., Engle-wood Cliffs, NJ: Prentice Hall, 2002.

Kellerer, W., Vogel, H.-J., and Steinberg, K.-E.: «A Communication Gateway for Infrastructure-Independent 4G Wireless Access», IEEE Commun. Magazine, vol. 40, pp. 126–131, March 2002.

Kerckhoff, A.: «La Cryptographic Militaire», J. des Sciences Militaires, vol. 9, pp. 5–38, Jan. 1883 and pp. 161–191, Feb. 1883.

Kim, J. B., Suda, T., and Yoshimura, M.: «International Standardization of B-ISDN», Computer Networks and ISDN Systems, vol. 27, pp. 5–27, Oct. 1994.

Kipnis, J.: «Beating the System: Abuses of the Standards Adoptions Process», IEEE Commun. Magazine, vol. 38, pp. 102–105, July 2000.

Kleinrock, L.: «On Some Principles of Nomadic Computing and Multi-Access Communications», IEEE Commun. Magazine, vol. 38, pp. 46–50, July 2000.

Kleinrock, L., and Tobagi, F.: «Random Access Techniques for Data Transmission over Packet-Switched Radio Channels», Proc. Nat. Computer Conf., pp. 187–201, 1975.

Krishnamurthy, B., and Rexford, J.: «Web Protocols and Practice», Boston: Addison-Wesley, 2001.

Kumar, V., Korpi, M., and Sengodan, S.: «IP Telephony with H.323», New York: Wiley, 2001.

Kurose, J. F., and Ross, K. W.: «Computer Networking: A Top-Down Approach Featuring the Internet», Boston: Addison-Wesley, 2001.

Kwok, T.: «A Vision for Residential Broadband Service: ATM to the Home», IEEE Network Magazine, vol. 9, pp. 14–28, Sept.-Oct. 1995.

Kyas, O., and Crawford, G.: «ATM Networks», Upper Saddle River, NJ: Prentice Hall, 2002.

Lam, C. K. M., and Tan, B. C. Y.: «The Internet Is Changing the Music Industry», Commun. of the ACM, vol. 44, pp. 62–66, Aug. 2001.

Lansford, J., Stephens, A, and Nevo, R.: «Wi-Fi (802.1 Ib) and Bluetooth: Enabling Coexistence», IEEE Network Magazine, vol. 15, pp. 20–27, Sept.-Oct. 2001.

Lash, D. A.: «The Web Wizard's Guide to Perl and CGI», Boston: Addison-Wesley, 2002.

Laubach, M. E., Farber, D. J., and Dukes, S. D.: «Delivering Internet Connections over Cable», New York: Wiley, 2001.

Lee, J. S., and Miller, L. E.: «CDMA Systems Engineering Handbook», London: Artech House, 1998.

Leeper, D. G.: «A Long-Term View of Short-Range Wireless», Computer, vol. 34, pp. 39–44, June 2001.

Leiner, B. M., Cole, R., Postel, J., and Mills, D.: «The DARPA Internet Protocol Suite», IEEE Commun. Magazine, vol. 23, pp. 29–34, March 1985.

Levine, D. A., and Akyildiz, I. A.: «PROTON: A Media Access Control Protocol for Optical Networks with Star Topology», IEEE/ACM Trans, on Networking, vol. 3, pp. 158–168, April 1995.

Levy, S.: «Crypto Rebels», Wired, pp. 54–61, May-June 1993.

Li, J., Blake, C., De Couto, D. S. J., Lee, H. I., and Morris, R.: «Capacity of Ad Hoc Wireless Networks», Proc. 7th Int'l Conf. on Mobile Computing and Networking, ACM, pp. 61–69, 2001.

Lin, F., Chu, P., and Liu, M.: «Protocol Verification Using Reachability Analysis: The State Space Explosion Problem and Relief Strategies», Proc. SIGCOMM '87 Conf., ACM, pp. 126–135, 1987.

Lin, Y.-D., Hsu, N.-B., and Hwang, R.-H.: «QoS Routing Granularity in MPLS Networks», IEEE Commun. Magazine, vol. 40, pp. 58–65, June 2002.

Listani, M., Eramo, V., and Sabella, R.: «Architectural and Technological Issues for Future Optical Internet Networks», IEEE Commun. Magazine, vol. 38, pp. 82–92, Sept. 2000.

Liu, C. L., and Layland, J. W.: «Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment», Journal of the ACM, vol. 20, pp. 46–61, Jan. 1973.

Metcalfe, R. M., and Boggs, D. R.: «Ethernet: Distributed Packet Switching for Local Computer Networks», Commun. of the ACM, vol. 19, pp. 395–404, July 1976.

Metz, C.: «Interconnecting ISP Networks», IEEE Internet Computing, vol. 5, pp. 74–80, March-April 2001.

Metz, C.: «Differentiated Services», IEEE Multimedia Magazine, vol. 7, pp. 84–90, July-Sept. 2000.

Metz, C.: «IP Routers: New Tool for Gigabit Networking», IEEE Internet Computing, vol. 2, pp. 14–18, Nov.-Dec. 1998.

Miller, B. A., and Bisdikian, C.: «Bluetooth Revealed», Upper Saddle River, NJ: Prentice Hall, 2001.

Miller, P., and Cummins, M.: «LAN Technologies Explained», Woburn, MA: Butterworth-Heinemann, 2000.

Minoli, D.: «Video Dialtone Technology», New York: McGraw-Hill, 1995.

Minoli, D., and Vitella, M.: «ATM & Cell Relay for Corporate Environments», New York: McGraw-Hill, 1994.

Mishra, P. P., and Kanakia, H.: «A Hop by Hop Rate-Based Congestion Control Scheme», Proc. SIGCOMM '92 Conf., ACM, pp. 112–123, 1992.

Misra, A., Das, S., Dutta, A., McAuley, A., and DAS, S.: «IDMP-Based Fast Hand-offs and Paging in IP-Based 4G Mobile Networks», IEEE Commun. Magazine, vol. 40, pp. 138–145, March 2002.

Mogul, J. C.: «IP Network Performance», in Internet System Handbook, Lynch, D.C. and Rose, M.T. (eds.), Boston: Addison-Wesley, pp. 575–675, 1993.

Mok, A. K., and Ward, S. A.: «Distributed Broadcast Channel Access», Computer Networks, vol. 3, pp. 327–335, Nov. 1979.

Moy, J.: «Multicast Routing Extensions», Commun. of the ACM, vol. 37, pp. 61–66, Aug. 1994.

Mullins, J.: «Making Unbreakable Code», IEEE Spectrum, pp. 40–45, May 2002.

Nagle, J.: «On Packet Switches with Infinite Storage», IEEE Trans, on Commun., vol. COM-35, pp. 435–438, April 1987. '

Nagle, J.: «Congestion Control in TCP/IP Internetworks», Computer Commun. Rev., vol. 14, pp. 11–17, Oct. 1984.

Narayanaswami, C., Kamijoh, N., Raghunath, M., Inoue, T., Cipolla, T., Sanford, J., Schlig, E., Ventkiteswaran, S., Guniguntala, D., Kulkarni, V., and Yamazaki, K.: «IBM's Linux Watch: The Challenge of Miniaturization», Computer, vol. 35, pp. 33–41, Jan. 2002.

Naughton, J.: «A Brief History of the Future», Woodstock, NY: Overlook Press, 2000.

Needham, R. M., and Schroeder, M. D.: «Authentication Revisited», Operating Systems Rev., vol. 21, p. 7, Jan. 1987.

Needham, R. M., and Schroeder, M. D.: «Using Encryption for Authentication in Large Networks of Computers», Commun. of the ACM, vol. 21, pp. 993–999, Dec. 1978.

Nelakuditi, S., and Zhang, Z.-L.: «A Localized Adaptive Proportioning Approach to QoS Routing», IEEE Commun. Magazine vol. 40, pp. 66–71, June 2002.

Nemeth, E., Snyder, G., Seebass, S., and Hein, T. R.: «UNIX System Administration Handbook», 3rd ed., Englewood Cliffs, NJ: Prentice Hall, 2000.

Nichols, R. K., and Lekkas, P. C.: «Wireless Security», New York: McGraw-Hill, 2002.

Nist: «Secure Hash Algorithm», U.S. Government Federal Information Processing Standardise, 1993.

O'Hara, B., and Petrick, A.: «802.11 Handbook: A Designer's Companion», New York: IEEE Press, 1999.

Otway, D., and Rees, O.: «Efficient and Timely Mutual Authentication», Operating Systems Rev., pp. 8–10, Jan. 1987.

Ovadia, S.: «Broadband Cable TV Access Networks: from Technologies to Applications», Upper Saddle River, NJ: Prentice Hall, 2001.

Palais, J. C.: «Fiber Optic Commun.», 3rd ed., Englewood Cliffs, NJ: Prentice Hall, 1992.

Pan, D.: «A Tutorial on MPEG/Audio Compression», IEEE Multimedia Magazine, vol. 2, pp.60–74, Summer 1995.

Pandya, R.: «Emerging Mobile and Personal Communication Systems», IEEE Commun. Magazine, vol. 33, pp. 44–52, June 1995.

Parameswaran, M., Susarla, A., and Whinston, A. B.: «P2P Networking: An Information-Sharing Alternative», Computer, vol. 34, pp. 31–38, July 2001.

Park, J. S., and Sandhu, R.: «Secure Cookies on the Web», IEEE Internet Computing, vol. 4, pp. 36–44, July-Aug. 2000.

Partridge, C., Hughes, J., and Stone, J.: «Performance of Checksums and CRCs over Real Data», Proc. SIGCOMM '95 Conf., ACM, pp. 68–76, 1995.

Paxson, V.: «Growth Trends in Wide-Area TCP Connections», IEEE Network Magazine, vol. 8, pp. 8–17, July-Aug. 1994.

Paxson, V., and Floyd, S.: «Wide-Area Traffic: The Failure of Poisson Modeling», Proc. SIGCOMM '94 Conf., ACM, pp. 257–268, 1995.

Pepelnjak, I., and Guichard, J.: «MPLS and VPN Architectures», Indianapolis, IN: Cisco Press, 2001.

Perkins, C. E.: «RTP: Audio and Video for the Internet», Boston: Addison-Wesley, 2002. *Perkins, C. E. (ed.)*: «Ad Hoc Networking», Boston: Addison-Wesley, 2001.

Perkins, C. E.: «Mobile IP Design Principles and Practices», Upper Saddle River, NJ: Prentice Hall, 1998a.

Perkins, C. E.: «Mobile Networking in the Internet», Mobile Networks and Applications, vol. 3, pp. 319–334, 1998b.

Perkins, C. E.: «Mobile Networking through Mobile IP», IEEE Internet Computing, vol. 2, pp. 58–69, Jan.-Feb. 1998c.

Perkins, C. E., and Royer, E.: «The Ad Hoc On-Demand Distance-Vector Protocol», in Ad Hoc Networking, edited by C. Perkins, Boston: Addison-Wesley, 2001.

Perkins, C. E., and Royer, E.: «Ad-hoc On-Demand Distance Vector Routing», Proc. Second Ann. IEEE Workshop on Mobile Computing Systems and Applications, IEEE, pp. 90–100, 1999.

Perlman, R.: «Interconnections», 2nd ed., Boston: Addison-Wesley, 2000.

Perlman, R.: «Network Layer Protocols with Byzantine Robustness», Ph.D. thesis, M.I.T., 1988.

Perlman, R., and Kaufman, C.: «Key Exchange in IPsec», IEEE Internet Computing, vol. 4, pp. 50–56, Nov.-Dec. 2000.

Peterson, L. L., and Da Vie, B. S.: «Computer Networks: A Systems Approach», San Francisco: Morgan Kaufmann, 2000.

Peterson, W. W., and Brown, D. T.: «Cyclic Codes for Error Detection», Proc. IRE, vol. 49, pp. 228–235, Jan. 1961.

Pickholtz, R. L., Schilling, D. L., and Milstein, L. B.: «Theory of Spread Spectrum Communication — A Tutorial», IEEE Trans, on Commun., vol. COM-30, pp. 855–884, May 1982.

Pierre, G., Kuz, I., van Steen, M., Tanenbaum, A. S.: «Differentiated Strategies for Replicating Web Documents», Computer Commun., vol. 24, pp. 232–240, Feb. 2001.

Pierre, G., van Steen, M., and Tanenbaum, A. S.: «Dynamically Selecting Optimal Distribution Strategies for Web Documents», IEEE Trans, on Computers, vol. 51, June 2002.

Piscitello, D. M., and Chapin, A. L.: «Open Systems Networking: TCP/IP and OSI», Boston: Addison-Wesley, 1993.

Pitt, D. A.: «Bridging — The Double Standard», IEEE Network Magazine, vol. 2, pp. 94–95, Jan. 1988.

Piva, A., Bartolini, F., and Barni, M.: «Managing Copyrights in Open Networks», IEEE Internet Computing, vol. 6, pp. 18–26, May-June 2002.

Pohlmann, N.: «Firewall Systems», Bonn, Germany: MITP-Verlag, 2001.

Puzmanova, R.: «Routing and Switching: Time of Convergence?», London: Addison-Wesley, 2002.

Rabinovich, M., and Spatscheck, O.: «Web Caching and Replication», Boston: Addison-Wesley, 2002.

Raju, J., and Garcia-Luna-Aceves, J. J.: «Scenario-based Comparison of Source-Tracing and Dynamic Source Routing Protocols for Ad-Hoc Networks», ACM Computer Communications Review, vol. 31, October 2001.

Ramanathan, R., and Redi, J.: «A Brief Overview of Ad Hoc Networks: Challenges and Directions», IEEE Communf Magazine, 50th Anniversary Issue, pp. 20–22, May 2002.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S.: «A Scalable Content-Addressable Network», Proc. SIGCOMM '01 Conf., ACM, pp. 161–172, 2001.

Rivest, R. L.: «The MD5 Message-Digest Algorithm», RFC 1320, April 1992.

Rivest, R. L., and Shamir, A.: «How to Expose an Eavesdropper», Commun. of the ACM, vol. 27, pp. 393–395, April 1984.

Rivest, R. L., Shamir, A., and Adleman, L.: «On a Method for Obtaining Digital Signatures and Public Key Cryptosystems», Commun. of the ACM, vol. 21, pp. 120–126, Feb. 1978.

Roberts, L. G.: «Dynamic Allocation of Satellite Capacity through Packet Reservation», Proc. NCC, AFIPS, pp. 711–716, 1973.

Roberts, L. G.: «Extensions of Packet Communication Technology to a Hand Held Personal Terminal», Proc. Spring Joint Computer Conference, AFIPS, pp. 295–298, 1972.

Roberts, L. G.: «Multiple Computer Networks and Intercomputer Communication», Proc. First Symp. on Operating Systems Prin., ACM, 1967.

Rose, M. T.: «The Simple Book», Englewood Cliffs, NJ: Prentice Hall, 1994.

Rose, M. T.: «The Internet Message», Englewood Cliffs, NJ: Prentice Hall, 1993.

Rose, M. T., and McCloghrie, K.: «How to Manage Your Network Using SNMP», Englewood Cliffs, NJ: Prentice Hall, 1995.

Rowstron, A., and Druschel, P.: «Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility», Proc. 18th Symp. on Operating Systems Prin., ACM, pp. 188–201, 2001a.

Rowstron, A., and Druschel, P.: «Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Storage Utility», Proc. 18th Int'l Conf. on Distributed Systems Platforms, ACM/1FIP, 2001b.

Royer, E. M., and Toh, C.-K.: «A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks», IEEE Personal Commun. Magazine, vol. 6, pp. 46–55, April 1999.

Ruiz-Sanchez, M. A., Biersack, E. W., and Dabbous, W.: «Survey and Taxonomy of IP Address Lookup Algorithms», IEEE Network Magazine, vol. 15, pp. 8–23, March-April 2001.

Sairam, K. V. S. S. S. S., Gunasekaran, N., and Reddy, S. R.: «Bluetooth in Wireless Communication», IEEE Commun. Mag., vol. 40, pp. 90–96, June 2002.

Saltzer, J. H., Reed, D. P., and Clark, D. D.: «End-to-End Arguments in System Design», ACM Trans, on Computer Systems, vol. 2, pp. 277–288, Nov. 1984.

Sanderson, D. W., and Dougherty, D.: «Smileys», Sebastopol, CA: O'Reilly, 1993.

Sari, H., Vanhaverbeke, F., and Moeneclaey, M.: «Extending the Capacity of Multiple Access Channels», IEEE Commun. Magazine, vol. 38, pp. 74–82, Jan. 2000.

Sarikaya, B.: «Packet Mode in Wireless Networks: Overview of Transition to Third Generation», IEEE Commun. Magazine, vol. 38, pp. 164–172, Sept. 2000.

Schneier, B.: «Secrets and Lies», New York: Wiley, 2000.

Schneier, B.: «Applied Cryptography», 2nd ed., New York: Wiley, 1996.

Schneier, B.: «E-Mail Security», New York: Wiley, 1995.

Schneier, B.: «Description of a New Variable-Length Key, 64-Bit Block Cipher [Blowfish]», Proc. of the Cambridge Security Workshop, Berlin: Springer-Verlag LNCS809, pp. 191–204, 1994.

Schnorr, C. P.: «Efficient Signature Generation for Smart Cards», Journal of Cryptology, vol. 4, pp. 161–174, 1991.

Scholtz, R. A.: «The Origins of Spread-Spectrum Communications», IEEE Trans, on Commun., vol. COM-30, pp. 822–854, May 1982.

Scott, R.: «Wide Open Encryption Design Offers Flexible Implementations», Cryptologia, vol. 9, pp. 75–90, Jan. 1985.

Seifert, R.: «The Switch Book», Boston: Addison-Wesley, 2000.

Seifert, R.: «Gigabit Ethernet», Boston: Addison-Wesley, 1998.

Senn, J. A.: «The Emergence of M-Commerce», Computer, vol. 33, pp. 148–150, Dec. 2000.

Serjantov, A.: «Anonymizing Censorship'Resistant Systems», Proc. First Int'l Workshop on Peer-to-Peer Systems, Berlin: Springer-Verlag LNCS, 2002.

Severance, C.: «IEEE 802.11: Wireless Is Coming Home», Computer, vol. 32, pp. 126–127, Nov. 1999.

Shahabi, C., Zimmermann, R., Fu, K., and Yao, S.-Y. D.: «YIMA: A Second-Generation Continuous Media Server», Computer, vol. 35, pp. 56–64, June 2002.

Shannon, C.: «A Mathematical Theory of Communication», Bell System Tech. J., vol. 27, pp. 379–423, July 1948; and pp. 623–656, Oct. 1948.

Shepard, S.: «SONET/SDH Demystified», New York: McGraw-Hill, 2001.

Shreedhar, M., and Varghese, G.: «Efficient Fair Queueing Using Deficit Round Robin», Proc. SIGCOMM '95 Conf., ACM, pp. 231–243, 1995.

Skoudis, E.: «Counter Hack, Upper Saddle River», NJ: Prentice Hall, 2002.

Smith, O. K., and Alexander, R. C.: «Fumbling the Future», New York: William Morrow, 1988.

Smith, R. W.: «Broadband Internet Connections», Boston: Addison Wesley, 2002.

Snoeren, A. C., and Balakrishnan, H.: «An End-to-End Approach to Host Mobility», Intel Conf. on Mobile Computing and Networking, ACM, pp. 155–166, 2000.

Sobel, D. L.: «Will Carnivore Devour Online Privacy», Computer, vol. 34, pp. 87–88, May 2001.

Solomon, J. D.: «Mobile IP: The Internet Unplugged», Upper Saddle River, NJ: Prentice Hall, 1998.

Spohn, M., and Garcia-Luna-Aceves, J. J.: «Neighborhood Aware Source Routing», Proc. ACMMobiHoc 2001, ACM, p. 2001.

Spurgeon, C. E.: «Ethernet: The Definitive Guide», Sebastopol, CA: O'Reilly, 2000.

Stallings, W.: «Data and Computer Communications», 6th ed., Upper Saddle River, NJ: Prentice Hall, 2000.

Steinmetz, R., and Nahrstedt, K.: «Multimedia Fundamentals». Vol. 1: «Media Coding and Content Processing», Upper Saddle River, NJ: Prentice Hall, 2002.

Steinmetz, R., and Nahrstedt, K.: «Multimedia Fundamentals». Vol. 2: «Media Processing and Communications», Upper Saddle River, NJ: Prentice Hall, 2003a.

Steinmetz, R., and Nahrstedt, K.: «Multimedia Fundamentals». Vol. 3: «Documents, Security, and Applications», Upper Saddle River, NJ: Prentice Hall, 2003b.

Steiner, J. G., Neuman, B. C., and Schiller, J. I.: «Kerberos: An Authentication Service for Open Network Systems», Proc. Winter USENIX Conf., USENIX, pp. 191–201, 1988.

Stevens, W. R.: «UNIX Network Programming», Volume 1: «Networking APIs — Sockets and XTI», Upper Saddle River, NJ: Prentice Hall, 1997.

Stevens, W. R.: «TCP/IP Illustrated», Boston: Addison-Wesley, 1994.

Stewart, R., and Metz, C.: «SCTP: New Transport Protocol for TCP/IP», IEEE Internet Computing, vol. 5, pp. 64–69, Nov.-Dec. 2001.

Stinson, D. R.: «Cryptography Theory and Practice», 2nd ed., Boca Raton, FL: CRC Press, 2002.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H.: «Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications», Proc. SIGCOMM '01 Conf., ACM, pp. 149–160, 2001.

Striegel, A., and Manimaran, G.: «A Survey of QoS Multicasting Issues», IEEE Commun. Mag., vol. 40, pp. 82–87, June 2002.

Stubblefield, A., Ioannidis, J., and Rubin, A. D.: «Using the Fluhrer, Mantin, and Shamir Attack to Break WEP», Proc Network and Distributed Systems Security Symp., SOC, pp. 1–11, 2002.

Summers, C. K.: «ADSL: Standards, Implementation, and Architecture», Boca Raton, FL: CRC Press, 1999.

Sunshine, C. A., and Dalal, Y. K.: «Connection Management in Transport Protocols», Computer Networks, vol. 2, pp. 454–473, 1978.

Tanenbaum, A. S.: «Modern Operating Systems», Upper Saddle River, NJ: Prentice Hall, 2001.

Tanenbaum, A. S., and van Steen, M.: «Distributed Systems: Principles and Paradigms», Upper Saddle River, NJ: Prentice Hall, 2002.

Teger, S., and Waks, D. J.: «End-User Perspectives on Home Networking», IEEE Commun. Magazine, vol. 40, pp. 114–119, April 2002.

Thyagarajan, A. S., and Deering, S. E.: «Hierarchical Distance-Vector Multicast Routing for the MBone», Proc. SIGCOMM '95 Conf., ACM, pp. 60–66, 1995.

Tittel, E., Valentine, C., Burmeister, M., and Dykes, L.: «Mastering XHTML», Alameda, CA: Sybex, 2001.

Tokoro, M., and Tamaru, K.: «Acknowledging Ethernet», Compcon, IEEE, pp. 320–325, Fall 1977.

Tomlinson, R. S.: «Selecting Sequence Numbers», Proc. SIGCOMM/SIGOPS Interprocess Commun. Workshop, ACM, pp. 11–23, 1975.

Tseng, Y.-C., Wu, S.-L., Liao, W.-H., and Chad, C.-M.: «Location Awareness in Ad Hoc Wireless Mobile Networks», Computer, vol. 34, pp. 46–51, 2001.

Tuchman, W.: «Hellman Presents No Shortcut Solutions to DES», IEEE Spectrum, vol. 16, pp. 40–41, July 1979.

Turner, J. S.: «New Directions in Communications (or Which Way to the Information Age)», IEEE Commun. Magazine, vol. 24, pp. 8–15, Oct. 1986.

Vacca, J. R.: «I-Mode Crash Course», New York: McGraw-Hill, 2002.

Valade, J.: «PHP & MySQL for Dummies», New York: Hungry Minds, 2002.

Varghese, G., and Lauck, T.: «Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility», Proc. 11th Symp. on Operating Systems Prin., ACM, pp. 25–38, 1987.

Varshney, U., Snow, A., McGivern, M., and Howard, C.: «Voice over IP», Commun. of the ACM, vol. 45, pp. 89–96, 2002.

Varshney, U., and Vetter, R.: «Emerging Mobile and Wireless Networks», Commun. of the ACM, vol. 43, pp. 73–81, June 2000.

Vetter, P., Goderis, D., Verpooten, L., and Granger, A.: «Systems Aspects of APON/VDSL Deployment», IEEE Commun. Magazine, vol. 38, pp. 66–72, May 2000.

Waddington, D. G., and Chang, F.: «Realizing the Transition to IPv6», IEEE Commun. Mag., vol. 40, pp. 138–148, July 2002.

Waldman, M., Rubin, A. D., and Cranor, L. F.: «Publius: A Robust, Tamper-Evident, Censorship-Resistant, Web Publishing System», Proc. Ninth USENIX Security Symp., USENIX, pp. 59–72, 2000.

Wang, Y., and Chen, W.: «Supporting IP Multicast for Mobile Hosts», Mobile Networks and Applications, vol. 6, pp. 57–66, Jan.-Feb. 2001.

Wang, Z.: «Internet QoS», San Francisco: Morgan Kaufmann, 2001.

Warneke, B., Last, M., Liebowitz, B., and Pister, K. S. J.: «Smart Dust: Communicating with a Cubic Millimeter Computer», Computer, vol. 34, pp. 44–51, Jan. 2001.

Wayner, P.: «Disappearing Cryptography: Information Hiding, Steganography, and Watermarking», 2nd ed., San Francisco: Morgan Kaufmann, 2002.

Webb, W.: «Broadband Fixed Wireless Access as a Key Component of the Future Integrated Communications Environment», IEEE Commun. Magazine, vol. 39, pp. 115–121, Sept. 2001.

Weiser, M.: «Whatever Happened to the Next Generation Internet?», Commun. of the ACM, vol. 44, pp. 61–68, Sept. 2001.

Weltman, R., and Dahbura, T.: «LDAP Programming with Java», Boston: Addison-Wesley, 2000.

Wessels, D.: «Web Caching», Sebastopol, CA: O'Reilly, 2001.

Wetteroth, D.: «OSI Reference Model for Telecommunications», New York: McGraw-Hill, 2001.

Wiljakka, J.: «Transition to IPv6 in GPRS and WCDMA Mobile Networks», IEEE Commun. Magazine, vol. 40, pp. 134–140, April 2002.

Williamson, H.: «XML: The Complete Reference», New York: McGraw-Hill, 2001.

Willinger, W., Taqqu, M. S., Sherman, R., and Wilson, D. V.: «Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level», Proc. SIGCOMM '95 Conf., ACM, pp. 100–113, 1995.

Wright, D.J.: «Voice over Packet Networks», New York: Wiley, 2001.

Wylie, J., Bigrigg, M. W., Strunk, J. D., Ganger, G. R., Kiliccote, H., and Khosla, P. K.: «Survivable Information Storage Systems», Computer, vol. 33, pp. 61–68, Aug. 2000.

Xylomenos, G., Polyzos, G. C., Mahonen, P., and Saaranen, M.: «TCP Performance Issues over Wireless Links», IEEE Commun. Magazine, vol. 39, pp. 52–58, April 2001.

Yang, C.-Q., and Reddy, A. V. S.: «A Taxonomy for Congestion Control Algorithms in Packet Switching Networks», IEEE Network Magazine, vol. 9, pp. 34–45, July-Aug. 1995.

Yuval, G.: «How to Swindle Rabin», Cryptologia, vol. 3, pp. 187–190, July 1979.

Zacks, M.: «Antiterrorist Legislation Expands Electronic Snooping», IEEE Internet Computing, vol. 5, pp. 8–9, Nov.-Dec. 2001.

Zadeh, A. N., Jabbari, B., Pickholtz, R., and Vojcic, B.: «Self-Organizing Packet Radio Ad Hoc Networks with Overlay (SOPRANO)», IEEE Commun. Mag., vol. 40, pp. 149–157, June 2002.

Zhang, L.: «Comparison of Two Bridge Routing Approaches», IEEE Network Magazine, vol. 2, pp. 44–48, Jan.-Feb. 1988.

Zhang, L.: «RSVP: A New Resource ReSerVation Protocol», IEEE Network Magazine, vol. 7, pp. 8–18, Sept.-Oct. 1993.

Zhang, Y., and Ryu, B.: «Mobile and Multicast IP Services in PACS: System Architecture, Prototype, and Performance», Mobile Networks and Applications, vol. 6, pp. 81–94, Jan.-Feb. 2001.

Zimmermann, P. R.: «The Official PGP User's Guide», Cambridge, MA: M.I.T. Press, 1995a.

Zimmermann, P. R.: «PGP: Source Code and Internals», Cambridge, MA: M.I.T. Press, 1995b.

Zipf, G. K.: «Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology», Boston: Addison-Wesley, 1949.

Ziv, J., and Lempel, Z.: «A Universal Algorithm for Sequential Data Compression», IEEE Trans, on Information Theory, vol. IT-23, pp. 337–343, May 1977.

Алфавитный указатель

!

10Base2, 318
10Base5, 317
10Base-F, 320
10Base-T, 319
3G, мобильная связь третьего поколения, 204
64-символьная кодировка, 679
802.16, стандарт, 171
802.3u, 331
802.3z, 334

A

AAL-SAP, точка доступа в сетях ATM, 564
AAL-уровень, ATM, 90
ACM, 27
ActiveX, управляющий элемент, 921
ADCCP, 276
ADSL, 163, 801
AES, 837
ALOHA, 296
  дискретная, 298
  чистая, 296
ALOHANET, 93
American National Standards Institute, 102
AMPS, сотовая телефония, 189
ANSI, 102
ANSNET, 81
AODV, алгоритм маршрутизации, 434
ARPA, 76
ARPANET, 66, 75, 104, 414, 648, 659, 669
ARP-прокси, 518
ASCII armor, 679
ASP, 730
ATM, 88
  виртуальный канал, 89
    постоянный, 89
  плоскость пользователя, 91
  плоскость управления, 91
  эталонная модель, 90
    ATM-уровень, 90
    SAR-подуровень, 92
    TC-подуровень, 92
    подуровень
      PMD, 92
      конвергенции, 91
      конвергенции передачи, TC, 92
    подуровень конвергенции, CS, 92
    уровень адаптации (AAL), 90
    физический уровень, 90
  ячейка, 89
ATM-уровень, 90
Authenticode, 922

B

base64, 679
BB84, 826
BBN, 77
Bell Operating Company, 153
Bell System, 151
Bluetooth, 44, 362
  архитектура, 362
  пикосеть, 362
  профиль, 364
  рассеянная сеть, 362
  соединение, 368
    ACL, 368

Bluetooth (*продолжение*)
SCO, 368
стек протоколов, 365
уровень радиосвязи, 367
BNC-коннектор, 318
BOOTP, 519
broadcast network, 37
B-кадр, 794

C

Carnivore, система электронной разведки ФБР, 36
CAS, 176
CCITT, 100, 173
CCS, сигнализация по общему каналу, 176
CD, 103
CDMA, сотовая телефония, 198
CDMA2000, 205
CDN, сети доставки содержимого, 745
cell, 43
CGI, 727
cHTML, 754
CLEC, 168
Committee Draft, 103
common carrier, 99
cookie-файл, 709
неустойчивый, 710
устойчивый, 710
CRC-код, 237
CSMA/CA, протокол, 345
CS-подуровень, ATM, 92

D

DCMA, 932
DDoS, 879
DEC, 26
DES, 834
побелка, 835
тройное шифрование, 836
DHTML, 730
distributed system, 22
DIX, стандарт, 94
DMT, дискретная мультитональная модуляция, 165
DNS, 68, 79, 516, 533, 658
обратный поиск, 664
DNSsec, 912
DOCSIS, 211
Domain Name Service, 68
Domain Name System, 79
DoS, 879
DS1, 175
DSL, цифровые абонентские линии, 163
DSLAM, мультиплексор доступа к DSL, 167
DSS, 856

E

E1, 176
EDGE, 205
e-mail, 668
ESMTP, 686
Ethernet, 39
ответвитель
зуб вампира, 318
BNC-коннектор, 318
история создания, 92
кабель
10Base2, 318
10Base5, 317
10Base-T, 319
коммутатор, 329
моноканал, 94
производительность, 326
пространство столкновений, 330

F

FCC, 138
FDD, дуплексная связь с частотным разделением, 357
FDDI, 330
FDM, 171
базовая группа, 173
главная группа, 173
супергруппа, 173
Frame Relay, 88
FTP, 514
FTTC, 801
FTTH, 801

G

G.992.2 (G.lite), ADSL стандарт, 168
Gates, Bill, 147
Globalstar, 147
Gopher, 707
GPS, 145

H

H.245, 775
H.323, 774
  зона, 775
  терминал, 775
  хранитель шлюза, 775
  шлюз, 775
HDLC, 276
HDTV, 784
HFC, 801
Honeywell DDP-316, 77
HR-DSSS, метод, 344
HTML, 712
  cHTML, 754
  XHTML, 726
  атрибут, 713
  директива, 713
  таблица, 717
    строка, 717
    ячейка, 717
  таблица стилей, 718
  тег, 713
  форма, 719
HTTP, 66, 735
  заголовок запроса, 738
  заголовок ответа, 738
  метод, 736
  устойчивое соединение, 736
HTTPS, 917

I

IAB, 104
IBM, 73, 81, 99, 121, 276, 834–836
IBM PC-RT, 81
IEEE, 27, 103
IEEE 802.11, стандарт беспроводных сетей, 45
IEEE 802.2
  протокол LLC, 339
IEEE 802.3
  коммутируемая сеть, 329
IETF, 105
IETF, группа проектирования Интернета, 105
ILEC, 168
IMAP, 690
i-mode
  бизнес-модель, 752
  программная структура, 753
IMP, ARPANET, 77
IMT-2000, 204
IMTS, мобильная телефонная связь, 189
inetd, интернет-демон, 608
Institute of Electrical and Electronics Engineers, 103
Interface Message Processor, 77
International Organization for Standardization, 102
International Standard, 103
Internet Engineering Task Force Internet, 105
Internet Protocol, 67
Internet Research Task Force, 105
Internet Society, 105
IP, 497
IPsec, 872
  HMAC, 875
  ISAKMP, 873
  заголовок ESP, 875
  заголовок идентификации, 873
  защищающая связь, 872
  режим туннелирования, 873
  транспортный режим, 873
IPv4, 498
IPv6, 532
  джамбограмма, 538
  дополнительный заголовок, 537
  основной заголовок, 534
  полемика, 539
IP-адрес, 501
IP-протокол, 67
Iridium, 145
IRTF, 105
IS, 103
ISM, 137
ISO, 62, 102
ITU, 787
ITU-D, 100
ITU-R, 100, 136
ITU-T, 100
IXC, 154
I-кадр, 792

J

JPEG, 787
JSP, 730
JVM, 734

K

Kerberos, 897

L

LATA, 154
LDAP, 668
LEC, 154
LMDS, 169
LSI-11, 80
LTP, 753

M

MACAW, 316
mailto, 707
MBone, 803
McCaw, Craig, 147
MD5, 858
metropolitan area network, 40
MIME, 677, 678
MMDS, 169
Mosaic, 694
MOSPF, 806
Motorola, 145
MP3, 764
MPEG
  B-кадр, 794
  I-кадр, 792
  MPEG-1, 791
  MPEG-2, 794
  P-кадр, 793
  макроблок, 793
MPLS, 479
  класс эквивалентности пересылок, 480
  создание записей таблицы, 480
  цветной поток, 480
MSC, 190
MTSO, 190
MTU, 610

N

NAK, 260
NAP, пункт доступа к сети, 81
NAT
  NAT-блок, 511
  недостатки, 513
Network Access Point, 81
NIST, 103
NPL, Национальная физическая лаборатория Англии, 77
NREN, 81
NSAP, 564
NSF, Национальный научный фонд США, 80
NSFNET, 80
NTSC, 784
NTT DoCoMo, 750

O

OC, оптический носитель, 181
OFDM, метод, 344
ONU, 801
OSI, эталонная модель, 62

P

packet-switched, 43
PAL, 784
PCM, кодово-импульсная модуляция, 175
PEM, 906
PGP, 901
PHP, 728
Physical Medium Dependent sublayer, 92
piggybacking, 252
PIM, 806
plug-in, 698
PMD-подуровень, ATM, 92
POP, 154
POP3, 687
POTS, диапазон обычной телефонной сети, 165
PPP, 281
primitive, 58
P-кадр, 793

Q

quoted-printable, 679

R

RAID, 799
RFC, 105
RFC 1034, 659
RFC 1035, 659
RFC 1112, 529
RFC 1341, 678

RFC 1421, 906
RFC 1422, 906
RFC 1423, 906
RFC 1424, 906
RFC 1550, 533
RFC 1661, 281, 284
RFC 1662, 281
RFC 1663, 281, 283
RFC 1700, 500
RFC 1939, 687
RFC 2045, 679
RFC 2045 - 2049, 678
RFC 2060, 690
RFC 2328, 521
RFC 3194, 536
RFC 821, 669, 675
RFC 822, 669–670, 675–678, 681, 735, 904, 906
RFC 826, 517
RFC 903, 519
Rijndael, 839
router, 42
RSA, 850
RTP, протокол реального времени, 603
RTSP, протокол, 768

S

S/MIME, 907
SAR-подуровень, ATM, 92
SDH, 179
SDLC-протокол, 276
SECAM, 784
Segmentation And Reassembly sublayer, 92
SHA, 859
SIP, 778
SMS, 751
SMTP, 683
SOAP, 726
SONET, синхронная оптическая сеть, 179
SSL, протокол защищенных сокетов, 916
store-and-forward, 43
STS-1, 180

T

T/TCP, 632
T1, 175
tariff, 99
TCM, решетчатое кодирование, 161
TCP, 610, 614, 629
  SCTP, 633
  алгоритм Джекобсона, 627
  алгоритм Карна, 629
  алгоритм Нагля, 621
  беспроводная сеть, 629
  борьба с перегрузкой, 623
  заголовок сегмента, 611
  модель службы, 608
  популярный порт, 608
  порт, 608
  порт источника, 512
  порт назначения, 512
  разрыв соединения, 616
  сегмент, 610
  синдром глупого окна, 621
  срочные данные, 609
  транзакционный, T/TCP, 632
  управление передачей, 619
  управление соединением, 614
  управление таймерами, 626
  установка соединения, 615
TCP/IP
  эталонная модель, 66
TC-подуровень, ATM, 92
TDD, дуплексная связь с временным разделением, 357
TDM, 171
Teledesic, 147
Telnet, 708
TLS, защита транспортного уровня, 920
TPDU-модуль, 555
trailer, 53
Transmission Control Protocol, 67
Transmission Convergence, 92
TSAP, точка доступа к службам транспортного уровня, 564

U

UDP, 598
UDP-протокол, 68
UHF, 131
unshielded twisted pair, 121
URL, 697, 705
URN, универсальное имя ресурса, 708
User Data Protocol, 68
UTP, неэкранированная витая пара, 121

V

V.32 bis, 161
V.34, 161
V.34 bis, 161
V.90, 163
VHF, 131
VHF диапазон, 134
VLF, 134
VLF диапазон, 134
VSAT, 143

W

W3C, консорциум WWW, 694
WAN, 42
WAP, 33, 748
  WDP, 749
  WML, 749
  стек протоколов, 749
W-CDMA, широкополосный CDMA, 204
WEP, протокол обеспечения конфиденциальности, 350
wide area network, 42
WiFi, стандарт 802.11, 96
World Wide Web, 22
WWW, 83, 693

X

X.25, 87
  формат пакетов, 87
X.400, 670
X.509, 866
Xerox, 93
XHTML, 726
XML, 722
XSL, 723

А

автоматический запрос повторной передачи, 250
автономная система, 491, 497
авторитетная запись, DNS, 666
авторское право, 931
агент передачи сообщений, 670, 672
Агентство национальной безопасности США, 836
адаптивная маршрутизация, 408
адаптивного дерева протокол, 309
адрес
  транспорт, 564
адресация, 55
активный повторитель, 127
алгоритм
  AODV, 434
  RC4, 882
  RSA, 850
  SHA, 859
  выбора маршрута, 406
  Джекобсона, 627
  дырявого ведра, 463
  затяжного пуска, 625
  Карна, 629
  кодирования длин серий, 790
  маркерного ведра, 465
  маршрутизации, 406, 44
  Нагля, 621
  противоточного обучения, 376
  с открытым ключом, 849, 852
  справедливого обслуживания, 471
    взвешенный, 472
алгоритм выделения частот
  аукцион, 137
  конкурс красоты, 136
  лотерея, 137
алгоритм маршрутизации, 403
альянс надежных вычислительных платформ, 933
амплитудная модуляция, 158
амплитудно-фазовая диаграмма, 161
анализ достижимости, 53, 271
анализ трафика, 873
анализ Фурье, 115
аналого-цифровой преобразователь, 762
апокалипсис двух слонов, 71
аппаратный криптопроцессор, 924
архитектура Интернета, 84
архитектура сети, 52
асимметричная цифровая абонентская линия, 801
атака шифра
  задача о днях рождения, 861
  повторное воспроизведение, 895
  пожарная цепочка, 893
  человек посередине, 893

аудио, цифровое, 761
аутентификация, 886, 887
  KDC-центр, 894
  протокол Kerberos, 897
  протокол Нидхэма–Шрёдера, 895
  протокол Отуэя–Риса, 897
  с открытым ключом, 900

Б

базовая группа, 173
Баркера, последовательность, 343
безопасность
  Authenticode, 922
  DNSsec, 912
  IPsec, 872
  S/MIME, 907
  SSL, 916
  TLS, 920
  WEP, 881
  X.509, 866
  анонимная рассылка, 925
  атака типа DDoS, 879
  атака типа DoS, 879
  в Bluetooth, 884
  в беспроводных сетях, 881
  во Всемирной паутине, 907
  защита соединений, 871
  защищенная файловая система, 914
  именования ресурсов, 909
  переносимая программа, 920
  песочница, 921
  самозаверяющийся URL, 915
  сертификат, 864
  управление открытыми ключами, 863
безопасность в сетях, 814
беспорядочный режим, 372
беспроводная локальная линия, WLL, 169
беспроводная локальная сеть
  протокол, 313
беспроводная сеть
  беспроводной протокол TCP, 629
  мобильные хосты, 529
  мобильный хост, 430
  электромагнитные волны, 130
беспроводные ЛВС, 341
  802.11a, 344
  802.11b, 344
  802.11g, 344
  DSSS, метод, 343
беспроводные ЛВС (*продолжение*)
  FHSS, метод, 343
  NAV, 347
  время пребывания, 343
  код Грея, 343
  код Уолша–Адамара, 344
  межкадровые интервалы
    DIFS, 349
    EIFS, 349
    PIFS, 349
    SIFS, 349
  метод HR-DSSS, 344
  метод OFDM, 344
  пачка фрагментов, 347
  последовательность Баркера, 343
  протокол CSMA/CA, 345
  режим DCF, 345
  режим PCF, 345
  структура кадра, 350
беспроводные региональные сети, 353
  иерархия протоколов, 355
беспроводные сети
  использование, 31
биграмма, 822
бит четности, 234
битовое заполнение, 229
блочный шифр, 832, 834
бод (baud), 159
борьба с перегрузкой, 444
  TCP, 623
  бит предупреждения, 452
  в дейтаграммных подсетях, 451
  в сетях виртуальных каналов
    управление допуском, 450
  дырявое ведро, 462
  нерегулярное раннее обнаружение, 456
  общие принципы, 446
  сброс нагрузки, 454
  сдерживающий пакет, 452
    для ретрансляционных участков, 453
борьба с флуктуациями, 456
брандмауэр, 876
  пакетный фильтр, 877
браузер, Всемирная паутина, 694
быстрая обработка TPDU-модулей, 644
быстрый Ethernet, 331
  100Base-4T, 332
  100Base-FX, 333
  100Base-T2, 333

быстрый Ethernet (*продолжение*)
100Base-TX, 333
4B/5B, 333
8B/6T, 333

В

веб-страница, 694
веб-фильтр, 927
вектор инициализации, 843
верификация протоколов, 270
видео, 782
аналоговое, 782
поступательное, 783
смешанный сигнал, 783
цифровое, 784
чересстрочная развертка, 783
видео по заказу, 795
видеосервер, 796
распределительная сеть, 800
видеосервер, 796
программное обеспечение, 798
виртуальная машина Java, 921
виртуальная частная сеть, 879
виртуальные ЛВС, 385
виртуальный канал, 402
сравнение с дейтаграммами, 405
виртуальный канал ATM, 89
витая пара, 120, 121
категория 3, 121
категория 5, 121
неэкранированная, 121
экранированная, 121
внезапная давка, 744
внешний агент, 431
внешний шлюзовый протокол, 491, 526
внутренний агент, 431
внутренний шлюзовый протокол, 491
водяной знак, 931
возврат на *n*, 258
вокодер, 194
волоконная оптика, 122
SONET, 179
многомодовое стекловолокно, 123
мода, 123
одномодовое стекловолокно, 123
основные принципы, 122
световая дисперсия, 125
спектральное уплотнение, 173
волоконно-оптическая сеть, 127
волоконно-оптический кабель, 125
восстановление после сбоев, 583
восходящее мультиплексирование, 582
Всемирная паутина, WWW, 83, 693
URL, 705
беспроводные технологии, 748
браузер, 694
вспомогательное приложение, 699
гиперссылка, 695
гипертекст, 694, 698
подключаемый модуль, 698
производительность, 742
внезапная давка, 744
протокол HTML, 735
входное дерево, 408
входное устройство, 41
выбор кратчайшего маршрута, 521
выбор кратчайшего пути, 409
выборочная заливка, 413
выборочный повтор, 259
вызов удаленной процедуры, RPC, 600
клиентская заглушка, 600
маршалинг, 601
серверная заглушка, 600
высокоуровневый протокол управления каналом, 276
выталкивающий сервер, 770
вычислительная сеть, 22

Г

гармоника, анализ Фурье, 115
гигабитная сеть, 648
гигабитный Ethernet, 334
8B/10B, 337
пакетная передача кадров, 336
расширение носителя, 336
гиперссылка, 695
гипертекст, 694, 698
главная группа, 173
глобальная сеть, 42
Грея, код, 343
группа исследования Интернета, 105
группа проектирования Интернета, IETF, 105
групповая рассылка, 322

Д

двоичный обратный отсчет, 305
двоичный экспоненциальный алгоритм отката, 326
двоичный экспоненциальный откат, 325
дежурный таймер, TCP, 629
дейтаграмма, 402
дейтаграммная сеть, 402
дейтаграммная служба, 57
  с подтверждениями, 57
дейтаграммный сервис
  сравнение с виртуальным каналом, 405
декодирование, 786
дельта-модуляция, 177
дерево с основанием в сердцевине, 430
десятичная нотация IP-адреса, 502
децибел, 119, 761
джиттер, 762
диагональный базис, 827
диапазон VHF, 134
диапазон обычной телефонной сети, POTS, 165
динамическая веб-страница, 727
  ASP, 730
  CGI, 727
  JSP, 730
  Perl, 728
  PHP, 728
  динамический HTML, 730
динамический HTML, 730
директива, HTML, 713
дисковая ферма, 799
дисковый массив, 799
дискретная ALOHA, 299
дискретная мультитональная модуляция, DMT, 165
дисперсия в оптическом волокне, 125
дифференциальная кодово-импульсная модуляция, 177
дифференциальный криптоанализ, 848
длина волны, 130
добровольное ARP-сообщение, 531
доктрина законного использования, 933
домен, 660
  верхнего уровня, 660
дуплексное соединение, 162
дырявое ведро, 462

З

заголовок, 53
  электронная почта, 672
заголовок запроса, 738
заголовок кадра, 242
заголовок ответа, 738
задача о днях рождения, 861
закон Ципфа, 797
заливка, 412
замирание вследствие многолучевого распространения, 97
запись ресурсов, 662
затяжного пуска алгоритм, 625
зашифрованный текст, 819
защита соединений
  IPsec, 872
защитная полоса, 172
защищенная файловая система, 914
зеркало, 744
зеркальная атака, 889
злоумышленник, 819
зона, DNS, 666

И

иерархическая маршрутизация, 424
иерархия протоколов, 50
иерархия хранения, 797
измененное окончательное судебное решение, 154
измерение отраженного сигнала, 318
Институт инженеров по электротехнике и электронике, 103
интегральное обслуживание, 472
интеллектуальная собственность, 931
интерактивная веб-страница, 731
  JavaScript, 731
интерактивная страница
  апплет, 734
Интернет
  IPv6, 532
  IP-протокол, 67
  MBone, 803
  TCP, 607
  архитектура, 84
  Всемирная паутина, 693
  история, 82
  маршрутизация, 521

Интернет (*продолжение*)
протокол внешнего шлюза, 521
протокол внутреннего шлюза, 521
межсетевой уровень, 496
многоадресная рассылка, 528
мобильный IP, 529
Общество Интернета, 105
подсеть, 503
порт, 564
протокол IP, 497, 498
управление соединением, 614
уровень передачи данных, 280
интернет-демон
inetd, 608
интернет-провайдер, 280
интернет-протоколы
ARP, 516
добровольное сообщение, 531
BGP, 526
DVMRP, 804
FTP, 707
HTTP, 706, 735
ICMP, 515
IGMP, 529, 805
IP, 67, 497, 498
OSPF, 521
PIM, 806
PPP, 281
RARP, 519
SMTP, 683
TCP, 67, 607, 610, 683
UDP, 68
интернет-радио, 771
интернет-телефония, 774
H.225, 776
H.245, 775
H.323, 774
Q.931, 776
RAS, канал, 776
SIP, 778
интернет-уровень, 66
интерсеть, 49, 481
интерфейс, 69
интерфейс межуровневый, 51
интерферометр
Маха–Цандера, 126, 312
Фабри–Перо, 126, 312
интрасеть, 85
информационный кадр, 277
инфракрасное излучение, 138
инфракрасные волны, 138
искажение сигнала, 157
исправление ошибок, 234

K

кабель, 171
кабельная система
оптоузел, 208
кабельное телевидение, 154, 207, 221
HFC, 208
абонентское телевидение, 207
распределительное устройство, 207
кабельный Интернет
CMTS, 211
кабельный модем, 211
DOCSIS, 211
измерение дальности, 212
мини-интервалы, 212
распределение спектра, 209
кабельный центр, 121
кадр, 223
видео, 782
данных, 64
заголовок, 242
подтверждения, 64
уровень передачи данных, 227
канал, 42
с множественным доступом, 292
с произвольным доступом, 292
T2, 178
T3, 178
канальный уровень, 222
каникулярный демон, 691
Карна алгоритм, TCP, 629
каталоговый сервер, 566
качество обслуживания, 56, 458
MPLS, 479
алгоритм дырявого ведра, 463
алгоритм справедливого обслуживания, 471
взвешенный, 472
гарантированная пересылка, 477
диспетчеризация пакетов, 470
дифференцированное обслуживание, 475
интегральное обслуживание, 472
класс эквивалентности пересылок, 480

качество обслуживания (*продолжение*)
коммутация меток, 479
маркерное ведро, 465
ориентированное на классы, 475
пропорциональная маршрутизация, 470
протокол резервирования ресурсов, RSVP, 472
резервирование ресурсов, 467
соглашение об уровне обслуживания, 461
срочная пересылка, 476
управление доступом, 468
спецификация потока, 469
формирование трафика, 461
квадратурная амплитудная модуляция, QAM-64, 160
квантобит, 828
квантование, 789
квантовая криптография, 826
диагональный базис, 827
квантобит, 828
прямолинейный базис, 827
усиление секретности, 829
фотон, 827
Кеплера, закон, 140
Кларк, Артур, 141
класс эквивалентности пересылок, 480
клиент, 24
клиент-серверная модель, 24
ключ сеанса, 887
ключ шифрования, 819
ключевой поток, 845
коаксиальный кабель, 121
код, 818
код Грея, 343
код с обнаружением ошибок, 233
код Уолша—Адамара, 344
кодек, 175
кодирование, 786
с предсказанием, 178
кодирование длин серий, 790
кодовое расстояние, 233
кодовое слово, 233
кодово-импульсная модуляция, PCM, 175
коды Уолша, 200
колесо времени, 647
команда SABM, 279
команда SNRM, 279
комбинированная перевозка, 252
коммуникационная подсеть, 42
коммуникационная среда, 25
коммутатор, 319, 329
мобильных телефонов, 190
коммутация
каналов, 182
пакетов, 182, 183, 185
сообщений, 184, 185
коммутируемая сеть Ethernet, 329
коммутируемая телефонная сеть общего пользования, 150
компьютерная сеть, 22
применение, 23
конвейерная обработка, 258
конверт, электронная почта, 672
конечный автомат, 270, 595
консорциум WWW, 694
контора рваных лент, 185
контрольная сумма, 227, 237
конференция, 27
концентратор, 318
корпорация по присвоению имен и номеров, ICANN, 502
корректирующие коды, 233
корректирующий код, 234
кратчайшего пути выбор, 409
кредитное сообщение, 595
криптоанализ, 820, 848
показатель трудозатрат, 821
проблема известного открытого текста, 821
проблема произвольного открытого текста, 821
проблема только зашифрованного текста, 821
три варианта задач, 821
криптография, 814
IDEA, 901
P-блок, 833
Rijndael, 839
RSA, 850
S-блок, 833
алгоритмы с открытым ключом, 849
безопасность за счет неясности, 820
зашифрованный текст, 819
квантовая, 826
ключ шифрования, 819

криптография (*продолжение*)
код, 818
открытый текст, 819
принцип Керкгофа, 820
принципал, 826
протокол BB84, 826
симметричный ключ, 832
сцепление блоков шифра, 842
традиционная, 818
шифр, 818
шифр AES, 837
шифр DES, 834
криптология, 820
критика эталонной модели TCP/IP, 73
кэширование, 742
иерархическое, 742
упреждающее, 744

Л

летающая локальная сеть, 46
лизинг IP-адресов, 520
линия связи, 42
локальная сеть, 39
Ethernet, 39
распределение канала, 292

М

магистраль, 42
магистральная область, OSPF, 522
макроблок, 793
Максвелл, Джеймс, 93
максимальная единица передачи, 610
манчестерское кодирование, 321
маркер, 95
маркерное ведро, 465
маркерное кольцо, 95
маршалинг, 601
маршрутизатор, 42
маршрутизация, 56, 406
IS-IS, 423
адаптивная, 408
Беллмана–Форда, 414
в объединенных сетях, 490
внешний шлюзовый протокол, 491
внутренний шлюзовый потокол, 491
в специализированных сетях, 434
AODV, алгоритм, 434
активный сосед, 438
маршрутизация (*продолжение*)
пакет запроса маршрута, 435
пакет наличия маршрута, 437
выбор кратчайшего пути, 409
заливка, 412
иерархическая, 424
регион, 424
многоадресная, 428
мобильный хост, 430
неадаптивная, 408
от источника, 501
пересылка, 407
по вектору расстояний, 414
продвижение по встречному пути, 427
пропорциональная, 470
протокол OSPF, 521
с учетом состояния линий, 417
сеансовая, 407
статическая, 408
Форда–Фулкерсона, 414
широковещательная, 426
многоадресная, 426
маска подсети, 505
Маха–Цандера интерферометр, 126, 312
медный провод
сравнение с оптическим волокном, 129
междугородная телефонная линия, 152
международная организация по стандартизации ISO, 62, 102
международный союз телекоммуникаций, 100
международный союз телекоммуникаций ITU, 787
международный стандарт, 103
ISO 3166, 660
ISO 8859-1, 714
межсетевой протокол управления группами, 529
межсетевой уровень, 66
межстанционная линия, 152
местная линией связи, 151
местная телекоммуникационная компания, LEC, 154
местная телефонная компания BOC, 153
метод с явным управлением, 481
метод хорды, 440
метод, HTTP, 736
метод, управляемый данными, 480

микроволновая связь, 135
миллиметровое излучение, 138
минимальное кодовое расстояние, 234
Министерство связи, 100
Мировая паутина, 22
многоадресная маршрутизация, 428
многоадресная передача, 38
многоадресная рассылка, 428
  Интернет, 528
многоадресный алгоритм OSPF, 806
многоадресный маршрутизатор, 803
многолучевое затухание, 135
многомодовое стекловолокно, 123
многопоточный сервер, 701
многосвязная сеть, 527
множественный доступ
  с контролем несущей, 300
  с предотвращением столкновений, 315
мобильная коммерция, 33
мобильная телефония
  третье поколение, 204
мобильные беспроводные сети, 32
мобильные системы связи, 188
мобильный IP-протокол, 529
мобильный коммутационный центр, 190
мобильный хост
  маршрутизация, 430
модем, 157, 158
модуль данных транспортного протокола, 555
модуляция, 158
  амплитудная, 158
  квадратурная амплитудная, 160
  квадратурная амплитудная, QAM-64, 160
  фазовая, 158
    квадратурная, QPSK, 159
  частотная, 158
моноалфавитный подстановочный шифр, 822
моноканал, 94
мост, 370
  между сетями стандарта IEEE 802, 373
  связующее дерево, 377
  удаленный, 378
мультимедиа, 761
  MBone, 803
  аудио, 761
  видео, 782
мультимедиа (*продолжение*)
  видео по заказу, 795
  сжатие данных, 786
мультиплексирование, 55, 582
  восходящее, 582
  нисходящее, 582
  с временным уплотнением, 171
мультиплексирование с разделением времени, 174
мультиплексор доступа к DSL, DSLAM, 167
муниципальная сеть, 40

Н

Нагля алгоритм, 621
надежная служба, 56
надежный алгоритм хэширования SHA, 859
назначенный маршрутизатор, 525
Найквиста ограничение, 118
Найквиста теорема, 118
настойчивости таймер, 629
Национальный институт стандартизации США, ANSI, 102
национальный институт стандартов и технологий, 837
Национальный институт стандартов и технологий США, 103
начальное состояние, 271
не зависящая от протокола многоадресная рассылка, 806
неадаптивная маршрутизация, 408
ненумерованный кадр, 277
непрямой протокол TCP, 630
несущая частота, 158
нисходящее мультиплексирование, 582
новости, 83
нонс, 895
носитель
  T1, 175
  T2, 178
  T3, 178
  T4, 178

О

область локального доступа и транспорта, LATA, 154

область, Kerberos, 899
область, OSPF, 522
облегченный протокол службы каталогов, LDAP, 668
обман DNS, 910
обмен ключами Диффи—Хеллмана, 892
обнаружение ошибок, 232
обрабатывающий сервер, 566
обработка ошибок, 230
образующий многочлен, 237
обратный поиск, 664
общество Интернета, 105
объединение сетей, 481
  дейтаграммный интерсетевой стиль, 486
  интерсеть, 481
  маршрутизация, 490
  не использующее соединений, 487
  ориентированное на соединение, 486
  туннелирование, 489
  фрагментация, 492
объединенная сеть, 49
однобитового скользящего окна протокол, 254, 255
одномодовое стекловолокно, 123
однонаправленная передача, 38
одноразовый блокнот, 824, 825
оклик—отзыв, 887
окно перегрузки, 624
оконечная телефонная станция, 151
оператор линии дальней связи IXC, 154
оператор связи, 99
оператор связи общего пользования, 99
оптимальности принцип, 408
оптический канал, 330
оптический носитель OC, 181
оптическое волокно
  сравнение с медным проводом, 129
оптоволоконная сеть, 127
оптоволоконная технология, 122
  мода, 123
оптоволоконный кабель, 125
Организация Объединенных Наций, 100
ортогональность, 200
ослабление сигнала, 157
ослабление силы света, 124
ответвительный кабель, 319
открытый текст, 819
отмычка, 885
отношение сигнал/шум, 118
отравленный кэш, 910
отсечение путей, 806

П

пакет, 37
пассивная звезда, 128
перегрузка, 444
передача TCP, 704
передача речи поверх IP, 774
передача с промежуточным хранением, 184
переключающий элемент, 42
перенос файлов, 83
переносимая программа, 920
перестановочный шифр, 823
переход, 271
персональные сети, 38
песочница, 921
Петри сетевая модель, 273
пиксел, 785
плоскость пользователя, ATM, 91
плоскость управления, ATM, 91
плотное WDM, 174
повторитель, 127, 320
пограничный межсетевой протокол, 526
подключаемый модуль, 698
подпись кода, 922
подпись, цифровая, 853
подсеть, 42
  Интернет, 503, 504
  с коммутацией пакетов, 43
    принцип работы, 43
  с промежуточным хранением, 43
подсеть виртуального канала, 402
подстановочный шифр, 821
подуровень
  MAC, 292
  PMD, ATM, 92
  конвергенции CS, ATM, 92
  сегментации и повторной сборки, 92
  управления доступом к среде, 292
показатель трудозатрат, криптоанализ, 821
поле, видео, 783
полиномиальный код, 237
политика, 73
политика трафика, 462
полоса пропускания, 117

полудуплексное соединение, 162
пользовательский агент, 670
популярный порт, 608
пороговое значение перегрузки, 625
порт, TCP, 608
  популярный порт, 608
портал, 98
последняя миля, 157
последовательность Баркера, 343
поставщик услуг Интернета, 280
постоянный виртуальный канал ATM, 89
посылающее окно, 253
поток, 458
поток T4, 178
потоковая информация, 761
потоковое аудио, 767
  верхний предел, 770
  выталкивающий сервер, 770
  метафайл, 768
  нижний предел, 770
  проталкивающий сервер, 770
потоковые алгоритмы, 472
почти видео по заказу, 795
почтовый ящик, 671
предлагаемый стандарт, 105
предотвращение перегрузки, 448
предсказание заголовка, 646
пригородно-междугородная станция, 152
приемопередатчик, 319
прикладной уровень, 65, 68
  DNS, 658
  Всемирная паутина, 693
примитивы, служба, 58
принимающее окно, 253
принцип оптимальности, 408
принципал, 826
проблема
  двух армий, 574
  засвеченной станции, 315
  скрытой станции, 315
  счета до бесконечности, 415, 417
проблемная группа проектирования Интернета, 472
провайдер услуг Интернета, 83
продвижение по встречному пути, 427
продукционный шифр, 833, 834
проект стандарта, 105
произведение пропускной способности и задержки, 636, 637
производительность, 633
производительность сетей, 633
  Ethernet, 326
прокси, 742
пропорциональная маршрутизация, 470
проталкивающий сервер, 770
протокол, 51, 61, 69
  элементарный передачи данных, 240
  ADCCP, 276
  ARP, 516, 517
    ARP-прокси, 518
  ARQ, 250
  BGP, 526
  BOOTP, 519
  CSMA, 300
  CSMA 1-настойчивый, 300
  CSMA ненастойчивый, 301
  CSMA с настойчивостью p, 301
  CSMA/CD, 302
  DHCP, 520
    агент ретрансляции, 520
  DVMRP, 804
  ESMTP, 686
  FTP, 707
  H.245, 775
  H.323, 514
  HDLC, 276
  HTTP, 706, 735
  HTTPS, 917
  ICMP, 515
  IGMP, 529, 805
  IMAP, 690
  IP, 67, 497, 498
  IPv5, 533
  IPv6, 533
  LAP, 276
  LAPB, 276
  LCP, 281
  LLC, 339
  LTP, 753
  MACA, 315
  MACAW, 316
  NCP, 281
  PAR, 250
  PIM, 806
  POP3, 687
  PPP, 281

протокол (*продолжение*)
RARP, 519
RTCP, 606
RTP, 603
RTSP, 768
SDLC, 276
SIP, 778
SIPP, 533
SMTP, 683
T/TCP, 632
TCP, 67, 607, 610, 629, 683
UDP, 68
WDMA, 310
WEP, 881
адаптивного дерева, 309
аутентификации, 886
аутентификации Kerberos, 897
аутентификации Нидхэма—Шрёдера, 895
аутентификации, Отуэя—Риса, 897
без столкновений, 304
беспроводная локальная сеть, 313
битовой карты, 304
внешний шлюзовый, 491
внутренний шлюзовый, 491
гигабитная сеть, 648
защищенных сокетов, SSL, 916
коллективного доступа, 295
множественного доступа с контролем несущей, 300
множественного доступа со спектральным разделением, 310
обмена ключами Диффи—Хеллмана, 892
оклик—отзыв, 887
передачи с контролем потока, SCTP, 633
передачи файлов, FTP, 514
резервирования ресурсов, RSVP, 472
с возвратом на *n*, 257
с выборочным повтором, 264
с двоичным обратным отсчетом, 306
с контролем несущей, 300
с ограниченной конкуренцией, 307
симплексный для каналов с шумом, 248
симплексный с ожиданием, 247
скользящего окна, 252, 253
однобитового, 254, 255
протокол mailto, 707
протокол TCP, 67
протокол внешнего шлюза, 521
протокол внутреннего шлюза, 521
протокол начального соединения, 566
протокол обратного определения адреса, 519
протокол передачи от точки к точке, 281
протокол разрешения адресов, 517
протокол с ожиданием, 247
протокол управления каналом связи, 281
протокол управления передачей, 607
протоколы с резервированием, 305
профиль пользователя, 674
профиль сообщения, 856, 857
процедура доступа к каналу, 276
прямое исправление ошибок, 233
прямолинейный базис, 827
псевдоним, электронная почта, 673
пункт доступа к сети, NAP, 81

P

равноранговая сеть, 439
идентификатор узла, 440
ключ, 441
метод хорды, 440
таблица указателей, 442
равноранговые сети, 28
равноранговые сущности, 51
равноранговые узлы сети, 51
радиосвязь, 133
радиотелефон, 187
разбиение на полосы, 799
разветвитель, 166
разностное манчестерское кодирование, 321
разрыв соединения, 573
разрыв соединения, TCP, 616
распознаватель, DNS, 659
распределение канала в локальных сетях, 292
распределенная система, 22
распределительная сеть, 800
расстояние по Хэммингу, 233
расширенный спектр
с перестройкой частоты, 132
с прямой последовательностью, 133
режим
группового шифра, 845
счетчика, 846
шифрованной обратной связи, 844

режим электронного шифроблокнота, 842
рекламное объявление, 531
рекурсивный запрос, 667
репликация серверов, 744
ретрансляция кадров, 88
речевой канал, 117
решетчатое кодирование, TCM, 161
ряд Фурье, 115

C

самозаверяющийся URL, 915
сброс нагрузки, 454
свобода слова, 927
связка
  закрытых ключей, 905
  открытых ключей, 906
связующее дерево, 377, 427
связующее ПО, 22
связь
  в видимом диапазоне, 138
сдерживающий пакет, 452
сеанс связи, 65
сеансовая маршрутизация, 407
сеансовый уровень, OSI, 65
сегмент, TCP, 610
сервер, 24
сервер Apache, 730
сервер имен, 566, 665
серверная ферма, 704
сервис, 69
  без установления соединения, 56
  с установлением соединения, 56
сертификат, 864
  атрибут, 865
  управление сертификации, 864
сетевая модель Петри, 273, 274
  входящая дуга, 274
  маркер, 274
  переход, 274
    разрешенный, 274
  позиция, 273
сетевая служба
  точка доступа, 564
сетевой протокол управления, 281
сетевой уровень, 399
  алгоритм маршрутизации, 406
  борьба с перегрузкой, 444
  вопросы проектирования, 400
сетевой уровень (*продолжение*)
  Интернет, 496
  объединение сетей, 481
  предоставляемые сервисы, 401
сетевой уровень, OSI, 64
сети
  стандартизация, 98
сеть
  ANSNET, 81
  ARPANET, 66, 75, 659
  Ethernet, 39
  NREN, 81
  NSFNET, 80
  множественного доступа, 522
  оптоволоконная, 127
сжатие данных, 786
  без потерь, 787
  с потерями, 787
сжатие звука, 764
  MP3, 764
  временное маскирование, 764
  кодирование формы сигналов, 764
  маскирование звука, 764
  перцепционное кодирование, 764
  психоакустика, 764
  частотное маскирование, 764
сигнализация
  ассоциированная с каналом, 176
  по общему каналу, CCS, 176
символьное заполнение, 229
симметричный ключ, криптография, 832
симплексное соединение, 162
симплексный протокол
  для каналов с шумом, 248
  с ожиданием, 247
симпозиум ACM SIGOPS, 76
синдром глупого окна, 621
синхронизация, 65
синхронная оптическая сеть SONET, 179
синхронная цифровая иерархия, 179
синхронное управление каналом, 276
синхронный транспортный сигнал
  STS-1, 180
система
  PGP, 901
система диалоговых сообщений, 27
система записи абстрактного
  синтаксиса 1, 866

система с конкуренцией, 295, 296
система с конфликтами, 302
система с открытыми ключами
  PKI, 867
    аннулирование, 870
    доверительная цепочка, 869
    доверительный якорь, 869
  инфраструктура, 867
сквозной коммутатор, 382
скользящего окна протокол, 252
скорость света, 130
слоны
  апокалипсис, 71
служба, 61
  DNS, 658
  дейтаграмм, 57
  дейтаграмм с подтверждениями, 57
  запросов и ответов, 57
  примитивы, 58
служба вечности, 928
служба имен доменов DNS, 68, 79, 516, 658
служебный примитив
  пример, 585
смайлик, 668
  emoji, 756
смежный маршрутизатор, 525
совет по архитектуре Интернета, IAB, 104
совместное использование
  информации, 24
  ресурсов, 23
соединение
  дуплексное, 162
  полудуплексное, 162
  разрыв, 573
  симплексное, 162
  установка, 567
сокет, 557
сотовая телефония
  2,5G, 205
  AMPS, 189
  CDMA, 198
    элементарная последовательность, 199
    элементарный сигнал, 199
  CDMA2000, 205
  EDGE, 205
  GPRS, 206
  PCS, 193
  UMTS, 205
  W-CDMA, 204
сотовая телефония (*продолжение*)
  базовая станция, 190
  выделенный управляющий канал, 198
  канал предоставления доступа, 198
  канал случайного доступа, 198
  микросоты, 190
  общий управляющий канал, 198
  пейджинговый канал, 198
  передача (handoff), 191
    жесткая, 191
    мягкая, 191
  передача с помощью телефона, MAHO, 195
  соты, 190
  управление вызовом, 192
  цифровая, 193
  широковещательный управляющий канал, 198
сотовый телефон, 187
соты, 190
социальный аспект сетей, 35
спам, 36
спектральное уплотнение, 173
специализированная сеть, 434
специальная сеть, 96
спецификация потока, 469
список рассылки, 671
спутник GPS, 145
спутник связи
  GEO, геостационарные спутники, 141
  Iridium, 145
  LEO, низкоорбитальные спутники, 145
  MEO, средневысотные спутники, 145
  Teledesic, 147
  геостационарный, 141
  концентратор, 144
  позиционрование, 142
  система Globalstar, 147
  точечный луч, 143
  частотные диапазоны, 142
сравнение эталонных моделей OSI и TCP, 69
среда распространения сигнала, 119
срочные данные, TCP, 609
стандарты
  802.11b, 344
  802.11g, 344
  802.11a, 98
  802.11b, 98
  AES, 837

стандарты (*продолжение*)
G.711, 775
G.723.1, 775
H.323, 774
JPEG, 787
MPEG, 791
TLS, 920
X.400, 670
X.509, 866
Интернета, 105
стандарт 802.11, 341
стандарт DIX, 94
стандарт цифровой подписи DSS, 856
стандарт шифрования данных DES, 834
стандартизация
ISO, 102
Интернет, 104
сетей, 98
телекоммуникаций, 99
стандарты
802.11a, 344
802.15, 362
802.16, 353
802.1Q, 387
802.3u, 331
802.3z, 334
de facto, 99
de jure, 99
статическая маршрутизация, 408
стационарные беспроводные сети, 32
стеганография, 929
стек протоколов, 52
стратегия
винная, 455
молочная, 455
супервизорный кадр, 277
супергруппа, 173
сцепление блоков шифра, 842
сцепленные виртуальные каналы, 486

Т

таймер настойчивости, TCP, 629
таймер повторной передачи, TCP, 626
тангентная система, 189
тег, HTML, 713
телевидение
аналоговое, 782
телевидение (*продолжение*)
высокой четкости, 784
цифровое, 784
телекоммуникации
стандартизация, 99
телефонная система, 149
коммутация, 182
носитель T1, 175
политика, 153
тело письма, электронная почта, 672
теорема Найквиста, 118
точка входа в сеть, 85
точка доступа, 96
точка присутствия, 84
точка присутствия, POP, 154
транзитная станция, 152
транзитные сети, 527
транспондер, 140
транспортная служба
пользователь, 553
поставщик, 553
точка доступа, 564
транспортная сущность, 552
транспортный объект, 552
транспортный протокол, 563
UDP, 598
адрес, 564
мультиплексирование, 582
управление потоком, 577
элементы, 563
транспортный уровень, 67, 551
OSI, 64
предоставляемые сервисы, 551
пример, 585
производительность сетей, 633
элементы протокола, 563
триграмма, 822
тройное рукопожатие, 571–572
туннелирование, 489
тупик, протокол, 273
тупиковая сеть, 527

У

удаленный доступ, 83
уединенная волна, 125
университет Карнеги—Меллона, 35
Уолша—Адамара, код, 344
уплотнение
частотное, 172–173

уплотнение каналов, 55
управление
  допуском, 450
  логическим соединением, 339
  маркерами, 65
  потоком, 231, 577
управление диалогом, 65
управление потоком, 55
  с обратной связью, 232
  с ограничением скорости, 232
управление почтово-телеграфной и телефонной связи, 100
управление сертификации, 864
управляющий элемент ActiveX, 921
уровень, 50
  AAL ATM, 90
  ATM, 90
  адаптации ATM, 90
  межсетевой, 66
  представления, 65
  прикладной, 65, 68
  сеансовый, 65
  сетевой, 64, 399
  транспортный, 64, 67
  физический, 63, 114
  хост-сетевой, 69
уровень передачи данных, 222
  элементарные протоколы, 240
  OSI, 63
  аспекты устройства, 223
  битовое заполнение, 229
  Интернет, 280
  кадр, 223, 227
  обработка ошибок, 230
  предоставляемые сервисы, 224
  пример протоколов, 276
  протокол HDLC, 276
  протокол LLC, 339
  протокол скользящего окна, 252
  символьное заполнение, 229
  управление потоком, 231
  флаговый байт, 228, 229
установка соединения, 567
установка соединения, TCP, 615
устройства маршрутизации, 379
устройство сопряжения с сетью, NID, 166

Ф

Фабри—Перо интерферометр, 126, 312
фаззбол, 80
фазовая модуляция, 158
  квадратурная, QPSK, 159
Федеральная комиссия связи США, 138
федеральный стандарт обработки информации, 838
физическая среда, 51
физический носитель, 119
физический уровень, 114
  OSI, 63
  беспроводная передача, 130
  носители информации, 119
  телефонная система, 149
фильтр частотный, 117
флуктуация, 456, 762
форма, HTML, 719
формирование трафика, 461
фотон, 827
фрагментация, при объединении сетей, 492
Фурье анализ, 115
Фурье ряд, 115

Х

хост, 42
хост-сетевой уровень, 69

Ц

цветность, 784
цензура, 927
  университет Карнеги—Меллона, 35
центр распространения ключей, 887, 894
циклический избыточный код, 237
Ципфа, закон, 797
цифровая подпись, 853
  MD5, 858
  с открытым ключом, 855
  с секретным ключом, 853
цифровая сотовая телефония, 193
цифровое видео, 784
цифровые абонентские линии, DSL, 163

Ч

частная равноранговая связь, 85
частная сеть, 879
частота, 130
частота среза, 117

частотная манипуляция, 158
частотная модуляция, 158
частотное мультиплексирование, 171
частотное уплотнение, 171, 173
частотный диапазон, 131
  UHF, 131
  VHF, 131
чат, 27
чересстрочная развертка, 783
чистая система ALOHA, 296

Ш

Шеннона ограничение, 118
широковещание, 38, 322
широковещательная сеть, 37, 39
широковещательный шторм, 635
широковещательный штурм, 384
широкополосная сеть, 163
шифр, 818
  перестановочный, 823
  AES, 837
  DES, 834
  блочный, 832, 834
  моноалфавитная подстановка, 822
  подстановочный, 821
  продукционный, 833
  Цезаря, 822
шифр AES, 837
шифр DES, 834
  полемика, 835
шифрование
  E_0, 885
  в канале связи, 816
  с открытым ключом, 849
шифрованная панковская рассылка, 926
шлюз, 49, 486
шлюз прикладного уровня, 878
шум, 157
  импульсный, 158
  перекрестные помехи, 157
  термальный, 157
шум квантования, 762

Э

электромагнитный спектр, 130, 131
электронная почта, 82, 668
  пользовательский агент, 672
  ESMTP, 686
электронная почта (*продолжение*)
  MIME, 677
  POP3, 687
  агент передачи сообщений, 670
  архитектура и службы, 670
  доставка сообщения, 686
  заголовок, 672
  команды пользователя, 674
  конверт, 672
  конфиденциальность, 901
  основные функции, 670
  отправление, 673
  пересылка писем, 683
  пользовательский агент, 670
  почтовый ящик, 671
  тело письма, 672
  фильтр, 691
  формат RFC 822, 675
  формат сообщений, 675
  чтение, 674
электронный бизнес, 26
эталонная модель, 62
  ATM, 90
  OSI, 62
  TCP/IP, 66
эталонная модель OSI, 62
  критика, 70
  сравнение с TCP/IP, 69
эталонная модель TCP/IP
  критика, 73
  сравнение с OSI, 69

Я

язык
  cHTML, 754
язык JavaScript, 731
язык Perl, 728
язык Python, 728
язык XML, 722
язык XSL, 723
язык разметки, 712
яркость, 784
ярусная перевозка, 252
ячейка, 43
ячейка, ATM, 89
ячейка, HTML, 717

Э. Таненбаум

Компьютерные сети

4-е издание

Перевел с английского В. Шрага

Главный редактор	*Е. Строганова*
Заведующий редакцией	*И. Корнеев*
Руководитель проекта	*А. Васильев*
Научный редактор	*С. Орлов*
Литературные редакторы	*Т. Маслова, В. Шрага*
Художник	*Н. Биржаков*
Иллюстрации	*В. Шендерова*
Корректор	*Н. Рощина*
Верстка	*С. Панич*

Лицензия ИД № 05784 от 07.09.01.
Подписано в печать 15.07.03. Формат 70×100/16. Усл. п. л. 79,98.
Тираж 4000 экз. Заказ № 258.
ООО «Питер Принт». 196105, Санкт-Петербург, ул. Благодатная, д. 67в.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953005 – литература учебная.
Отпечатано с готовых диапозитивов в ФГУП «Печатный двор» им. А. М. Горького
Министерства РФ по делам печати, телерадиовещания и средств массовых коммуникаций.
197110, Санкт-Петербург, Чкаловский пр., 15.