1. **Предпосылки возникновения сетей. Краткая история развития ЭВМ и методов доступа к ним. Кто, как и для чего использует сеть ЭВМ. Организация вычислительных сетей. Классификация сетей ЭВМ. Организация программного обеспечения сетей ЭВМ**

1. Появление технически сложных систем.

Новые способы получения энергии открыли новые горизонты в развитии промышленности. Однако создание технически сложных систем, в свою очередь, стало требовать принципиально новых технологий проектирования. Так, например, технически сложную систему не всегда можно представить «в натуральную величину», например самолет, космический корабль, спутниковую систему. Требуется моделирование, специальные методы борьбы со сложностью. Например, увеличение числа взаимодействующих компонентов системы ведет к усложнению конструкции самих компонентов, а следовательно к снижению их надежности.

2. Необходимость быстрого получения информации.

Помимо технических предпосылок, определяющую роль в развитии информационных технологий сыграли предпосылки социальные. В обществе XX века налицо были следующие тенденции:

A. Демографический рост.
B. Территориальная децентрализация населения.
C. Рост числа людей, вовлекаемых в процесс принятия решений.

При развитии данных тенденций отсутствие эффективных методов коммуникации, распределенного доступа к информации, ее автоматического сбора, обработки и хранения тормозили развитие экономики - как на внутригосударственном, так и на межгосударственном уровне. Таким образом, в XX веке стало ясно: развитие информационных технологий напрямую связано с конкурентоспособностью - как отдельного предприятия, так и государства в целом. Как следствие первой названной нами причины совершенствуются методы обработки информации: от механических арифмометров, табуляторов с управлением от перфокарт (1928) до ЭВМ. Возможности компьютеров, их размеры, технические характеристики растут с фантастической скоростью. Появляются устройства памяти объемом в несколько терабайт. Вторая причина обуславливает изменение технологий сбора и передачи информации. Появляются телефонные сети всемирного масштаба, теле- и радиосети, спутники связи.

Способы доступа к вычислительным установкам

- Однопользовательские ЭВМ
- Системы пакетной обработки
- Системы с разделением времени и многотерминальные системы

Виды вычислительных установок

- Карманные персональные компьютеры (КПК)
- Персональные компьютеры (ПК)
- Вычислительные комплексы
- Встроенные системы
- Сети ЭВМ
- Распределенные системы (GRID)

Приведем основные преимущества, которые получают организации, используя сети.

- управление ресурсами
- повышение надежности функционирования предприятия за счет оперативности управления и использования имеющихся ресурсов.
- сокращение затрат на функционирование предприятия – оптимизация бизнес-процессов.

- повышение экономической эффективности за счет гибкой организации работы информационных систем (отсутствие складов, принятие решений)
- средство общения и связи (телеконсультации и конференции, оперативность принятия решений)
- офис в кармане – позволяет сотрудникам получить доступ ко всем устройствам, файлам, базам данных и т.п. вне зависимости от их физического местоположения
- удобства при подготовке персонала (в некоторых крупных западных фирмах стоимость подготовки вновь принятого сотрудника достигает 50 000 долларов).
- управление производством и стратегией развития (ERP-системы ЦБ РФ, FedExp, GM склад, Газпром)

Сети для индивидуальных пользователей

Преимущества использования сетей для индивидуальных пользователей:

- доступ к информации (Интернет)
- общение с другими людьми (новости, электронная почта, видеоконференции)
- обучение
- развлечение
- получение услуг (взаимодействие с предприятиями, государственными структурами)
- средство исследования

Организация вычислительных сетей.

Все оборудование сети можно разделить на абонентские машины и транспортную среду или транспортную подсеть, которую иногда называют просто подсеть. Однако здесь надо быть осторожным, так как у термина подсеть есть и второй смысл, связанный с адресацией в сети Интернет. О подсети в этом смысле мы поговорим позже.

Машины, на которых работают приложения, называют абонентскими или хост-машинами (host). Абонентские машины обеспечивают интерфейс пользователей и работу приложений в сети. Хосты подсоединены к транспортной среде. Назначение транспортной среды обеспечить передачу данных от одного хоста к другому.

Транспортная среда состоит из системы передачи данных и коммуникационных машин (далее К-машины или для краткости КМ). Коммуникационные машины - это специализированные компьютеры, соединяющие две и более систем передачи данных. Примером К-машины является маршрутизатор - компьютер, который выбирает маршрут, по которому должны следовать данные между абонентскими машинами в сети. На рисунке 1-1 абонентские машины (далее А-машины или кратко АМ) показаны в виде прямоугольников, коммутирующие элементы - в виде кружков, а сплошными линиями - системы передачи данных.

Системы передачи данных

Система передачи данных обеспечивает передачу данных между машинами в сети. Эти машины не обязательно абонентские. Система передачи данных состоит из каналов, каналообразующей аппаратуры, коммутирующих элементов (например, коммутаторов). Каналы передачи данных – это линии связи самой различной природы и каналообразующая аппаратура. Каналы соединяют две или более коммуникационных машины, либо коммутатора. Каналообразующая аппаратура обеспечивает интерфейс между линией связи и оконечным устройством, например, А-машиной, К-машиной или коммутатором. На сегодня отсутствует общепринятая классификация систем передачи данных. Для них, как и для сетей, есть три общепризнанных критерия, по которому их различают, это:

- Способ коммутации потоков данных
- Тип каналов
- Топология системы

Выделяют два основных способа коммутации потоков данных – коммутацию каналов и коммутацию пакетов. Коммутация каналов – метод управления потоком данных в реальном времени, для которого характерно следующее:

- Темп передачи данных определяется передатчиком.
- Канал передачи создается до начала передачи (установление соединения) и фиксируется на все время передачи.
- Сохраняет порядок передаваемых данных.
- Имеется большой опыт его создания.
- Есть хорошо развитая инфраструктура.

В то же время этот способ управления потоками данных отличается от других способов рядом недостатков:

- неэффективным использованием ресурсов
- низкой надежностью
- медленным установлением соединения

Коммутация пакетов - способ управления передачей, отличающийся следующими особенностями:

- Высокая скорость установления соединения (передатчик сразу начинает передачу и не ждет физического установления соединения).
- Низкий уровень ошибок в канале.
- Надежность.
- Рациональное использование ресурсов.
- Сильная зависимость времени передачи от загрузки сети.

В общем случае, все каналы по типу можно разделить на:

- Каналы «точка-точка». Они соединяют между собой только две машины. Все потоки данных, протекающие по каналу этого типа, доступны лишь этим двум машинам.
- Каналы с множественным доступом. Образуют линию передачи данных, общую для нескольких машин.

СПД с каналами с множественным доступом можно разделить по методам выделения канала на динамические и статические:

- Статические - временное разделение (time-shearing) канала между машинами; канал простаивает, если машине нечего передавать.
- Динамические - централизованные и распределенные механизмы выделения канала по запросу.

Топология соединения маршрутизаторов - важный фактор конструкции транспортной среды. От нее зависит время задержки данных при передаче, перегрузки и многие другие параметры функционирования сети. На рисунке 1-2 показаны типичные топологии, встречающиеся при организации транспортных сред.

Абонентские машины Абонентские машины обеспечивают работу приложений.

Сопряжение транспортных сред

Необходимость в сопряжении транспортных сред возникает, когда нужно обеспечить взаимодействие приложений, расположенных в разных сетях.

- Мост соединяет две однородные ТС.
- Шлюз – две разные по архитектуре ТС.

Классификация сетей ЭВМ.

Есть два общепризнанных фактора для их различения: технология передачи данных и масштаб. Технология передачи определяется системой СПД.

Масштаб сети - другой критерий для классификации сетей (таблица 1-4).

- многомашинный комплекс (система)
- локальная сеть (комната, здание, комплекс)
- городская сеть (город)
- региональная сеть (страна, континент)
- Интернет (планета)

Локальная сеть

Локальная вычислительная сеть (ЛВС) отличается от остальных по следующим характеристикам:

- Размер: комната, корпус, группа корпусов (отсюда известна максимальная задержка при передаче)
- Система передачи данных: как правило, канал с множественным доступом (вещание, скорость передачи от 10 Мбит/сек. до нескольких Гбит/сек., Ethernet)
- топология ТС ЛВС (рисунок 1-5)
  - линейная
  - кольцо
  - дерево

Городская сеть

Городская вычислительная сеть (MAN - Metropolitan Area Network) охватывает несколько зданий в пределах одного города либо город целиком. Как правило, она поддерживает передачу как данных, так и голоса и иногда объединяется с кабельной телевизионной сетью. Не имеет коммутаторов, базируется на одном или двух кабелях. Основная причина выделения этой категории сетей состоит в том, что для них был создан специальный стандарт IEEE 802.6 - DQDB - двойная магистраль с распределенной очередью (Distributed Queue Dual Bus).

Региональная сеть

Региональная вычислительная сеть (WAN - Wide Area Network) охватывает крупные географические области, такие как страны, континенты. Транспортная среда таких сетей строится на основе коммутации пакетов с помощью каналов типа «точка-точка». Часто в качестве СПД в таких сетях используют уже существующие системы связи, например, телефонные сети, спутниковые и радиосистемы.

Программное обеспечение сетей ЭВМ.

программное обеспечение определяет функционирование транспортной среды, системы передачи данных, взаимодействие абонентских машин с транспортной средой.

Иерархия протоколов

Сеть является сложной инженерно-технической системой. В целях борьбы с ее сложностью программное обеспечение сети, как правило, строго структурировано и организовано в виде иерархии слоев, или уровней. В разных сетях число уровней, их название, состав и функции могут быть разными. Однако во всех сетях назначение каждого уровня состоит в следующем:

- обеспечить определенный сервис верхним уровням
- сделать независимыми верхние уровни от деталей реализаций сервиса на нижних уровнях

Программное обеспечение уровня n на одной машине обеспечивает связь с программным обеспечением уровня n на другой машине. Правила и соглашения по установлению такой связи и ее поддержанию называются протоколом.

Основные понятия

Между каждой парой уровней есть интерфейс. Интерфейс определяет, какие услуги (сервис) нижележащий уровень должен обеспечивать для верхнего уровня, и с помощью каких примитивов - элементарных операций – верхний уровень может получить доступ к этим услугам. Интерфейс обеспечивает вышележащему уровню доступ к сервису нижележащего уровня. Например, как президент может подключить (отключить), если он на это имеет право, переводчика к его разговору с другим президентом.

Набор уровней и протоколов называется архитектурой сети. Конкретный набор протоколов, используемый на конкретной машине, называется стеком протоколов.

Основные вопросы организации уровней

Все функции организации и функционирования сети распределены между уровнями. В сетях с разной архитектурой это распределение между уровнями разное. Однако на каждом уровне необходимо решать следующие вопросы:

- адресация отправителя и получателя на уровне: на каждом уровне нужен механизм для адресации отправителей и получателей
- правила установления соединения с одноименным уровнем
- правила передачи данных
  - только в одном направлении - simplex, поочередно в обоих направлениях - half-duplex или в оба направления одновременно - duplex
  - допустимо ли совмещать виртуальные соединения вышележащего уровня через одно и то же соединение на нижележащем уровне; каково максимальное число совмещаемых так виртуальных соединений, каковы приоритеты в их обслуживании;
  - мультиплексирование и демультиплексирование виртуальных каналов
- обнаружение и исправление ошибок при передаче
- сохранение исходной последовательности данных при передаче
- на каждом уровне нужен механизм, предотвращающий ситуацию, когда одноименный уровень получателя начинает «захлебываться», т.е. когда отправитель отправляет пакеты с большей скоростью, чем получатель успевает их обрабатывать
- выбор маршрута при передаче: когда между получателем и отправителем есть несколько маршрутов, то какой из них выбрать?
- не все процессы на любом уровне могут работать с сообщениями произвольной длины, поэтому при передаче необходимо осуществлять:
  - разбиение, передачу и сборку сообщений
  - выбирать оптимальную длину фрагмента при разбиении или, наоборот, соединение нескольких коротких сообщений в одно более длинное (например, как быть, если процесс работает со столь короткими сообщениями, что их раздельная пересылка не эффективна?)

Интерфейсы и сервис

Как уже было сказано, одно из главных предназначений каждого уровня - обеспечить надлежащий сервис для вышележащего уровня.

Активные элементы уровня, т.е. те, которые могут сами совершать действия, в отличие от тех, над которыми совершают действия, будем называть активностями. Активности могут быть программными и аппаратными. Активности одного и того же уровня на разных машинах будем называть равнозначными или одноименными активностями. Активности уровня n+1 являются пользователями сервиса, создаваемого активностями уровня n, которые, в свою очередь, называются поставщиками сервиса. Сервис может быть разного качества,

например, быстрая и дорогостоящая связь или медленная и дешевая. Доступ к сервису осуществляется через так называемые точки доступа к сервису - SAP (Service Access Points). Каждая точка доступа к сервису имеет уникальный адрес. Например, телефонная розетка на стене - это точка доступа к сервису АТС. Каждой розетке сопоставлен определенный номер - номер телефона. Чтобы осуществить обмен информацией между двумя уровнями, нужно определить интерфейс между ними. Типичный интерфейс: активность на уровне n+1 передает IDU (Interface Data Unit - интерфейсную единицу данных) активности на уровне n через SAP (рисунок 1-10). IDU состоит из SDU (Service Data Unit - сервисной единицы данных) и управляющей информации. SDU передается по сети равнозначной сущности, а затем - на уровень n+1. Управляющая информация нужна нижележащему уровню, чтобы правильно передать SDU, но она не является частью передаваемых данных. Чтобы передать SDU по сети нижележащему уровню, может потребоваться разбить его на части. Каждая часть снабжается заголовком (header) и концевиком, и передается как самостоятельная единица данных протокола - PDU (Protocol Data Unit - единица данных протокола). Заголовок в PDU используется протоколом при передаче. В нем указано, какой PDU содержит управляющую информацию, а какой - данные, порядковый номер и т.д.

Сервис с соединением и сервис без соединения

Уровни могут предоставлять вышележащим уровням два вида сервисов: ориентированный на соединение и без соединения. Сервис с соединением предполагает, что между получателем и отправителем сначала устанавливается соединение, и только потом доставляется сервис. Пример - телефонная сеть. Сервис без соединения действует подобно почтовой службе. Каждое сообщение имеет адрес получателя. В надлежащих точках оно маршрутизируется по нужному маршруту, независимо от других сообщений. При таком сервисе вполне возможно, что сообщение, позже посланное, придет раньше. В сервисе с соединением это невозможно.

Примитивы сервиса

Формально сервис можно описать в терминах примитивных операций, или примитивов, с помощью которых пользователь или какая-либо активность получает доступ к сервису. С помощью этих примитивов активность на вышележащем уровне сообщает активности на нижележащем уровне, что необходимо сделать, чтобы вышележащая активность получила нужную услугу (сервис). В свою очередь, нижележащая активность может использовать эти примитивы, чтобы сообщить вышележащей активности о действии, выполненном равнозначной активностью. Для иллюстрации работы примитивов рассмотрим, как соединение устанавливается и разрывается. Сначала активность выполняет CONNECT.request, в результате чего в сеть выпускается пакет. Получатель получает CONNECT.indication, указывающий на то, что с ним хотят установить соединение. В ответ получатель через примитив CONNECT.response сообщает, готов он установить соединение или отказывает в установлении соединения. В результате активность - инициатор установления соединения получает ответ через примитив CONNECT.confirm, чего следует ожидать.

2. Модели сетевого взаимодействия OSI ISO и TCP/IP.

Эталонная модель OSI

Модель OSI (Open Systems Interconnection - модель взаимодействия открытых систем (рисунок 1-13) была разработана Международной организацией по стандартизации (МОС - International Standards Organization (ISO)) - для определения международных стандартов компьютерных сетей. Эта модель описывает, как должна быть организована система, открытая для взаимодействия с другими системами.

Физический уровень

Физический уровень отвечает за передачу последовательности битов через канал связи. Одной из основных проблем, решаемых на этом уровне, является то, как гарантировать, что если на одном конце отправили 1, то на другом получили 1, а не 0. Здесь в основном решаются вопросы механики и электрики.

Уровень канала данных

Основная задача уровня канала данных - превратить несовершенную физическую среду передачи в надежный канал, свободный от ошибок передачи. Эта задача решается разбиением данных отправителя на фреймы (обычно от нескольких сотен до нескольких тысяч байтов), последовательной передачей фреймов и обработкой фреймов уведомления, поступающих от получателя. И это уже заботы уровня - как бороться с дубликатами одного и того же фрейма, потерями или искажениями фреймов. Другая проблема, возникающая на уровне канала данных (равно как и на других вышележащих уровнях), - как управлять потоком передачи. Если канал позволяет передавать данные в обоих направлениях одновременно, т.е. если фреймы уведомления для потока от А к В используют тот же канал, что и трафик от В к А, то можно использовать для передачи фреймов уведомлений от В к А фреймы DU от А к В.

Сетевой уровень

Основная проблема, решаемая на сетевом уровне, - как маршрутизировать пакеты от отправителя к получателю. Если в транспортной подсети циркулирует слишком много пакетов, то они могут использовать одни и те же маршруты, что будет приводить к заторам или перегрузкам. Эта проблема также решается на сетевом уровне. Поскольку за использование транспортной подсети, как правило, предполагается оплата, то на этом уровне присутствуют функции учета: как много байт (или символов) послал или получил абонент сети.

Транспортный уровень

Основная функция транспортного уровня - принять данные с уровня сессии, разделить, если надо, на более мелкие единицы, передать на сетевой уровень и позаботиться, чтобы все они дошли в целостности до адресата. Сетевой уровень определяет, какой тип сервиса предоставить вышележащим уровням и пользователям сети. Наиболее часто используемым сервисом является канал «точка-точка» без ошибок, обеспечивающий доставку сообщений или байтов в той последовательности, в какой они были отправлены. Другой вид сервиса - доставка отдельных сообщений без гарантии сохранения их последовательности или, например, рассылка одного сообщения многим в режиме вещания. В каждом конкретном случае сервис определяют при установлении транспортного соединения. Транспортный уровень - это уровень, обеспечивающий соединение «точка-точка». Активности транспортного уровня на машине отправителя общаются с равнозначными активностями транспортного уровня на машине получателя. Этого нельзя сказать про активности на нижележащих уровнях. Они общаются с равнозначными активностями на соседних машинах. В этом одно из основных отличий уровней 1-3 от уровней 4-7. Последние уровни обеспечивают соединение «точка-точка». Это хорошо видно на рисунке 1-13. Транспортный уровень также отвечает за установление и разрыв транспортного соединения в сети. Транспортный уровень также должен предотвращать «захлебывание» получателя в случае «очень быстро говорящего» отправителя. Механизм для этого называется управление потоком.

Уровень сессии

Уровень сессии позволяет пользователям на А-машинах (напомним, что пользователем может быть программа) устанавливать между собой сессии. Сессия позволяет передавать данные, как это может делать транспортный уровень, но, кроме того, этот уровень имеет более сложный сервис, полезный в некоторых приложениях. Один из видов услуг на этом уровне - управление диалогом. Другой вид сервиса на этом уровне - управление маркером. Другим примером сервиса на этом уровне является синхронизация. Уровень сессии позволяет расставлять контрольные точки.

Уровень представления

Уровень представления предоставляет решения для часто возникающих проблем, чем облегчает участь пользователей. В основном это проблемы семантики и синтаксиса передаваемой информации. Данный уровень имеет дело с информацией, а не с потоком битов. Типичным примером услуги на этом уровне является унифицированная кодировка данных.

Уровень приложений

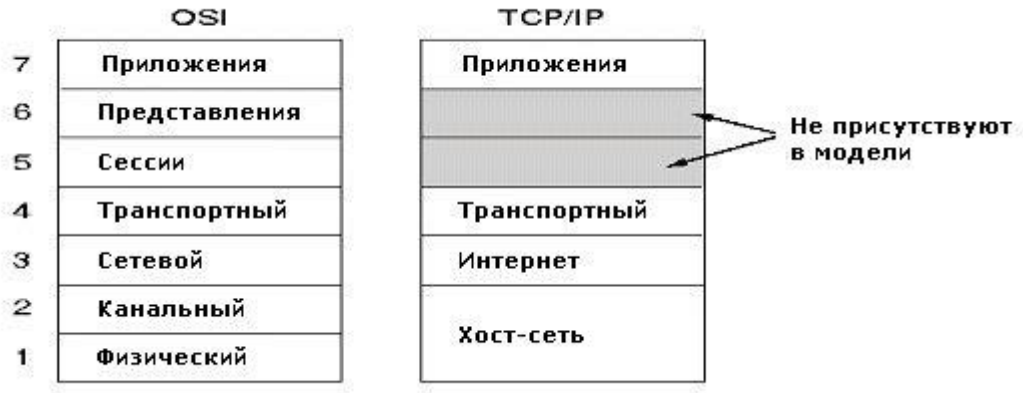
Уровень приложений обеспечивает работу часто используемых протоколов. Существуют сотни разных типов терминалов. Если мы захотим создать сетевой экранный редактор, то нам придется прописывать для каждого типа терминала свою версию. Другой пример - передача файлов.

Эталонная модель TCP/IP

Межсетевой уровень

Итак, назначение межсетевого уровня в TCP/IP - доставить IP-пакет по назначению. Это как раз то, за что отвечает сетевой уровень в МОС-модели. На рисунке 1-15 показано соответствие между уровнями этих двух эталонных моделей.

Рисунок 1-15. Соответствие между МОС и TCP/IP



Транспортный уровень

Над межсетевым уровнем расположен транспортный уровень. Как и в МОС-модели, его задача - обеспечить связь «точка-точка» между двумя равнозначными активностями. В рамках TCP/IP модели было разработано два транспортных протокола. Первый - TCP (Transmission Control Protocol) - надежный протокол с соединением. Он получает поток байт, фрагментирует его на отдельные сообщения и передает их на межсетевой уровень. На машине-получателе равнозначная активность TCP-протокола собирает эти сообщения в поток байтов. TCP-протокол также обеспечивает управление потоком.

Второй протокол - UDP (User Datagram Protocol). Это ненадежный протокол без соединения для тех приложений, которые используют свои механизмы фрагментации и управления потоком. Он часто

используется для передачи коротких сообщений в клиент-серверных приложениях, а также там, где скорость передачи важнее ее точности. Соотношение этих протоколов и их приложений показано на рисунке 1-16.

Уровень приложений

В TCP/IP-модели нет уровней сессии и представления. Необходимость в них была неочевидна для ее создателей. На сегодня дело обстоит так, что разработчик сложного приложения берет проблемы этих уровней на себя. Над транспортным протоколом располагается уровень приложений. Этот уровень включает следующие приложения: виртуальный терминал - TELNET, передачу файлов - FTP, электронную почту - SMTP. Позднее к ним добавились: служба имен домена - DNS (Domain Name Service), отображающая логические имена хост-машин на их сетевые адреса, протокол для передачи новостей - NNTP и протокол для работы с гипертекстовыми документами во Всемирной паутине - HTTP. Под межсетевым уровнем в TCP/IP-модели великая пустота. Модель ничего не говорит, что там происходит, кроме того, что хост-машина должна быть связана с сетью через некоторый протокол. Никаких ограничений на этот протокол, равно как и рекомендаций нет.

3. Примеры систем передачи данных (SMDS, X.25, Frame Realy, ISDN, B-ISDN, ATM) и их сравнение

Напомним, что основной задачей СПД, которая является частью транспортной среды любой сети, является обеспечение среды для передачи данных между А-машинами и коммуникационными машинами в сети. Основные элементы СПД - каналы и коммутаторы.

Сети X.25

Стандарт X.25 используют некоторые телефонные сети, особенно в Европе. Этот стандарт, разработанный МКТТ в 70-х годах, определяет интерфейс между сетью с коммутацией пакетов и терминалом, а также взаимодействие терминалов через сеть передачи данных.

Рекомендации этого стандарта в терминах модели МОС охватывают физический, канальный и сетевой уровни. Они определяют способ передачи цифровых данных по телефонным каналам.

- Протокол X.21 определяет физический, электрический интерфейс и процедуры взаимодействия терминала и сети передачи данных. Сетей, поддерживающих этот стандарт, не так много. Это связано с тем, что он требует использования цифровых сигналов, а не аналоговых. Как временная мера был предложен интерфейс типа RS-232.
- Уровень канала данных отвечает за исправление ошибок на линии.
- Сетевой уровень отвечает за адресацию, управление потоком, подтверждение доставки, прерывания и т.п. внутри СПД
- Пакеты в X.25 имеют длину до 128 байт.
- Обычная скорость - 64 кбит/сек.
- Стандарт ориентирован на соединение и поддерживает режим коммутируемых виртуальных каналов и режим постоянного виртуального канала.
- Поскольку в мире уже много оконечных устройств, не рассчитанных на X.25, то было предложено решение - устройство PAD (Packet Assembler Disassembler), которое работает, как черный ящик. Его работу определяют три протокола X.3, X.28 и X.29.

Frame Relay

Ретрансляция кадров (Frame Relay - FR) - это метод доставки сообщений в сетях передачи данных (СПД) с коммутацией пакетов (в отличие от СПД с коммутацией каналов и сообщений). Первоначально разработка стандарта FR ориентировалась на цифровые сети с интегрированным сервисом (ISDN - Integrated Services Digital Networks), однако позже стало ясно, что FR применим и в других СПД (здесь под данными понимается любое сообщение, представленное в цифровой форме). Эта служба предоставляет минимальный сервис. Если фрейм поступил с ошибкой, то он просто сбрасывается. Дело пользователя - определить, какой фрейм пропущен и как его восстановить. В отличие от X.25, FR не поддерживает уведомления о доставке и

обычного управления потоком. Любой международный стандарт имеет (и всегда будет иметь) множество прикладных реализаций, что зачастую приводит к несовместимости аппаратно-программных средств разных производителей. Международные организации неоднократно пытались решить данную проблему. Результатом одной из таких попыток (предпринятой FRF) стал проект стандарта, включающего в себя спецификации ANSI, которые обязательны для выполнения членами FRF. В январе 1992 г. этот проект был доработан Техническим комитетом FRF и утвержден собранием членов FRF.

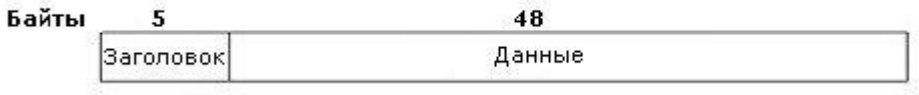
Высокоскоростной ISDN и ATM

Кроме проблем, связанных с быстро растущими требованиями в области сервиса, есть и еще одна - интеграция разных сетей. Например, X.25, SMDS и FR сетей с сетью DQDB. Связывать и обслуживать все это разнообразие сетей - огромная головная боль. А есть еще кабельное телевидение и т.д. и т.д. Выход из этого ада - создать единую сеть, обеспечивающую такую высокую скорость передачи, что она будет способна поддерживать любую услугу. Это нельзя сделать быстро за одну ночь. Это очень масштабный проект и он уже начался.

Этот новый сервис передачи данных называется Broadband ISDN - высокоскоростной ISDN. Этот сервис будет поддерживать передачу видео, аудио и цифровых данных высокого качества, обеспечивать высокоскоростную связь между локальными сетями. Основной технологией, которая делает возможным реализацию сервиса B-ISDN, является ATM (Asynchronous Transfer Mode) - асинхронный способ передачи.

Главная идея ATM - передавать данные малыми порциями, фиксированной длины, называемыми ячейками. Каждая ячейка имеет длину 53 байта - 48 на данные и 5 на заголовок. На рисунке 1-21 показана ATM-ячейка. ATM - это и технология, т.е. невидимая для пользователя сущность, и сервис, т.е. то, что пользователь видит.

Рисунок 1-21. Формат ячейки ATM



Переход от 100-летней технологии коммутации каналов на коммутацию пакетов - это гигантский шаг. Есть много причин, почему данные удобно передавать небольшим пакетами - ячейками:

- Ячейки удобно использовать для управления и передачи разнородных данных - звук, видео, цифра.
- При больших скоростях проще управлять переключением небольших ячеек, чем использовать старую технику мультиплексирования.
- ATM - это технология, ориентированная на соединение: прежде чем передавать данные, устанавливается соединение и лишь потом передаются данные. Доставка данных не гарантируется, но порядок - да.
- ATM-сеть, как любая другая ПД, состоит из каналов и коммутаторов. В настоящее время достигнута скорость 155 Мбит/сек. и 622 Мбит/сек.
- Когда ATM появился, основной областью применения этого сервиса считалось видео по заказу. В настоящее время появились и другие приложения, которые также требуют высокой пропускной способности.

1.9.4. Эталонная модель B-ISDN ATM

Рассмотрим эталонную модель ATM в том виде, как она представлена в области телефонии. Эта модель изображена на рисунке 1-22 в виде куба. Она состоит из трех уровней: физического, ATM и уровня адаптации. Сверху пользователь может поместить любое приложение, например, стек TCP/IP.

Рисунок 1-22. Модель ATM

Физический уровень в ATM определяет правила передачи и приема данных в форме потока битов и преобразования их в ячейки. Носителями этого потока могут быть разные среды. ATM не ограничивает их число.

ATM-уровень отвечает за транспорт ячеек. Он определяет формат ячейки, заголовок, его содержимое, отвечает за установление и поддержание виртуальных соединений. Управление потоком и перегрузками также сосредоточено здесь.

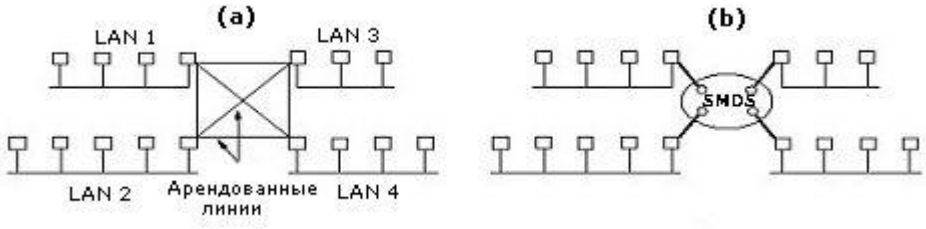
Уровень адаптации (AAL) обеспечивает приложениям-пользователям возможность работы в терминах пакетов или подобных им единиц, а не ячеек.

Плоскость пользователя отвечает за транспорт данных, управление потоком, исправление ошибок и другие функции пользователя. Плоскость управления отвечает за управление соединением. Уровни управления уровнем и плоскостью отвечают за управление ресурсами и координацию межуровневых взаимодействий.

SMDS - Мегабитная система передачи данных с коммутацией

Эта СПД была разработана фирмой Bellcore для тех пользователей, у которых есть несколько LAN-подразделений, территориально разобщенных. Для их соединения либо надо арендовать 6 телефонных линий (рисунок 1-24 (a)), либо поступить так, как показано на рисунке 1-24 (b). В последнем случае надо арендовать четыре короткие линии от LAN до точки подключения к SMDS-сети.

Рисунок 1-24. Соединение LAN через SMDS



Преимущества SMDS следующие:

- Обычные телефонные линии рассчитаны на постоянную загрузку, SMDS-сеть - на взрывную, т.е. большая часть трафика будет сосредоточена в рамках каждой LAN, и лишь иногда пара LAN будет соединяться.
- Такое решение дешевле. Надо платить за n арендуемых линий, а не за n(n-1)/2, как в случае полного соединения обычными линиями.
- Скорость передачи - 45 Мбит/сек.
- Это решение лучше, чем решение через MAN, которое осуществимо лишь в условиях города.
- На рисунке 1-25 показан формат SMDS-пакета. SMDS-служба поддерживает только одну услугу - простую передачу потока пакетов.
- При этом не важно содержимое пакета. Это может быть IP-пакет, IBM маркерный пакет и т.п.

Рисунок 1-25. Формат пакета SMDS

Развитие этой службы идет в направлении вещательной передачи, когда пользователь может определить несколько адресов для доставки пакета. В то же время, если допустить возможность предопределения тех телефонных номеров, от которых можно получать пакеты, пользователи получат прекрасную возможность создания своей индивидуальной сети на основе телефонной службы.

Сравнение СПД

В таблице 1-26 приведены основные данные по каждой из ранее рассмотренных СПД. Может возникнуть вопрос: почему их так много? Все они появились в разное время, под давлением потребностей разных категорий пользователей, разрабатывались разными компаниями из разных областей: телефония, цифровые сети, телевизионные сети.

Таблица 1-26. Возможности разных СПД

Свойство	DQDB	SMDS	X.25	Frame Relay	ATM AAL
Ориентированность на соединение	Есть	Нет	Есть	Есть	Есть
Стандартная скорость передачи (Мбит/сек.)	45	45	0,064	1,5	155
Коммутируемость	Нет	Есть	Есть	Нет	Есть
Фиксируемая нагрузка	Есть	Нет	Нет	Нет	Нет
Максимальная нагрузка	44	9188	128	1600	Переменная
Постоянные виртуальные каналы	Нет	Нет	Да	Да	Да
Групповое вещание	Нет	Да	Нет	Нет	Да

4.Требования, предъявляемые к современным вычислительным сетям. Что такое стандарт на взаимодействие в сетях, кто, как и для чего вводит стандарты?

Главным требованием является обеспечение пользователям доступ к вычислительным сервисам сети. Все остальные требования – производительность, надежность, безопасность, расширяемость и масштабируемость, управляемость, совместимость – характеризуют качество реализации главного требования.

1. **Производительность**

Производительность сети характеризует скорость работы сети. Эта характеристика измеряется в количестве услуг в единицу времени.

2. **Надежность**

Эта характеристика сети определяет, всегда ли сеть способна выполнять операции и, если операция запущена, то всегда ли она корректно завершится. Есть несколько подходов измерения этой характеристики:

- через измерение надежности устройств (время наработки на отказ, вероятность отказа, интенсивность отказов)
- коэффициент готовности – доля времени, в течение которого система может быть использована

- вероятность доставки пакета через ТС
- вероятность искажения пакета в ТС
- отказоустойчивость

3.Безопасность

Характеризует степень защищенности сети от несанкционированного использования и изменения состояния ее ресурсов:

- ТС
- СПД
- Вычислительные ресурсы
- Информация (доступ, изменение)

В случае информации говорят о конфиденциальности данных, когда доступ к данным получает лишь тот, кто имеет на это право, и целостности, когда изменять данные может только тот, кто имеет на это право.

4.Расширяемость и масштабируемость

Расширяемость характеризует то, насколько сложно изменить конфигурацию сети: СПД, добавить новый узел и т.п. Масштабируемость характеризует способность сети плавно увеличивать вычислительную мощность без деградации производительности сети в целом.

5.Прозрачность

Эта характеристика показывает, насколько «просто» пользоваться сетью. Чем сложнее доступ для пользователя к нужному сервису в сети, тем менее прозрачна сеть. В идеале должен быть реализован принцип «Сеть – это компьютер».

- сама распределяет ресурсы и управляет ими
- среда для разработки и выполнения программ
- поставщик сервиса
- для пользователя она прозрачна (он ее не видит)
- концепция метакомпьютера

6.Передача разнородных потоков данных (видео, звук, цифра)

Слияние средств вычислений и средств передачи разнородных данных. Здесь основную сложность представляет синхронность передачи.

7.Управление

Возможность управлять и контролировать работу каждого отдельного устройства в сети из единого центра.

8.Совместимость

Характеризует способность подключать разное оборудование и программное обеспечение.

Кто, как и для чего вводит стандарты

- Функции стандарта:
  - унификация (вспомним Вавилонскую башню)
  - координация
  - защита пользователей
  - защита инвестиций

- Стандарты
  - международные, государственные, отраслевые
  - de jure, de facto
- Международная организация по стандартизации (ISO)
  - Образована в 1946 году, распространена на 89 стран, включая Россию.
  - Имеет 200 технических комитетов, рабочие группы, более 100 000 добровольцев.
  - Этапы стандарта - CD, DIS, IS.
- Международный Союз электросвязи (орган ООН)
  - сектор радио коммуникаций (ITU-R)
  - сектор телекоммуникационной стандартизации (ITU-T)
  - сектор разработок
- Европейская ассоциация производителей компьютеров (ECMA)
- Американский национальный институт стандартов
  - стандартизация языков
  - развитие SNA совместно с IBM
- Министерство обороны США
- Институт инженеров по электротехнике и радиоэлектронике (IEEE)
- Госстандарт
- Техническая комиссия

Кто есть кто в мире стандартов для Интернета

- Интернет-сообщество (ISOC) - развитие инфраструктуры, общие вопросы развития и роста Интернета.
- Совет по архитектуре Internet (IAB) - технический контроль и координация работ по разработке новых стандартов и их реализации.
  - IETF - решение краткосрочных проблем, спецификация предложений для стандартизации
  - IRTF - долгосрочные проблемы, требующие отдельных исследования
- IETF формирует draft стандарта, которому присваивают RFC
  - standard proposal (6 месяцев)
  - standard draft (4 месяца)
  - официальный стандарт Интернета

2. Теоретические основы передачи данных (ограничения на пропускную способность передачи сигналов, взаимосвязь пропускной способности канала и ширины его полосы пропускания). Среды передачи (магнитные носители, витая пара, среднеполосный и широкополосный кабели, оптоволокно, сравнение кабелей и оптоволокна).

Все виды информации могут быть представлены при передаче в виде электромагнитных импульсов. В зависимости от среды передачи и организации СПД могут применяться либо аналоговые, либо цифровые сигналы. Любой сигнал можно рассматривать либо как функцию времени, т.е. то, как различные параметры сигнала изменяются со временем, либо как функцию частоты.

Разные формы представления сигнала

Как уже было сказано, любой сигнал можно рассматривать как функцию времени, либо как функцию частоты. В первом случае эта функция показывает, как меняются со временем параметры сигнала, например, напряжение или сила тока. Если эта функция имеет непрерывный характер, то говорят о непрерывном сигнале. Если эта функция имеет дискретный вид, то говорят о дискретном сигнале

Частотное представление функции основано на том факте, что любая функция от вещественной переменной может быть представлена в виде ряда Фурье:

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

где f =1/T - частота, a_n, b_n - амплитуды n-ой гармоники.

Ясно, что на практике нельзя учесть бесконечно много гармоник. Все их учитывать и не надо потому, что энергия сигнала распределяется не равномерно между гармониками разной частоты. В общем случае соотношение здесь таково, что низкочастотные составляющие несут большую часть энергии. Ни в какой среде сигнал не может передаваться без потери энергии. С ростом частоты искажения растут. Любая среда передачи ограничивает максимальную частоту передаваемого сигнала, а следовательно, и частоту гармоник, которые можно использовать для аппроксимации функции g(t). Тем самым аппроксимация (точность воспроизведения формы) сигнала ухудшается и скорость передачи понижается. Характеристику канала, определяющую спектр частот, которые физическая среда канала пропускает без существенного понижения мощности сигнала, называют **полосой пропускания** канала.

Сигналы, данные, передача

Важно различать три основные понятия:

- Данные
- Сигнал
- Передача

Данные – это то, с помощью чего мы описываем явление или объект. Сигнал – это представление данных. Передача – это процесс взаимодействия передатчика и приемника с целью получения приемником сигналов от передатчика. Большое значение также имеет количество уровней, которое может иметь сигнал. Чем больше число уровней сигнала, тем больше информации можно передать за один переход с уровня на уровеньПроцесс передачи также может иметь аналоговую или цифровую формы. Аналоговая передача предполагает непрерывное изменение параметров передачи. Цифровая передача - резкое, дискретное изменение параметров передаваемого сигнала или импульса

Сигнал в цифровой форме нельзя напрямую передавать с помощью аналоговой передачи или, как ее еще называют, аналоговой модуляции, в то время как цифровое кодирование или цифровая передача позволяет передавать оба вида сигнала.

Взаимосвязь пропускной способности канала и его полосы пропускания

Максимальную скорость, с которой канал способен передавать сигнал, называют **пропускной способностью канала.**

Теорема Найквиста

max data rate = $2H \log_2 V$ ит/сек,

где H – ширина полосы пропускания канала, выраженная в Гц, V - количество уровней в сигнале. Эта теорема также показывает, что, например, бессмысленно сканировать линию чаще, чем удвоенная ширина полосы пропускания. Действительно, все частоты выше этой отсутствуют в сигнале. Однако теорема Найквиста не учитывает шум в канале, который измеряется как отношение мощности полезного сигнала к мощности шума: S/N. Эта величина измеряется в децибелах: $10 \log_{10}(S/N)$ dB. Например, если отношение S/N равно 10, то говорят о шуме в 10 dB, если отношение равно 100, то - 20 dB. На случай канала с шумом есть теорема Шеннона, по которой максимальная скорость передачи по каналу с шумом равна

$H \log_2 (1+S/N)$ бит/сек.,

где S/N - соотношение сигнал-шум в канале.

Как мы уже отмечали в разделе 2.1.2, скорость передачи данных зависит от способа представления данных на физическом уровне и сигнальной скорости, или скорости модуляции - скорости изменения значения сигнала. Скорость изменений сигнала в секунду измеряется в единицах, называемых бот. Если скорость изменения значения сигнала b бот, то это не означает, что данные передается со скоростью b бит/сек. Многое зависит способа кодирования сигнала: одно изменение значения может кодировать сразу несколько бит. Если используется 8 значений (уровней) сигнала, то каждое изменение его значения кодирует сразу 3 бита. Если используется только два значения сигнала, то скорость в битах равна скорости в ботах.

Среды передачи

Назначение физического уровня - передавать данные в виде потока бит от одной машины к другой. Для передачи можно использовать разные физические среды. Каждую из них характеризуют следующими параметрами:

- полоса пропускания
- пропускная способность
- задержка
- стоимость
- простота прокладки
- сложность в обслуживании

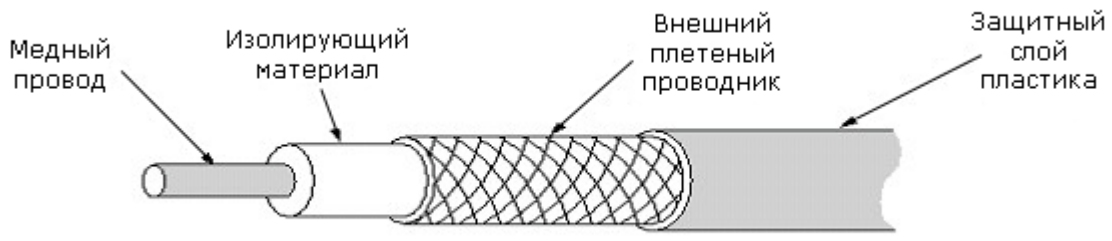
Магнитные носители - *Магнитная лента или магнитный диск*

Витая пара

Самой старой и все еще используемой средой передачи со времен появления телефона является витая пара. Витая пара состоит из двух медных изолированных проводов, один из которых обвит вокруг другого. Этот второй, вьющийся провод предназначен для устранения взаимного влияния между соседними витыми парами. Витая пара широко используется в телефонии. Между абонентами и АТС линии из витой пары могут иметь протяженность до нескольких километров без промежуточного усиления. Например, в России в городских условиях средняя длина абонентской линии равна 1,5 км. Витые пары объединяются в многопарные кабели. Витая пара может быть использована для передачи как цифровых, так и аналоговых сигналов. Ее пропускная способность зависит от толщины используемых проводов и расстояния. Наиболее часто используемыми являются кабели категории 3 и категории 5. Кабель категории 3 содержит по четыре витые пары с невысокой плотностью навивки и имеет полосу пропускания до 16 МГц. Кабель категории 5 имеет тоже четыре пары, но с более плотной навивкой, что позволяет достичь более высоких скоростей, и имеет полосу пропускания 100 МГц.

Коаксиальные кабели

Из этого рисунка видно, что коаксиальные кабели работают на частотах от 1 МГц до 500 МГц. Поэтому эти кабели применяют на больших расстояниях и по ним могут передаваться одновременно несколько потоков данных от разных компьютеров.



- передача телевизионных сигналов, включая системы кабельного телевидения

- передача нескольких телефонных разговоров одновременно на большие расстояния между телефонными станциями, построение ЛВС
- подключение компьютерных периферийных устройств на небольших расстояниях

Коаксиальные кабели используют для передачи как аналоговых, так и цифровых сигналов. У них шире полоса пропускания, а следовательно, выше скорость передачи данных. Основными ограничителями скорости и расстояния при передаче без усиления являются в этих кабелях затухание сигнала, тепловой шум и интермодуляционный шум. Последний вид шума возникает, когда всю полосу пропускания кабеля разбивают на более узкие полосы и каждую такую полосу используют как отдельный канал. Интермодуляционный шум возникает на границах таких каналов. Есть два основных вида коаксиальных кабелей: узкополосный с волновым сопротивлением 50 Ом и широкополосный с волновым сопротивлением 75 Ом. Узкополосный кабель позволяет достигать скорости в несколько Гбит/сек при длине в 1-2 км и высокой помехозащищенности. При большем расстоянии нужны промежуточные усилители. Существенное различие между узкополосным кабелем и широкополосным в том, что широкополосный кабель применяется для передачи аналоговых сигналов на больших расстояниях и, следовательно, требует промежуточных аналоговых усилителей. Эти промежуточные усилители пропускают сигналы только в одном направлении. Поэтому машина, получившая поток битов, не может использовать для ответа тот же путь, по которому поток битов к ней пришел. Для решения этой проблемы есть два вида систем: двухкабельные и однокабельные системы. В двухкабельных системах прокладывается сразу два кабеля: один кабель используется для входящего потока, а второй для исходящего. Компьютер соединен этими кабелями со специальной головной станцией, которая перебрасывает трафик с одного кабеля на другой, идущий в нужном направлении. В однокабельных системах полоса частот разделяется между входящим и исходящим трафиками. Например, полоса от 5 до 30 МГц служит для входного трафика, а полоса от 40 до 300 МГц – для выходного. Эта граница в каждой стране устанавливается своя. Низкая полоса частот используется для передачи данных от компьютера к головной станции, которая сдвигает их в сторону высоких частот и передает на другие компьютеры. Коаксиальные кабели активно используют в системах кабельного телевидения

Оптоволокно

Для использования оптической связи нужен источник света, светопроводящая среда и детектор, преобразующий световой поток в электрический. На одном конце волоконнооптической линии находится передатчик - источник света, световой импульс от этого источника проходит по светопроводящему волокну и попадает на детектор, который расположен на другом конце этой линии и преобразует этот импульс в электрический. Одна из основных проблем создания оптоволоконных систем состояла в том, чтобы не дать световому пучку рассеяться через боковую поверхность силиконового шнура. Поскольку можно испускать несколько лучей разной длины волны так, чтобы они попадали на границы шнура под углом, большим угла полного внутреннего отражения, то по одному шнуру можно пускать несколько лучей. Каждый луч, как говорят, имеет свою моду. Так получается многомодовый шнур.

Прохождение света через оптоволокно

Оптоволокно делают из стеклоподобного материала, который, в свою очередь, производят из песка и других широко распространенных материалов. Затухание оптического сигнала в стекле зависит от длины волны источника света. На рисунке 2-20 показана зависимость затухания от длины волны. Затухание измеряется в dB по следующей формуле:

$$10 \log_{10} \frac{Tp}{Rp}$$

где Tp – мощность передаваемого сигнала, Rp – мощность полученного сигнала

Другую проблему при использовании оптоволокна представляет дисперсия: исходный световой импульс по мере распространения теряет начальную форму и размеры. Это явление называется дисперсией. Величина

этих искажений также зависит от длины волны. Одно из возможных решений - увеличить расстояние между соседними сигналами. Однако это сократит скорость передачи. К счастью, исследования показали, что если придать сигналу некоторую специальную форму, то дисперсионные эффекты почти исчезают и сигнал можно передавать на тысячи километров. Сигналы в этой специальной форме называются силитонами.

2.3.4.2. Оптоволоконный кабель.

Такой кабель прокладывают и под землей, где он нередко становится жертвой экскаваторов и другой землеройной техники, и под водой, где он становится добычей тралов и акул. Соединяют его электрически с помощью специальных коннекторов, механически, прижимая один край к другому, либо сваривая воедино оба конца. Все эти манипуляции приводят к потере от 5 до 20% мощности сигнала в точке соединения.

Используются два вида источников света: светодиод (LED) и полупроводниковый лазер. У них разные свойства, которые показаны в таблице 2-22. С помощью специальных интерферометров эти источники света можно настроить на нужную длину волны. На принимающем конце стоит фотодиод, время срабатывания которого равно 1 нсек., что ограничивает максимальную скорость передачи до 1 Гбит/сек.

Оптоволоконные сети

С помощью оптоволокна можно строить как LAN, так и сети большего масштаба. Подключение к оптоволоконной сети более сложное, чем к Ethernet-сети. Чтобы понять, как решается проблема построения сети из оптоволокна, надо осознать, что сеть типа «кольцо» представляет из себя цепочку соединений типа «точка-точка. Такие соединения могут быть двух видов: пассивное и активное. У пассивного есть светодиод либо лазер, и фотодиод. Принимая сигнал через фотодиод, это соединение передает электрический сигнал компьютеру или транслирует его дальше с помощью светодиода или лазера. Это абсолютно надежное соединение. Выход из строя любого из компонентов не нарушает связь по кольцу, а лишь блокирует работу отдельного компьютера. Активное подключение содержит промежуточный усилитель электрического сигнала. Фотодиод преобразует оптический сигнал в электрический. Этот сигнал усиливается, передается компьютеру либо транслируется дальше с помощью лазера или светодиода. Кроме кольца, возможны соединения типа пассивной звезды. Все линии, по которым оптический сигнал передается от компьютера, заходят в специальное устройство пассивной звезды, сигналы от них воспринимаются по всем линиям, исходящим из этого устройства и передают к надлежащим приемникам.

Сравнение возможностей медного кабеля и оптоволокна

В заключение сравним возможности медного кабеля и оптоволокна:

1. Ширина полосы пропускания у оптоволокна несравненно больше, чем у медного кабеля, что позволяет достичь скорости в сотни Гбит/сек на расстояниях в десятки километров. Напомним, что коаксиал дает скорость максимум в несколько сотен Мбит/сек. примерно на 1 километре. Витая пара дает несколько Мбит/сек. на 1 километр и из нее можно выжать до 1 Гбит/сек. на расстоянии до 100 м.

2. Оптоволокно компактнее и меньше весит. При той же пропускной способности коаксиальный кабель и кабель из витых пар существенно тяжелее оптоволокна. Это существенный фактор, влияющий на стоимость и требования к опорным конструкциям. Например, 1 км 1000-парника весит 8 тонн, а оптоволокно аналогичной пропускной способности – 100 кг.

3. Затухание сигнала в оптоволокне существенно меньше, чем в коаксиале и витой паре, и остается постоянным для широкого диапазона частот.

4. Оптоволокно не восприимчиво к внешним электромагнитным излучениям. Поэтому ему не страшны интерференция, импульсные шумы и взаимные наводки. Оптоволокно не излучает энергию, поэтому не влияет на работу другого оборудования. Его трудно обнаружить, следовательно найти и повредить.

5. Чем меньше репитеров, тем дешевле система и меньше источников ошибок. С этой точки зрения оптоволоконные системы достигли большего совершенства. Для этих систем среднее расстояние между репитерами – сотни километров. Для коаксиала или витой пары тот же показатель равен нескольким километрам.

	Диапазон частот	Стандартное затухание	Стандартная задержка	Расстояние между репитерами
Витая пара	0-3,5 кГц	0,2 дБ при 1 кГц	50 мсек./км	2 км
Многопарный кабель	0-1 МГц	3 дБ/км при 1 кГц	5 мсек./км	2 км
Коаксиал	0-500 МГц	7 дБ/км	5 мсек./км	1-9 км
Оптический кабель	180-370 ТГц	0,2-0,5 дБ/км	5 мсек./км	40 км

6. Теоретические основы передачи данных (ограничения на пропускную способность передачи сигналов, взаимосвязь пропускной способности канала и ширины его полосы пропускания). Передача цифровых данных цифровыми сигналами.

Все виды информации могут быть представлены при передаче в виде электромагнитных импульсов. В зависимости от среды передачи и организации СПД могут применяться либо аналоговые, либо цифровые сигналы (подробно об этом см. раздел 2.1.2).

Любой сигнал можно рассматривать либо как функцию времени, т.е. то, как различные параметры сигнала изменяются со временем, либо как функцию частоты.

Разные формы представления сигнала

Как уже было сказано, любой сигнал можно рассматривать как функцию времени, либо как функцию частоты. В первом случае эта функция показывает, как меняются со временем параметры сигнала, например, напряжение или сила тока. Если эта функция имеет непрерывный характер, то говорят о непрерывном сигнале. Если эта функция имеет дискретный вид, то говорят о дискретном сигнале

Частотное представление функции основано на том факте, что любая функция от вещественной переменной может быть представлена в виде ряда Фурье:

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

где f =1/T - частота, a_n, b_n - амплитуды n-ой гармоники.

Ясно, что на практике нельзя учесть бесконечно много гармоник. Все их учитывать и не надо потому, что энергия сигнала распределяется не равномерно между гармониками разной частоты. В общем случае соотношение здесь таково, что низкочастотные составляющие несут большую часть энергии. Ни в какой среде сигнал не может передаваться без потери энергии. С ростом частоты искажения растут. Любая среда передачи ограничивает максимальную частоту передаваемого сигнала, а следовательно, и частоту гармоник, которые можно использовать для аппроксимации функции g(t). Тем самым аппроксимация (точность воспроизведения формы) сигнала ухудшается и скорость передачи понижается. Характеристику канала, определяющую спектр частот, которые физическая среда канала пропускает без существенного понижения мощности сигнала, называют **полосой пропускания** канала.

Взаимосвязь пропускной способности канала и его полосы пропускания

Максимальную скорость, с которой канал способен передавать сигнал, называют **пропускной способностью канала.**

Теорема Найквиста

max data rate = 2H $^{\log_2 V}$ ит/сек,

где H – ширина полосы пропускания канала, выраженная в Гц, V - количество уровней в сигнале. Эта теорема также показывает, что, например, бессмысленно сканировать линию чаще, чем удвоенная ширина полосы пропускания. Действительно, все частоты выше этой отсутствуют в сигнале. Однако теорема Найквиста не учитывает шум в канале, который измеряется как отношение мощности полезного сигнала к мощности шума: S/N. Эта величина измеряется в децибелах: 10 $\log_{10}$(S/N) dB. Например, если отношение S/N равно 10, то говорят о шуме в 10 dB, если отношение равно 100, то - 20 dB. На случай канала с шумом есть теорема Шеннона, по которой максимальная скорость передачи по каналу с шумом равна

H $\log_2$ (1+S/N) бит/сек., где S/N - соотношение сигнал-шум в канале.

Как мы уже отмечали в разделе 2.1.2, скорость передачи данных зависит от способа представления данных на физическом уровне и сигнальной скорости, или скорости модуляции - скорости изменения значения сигнала. Скорость изменений сигнала в секунду измеряется в единицах, называемых бот. Если скорость изменения значения сигнала b бот, то это не означает, что данные передается со скоростью b бит/сек. Многое зависит способа кодирования сигнала: одно изменение значения может кодировать сразу несколько бит. Если используется 8 значений (уровней) сигнала, то каждое изменение его значения кодирует сразу 3 бита. Если используется только два значения сигнала, то скорость в битах равна скорости в ботах.

> Цифровые данные – цифровой сигнал. Оборудование для преобразования данных в цифровой форме в цифровой сигнал дешевле и проще, чем оборудование для преобразования данных в аналоговой форме в цифровой сигнал.

Цифровые данные – Цифровые сигналы

Цифровой сигнал – это дискретная последовательность импульсов по напряжению, каждый из которых имеет ступенчатую форму. Каждый импульс – это единичный сигнал. В общем случае данные в двоичной форме при передаче кодируются так, что один бит данных может быть отображен в несколько единичных сигналов. В простейшем случае это соответствие имеет однозначный характер: один бит – один единичный сигнал. В примерах, приведенных в предыдущих разделах, мы как раз встречали именно этот простейший случай, когда двоичная 1 была представлена высоким потенциалом, а двоичный 0 – низким. В этом разделе мы рассмотрим разные схемы кодирования данных на физическом уровне. **Скорость передачи данных** – это количество бит в секунду, которые передают с помощью сигналов. Эту скорость также называют **битовой скоростью. Продолжительность (длина) бита** – это интервал времени, которое нужно передатчику, чтобы испустить последовательность надлежащих единичных сигналов. При скорости передачи данных R бит/сек, длина бита равна 1/R. Напомним, что скорость модуляции или сигнальная скорость измеряется в бот – это скорость изменения уровня сигнала. Очень многое зависит от способа кодировки данных. Теперь

рассмотрим, какие задачи должен решать приемник при передаче. Прежде всего, приемник должен быть точно настроен на длину бита. Он должен уметь распознавать начало и конец передачи каждого бита, а также уровень сигнала: низкий или высокий.

Основными критериями сравнения различных способов кодирования данных на физическом уровне являются:

- Ширина спектра сигнала. Чем шире спектр, тем сильнее искажения.
- Синхронизация между приемником и передатчиком. Мы уже отмечали необходимость для приемника точно определять начало и конец битового интервала.
- Обнаружение ошибок. Хотя методы обнаружения и исправления ошибок располагаются на канальном уровне, который находится над физическим уровнем, тем не менее, и на физическом уровне весьма полезно иметь такие возможности.
- Чувствительность к шуму.
- Стоимость и скорость.

Все схемы кодирования делятся на потенциальные и импульсные. У потенциальных кодов значение бита передается удержанием потенциала сигнала на определенном уровне в течение битового интервала. У импульсных кодов это значение передается перепадом (фронтом) уровня сигнала. Направление перепада с низкого на высокий или с высокого на низкий уровень определяет значение бита.

Потенциальный NRZ-код

в потенциальной схеме кодирования NRZ (NRZ – Non return to zero – без возврата к нулю на битовом интервале) логическому 0 и логической 1 сопоставлены два устойчиво различаемых потенциала. К достоинствам этого кода следует отнести простоту реализации, устойчивость к ошибкам, достаточно узкий частотный спектр сигнала. Основным недостатком этого кода является отсутствие синхронизации. На длинных последовательностях нулей или единиц, т.е. когда потенциал на линии не меняется, может произойти рассинхронизация между приемником и передатчиком, что приведет к ошибкам. Однако если исключить возможность появления длинных последовательностей 0 или 1, то этот метод может быть весьма эффективен. Обеспечить отсутствие таких последовательностей могут специальные устройства, называемые скремблеры. Модификацией NRZ-кода и хорошим примером дифференциального кодирования является код NRZ-I. Идея дифференциальных кодов состоит в том, чтобы кодировать не абсолютное значение текущего бита, а разницу значений между предыдущим битом и текущим. В случае кода NRZ-I если текущий бит – 0, то он кодируется тем же потенциалом, что и предыдущий бит, если текущий бит – 1, то он кодируется другим потенциалом, чем предыдущий. Основным достоинством этого кода по отношению NRZ-коду является большая устойчивость к шуму.

Биполярный код AMI

Другим примером потенциального кода является метод биполярного кодирования с альтернативной инверсией (Bipolar Alternate Mark Inversion – AMI). В этом методе используются не два уровня сигналов, как в NRZ-методах, а три: положительный, ноль и отрицательный. Значению 0 соответствует нулевой потенциал на линии; значению 1 - либо положительный, либо отрицательный потенциал. При этом потенциал каждой последующей единицы противоположен потенциалу предыдущей. У этого метода есть несколько существенных преимуществ по сравнению с NRZ-кодами. Во-первых, в случае длительной последовательности единиц рассинхронизации не происходит. Каждая единица сопровождается изменением потенциала, устойчиво распознаваемым приемником. Поскольку каждая единица сопровождается изменением потенциала, то не возникнет постоянной составляющей. Однако длинная последовательность 0 остается проблемой, и требуются дополнительные усилия, которые позволили бы избежать ее появления. Во-вторых, спектр сигнала здесь уже, чем у NRZ-кодов. И, наконец, свойство чередования уровней позволяет обнаруживать единичные ошибки.

Биполярные импульсные коды

Существует другая группа методов кодирования, известная как биполярное импульсное кодирование. Манчестерский и дифференциальный Манчестерский коды. В Манчестерском коде данные кодируются фронтами в середине битового интервала. Этим достигаются две цели: синхронизация приемника и передатчика, и передача данных: фронт перехода от низкого потенциала к высокому соответствует 1, а фронт перехода от высокого потенциала к низкому – 0. Этот код показан на рисунке 2-6.

В дифференциальном Манчестерском коде сигнал может менять свой уровень дважды в течение битового интервала. В середине интервала обязательно происходит изменение уровня. Этот перепад используется для синхронизации. При передаче 0 в начале битового интервала происходит перепад уровней, при 1 – такой перепад отсутствует. Все биполярные импульсные методы требуют от одного до двух перепадов уровня сигнала за один битовый интервал. Поэтому их сигнальная скорость в два раза выше, чем у потенциальных кодов. Это означает, что они требуют более широкой полосы пропускания, чем потенциальные коды. Однако у них есть несколько существенных преимуществ:

- самосинхронизация
- отсутствие постоянной составляющей
- обнаружение единичных ошибок

Потенциальный код 2B1Q

В этом методе каждые два последовательных бита (2B) передаются за один битовый интервал сигнала, который может иметь четыре состояния (1Q). Паре 00 соответствует потенциал -2.5 В, 01 соответствует -0.833 В, 11 – +0.833 В, 10 – +2.5 В. У этого метода сигнальная скорость в два раза ниже, чем у кодов NRZ и AMI, а спектр сигнала в два раза уже. Поэтому с помощью 2B1Q-кода можно по одной и той же линии передавать данные в два раза быстрее. Однако реализация этого метода требует более мощного передатчика и более сложного приемника, который должен различать не два уровня, а четыре.

Сигнальная скорость

Здесь мы рассмотрим, как тот или иной метод кодирования влияет на скорость передачи данных (битовую скорость) и сигнальную скорость.

В общем случае D = R/b,

где D – сигнальная скорость R – битовая скорость в бит/сек. b – количество бит на единичный сигнал

8.Теоретические основы передачи данных (ограничения на пропускную способность передачи сигналов, взаимосвязь пропускной способности канала и ширины его полосы пропускания). Передача цифровых данных аналоговыми сигналами

Все виды информации могут быть представлены при передаче в виде электромагнитных импульсов. В зависимости от среды передачи и организации СПД могут применяться либо аналоговые, либо цифровые сигналы. Любой сигнал можно рассматривать либо как функцию времени, т.е. то, как различные параметры сигнала изменяются со временем, либо как функцию частоты.

Разные формы представления сигнала

Как уже было сказано, любой сигнал можно рассматривать как функцию времени, либо как функцию частоты. В первом случае эта функция показывает, как меняются со временем параметры сигнала, например, напряжение или сила тока. Если эта функция имеет непрерывный характер, то говорят о непрерывном сигнале. Если эта функция имеет дискретный вид, то говорят о дискретном сигнале

Частотное представление функции основано на том факте, что любая функция от вещественной переменной может быть представлена в виде ряда Фурье:

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

где f =1/T - частота, a_n, b_n - амплитуды n-ой гармоники.

Ясно, что на практике нельзя учесть бесконечно много гармоник. Все их учитывать и не надо потому, что энергия сигнала распределяется не равномерно между гармониками разной частоты. В общем случае соотношение здесь таково, что низкочастотные составляющие несут большую часть энергии. Ни в какой среде сигнал не может передаваться без потери энергии. С ростом частоты искажения растут. Любая среда передачи ограничивает максимальную частоту передаваемого сигнала, а следовательно, и частоту гармоник, которые можно использовать для аппроксимации функции g(t). Тем самым аппроксимация (точность воспроизведения формы) сигнала ухудшается и скорость передачи понижается. Характеристику канала, определяющую спектр частот, которые физическая среда канала пропускает без существенного понижения мощности сигнала, называют **полосой пропускания** канала.

Взаимосвязь пропускной способности канала и его полосы пропускания

Максимальную скорость, с которой канал способен передавать сигнал, называют **пропускной способностью канала.**

Теорема Найквиста

max data rate = 2H $\log_2 V$ ит/сек,

где H – ширина полосы пропускания канала, выраженная в Гц, V - количество уровней в сигнале. Эта теорема также показывает, что, например, бессмысленно сканировать линию чаще, чем удвоенная ширина полосы пропускания. Действительно, все частоты выше этой отсутствуют в сигнале.

Однако теорема Найквиста не учитывает шум в канале, который измеряется как отношение мощности полезного сигнала к мощности шума: S/N. Эта величина измеряется в децибелах: 10 $\log_{10}$(S/N) dB. Например, если отношение S/N равно 10, то говорят о шуме в 10 dB, если отношение равно 100, то - 20 dB. На случай канала с шумом есть теорема Шеннона, по которой максимальная скорость передачи по каналу с шумом равна

H $\log_2$ (1+S/N) бит/сек., где S/N - соотношение сигнал-шум в канале.

Скорость передачи данных зависит от способа представления данных на физическом уровне и сигнальной скорости, или скорости модуляции - скорости изменения значения сигнала. Скорость изменений сигнала в секунду измеряется в единицах, называемых бот. Если скорость изменения значения сигнала b бот, то это не означает, что данные передается со скоростью b бит/сек. Многое зависит способа кодирования сигнала: одно изменение значения может кодировать сразу несколько бит. Если используется 8 значений (уровней) сигнала, то каждое изменение его значения кодирует сразу 3 бита. Если используется только два значения сигнала, то скорость в битах равна скорости в ботах.

- Цифровые данные – аналоговый сигнал. Некоторые физические среды передачи, например, оптоволокно, электромагнитные поля могут передавать сигналы только в аналоговой форме.

Цифровые данные – Аналоговый сигнал

Теперь мы рассмотрим передачу данных в цифровой форме с помощью аналоговых сигналов. Широко известным примером такой передачи является использование телефонных сетей для передачи цифровых данных. Телефонные сети (их устройство и принципы функционирования мы рассмотрим в разделе 2.5) были

созданы для передачи и коммутации аналоговых сигналов в голосовом диапазоне частот от 300 до 3400 Гц. Этот диапазон не совсем подходит для передачи цифровых данных. Поэтому подключить источник таких данных напрямую в телефонную сеть нельзя. Для этого используют специальное устройство - модем (МОдулятор–ДЕМодулятор). Этот прибор преобразует как цифровой сигнал в аналоговый в надлежащем диапазоне частот, так и наоборот: из аналоговой формы в цифровую. В этом разделе мы познакомимся с основными принципами такого преобразования.

Как мы уже отмечали, аналоговая модуляция заключается в управляемом изменении одного или нескольких основных параметров несущего сигнала: амплитуды, частоты и фазы.

- амплитудная модуляция
- частотная модуляция
- фазовая модуляция

Во всех этих случаях спектр гармоник получаемого сигнала сконцентрирован в области частоты несущего сигнала. В случае амплитудной модуляции двоичные 0 и 1 представлены аналоговым сигналом на частоте несущей, но разной амплитуды. Обычно 0 соответствует сигнал с нулевой амплитудой. Таким образом, при амплитудной модуляции сигнал S(t) имеет вид:

$$S(t) = \begin{cases} A\cos(2\pi f_c t) & двоичная - 1 \\ 0 & двоичный - 0 \end{cases}$$

где $A\cos(2\pi f_c t)$ - несущий сигнал с амплитудой A. Метод амплитудной модуляции не очень эффективен по сравнению с другими методами, т.к. он очень чувствителен к шумам. Чаще всего он используется в сочетании с другими видами модуляции. В чистом виде он применяется на телефонной линии на скоростях до 1200 бит/сек., а также для передачи сигналов по оптоволоконным каналам.

При частотной модуляции двоичные 0 и 1 представляют сигналами разной частоты, сдвинутой, как правило, по отношению к частоте несущей на одинаковую величину, но в противоположном направлении:

$$S(t) = \begin{cases} A\cos(2\pi f_1 t) & для - 1 \\ A\cos(2\pi f_2 t) & для - 0 \end{cases}$$

где $f_c = f_1 - \Delta = f_2 + \Delta$, где Δ - сдвиг по частоте.

Частотная модуляция менее чувствительна к шумам, чем амплитудная. Фазовая модуляция состоит в представлении цифровых данных сдвигом фазы несущего сигнала. Для дифференциальной фазовой модуляции получаем:

$$S(t) = \begin{cases} A\cos(2\pi f_c t + \pi) & для - 1 \\ A\cos(2\pi f_c t) & для - 0 \end{cases}$$

$$\begin{cases} A\cos(2\pi f_c t + \frac{\pi}{4}) & для \quad 11 \\ A\cos(2\pi f_c t + \frac{3\pi}{4}) & для \quad 10 \\ A\cos(2\pi f_c t + \frac{5\pi}{4}) & для \quad 00 \\ A\cos(2\pi f_c t + \frac{7\pi}{4}) & для \quad 01 \end{cases}$$

Эффективность использования полосы пропускания можно существенно повысить, если единичный сигнал будет кодировать несколько бит.

S(t) =

Наш пример хорошо иллюстрирует различие битовой скорости R бит/сек. и скорости модуляции D бит. Предположим, что последняя схема с 12 фазовыми углами применяется, когда на вход подаются данные, закодированные с помощью NRZ-кода. Битовая скорость R=1/tb, где tb – длина бита в NRZ-коде. Однако на выходе закодированный единичный сигнал будет нести b=4 бита, используя L=16 различных комбинаций фазы и амплитуды. Поэтому скорость модуляции будет R/4. Это означает, что при скорости модуляции в 2400 бит битовая скорость будет 9600 бит/сек. В общем случае:

$$D = \frac{R}{b} = \frac{R}{\log_2 L}$$

где D – скорость модуляции (сигнальная скорость) R – битовая скорость (скорость передачи данных)

L – число разных уровней единичных сигналов .b – число бит на единичный сигнал

7. **Теоретические основы передачи данных (ограничения на пропускную способность передачи сигналов, взаимосвязь пропускной способности канала и ширины его полосы пропускания). Передача аналоговых данных цифровыми сигналами.**

Все виды информации могут быть представлены при передаче в виде электромагнитных импульсов. В зависимости от среды передачи и организации СПД могут применяться либо аналоговые, либо цифровые сигналы. Любой сигнал можно рассматривать либо как функцию времени, т.е. то, как различные параметры сигнала изменяются со временем, либо как функцию частоты.

Разные формы представления сигнала

Как уже было сказано, любой сигнал можно рассматривать как функцию времени, либо как функцию частоты. В первом случае эта функция показывает, как меняются со временем параметры сигнала, например, напряжение или сила тока. Если эта функция имеет непрерывный характер, то говорят о непрерывном сигнале. Если эта функция имеет дискретный вид, то говорят о дискретном сигнале. Частотное представление функции основано на том факте, что любая функция от вещественной переменной может быть представлена в виде ряда Фурье:

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

где f =1/T - частота, a_n, b_n - амплитуды n-ой гармоники.

Ясно, что на практике нельзя учесть бесконечно много гармоник. Все их учитывать и не надо потому, что энергия сигнала распределяется не равномерно между гармониками разной частоты. В общем случае соотношение здесь таково, что низкочастотные составляющие несут большую часть энергии. Ни в какой среде сигнал не может передаваться без потери энергии. С ростом частоты искажения растут. Любая среда передачи ограничивает максимальную частоту передаваемого сигнала, а следовательно, и частоту гармоник, которые можно использовать для аппроксимации функции g(t). Тем самым аппроксимация (точность воспроизведения формы) сигнала ухудшается и скорость передачи понижается. Характеристику канала, определяющую спектр частот, которые физическая среда канала пропускает без существенного понижения мощности сигнала, называют **полосой пропускания** канала.

Взаимосвязь пропускной способности канала и его полосы пропускания

Максимальную скорость, с которой канал способен передавать сигнал, называют **пропускной способностью канала.**

Теорема Найквиста

max data rate = 2H $\log_2 V$ ит/сек,

где H – ширина полосы пропускания канала, выраженная в Гц, V - количество уровней в сигнале. Эта теорема также показывает, что, например, бессмысленно сканировать линию чаще, чем удвоенная ширина полосы пропускания. Действительно, все частоты выше этой отсутствуют в сигнале.

Однако теорема Найквиста не учитывает шум в канале, который измеряется как отношение мощности полезного сигнала к мощности шума: S/N. Эта величина измеряется в децибелах: 10 $\log_{10}$(S/N) dB. Например, если отношение S/N равно 10, то говорят о шуме в 10 dB, если отношение равно 100, то - 20 dB. На случай канала с шумом есть теорема Шеннона, по которой максимальная скорость передачи по каналу с шумом равна

H $\log_2$ (1+S/N) бит/сек., где S/N - соотношение сигнал-шум в канале.

Скорость передачи данных зависит от способа представления данных на физическом уровне и сигнальной скорости, или скорости модуляции - скорости изменения значения сигнала. Скорость изменений сигнала в секунду измеряется в единицах, называемых бот. Если скорость изменения значения сигнала b бот, то это не означает, что данные передается со скоростью b бит/сек. Многое зависит способа кодирования сигнала: одно изменение значения может кодировать сразу несколько бит. Если используется 8 значений (уровней) сигнала, то каждое изменение его значения кодирует сразу 3 бита. Если используется только два значения сигнала, то скорость в битах равна скорости в ботах.

- Аналоговые данные – цифровой сигнал. Использование сигнала в цифровой форме позволяет применять современные средства цифровой передачи, достоинства которой перед аналоговой отмечались выше.

Аналоговые данные – Цифровой сигнал

Преобразование аналоговых данных в цифровой сигнал можно представить как преобразование аналоговых данных в цифровую форму. Устройство АЦП (аналогово-цифровой преобразователь) превращает аналоговые данные в цифровую форму, а устройство ЦАП (цифро-аналоговый преобразователь) выполняет обратную процедуру. Устройство, объединяющее в себе функции и АЦП, и ЦАП, называют кодеком (кодер-декодер). мы рассмотрим два основных метода преобразования аналогового сигнала в цифровую форму: импульсно-кодовую модуляцию и дельта-модуляцию.

Импульсно-кодовая модуляция . Импульсно-кодовая модуляция (ИКМ) основана на следствии из теоремы Найквиста, которое утверждает, что если измерять параметры сигнала f(t) через регулярные интервалы времени с частотой не меньше, чем удвоенная частота самой высокочастотной составляющей сигнала, то полученная серия измерений будет содержать всю информацию об исходном сигнале и этот сигнал может быть восстановлен. Важно иметь в виду, что т.к. каждый из 16 уровней является лишь приближением реального значения амплитуды сигнала, то точное восстановление исходного сигнала будет невозможно. На стороне приемника по полученному цифровому коду восстанавливают аналоговый сигнал. Однако, как мы уже отметили, вследствие «округления» точное восстановление сигнала невозможно. Этот эффект называют ошибкой квантования или шумом квантования. Существуют методы его понижения за счет нелинейных методов квантования.

Дельта-модуляция .

Другой альтернативой ИКМ является метод дельта-модуляции. На исходную непрерывную функцию, представляющую аналоговый сигнал, накладывают ступенчатую функцию. Значения этой ступенчатой функции меняются на каждом шаге квантования по времени Ts на величину δ. Замена исходной функции на эту дискретную, ступенчатую функцию интересна тем, что поведение последней носит двоичный характер. На каждом шаге значение ступенчатой функции либо увеличивается на δ, будем представлять этот случай 1,

либо уменьшается на δ – случай 0. Внизу рисунка показан оцифрованный вид этой функции. Процесс передачи при использовании дельта-модуляции организован следующим образом. В момент очередного замера текущее значение исходной функции сравнивается со значением ступенчатой функции на предыдущем шаге. Если значение исходной функции больше, передается 1, в противном случае – 0. Таким образом, ступенчатая функция всегда меняет свое значение. У метода дельта-модуляции есть два параметра: величина шага d и частота замеров, или шаг квантования. Выбор шага d – это баланс между ошибкой квантования и ошибкой перегрузки по крутизне (см. рисунок). Когда исходный сигнал изменяется достаточно медленно, то возникает только ошибка квантования, чем больше d, тем больше эта ошибка. Если же сигнал изменяется резко, то скорость роста ступенчатой функции может отставать. Это вид ошибки растет с уменьшением d.

9.Теоретические основы передачи данных (ограничения на пропускную способность передачи сигналов, взаимосвязь пропускной способности канала и ширины его полосы пропускания). Передача аналоговых данных аналоговыми сигналами.

Все виды информации могут быть представлены при передаче в виде электромагнитных импульсов. В зависимости от среды передачи и организации СПД могут применяться либо аналоговые, либо цифровые сигналы. Любой сигнал можно рассматривать либо как функцию времени, т.е. то, как различные параметры сигнала изменяются со временем, либо как функцию частоты.

Разные формы представления сигнала

Как уже было сказано, любой сигнал можно рассматривать как функцию времени, либо как функцию частоты. В первом случае эта функция показывает, как меняются со временем параметры сигнала, например, напряжение или сила тока. Если эта функция имеет непрерывный характер, то говорят о непрерывном сигнале. Если эта функция имеет дискретный вид, то говорят о дискретном сигнале

Частотное представление функции основано на том факте, что любая функция от вещественной переменной может быть представлена в виде ряда Фурье:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi nft) + \sum_{n=1}^{\infty} a_n \cos(2\pi nft)$$ (1), где f =1/T - частота, a_n, b_n - амплитуды n-ой гармоники.

Ясно, что на практике нельзя учесть бесконечно много гармоник. Все их учитывать и не надо потому, что энергия сигнала распределяется не равномерно между гармониками разной частоты. В общем случае соотношение здесь таково, что низкочастотные составляющие несут большую часть энергии. Ни в какой среде сигнал не может передаваться без потери энергии. С ростом частоты искажения растут. Любая среда передачи ограничивает максимальную частоту передаваемого сигнала, а следовательно, и частоту гармоник, которые можно использовать для аппроксимации функции g(t). Тем самым аппроксимация (точность воспроизведения формы) сигнала ухудшается и скорость передачи понижается. Характеристику канала, определяющую спектр частот, которые физическая среда канала пропускает без существенного понижения мощности сигнала, называют **полосой пропускания** канала.

Взаимосвязь пропускной способности канала и его полосы пропускания

Максимальную скорость, с которой канал способен передавать сигнал, называют **пропускной способностью канала.**

Теорема Найквиста

max data rate = $2H \log_2 V$ ит/сек,

где H – ширина полосы пропускания канала, выраженная в Гц, V - количество уровней в сигнале. Эта теорема также показывает, что, например, бессмысленно сканировать линию чаще, чем удвоенная ширина полосы пропускания. Действительно, все частоты выше этой отсутствуют в сигнале. Однако теорема Найквиста не

учитывает шум в канале, который измеряется как отношение мощности полезного сигнала к мощности шума: S/N. Эта величина измеряется в децибелах: 10 $\log_{10}$(S/N) dB. Например, если отношение S/N равно 10, то говорят о шуме в 10 dB, если отношение равно 100, то - 20 dB. На случай канала с шумом есть теорема Шеннона, по которой максимальная скорость передачи по каналу с шумом равна

H $\log_2$ (1+S/N) бит/сек., где S/N - соотношение сигнал-шум в канале.

Как мы уже отмечали в разделе 2.1.2, скорость передачи данных зависит от способа представления данных на физическом уровне и сигнальной скорости, или скорости модуляции - скорости изменения значения сигнала. Скорость изменений сигнала в секунду измеряется в единицах, называемых бот. Если скорость изменения значения сигнала b бот, то это не означает, что данные передается со скоростью b бит/сек. Многое зависит способа кодирования сигнала: одно изменение значения может кодировать сразу несколько бит. Если используется 8 значений (уровней) сигнала, то каждое изменение его значения кодирует сразу 3 бита. Если используется только два значения сигнала, то скорость в битах равна скорости в ботах.

- Аналоговые данные – аналоговый сигнал. Аналоговые данные в электрической форме могут легко и дешево передаваться с помощью аналоговых сигналов. Хорошим примером этому случаю является телефония, которую мы рассмотрим в разделе 2.5.

Аналоговые данные – Аналоговый сигнал

Анализ этого случая начнем с того, чтобы понять, где может возникнуть потребность в такого вида преобразованиях. Аналоговая модуляция цифровых данных возникает там, где нет цифровых каналов. Цифровое кодирование аналоговых данных возникает тогда, когда есть цифровые каналы. Прежде всего, такая потребность возникает при использовании радиоканалов. Если передавать аудиоинформацию в голосовом диапазоне (300 – 3000 Гц), то потребуется антенна диаметром в несколько километров. Модуляция, т.е. объединение исходного сигнала m(t) и несущей частоты f_c, позволяет нужным образом изменять параметры исходного сигнала и тем самым упростить решение ряда технических проблем. Кроме этого, модуляция позволяет использовать методы мультиплексирования или уплотнения. (О мультиплексировании мы поговорим в разделе 2.4, а в разделах 2.3 и 2.5 мы рассмотрим подробнее использование электромагнитных волн для передачи).

При амплитудной модуляции форма результирующего сигнала определяется формулой:

$S(t) = \left[1 + n_a x(t)\right] \cos 2\pi f_c t$, где f_c - частота несущей,

n_a– индекс модуляции, который определяют как отношение амплитуды исходного сигнала к амплитуде несущего сигнала.

Форма результирующего сигнала при частотной модуляции определяется следующим выражением:

$S(t) = A_c \cos\left(2\pi f_c t + n_f m(t)\right)$, где n_f - индекс частотной модуляции, $m(t)=1+n_a x(t)$.

Сигнал, получаемый фазовой модуляцией, определяет соотношение:

$S(t) = A_c \cos\left(2\pi f_c t + n_p m(t)\right)$, где n_p – индекс фазовой модуляции.

Все эти три вида модуляции порождают сигнал S(t), спектр которого симметричен относительно f_c.

11.Беспроводная связь (электромагнитный спектр, радиопередача, микроволновая передача, видимое излучение). TDMA, FDMA, CDMA - методы множественного доступа к беспроводному каналу.

Беспроводная связь

Электромагнитный спектр

Как известно, электроны при движении образуют электромагнитные колебания. Это явление Максвелл предсказал в 1865, а Генрих Герц экспериментально обнаружил в 1887 году. Если к источнику электромагнитных волн подключить антенну соответствующего размера, то волны будут распространяться и регистрироваться приемниками. Длина антенны, как у приемника, так и у передатчика, и длина излучаемой/принимаемой ею волны связаны определенными соотношениями. Например, длина антенны приемника не может быть короче половины длины принимаемой ею волны. При определенных условиях, о которых мы будем разговаривать ниже, волны будут распространяться в строго определенном направлении. В этом случае антенна приемника должна быть должным образом ориентирована в пространстве по отношению к антенне передатчика, чтобы принимать сигналы. При других условиях антенна передатчика распространяет электромагнитные волны во всех направлениях. В вакууме электромагнитная волна распространяется со скоростью света (C=3x108 м/сек.). В медном проводнике эта скорость составляет 2/3 от скорости в вакууме. Будем обозначать f - частоту, а λ - длину волны. Фундаментальное соотношение, соединяющее f, C и λ, таково:

$f \cdot \lambda = C$ (2-1)

Поскольку C - константа, зная λ, мы знаем f, и наоборот. Например, волны с частотой в 1 МГц, согласно этому соотношению, имеют длину волны 300 метров, а волны длиной в 1 см имеют частоту 30 ГГц. Напомним, что длина волны определяет размер и геометрию антенны. Для передачи информации из всего этого спектра используется только следующие диапазоны: радио, микроволновый, инфракрасный, видимый и, частично, ультрафиолетовый. Диапазоны рентгеновского излучения, гамма-излучения и большая часть ультрафиолетового, хотя и имеют большие частоты, а потому и более предпочтительны для передачи, однако требуют сложной аппаратуры для генерации и модуляции, плохо преодолевают препятствия и, что самое главное, опасны для живой материи. Количество данных, передаваемых электромагнитной волной, определяется ее шириной, т.е. спектром частот гармоник, составляющих эту волну.

Таблица 2-27. Характеристики частотных диапазонов

Диапазон частот	Название	Аналоговые данные		Цифровые данные		Область применения
		Модуляция	Полоса пропускания	Модуляция	Скорость передачи	
30-300 кГц	LF (low frequency - низкие частоты, НЧ)	Обычно не используется.		ASK, FSK, MSK	0,1-100 бит/сек.	Навигация
300-3000 кГц	MF (medium frequency - средние частоты, СЧ)	AM	до 4 кГц	ASK, FSK, MSK	10-1000 бит/сек.	AM-радио
3-30 МГц	HF (high frequency - высокие частоты, ВЧ)	AM, SSB	до 4 кГц	ASK, FSK, MSK	10-3000 бит/сек.	Коротковолновое радио
30-300 МГц	VHF (very high frequency - очень высокие частоты, ОВЧ)	AM, SSB, FM	5 кГц - 5 МГц	FSK, PSK	до 100 кбит/сек.	Телевидение метрового диапазона
300-3000 МГц	UHF (ultrahigh frequency - ультравысокие частоты, УВЧ)	FM, SSB	до 20 МГц	PSK	до 10 Мбит/сек.	Телевидение дециметрового диапазона, наземные микроволны

3-30 ГГц	SHF (superhigh frequency - сверхвысокие частоты, СВЧ)	FM	до 500 МГц	PSK	до 100 Мбит/сек.	Наземные и спутниковые микроволны
30-300 ГГц	EHF (superhigh frequency - чрезвычайно высокие частоты, ЧВЧ)	FM	до 1 ГГц	PSK	до 750 Мбит/сек.	Экспериментальные соединения «точка-точка»

Рассмотрим уравнение 2-1. Разрешив его относительно f и продифференцировав по λ, получим:

$$\frac{df}{d\lambda} = -\frac{c}{\lambda^2} \quad (2\text{-}2)$$

Переписав уравнение 2-2 в разностной форме, получим:

$$\Delta f = \frac{c\Delta\lambda}{\lambda^2} \quad (2\text{-}3)$$

Задав некоторую полосу длин волн, мы получим полосу частот, откуда получим скорость передачи для этой полосы частот. Чем шире полоса, тем выше битовая скорость. На практике чаще всего используются узко-частотные полосы ($\Delta f/f<<1$). В дальнейшем, рассматривая использование отмеченных выше частей электромагнитного спектра, мы будем предполагать именно узко-частотную передачу. В противоположность такой передаче используется, особенно военными и спецслужбами, так называемая широко-частотная передача. Идея ее состоит в том, что при передаче частота несущей волны меняется по определенному закону в диапазоне полосы. Перехватить такую передачу можно, только если известен закон изменения частоты несущей.

Радиопередача

Радиоволны распространяются на большие расстояния, легко преодолевают преграды, техника их генерации и приема хорошо изучена, есть много специалистов по ее применению. Поэтому они широко используются для связи как вне, так внутри помещений. Поскольку радиоволны распространяются во всех направлениях, то принимающая и передающая антенны не требуют дополнительной настройки и взаимного расположения. Свойства радиоволн зависят от их частоты. На низких частотах, т.е. длинных волнах, они прекрасно преодолевают препятствия, но мощность сигнала падает пропорционально 1/r3 , где r - расстояние до источника. На высоких частотах радиоволны распространяются по прямой, но хуже преодолевают препятствия. Для некоторых частот помехой становится даже дождь. На всех частотах радиоволны чувствительны к помехам от электрических устройств. В силу перечисленных выше свойств лицензирование, т.е. право на использование частот в радиодиапазоне, находится под жестким контролем государства.

Микроволновая передача

При частоте выше 10 МГц мы попадаем в область микроволнового диапазона. Волны в этом диапазоне распространяются в строго определенном направлении и могут быть сфокусированы с помощью параболической антенны, имеющей вид телевизионной тарелки. Однако приемная и передающая антенны должны быть тщательно ориентированы в пространстве по отношению друг к другу. Такая направленность позволяет строить цепочку ретрансляторов и таким образом передавать сигнал на большие расстояния. До появления оптоволокна радиорелейная связь составляла основу телефонных систем на больших расстояниях. На определенном расстоянии друг от друга ставили башни с ретрансляторами. Высота башни зависела от расстояния и мощности передатчика. Обычно 100-метровая башня покрывает расстояние в 80 км.

Микроволны не проходят сквозь здания так же хорошо, как низкочастотные волны. Кроме этого, из-за рефракции в нижних слоях атмосфер они могут отклоняться от прямого направления. При этом увеличивается задержка, нарушается передача. Передача на этих частотах зависит также и от погоды. Как уже не раз отмечалось, при повышении влажности (дождь, туман и т.п.) ширина полосы резко сужается, растет шум, сигнал рассеивается. Обычно операторы держат определенный частотный резерв (около 10% каналов) на случай подобных нарушений и при необходимости переключаются на резервные частоты, чтобы обойти зону осадков. На сегодня микроволновый диапазон широко используется в телефонии, сотовой телефонии, телевидении и других приложениях. Одно из главных достоинств микроволнового диапазона - не надо ничего прокладывать.

Инфракрасные и миллиметровые волны

Инфракрасное излучение и излучение в миллиметровом диапазоне используется на небольших расстояниях в блоках дистанционного управления. Основной недостаток излучения в этом диапазоне - оно не проходит через преграду. Для инфракрасного излучения лист бумаги – непреодолимое препятствие. Этот недостаток одновременно является преимуществом, когда излучение в одной комнате не интерферирует с излучением в другой. На эту частоту не надо получать разрешения. Это прекрасный канал для передачи данных внутри помещений на небольших расстояниях.

Видимое излучение

Видимый диапазон также используется для передачи. Обычно источником света является лазер. Монохромное когерентное излучение легко фокусируется. Однако дождь или туман портят дело. Передачу способны испортить даже конвекционные потоки на крыше, возникающие в жаркий день. Они вызывают дрожание луча вокруг приемника, что ухудшает качество передачи.

CDMA (Code Division Multiple Access) – множественный доступ на основе разделения кодов

GSM – пример системы, где использована довольно сложная комбинация техник FDM, TDM, ALOHA для беспроводной сотовой связи. В ней ни один из пользователей системы не может использовать всю полосу пропускания, предоставленную системе. Если при этом принять в расчет сужение полосы пропускания из-за проблем на границе сот, падение мощности сигналов от мобильных терминалов в пограничных сотовых зонах, накладных расходов на шифрование в целях безопасности, то становится ясно, что высокую скорость передачи в этой системе получить не просто. Метод CDMA основан на принципиально иной идее – каждый участник связи может использовать всю полосу пропускания канала. У каждого свой уникальный «язык», поэтому все могут говорить сразу. Понимать друг друга будут только те, кто говорит на одном языке. В CDMA-системе каждый бит сообщения кодируется последовательностью из m частиц. Бит со значением 0 передается инвертированной последовательностью частиц, бит 1 – прямой. Каждой мобильной станции присваивается уникальный код – последовательность частиц. Кроме этого, поскольку каждая станция имеет уникальную последовательность частиц, то не требуется дополнительного шифрования. Метод ортогональных последовательностей. Как получатель узнает последовательность частиц отправителя? Например, за счет соответствующего быстродействия он может слышать всех, обрабатывая алгоритмом декодирования для каждой последовательности в параллель.

12.Телефонные сети: структура, локальная петля, магистраль и мультиплексирование.

Когда требуется соединить несколько рядом стоящих компьютеров, то обычно прокладывают кабель. Когда кабель должен пересечь дорогу или городские коммуникации, дело становиться сложнее, а стоимость такой операции возрастает. В этих случаях обычно обращаются к телефонной компании.

Структура телефонной сети

Структура современной телефонной сети весьма избыточная и многоуровневая. На этом рисунке используются следующие обозначения: АКТС - автоматическая коммутируемая телефонная сеть; ТФОП - телефоны общего пользования. Позднее мы рассмотрим, что такое зоновая телефонная сеть, городская, сельская, учрежденческая. Описание, которое мы приведем здесь, является существенным упрощением реальности, но дает достаточно полное представление об устройстве телефонной сети. Каждый абонент соединен двумя витыми парами с ближайшей местной телефонной станцией (ТС), это соединение называют локальным соединением, абонентской линией или последней милей. В России протяженность локального соединения колеблется от сотен метров до 6-8 км. В городе оно короче, в сельской местности длиннее. Местная ТС соединена в крупных городах с районной ТС либо городской ТС. Районные и городские ТС соединены с региональными или междугородными ТС, и т.д. в соответствии со структурой телефонного номера. В результате создается прямое соединение между абонентами. ТС соединяются между собой магистральными линиями. На самом деле иерархия промежуточных узлов коммутации несколько сложнее. Главное что надо уяснить, - есть несколько уровней ТС, каждая из которых может осуществлять коммутацию. В дальнейшем телефонные станции любого уровня мы будем просто называть узлами коммутации. Соединения между узлами коммутации должны обладать большой пропускной способностью, чтобы по ним можно было передавать одновременно несколько разговоров. Пропускная способность местной линии должна быть достаточной для одного телефонного разговора. Для абонентских линий чаще всего применяли и применяют витую пару. Для магистралей между узлами коммутации используют коаксиальные кабели, оптоволокно и радиорелейные линии на микроволнах.

Итак, современная телефонная сеть состоит из:

- абонентской линии - локального соединения или последней мили (соединение «клиент - местная ТС»)
- магистралей - оптоволоконных или микроволновых (соединение ТС-ТС)
- станций коммутации (ТС)

Локальное соединение. Локальное соединение, или абонентская линия связывает абонента с ближайшим узлом коммутации. Это соединение также называют последней милей. при передаче данных приходится преобразовывать данные четыре раза из цифровой формы в аналоговую и обратно. Несмотря на то, что между узлами коммутации передача осуществляется в цифровой форме, в локальном соединении она пока аналоговая.

Модем. Из-за вышерассмотренных искажений сигнала желательно использовать при передаче как можно меньше гармоник. Однако скачкообразная форма цифрового сигнала как раз требует большого числа гармоник при передаче, чтобы как можно точнее воспроизвести форму сигнала, что требует от канала в свою очередь широкой полосы пропускания. Решение проблемы лежит в использовании несущей частоты в сочетании с разными способами модуляции сигнала. Три основные способа модуляции - амплитудная, частотная, фазовая и их комбинации. Как мы уже знаем, устройство, которое преобразует поток битов в модулированный сигнал и обратно, называется модем. Чтобы увеличить скорость передачи, недостаточно увеличивать частоту несущей волны. Надо увеличивать число бит на осцилляцию, т.е. изменение уровня сигнала. Другой важной проблемой при использовании телефонной линии является эхо. Причина этого явления проста - когда сигнал достигает приемника, часть его энергии отражается и возвращается к передатчику. При небольших расстояниях между приемником и передатчиком это практически незаметно. Когда расстояние велико, задержка между сигналом и эхом становится значительной.

Соединение RS-232

Важным элементом протокола физического уровня является интерфейс между компьютером или телефоном и модемом.

Проблема «последней мили»

Возникла проблема, как обеспечить частные квартиры и дома линиями связи надлежащей пропускной способности, - так называемая «проблема последней мили».

Работы по решению этой проблемы велись в 4-х направлениях. Первое направление, достаточно «прямолинейное», было связано с подведением оптоволокна прямо в квартиру. Это направление называется FTTH (Fiber To The Home). Второе направление было связано со стремлением сократить длину локального соединения до минимума. По имеющимся данным («Электросвязь» №11, 1997, с.13), в городских телефонных сетях России средняя длина абонентской линии составляет 1280 м (коэффициент вариации 0.59), ни одна абонентская линия ни в городе, ни в сельской местности не превышает 5 км. Было предложено подтянуть оптоволокно от местного узла коммутации до опорного шкафа развязки внутри микрорайона, а далее возможны были два варианта. От опорного шкафа использовать обычную витую пару с технологией HDSL из семейства xDSL (семейство этих технологий мы рассмотрим чуть ниже), либо использовать коаксиальные кабели сети кабельного телевидения (это решение получило название Hybrid Fiber Coac – HFC). Коаксиальный кабель в сочетании с оптоволокном обеспечивает одновременную передачу 40-50 аналоговых каналов, в том числе радиовещание, телевизионные передачи, телетекст. При использовании ADSL – асимметричной DSL-технологии (о которой речь пойдет чуть ниже), обеспечивающей интерактивность, добавляются видео по заказу, игры, доступ в Интернет. Третий вариант решения – это использовать беспроводные технологии (WLL – Wireless Local Loop). Мы их будем рассматривать позднее. Сейчас лишь отметим, что доступный для них диапазон частот сильно ограничен международными соглашениями. Скорости передачи данных уступают проводным технологиям.

Четвертый вариант решения – это использовать стандарты серии xDSL.

В таблице 2-39 собраны краткие характеристики этих 4-х направлений решения проблемы последней мили.

Технологии xDSL

xDSL – это семейство технологий, предназначенных для организации цифровых абонентских линий – DSL (Digital Subscriber Line) – с использованием в качестве среды передачи медных витых пар существующих локальных соединений телефонных кабельных систем. На современном этапе развития семейство xDSL включает следующие технологии:

- DSL
- IDSL
- HDSL, SDSL
- ADSL, RADSL, UADSL
- VDSL

Это весьма важное направление развития физических линий связи, поэтому мы хотя бы кратко опишем каждую из технологий этого семейства. Родившееся как технология цифровых каналов в ISDN-сетях, семейство технологий xDSL получило развитие в новой сфере – абонентский доступ в Интернет. ISDN-сети (Integrated Service Digital Network – сети с интегрированным сервисом) будут рассмотрены позже. По аналогии с модемами для работы по физической линии, модемы xDSL не ограничиваются для передачи информации спектром канала телефонных частот. Они используют всю полосу пропускания витой пары. Широкая полоса сигнала, используемого в этом семействе технологий, не позволяет работать по коммутируемым телефонным линиям (телефонные коммутаторы не рассчитаны на такой спектр частот). Поэтому xDSL-модемы могут работать только на участке телефонных кабельных систем между абонентом и сетью поставщика услуг или между двумя абонентами при непосредственном соединении их абонентских линий (без участия станции коммутации) - выделенные линии. Отличительной чертой семейства xDSL, по сравнению с модемами для физических линий, является использование спектра частот, не пересекающегося со спектром канала телефонных частот, благодаря чему по абонентской линии можно вести телефонные переговоры одновременно с передачей цифровой информации.

Технология DSL. Технология DSL – «цифровая абонентская линия» – позволяет использовать существующие линии связи для передачи цифровой информации по одной витой паре со скоростью до 160 кбит/сек. (при этом в прямом и обратном направлении поддерживается одинаковая скорость). Технология разрабатывалась для организации цифровой абонентской линии для сетей ISDN BRI (сети ISDN будут рассмотрены в разделе 2.5.8). Реализация в оборудовании DSL-интерфейса ISDN BRI получила название IDSL. В оборудовании IDSL не предусматривается поддержка аналоговой телефонной линии, так как

телефонная связь может осуществляться по цифровым каналам ISDN. Сейчас существуют модификации оборудования DSL – Fast DSL, передающие информацию со скоростью до 256 кбит/сек. Технология DSL поддерживает аналоговую телефонную линию. Стандартный метод линейного кодирования – 2B1Q (мы рассматривали этот метод в разделе 2.2.1) применяется практически во всех типах оборудования xDSL, за исключением оборудования подсемейств ADSL и VDSL, речь о которых чуть ниже. Максимальное расстояние (то есть максимальная длина двухпроводной линии, на которой может работать аппаратура) для этой технологии составляет 7,5 км при диаметре жилы кабеля 0,5 мм, что вполне покрывает длину абонентских линий в России.

Технология HDSL . Дальнейшим развитием DSL стала технология высокоскоростной цифровой абонентской линии HDSL (High-data-rate DSL). Оборудование HDSL обеспечивает дуплексный (симметричный) обмен на скорости 768 или 1024 кбит/с по одной витой паре и 2048 кбит/с по двум – трем витым парам. Система является однокабельной: по каждой паре проводов осуществляется и прием, и передача информации. Неисправность в одной паре кабеля не приводит к прекращению передачи, а только уменьшает ее скорость. Максимальная удаленность между репитерами (промежуточными усилителями) не более 3 км. Поэтому применение этой технологии в России требует в среднем использовать один репитер на каждую абонентскую линию. Стандартная ширина сигнала, используемого при передаче, - 80–196 кГц. **Оборудование HDSL в основном предназначено для применения в корпоративных сетях. Отсутствие поддержки аналоговой телефонной линии компенсируется возможностью передачи речи в цифровом виде через интерфейсы E1 (стандарт E1 будет рассмотрен в разделе 2.5.5.3).**

Технология SDSL

SDSL (Single Line DSL) – разновидность технологии HDSL. Системы SDSL обеспечивают дуплексную передачу потока на скорости 2048 кбит/сек. по одной витой паре проводов на расстояние 3–4 км при диаметре жилы кабеля 0,4–0,5 мм. Сейчас не делают существенного различия между технологиями HDSL и SDSL и выпускают оборудование HDSL, передающее информацию как по нескольким, так и по одной паре проводов. Также иногда название SDSL расшифровывают как Symmetric DSL, подчеркивая тем самым симметричность потоков информации.

Технология VDSL

Технология VDSL (Very High-data-rate DSL) находится в стадии разработки. Ожидается, что с ее помощью будет достигнута скорость передачи по абонентской линии от 12 до 51 Мбит/с. Наряду с медным кабелем, рассматривается возможность использования оптического кабеля. Оборудование VDSL может функционировать в режиме как асимметричных, так и симметричных цифровых потоков. Метод кодирования – DMT. Дискретное многочастотное кодирование (DMT – Discrete Multitone) предполагает разбиение всей полосы пропускания на подполосы по 4 КГц и в каждой подполосе использовать свою несущую. Метод кодирования в подполосе – квадратичная амплитудная модуляция (QAM), которую мы рассмотрели в разделе 2.2.1.

Технология ADSL

Асимметричная DSL (Asymmetric DSL) – дальнейшее развитие технологии HDSL – в настоящее время является наиболее продвинутой в семействе xDSL. Она обеспечивает передачу по витой паре потоков до 9 Мбит/с в одном направлении (как правило, в сторону пользователя) и до 640 кбит/с – в другом. По широкому входящему каналу абонент получает данные или видео из Интернета, а исходящий используется для отправки запросов на получение информации. Следует отметить, что пропускной способности исходящего канала достаточно для передачи электронной почты, файлов и для проведения голосовых переговоров через Интернет. ADSL ориентирована на абонентов индивидуального сектора и, благодаря применению внутренних или внешних речевых разделителей, позволяет вести обычные телефонные переговоры.

Технология RADSL

Разновидностью ADSL-технологии является технология RADSL (Rate-adaptive DSL), которая может функционировать в асимметричном режиме как ADSL и в симметричном – как HDSL. Технология RADSL

позволяет отслеживать текущее состояние кабеля (электрические параметры и уровень шума (помех)) и динамически регулировать пропускную способность каналов связи, а также поддерживать максимально возможную степень передачи при требуемом минимальном уровне ошибок в канале связи.

Технология UADSL

Существует вариант технологии ADSL, называемый UADSL (Universal ADSL). Эта версия является упрощенным вариантом цифрового доступа и потому более дешева. Она ориентирована на индивидуальных абонентов. Максимальные скорости обмена в ней снижены до 1,5/0,384 Мбит/сек. и упрощена настройка. При скорости 1,5 Мбит/сек. невозможно получать передачи кабельного ТВ, как в ADSL, но этого вполне достаточно для доступа абонента в Интернет.

13. Телефонные сети: структура, локальная петля, магистраль и мультиплексирование.
современная телефонная сеть состоит из:

- абонентской линии - локального соединения или последней мили (соединение «клиент - местная ТС»)
- магистралей - оптоволоконных или микроволновых (соединение ТС-ТС)
- станций коммутации (ТС)

Магистрали и мультиплексирование

Наряду с абонентской линией, следующим важным компонентом телефонных систем являются магистрали, соединяющие узлы коммутации разного уровня. Здесь мы рассмотрим их организацию и функционирование. Одним из существенных факторов при организации магистрали был и остается экономический. Дело в том, что затраты на прокладку кабеля в значительной степени определяют внешние условия (город, сельская местность, глубина залегания, наличие инженерных коммуникаций и т.д.), а не технические характеристики, например, пропускная способность. Поэтому чем больше абонентов смогут использовать один и тот же кабель, тем быстрее окупятся затраты на его прокладку, тем дешевле будет стоить каждому из них его эксплуатация. Вспомним историю. Вестерн Юнион объявила в свое время конкурс на решение проблемы передачи нескольких телеграмм по одной линии, которым заинтересовался Александр Белл. За 100 лет существования телефона были инвестированы огромные средства в создание методов и оборудования, позволяющих использовать одну и ту же магистраль одновременно для передачи нескольких разговоров. Такой технический прием называют мультиплексированием, или уплотнением. Созданные в телефонии схемы мультиплексирования можно разделить на два больших класса: мультиплексирование с разделением частот и мультиплексирование с разделением по времени. Кроме этого, были разработаны методы мультиплексирования на основе разделения длин волн и на основе разделения кодов. Метод разделения длин волн применяют в оптоволоконных системах. Методы разделения кодов используют в системах беспроводной связи.

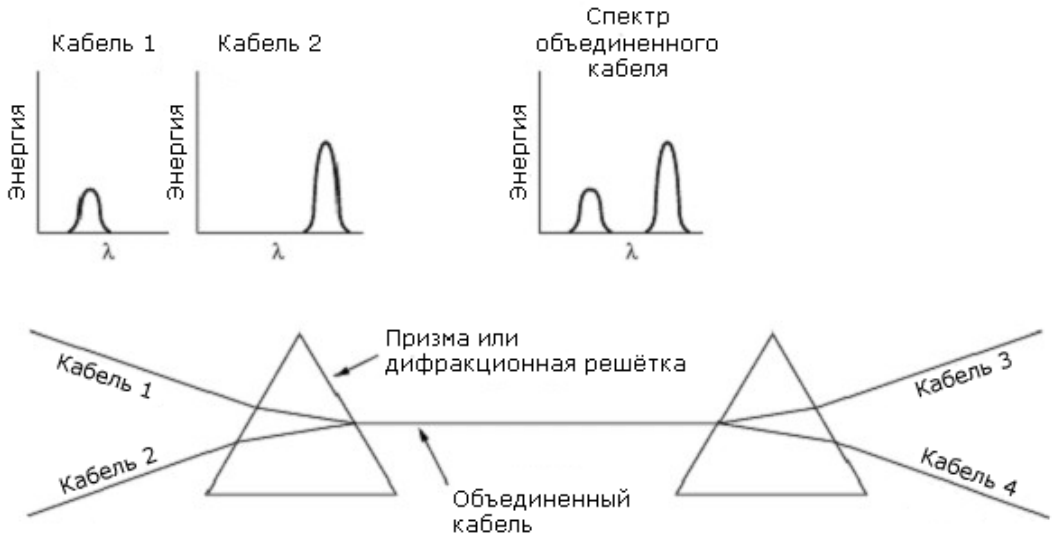
Мультиплексирование с разделением частот

Идея мультиплексирования с разделением частот очень проста: весь диапазон частот полосы пропускания кабеля разбивают на поддиапазоны, которые называют каналами. По каждому каналу идет независимая передача.

Мультиплексирование с разделением длины волны

Этот способ мультиплексирования используется для волоконноптических каналов, пример которых мы рассмотрим в разделе 2.5.5.4. Самый простой способ такого мультиплексирования показан на рисунке 2-42.

Рисунок 2-42. Мультиплексирование с разделением длины волны



Два волоконнооптических кабеля с импульсами разной длины волны подводят к одной призме. Свет, пройдя через призму (или дифракционную решетку), смешивается в единый луч, который на другом конце разделяется с помощью другой призмы.

Мультиплексирование с разделением по времени

Частотное мультиплексирование требует применения аналоговых схем и малопригодно для управления компьютером. Мультиплексирование с разделением времени или TDM-мультиплексирование (Time Division Multiplexing), наоборот, предполагает использование цифрового оборудования и хорошо соответствует возможностям компьютера. Следует отметить, что оно подходит только для работы с данными в цифровой форме. Поскольку по абонентской линии телефонный сигнал передают в аналоговой форме, то его надо сначала оцифровать. Оцифровка сигнала происходит на местном узле коммутации, куда сходятся абонентские линии с аналоговыми сигналами. На местном узле коммутации аналоговые сигналы с абонентских линий оцифровываются, объединяются и передаются на узлы коммутации следующего уровня по магистральным шинам. Здесь мы рассмотрим, как это все происходит. преобразование сигнала в цифровую форму и обратно осуществляет специальное устройство, называемое кодек (coder-decoder). Есть два основных метода преобразования аналогового сигнала в цифровую форму и обратно. Это метод импульсно-кодовой модуляции (ИКМ-метод) и разностный метод Дельта-модуляции. Напомним, что в ИКМ-методе аналоговая линия сканируется, в соответствии с теоремой Найквиста, с удвоенной частотой старшей гармоники - в случае телефонных систем с частотой 8 000 Гц. Амплитуда аналогового сигнала разбивается на определенное количество уровней. При каждом замере определяется не абсолютное значение сигнала, а его уровень. Номер уровня и передается в виде двоичного кода.

Когда метод ИКМ начал развиваться, МКТТ не смогло сразу договориться и ввести единый стандарт на применение этого метода в телефонии. В результате возникло два варианта: европейский (E1) и T1, получивший распространение в США и Японии.

Стандарт E1 предполагает мультиплексирование 30 каналов. Стандарт T1 позволяет мультиплексировать 24 линии.

14.Телефонные сети: структура, методы коммутации.

Итак, современная телефонная сеть состоит из:

- абонентской линии - локального соединения или последней мили (соединение «клиент - местная ТС»)
- магистралей - оптоволоконных или микроволновых (соединение ТС-ТС)
- станций коммутации (ТС)

Коммутация

В телефонных сетях используются два разных способа коммутации: коммутация каналов и коммутация пакетов.

Коммутация каналов и коммутация пакетов

Основные различия между коммутацией каналов и коммутацией пакетов приведены ниже:

- При коммутации каналов создается соединение, пропускная способность которого полностью резервируется за двумя абонентами, вне зависимости от того, какая пропускная способность реально им потребуется. При коммутации пакетов физическая линия может быть использована пакетами разных абонентов. Следует иметь в виду, что так как при коммутации пакетов не происходит жесткого закрепления канала, то резкое увеличение потока пакетов в узле коммутации (в случае коммутации пакетов эти узлы называют маршрутизаторами), может привести к их перегрузке и потере части пакетов.
- При коммутации каналов гарантировано, что все данные поступят абоненту и в том порядке, в каком их послали. При коммутации пакетов из-за ошибок маршрутизации пакеты могут быть направлены не по назначению, сохранение их исходного порядка получателю не гарантируется.
- Коммутация каналов абсолютно прозрачна для абонентов. Они могут пересылать данные в любой кодировке и формате. При коммутации пакетов формат и способ кодировки пакетов задан заранее и определяется оператором связи.
- При коммутации пакетов плата взимается за время соединения и число переданных пакетов. При коммутации каналов плата берется исключительно за время и длину соединения.

Иерархия узлов коммутации

Совокупность узлов коммутации, оконечных абонентских устройств и соединяющих их каналов и линий связи называют сетью телефонной связи. Сети связи создаются для передачи информации между абонентами и бывают коммутируемыми и некоммутируемыми. Сеть называется коммутируемой, когда тракт передачи информации создается по запросу абонента на время сообщения, и некоммутируемой, когда тракт передачи информации обеспечивается постоянным соединением между определенными абонентами и нет необходимости в коммутации. Телефонные сети являются коммутируемыми. Общегосударственная телефонная сеть (ОАКТС) состоит из междугородной телефонной сети и зоновых телефонных сетей. Междугородная телефонная сеть обеспечивает соединение автоматических междугородных телефонных станций (АМТС) различных зон. См. рисунок 2-31 в разделе 2.5.2.

Зоновая телефонная сеть состоит из местных телефонных сетей, расположенных на территории зоны, и внутризоновой телефонной сети, которая соединяет между собой эти сети. Местные телефонные сети разделяются на городские, обслуживающие город и ближайшие пригороды (ГТС), и сельские (СТС), обеспечивающие связь в пределах сельского административного района. Учрежденческо-производственная телефонная сеть (УПТС) служит для внутренней связи предприятий, учреждений, организаций и может быть соединена с сетью общего пользования либо быть автономной. Зоновая телефонная сеть включает всех абонентов определенной территории, охватываемой единой семизначной нумерацией (см. раздел 2.5.2), и является частью ОАКТС. Территории зоновых сетей совпадают с территориями административных областей (республик). В зависимости от конфигурации области и телефонной плотности территории нескольких областей могут быть объединены в одну зону и, наоборот, одна область может быть разделена на две зоны и более. Зоновая сеть включает в себя ГТС и СТС, причем на территории одной зоны могут быть несколько ГТС и СТС. Крупные города с семизначной нумерацией выделяются в отдельные зоны. Сельские телефонные сети охватывают более обширные территории, чем городские, но плотность телефонных аппаратов

значительно меньше. Поэтому емкость автоматических телефонных станций АТС в сельских местностях значительно меньше, чем в городах. Городская телефонная сеть состоит из комплекса сооружений (станционное оборудование, здание, линейные сооружения, абонентские устройства и др.), обеспечивающих телефонной связью абонентов города и прилегающих к нему пригородов. Стоимость линейных сооружений в значительной степени зависит от принципа построения ГТС и ее емкости.

Коммутаторы каскадные

Теперь, познакомившись с иерархией телефонных станций (узлов коммутации), давайте рассмотрим, как устроен сам коммутатор. Самый простой вид коммутаторов - это прямой коммутатор n x n, у которого есть n входных и n выходных линий. Он показан на рисунке 2-52. В каждой точке пересечения стоит полупроводниковый переключатель, который замыкает соответствующие линии. Основной недостаток этого типа коммутаторов - квадратичный рост сложности при увеличении n. Сложность коммутатора измеряется в количестве точек пересечения. Даже если учесть, что в случае дуплексных линий и отсутствии самосоединений нам требуется только половина пересечений (выше или ниже диагонали), то все равно нам надо порядка n(n-1)/2 переключателей. Идея построения этого типа коммутаторов такова: разделить простой коммутатор на части, соединить эти части между собой промежуточными дополнительными коммутаторами. Рассмотрим пример трехслойного каскадного коммутатора. В первом слое N входных линий разбиваются на группы по n линий в каждой. На втором слое N/n прямых коммутаторов n x k линий каждый соединяются с k коммутаторами N/n x N/n линий. Третий каскад повторяет первый в обратном порядке: не n x k, а k x n.

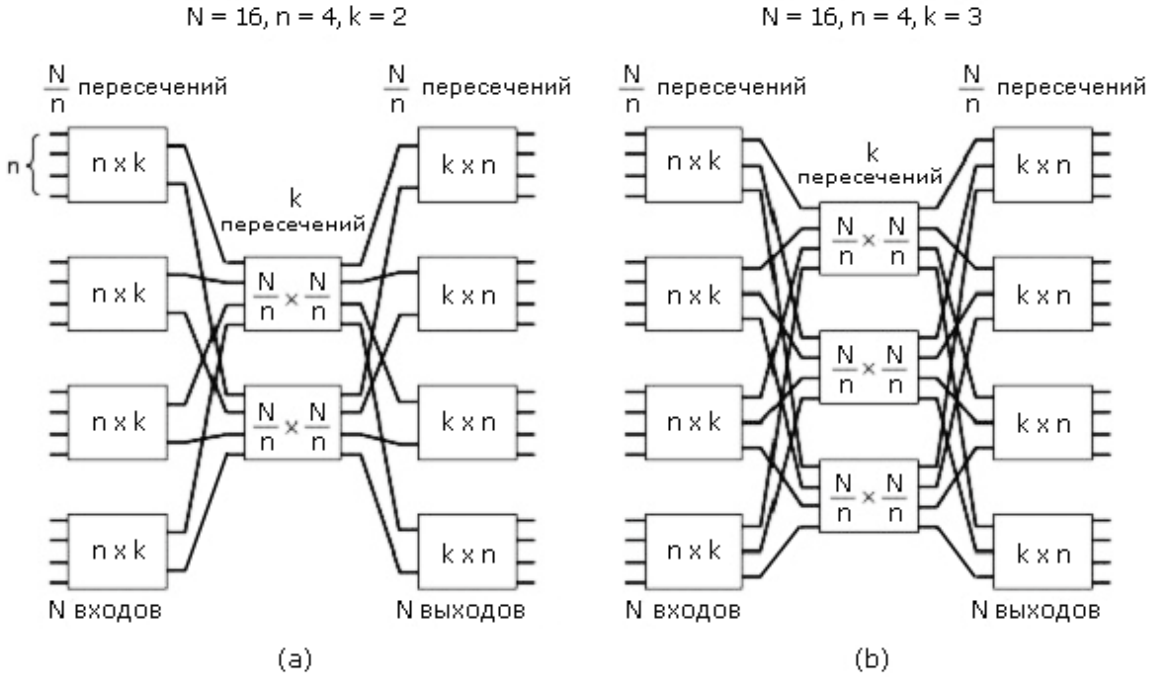
Подсчитаем сложность такого каскадного коммутатора. Первый каскад содержит

$\frac{N}{n} \times nk = Nk$ точек пересечения.

Второй каскад имеет $k\left(\frac{N}{n}\right)^2$ точек пересечения. Третий каскад по сложности такой же как и первый. Таким образом, получаем $2kN + k(N/n)^2$ точек пересечения.

При N=1000, n=50 и k=10 нам потребуется всего 24000 точек пересечения вместо 499500, как было бы при прямом коммутаторе.

Рисунок 2-53. Устройство каскадных коммутаторов



Каскадные коммутаторы имеют недостаток - блокировка коммутаторов второго слоя. На рисунке 2-53 (a) второй слой может коммутировать одновременно только 8 звонков. Девятый звонок буден заблокирован. Коммутатор на 2-53 (b) лучше. В нем 12 входов на втором каскаде, но он и дороже. Клос (Clos) в 1953 году показал, что при k=2n-1 блокировок в каскадных коммутаторах не будет.

Коммутаторы с разделением времени

Пусть у нас есть n линий, которые нам надо коммутировать. Эти линии сканируют последовательно одна за другой в течение определенного временного слота. Образуется кадр из n ячеек по k битов в каждой. Например, в стандарте E1 каждая ячейка содержит по 8 бит, кадр – 32 ячейки, а всего за секунду проходит 8000 кадров.

Рисунок 2-54. Коммутатор с разделением времени

Затем кадр попадает в коммутатор ячеек. Коммутатор ячеек переставляет ячейки в соответствии с таблицей коммутации. Обработка кадра происходит следующим образом. Входной кадр записывается в память в том

порядке, как ячейки считывались с линий. Затем ячейка считываются из памяти в порядке, задаваемом таблицей коммутации.

15.Цифровые сети с интегрированным сервисом (ISDN сети).

Цифровые сети с интегрированным сервисом (ISDN)

принято решение о создании новой полностью цифровой телекоммуникационной сети, которой дали название «Цифровая сеть с интегрированным сервисом» (ISDN - Integrated Service Digital Network). ISDN задумывалась как всемирная телекоммуникационная сеть, которая должна была заменить телефонные сети. С точки зрения приложений, ISDN должна была поддерживать передачу голоса, звука, изображения и данных. ISDN-телефон по замыслу проекта должен был обеспечивать самый разнообразный сервис: программируемые функции, показ номера телефона, от которого поступил звонок, имя звонящего, умение работать с компьютером - выдать запрос к базе данных и высветить на экране ответ, переадресовать звонки, удаленный доступ к своему телефону, автоматические звонки в скорую помощь, полицию, пожарную службу в случае опасности и т.д. Эта технология должна обеспечивать подключение прямо в сеть, без использования модемов, цифровые приборы и оборудование. Проект ISDN постоянно находится в развитии. Он оказывает огромное влияние как на операторов связи, так и на производителей оборудования. В рамках проекта ISDN значительные усилия сосредоточены на стандартизации интерфейсов разных уровней. Несмотря на то что ISDN еще не достиг того же уровня распространения, как обычный телефон, уже появилось второе поколение этого проекта. Первое поколение называют narrowband ISDN – узкополосный, или низкоскоростной ISDN (N-ISDN). Он поддерживает аналоговые и цифровые каналы с пропускной способностью 64 Кбит/сек. и основан на коммутации каналов. Одним из важных технических новшеств N-ISDN стал метод передачи Frame Relay. Второе поколение ISDN, называемое broadband ISDN, – широкополосный, или высокоскоростной ISDN, поддерживает высокую скорость передачи данных (сотни Мбит/сек.) и функционирует на основе коммутации пакетов. Одним из основных технических новшеств B-ISDN стал асинхронный метод передачи (ATM).

Принципы ISDN

Принципы ISDN были определены МСС (бывшей МКТТ) и опубликованы в рекомендации I.120 в 1993 году. Они приведены ниже:

1. Поддержка голосовых и неголосовых приложений с использованием определенного набора стандартизированных средств. Этот принцип определяет цели ISDN и средства их достижения. ISDN поддерживает разнообразные сервисы, как голосовую связь (телефон), так и неголосовую (обмен данными в цифровой форме). Эти сервисы предоставляются в строгом соответствии со стандартами МСС, которые определяют интерфейсы и виды передачи данных.

2. Поддержка как коммутируемых, так и некоммутируемых приложений. ISDN использует коммутацию каналов и коммутацию пакетов. Также ISDN поддерживает некоммутируемые приложения, использующие выделенные линии.

3. Основа на соединениях 64 Кбит/сек. ISDN-соединения, основанные как на коммутируемых каналах, так и на коммутации пакетов, должны обеспечивать скорость передачи в 64 Кбит/сек. Это один из основных конструктивных элементов ISDN. Эта скорость была выбрана потому, что она была стандартной для передачи голоса в оцифрованной форме и поддерживалась интегрированными цифровыми сетями (Integrated Digital Network – IDN). Однако очень скоро оказалось, что этой скорости недостаточно. Второе поколение ISDN – B-ISDN обеспечивает большую гибкость.

4. Интеллектуальные сети. ISDN должна поддерживать сервис высокого уровня: например, выполнять переадресацию звонков, автоматически определять разные виды терминалов.

5. Уровневая архитектура. Протоколы доступа в ISDN-сеть должны иметь уровневую архитектуру, соответствующую OSI-модели. Этим обеспечивается целый ряд преимуществ:

   - Для OSI-приложений уже создано много стандартов. Пример - HDLC, уровень 3 в стандарте X.25 для доступа к сервису с коммутацией пакетов в ISDN.
   - Новые ISDN-стандарты могут быть основаны на уже существующих стандартах, тем самым сокращается стоимость их реализации.
   - Стандарты разных уровней можно независимо развивать и реализовывать.

6. Разнообразие конфигураций. Реализация ISDN предполагает разнообразные физические конфигурации. Это обеспечивает приспособляемость ISDN к различиям в государственной политике, уровням технологий, имеющемуся оборудованию.

Архитектура сетей N-ISDN

Основой ISDN-архитектуры является концепция битового потока в цифровом тракте или просто цифрового тракта между пользователем и транспортной средой, через которую поток битов передается. При этом не важно, как был сформирован этот поток битов - телефоном, факс-машиной, компьютером и т.п. Важно, что биты можно передавать по тракту в обоих направлениях. Цифровые тракты могут мультиплексировать с разделением по времени несколько независимых каналов. Концепция цифрового тракта строго специфицирована. В этой спецификации определены интерфейсы, формат цифрового потока и правила мультиплексирования потоков. Было разработано два стандарта: один для низкоскоростной передачи (для домашнего использования) и высокоскоростной (для бизнес приложений). На рисунке 2-59 (a) показаны основные конфигурации для дома или небольшой организации. Поставщик сервиса, или, как его еще называют, оператор, устанавливает оконечное сетевое устройство - NT1. NT1 соединено, с одной стороны, с ISDN-оборудованием пользователя, а с другой - с ISDN-устройством обмена в помещении поставщика сервиса. NT1 может быть удалено от ISDN-устройства обмена на несколько километров и соединено с ним витой парой, оставшейся от обычного телефонного соединения. К одному NT1 может быть подключено до 8 ISDN-устройств пользователя. С точки зрения пользователя, граница сети передачи данных – NT1-устройство. Для производственных нужд конфигурация 2-59(a) не подходит, так как может потребоваться существенно больше оконечных ISDN-устройств, функционирующих одновременно, например, телефонов. Поэтому в промышленности используется конфигурация, представленная на рисунке 2-59(b). В этой конфигурации используется устройство NT2 - PBX (Private Branch eXchange), которое мы будем называть устройством обмена второго уровня. PBX соединен с NT1 и обеспечивает связь с телефонами, терминалами в офисе и их мультиплексирование. Таким образом, PBX - это по существу небольшой ISDN-коммутатор. МКТТ определило четыре вида точек подключения для ISDN-сетей: R, S, T, U. U-соединение определяет соединение между ISDN-устройством обмена и NT1. На сегодня это либо медная витая пара, либо оптоволоконная линия. T - определяет подключение NT1 к оборудованию в офисе пользователя. S - подключение PBX- и ISDN-терминалов. R - адаптер между ISDN-терминалом и не-ISDN оборудованием. Подключение типа T позволяет подключить 23 канала по 64 Кбит/сек., что хорошо укладывается в стандарт T1 в США и Японии, и 30 каналов по 64 Кбит/сек. для Европы. Однако надо подчеркнуть, что для одного N-ISDN терминала доступна скорость не более 64 Кбит/сек. Битовый тракт в ISDN подразумевает мультиплексирование нескольких стандартных каналов. Стандарты ISDN определяют следующие типы каналов:

- A – 4 КГц, аналоговый телефонный канал
- B – 64 Кбит/сек., цифровой канал с импульсно-кодовой модуляцией для голоса или данных
- D – 16 или 64 Кбит/сек., цифровой канал
- H – 384 (H0), 1536 (H11), 1920 (H12) Кбит/сек., цифровой канал

Канал типа B подразумевает четыре вида соединений:

- С коммутацией каналов. Абонент инициирует вызов, под воздействием которого устанавливается соединение с коммутацией каналов, которое соединяет абонента с другим абонентом сети.
- С коммутацией пакетов. Абонент подключен к узлу сети с коммутацией пакетов и обменивается данными с другими абонентами посредством протоколов X.25.

- Соединение Frame Relay. Абонент подсоединяется к узлу сети Frame Relay, через которую происходит обмен данными.
- Постоянное соединение. Это соединение с другим абонентом, которое было установлено заранее и динамически изменено быть не может. Это соединение подобно выделенной линии.

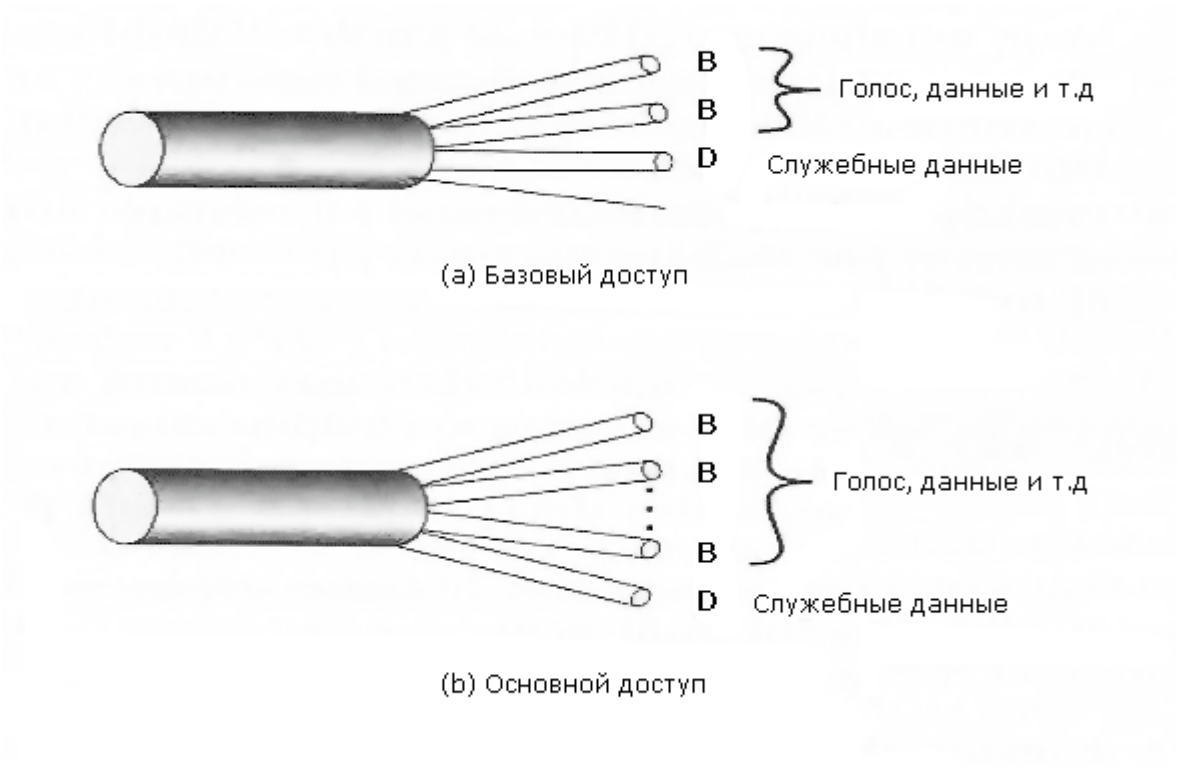
Канал типа D служит двум целям. Во-первых, он служит для управления коммутацией каналов, инициированной вызовом по интерфейсу, с абонентом через канал B. Кроме этого, канал D можно использовать, когда он свободен, для коммутации пакетов или получения данных от оборудования на низкой скорости (до 100 бит/сек.).

Каналы типа H служат для высокоскоростной передачи данных. Абонент может использовать такой канал как высокоскоростную магистраль, либо разделить ее с помощью метода TDM на подканалы. Обычно канал этого типа используют такие приложения, как факс, видео, высококачественные звуковые устройства.

Эти каналы объединяют в так называемые структуры передачи, или канальные структуры. На сегодня лучше всего определена и часто используется базовая канальная структура (BRI - Basic Rate Interface) или базовый доступ (BA) и основная канальная структура (или основной доступ (PA)).

На рисунке 2-60 показаны эти структуры. Базовый доступ состоит из двух полнодуплексных B-каналов 64 Кбит/сек. и одного полнодуплексного D-канала 16 Кбит/сек. Базовый доступ обеспечивает максимальную скорость 192 Кбит/сек.

Рисунок 2-60. Структура ISDN-каналов



Основной доступ предназначен для пользователей, которым нужна высокая скорость передачи. Как видно на рисунке, есть несколько вариантов основного доступа: для поддержки стандарта T1 и для поддержки стандарта E1. (Эти стандарты мы обсуждали в разделе 2.5.5.3).

ISDN-сети предоставляют четыре вида соединений конечных пользователей:

- с коммутацией каналов через канал B
- через канал B

- с коммутацией пакетов через канал B
- с коммутацией пакетов через канал D

При установлении соединений с коммутацией пакетов используют как каналы B, так и каналы D. При подключении через канал B пользователи могут использовать любой протокол обмена. Канал D используют для передачи управляющей информации между пользователем и сетью при установлении, разрыве соединения, доступе к сетевым сервисам.

Канал B подключают через устройство NT1 или NT2, используя протоколы физического уровня. Канал D предполагает использование трехуровневого протокола доступа, например, X.25.

Постоянное соединение может быть предоставлено на неопределенное время, предопределенный период, либо выделенные дни, недели, месяцы. Сетевой интерфейс поддерживает только физический уровень. Управление вызовом не нужно, так как соединение уже предоставлено.

ISDN-сети также должны предоставлять доступ к передаче данных через соединения с коммутацией пакетов. Для этого есть две возможности. Либо это обеспечивает внешняя сеть, называемая сетью передачи данных общего доступа с коммутацией пакетов (Packet-Switched Public Data Network – PSPDN), либо возможность коммутации пакетов интегрируется в ISDN-сеть. В первом случае сервис обеспечивается через B-канал, во втором – либо через B-канал, либо через D-канал. Начнем рассмотрение этих случаев с использования B-канала для доступа к сервису с коммутации пакетов.

Когда сервис с коммутацией пакетов осуществляется с помощью внешней PSPDN-сети, доступ к этому сервису обеспечивается через B-канал. Как пользователь, так и PSPDN-сеть должны в этом случае быть абонентами ISDN-сети. В этом случае один или несколько узлов PSPDN-сети, называемых PH-узлами (Packet Handler), должен быть соединен с ISDN-сетью. Эти узлы можно считать обычными устройствами X.25 DCE с возможностью подключения к ISDN-сети. В этом случае абонент ISDN-сети – это X.25 DTE, и ISDN-сеть просто соединяет X.25 DTE с X.25 DCE, которое одновременно является узлом PSPDN-сети.

Теперь любой абонент ISDN-сети может обмениваться данными через X.25 с любым абонентом PSPDN-сети.

16.Передача данных в ATM сетях.

ATM - технология с коммутацией пакетов. В области коммутации каналов накоплен огромный опыт, поэтому переход на коммутацию пакетов - это технологический, принципиальный сдвиг. Ясно, что для B-ISDN витая пара – основной вид абонентской линии, скорее всего, не подойдет. Существующие телефонные коммутаторы не годятся и должны быть заменены коммутаторами нового поколения, работающими на иных принципах. Единственное, что, похоже, удастся сохранить - оптоволоконные магистрали.

Виртуальные каналы и коммутация каналов

Передача в ATM-сетях

Как уже было сказано ATM - это асинхронный способ передачи. В стандарте T1 данные передаются строго синхронно, так, как показано на рисунке 2-64. Каждые 125 мксек порождается новый кадр. Эта скорость поддерживается специальными часами - мастер-таймером. Каждый слот в кадре содержит один бит из определенного источника. Порядок сканирования источников строго фиксирован.

Рисунок 2-64. Синхронный и асинхронный способы передачи

В ATM нет строго порядка поступления ячеек от различных источников. Пример потока ATM-ячеек показан на рисунке 2-64(b). Ячейки могут поступать от разных источников и в разном порядке. Не важно даже, чтобы поток ячеек от одного компьютера был непрерывен. Если возникают разрывы, то они заполняются ячейками ожидания.

В ATM не стандартизован формат самой ячейки. Требуется только, чтобы ячейки могли передаваться носителями (кадрами, фреймами и т.п.) в рамках таких стандартов, как T1, T3, E1, SONET, FDDI и т.п.

В настоящее время скорость 155,52 Мбит/сек. является стандартной для ATM, равно как и учетверенная скорость - 622,08 Мбит/сек. Однако в ближайшем будущем ожидается достижение 44 736 Мбит/сек.

Стандартной средой передачи для ATM является оптоволокно. Однако на расстояниях в сотни метров можно использовать коаксиал или витую пару 5-й категории. Оптоволокно может покрывать расстояния на многие километры. Каждая волоконно-оптическая линия соединяет либо компьютер с ATM-переключателем, либо два ATM-переключателя. ATM-линии – это соединения типа «точка-точка». На одной линии не может находиться более одного источника ячеек. По каждой линии передача возможна только в одном направлении, поэтому для обеспечения полного дуплекса нужны две ATM-линии. С помощью ATM-переключателей возможно дублирование одной и той же ячейки для передачи этой ячейки по нескольким линиям. Так реализуют режим вещания, т.е. передачу от одного ко многим.

Подуровень сопряжения с физической средой (PMD) в стандарте ATM обеспечивает съем битов с линии и передачу их на линию. Для физически разных линий (коаксиал, оптоволокно и т.п.) используют разное оборудование. Подуровень преобразования при передаче (TC) обеспечивает единый интерфейс с ATM-уровнем при передаче ячеек в обоих направлениях. Именно TC-подуровень обеспечивает сопряжение ATM-уровня с протоколом передачи в выбранной среде, например, в случае SONET это будет интерфейс STS-3, поддерживающий скорость 155,52 Мбит/сек. ATM-уровень обеспечивает поток ячеек, а PMD-подуровень преобразует их в поток битов в физической среде.

При входящем потоке PMD-подуровень передает поток битов на TC-подуровень. Задача TC-подуровня - определить, где кончается одна ячейка, а где начинается другая. Поскольку в поступающем потоке битов нет никаких признаков деления между ячейками, то это весьма сложная задача. Как она решается, мы рассмотрим в разделе, посвященном канальному уровню, поскольку именно канальный уровень отвечает за преобразование потока битов в поток кадров или ячеек.

ATM-переключатели

Здесь мы рассмотрим основные принципы организации ATM переключателей и их функционирования.

На рисунке 2-65 показана общая схема организации ATM-переключателя. Есть набор входных линий, по которым ячейки поступают в переключатель, и, как правило, такое же число выходных линий, по которым ячейки двигаются после коммутации. Обычно переключатель работает синхронно: длительность цикла строго фиксирована. В течение каждого цикла просматриваются все входные линии и, если на линию к этому

моменту целиком поступила ячейка, то она считывается и передается в центр коммутации, а затем на выходную линию.

Переключатель может быть конвейерным, т.е. обработка одной ячейки может занимать более одного цикла. Ячейки поступают асинхронно, т.е. таймер переключателя отмечает момент начала очередного цикла. Если ячейка не поступила целиком за один цикл, то она должна ожидать начала следующего цикла.

Все АТМ-переключатели должны удовлетворять следующим требованиям:

- терять как можно меньше ячеек
- никогда не менять порядок поступления ячеек по каждому виртуальному соединению

Первое требование означает, что АТМ-переключатель должен обеспечивать достаточно большую скорость переключения, но так, чтобы не терять ячейки. Считается допустимой потеря 1 ячейки на каждые 1012. В больших переключателях считается допустимой потеря 1-2 ячеек за час работы. Второе требование - сохранять порядок поступления ячеек неизменным - существенно усложняет конструкцию переключателя, но таково требование АТМ-стандарта.

Одна из ключевых проблем конструкции АТМ-переключателей состоит в следующем: что делать, когда сразу по нескольким линиям пришли ячейки, которые должны быть отправлены по одной и той же выходной линии? Напрашивается решение: взять одну ячейку, обработать ее, а другую сбросить. Но в силу требования 1 оно не годится. Возможно другое решение: буферизовать ячейки на входе. Пусть в начале цикла 1 (рисунок 2-66(a)) поступило четыре ячейки, две из которых должны быть отправлены по линии 2. Поскольку из-за линии 2 возник конфликт, то только три ячейки передаются на выходные линии. Поэтому к началу цикла 2 (рисунок 2-66(b)) на выходе переключателя появятся три ячейки, но на вход поступят новые. К началу цикла 3 (рисунок 2-66(c)) на входе останется только одна ячейка, и очередь рассосется только на четвертом цикле. В случае буферизации на входе надо следить за тем, чтобы дисциплина обслуживания возникающих очередей была бы справедливой и равномерно обслуживала очереди на всех линиях. Недостаток этого решения в том, что очередь на входе может блокировать даже те ячейки, которые должны быть перекоммутированы на линии, на которых нет конфликтов. Поэтому по соответствующему виртуальному соединению скорость упадет. Этот эффект называется блокировкой на входе. Кроме этого, буферизация ячейки на входе требует дополнительной логики в схемах, что усложняет конструкцию АТМ-переключателя. Альтернативным решением может быть буферизация на выходе. Это решение показано на рисунке 2-67. Если несколько ячеек должны уйти по одной и той же линии, то они передаются на выход и буферизуются там. Это требует меньше циклов, в нашем примере только 3. В общем случае Karol 1987 показал, что буферизация на выходе эффективнее, чем буферизация на входе. Рассмотрим конструкцию АТМ-переключателя, использующего буферизацию на выходе. Этот тип переключателей называется переключатель выталкивающего типа. Он показан на рисунке 2-68 для конфигурации 8x8 линий. Здесь каждая входная линия соединена с шиной, к которой подключены все выходные линии. Каждая входная шина имеет свой механизм управления, не зависящий от других, что существенно упрощает конструкцию. У каждой поступающей ячейки аппаратно анализируется заголовок, чтобы определить, какому виртуальному соединению она принадлежит. Затем, с помощью таблицы коммутации, определяется выходная линия, через которую эта ячейка должна покинуть переключатель. Пересечение с соответствующей выходной линией активизируется, и, когда ячейка доходит до этого пересечения, она попадает в буфер. Ресурсов переключателя достаточно, чтобы буферизовать на одном выходе ячейки со всех входов, если это потребуется, или размножить ячейки, если их надо разослать по нескольким виртуальным соединениям. Естественно было бы буферизовать все конфликтующие ячейки в выходном буфере. Однако для переключателей, например, на 1024 линий, нам потребовалось бы 1024 буферов по 1024x53 байтов. Слишком много! Выход из этой ситуации - выделение лишь n байтов на буфер, где n – параметр настройки. Если конфликтующих ячеек больше, то ячейки, не попавшие в буфер, сбрасываются. Здесь опять-таки надо быть осторожным, определяя на каких входных линиях сбрасывать ячейки, из каких выходных буферов выталкивать ячейки на очередном цикле так, чтобы не было дискриминации. Регулируя параметр n, можно варьировать стоимость и число сбрасываемых ячеек, что влияет на цену переключателя.

Переключатели Батчера-Баньяна

Основным недостатком переключателей выталкивающего типа является то, что центр коммутации - простой коммутатор, а это означает, что его сложность растет квадратично от числа коммутируемых линий. Из рассмотрения принципов построения коммутаторов для коммутации каналов мы уже знаем, что одно из решений - каскадные коммутаторы. Аналогичное решение возможно и для коммутации пакетов. Это решение называют переключателем Батчера-Баньяна. Как и переключатели выталкивающего типа, переключатель Батчера-Баньяна синхронный, т.е. за один цикл он может обрабатывать несколько входных линий. Он называется так, поскольку похож на корни баньянового дерева. В баньяновых переключателях для каждого входа существует ровно один путь к любому из выходов. Маршрутизация пакета происходит в каждом узле на основе адреса выходной линии, которой должен достичь пакет. Адрес выходной линии определяют на входе по номеру виртуального соединения. В данном случае трехбитовый номер впереди ячейки используется в каждом узле для маршрутизации. В каждом из 12 переключающих элементов есть два входа и два выхода. В зависимости от значения соответствующего разряда ячейка направляется либо в порт 0, либо в порт 1. Если обе ячейки, поступившие на вход одного и того же коммутирующего элемента, должны быть направлены на один и тот же порт, то направляется одна, а вторая сбрасывается. Коллизии в баньяновской сети возникают, когда в одном и том же элементе в одно и тоже время надо использовать один и тот же порт. На рисунке 2-70 (а) показаны коллизии. Идея Батчера состояла в том, чтобы переставить ячейки на входах так, чтобы в баньяновской сети конфликтов не возникало. Для сортировки входов Батчер в 1968 году предложил специальный переключатель. Подобно баньяновскому переключателю, переключатель Батчера строится из элементов 2x2, работает синхронно и дискретно. В каждом элементе выходные адреса ячеек сравниваются. Больший направляется по стрелке, а меньший - в противоположном направлении. Если ячейка одна, то против стрелки. Подчеркнем, что сравниваются не отдельные биты, а весь адрес как число.

Известны две трудности, которые переключатели Батчера-Баньяна не могут преодолевать:

- если коллизия на выходе все-таки возникает, то решением является только сброс
- рассылка одной и той же ячейки сразу на несколько выходов

27.Сотовая связь: пейджинг, сотовые и радиотелефоны (система AMPS, GSM, GPRS, UMTS, CDMA).

Paging

Примером paging-связи (от page - страница, листок) является громкоговорящая связь на крупных предприятиях (больницы, автобазы, депо, вокзалы и т.д.), т.е. это системы однонаправленные. Современное развитие этого вида систем состояло в адресации сообщения кому-то одному, а не всем, кто находится в зоне слышимости.

Желающий послать сообщение на пейджер звонит в пейджинговую компанию, называет код абонента и текст сообщения. Оператор вводит сообщение в систему и компьютер через сеть передает это сообщение, которое через ретранслятор передается в эфир. Пейджер получателя, обнаружив в тексте сообщения свой номер, принимает сообщение, запоминает его в буфере и высвечивает его на экране. Подобную услугу имеют все современные сотовые телефоны.

Эти системы однонаправленные, от одного ко многим, поэтому в них нет проблем с конкуренцией за доступ к каналу передачи. В системе есть только один передатчик.

Развитая мобильная телефонная система - AMPS

Ситуация с мобильной телефонной связью резко изменилась, когда в 1982 году компания Bell Labs предложила систему AMPS (Advanced Mobil Telephone System). Идея этой системы очень проста. Вместо того чтобы охватить сразу всю территорию небольшим числом каналов, эту территорию разбивают на небольшие части – соты. В каждой соте используют свой набор каналов, но так, чтобы частоты каналов у соседних сот не пересекались, т.е. не было общих частот. Такая организация системы дает выигрыш в использовании частот из-за их повторного использования, увеличивается емкость сети – число одновременно обслуживаемых пользователей. Кроме этого, в системе можно использовать маломощные сигналы, а следовательно, передатчик может быть компактным, т.к. не требуется мощных источников питания. Если в каких-то сотах из-за большого числа пользователей отказы в соединении становятся слишком частыми из-за большого числа пользователей, то эту соту можно разделить на несколько новых.

Каждая сота имеет базовую станцию (базу), состоящую из компьютера и приемно-передающей аппаратуры. Несколько баз подключаются к Центру мобильной коммутации (MSC). В небольших системах может быть достаточно одного центра. В больших системах может потребоваться несколько центров. MSC-центры соединяются друг с другом и с обычной наземной телефонной сетью и, при необходимости, коммутируют звонок с мобильного телефона на обычный телефон.

В системе AMPS используется метод разделения частот - FDMA. Весь диапазон частот 824-894 МГц разделены на 832 дуплексных канала: 824-859 МГц для передачи и 860–894 МГц - для приема. Каждый канал имеет ширину 30 КГц. Все каналы делятся на четыре категории:

- Управляющие
- Для сообщений
- Установки доступа и распределения каналов
- Данные - голос, факс и прочие

В системе AMPS у каждого телефона есть встроенный 32-битовый серийный номер и телефонный номер, состоящий из 10 цифр: 3 цифры – код зоны (10 бит) и 7 цифр (24 бита) – номер абонента. Когда телефон включают, он начинает сканировать запрограммированный в нем список из 21 каналов управления, чтобы обнаружить наиболее мощный сигнал. По информации из управляющего канала он узнает распределение каналов для сообщений, установки соединений и доступа, передачи данных. Затем телефон сообщает свой 32-битовый серийный номер и 34-битовый телефонный номер. Эта информация в AMPS-системе передается пакетом в цифровом виде несколько раз, кодируется специальным кодом с коррекцией ошибок, хотя голос передают по аналоговому каналу. Когда базовая станция получает такой пакет от телефона, она запрашивает у своего MSC-центра информацию о новом клиенте и сообщает домашней MSC, т.е. MSC, к которой приписан этот телефон, о его текущем местоположении. Обычно такая перерегистрация телефона происходит каждые 15 минут. Чтобы позвонить, абонент включает телефон, набирает номер нужного абонента и нажимает кнопку «Послать» (Send). Телефон по каналу установки доступа посылает в цифровом виде пакет, содержащий информацию о нем и о телефоне вызываемого абонента. Если происходит коллизия или ошибка, то попытка повторяется несколько раз. Получив запрос, базовая станция информирует о нем MSC. Если нужный абонент – это абонент компании, которой принадлежит MSC, то MSC ищет свободный канал для данных. Если такой найден, то MSC информирует о нем вызывающий телефон по каналу управления. Вызывающий телефон переключается на прием по указанному каналу и ждет, когда на вызываемом телефоне поднимут трубку (нажмут кнопку «Прием»). Входящий звонок обрабатывается несколько иначе. В режиме ожидания телефон постоянно следит за каналом сообщений: не появится ли там сообщение для него. Когда вызывающий телефон сгенерировал запрос, то от MSC поступает запрос на домашнюю MSC вызываемого телефона, чтобы определить, в какой соте находится вызываемый телефон. Пакет с вызовом направляется последней базовой станции, зарегистрировавшей телефон с искомым номером, например, 46. Базовая станция распространяет по каналу сообщений специальное сообщение типа: «46-й, ты здесь?» Вызываемый телефон отвечает по каналу управления специальным пакетом типа «Да». Тогда базовая станция шлет по каналу управления пакет «46-ой, для вас вызов на канале 8». К сожалению, аналоговые сотовые телефоны абсолютно не защищены. Любой, у кого есть радиоприемник нужного диапазона, может, настроив его на один из

голосовых каналов, просто прослушать разговор. Злоумышленник может перехватывать информацию из каналов управления, содержащую 32-битовые номера телефонных трубок и 34-битовые номера, а затем разговаривать за чужой счет. И многое, многое другое. Это один из главных недостатков аналоговых сотовых телефонов.

Цифровая сотовая телефония

Итак, GSM - это полностью цифровая система. Ее успех был во многом связан с тем, что она проектировалась без оглядки на уже существующие аналоговые системы, ее авторы не пытались сделать ее совместимой с ними. Основная цель стандарта GSM была обеспечить людям возможность, свободно передвигаясь, как внутри страны, так и между странами, поддерживать связь с любыми абонентами сети. При этом в каждой стране может быть одна или несколько функционирующих сетей. Каждая такая сеть называется Региональной мобильной сетью оператора (PLMN). Зона действия каждой PLMN-сети ограничена национальными границами, в одной стране, впрочем, может быть несколько PLMN-сетей. GSM-пользователь заключает контракт с одной из PLMN-сетей, называемой домашней. В этом контракте указаны услуги, доступные этому пользователю. При желании во время работы пользователь может выбрать другую PLMN-сеть, **если ему доступны ее услуги**. Терминал пользователя (в GSM его называют мобильной станцией – MS) обеспечивает пользователю такой выбор и показывает список доступных PLMN-сетей. Выбор из этого списка пользователь может сделать сам явно, или MS-терминал сделает это автоматически с помощью заложенного в нее программного обеспечения. Как и в AMPS-системе, в GSM территория разбивается на области, обслуживаемые Центром Мобильной Коммутации (MSC). Оператор PLMN-сети абсолютно свободен в разбиении области действия MSC-станции на соты. У каждой PLMN-сети есть логически единая база данных, называемая Home Location Registers (HLR), где хранится информация обо всех пользователях, для которых эта PLMN-сеть домашняя. Физически HLR-база может быть распределенной. У каждой MSC-станции есть база данных визитеров – Visitor Location Registers (VLR). Одна VLR-база обычно обслуживает одну MSC-станцию, но может обслуживать и несколько. HLR- и VLR-базы данных обеспечивают отслеживание текущего местонахождения каждого MS-терминала, находящегося в зоне действия MSC-станции, запрашиваемых услуг и т.д. Мобильная станция GSM, в просторечии «трубка», разделяется на две части. Одна обеспечивает радиоинтерфейс, другая - интерфейс с базами HLR и VLR и содержит информацию, идентифицирующую пользователя (Subscriber Identify Module - SIM). SIM-карта идентифицирует пользователя, а не MS-терминал. Поэтому она может быть вынута из одного MS-терминала и вставлена в другой. Каждая SIM-карта уникальна в системе GSM и связана с идентификатором IMSI (International Mobil System Identify). На этой карте хранится идентификационная информация, список услуг, список выбираемых PLMN-сетей и т.п. Она защищена паролем (PIN – Personal Identification Number). Вставив свою SIM-карту в трубку, пользователь тем самым персонифицирует ее. Благодаря SIM-карте поддерживается роуминг, т.е. доступ к услугам связи в чужую PLMN-сеть.

GPRS-служба

Вполне естественно возникновение идеи по применению GSM-сетей для организации связи между компьютерами. Одним из существенных недостатков сетей сотовой связи стандарта GSM на сегодняшний день является **низкая скорость передачи данных (максимум 9,6 кбит/сек.)** по одному каналу. Для передачи данных абоненту выделяется всего один голосовой канал, а оплата осуществляется, исходя из времени соединения (причем по тарифам, мало отличающимся от голосовых). Для высокоскоростной передачи данных посредством существующих GSM-сетей была разработана GPRS (General Packet Radio Service) - служба пакетной передачи данных по радиоканалу. Необходимо отметить, что, кроме повышения скорости (максимум составляет 171,2 кбит/сек.), новая система предполагает иную схему оплаты услуги передачи данных - при использовании GPRS-службы расчеты производятся пропорционально объему переданной информации, а не времени использования канала.

Стандарт услуги GPRS предусматривает два режима соединений:

- PTP (Point-To-Point - точка-точка)
- PTM (Point-To-Multipoint - точка-многоточка)

Широковещательный режим PTM, в свою очередь, подразделяется на два класса:

- PTM-M (PTM-Multicast) - передача необходимой информации всем пользователям, находящимся в определенной географической зоне;
- PTM-G (PTM-Group Call) - данные направляются определенной группе пользователей.

Новый стандарт для 3G-сетей

Прежде чем мы перейдем к рассмотрению стандартов для 3G-сетей – сетей третьего поколения, следует упомянуть стандарт IS-95, в котором используется принципиально иной, по сравнению с AMPS- или GSM-системами, метод доступа. Этот метод называют разделением кодов – CDMA (Code Division Multiple Access), и он не совместим с методами, используемыми в AMPS- и GSM-системах. Мы подробно рассмотрим этот метод в разделе 4. Следующим шагом от GSM к сетям третьего поколения (3G-сети) или UMTS-системам (Universal Mobile Telephone System) является EDGE-служба (Enhanced Data Rates for GSM Evolution, в вольном переводе - «ускоренная передача данных»), позволяющая осуществлять передачу информации на скоростях до 384 кбит/сек. в восьми GSM-каналах (48 Кбит/сек. на канал). С EDGE-службой мобильный Интернет становится реальностью. Добавление EDGE-службы к существующим сетям второго поколения делает их совместимыми со стандартами ITU для 3G-сетей. EDGE-служба – это решение для 3G-сетей, которое позволит существующей сетевой инфраструктуре предоставлять мощные современные мультимедийные услуги для мобильных терминалов. Реализация EDGE позволяет усилить и основные преимущества технологии GPRS-службы: быстрое установление соединений пакетной передачи и более высокая скорость в радиоинтерфейсе. Для внедрения EDGE-службы «поверх GPRS» операторам необходимо заменить аппаратуру базовых станций BS, а пользователям - приобрести поддерживающие EDGE телефонные аппараты. Хотя на настоящий момент сложно представить, какие приложения должен использовать абонент сотовой сети GSM, чтобы ему не хватало скорости в 170 кбит/сек., предлагаемой GPRS. Но в наше время бурно развивающихся цифровых технологий прогнозы - дело неблагодарное…

UMTS (Universal Mobile Telecommunications System) - Универсальная система мобильных телекоммуникаций – это один из стандартов, разрабатываемый Европейским институтом стандартов телекоммуникаций (ETSI) для внедрения 3G-сетей в Европе. Сегодня основным фактором, определяющим развитие мобильной связи, является голосовая телефония. Появление GPRS и EDGE, а затем переход к UMTS-системе открывают дорогу ко многим дополнительным возможностям, помимо голосовой связи. UMTS - это высокоскоростная передача данных, мобильный Интернет, различные приложения на основе Интернета, интранета и мультимедиа. Ключевой технологией для UMTS является широкополосный многостанционный доступ с разделением кодов (WCDMA; технология CDMA будет рассмотрена в разделе 4). Эта революционная технология радиодоступа, выбранная в сентябре 1998 года Европейским институтом стандартов телекоммуникаций, поддерживает все мультимедийные услуги 3G-сетей. Системы WCDMA/UMTS включают усовершенствованную базовую сеть GSM и радиоинтерфейс по технологии WCDMA. Скорость передачи в радиоканале для мобильного абонента достигает 2 Мбит/сек. WCDMA предназначена для использования в системах, работающих в частотном диапазоне 2 ГГц, который позволит в полной мере использовать все преимущества этой технологии. Например, всего одна несущая WCDMA шириной 5 МГц должна обеспечить предоставление смешанных услуг, требующих скоростей передачи от 8 кбит/сек. до 2 Мбит/сек. А мобильные терминалы, совместимые с WCDMA, смогут в соответствии с рекомендациями ITU работать сразу с несколькими услугами.

Спутниковые системы связи

Идея создания системы связи на основе отражающего объекта, расположенного высоко над землей, давно витала в головах исследователей.

Геостационарные спутники

Согласно третьему закону Кеплера, период вращения спутника пропорционален радиусу орбиты в степени 3/2. На высоте примерно 36000 км над экватором период вращения спутника будет равен 24 часам. Такой спутник наблюдателю на экваторе будет казаться неподвижным. Благодаря этой неподвижности можно существенно упростить устройство наземной приемно-передающей антенной системы. Из-за интерференции волн неразумно было бы размещать такие спутники ближе, чем 2 градуса экваториальной плоскости друг от друга, если они работают на одинаковых частотах. Таким образом, в одно и тоже время на экваториальной

орбите может находиться не более 180 спутников, работающих на одной и той же частоте. Так как часть из этих орбит зарезервирована не только для целей связи, то спутников связи на самом деле меньше. Обычно спутник связи имеет 12-20 транспондеров с полосой пропускания 36-50 МГц каждый. Транспондер с пропускной способностью в 50 Мбит/сек. может быть использован для передачи одного потока данных на скорости 50 Мбит/сек., либо для передачи 800 телефонных разговоров на скорости 64 Кбит/сек. каждый, либо иначе комбинируя скорости и количество передаваемых потоков данных. За счет поляризации сигнала можно сделать так, что два транспондера смогут использовать одну и ту же частоту. Первые спутники связи имели один широкий луч. Современные имеют несколько более узких лучей, пятно которых охватывает несколько сот километров поверхности Земли. Относительно новой технологией является технология малых антенн, называемых VSAT (Very Small Aperture Terminals) - терминалов с очень маленькой апертурой, т.е. антенной с маленьким радиусом. Такой терминал имеет антенну с диаметром от 1,8 до 2,5 метра, способную излучать сигнал мощностью в 1 ватт. Он может передавать данные со скоростью примерно 19,2 Кбит/сек. и принимать - 512 Кбит/сек. Из-за малой мощности сигнала такие терминалы не могут взаимодействовать напрямую, но прекрасно могут это делать через специальный спутниковый хаб. Спутниковые системы связи имеют существенные отличия от наземных систем точка-точка. Несмотря на то что сигнал распространяется со скоростью света, из-за больших расстояний задержка при передаче велика - 250-300 мсек., против 3-5 мксек./км на коаксиале, оптоволокне и т.д. Спутниковые системы принципиально вещательного типа. Для некоторых приложений это очень важно. Стоимость передачи не зависит, скольким получателям сообщение предназначено. Однако проблема безопасности передаваемой информации здесь требует особого внимания - все слышат все, что передается. Решение этой проблемы - только шифрование. Стоимость передачи не зависит от расстояния. Такой способ передачи имеет очень низкий коэффициент ошибок при передаче.

Низкоорбитальные спутники

Изначально для целей передачи данных низколетящие спутники серьезно не рассматривались. Слишком быстро они проносились над определенным местом на поверхности Земли. В 1990 компания Моторола выдвинула проект системы низколетящих спутников. Идея была очень проста: когда пятно луча одного спутника уходило из определенного места, к этому месту подлетал другой спутник, пятно которого охватывало это место. Подлетевший спутник подхватывал передачу/прием, которую вел улетающий спутник, и связь сохранялась. Компания подсчитала, что для реализации этой идеи потребуется 77 спутников на высоте 750 км. Позднее, после уточнения параметров проекта, это число сократилось до 66. Этот проект получил название Иридиум (по названию 77-го элемента в таблице Менделеева). Основной целью этого проекта являлось обеспечение связи с наземными средствами, даже портативными, всей поверхности Земли. Этот проект вызвал ожесточенную конкуренцию со стороны других компаний. Все захотели строить низколетящие спутниковые системы. Было предложено множество других проектов, но все они похожи на Иридиум. Поэтому мы рассмотрим его. Вдоль меридиана на расстоянии 32 градуса располагаются 11 спутников, летящих на высоте 750 км. Таких ожерелий 6, они охватывают всю Землю. Каждый спутник имеет 48 пятен, так что 1628 пятен (сот) покрывают Землю (рисунок 2-74 (b)). Каждая сота имеет 174 дуплексных канала на частоте обычного сотового радиотелефона. Таким образом, во всем мире поддерживаются 283 272 канала. Некоторые из них используются для пейджинга и для навигации и не требуют большой пропускной способности. Прием и передача идут на частоте 1,6 ГГц, что позволяет использовать устройства, работающие от батарей. Если сообщение, принятое одним спутников, адресовано в область, покрываемую другим, то оно будет передано от одного спутника другому. На время оставим рассмотрение этого проекта.

Основные категории СЗ

Системы спутниковой связи, с точки зрения наземного терминального оборудования, можно условно развить на три вида. Первый - сети персональной спутниковой связи, такие как Iridium, Inmarsat, Globalstar и строящиеся ICO, Ellipso и Thuraya. Терминалы персональной связи существенно отличаются от своих старших собратьев – VSAT-станций. Они более компактны, универсальны, сопрягаются с сетями сотовой связи, а самое главное – работают при движении абонента. Вместе с тем персональная связь пока не способна обеспечить тот же комплекс и качество услуг, которые предоставляют VSAT-станции, да и тарифы в сетях персональной связи существенно выше.

Второй, наиболее многочисленный, связан с развитием корпоративных сетей, базирующихся на технологии VSAT, т.е. на использовании малогабаритных спутниковых терминалов с антеннами диаметром от 1,8 до 2,5 м. На сегодняшний день в мире насчитывается около 300 тыс. станций VSAT. Третий вид охватывает системы непосредственного телевизионного вещания, работающие главным образом в Ku-диапазоне частот (14/11 ГГц), что позволяет использовать на приеме малые земные станции, стоимость которых не превышает 500 долл. Этот вид спутникового вещания ориентируется в первую очередь на сельское население и малые города со слаборазвитой кабельной инфраструктурой. Именно эта категория составляет большую часть населения России. Далее мы подробно рассмотрим каждый из вышеперечисленных видов сетей.

Проблемы передачи данных на канальном уровне (Сервис, предоставляемый сетевому уровню, Разбиение на кадры, Контроль ошибок, Управление потоком). Простейшие протоколы канала данных (Симплекс протокол без ограничений, Симплекс старт стопный протокол, Симплексный протокол для канала с шумом).

На уровне канала данных решается ряд проблем, присущих только этому уровню: реализация сервиса для сетевого уровня, объединение битов, поступающих с физического уровня в кадры, обработка ошибок передачи, управление потоком кадров и другие. Основная задача канального уровня - обеспечить сервис сетевому уровню по передаче и приему данных. Назначение этого сервиса - передать данные от процесса на сетевом уровне одной машины процессу на сетевой уровень другой машины. Однако для простоты изложения мы будем придерживаться первой схемы. Канальный уровень может обеспечивать различный сервис. Хотя этот сервис может варьироваться от системы к системе, есть три общих видов сервиса:

1. Сервис без уведомления и без соединения
2. Сервис с уведомлением и без соединения
3. Сервис с уведомлением и с соединением

Сервис без уведомления и без соединения не предполагает, что до начала передачи должно быть установлено соединение, которое после передачи должно быть разорвано, что факт приема переданного кадра должен подтверждаться специальным сообщением. Если в результате помех на физическом уровне кадр будет потерян, то никаких попыток его восстановить на канальном уровне произведено не будет. Следующий вид сервиса – сервис с уведомлением без соединения. В этом виде сервиса получение каждого посланного кадра должно быть подтверждено. Если подтверждения не пришло в течение определенного промежутка времени, то считают, что кадр не принят и должен быть послан опять. Наиболее сложный класс сервиса на канальном уровне - сервис с соединением и уведомлением. Этот класс сервиса предполагает, что до начала передачи между машинами устанавливают соединение и данные передают по этому соединению. Каждый передаваемый кадр нумеруется, и канальный уровень гарантирует, что он будет обязательно получен, причем только один раз, а также что все кадры будут получены в надлежащей последовательности. При сервисе без соединения этого гарантировать нельзя, потому что потеря подтверждения получения кадра приведет к его пересылке, что, в свою очередь, приведет к появлению нескольких идентичных кадров. Использование сервиса с соединением особенно полезно в том случае, когда канал образует СПД. Как мы уже видели, в СПД может быть достаточно сложная организация канала, при которой может происходить коммутация потоков данных самыми разными способами. Здесь полезно вспомнить структуру кадра в X.25, рассмотренную в разделе 2.3. С точки зрения структуры, в X.25 есть три вида кадров: с 3-битовым полем номера кадра, 7- и 12-битовым. Таким образом, в X.25 предусмотрен сервис с соединением, причем, в зависимости от длины передаваемых данных, можно оптимизировать затраты на поддержку соединения. При сервисе с соединением и уведомлением передача данных разбивается на три этапа. На первом этапе устанавливают соединение: на обеих машинах инициируют переменные и счетчики, отслеживающие, кадры с какими номерами были приняты, а с какими нет. На втором этапе передают нужные кадры. На третьем - соединение разрывают.
Разбиение на кадры. Сервис, создаваемый канальным уровнем для сетевого уровня, опирается на сервис, создаваемый физическим уровнем. Отправленное количество битов не обязательно должно быть

равно принятому количеству битов, значение посланного бита также не обязательно должно быть равно принятому значению бита. Поэтому на канальном уровне нужны специальные действия по обнаружению и исправлению таких ошибок. Типовой подход к решению подобных проблем - разбиение потока битов на кадры, подсчет контрольной суммы для каждого кадра и передача этой суммы вместе с кадром данных. При приеме контрольная сумма вычисляется для каждого кадра заново и сравнивается с той, что хранится в кадре. Если они различаются, то это признак ошибки передачи. На канальном уровне должны быть приняты меры к исправлению ошибки, например, сбросить плохой кадр и послать сообщение об ошибке тому, кто прислал этот кадр. четыре основных метода:

1. счетчик символов . 2. вставка специальных стартовых и конечных символов

3.вставка стартовых и концевых битов; 4.нарушение кодировки на физическом уровне

1 - В начале каждого кадра указывают, сколько символов в кадре. При приеме кадра вновь подсчитывают число принятых символов. Если число полученных символов отлично от ожидаемого числа, то этот факт воспринимают как ошибку. Однако этот метод имеет существенный недостаток: счетчик символов может быть искажен при передаче. Тогда принимающая сторона не сможет обнаружить границы кадра. Даже обнаружив несовпадение контрольных сумм, принимающая сторона не сможет сообщить передающей, какой кадр надо переслать и сколько символов пропало. Этот метод сейчас используется редко. Второй метод построен на вставке специальных символов. Обычно для этого используют последовательность символов DLE STX для начала кадра и DLE ETX для конца кадра. DLE (Data Link Escape), STX (Start TeXt), ETX (End TeXt) – это специальные символы, имеющие специальную кодировку. При этом методе, если даже была потеряна граница текущего кадра, нужно просто найти ближайшую последовательность DLE STX или DLE ETX. Однако здесь есть одна опасность: при передаче чисел или программы в объектном коде такие последовательности могут уже содержаться в передаваемых данных. Для решения этой проблемы используют прием экранирования: каждая последовательность DLE или STX просто дублируется в передаваемых данных. Поэтому, если при приеме есть два последовательных DLE, то один удаляется. Основным недостатком только что рассмотренного метода является то, что он жестко связан с размером байта и конкретным методом кодировки символов - ASCII. По мере развития сетей эта связь становилась все более и более обременительной. Кроме этого, на стороне отправителя надо было просматривать кадр, чтобы обнаружить недопустимые последовательности. Был предложен иной прием, позволяющий использовать любое число битов на символ и любую кодировку. Его идея состоит в том, что каждый кадр начинается и заканчивается специальным флаг-байтом: 01111110. Этот метод прозрачен для сетевого уровня так же, как и метод вставки байтов. Таким образом, кадр легко может быть распознан по флаг-байту. Если граница очередного кадра по какой-то причине была потеряна, то все что надо делать – «ловить» ближайший флаг-байт. Последний метод - нарушение кодировки используется там, где применяется специальная кодировка битов на физическом уровне. Например, пусть для передачи одного бита используется два импульса. «1» кодируется как переход «высокое-низкое», «0» - как переход «низкое-высокое». Их и используют для границ кадра. На практике используют, как правило, комбинацию этих методов. Например, счетчик символов с одним из выше перечисленных.
Обнаружение ошибок. Для решения этой проблемы устанавливают обратную связь между отправителем и получателем в виде кадра подтверждения. Если кадр-подтверждение несет положительную информацию, то считается, что переданные кадры прошли нормально, если же в нем сообщение об ошибке, то переданные кадры надо передать заново. Однако возможны ситуации, когда из-за ошибок в канале кадр исчезнет целиком. В этом случае получатель не будет никак реагировать, а отправитель будет сколь угодно долго ждать подтверждения. Для решения этой проблемы на канальном уровне вводят таймеры. Когда передается очередной кадр, то одновременно устанавливается таймер на определенное время. Этого времени должно хватать на то, чтобы получатель получил кадр и отправил уведомление, а отправитель получил его. Если отправитель не получит уведомление раньше, чем истечет время, установленное на таймере, то он будет считать, что кадр потерян и повторит его еще раз. Однако если кадр-подтверждение был утерян, то вполне возможно, что один и тот же кадр получатель получит дважды. Как быть? Для решения этой проблемы каждому кадру присваивают порядковый номер. С помощью этого номера получатель может обнаружить дубли. Итак, таймеры и нумерация кадров - основные средства на

канальном уровне, обеспечивающие доставку каждого кадра до сетевого уровня в единственном экземпляре и в нужном порядке. **Управление потоком** .Другая важная проблема, которую надо решать на канальном уровне - управление потоком. Вполне может случиться, что отправитель будет посылать кадры столь часто, что получатель не будет успевать их обрабатывать. Для борьбы с такими ситуациями вводят специальный механизм управления потоком. Существует много схем управления потоком, но все они в основе своей используют следующий сценарий. Прежде чем отправитель начнет передачу, он спрашивает у получателя, сколько кадров тот может принять. Получатель сообщает ему определенное число. Отправитель, после того как передаст это число кадров, должен приостановить передачу и спросить у получателя еще раз, сколько кадров тот может принять, и т.д. Позднее на примерах мы познакомимся с конкретными механизмами управления потоком.

Простейшие протоколы канала данных

Рассмотрение протоколов уровня канала данных мы начнем с нескольких предположений. Будем предполагать, что физический уровень, уровень канала данных, сетевой уровень – реализованы в виде независимых процессов, взаимодействующих с помощью передачи сообщений. Также мы предположим, что есть две машины: A и B. У машины A есть бесконечно длинный набор данных, который надо передать машине B с помощью надежного сервиса, ориентированного на соединение. Передача всегда происходит от A к B, хотя позднее мы допустим одновременную передачу от B к A. Также будем предполагать, что если канальный уровень на машине A запрашивает данные для передачи от сетевого уровня, то они всегда есть и нет задержки на их подготовку. Канальный уровень рассматривает данные, которые он получает от сетевого, как неструктурированные, несмотря на то, что там есть хотя бы заголовок сетевого уровня. Все эти данные должны быть переданы равнозначному сетевому уровню. Когда канальный уровень получает пакет, он погружает его в кадр, добавляя заголовок и концевик. Этот кадр затем передается по физическому уровню.

Симплекс-протокол без ограничений

Данные передаются только в одном направлении. Получатель и отправитель всегда готовы к отправке и получению данных. Время обработки данных игнорируется. Предполагается, что буфер неограниченного размера. Данные в канале не теряются и не искажаются.

Симплексный старт-стопный протокол

Теперь снимем одно из ограничений предыдущего протокола - способность сетевого уровня обрабатывать поступающие данные сколь угодно быстро. Все остальные предположения остаются в силе: канал абсолютно надежный, трафик однонаправленный. Основная проблема - как предотвратить ситуацию, когда отправитель «заваливает» данными получателя. Если получателю требуется время Δt, чтобы исполнить from_physical_layer плюс to_network_layer, то отправитель должен передавать данные со средней скоростью один кадр в Δt. Решением такой проблемы может быть введение коротких специальных служебных сообщений. Получатель, получив один или несколько кадров, отправляет отправителю короткий специальный кадр, означающий, что отправитель может передавать следующий.

Симплексный протокол для канала с шумом

Основная проблема при передаче состоит в том, что кадр с подтверждением о получении может потеряться целиком. Как отличить кадр, переданный первый раз, от кадра, переданного повторно? Одно из очевидных решений - нумерация передаваемых кадров. Однако сколько места отводить под эту нумерацию? Поскольку проблема различения стоит для кадров m и m+1, то достаточно одного разряда. 0 - для только что посланного кадра и 1 - для следующего ожидаемого. Все кадры, не содержащие корректной нумерации, просто сбрасываются при приеме.

Протоколы скользящего окна

Для передачи в обоих направлениях можно потребовать на физическом уровне двух симплексных каналов. Один для передачи кадров, другой - для передачи подтверждений. Однако использование канала только для подтверждений - довольно дорогое удовольствие. Можно смешивать кадры с данными и кадры с подтверждениями на одном канале. Это, конечно. решение проблемы, но по-прежнему на подтверждения будет тратиться полезная пропускная способность канала. А что, если для подтверждения использовать полезные кадры с данными? Получатель не сразу отправляет подтверждение, а ожидает от сетевого уровня очередного пакета. Как только такой пакет возникает, то канальный уровень помещает в кадр с пакетом также уведомление о получении в специальное поле ack. Что делать, если тайм-аут у отправителя на получения подтверждения заканчивается, а с сетевого уровня получателя не поступает запроса на передачу пакета? Поэтому на канальном уровне должен быть фиксированный интервал времени, в течение которого канальный уровень ждет от сетевого попутного кадра. Если до истечения этого срока пакет с сетевого уровня не поступил, то канальный уровень отправляет подтверждение отдельным кадром. Рассмотренный здесь протокол является представителем класса протоколов скользящего окна. Кроме вышесказанного, протоколы этого класса делают следующее: у отправителя и получателя есть определенная константа n - число кадров, которое отправитель может послать, не ожидая подтверждения для каждого кадра. По мере получения подтверждений отправленные кадры будут сбрасываться из буфера отправителя, и буфер будет пополняться новыми кадрами. У получателя и отправителя есть набор последовательных чисел - номеров кадров, которые отправитель может отправить, не ожидая подтверждения каждого. Эти кадры образуют окно отправки. Аналогично, у получателя есть буфер для получения и временного хранения получаемых кадров - окно получения. Хотя в этих условиях у отправителя есть определенная свобода в порядке отправления кадров, мы по-прежнему будем считать, что кадры отправляют в соответствии с порядковыми номерами. У окон отправки и получения есть верхняя и нижняя границы. Порядковые номера кадров в окне отправки - кадры отправленные, но не подтвержденные. Как только от сетевого уровня поступил еще один пакет, ему присваивают первый свободный наибольший номер, и верхняя граница окна отправителя поднимается. Как только приходит подтверждение, нижняя граница окна поднимается. Таким образом, в окне все время находятся неподтвержденные кадры.

Протокол скользящего окна в 1 бит

Прежде чем переходить к общему случаю, рассмотрим протокол скользящего окна с максимальным размером окна в 1 бит. Такой протокол использует старт-стопный режим и, послав кадр, не шлет другой, пока не придет подтверждение на первый. На рисунке 3-13 показан текст протокола для этого простейшего случая. Как и все, он начинается с определения переменных. Next_frame_to_send указывает, какой кадр посылается. Переменная frame_expected определяет, какой кадр получатель ожидает. Есть только два значения - 0 или 1. Есть два случая: первый - простой и наиболее удобный, когда только один из канальных уровней первым начинает передачу. В этом случае вне тела основного цикла одной из программ канального уровня есть обращения к процедурам to_phisical_layer и start_timer. Случай, когда оба уровня одновременно могут начинать передачу, описывается позже, поскольку он требует более детального рассмотрения. Машина, инициирующая обмен, берет пакет от сетевого уровня, формирует кадр и посылает его. Когда он (или любой другой кадр) поступает, канальный уровень-получатель проверяет: не является ли этот кадр дубликатом. Если поступивший кадр тот, что ожидался, то он передается на сетевой уровень и окно получателя сдвигается вверх.

Поле уведомления содержит номер последнего кадра, полученного без ошибок. Если этот номер согласуется с номером кадра, который уровень-отправитель старается послать, то он считает, что кадр, хранящийся в буфере, послан, и сбрасывает его оттуда, забирая новый с сетевого уровня. Если номера не согласуются, то отправитель старается послать тот же кадр еще раз. В любом случае, после получения кадра отправляется новый кадр.

Протокол с возвратом на n кадров и протокол с выборочным повтором

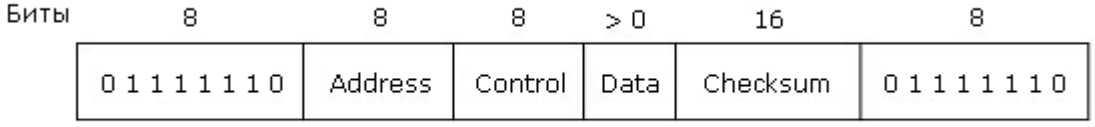
До сих пор мы предполагали, что время доставки кадра и время доставки подтверждения пренебрежимо малы. В некоторых случаях это предположение очевидно не работает. Оно может приводить к серьезным бесполезным тратам пропускной способности канала. потеря пропускной способности канала. Эта проблема есть следствие правила, по которому отправитель ждет подтверждения прежде, чем пошлет следующий кадр. Это требование можно ослабить - разрешить отправителю отправлять до w кадров, не дожидаясь их подтверждения. Надлежащим выбором значения w отправитель может заполнить все время, необходимое на отправку кадра и получение его подтверждения. Эта техника известна как конвейер. Ее применение в случае ненадежного канала наталкивается на ряд проблем. Первая - что делать, если в середине потока пропадет или попадется поврежденный кадр? Получатель уже получит большое количество кадров к тому моменту, когда отправитель обнаружит, что что-то произошло. Когда получатель получил поврежденный кадр, он его должен сбросить, что делать с последующими кадрами? Помните, что канальный уровень обязан передавать пакеты на сетевой уровень в том порядке, в каком их отправлял отправитель. Есть два приема для решения этих вопросов: откат и выборочный повтор. При откате все кадры, поступившие после поврежденного кадра, сбрасываются и не подтверждаются. Отправитель по тайм-ауту повторно отправляет все кадры, начиная с первого неподтвержденного кадра. Этот подход показан на рисунке 3-15 (а), где размер окна у получателя - 1. При выборочном повторе у получателя длина окна такая же, как и у отправителя. Отправитель отмечает неподтвержденный кадр и посылает его еще раз. Получатель не передает на сетевой уровень последовательность пакетов, если в ней есть разрывы.

Примеры протоколов канала данных.

Протокол HDLC (High Level Data Link Control)

До сих пор мы рассматривали решение основных проблем, с которыми приходится иметь дело на канальном уровне. Теперь мы познакомимся с группой давно известных, но по-прежнему широко используемых на практике протоколов. Все они имеют одного предшественника - SDLC (Synchronous Data Link Control) - протокола управления синхронным каналом, предложенного фирмой IBM в рамках архитектуры SNA. ISO модифицировало этот протокол и выпустило под название HDLC - High level Data Link Control. МКТТ модифицировало HDLC для X.25 и выпустило под именем LAP - Link Access Procedure. Позднее он был модифицирован в LAPB. Все эти протоколы построены на одних и тех же принципах. Они используют технику вставки специальных последовательностей битов и являются бит–ориентированными протоколами. Различия между ними незначительные.

Рисунок 3-16. Типовая структура кадра протокола HDLC



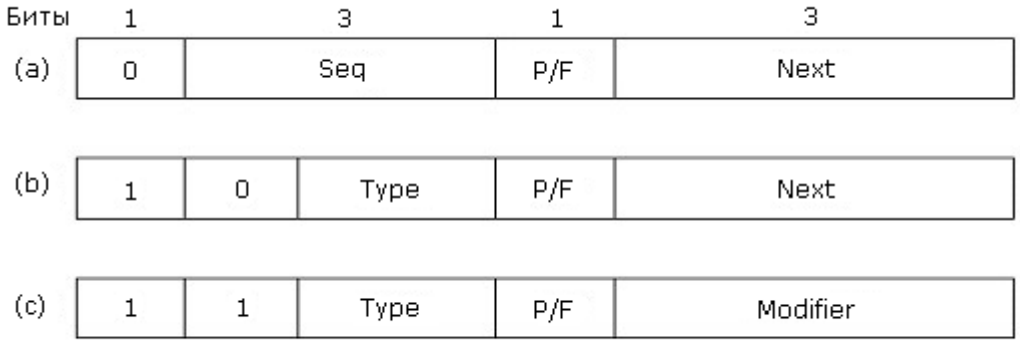
На рисунке 3-16 показана типовая структура кадра протокола HDLC.

- Поле Address используют для адресации терминала, если их несколько на линии. Для линий точка-точка это поле используется для того, чтобы отличать команду от ответа.
- Поле Control используется для последовательных номеров кадров, подтверждений и других нужд.

- Поле Data может быть сколь угодно большим и используется для передачи данных. Надо только иметь в виду, что чем длиннее это поле, тем больше вероятность повреждения кадра на линии.
- Поле Checksum - это поле используется для передачи CRC-кода.

Флаговые последовательности 01111110 используются для разделения кадров и постоянно передаются по незанятой линии в ожидании кадра. Существует три вида кадров: Information, Supervisory, Unnumbered. Организация поля Control для этих трех видов кадров показана на рисунке 3-17. Как видно из размера поля Seq, в окне отправителя может находиться до 7 неподтвержденных кадров. Поле Next используется для отправки подтверждения вместе с передаваемым кадром. Подтверждение может быть в форме номера последнего правильно переданного кадра, а может быть в форме первого, еще не переданного кадра. Какой вариант будет использован - зависит от параметров протокола.

Рисунок 3-17. Поле Control для кадров: Information (a), Supervisory (b), Unnumbered (c)



Разряд P/F используют при работе с группой терминалов. Когда компьютер приглашает терминал к передаче, он устанавливает этот разряд в P (все кадры, посылаемые терминалами, имеют здесь P). Если это последний кадр, посылаемый терминалом, то значение этого разряда устанавливается в F.

Кадры Supervisory бывают четырех типов.

- Тип 0 - уведомление в ожидании следующего кадра (RECEIVE READY). Используется, когда нет встречного трафика, чтобы передать уведомление в кадре с данными.
- Тип 1 - негативное уведомление (REJECT) - указывает на ошибку при передаче. Поле Next указывает номер кадра, начиная с которого надо перепослать кадры.
- Тип 2 - RECEIVE NOT READY. Подтверждает все кадры, кроме указанного в Next. Используется, чтобы сообщить источнику кадров о необходимости приостановить передачу в силу каких-то проблем у получателя. После устранения этих проблем получатель шлет RECEIVE REDAY, REJECT или другой надлежащий управляющий кадр.
- Тип 3 - SELECTIVE REJECT - указывает на необходимость перепослать только кадр, указанный в Next. LAPB и SDLC не используют кадры этого типа.

Третий класс кадров - Unnumbered. Кадры этого класса иногда используются для целей управления, но чаще для передачи данных при ненадежной передаче без соединения. Все протоколы имеют команду DISConnect - для сообщения о разрыве соединения. Команды SNRM и SABM используются для установки счетчиков кадров в ноль, сброса соединения в начальное состояние, установки соподчиненности на линии. Команда FRMR указывает на повреждение управляющего кадра (например, когда контрольная сумма верна, а значения полей противоречивы).

Frame Relay.Проблемы стандартизации

Ретрансляция кадров (Frame Relay, FR) - это метод доставки сообщений в сетях передачи данных (СПД) с коммутацией пакетов. Первоначально разработка стандарта FR ориентировалась на цифровые сети интегрированного обслуживания (ISDN - Integrated Services Digital Networks), однако позже стало ясно, что FR применим и в других СПД (здесь под данными понимается любое сообщение, представленное в цифровой форме). К числу достоинств рассматриваемого метода прежде всего необходимо отнести малое время задержки, простой формат кадров, содержащих минимум управляющей информации, и независимость от протоколов верхних уровней модели OSI. Любой международный стандарт имеет (и всегда будет иметь) множество прикладных реализаций, что зачастую приводит к несовместимости аппаратно-программных средств разных производителей. Международные организации неоднократно пытались решить данную проблему. Результатом одной из таких попыток (предпринятой FRF) стал проект стандарта, включающего в себя спецификации ANSI, которые обязательны для выполнения членами FRF. В январе 1992 г. этот проект был доработан Техническим комитетом FRF и утвержден собранием членов FRF.

Логическая характеристика протокола FR

FR является бит-ориентированным синхронным протоколом и использует кадр в качестве основного информационного элемента - в этом смысле он очень похож на протокол HDLC (High Level Data Link Control), рассмотренного нами в предыдущем разделе. Однако FR обеспечивает не все функции протокола HDLC. Многие элементы кадра HDLC исключены из основного формата кадра FR (в последнем адресное поле и поле управления HDLC совмещены в едином адресное поле), что привело к сокращению набора функций в этом протоколе.

Рисунок 3-18. Структура и формат кадра Frame Relay

Структура кадра FR (рисунок 3-18) включает в себя следующие элементы:

1. Флаг. Все кадры начинаются и заканчиваются комбинацией "флаг": "01111110".
2. Заголовок:

- Адрес в пределах кадра FR (стандарт FRF), состоит из шести бит первого байта и четырех бит второго байта заголовка кадра (стандарты ANSI и ITU-T допускают размер заголовка до 4 байтов). Эти 10 бит представляют собой идентификатор канала передачи данных (Data Link Connection Identifier, DLCI) и определяют абонентский адрес в сети FR.
- Бит «опрос/финал» (Command/ Response - CR) зарезервирован для возможного применения в различных протоколах более высоких уровней управления OSI. Этот бит не используется протоколом FR и «прозрачно» пропускается аппаратно-программными средствами сети FR.
- Бит расширения адреса (Extended Address - EA). DLCI содержится в 10 битах, входящих в два байта заголовка. Однако возможно расширение заголовка на целое число дополнительных байтов с целью указания адреса, состоящего более чем из 10 бит. Бит EA устанавливают в конце каждого байта заголовка; если он имеет значение «1», то это означает, что данный байт в заголовке последний. Стандарт FRF рекомендует использовать заголовки, состоящие

из двух байтов. В этом случае значение бита EA первого байта будет соответствовать «0», а второго - «1».

- Бит уведомления (сигнализации) приемника о явной перегрузке (Forward Explicit Congestion Notification - FECN) устанавливается в «1», если надо информировать получателя о том, что произошла перегрузка в направлении передачи данного кадра (рисунок 3-19).
- Бит уведомления (сигнализации) отправителя о явной перегрузке (Backward Explicit Congestion Notification - BECN). Этот бит устанавливают в «1» для уведомления отправителя сообщения о том, что произошла перегрузка в направлении, обратном направлению передачи содержащего этот бит кадра. Бит BECN может не использоваться терминалами абонентов (см. рисунок 3-19), т.е. в этом направлении возник слишком большой поток кадров.
- Бит разрешения сброса (Discard Eligibility - DE) устанавливают в «1» в случае явной перегрузки. Он указывает на то, что данный кадр может быть уничтожен в первую очередь, т.е. пользователю предоставлено право выбирать, какими кадрами он может «пожертвовать». Однако при перегрузках узлы коммутации сети FR уничтожают не только кадры с битом DE.

3. Информационное поле содержит данные пользователя и состоит из целого числа байтов. Его максимальный размер определен стандартом FRF и составляет 1600 байтов (минимальный размер - 1 байт), но возможны и другие максимальные размеры (вплоть до 4096 байтов). Содержание информационного поля пользователя передается неизменным.
4. Проверочная последовательность кадра (Frame Check Sequence - FCS) используется для обнаружения возможных ошибок при его передаче и состоит из двух байтов. Данная последовательность формируется аналогично циклическому коду HDLC.

Все указанные поля должны присутствовать в каждом кадре FR, который передается между двумя оконечными пользовательскими системами. Одним из основных отличий протокола FR от HDLC является то, что он не предусматривает передачу управляющих сообщений (нет командных или супервизорных кадров, как в HDLC). Для передачи служебной информации используется специально выделенный канал сигнализации. Другое важное отличие - отсутствие нумерации последовательно передаваемых (принимаемых) кадров. Дело в том, что протокол FR не имеет никаких механизмов для подтверждения правильно принятых кадров.

Процедурная характеристика протокола FR

Протокол FR является весьма простым по сравнению с HDLC и включает в себя небольшой свод правил и процедур организации информационного обмена. Основная процедура состоит в том, что если кадр получен без искажений, он должен быть направлен далее по соответствующему маршруту. При возникновении проблем, связанных с перегрузкой сети FR, ее узлы могут сбрасывать любой кадр. Узлам сети FR разрешено уничтожать искаженные кадры, не уведомляя об этом пользователя. Искаженным считается кадр, которому присущ какой-либо из следующих признаков:

- Нет корректного ограничения флагами.
- Имеется менее пяти байтов между флагами.
- Нет целого числа байтов после удаления бит обеспечения прозрачности.
- Присутствует ошибка контрольной суммы.
- Искажено поле адреса (для случая, когда проверка не выявила ошибки в FCS).
- Содержится несуществующий DLCI.
- Превышен допустимый максимальный размер (в некоторых вариантах реализации стандартов FR возможна принудительная обработка кадров, превышающих допустимый максимальный размер).

Для FR характерно:

- заполнение канала связи комбинацией «флаг» при отсутствии данных для передачи

- резервирование одного DLCI для интерфейса локального управления и сигнализации
- содержание поля данных пользователя в любом кадре не должно подвергаться какой-либо обработке со стороны аппаратуры канала данных (могут обрабатываться лишь данные в локальном канале управления)

Уровень канала данных в Интернете

Здесь мы рассмотрим протоколы, которые используются для каналов «точка-точка» в Интернете. На уровне канала данных соединения «точка-точка» возникают между маршрутизаторами либо коммутирующими элементами в СПД. Другой часто встречающийся случай для таких соединений - соединение из дома через модем с интернет-провайдером. Эта ситуация показана на рисунке 3-20.

Serial Line IP

SLIP - наиболее старый из этих двух протоколов. Он был создан в 1984 году для соединения рабочих станций SUN через модем. Этот протокол был описан в RFC 1055. Его работа очень проста: он вставляет специальные флаг-байты в начало и конец IP-пакета. Последние версии этого протокола осуществляют также сжатие заголовков TCP и IP у последовательных пакетов, так как они несут очень много одинаковой информации. Одна из последних версий этого протокола описана в RFC 1144. SLIP имеет ряд серьезных недостатков - он не занимается контролем и исправлением ошибок, оставляя это протоколам верхних уровней. Во-вторых, он работает только с IP-пакетами. В современных условиях, когда Интернет объединяет самые разнообразные сети, это серьезный недостаток. В-третьих, IP-адреса взаимодействующих сторон должны быть известны заранее. В условиях нехватки IP-адресов это недостаток, так как было бы удобнее задавать IP-адрес динамически, лишь на период действия соединения. В-четвертых, этот протокол не обеспечивает какой-либо проверки аутентичности взаимодействующих сторон. Так что вы не можете быть уверены, с кем вы общаетесь. В-пятых, для этого протокола нет стандарта, и существует множество его версий, не все из которых совместимы.

PPP - протокол «точка-точка»

Чтобы исправить указанные выше недостатки, комитет IETF (Internet Engineering Task Force) создал группу, которой было поручено разработать новый протокол. В результате ее усилий появился протокол PPP (Point-to-Point Protocol), описанный в RFC 1661, 1662 и 1663. Протокол PPP обеспечивает обнаружение ошибок, поддерживает разные протоколы, позволяет динамически выделять IP-адрес только на период соединения, выполняет аутентификацию абонентов и имеет ряд других преимуществ перед SLIP. Протокол PPP обеспечивает три основных функции:

1. Распознавание кадров. Однозначно определяется конец кадра и начало нового. Здесь же происходит обнаружение ошибок.
2. Управление линией, т.е. активизация линии, ее проверка, определение основных параметров передачи в диалоге, корректное завершение передачи со сбросом параметров. Этот протокол называет LCP (Link Control Protocol).
3. Определение основных параметров соединения между сетевыми уровнями, чтобы обеспечить независимость от реализации сетевого уровня. Выбранный метод предполагает наличие разных NCP (Network Control Protocol) на каждом поддерживаемом сетевом уровне.

Чтобы лучше понять, как это все работает вместе, рассмотрим типичный сценарий, когда пользователь из дома по телефонной линии хочет подключить свой PC к Интернету. PC звонит на маршрутизатор сервис-провайдера. После того как маршрутизатор принял звонок и установил физическое соединение, PC посылает несколько LCP-пакетов в PPP-кадрах. Маршрутизатор отвечает LCP-пакетами в PPP-кадрах. В результате такого обмена определяются параметры соединения. После этого следует обмен NCP-пакетами для настройки сетевого уровня. В частности, здесь происходит временное присваивание PC IP-адреса, который действует только на период соединения. Это происходит, если обе стороны хотят использовать TCP/IP-стек. Теперь, когда PC стала полноправной машиной в Интернете, она может обмениваться IP-пакетами с

другими машинами. Когда пользователь закончит работу, NCP разрывает соединение с сетевым уровнем и освобождает ранее занятый IP-адрес. После этого LCP-протокол разрывает соединение на канальном уровне. А затем компьютер говорит модему: «Положи трубку». PPP-кадры имеют формат, очень близкий к HDLC-кадрам. Основное различие состоит в том, что PPP - байт-ориентированный, а HDLC - бит-ориентированный. Для HDLC возможен кадр размером в 30,25 байт, а для PPP - нет. Так как значения полей «Address» и «Control» - константы, то LCP-протокол опускает их, экономя два байта на передаче. В поле «Protocol» указывается, какой тип пакетов будет в поле «Payload». Там допускаются пакеты протоколов LCP, NCP, IP, IPX, Apple Talk и других. Поле «Payload» имеет переменную длину, по умолчанию она равна 1600 байт.

Уровень канала данных в ATM.

Теперь вернемся к уровням ATM-протокола. Физический уровень в ATM покрывает физический уровень и уровень канала данных в OSI. Проследим, что с ним происходит, когда ячейки достигают TC-подуровня и далее. **Передача ячеек.**

Первый шаг - вычисление контрольной суммы заголовка. Заголовок состоит из 5 байт - 4 байта идентифицируют виртуальное соединение и несут контрольную информацию, за ними следует 1 байт с контрольной суммой. Контрольная сумма защищает только первые четыре байта и не затрагивает данные в ячейке. Контрольная сумма вычисляется как остаток от деления содержимого 4 байтов на полином x8+x2+x+1. К этому остатку добавляется константа 01010101 для повышения надежности, в случае если заголовок содержит много нулей. Решение защищать контрольной суммой только управляющую информацию было принято с целью сократить затраты на обработку на нижних уровнях. Защита собственно данных возложена на верхние уровни, если это необходимо. Как мы уже отмечали, многие приложения реального времени - передача видео-, аудиоданных - более критичны к времени передачи, чем к степени искажения отдельных ячеек. Поскольку контрольная сумма покрывает только заголовок, то этот байт так и называется - HEC (Header Error Control - контроль ошибки в заголовке). Другим важным фактором, повлиявшим на выбор этой схемы контрольной суммы, было то, что основной средой для ATM является оптоволокно. Исследования, выполненные компанией AT&T, показали, что оптоволокно - высоконадежная среда и единичные ошибки происходят в ней с вероятностью менее 1%. Схема HEC прекрасно справляется как с однобитными ошибками, так и множественными. Для надежной передачи ячеек была предложена схема, когда две последовательные ячейки объединяются через EXCLUSIVE OR, после чего получается новая ячейка, которая добавляется в последовательность после первых двух. В результате если хоть одна ячейка была принята с ошибкой или потеряна, то она легко может быть восстановлена. После того как HEC вычислен и добавлен в заголовок, ячейка готова к передаче. Среда передачи может быть двух категорий - синхронной и асинхронной. В асинхронной среде ячейка посылается сразу, как только она готова к передаче. В синхронной среде ячейка передается в соответствии с временными соглашениями. Если нет ячейки для передачи, то TC-подуровень должен сгенерировать специальную ячейку ожидания. Другой вид служебных ячеек - OAM (Operation And Maintenance). Эти ячейки используются ATM-переключателями для проверки работоспособности системы. Ячейки ожидания обрабатываются соответствующим TC-подуровнем, а OAM-ячейки передаются на ATM-уровень. Другой важной функцией TC подуровня является генерирование ячеек в формате физической среды передачи. Это значит, что TC-подуровень генерирует обычную ATC-ячейку и упаковывает ее в кадр надлежащей среды передачи. **Прием ячеек.** Итак, на выходе TC-подуровень формирует HEC-заголовок, преобразует ячейку в кадр, формирует ATM-ячейки и передает поток битов на физический уровень. На противоположном конце TC-подуровень производит те же самые действия, но в обратном порядке: разбивает поток бит на кадры, выделяет ячейки, проверяет HEC-заголовки и передает ячейки на ATM-уровень. На TC-подуровне есть сдвиговый регистр на 40 бит. Если в этих 40 бит правые 8 представляют собой HEC, то последующие 32 левых бита - заголовок ячейки. Если условие не выполнено, то все сдвигается на один бит и проверка повторяется. Этот процесс продолжается до тех пор, пока не будет обнаружен HEC. Схема распознавания в том виде, как она описана не надежна. Вероятность того, что случайный байт будет выглядеть как HEC, равна 1/256. Чтобы исправить эту схему, используют автомат, схема состояний которого изображена на рисунке 3-24. Есть три состояния: HUNT, PRESYNCH, SYNCH. В состоянии HUNT ищется HEC. Как только найден похожий байт,

автомат переходит в состояние PRESYNCH и отчитывает следующие 53 байта. Если предположение о том, что найденный HEC - начало ячейки, то сдвиг на 53 байта приведет к следующему HEC. Происходит проверка последовательно δ ячеек, после этого происходит переход в состояние SYNCH. Если в состоянии SYNCH α последовательных ячеек оказались плохими, происходит переход в состояние HUNT.

Протоколы множественного доступа к каналу (динамическое vs статическое выделение канала). Модель системы ALOHA. Сравнение производительности систем: чистая ALOHA, слотированная ALOHA. Протоколы множественного доступа с обнаружением несущей (настойчивые и не настойчивые CSMA, CSMA с обнаружением коллизий).

Статическое предоставление канала

Как мы уже рассматривали ранее, есть два основных подхода к мультиплексированию нескольких конкурирующих пользователей на одном канале - частотное разделение (FDM) и временное разделение (TDM). статическое разделение канала на подканалы является неэффективным решением при предположении о постоянстве числа пользователей в среднем. Те же самые рассуждения можно применить и к временному разделению. Если каждому пользователю выделить свой слот и тот его не использует, то это пустая трата пропускной способности канала. Таким образом, ни один из известных статических методов не позволяет эффективно распределять нагрузку. Поэтому мы сосредоточимся на динамических методах распределения доступа к каналу.

Динамическое предоставление канала

Прежде чем перейти к описанию многочисленных динамических способов предоставления доступа к каналу, сформулируем основные пять предположений, которые и будут составлять основу моделей, которые мы будем использовать при оценке этих способов:

1. Станции. Модель состоит из N независимых станций (компьютеров, телефонов, факс-машин и т.п.). На каждой работает пользователь или программа, которые генерируют кадры для передачи. Вероятность появления кадра в интервале длины Δt равна $\lambda\Delta t$, где λ - константа и $0<\lambda<1$. Предполагается, что если кадр сгенерирован, то станция блокируется, и новый кадр не появится, пока не будет передан первый. Это предположение означает, что станции независимы, и на каждой работает только одна программа или пользователь, которые генерируют нагрузку с постоянной скоростью.
2. Единственность канала. Канал один и он доступен всем станциям. Все станции равноправны. Они получают кадры и передают кадры только через этот единственный канал. Аппаратные средства всех станций для доступа к каналу одинаковы, но программно можно устанавливать станциям приоритеты.
3. Коллизии. Если две станции передают кадры в одно и то же время, то сигналы накладываются и разрушаются. Этот случай будем называть коллизией. Любая станция может обнаружить коллизию. Кадры, разрушенные при коллизии, должны быть посланы повторно позднее. Кроме коллизий, других ошибок передачи нет.
4. Время. Мы будем предполагать две модели времени – непрерывное время и дискретное время.

   A. Непрерывное время. Передача кадра может начаться в любой момент. Нет единых часов в системе, которые разбивают время на слоты.
   B. Дискретное время. В слоте может оказаться 0 кадров, если это слот ожидания, 1 кадр - если в этом слоте передача кадра прошла успешно, несколько кадров, если в этом слоте произошла коллизия.

5. Доступ к каналу: возможны два способа доступа станции к каналу.

   A. С обнаружением несущей. Станция, прежде чем использовать канал, всегда определяет, занят он или нет. Если он занят, то станция не начинает передачу.
   B. Отсутствие несущей. Станция ничего не знает о состоянии канала, пока не начнет использовать его. Она сразу начинает передачу и лишь позднее обнаруживает коллизию.

Есть и другие модели, которые предусматривают многопользовательские станции, но эти модели намного сложнее. Единый канал передачи - это краеугольное предположение.

Протоколы множественного доступа.

ALOHA

Система состояла из наземных радиостанций, связывающих острова между собой. Идея была позволить в вещательной среде любому количеству пользователей неконтролируемо использовать один и тот же канал. Мы здесь рассмотрим два варианта системы: чистая ALOHA и слотированная ALOHA, т.е. разбитая на слоты. Основное различие - в первом случае никакой синхронизации пользователей не требуется, во втором она нужна.

Чистая ALOHA

Идея чистой ALOHA проста - любой пользователь, желающий передать сообщение, сразу пытается это сделать. Благодаря тому, что в вещательной среде он всегда имеет обратную связь, т.е. может определить, пытался ли кто-то еще передавать на его частоте, то он может установить возникновение конфликта при передаче. Такая обратная связь в среде LAN происходит практически мгновенно, в системах спутниковой связи задержка составляет около 270 мсек. Обнаружив конфликт, пользователь ожидает некоторый случайный отрезок времени, после чего повторяет попытку. Интервал времени на ожидание должен быть случайным, иначе конкуренты будут повторять попытки в одно и то же время, что приведет к их блокировке. Системы подобного типа, где пользователи конкурируют за получение доступа к общему каналу, называются системами с состязаниями. Не важно, когда произошел конфликт: когда первый бит одного кадра «наехал» на последний бит другого кадра или как-то иначе, оба кадра считаются испорченными и должны быть переданы повторно. Контрольная сумма, защищающая данные в кадре, не позволяет различать разные случаи наложения кадров. Какова эффективность системы ALOHA, измеренная в количестве кадров, которые избежали коллизий? Для ответа на этот вопрос рассмотрим следующую модель. Есть неограниченное число пользователей, работающих на компьютерах. Все что они могут делать, - это либо набирать текст, либо ждать, пока набранный текст будет передан. Когда пользователь заканчивает набирать очередную строку, он останавливается и ждет ответа от системы. Система пытается передать эту строку. Когда она сделает это, пользователь видит отклик и может продолжать работу. Назовем временем кадра время, необходимое на передачу кадра стандартной фиксированной длины. Предполагаем, что число пользователей неограниченно, и они порождают кадры по закону Пуассона со средним S кадров за время кадра. Поскольку при S>1 очередь на передачу будет только расти и все кадры будут страдать от коллизий, то мы будем предполагать 0<S<1. Также будем предполагать, что вероятность за время кадра сделать k попыток передачи распределена по закону Пуассона со средним G. Понятно, что должно быть GiS, иначе очередь будет расти бесконечно. При слабой загрузке (S»0) будет мало передач, а следовательно и коллизий, поэтому допустимо G»S. При высокой загрузке должно быть G>S. При любой нагрузке пропускная способность это - число кадров, которые надо передать, умноженное на вероятность успешной передачи. Если обозначить P0 вероятность отсутствия коллизий при передаче кадра, то S=GP0.

Рассмотрим внимательно, сколько времени нужно отправителю, чтобы обнаружить коллизию. Пусть он начал передачу в момент времени t_0 и пусть требуется время t, чтобы кадр достиг самой отдаленной станции. Тогда, если в тот момент, когда кадр почти достиг этой отдаленной станции, она начнет передачу (ведь в системе ALOHA станция сначала передает, а потом слушает), то отправитель узнает об этом только через t_0+2t . Вероятность появления k кадров при передаче кадра при распределении Пуассона равна

$$P[k] = \frac{G^k e^{-G}}{k!}$$ поэтому вероятность, что появится 0 кадров, равна e^{-G}.

За двойное время кадра среднее число кадров будет равна 2G, отсюда

$$P0=e^{-2G}$$

а так как $S=GP_0$, то

$$S=Ge^{-2G}$$

Зависимость между нагрузкой и пропускной способностью показана на рисунке 4-2 (нижний график). Максимальная пропускная способность достигается при G=0,5 при S=1/2e, что составляет примерно 18% от номинальной пропускной способности. Это означает, что если мы будем генерировать кадры с большей скоростью, чем 18% от скорости канала, то очереди переполнятся и система «захлебнется». Результат не очень вдохновляющий, но это плата за удобство: каждый передает, когда захочет.

Слотированная ALOHA

Все время работы канала разделяют на слоты. Размер слота определяют так, чтобы он был равен максимальному времени кадра. Ясно, что такая организация работы канала требует синхронизации. Кто-то, например, одна из станций испускает сигнал начала очередного слота. Поскольку передачу теперь можно начинать не в любой момент, а только по специальному сигналу, то время на обнаружение коллизии сокращается вдвое. Отсюда

$$S=Ge^{-G}$$

Как видно из рисунка 4-3, максимум пропускной способности слотированной ALOHA наступает при G=1, где S=1/e, т.е. около 0,37, что вдвое больше, чем у чистой ALOHA.

Рассмотрим, как G влияет на пропускную способность. Для этого подсчитаем вероятность успешной передачи кадра за k попыток. Так как e^{-G} - вероятность отсутствия коллизии при передаче, то вероятность, что кадр будет передан ровно за k попыток, равна

$$P_k=e^{-G}(1-e^{-G})^{k-1}$$

Среднее ожидаемое число повторных передач будет равно

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1-e^{-G})^{k-1} = e^{G}$$

Эта экспоненциальная зависимость показывает, что с ростом G резко возрастает число повторных попыток. Поэтому незначительное увеличение загрузки канала ведет к резкому падению пропускной способности.

Протоколы множественного доступа с обнаружением несущей

Максимальная пропускная способность, какую мы можем получить для системы ALOHA, достигается при S=1/e. Это не удивительно, так как в этих системах станция не обращает внимания на то, что делают другие. Поэтому вероятность коллизии чрезвычайно высока. В локальных сетях есть возможность определить, что делают другие станции, и только после этого решать, что делать самому. Протоколы, которые реализуют именно эту идею – сначала определить, занят канал или нет и только после этого действовать - называются протоколами с обнаружением несущей CSMA (Carrier Sensitive Multiple Access).

Настойчивые и ненастойчивые CSMA-протоколы

Согласно протоколу, который мы сейчас рассмотрим, станция, прежде чем что-либо передавать, определяет состояние канала. Если канал занят, то она ждет. Как только канал освободился, она пытается начать передачу. Если при этом произошла коллизия, она ожидает случайный интервал времени и все начинает с начала. Этот протокол называется настойчивым CSMA-протоколом первого уровня или 1-настойчивым CSMA-протоколом, потому что станция, следуя этому протоколу, начинает передачу с вероятностью 1, как только обнаруживает, что канал свободен. Здесь важную роль играет задержка распространения сигнала в канале. Всегда есть шанс, что, как только одна станция начала передачу, другая также стала готовой

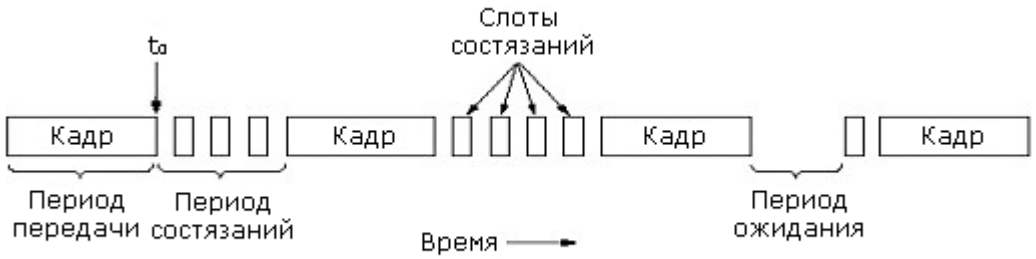
передавать. Если вторая станция проверит состояние канала прежде, чем до нее дойдет сигнал от первой о том, что она заняла канал, то вторая станция сочтет канал свободным и начнет передачу. В результате - коллизия. Чем больше время задержки, тем больше вероятность такого случая, тем хуже производительность канала. Однако даже если время задержки будет равно 0, коллизии все равно могут возникать. Например, если готовыми передавать оказались две станции, пока одна станция продолжает передавать. Они вежливо подождут, пока первая закончит передачу, а потом будут состязаться между собой. Тем не менее, этот протокол более эффективен, чем любая из систем ALOHA, так как станция учитывает состояние канала, прежде чем начать действовать. Другой вариант CSMA-протокола - ненастойчивый CSMA-протокол. Основное отличие его от предыдущего в том, что готовая к передаче станция опрашивает канал. Если он свободен, то она начинает передачу. Если он занят, то она не будет настойчиво его опрашивать в ожидании, когда он освободится, а будет делать это через случайные отрезки времени. Это несколько увеличивает задержку при передаче, но общая эффективность протокола возрастает. И, наконец, настойчивый CSMA-протокол уровня p. Он применяется к слотированным каналам. Когда станция готова к передаче, она опрашивает канал. Если он свободен, то она с вероятностью p передает свой кадр и с вероятностью q=1-p ждет следующего слота. Так она действует, пока не передаст кадр. Если произошла коллизия во время передачи, она ожидает случайный интервал времени и опрашивает канал опять. Если при опросе канала он оказался занят, станция ждет начала следующего слота, и весь алгоритм повторяется. На рисунке 4-3 показана пропускная способность этого протокола в зависимости от нагрузки.

CSMA-протокол с обнаружением коллизий

Настойчивые и ненастойчивые CSMA-протоколы – несомненное улучшение протокола ALOHA, т.к. они начинают передачу, только проверив состояние канала. Другое улучшение этого протокола, которое можно сделать, состоит в том, что станции должны уметь определять коллизии как можно раньше, а не по окончании отправки кадра. Это экономит время и пропускную способность канала. Такой класс протоколов известен, как CSMA/CD - Carrier Sensitive Multiple Access with Collision Detection, т.е. протокол множественного доступа с контролем несущей и обнаружением коллизий. Протоколы этого класса широко используется в локальных сетях.

На рисунке 4-4 показана модель, которая используется во многих протоколах этого класса. В момент t0 станция заканчивает передачу очередного кадра. Все станции, у которых есть кадр для передачи, начинают передачу. Естественно, происходят коллизии, которые быстро обнаруживаются сравнением отправленного сигнала с тем, который есть на линии. Обнаружив коллизию, станция сразу прекращает передачу на случайный интервал времени, после чего все начинается сначала. Таким образом, в работе протокола CSMA/CD можно выделить три стадии: состязания, передачи и ожидания, когда нет кадров для передачи.

Рисунок 4-4. Стадии работы протокола CSMA/CD



Надо также отметить, что MAC-подуровень обеспечивает надежную передачу, используя специальные приемы кодирования данных. Примеры таких кодировок мы рассматривали в гл. 2 (см. Манчестерские коды).

3. **Протоколы множественного доступа к каналу: Бесконфликтные протоколы (Bit-Map протокол, Адресный счетчик). Протоколы с**

Хотя в протоколе CSMA/CD коллизии могут возникать только в период состязаний, тем не менее, при больших t и коротких кадрах они съедают часть пропускной способности канала. Здесь мы рассмотрим, как можно этих коллизий избежать. Мы будем предполагать, что у нас есть N станций с адресами от 0 до N-1. Все адреса уникальны. Основным является вопрос: как определить, кто будет владеть каналом, когда закончится текущая передача?

Bit-Map-протокол

Выделяют специальный период состязаний, где количество слотов равно числу станций. Каждая станция, имеющая кадр для передачи, проставляет 1 в свой слот. Поскольку мы рассматриваем канал с множественным доступом (т.е. все видят, что проходит в канале), то в конце состязаний все станции знают, кто будет передавать кадры и в каком порядке. Передача происходит в том же порядке, в каком пронумерованы слоты. Раз станции знают, кто будет передавать и в каком порядке, то конфликтов не будет. Если станция опоздала с заявкой на передачу, то она должна ждать следующего периода состязаний, который начнется по окончании передач, заявленных на предыдущем периоде состязаний. Такие протоколы, когда заявки на передачу откладываются и могут быть сделаны лишь в определенные периоды времени, называются протоколами с резервированием.

Рисунок 4-5. Bit-Map-протокол

Теперь рассмотрим производительность этого метода. Будем для удобства измерять время в количестве слотов состязаний. Будем также предполагать, что передача одного кадра будет занимать ровно d таких слотов. Для станции с небольшим номером, например, 0 или 1, время ожидания на передачу в среднем будет равняться 1,5 N, потому что она, пропустив начало состязаний, будет ждать 0,5 N в первом периоде состязаний и N единиц времени во втором. Другая участь у станций со старшими номерами. Эти станции будут ожидать в среднем N/2 слотов до начала передачи. Таким образом, в среднем любая станция должна будет ждать N слотов до передачи. При небольшой нагрузке накладные расходы на передачу одного кадра будут N бит, а эффективность передачи одного кадра - d/(d+N), где N - накладные расходы на передачу кадра. При плотной загрузке, когда практически каждая станция каждый раз что-то посылает, накладные расходы будут 1 бит на кадр, т.е. d/(d+1). Средняя задержка кадра будет равна средней задержке кадра внутри очереди в станции плюс N(d+1)/2 слотов ожидания, когда кадр достигнет заголовка очереди. Отсюда видно, что с ростом N, хотя накладные расходы на передачу одного кадра и падают, задержка кадра в канале существенно возрастает, и эффективность падает. Следует также отметить, что если d и N - сопоставимые величины, то значительную часть пропускной способности канала мы будем тратить на состязания.

Адресный счетчик

Один из недостатков bit-map протокола - затраты в 1 бит на кадр. При коротких кадрах это накладно. Есть другая возможность, позволяющая повысить эффективность использования канала. Она основана на двоичном представлении адреса станции. В этом методе каждая станция, готовая к передаче, выставляет свой адрес бит за битом, начиная со старшего разряда. Эти разряды подвергаются логическому сложению. Если станция выставила на очередном шаге 0, а результат логического сложения - 1, то она должна ждать и в текущих состязаниях участия не принимает. Этот метод проиллюстрирован на рисунке 4-6. Эффективность использования канала в этом методе - d/(d+lnN). Если структура заголовка кадра была выбрана так, чтобы его

можно было использовать для выбора очередной станции для передачи, то lnN битов также будет использовано, тем самым эффективность использования канала достигнет 100%.

Этот метод имеет один существенный недостаток – он не справедливый: чем больше номер станции, тем скорее она захватит канал. В 1979 году Мок (Mok) и Уорд (Ward) предложили модификацию этого метода, когда у станций динамически изменяется приоритет, на основе которого определяется победитель. Победивший в текущих состязаниях получает наименьший приоритет, который будет увеличиваться от состязания к состязанию.

Протоколы с ограниченным числом конфликтов

Рассмотренные нами только что протоколы показывают, что при небольшой загрузке конфликты не опасны ввиду небольшой задержки на передачу. По мере роста нагрузки они снижают эффективность использования канала. Поэтому при высокой загруженности канала арбитраж желателен и протоколы без коллизий предпочтительнее. А вот при низкой загрузке они лишь вызывают дополнительные накладные расходы. Естественно попытаться создать протокол, объединяющий достоинства этих двух групп методов, т.е. использовать состязания при небольших нагрузках и бесконфликтные методы - при высоких. Такие протоколы существуют и называются протоколами с ограниченным числом конфликтов.

1. Симметричная конфигурация протоколов с состязаниями

До сих пор мы рассматривали протоколы с состязаниями только в так называемой симметричной конфигурации: все станции, пытающиеся передать кадр, получали канал с одной и той же для всех вероятностью p. Оказывается, общая производительность системы может быть улучшена, если разным станциям будет сопоставлена разная вероятность. Рассмотрим производительность в случае симметричного случая. Пусть у нас есть k станций, каждая из которых с вероятностью p готова передать кадр. Тогда вероятность, что какая-то станция успешно передаст свой кадр, равна $kp(1-p)^{k-1}$. Эта вероятность достигает максимума при p=1/k. Тогда вероятность передать сообщение какой-либо станцией равна

$$\left(\frac{k-1}{k}\right)^{k-1}$$

График этой функции показан на рисунке 4-7. При небольшом числе станций шансы передать кадр достаточно велики, но с ростом числа станций эти шансы резко падают. Единственным способом увеличить шансы на передачу является сократить конфликты. Для этого в протоколах с ограниченным числом конфликтов все станции разбивают на непересекающиеся группы. За слот с номером 0 состязаются только станции из группы 0. Если передавать нечего или была коллизия, то начинают состязания за слот 1 члены группы 1, и т.д. В результате в каждом слоте конкуренция падает и мы имеет случай левой части кривой из рисунка 4-7. Основную сложность в этом методе составляет распределение станций по группам.

2. Адаптивный древовидный протокол

Этот протокол устроен по принципу тестирования солдат американской армии на сифилис во второй мировой войне. У n солдат брали кровь на анализ. В первой пробе в общей пробирке смешивали часть крови каждого солдата. Если тест давал отрицательный результат, то все n считались здоровыми. Если тест давал положительную реакцию, то в пробирке смешивали только кровь первой половины солдат и опять тестировали. Если был положительный результат, то эту половину делили опять пополам и т.д., пока не обнаруживали носителя. Станции - листья. За слот 0 борются все станции. Если какая-то победила - хорошо. Если нет, то за слот 1 борются только станции поддерева с корнем в вершине 2. Если какая-то победила, то следующий слот резервируется для станций поддерева 3. Если был конфликт, то за следующий слот борются станции поддерева 4, и т.д. Когда число станций велико и все они готовы передавать, то вряд ли целесообразно начинать поиск с уровня 0 в дереве. Возникает вопрос: с какого уровня надо начинать эту процедуру при заданном числе станций? Пусть число станций, готовых к передаче, нормально распределено.

Обозначим это число через q. Тогда число станций, готовых к передаче и расположенных ниже уровня i, будет ровно $2^{-i}q$. Заметим, что их доля от общего числа станций, расположенных в дереве ниже уровня i, равна 2^{-i}. Естественно, надо подобрать такое соотношение между i и q, когда количество конкурирующих станций будет 1, т.е. $2^{-i}q=1$, или $\log_2 q=i$.

Протоколы с множественным доступом и разделением частот

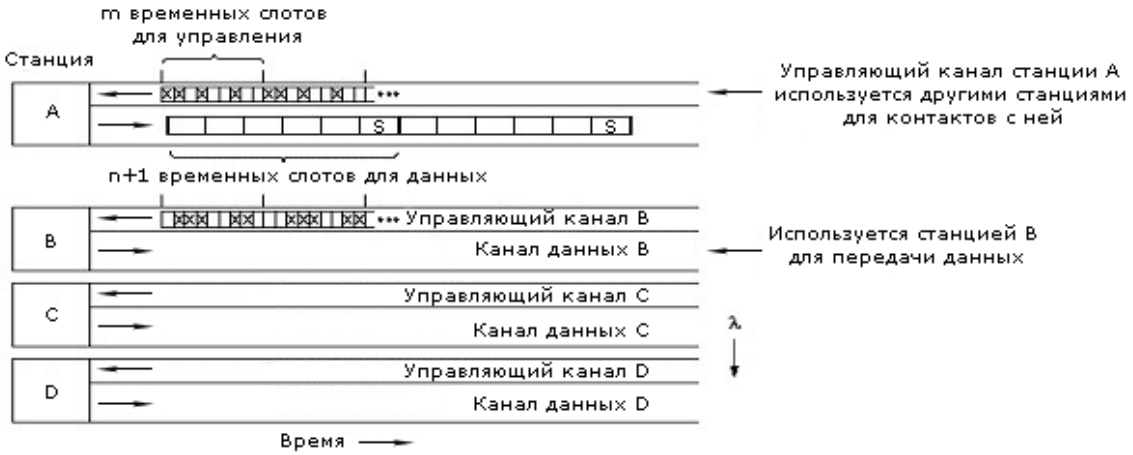
Иной подход к распределению доступа к каналу основан на разделении канала на подканалы, используя FDM-, TDM-метод или сразу оба этих метода. Здесь мы рассмотрим работу протоколов множественного доступа для оптоволоконных систем. Они построены на идее разделения частот. Вся полоса разделяется на каналы по два на станцию. Один, узкий, - управляющий канал, второй, широкий, - для передачи данных. Каждый канал разбит на слоты. Все слоты синхронизируются от единых часов. Отмечается только нулевой слот, чтобы можно было определить начало каждого цикла. Обозначим через m число слотов в управляющем канале и через n+1 - в канале данных. Из них n слотов - для данных, а последний - для сообщения о статусе канала. Протокол поддерживает три класса трафика:

1. Постоянный с соединением (видео)
2. Переменный с соединением (передача файлов)
3. Дейтаграммный (типа UDP)

У каждой станции есть два ресивера и два трансивера:

1. Ресивер для фиксированной длины волны для канала управления
2. Настраиваемый трансивер для передачи в каналы управления других станций
3. Трансивер для фиксированной длины волны для передачи данных
4. Настраиваемый ресивер для получения данных

Другими словами, каждая станция постоянно слушает свой управляющий канал, но должна настраиваться при приеме на волну передающей станции. Рассмотрим, как станция A устанавливает соединение класса 2 со станцией B для передачи файла. A настраивается на управляющий канал станции B, чтобы определить, какие слоты уже заняты, а какие свободны. Рисунок 4-9. Множественный доступ с разделением частот



A выбирает, например, 4-й слот и помещает туда свой CONNECTION REQUEST. Станция B видит этот запрос и закрепляет слот 4 за станцией A, о чем сообщает ей через статусный слот. Для станции A это означает, что установлено однонаправленное соединение от A к B. Если нужно двунаправленное соединение, то B должна повторить все, что сделала A. Если в момент попытки A захватить слот у B другая станция, например, C, также попытается это сделать, возникнет конфликт, о котором и A, и C узнают через статусный слот управляющего канала. После того как соединение установлено, A посылает B управляющее сообщение типа: «Жди. В слоте 3 канала данных есть кадр». Получив такое сообщение, B настраивается на волну канала A и считывает кадр. Таким образом, мы имеем бесконфликтный канал. Хотя может случиться, что если A и C

имеют соединение с B и оба скажут «смотри на слот 3, там кадр от меня», то от какого из двух получит сообщение B, сказать заранее нельзя. В случае дейтаграмм A шлет не запрос на соединение, а сообщение типа: «В слоте 3 для тебя есть кадр». Существует несколько вариантов этого WDMA-протокола.

Стандарты IEEE 802.x для локальных и муниципальных сетей: Стандарт IEEE 802.3 и Ethernet (кабели, способ физического кодирования, алгоритм вычисления задержки, MAC подуровень, производительность). Стандарт IEEE 802.2: управление логическим каналом.

Перейдем от изучения абстрактных протоколов к рассмотрению конкретных стандартов, используемых в реальных транспортных системах. Все стандарты для локальных сетей сконцентрированы в документе IEEE 802. Этот документ разделен на части. IEEE 802.1 содержит введение в стандарты и описание примитивов. IEEE 802.2 описывает протокол LLC (Logical Link Control – управление логическим каналом), который является верхней частью канального протокола. Стандарты с IEEE 802.3 по IEEE 802.5 описывают протоколы CSMA/CD для локальных сетей, шину с маркером и кольцо с маркером. Каждый стандарт покрывает физический уровень и MAC-подуровень. К их изучению мы и переходим.

Стандарт IEEE 802.3 и Ethernet

Стандарт IEEE 802.3 относится к 1-настойчивым протоколам CSMA/CD для локальных сетей. Напомним, что прежде чем начать передачу, станция, использующая такой протокол, опрашивает канал. Если он занят, то она ждет и как только он освободится, она начинает передачу. Если несколько станций одновременно начали передачу, то возникает коллизия. Тут же передача прекращается. Станции ожидают некоторый случайный отрезок времени, и все начинается сначала. Стандарт IEEE 802.3 имеет очень интересную историю. Начало положила ALOHA. Потом компания XEROX построила CSMA/CD канал на 2,94 Мбит/сек., объединивший 100 персональных компьютеров на 1 километре кабеля. Эта система была названа Ethernet (сетевой эфир) по аналогии с люминофорным эфиром, который был той средой, которая передавала свет. Когда Максвелл теоретически описал электромагнитное излучение, долгое время считалось, что оно распространяется в некоей среде - эфире. И лишь в 1887 году Мейхельсон и Морли экспериментально показали, что электромагнитное излучение может распространяться в вакууме. Ethernet Xerox’а получил такой большой успех, что Xerox, DEC и Intel решили объединиться и создали Ethernet 10 Мбит/сек. Эта разработка и составила основу стандарта IEEE 802.3. Отличие стандарта от оригинальной разработки состояло в том, что стандарт охватывал все семейство 1-настойчивых алгоритмов, работающих со скоростью от 1-10 Мбит/сек. Есть отличия в заголовке кадров. Стандарт определяет также параметры физической среды для 50-омного коаксиального кабеля.

IEEE 802.3 Кабели

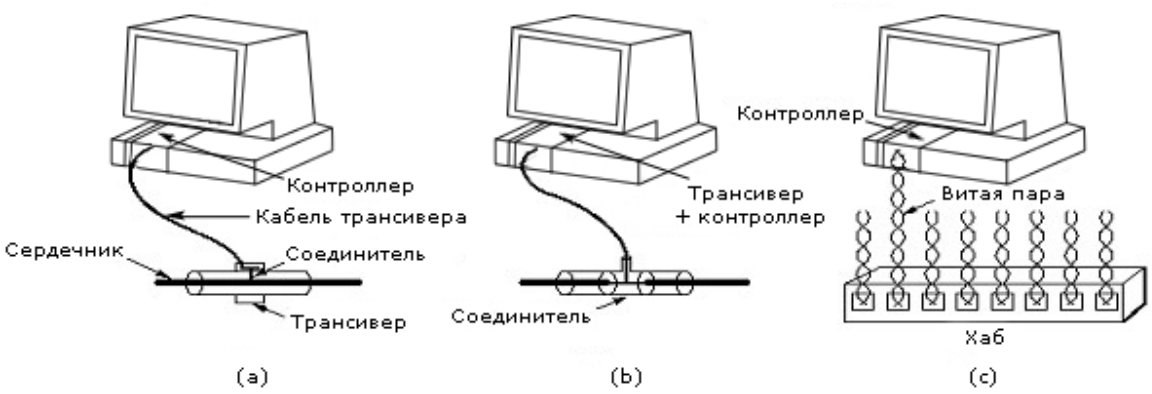
Всего по стандарту допускается четыре категории кабелей, которые перечислены в таблице 4-15. Исторически первым был так называемый «толстый» Ethernet - 10Base5. Это желтого цвета кабель с отметками через каждые 2,5 метра, которые указывают, где можно делать подключения. Подключение делается через специальные розетки с трансивером, которые монтируются прямо на кабеле. 10Base5 означает, что кабель обеспечивает пропускную способность на 10 Мбит/сек., использует аналоговый сигнал, и максимальная длина сегмента равна 500 метров.

Таблица 4-15. Наиболее распространенные средства передачи данных стандарта IEEE 802

Название	Тип кабеля	Максимальная длина сегмента	Кол-во узлов на сегмент	Преимущества
10Base5	Толстый коаксиал	500 м	100	Подходит для магистралей
10Base2	Тонкий коаксиал	200 м	30	Самый дешевый
10Base-T	Витая пара	100 м	1024	Простое обслуживание
10Base-F	Оптоволокно	2000 м	1024	Идеально для соединения зданий

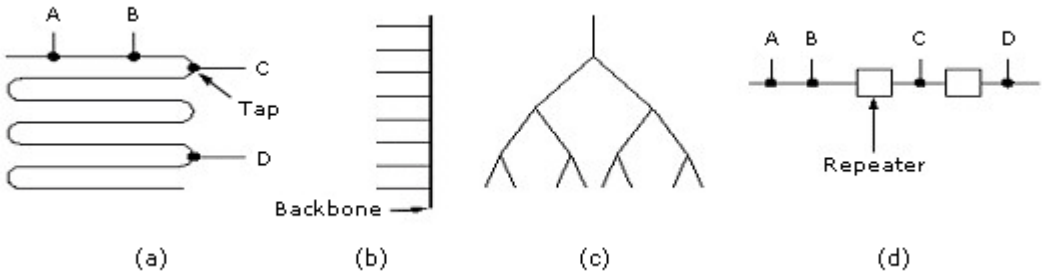
Вторым исторически появился кабель 10Base2 - «тонкий» Ethernet. Это более простой в употреблении кабель с простым подключением через BNC-коннектор. Этот коннектор представляет собой Т-образное соединение коаксиальных кабелей. Кабель для тонкого Ethernet дешевле. Однако его сегмент не должен превосходить 200 метров и содержать более 30 машин. Проблемы поиска обрыва, частичного повреждения кабеля или плохого контакта в коннекторе привели к созданию совершенно иной кабельной конфигурации на витой паре. Здесь каждая машина соединена со специальным устройством - хабом (hub) витой парой. Этот способ подключения называется 10Base-T. Данные три способа подключения показаны на рисунке 4-16. В 10Base5 (рисунок 4-16 (a)) трансивер размещается прямо на кабеле. Он отвечает за обнаружение несущей частоты и коллизий. Когда трансивер обнаруживает коллизию, он посылает специальный сигнал по кабелю, чтобы гарантировать, что другие трансиверы услышат коллизию. Трансивер на кабеле соединяется с компьютером трансиверным кабелем. Его длина не должна превосходить 50 метров. Он состоит из 5 витых пар. Две - для передачи данных к компьютеру и от него, две - для передачи управляющей информации в обе стороны, и пятая пара - для подачи питания на трансивер. Некоторые трансиверы позволяют подключать к себе до восьми машин.

Рисунок 4-16. Три способа подключения по стандарту IEEE 802



Трансиверный кабель подключается к контроллеру в компьютере. На этом контроллере есть специальная микросхема, которая отвечает за прием кадров и их отправку, проверку и формирование контрольной суммы. В некоторых случаях она отвечает за управление буферами на канальном уровне, очередью буферов на отправку, прямой доступ к памяти машины и другие вопросы доступа к сети. У 10Base2 трансивер расположен на контроллере. Каждая машина должна иметь свой индивидуальный трансивер (рисунок 4-16 (b)). У 10Base-T трансивера нет вовсе (рисунок 4-16 (c)). Машина соединяется с хабом витой парой, длина которой не должна превосходить 100 метров. Вся электроника сосредоточена в хабе. Наконец, последний вид кабеля 10Base-F - оптоволоконный вариант. Он относительно дорог, но низкий уровень шума и длина одного сегмента - важные достоинства этого кабеля. На рисунке 4-17 показаны различные топологии использования Ethernet. Чтобы увеличить длину сегмента, используются репитеры. Это устройство физического уровня, которое отвечает за очистку, усиление и передачу сигнала. Репитеры не могут отстоять более чем на 2,5 км, и на одном сегменте их не может быть более четырех.

Рисунок 4-17. Топологии Ethernet



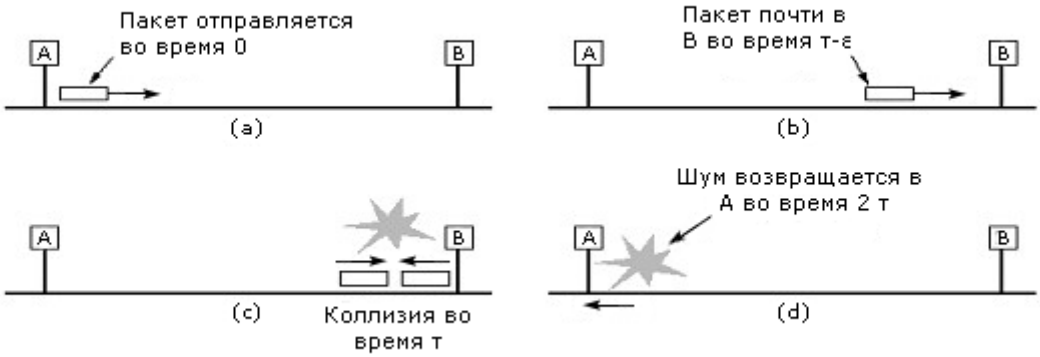
IEEE 802.3: протокол MAC-подуровня

Структура кадра в IEEE 802.3 показана на рисунке 4-19. Кадр начинается с преамбулы - 7 байт вида 10101010, которая в манчестерском коде на скорости 10 МГц обеспечивает 5,6 мксек для синхронизации приемника и передатчика. Затем следует стартовый байт 10101011, обозначающий начало передачи.

Рисунок 4-19. Структура кадра IEEE 802.3

Хотя стандарт допускает двух- и шестибайтные адреса, для 10Base используются только 6-байтные. 0 в старшем бите адреса получателя указывает на обычный адрес. Если там 1, это признак группового адреса. Групповой адрес позволяет обращаться сразу к нескольким станциям одновременно. Если адрес получателя состоит из одних единиц - это вещательный адрес, т.е. этот кадр должны получить все станции в сети. Другой интересной возможностью адресации является различение локального адреса и глобального. На то, какой адрес используется, указывают 46 бит. Если этот бит 1 - это локальный адрес, который устанавливает сетевой администратор и вне данной сети этот адрес смысла не имеет. Глобальный адрес устанавливает IEEE и гарантирует, что нигде в мире нет такого второго. С помощью 46 битов можно получить $7x10^{13}$ глобальных адресов. Поле длины указывает на длину поля данных. Она может быть от 0 до 1500 байт. То, что поле данных может иметь длину 0, вызывает проблему для обнаружения коллизий. Поэтому IEEE 802.3 предписывает, что кадр не может быть короче 64 байт. Если длина поля данных недостаточна, то поле Pad компенсирует нехватку длины. Ограничение на длину кадра связано со следующей проблемой. Если кадр короткий, то станция может закончить передачу прежде, чем начало кадра достигнет самого отдаленного получателя. В этом случае она может пропустить коллизию и ошибочно считать, что кадр доставлен благополучно. Пусть τ - минимальное время распространения сигнала до самой удаленной станции. Тогда минимальная длина кадра должна быть такой, чтобы время передачи кадра такой длины занимало не менее 2τ секунд. Эту ситуацию поясняет рисунок 4-20. Для IEEE 802.3 (2,5 км и четырех репитерах) это время равно 51,2 мксек., что соответствует 64 байтам. При больших скоростях длина кадра должна быть еще больше. Например, на скорости 1 Гбит при длине сегмента 2,5 км она будет равна 6400 байтам. И это становится проблемой при переходе на высокие скорости передачи.

Рисунок 4-20. Обнаружение коллизии



Последнее поле - контрольная сумма, которая формируется с помощью CRC-кода. Мы рассматривали эти коды в главе 3.

Двоичный экспоненциальный алгоритм задержки

Теперь рассмотрим, как определяется случайная величина задержки при возникновении коллизий. При возникновении коллизии время разбивается на слоты длиной, соответствующей наибольшему времени распространения сигнала в оба конца (2τ). Для 802.3, как уже было указано, это время при длине линии 2,5 км и четырех репитерах равно 51,2 мксек. При первой коллизии станции, участвовавшие в ней, случайно выбирают 0 или 1 слот для ожидания. Если они выберут одно и то же число, то коллизия возникнет опять. Тогда выбор будет происходить среди чисел 0, 2^i, 1, где i - порядковый номер очередной коллизии. После 10 коллизий число слотов достигает 1023 и далее не увеличивается, после 16 коллизий Ethernet-контроллер фиксирует ошибку и сообщает о ней более высокому уровню стека протоколов. Этот алгоритм называется алгоритм двоичной экспоненциальной задержки. Он позволяет динамически подстраиваться под число конкурирующих станций. Если для каждой коллизии случайный интервал был бы равен 1023, то вероятность повторной коллизии для двух станций была бы пренебрежимо мала. Однако среднее время ожидания разрешения коллизии было бы сотни слотов. Если бы случайный интервал был бы постоянно 0 или 1, то при 100 станциях разрешение коллизии потребовало бы годы, так как 99 станций должны были бы случайно выбрать, скажем, 0 и лишь одна - 1.

Производительность IEEE 802.3

Здесь мы рассмотрим производительность 802.3 при условии плотной и постоянной нагрузки. У нас есть k станций, всегда готовых к передаче. С целью упрощения анализа при коллизиях мы будем рассматривать не алгоритм двоичной экспоненциальной задержки, а постоянную вероятность повторной передачи в каждом слоте. Если каждая станция участвует в состязаниях в слоте с вероятностью p, то вероятность A, что некоторая станция захватит канал в этом слоте, равна

$$A = kp(1-p)^{k-1}$$

A достигает максимума при p=1/k , A→1/e при k→∞. Вероятность, что период состязаний будет иметь j слотов, равна $A(1-A)^{j-1}$. Отсюда среднее число слотов в состязаниях равно

$$\sum_{j=0}^{\infty} jA(1-A)^{j-1} = \frac{1}{A}$$

Так как каждый слот имеет длительность 2τ, то средний интервал состязаний w равен 2τ/A. Предполагая оптимальное значение p, $w \leq 2\tau e \approx 5.4\tau$. Если передача кадра средней длины занимает m сек, то при условии большого числа станций, постоянно имеющих кадры для передачи, эффективность канала равна

$$\frac{m}{m + 2\tau/A}$$

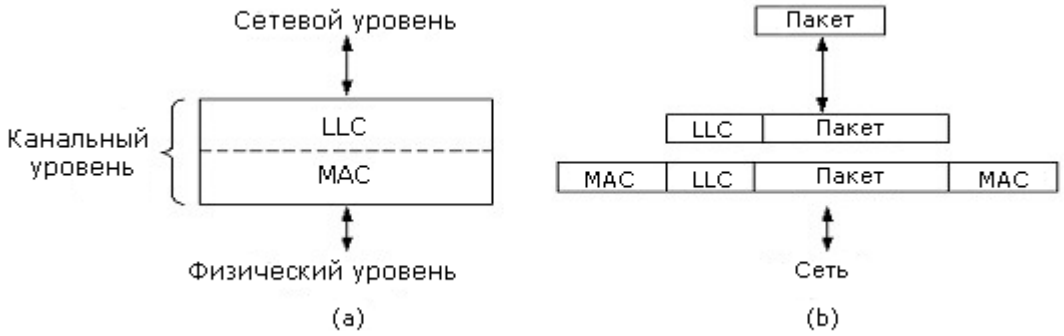
Из этой формулы видно, что чем длиннее кабель, тем хуже эффективность, т.к. растет длительность периода состязаний. При длительности 51,2 мксек, что соответствует 2,5 км при четырех репитерах и скорости передачи 10 Мбит/сек., минимальный размер кадра - 512 бит, или 64 байта. Как показали экспериментальные исследования, предположение о том, что трафик в канале может быть описан распределением Пуассона (опираясь на это предположение, было сделано очень много теоретических исследований) не верно. Увеличение длительности наблюдений не сглаживает трафик, не дает определенного среднего значения.

Стандарт IEEE 802.2: управление логическим каналом

До сих пор мы ни слова не сказали о надежности коммуникаций через 802.x. Как правило, сети типа 802 обеспечивают дейтаграммный сервис. Иногда этого достаточно. Например, при передаче IP-пакетов никаких гарантий не требуется и не предполагается. Поэтому IP-пакет просто вставляется в кадр и передается. Если он потерялся, то так тому и быть. Тем не менее, есть системы, где нужен контроль ошибок, а управление

потоком весьма желательно. IEEE создало такой протокол, который работает над всеми 802.x. Он называется LLC (Logical Link Control). Он скрывает различия между версиями 802, определяя единый интерфейс и формат для сетевого уровня. LLC-протокол образует верхний уровень канального протокола с MAC-протоколом под ним так, как это показано на рисунке 4-29.

Рисунок 4-29. Действие LCC-протокола: (a) его расположение; (b) форматы протокола



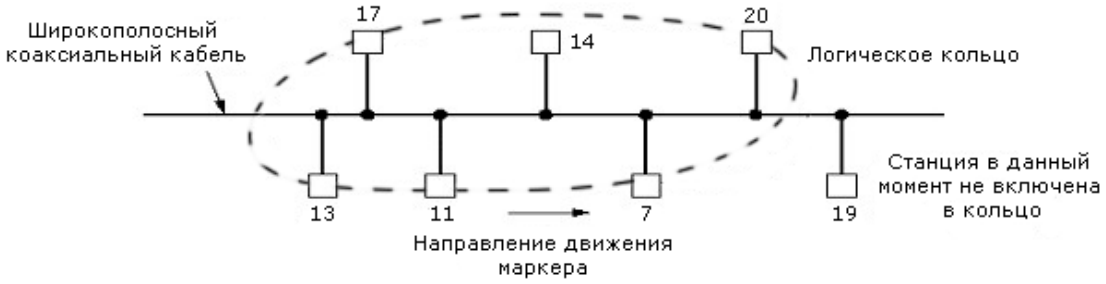
LLC предоставляет три вида сервиса: ненадежный: дейтаграммы без уведомления, дейтаграммы с уведомлением и надежный сервис, ориентированный на соединение.

Стандарты IEEE 802.x для локальных и муниципальных сетей: Стандарт IEEE 802.4 – шина с маркером (область применения, протокол MAC подуровня, логическая поддержка логического кольца). Стандарт IEEE 802.2: управление логическим каналом.

Стандарт IEEE 802.4: шина с маркером

Стандарт 802.3 получил очень широкое распространение. Однако там, где возникала потребность в режиме реального времени, он вызывал нарекания. Во-первых, потому что с ненулевой вероятностью станция может ожидать сколь угодно долго отправки кадра. В стандарте нет понятия приоритета кадра, что очень важно для приложений реального времени. Простейшая система с заранее известным наихудшим случаем ожидания - кольцо. Если есть n станций, соединенных в кольцо, и передача кадра занимает T сек., то максимальное время ожидания передачи кадра будет не более nT. Специалистам по системам реального времени нравилась идея кольца, но не нравилась ее физическая реализация. Во-первых, кольцо не надежно - обрыв в одном месте разрушает всю систему. Во-вторых, оно плохо соответствовало топологии многих сборочных линий на заводах. В результате был разработан стандарт, который объединял достоинства 802.3 с гарантированным наихудшим временем передачи и приоритетностью кадров. Этот стандарт был назван 802.4 и описывал шину с маркером. Физически шина с маркером имеет линейную или древовидную топологию. Логически станции объединены в кольцо (рисунок 4-22), где каждая станция знает своего соседа справа и слева. Когда кольцо инициализировано, станция с наибольшим номером может послать первый кадр. После этого она передает разрешение на передачу кадра своему непосредственному соседу, посылая ему специальный управляющий кадр - маркер. Передача кадра разрешена только той станции, которая владеет маркером. Так как маркер один, то всегда только одна станция может осуществлять передачу, и коллизий не возникает.

Рисунок 4-22. Маркерная шина



Важно отметить, что на порядок передач влияет только логические номера станций, а не их физическое размещение. Маркер передается только логическому соседу. Естественно, протокол должен учитывать случай, когда станция подключается к кольцу в ходе функционирования. 802.4 MAC - очень сложный протокол, который поддерживает 10 таймеров и более 24 внутренних переменных. На физическом уровне 802.4 использует коаксиальный 75-омный кабель, три разные схемы аналоговой модуляции, скорость передачи - 1,5 и 10 Мбит/сек. Он полностью несовместим с физическим уровнем 802.3.

MAC-протокол для шины с маркером

При инициализации станции образуют кольцо в соответствии с их адресами от старших к младшим. Маркер передают от станций с большими адресами к станциям с меньшими адресами. Каждый раз, когда станция получает маркер, она может передавать кадры в течение определенного промежутка времени. После этого она должна передать маркер следующей станции. Если кадры достаточно короткие, то может быть послано несколько последовательных кадров. Если у станции нет данных для передачи, то она передает маркер дальше, немедленно по его получении. Шина с маркером определяет четыре приоритета для кадров: 0, 2, 4 и 6. Для простоты можно представить, что станция разделена внутри на четыре подстанции, по одной на уровень приоритета. Как только кадр поступает сверху, он распределяется на одну из подстанций в соответствии с приоритетом. Таким образом, каждая подстанция имеет свою очередь кадров на передачу. Когда маркер поступил по кабелю, он попадает на подстанцию с приоритетом 6. Если у нее есть кадр на передачу, она его передает, если нет, то маркер передается подстанции с приоритетом 4. Эта подстанция передает свои кадры в течение своего интервала времени, либо по истечении определенного временного промежутка передает маркер подстанции с приоритетом 2. Так продолжается до тех пор, пока либо подстанция с приоритетом 0 перешлет свои кадры, либо ее таймер исчерпается и она отдаст маркер следующей станции. Из приведенной схемы ясно, что подстанция с номером 6 имеет наивысший приоритет и в любом случае ее кадрам обеспечена некая гарантированная пропускная способность. Эта подстанция и используется для передачи трафика реального времени. Например, пусть имеется сеть из 50 станций, работающая на скорости 10 Мбит/сек. и настроенная так, что на подстанции с приоритетом 6 остается 1/3 пропускной способности, тогда каждая станция имеет гарантированно для приоритета 6 скорость не менее 67 Кбит/сек. Такая пропускная способность может быть использована для управления устройствами в масштабе реального времени. На рисунке 4-23 показан формат кадра для шины с маркером. Поле Preamble предназначено для синхронизации таймера получателя. Его длина не короче одного байта. Поля Start delimiter и End delimiter предназначены для распознавания начала и конца кадра. Они имеют специальную кодировку, которая не может встретиться у пользователя. Поэтому поля длины кадра не требуется. Поле Frame control отделяет управляющие поля от полей данных. Для кадров данных здесь указывается приоритет кадра. Это поле также используется станцией-получателем для подтверждения корректного или некорректного получения кадра. Для этого отправитель устанавливает в этом поле специальный индикатор подтверждения. При наличии такой установки станция-получатель, даже не имея маркера, может послать подтверждение. Без этого поля получатель был бы лишен возможности давать подтверждения - у него было бы маркера.

Рисунок 4-23. Формат кадра для шины с маркером

В управляющих кадрах это поле используется для указания типа кадра. Среди них передача маркера, всевозможные кадры для поддержки кольца, например, включение станции в кольцо и исключение станции из кольца. Поле адреса получателя и адреса отправителя такие же, как и в стандарте 802.3. В нем адреса могут быть 2-байтные или 6-байтные. Поле данных может иметь длину не более 8182 байта при 2-байтном адресе и 8174 - при 6-байтном адресе. Это в пять раз длиннее, чем в 802.3, т.к. в нем необходимо предотвратить захват одной станцией канала надолго. Здесь это не опасно, т.к. есть таймер, а для реального времени бывает полезно иметь длинные кадры. Контрольная сумма, как и в 802.3, используется для обнаружения ошибок.

Поддержка логического кольца

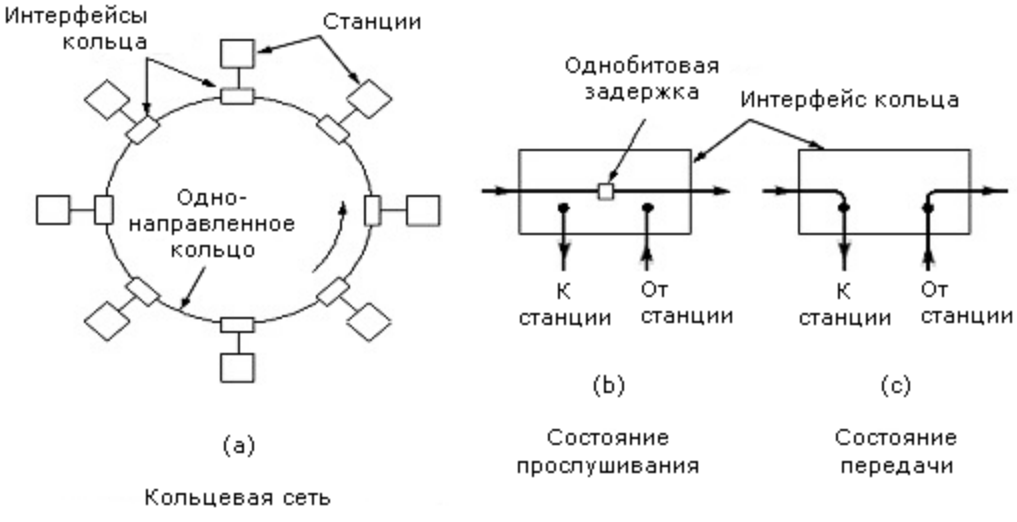
Поддержка логического кольца в основном связана с проблемами включения и выключения станций. MAC-подуровень 802.4 детально описывает алгоритм, позволяющий сохранять известным наихудший случай при передаче маркера. Ниже мы рассмотрим кадры, которые используются в этом случае. Когда кольцо установлено, интерфейс каждой станции хранит адреса предшествующей и последующей станции. Периодически держатель маркера рассылает один из кадров SOLICIT_SUCCESSOR, предлагая новым станциям присоединиться к кольцу. В этом кадре указаны адрес отправителя и адрес следующей за ним станции в кольце. Станции с адресами в этом диапазоне адресов могут присоединиться к кольцу. Таким образом, сохраняется упорядоченность (по возрастанию) адресов в кольце. Если ни одна станция не откликнулась на SOLICIT_SUCCESSOR, то станция-обладатель маркера закрывает окно ответа и продолжает функционировать, как обычно. Если есть ровно один отклик, то откликнувшаяся станция включается в кольцо и становится следующей в кольце. Если две или более станции откликнулись, то фиксируется коллизия. Станция-обладатель маркера запускает алгоритм разрешения коллизий, посылая кадр RESOLVE_CONTENTION. Этот алгоритм - модификация алгоритма обратного двоичного счетчика на два разряда. У каждой станции в интерфейсе есть два бита, устанавливаемых случайно. Их значения 0, 1, 2 и 3. Значение этих битов определяют величину задержки при отклике станции на приглашение подключиться к кольцу. Значения этих бит переустанавливаются каждые 50 мсек. Процедура подключения новой станции к кольцу не нарушает наихудшее гарантированное время для передачи маркера по кольцу. У каждой станции есть таймер, который сбрасывается, когда станция получает маркер. Прежде чем он будет сброшен, его значение сравнивается с некоторой величиной. Если оно больше, то процедура подключения станции к кольцу не запускается. В любом случае за один раз подключается не более одной станции. Теоретически станция может ждать подключения к кольцу сколь угодно долго, на практике, не более нескольких секунд. Однако с точки зрения приложений реального времени это одно из наиболее слабых мест 802.4. Отключение станции от кольца очень просто. Станция X с предшественником S и последователем P шлет кадр SET_SUCCESSOR, который указывает P, что отныне его предшественником является S. После этого X прекращает передачу. Инициализация кольца - это специальный случай подключения станции к кольцу. В начальный момент станция включается и слушает канал. Если она не обнаруживает признаков передачи, то она генерирует маркер CLAIM_TOKEN. Если конкурентов не обнаружилось, то она генерирует маркер сама и устанавливает кольцо из одной станции. Периодически она генерирует кадры SOLICIT_SUCCESSOR, приглашая другие станции включиться в кольцо. Если в начальный момент сразу две станции были включены, то запускается алгоритм обратного двоичного счетчика с двумя разрядами. Из-за ошибок передач и сбоев оборудования могут возникать проблем с передачей маркера. Например, станция передала маркер соседней, а та неожиданно «грохнулась» - что делать? Стандарт дает прямолинейное решение - передав маркер, станция слушает. Если не последует передач кадра или маркера, то маркер посылается вторично. Если и при повторной передаче маркера ничего не последовало, то станция посылает кадр WHO_FOLLOWS, где указан не отвечающий сосед. Увидев этот кадр, станция, для которой не отвечающая станция -

предшественник, шлет кадр SET_SUCCESSOR и становится новым соседом. При этом не отвечающая станция за плохое поведение исключается из кольца. Теперь предположим, что остановилась не только следующая станция, но и следующая за ней. В этом случае запускается новая процедура посылкой кадра SOLICIT_SUCCESSOR_2. В ней участвует процедура разрешения конфликтов. При этом все, кто хочет подключиться к кольцу, могут это сделать. Фактически кольцо переустанавливается. Другой вид проблем возникает, когда останавливается держатель маркера и маркер исчезает из кольца. Эта проблема решается запуском процедуры инициализации кольца. У каждой станции есть таймер, который сбрасывается каждый раз, когда маркер появляется. Если значение этого таймера превысит некоторой заранее установленное значение, то станция генерирует кадр CLAIM_TOKEN. При этом запускается алгоритм обратного двоичного счетчика. Если оказалось два и более маркера на шине, станция, владеющая маркером, увидев передачу маркера на шине, сбрасывает свой маркер. Так повторяется до тех пор, пока не останется ровно один маркер в системе.

Стандарты IEEE 802.x для локальных и муниципальных сетей: Стандарт IEEE 802.5 – кольцо с маркером (область применения, протокол MAC подуровня, логическая поддержка кольца). Стандарт IEEE 802.2: управление логическим каналом.

Сети с кольцевой топологией известны давно и используются широко. Среди их многочисленных достоинств есть одно особенно важное - это не среда с множественным доступом, а последовательность соединений точка-точка, образующих кольцо. Соединения точка-точка хорошо изучены, могут работать на разных физических средах: витая пара, коаксиал или оптоволокно. Способ передачи в основном цифровой, в то время как у 802.3 есть значительный аналоговый компонент. Кольцо также представляет справедливую среду с известной верхней границей доступа к каналу. В силу этих причин IBM выбрало кольцо как основу своего стандарта, а IEEE включило его как стандарт 802.5 - кольцо с маркером. Важной проблемой при создании кольцевой сети является «физическая длина» бита. Пусть данные передаются со скоростью R Мбит/сек. Это значит, что через каждые 1/R мксек. на линии появляется бит. Учитывая, что сигнал распространяется со скоростью 200 м/мксек., то один бит занимает 200/R метров кольца. Отсюда, при скорости 1 Мбит/сек. и длине окружности 1 км кольцо вмещает не более 5 бит одновременно. Значение этого факта станет ясно позднее. Как уже отмечалось, кольцо - это последовательность соединений точка - точка. Бит, поступая на интерфейс, копируется во внутренний буфер интерфейса и передается по кольцу дальше (см. рисунок 4-25). В буфере бит может быть проанализирован и, возможно, изменен. Эти операции вносят задержку на один бит в каждом интерфейсе.

Рисунок 4-25. Устройство кольца

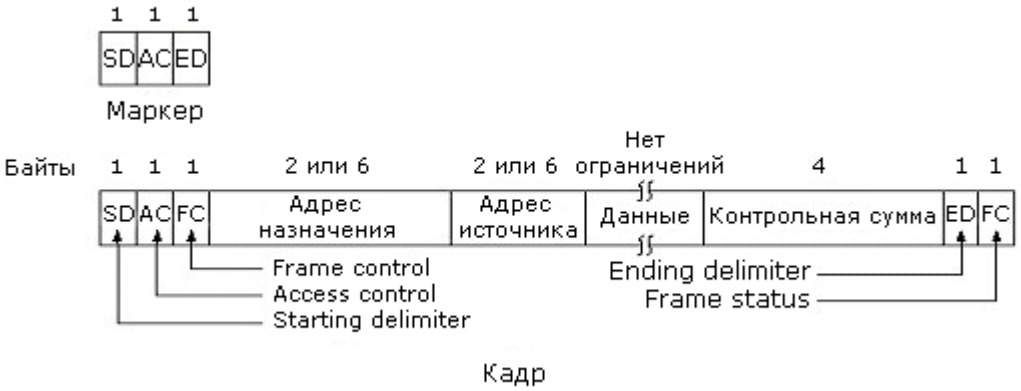


Пока станциям нечего передавать, в кольце циркулирует маркер - особая последовательность бит. Если станции нужно передать данные, она должна захватить маркер и удалить его из кольца. Это достигается изменением одного бита в 3-х байтном маркере, в результате чего маркер тут же превращается в заголовок обычного кадра. Поскольку в кольце может быть только один маркер, то только одна станция может передавать данные. Так в сети «кольцо с маркером» решается вопрос доступа. Как следствие конструкции кольца с маркером, сеть должна иметь достаточную протяженность, чтобы маркер мог уместиться в ней целиком, даже когда все станции находятся в ожидании. Задержки складываются из двух компонентов: 1 бит - задержка на интерфейсе станции и задержка на распространение сигнала. Учитывая, что станции могут выключаться, например, на ночь, следует, что на кольце должна быть искусственная задержка, если оно недостаточно длинное. Интерфейс станций может работать в двух режимах: прослушивания и передачи. В режиме прослушивания он лишь копирует бит в свой буфер и передает этот бит дальше по кольцу. В режиме передачи, предварительно захватив маркер, интерфейс разрывает связь между входом и выходом и начинает передачу. Чтобы быть способным за однобитовую задержку переключиться из одного режима в другой, интерфейс должен предварительно забуферизовать данные для передачи. По мере распространения передаваемых битов по кольцу они будут возвращаться к передающей станции. Она должна убирать их либо с линии, либо в буфер для последующего анализа с целью повышения надежности передачи, либо просто сбрасывая. Поскольку кадр целиком никогда не появляется на линии, то архитектура сети кольца с маркером не накладывает никаких ограничений на длину кадра. Как только станция закончила передачу и удалила последний переданный бит с линии, она должна сгенерировать маркер и переключиться в режим прослушивания. В такой сети просто уведомлять о получении кадра. В каждом кадре есть бит уведомления. Станция-получатель устанавливает этот бит при получении кадра. Станция-отправитель при возвращении кадра анализирует этот бит и может определить, был ли этот кадр получен. Так же можно поступать и с проверкой контрольной суммы, главное, чтобы эта проверка могла быть выполнена за однобитовую задержку. При малой загрузке станции в кольце сразу могут передавать свои сообщения. По мере роста загрузки у станций будут расти очереди на передачу и они в соответствии с кольцевым алгоритмом будут захватывать маркер и вести передачу. Постепенно загрузка кольца будет расти, пока не достигнет 100%. Теперь обратимся **непосредственно к стандарту IEEE 802.5**. На физическом уровне он использует витую пару со скоростью 1 или 4 Мбит/сек., хотя IBM позднее ввела 16 Мбит/сек. Сигнал на линии кодируется с помощью дифференциального манчестерского кода, используя запрещенные комбинации low-low и high-high для управляющих байтов. С кольцом связана одна серьезная проблема - если связь в кольце где-то нарушается, то вся конфигурация становится неработоспособной. Проблема решается с помощью так называемого кабельного центра. Это решение показано на рисунке 4-26. В случае, если какая-то станция выходит из строя, реле замыкается и станция исключается из кольца. Реле может управляться и программно, выводя временно станцию из кольца, например, для тестирования. Хотя стандарт 802.5 непосредственно не предписывает использование кабельного центра, на практике он часто используется с целью повышения надежности и удобства обслуживания сети. Вместо станции к кабельному центру может присоединяться другой кабельный центр. Таким образом, кабельные центры могут объединяться в структуры, подобно тому как хабы соединяются в 802.3. Однако форматы и протоколы у них разные.

Кольцо с маркером: протокол MAC-подуровня

Основные операции MAC-протокола довольно просты. При отсутствии данных по кольцу циркулирует 3-байтный маркер. Как только какой-то станции надо передать данные, она инвертирует специальный бит в маркере с 0 на 1, превращая маркер в стартовую последовательность байтов для передачи кадров и добавляя данные для передачи, как это показано на рисунке 4-27.

Рисунок 4-27. Устройство маркера и кадра передачи данных



В нормальных условиях станция-отправитель должна постоянно забирать с линии биты, которые возвращаются к ней, обойдя кольцо. Даже очень длинное кольцо вряд ли будет способно вместить короткий кадр. Поэтому ранее посланные биты начнут возвращаться прежде, чем станция закончит передавать кадр. Станция может держать маркер не более 10 мсек., если при инсталляции не было установлено иного значения. Если после отправки кадра остается достаточно времени, то посылаются следующие. После того как посланы все кадры или истекло время владения маркером, станция обязана сгенерировать маркер и вернуть его на линию. Байты Starting delimiter и Ending delimiter отмечают начало и конец кадра соответственно. Они содержат запрещенные в дифференциальных манчестерских кодах последовательности. Байт Access control содержит маркерный бит, Monitor bit, Priority bits, Reservations bits (они будут описаны позднее). Поля Destination address и Source address такие же, как и в стандартах 802.3 и 802.4. За ними следует поле данных, которое может быть сколь угодно длинное, лишь бы его передача уместилась во время владения маркером. Поле контрольной суммы такое же, как и в 802.3 и 802.4. Байт, которого нет ни в 802.3 ни в 802.4 - Frame status. В нем есть биты A и C. Когда кадр поступает к станции-получателю, ее интерфейс инвертирует бит A. Если кадр успешно скопирован, то инвертируется и бит C. Кадр может быть не скопирован в силу разных причин: задержки, отсутствия места в буфере и т.п. Когда станция-получатель снимает ранее посланные биты с линии, она анализирует биты A и C. По их комбинации она может определить, успешно ли прошла передача. Возможны три комбинации значений этих битов:

1. A=0, C=0 - получатель отсутствует
2. A=1, C=0 - получатель есть, но кадр не принят
3. A=1, C=1 - получатель есть и кадр принят

Биты A и C обеспечивают автоматическое уведомление о получении кадра. Если кадр почему-то не был принят, то у станции есть несколько попыток передать его. Биты A и C дублируются в байте Frame status с целью повысить надежность, так как этот байт не подпадает под контрольную сумму. Ending delimiter содержит специальный бит, который устанавливает интерфейс любой станции, если он обнаруживает ошибку. Там также есть бит, которым можно помечать последний кадр в логической последовательности кадров. В 802.5 есть тщательно проработанная схема работы с приоритетами. В среднем байте 3-байтного маркера есть поле, отведенное для приоритета. Если станции надо передать кадр с приоритетом n, то ей придется ждать, пока появится маркер с приоритетом, меньшим или равным n. Кроме того, когда кадр с данными проходит по кольцу, станция может указать значение приоритета, который ей нужен. Для этого она записывает нужное значение в поле Reservation bits. Однако если в нем уже записан более высокий приоритет, станция не может этого делать. После завершения передачи кадра генерируется маркер с приоритетом, зарезервированном в этом кадре. Описанный механизм приоритетов имеет один недостаток: приоритет все время растет. Поэтому 802.5 предусматривает довольно сложные правила понижения приоритета. Суть этих правил сводится к тому, что станция, установившая наивысший приоритет, обязана его понизить после передачи кадра. Заметим, что работа с приоритетами в кольце с маркером и шине с маркером организована по-разному. В шине с маркером каждая станция получает свою долю пропускной способности канала. В кольце с маркером станция с низким приоритетом может ждать сколь угодно долго

маркера с надлежащим приоритетом. Это различие отражает различие подходов двух стандартов: что важнее - высокоприоритетный трафик или равномерное обслуживание всех станций.

Поддержка кольца

В стандарте 802.5 поддержка кольца организована иначе, чем это сделано в 802.4. Там был создан довольно длинный протокол для полностью децентрализованной поддержки. В 802.5 предусмотрено, что в кольце всегда есть станция-монитор, контролирующая кольцо. Если станция-монитор по какой-либо причине потеряет работоспособность, есть протокол выбора и объявления другой станции-монитора на кольце. Любая станция способна быть монитором. При включении или если какая-то станция заметит отсутствие монитора, она посылает кадр CLAIM_TOKEN. Если она первая, кто послал такой кадр, то она и становится монитором. Среди задач, которые должен решать монитор, есть следующие: слежение за наличием маркера, выполнение определенных действий, если нарушено, устранение грязи или беспризорных кадров. Такие кадры могут появиться, если станция начала передачу и не закончила по какой-либо причине. Монитор обнаруживает отсутствие маркера с помощью специального таймера, отмечающего время отсутствия маркера на кольце. Если значение этого таймера превысит некоторое значение, то считается, что маркер потерян, и монитор обязан принять надлежащие меры. При появлении грязи, т.е. кадра с неверным форматом или контрольной суммой, монитор снимает его с линии и генерирует маркер. Беспризорные кадры монитор обнаруживает с помощью Monitor bit в байте Access control. Когда кадр проходит через монитор первый раз, монитор устанавливает этот бит в единицу. Поэтому, если очередной кадр пришел с единицей в этом бите, то этот кадр не был принят и монитор должен его удалить из кольца. Важной функцией монитора является установка задержки на кольце, достаточной, чтобы в кольце уместился 24-битный маркер. Монитор не может определить, где произошел разрыв кольца. Поэтому, если какая-то станция заметит отсутствие соседа, она генерирует BEACON-кадр. По мере распространения этого кадра по кольцу определяется, какие станции живы, а какие нет, и последние удаляют из кольца с помощью кабельного центра. Любопытно сравнить подходы к управлению кольцом в 802.4 и 802.5. Комитет, разработавший шину с маркером, видимо, до смерти боялся допустить хоть какую-нибудь централизацию в системе. Поэтому все станции в системе одинаковы и лишь одна получает особые права, но лишь на определенный промежуток времени. В 802.5 разрешен монитор - это явно централизованный механизм. Разработчики считали, что станции достаточно надежные и редко выходят из строя. Поэтому, если монитор генерирует с заданной периодичностью кадр ACTIVE MONITOR PRESENT, никто не попытается его заменить. Нет механизма, позволяющего сменить существующий монитор. Эти различия отражают различия приложений, которые имели в виду представители этих комитетов. Разработчики 802.4 имели дело с приложениями на заводах, где много металла, есть агрессивные среды и т.п. Поэтому возможны отказы станций, разрушения сети. Разработчики 802.5 ориентировались на офисные приложения, где защита от отдельных ошибок вряд ли оправдает серьезные затраты на нее.

Сравнение стандартов 802.3, 802.4 и 802.5. Стандарт IEEE 802.2: управление логическим каналом.

Итак, есть три разные несовместимые MAC-среды. Какую из них выбрать? Здесь мы сравним все три стандарта 802. Начать следует с того, что все три стандарта используют примерно одинаковую технологию и имеют примерно одинаковую производительность. Инженеры могут бесконечно спорить о том, что лучше - коаксиал или витая пара, но рядовому пользователю, компьютерному обывателю это, как правило, не важно.

Достоинства 802.3

1. Очень широко используется, имеет огромную инсталляционную базу.
2. Широко известен, многие инженеры знают как с ним работать.
3. Протокол простой.
4. Станция может быть подключена без остановки сети.
5. Используется пассивный кабель, модемы и прочее оборудование не требуется.
6. При малой загрузке задержки практически равны 0.

Недостатки 802.3

1. Есть значительный аналоговый компонент.

2. Минимальная длина кадра 64 байта.
3. Не детерминирован, что плохо для приложений реального времени.
4. Нет приоритетов.
5. Ограничена длина кабеля.
6. С ростом скорости передачи время состязаний не сокращается, следовательно минимальная длина кадра растет.

Достоинства 802.4

1. Используется обычный телевизионный кабель, высоко надежный.
2. Более детерминирован чем 802.3.
3. Минимальная длина кадра короче.
4. Поддерживает приоритеты.
5. Есть гарантированная доля трафика.
6. Прекрасная эффективность и пропускная способность при высокой загрузке.
7. Коаксиальный кабель может поддерживать несколько каналов.

Недостатки 802.4

1. Используется много аналоговой аппаратуры.
2. Очень сложный протокол.
3. Большие задержки при низкой загрузке.
4. Используется относительно немногими пользователями.
5. Нельзя использовать оптоволокно.

Достоинства 802.5

1. Использует соединения точка-точка: полностью цифровое и простое в обращении.
2. Можно использовать любую физическую среду - от почтовых голубей до оптоволокна.
3. Стандартная витая пара дешева и проста в обращении.
4. Наличие кабельного центра делает кольцо с маркером единственным стандартом, где нарушения физической среды могут восстанавливаться автоматически.
5. Есть приоритеты, однако схема их установки не так справедлива, как в шине с маркером.
6. Небольшой размер минимального кадра и неограниченный размер передаваемого кадра.
7. Прекрасная пропускная способность при высокой загрузке, как и у 802.4.

Недостатки 802.5

1. Основной недостаток - наличие централизованного монитора, который является критическим компонентом.
2. Из-за схемы передачи маркера относительно большие задержки при небольшой загрузке.

Таким образом, все три платформы более или менее равны, и дело, скорее всего, будут решать отдельные нюансы, значения параметров, особенности конкретных приложений.

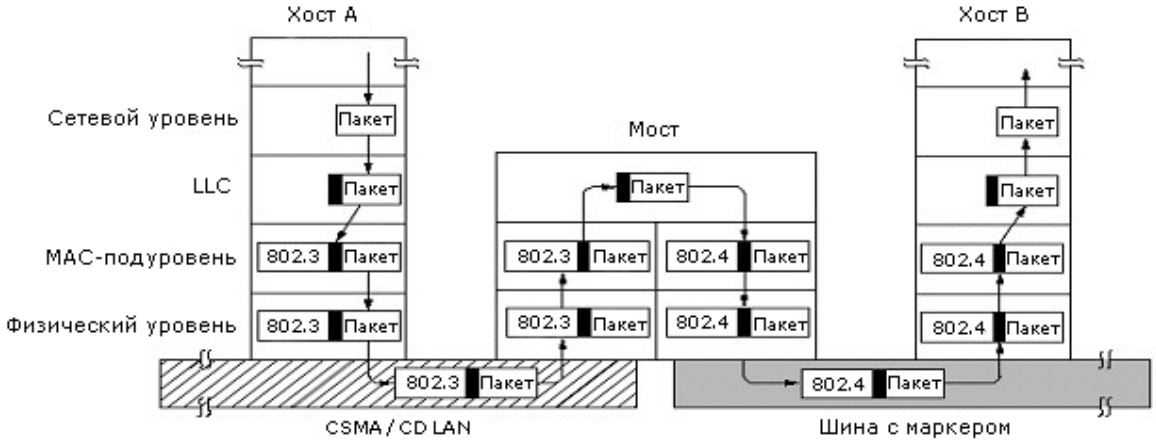
Мосты: организация, основные функции, принципы функционирования. Мосты из 802.x в 802.x. Сравнение мостов для 802.x.

Довольно часто в организации возникает потребность соединить между собой несколько ЛВС. Этой цели служат специальные устройства, называемые мосты, которые функционируют на уровне канала данных. Это означает, что это устройство не анализирует заголовки пакетов сетевого уровня и выше и, таким образом, может просто копировать пакеты IP, IPX, OSI, в то время как маршрутизаторы могут работать только с определенными пакетами. Далее мы рассмотрим устройство мостов, а частности, для соединения сетей в стандарте 802.3, 802.4, 802.5. Но прежде рассмотрим типичные ситуации, где применяются мосты. Их как минимум шесть:

1. Многие подразделения в организации имеют свои собственные локальные сети. Например, сети факультетов в университетах, отделы в институтах и т.п. В силу различий стоящих перед ним задач, они используют разные приложения и, что естественно, сети, отличные от сетей других подразделений. Рано или поздно наступает момент, когда необходимо интегрировать информационные потоки всей организации и, соответственно, объединять сети между собой.
2. Организация может занимать несколько зданий и, возможно, будет целесообразно в каждом здании иметь свою сеть, объединив их через мосты.
3. Иногда, при высоких рабочих нагрузках, приходится разбивать сеть на несколько, с целью локализации трафика в каждой подсети. Например, ясно, что класс рабочих станций для студентов лучше оформлять как самостоятельную сеть, локализовав трафик в этой сети: к чему распространять по всему факультету трафик с фотографиями и музыкой?
4. В некоторых случаях причиной для использования моста может служить большое расстояние между объединяемыми сетями. Дело в том, что, используя мост, можно увеличить длину сегмента.
5. Мост может увеличить надежность сети. В локальной сети один узел может нарушить работоспособность сети в целом. Мосты, размещенные в критических точках сети, подобно запасным пожарным выходам, могут заблокировать такой узел и предотвратить нарушение работы системы в целом.
6. Мост может повысить безопасность сети. Большинство интерфейсов в ЛВС имеют специальный режим, при котором компьютер получает все пакеты, проходящие в сети, а не только адресованные ему. Всякого рода злоумышленники и просто любопытствующая публика просто обожают эту возможность. С помощью правильно расставленных мостов можно добиться, чтобы определенный трафик проходил лишь по определенным маршрутам, где он бы не мог попасть в чужие руки.

Теперь, когда мы знаем, для чего нужны мосты, рассмотрим, как они работают. Рисунок 4-31 показывает работу простого двухпортового моста.

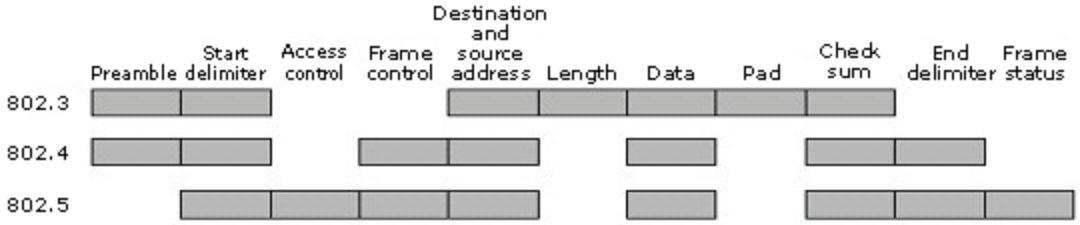
Рисунок 4-31. Двухпортовый мост между 802.3 и 802.4



Мосты из 802 в 802

На первый взгляд может показаться, что построить мост из 802 в 802 не сложно. Но это не так. У каждой из девяти возможных пар есть свои трудности. Но прежде чем начать рассматривать эти индивидуальные проблемы, начнем с общих. Первая - каждый стандарт имеет свой собственный формат кадра. Они показаны на рисунке 4-32. Для этих различий не было никаких технических оснований. Просто корпорации, поддерживавшие их разработки, - Xerox, GM, IBM - не захотели пойти на встречу друг другу и что-то изменить у себя. В результате при переходе из сети в сеть требуется реформатирование кадра, на что тратится время процессора, перевычисляется контрольная сумма. Ничего этого не потребовалось бы, если бы три комитета смогли договориться о едином формате.

Рисунок 4-32. Форматы кадров IEEE 802



Следующая проблема - разные сети могут работать с разной скоростью. Если передача идет из скоростной сети в медленную, то мост должен обладать достаточным буфером. Эта проблема может усугубляться непостоянством скорости передачи из-за коллизий, как, например, в 802.3. К тому же несколько сетей могут посылать трафик одной и той же сети, что опять приведет к перекосу скоростей. Другой тонкой, но важной проблемой является проблема моста как источника временной задержки, которая может влиять на тайм-аут на верхних уровнях. Предположим, что сетевой уровень над 802.4 пытается послать длинное сообщение в виде последовательности кадров. После отправки последнего кадра таймер устанавливается на ожидание уведомления о получении. Если сообщение проходит через мост с медленной 802.5, то есть опасность, что тайм-аут наступит прежде, чем последний кадр будет передан в медленную сеть. Сетевой уровень решит, что все сообщение утеряно, и начнет все сначала. После нескольких попыток сетевой уровень сообщит транспортному, что получатель отсутствует. Третьей, и наиболее серьезной проблемой является то, что все три стандарта имеют разную максимальную длину кадра. Для 802.3 на 10 Мбит/сек. это 1500 байт, для 802.4 - 8191 байт, для 802.5 максимальная длина ограничена временем удержания маркера. Последняя величина по умолчанию имеет значение 10 мсек., что соответствует 5000 байтам. Очевидная проблема возникает, когда длинный кадр надо доставить в сеть с короткими кадрами. Разбиение кадра на части не является задачей данного уровня. Это не значит, что таких протоколов не было создано, просто стандарты 802 не предусматривают надлежащих средств. Слишком длинные кадры просто сбрасываются.

Теперь рассмотрим все девять пар.

Рисунок 4-33. Проблемы, возникающие при построении мостов из 802.x в 802.y

ЛВС-адресат

ЛВС-отправитель	802.3 (CSMA/CD)	802.4 (Шина с маркером)	802.4 (Кольцо с маркером)
802.3		1, 4	1, 2, 4, 8
802.4	1, 5, 8, 9, 10	9	1, 2, 3, 8, 9, 10
802.5	1, 2, 5, 6, 7, 10	1, 2, 3, 6, 7	6, 7

Действия:

1. Изменение формата кадра и подсчет новой контрольной суммы
2. Обращение порядка бит
3. Копирование приоритета, вне зависимости от значимости
4. Порождение фиктивного приоритета
5. Сбрасывание приоритета
6. Фильтрование кольца
7. Установка битов A и C
8. Проверка на перегрузку (от быстрых ЛВС к медленным)
9. Проверка на задержку или невозможность получения уведомления о передаче маркера
10. Объявление тревоги в случае, если кадр слишком большой для сети назначения

Установленные параметры:

802.3:	1500-байтовые кадры	10 Мбит/сек. (не считая коллизий)
802.4:	8191-байтовые кадры	10 Мбит/сек.
802.5:	5000-байтовые кадры	4 Мбит/сек.

802.3 - 802.3

Здесь есть только одна опасность, что сеть, в которую передают, сильно перегружена и мост не успевает вставлять свои кадры. Есть опасность переполнения буфера у моста, в случае чего мост начнет сбрасывать кадры. Такая проблема потенциально есть всегда и мы о ней здесь более упоминать не будем. По отношению к двум другим стандартам все проще, так как мост обязательно получит маркер и свой временной слот на передачу кадров.

802.4 - 802.3

Здесь две проблемы. Первая - кадры из 802.4 содержат биты приоритета, а в кадрах 802.3 таких нет. В результате, если две 802.4 взаимодействуют через 802.3, то информация о приоритетах будет уничтожена.

Вторая проблема вызвана исключительно тем, что в кадре 802.4 есть бит временной передачи маркера получателю для уведомления. Что делать мосту, если ему поступит такой кадр? Послать подтверждение самому нельзя - получатель может быть неработоспособен. С другой стороны, если не дать подтверждения, то отправитель решит, что получатель неработоспособен, что может быть не так, и предпринять соответствующие меры. Похоже, что у этой проблемы нет решения.

802.5 - 802.3

Здесь мы имеем проблемы, аналогичные тем, что мы обсуждали выше. Например, есть биты A и C, которые используются для анализа информации о доставке и получении кадра. Как поступать мосту с такими кадрами? Если мост сам начнет имитировать за получателя значения этих разрядов, то очевидно, здесь могут возникать трудноисправимые ошибки. Ситуация выглядит так, что появление моста может менять семантику отдельных разрядов кадра, и как решить эту проблему - представить трудно.

802.3 - 802.4

Здесь основную трудность представляют биты приоритета. Что в них писать? Возможно, здесь стоит все кадры пускать с наивысшим приоритетом, так как они уже настрадались от задержек при передаче.

802.4 - 802.4

Единственная проблема, которая здесь существует, - это как поступать с временной передачей маркера? Мост может посылать такой кадр как можно быстрее, в надежде что ответ придет раньше, чем истечет тайм-аут. Можно посылать его с наивысшим приоритетом. При этом мост, так сказать, покривит душой, но это увеличит вероятность получения ответа до истечения тайм-аута.

802.5 - 802.4

Здесь проблему представляют биты A и C. Кроме этого, семантика приоритетов в этих сетях немного разная. Но выбора нет. Остается просто копировать биты приоритетов в надежде на хороший исход.

802.3 - 802.5

В этом случае надо генерировать биты приоритета. Других проблем здесь нет.

802.4 - 802.5

Здесь может возникнуть проблема слишком длинного кадра. Опять-таки здесь присутствует передача маркера.

802.5 - 802.5

Здесь надо только решить, что делать с битами A и C.

На рисунке 4-33 перечислены все эти проблемы.

Здесь мы рассмотрели соединение двух сетей IEEE 802 с помощью моста. Далее мы рассмотрим, как соединять несколько ЛВС-сетей между собой с помощью системы мостов и два основных подхода IEEE в конструкции таких мостов.

Сравнение мостов для 802

Оба вида мостов, как прозрачные, так и с маршрутизацией от источника, имеют как достоинства, так и недостатки. В ртаблице 4-38 они представлены в виде таблицы.

Таблица 4-38. Сравнение мостов для 802

Признак	Прозрачный	С маршрутизацией от источника
Ориентация	Без соединения	С соединением
Прозрачность	Полностью прозрачный	Непрозрачный
Настройка	Автоматическая	Вручную
Выбор маршрута	Частично оптимальный	Оптимальный
Способ локализации моста	«Обучение с запаздыванием»	«Поисковый кадр»
Сбои	Устраняются мостами	Устраняются хостами
Наибольшая сложность	В мостах	В хостах

Одно из основных различий между этими мостами - это различие между сетями, ориентированными на соединение, и не ориентированными на соединение. Прозрачные мосты не поддерживают концепции виртуального соединения и маршрутизируют каждый кадр независимо друг от друга. Мосты с маршрутизацией от источника, наоборот, определяют маршрут с помощью поискового кадра, а затем используют этот маршрут постоянно.

Прозрачные мосты полностью совместимы со всеми продуктами в стандартах 802. Это не так по отношению к мостам с маршрутизацией от источника. Все хосты в системе должны знать схему подключения мостов в сети. Любое изменение в сети, затрагивающее мосты, вызывает необходимость в перенастройке хостов.

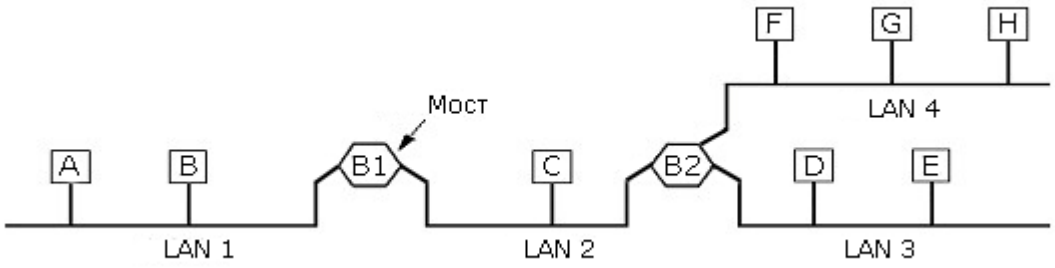
При использовании прозрачных мостов никаких специальных усилий по управлению сетью не требуется. Мосты автоматически поддерживают конфигурацию сети. При использовании мостов с маршрутизацией от источника требуется значительная ручная работа. Любая ошибка в адресах сетей или мостов может вызвать тяжелые последствия, обнаружить ее бывает очень трудно. При соединении двух функционирующих сетей через прозрачный мост ничего делать не надо, кроме как подключить его. При их соединении через мост, маршрутизирующий от источника, может потребоваться изменение номеров сетей и мост, дабы избежать дублирования.

Одно из основных преимуществ мостов с маршрутизацией от источника - они хотя бы теоретически могут строить оптимальные маршруты, в то время как прозрачные мосты ограничены в выборе маршрута деревом соединений. Мосты с маршрутизацией от источника также могут работать с мостами, включенными в параллель. Механизмы определения места доставки кадра также имеют свои достоинства и недостатки. В случае с прозрачными мостами недостаток тот, что надо ждать, когда придет кадр от машины получателя. В другом случае - имеем экспоненциальный рост числа поисковых кадров. Реакция на ошибки в сетях в обоих случаях разная. В первом случае все происходит автоматически, мосты сами слушают, что происходит в сетях. и реагируют должным образом. Самим хостам делать ничего не приходится. В случае мостов с маршрутизацией от источника ситуация противоположная: вся сложность исправления ошибок и отказов в сети ложится на хосты.

Прозрачные мосты (Мосты с соединяющими деревьями). Мосты с маршрутизацией от источника. Удаленные мосты.

Первым мостом 802 является прозрачный мост, или мост с деревом соединений. Основной заботой разработчиков этого моста было обеспечение его полной прозрачности. Они хотели создать устройство по стандарту IEEE, которое пользователь мог бы купить в магазине, подключить кабели своих многочисленных сетей и работать. Подключение в сеть этого устройства не должно было бы требовать каких-либо изменений в оборудовании, программном обеспечении, переинсталляции сетей, загрузки каких-либо таблиц и т.п. Просто купил, принес, включил, и все работает. Как это ни удивительно, но они почти достигли своей цели. Прозрачный мост функционирует в режиме общедоступности, т.е. ему доступны все пакеты от всех сетей, подключенных к нему. Рассмотрим пример на рисунке 4-34. Кадр для A, поступивший из сети LAN 1, должен быть сброшен, т.к. он уже в нужной сети, а вот кадр из LAN 1 для C или F надо передать в нужную сеть.

Рисунок 4-34. Конфигурация из четырех сетей и двух мостов



По каждому поступающему кадру мост должен принять следующее решение: надо ли его передавать дальше или сбросить; если передавать дальше, то в какую сеть? Для этого каждый мост должен иметь таблицу, где каждой станции сопоставлен номер сети, в которой она находится. Эта таблица, как правило, имеет огромные размеры и организована как таблица перемешивания.

Когда мосты включаются первый раз, все таблицы пусты. Для заполнения своей таблицы каждый мост использует следующий алгоритм:

- Каждый кадр с неизвестным мосту адресом доставки рассылается во все сети, подключенные к данному мосту, кроме той, из которой поступил этот кадр.
- По реакции из каждой сети на этот кадр мост определяет, в какой конкретно сети находится адрес доставки и фиксирует эту информацию в таблице (как это происходит, мы рассмотрим позже).

Далее кадры с таким же адресом доставки будут посылаться только в сеть, определенную этим алгоритмом, который называется обучение с запаздыванием.

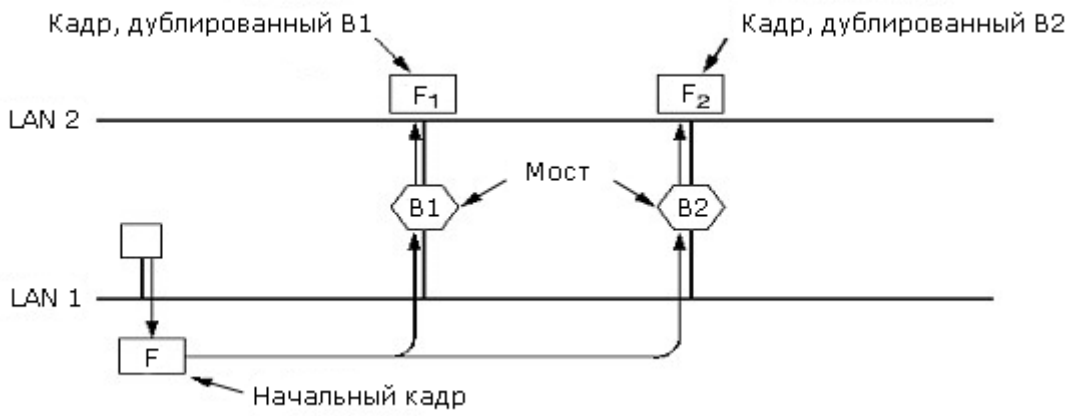
Топология сети может изменяться динамически. Машины и мосты могут включаться в сеть и выключаться из нее. Поэтому для каждого элемента таблицы указывается время, когда от этой машины или моста поступал кадр.

Периодически таблица просматривается, и для всех ее элементов, у которых время последнего поступления кадра отличается от текущего более чем на несколько минут, запускается процедура поиска его в сети. Поэтому все изменения в сети отслеживаются динамически. Если какую-то машину выключат из одной сети, перенесут и включат в другой, описанный алгоритм отметит это изменение через несколько минут. Итак, каждый раз, когда поступает кадр, мост выполняет следующие действия:

1. Если адрес отправителя и адрес получателя один и тот же, кадр сбрасывается.
2. Если адрес отправителя и адрес получателя разные, то кадр направляется в надлежащую сеть.
3. Если нет информации в таблице, куда направлять кадр, его посылают во все доступные сети.

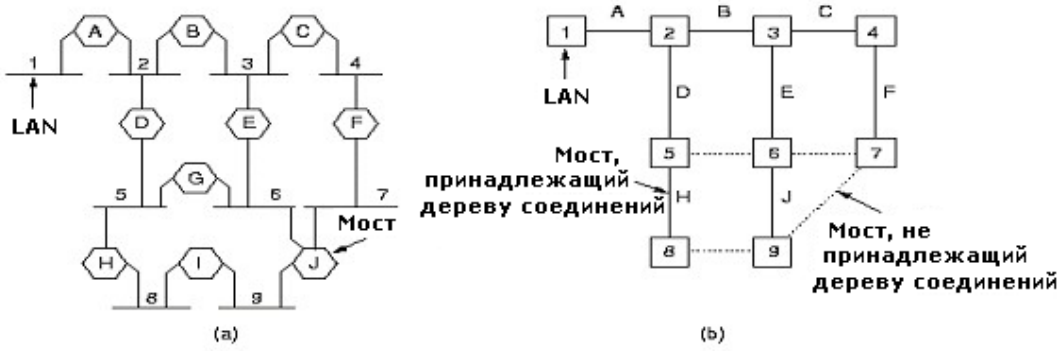
В некоторых случаях для большей надежности две сети соединяют двумя мостами, как это показано на рисунке 4-35. Однако у такой конфигурации есть одна опасность. Пусть в сети 1 был выпущен кадр F. Этот кадр будет дублирован во все сети и мостом B1, и мостом B2. Пусть мост B1 породил F1, а B2 - F2. Мост B1 увидит F2 с неизвестным адресом доставки и дублирует его в сеть 1 как F3. То же самое сделает B2 с кадром F1 в виде кадра F4. Этот цикл будет длиться до бесконечности.

Рисунок 4-35. Два параллельных прозрачных моста



Решение этой проблемы состоит в том, чтобы мосты во взаимодействии друг с другом накладывали на фактическую структуру соединений дерево соединений и делали пересылку так, чтобы избегать таких мнимых циклов. На рисунке 4-36 показана сеть с 10 мостами. Граф соединений на рисунке 4-36 (a) можно сократить до дерева соединений на рисунке 4-36 (b). В этом дереве для каждой сети есть только один путь.

Рисунок 4-36. Использование дерева соединений в конструкциях с мостами



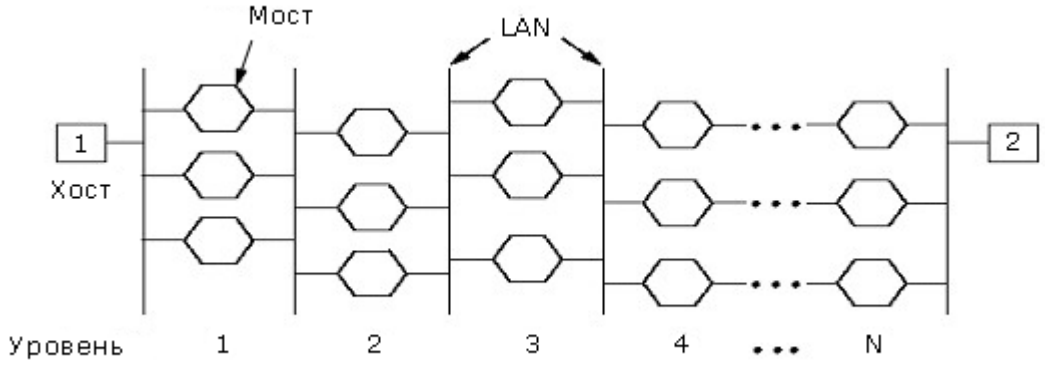
Как строится дерево соединений? Прежде всего, надо выбрать один мост из всех в качестве корня дерева. Это делается с помощью уникальных адресов, которые присваиваются мостам при изготовлении, подобно Ethernet-адресам. Из всех мостов корневым выбирается мост с наименьшим номером. Затем для полученного корня строится дерево кратчайших путей, соединяющих корень с каждым мостом и сетью. Полученное дерево и есть дерево соединений. В результате алгоритм строит единственный маршрут от корня в любую сеть. Хотя дерево соединений охватывает все сети, в нем представлены не все мосты.

Мосты с маршрутизацией от источника

Прозрачные мосты хороши тем, что их достаточно только подключить, и все работает. Однако они никак не учитывают оптимальное распределение пропускной способности и не могут этого делать. Это привело к появлению иной схемы работы мостов - маршрутизации от источника. Будем предполагать, что отправитель знает, находится получатель в его локальной сети или нет. Если получатель не в его локальной сети, то отправитель устанавливает старший разряд в адресе получателя в 1. Кроме этого, в заголовке кадра

указывается точный маршрут, по которому будут следовать кадр. Этот маршрут представляет собой чередование 12-разрядного адреса сети и 4-разрядного адреса моста. (См. рисунок 4-34.) Мост с маршрутизацией от источника ловит только те кадры, у которых старший разряд в адресе получателя равен 1. Для каждого такого кадра просматривается описание маршрута и определяется, в какую сеть отправлять этот кадр. Номер сети указан после номера моста в описании маршрута. Этот алгоритм предполагает, что каждый отправитель знает или может определить наилучший маршрут. Основная идея алгоритма поиска такого маршрута состоит в следующем. Если маршрут к получателю не известен, то отправитель посылает так называемый поисковый кадр. Этот поисковый кадр рассылается всеми мостами по всем сетям. Когда поисковый кадр возвращается обратно, каждый мост оставляет в нем информацию о себе. Таким образом, отправитель, получив ответы на свой поисковый кадр, может выбрать наилучший маршрут среди всех имеющихся. Этот алгоритм действительно позволяет найти наилучший маршрут, но он имеет один серьезный недостаток - экспоненциальный рост числа поисковых кадров. Это видно на рисунке 4-37. К тому моменту, как поисковый кадр достигнет уровня N, число поисковых кадров в сети будет порядка 3N-1. Нечто подобное происходит и в сетях с прозрачными мостами, но там этот рост происходит только вдоль дерева связей.

Рисунок 4-37. Сети, соединенные тройными мостами

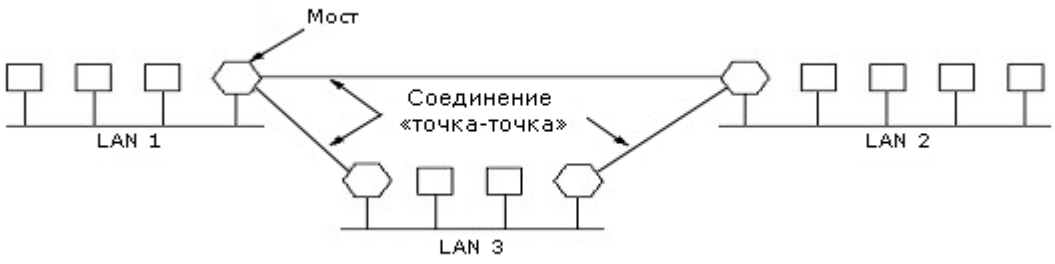


Обнаружив наилучший путь, каждый хост в сети хранит его. Естественно, это накладывает определенные требования на хосты в сети, что делает использование этого подхода не столь прозрачным, как в первом случае.

Удаленные мосты

Как мы уже говорили, основная цель использования мостов – соединение отдельных ЛВС между собой. Это можно сделать, сопоставив каждой ЛВС мост и соединив мосты между собой каналом «точка-точка». Пример такого соединения показан на рисунке 4-39. Здесь каждый канал точка-точка можно рассматривать как ЛВС без абонентских машин (кто сказал, что сеть обязана всегда их иметь). Тогда у нас есть шесть ЛВС, соединенных через четыре моста.

Рисунок 4-39. Удаленные мосты, соединяющие ЛВС



Для соединений точка-точка можно использовать разные протоколы. Например, можно использовать любой протокол точка-точка и в его кадрах целиком размещать кадры MAC-подуровня. Этот прием хорошо

работает в случае идентичных сетей. При этом возникает только одна трудность – маршрутизация кадров в нужную сеть. Другая возможность – отрезать заголовки MAC-подуровня и на их место вставить заголовки соответствующего канального уровня. Новый MAC-заголовок будет сгенерирован на стороне моста-получателя. Здесь трудности возникают при работе с полем контрольной суммы. Либо надо ее каждый раз перевычислять, либо мы потеряем возможность контролировать ошибки при передаче.

Протоколы для высокоскоростных локальных сетей.

В этом разделе мы рассмотрим стандарты для построения высокоскоростных сетей, появившиеся в 90-е годы. Эти стандарты предполагают использование оптоволоконных линий связи, скорость не ниже 100 Мбит/сек. и действуют на большие расстояния, чем рассмотренные до сих пор стандарты IEEE 802.

Основы технологии FDDI

Технология FDDI является высокопроизводительным развитием технологии Token Ring, позволяющей работать на скоростях не ниже 100 Мбит/сек., расстоянии до 200 км и до 1000 рабочих станций. Разработчики технологии FDDI ставили перед собой в качестве наиболее приоритетных следующие цели:

- Повысить скорость передачи данных до 100 Мбит/сек.
- Повысить отказоустойчивость сети за счет стандартных процедур восстановления после отказов различного рода - повреждения кабеля, некорректной работы узла, концентратора, возникновения высокого уровня помех на линии и т.п.
- Максимально эффективно использовать потенциальную пропускную способность сети как для асинхронного, так и для синхронного трафиков.

Сеть FDDI строится на основе двух оптоволоконных колец. По одному трафик направлен по часовой стрелке, по другому – против. В случае выхода из строя одного из колец его трафик может быть запущен через второе кольцо. Если оба кольца окажутся поврежденными в одном и том же месте, то они могут быть объединены в одно кольцо, как показано на рисунке 4-41. Такая реконфигурация сети происходит силами концентраторов и/или сетевых адаптеров FDDI. Использование двух колец - это основной способ повышения отказоустойчивости в сети FDDI, и узлы, которые хотят им воспользоваться, должны быть подключены к обоим кольцам. В нормальном режиме работы сети данные проходят через все узлы и все участки кабеля первичного (Primary) кольца, поэтому этот режим назван режимом Thru - «сквозным» или «транзитным». Вторичное кольцо (Secondary) в этом режиме не используется. При образовании общего кольца из двух колец передатчики станций по-прежнему остаются подключенными к приемникам соседних станций, что позволяет правильно передавать и принимать информацию соседними станциями. В стандартах FDDI отводится много внимания различным процедурам, которые позволяют определить наличие отказа в сети, а затем произвести необходимую реконфигурацию. Сеть FDDI может полностью восстанавливать свою работоспособность в случае единичных отказов ее элементов. При множественных отказах сеть распадается на несколько не связанных сетей.

Рисунок 4-40. Кольцо FDDI в качестве магистрали для ЛВС и абонентских машин

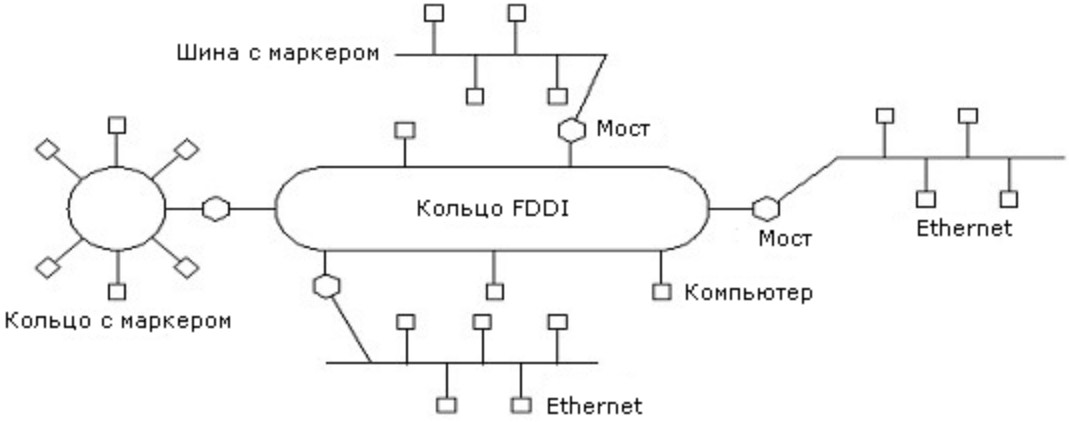
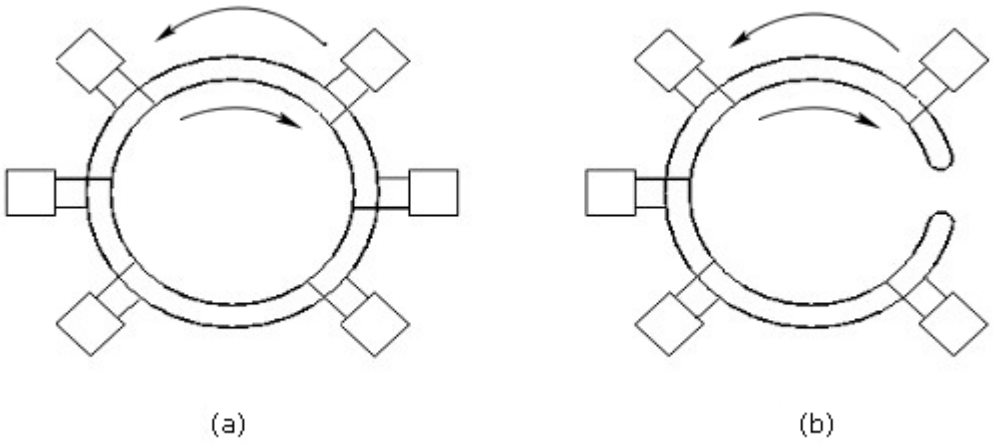


Рисунок 4-41. Объединение двух колец в одно



Кольца в сетях FDDI рассматриваются как общая, разделяемая среда передачи данных, поэтому для нее определен специальный метод доступа. Этот метод очень близок к методу доступа сетей Token Ring и также называется методом кольца с маркером. Станция может начать передачу своих собственных кадров данных только в том случае, если она получила от предыдущей станции специальный кадр - маркер доступа. После этого она может передавать свои кадры в течение времени, называемого временем удержания маркера, - Token Holding Time (THT). После истечения времени THT станция обязана завершить передачу своего очередного кадра и передать маркер доступа следующей станции. Если же в момент получения маркера у станции нет кадров для передачи по сети, то она немедленно передает маркер следующей станции. Как и в ранее рассмотренных способах доступа с маркером, в сети FDDI у каждой станции есть предшествующий сосед (upstream neighbor) и последующий сосед (downstream neighbor), определяемые ее физическими связями и направлением передачи информации.

Каждая станция в сети постоянно принимает передаваемые ей предшествующим соседом кадры и анализирует их адрес назначения. Если адрес назначения не совпадает с ее собственным, то она транслирует кадр своему последующему соседу. Напомним, что, если станция получила маркер и передает свои собственные кадры, то на протяжении этого периода времени она не транслирует приходящие кадры, а удаляет их из сети. Если же адрес кадра совпадает с адресом станции, то она копирует этот кадр в свой внутренний буфер, проверяет его корректность (с помощью контрольной суммы), передает его поле данных для последующей обработки протоколу, лежащего выше FDDI-уровня (например, IP), а затем передает исходный кадр по сети последующей станции.

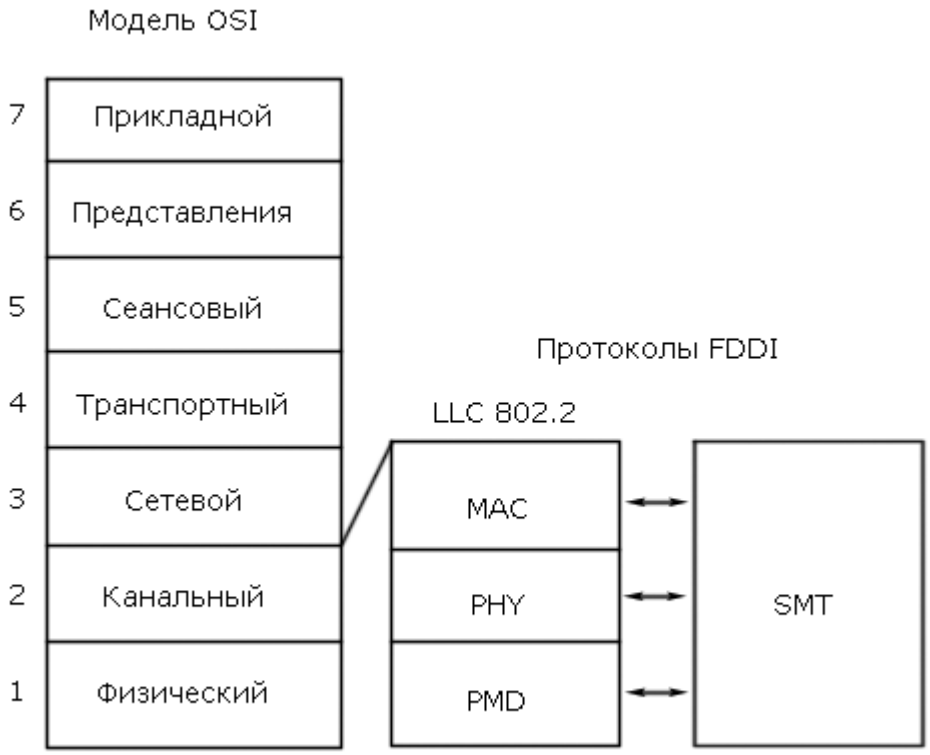
Как и в рассмотренном ранее протоколе кольца с маркером, в передаваемом в сеть кадре станция назначения отмечает три признака: распознавания адреса, копирования кадра и отсутствия или наличия в нем ошибок.

Станция, являющаяся источником кадра для сети, проверяет признаки кадра, дошел ли он до станции назначения, и не был ли при этом поврежден. Процесс восстановления информационных кадров не входит в обязанности протокола FDDI, этим должны заниматься протоколы более высоких уровней.

4.5.1.1. Структура протоколов технологии FDDI

На рисунке 4-42 приведена структура протоколов технологии FDDI в сравнении с семиуровневой моделью OSI. FDDI определяет протокол физического уровня и протокол подуровня доступа к среде (MAC) канального уровня. Как и многие другие технологии локальных сетей, технология FDDI использует протокол 802.2 подуровня управления каналом данных (LLC), определенный в стандартах IEEE 802.2 и ISO 8802.2. В FDDI используется первый тип процедур LLC, при котором узлы работают в дейтаграммном режиме - без установления соединений и без восстановления потерянных или поврежденных кадров.

Рисунок 4-42. Структура протоколов технологии FDDI



Физический уровень разделен на два подуровня: независимый от среды подуровень PHY (Physical) и зависящий от среды подуровень PMD (Physical Media Dependent). Работу всех уровней контролирует протокол управления станцией SMT (Station Management). Здесь видна аналогия с организацией физического уровня в СПД ATM. Уровень PMD обеспечивает необходимые средства для передачи данных от одной станции к другой по оптоволокну. Уровень PHY выполняет кодирование и декодирование данных, циркулирующих между MAC-уровнем и уровнем PMD, а также обеспечивает тактирование информационных сигналов. Уровень MAC ответственен за управление доступом к сети, а также за прием и обработку кадров данныхУровень SMT выполняет все функции по управлению и мониторингу всех остальных уровней стека протоколов FDDI. В управлении кольцом принимает участие каждый узел сети FDDI. Поэтому все узлы обмениваются специальными кадрами SMT для управления сетью. В спецификации SMT определено следующее:

- Алгоритмы обнаружения ошибок и восстановления после сбоев
- Правила мониторинга работы кольца и станций
- Управление кольцом
- Процедуры инициализации кольца

Отказоустойчивость сетей FDDI обеспечивается за счет того, что уровень SMT управляет другими уровнями: с помощью уровня PHY устраняются отказы сети по физическим причинам, например, из-за обрыва кабеля, а с помощью уровня MAC - логические отказы сети, например, потеря нужного внутреннего пути передачи маркера и кадров данных между портами концентратора.

Типы узлов и правила их соединения в сеть

Все станции в сети FDDI делятся на несколько типов по следующим признакам:

- конечные станции или концентраторы
- по способу присоединения к первичному и вторичному кольцам
- по количеству MAC-узлов и, соответственно, MAC-адресов у одной станции

Как и в стандарте IEEE 802.5, для того чтобы иметь возможность передавать собственные данные в кольцо (а не просто ретранслировать данные соседних станций), станция должна иметь в своем составе хотя бы один MAC-узел, который имеет свой уникальный MAC-адрес. Станции могут не иметь ни одного MAC-узла, и, значит, участвовать только в ретрансляции чужих кадров. Но обычно все станции сети FDDI, даже концентраторы, имеют хотя бы один MAC. Концентраторы используют MAC-узел для захвата и генерации служебных кадров, например, кадров инициализации кольца, кадров поиска неисправности в кольце и т.п.

4.5.1.7. Инициализация кольца

Процедура инициализации кольца, известная под названием Claim Token, выполняется для того, чтобы все станции кольца убедились в его потенциальной работоспособности. Кроме этого, в ходе этой процедуры они должны прийти к соглашению о значении параметра T_Opr - максимально допустимому времени оборота маркера по кольцу, на основании которого все станции вычисляют время удержания маркера THT. Процедура Claim Token выполняется в нескольких ситуациях:

- при включении новой станции в кольцо и при выходе станции из кольца
- при обнаружении какой-либо станцией факта утери маркера (маркер считается утерянным, если станция не наблюдает его в течение двух периодов времени максимального оборота маркера T_Opr)
- при обнаружении длительного отсутствия активности в кольце, когда станция в течение определенного времени не наблюдает проходящих через нее кадров данных
- по команде от блока управления станцией SMT

Для выполнения процедуры инициализации каждая станция сети должна знать о своих требованиях к максимальному времени оборота маркера по кольцу. Эти требования содержатся в параметре, называемом «требуемое время оборота маркера» - TTRT (Target Token Rotation Time). Параметр TTRT отражает степень потребности станции в пропускной способности кольца - чем меньше время TTRT, тем чаще станция желает получать маркер для передачи своих кадров. Процедура инициализации позволяет станциям узнавать о требованиях к времени оборота маркера других станций и выбрать минимальное время в качестве общего параметра T_Opr, на основании которого в дальнейшем будет распределяться пропускная способность кольца. Параметр TTRT должен находиться в пределах от 4 мсек. до 165 мсек. и может изменяться администратором сети.

Если какая-либо станция решает начать процесс инициализации кольца по своей инициативе, то она формирует кадр Claim Token со своим значением требуемого времени оборота маркера. Захвата маркера для этого не требуется. Любая другая станция, получив кадр Claim Token, начинает выполнять процедуру Claim Token. Для выполнения процедуры инициализации каждая станция поддерживает таймер текущего времени оборота маркера TRT (Token Rotation Timer), который используется также и в дальнейшем при работе кольца в нормальном режиме. Таймер TRT запускается каждой станцией при обнаружении начала процедуры Claim Token. В качестве предельного значения таймера выбирается максимально допустимое время оборота маркера, то есть 165 мсек. Истечение таймера TRT до завершения процедуры означает ее неудачное окончание - кольцо не удалось инициализировать. В случае неудачи процесса Claim Token запускается процедуры, с помощью которых станции кольца пытаются выявить некорректно работающую часть кольца и отключить ее от сети. Схематично работа процедуры Clime Token выглядит следующим образом. Каждая

станция генерирует кадр Clime со своим значением T_Req, равным значению ее параметра TTRT. При этом она устанавливает значение T_Opr, равное значению TTRT. Станция, приняв кадр Claim от предыдущей станции, обязана сравнить значение T_Req, указанное в кадре со своим предложенным значением TTRT. Если другая станция просит установить время оборота маркера меньше, чем данная (то есть, T_Req < TTRT), то данная станция перестает генерировать собственные кадры Claim и начинает повторять чужие кадры Claim, так как видит, что в кольце есть более требовательные станции. Одновременно станция фиксирует в своей переменной T_Opr минимальное значение T_Req, которое ей встретилось в чужих кадрах Claim. Если же пришедший кадр имеет значение T_Req больше, чем собственное значение TTRT, то он удаляется из кольца. Процесс Claim завершается для станции в том случае, если она получает кадр Claim со своим адресом назначения. Это означает, что данная станция является победителем состязательного процесса и ее значение TTRT оказалось минимальным. При равных значениях параметра TTRT преимущество отдается станции с большим значением MAC-адреса.

После того как станция обнаруживает, что она оказалась победителем процесса Claim Token, она должна сформировать маркер и отправить его по кольцу. Первый оборот маркера - служебный, так как за время этого оборота станции кольца узнают, что процесс Claim Token успешно завершился. При этом они устанавливают признак Ring_Operational в состояние True, означающее начало нормальной работы кольца. При следующем проходе маркера его можно будет использовать для захвата и передачи кадров данных. Если же у какой-либо станции во время выполнения процедур инициализации таймер TRT истек, а маркер так и не появился на входе станции, то станция начинает процесс Beacon. После нормального завершения процесса инициализации у всех станций кольца устанавливается одинаковое значение переменной T_Opr.

4.5.2. Fast Ethernet

Термином Fast Ethernet называют набор спецификаций, разработанных комитетом IEEE 802.3, чтобы обеспечить недорогой, Ethernet-совместимый стандарт, способный обеспечить работу ЛВС на скорости 100 Мбит/сек. Отличия Fast Ethernet от Ethernet сосредоточены на физическом уровне. Отметим главные особенности эволюционного развития от сетей Ethernet к сетям Fast Ethernet стандарта IEEE 802.3u:

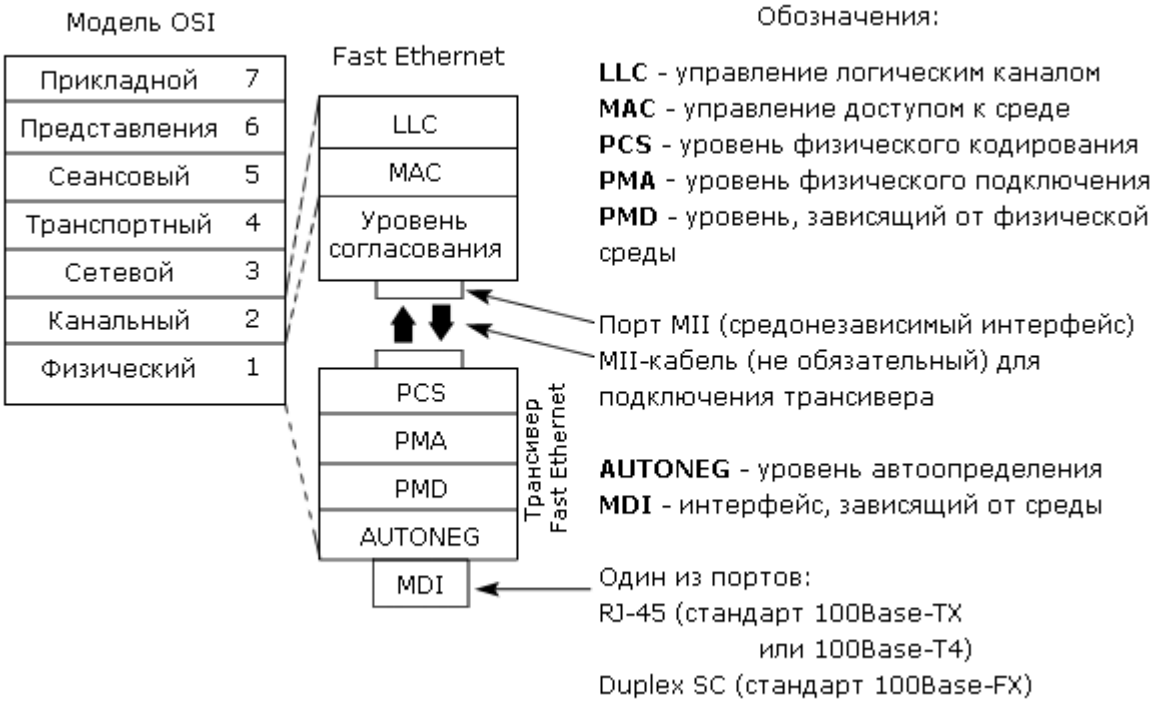
- десятикратное увеличение пропускной способности сегментов сети
- сохранение метода случайного доступа CSMA/CD, принятого в Ethernet
- сохранение формата кадра, принятого в Ethernet
- поддержка традиционных сред передачи данных - витой пары и волоконно-оптического кабеля

Кроме указанных свойств, важной функцией этого стандарта является поддержка двух скоростей передачи 10/100 Мбит/сек. и автоматический выбор одной из них, встраиваемая в сетевые карты и коммутаторы Fast Ethernet. Все это позволяет осуществлять плавный переход от сетей Ethernet к более скоростным сетям Fast Ethernet, обеспечивая выгодную преемственность по сравнению с другими технологиями. Еще один дополнительный фактор - низкая стоимость оборудования Fast Ethernet.

4.5.2.1. Архитектура стандарта Fast Ethernet

На рисунке 4-46 показана структура уровней Fast Ethernet. Более сложная структура физического уровня технологии Fast Ethernet вызвана тем, что в ней используются три варианта кабельных систем - оптоволокно, двухпарная витая пара категории 5 и четырехпарная витая пара категории 3.

Рисунок 4-46. Структура уровней стандарта Fast Ethernet, MII-интерфейс и трансивер Fast Ethernet



Интерфейс MII (medium independent interface) в стандарте Fast Ethernet является аналогом интерфейса AUI в стандарте Ethernet. MII-интерфейс обеспечивает связь между подуровнями согласования и физического кодирования. Основное его назначение - упростить использование разных типов среды. MII-интерфейс предполагает дальнейшее подключение трансивера Fast Ethernet. Для связи используется 40-контактный разъем. Максимальное расстояние по MII-интерфейсному кабелю не должно превышать 0,5 м.

Основные категории устройств, применяемых в Fast Ethernet, такие же, как и в Ethernet: трансиверы, конвертеры, сетевые карты (для установки на рабочие станции/файл-серверы), повторители, коммутаторы.

Трансивер - это (по аналогии с трансивером Ethernet) двухпортовое устройство, охватывающее подуровни PCS, PMA, PMD и AUTONEG, и имеющее с одной стороны MII-интерфейс, с другой - один из средозависимых физических интерфейсов (100Base-FX, 100Base-TX или 100Base-T4). Трансиверы используются сравнительно редко, как и редко используются сетевые карты, повторители и коммутаторы с интерфейсом MII. Сетевая карта.Наиболее широкое распространение сегодня получили сетевые карты с интерфейсом 100Base-TX на шину PCI. Необязательными, но крайне желательными функциями порта RJ-45 является автоконфигурирование 100/10 Мбит/сек. и поддержка дуплексного режима. Большинство современных выпускаемых карт поддерживают эти функции. Конвертер (media converter) - это двухпортовое устройство, оба порта которого представляют средозависимые интерфейсы. Конвертеры, в отличие от повторителей, могут работать в дуплексном режиме, за исключением случая, когда имеется порт 100Base-T4. Распространены конвертеры 100Base-TX/100Base-FX. Коммутатор - одно из наиболее важных устройств при построении корпоративных сетей. Большинство современных коммутаторов Fast Ethernet поддерживает автоконфигурирование 100/10 Мбит/с по портам RJ-45 и могут обеспечивать дуплексный канал связи по всем портам (за исключением 100Base-T4). Коммутаторы могут иметь специальные дополнительные слоты для установления uplink-модуля. В качестве интерфейсов у таких модулей могут выступать оптические порты типа Fast Ethernet 100Base-FX, FDDI , ATM (155 Мбит/сек.), Gigabit Ethernet и др.

4.5.3. Gigabit Ethernet

Интерес к технологиям для локальных сетей с гигабитными скоростями повысился в связи с двумя обстоятельствами - во-первых, успехом сравнительно недорогих (по сравнению с FDDI) технологий Fast Ethernet, во-вторых, со слишком большими трудностями, испытываемыми технологией ATM на пути к конечному пользователю. В марте 1996 года комитет IEEE 802.3 одобрил проект стандартизации Gigabit Ethernet 802.3z. В мае 1996 года 11 компаний (3Com Corp., Bay Networks Inc., Cisco Systems Inc., Compaq Computer Corp., Granite Systems Inc., Intel Corporation, LSI Logic, Packet Engines Inc.,

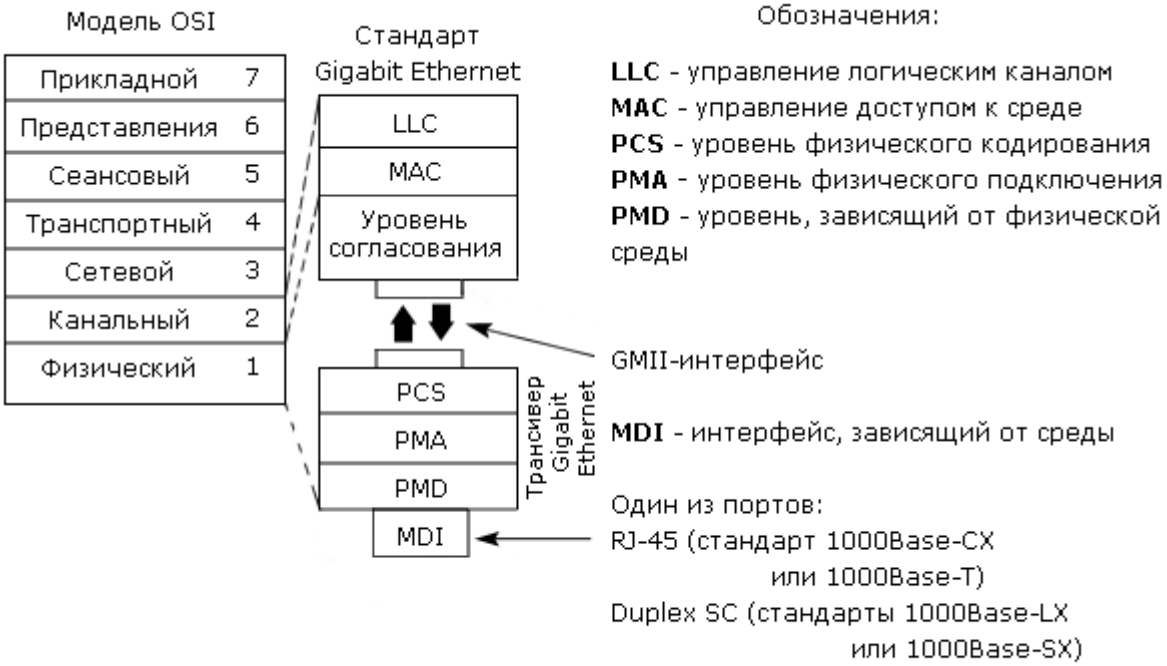
Sun Microsystems Computer Company, UB Networks и VLSI Technology) организовали Gigabit Ethernet Alliance. Альянс объединил усилия большого числа ведущих производителей сетевого оборудования на пути выработки единого стандарта и выпуска совместимых продуктов Gigabit Ethernet и преследовал следующие цели:

- поддержка расширения технологий Ethernet и Fast Ethernet в ответ на потребность в более высокой скорости передачи
- разработка технических предложений с целью включения в стандарт
- выработка процедур и методов тестирования продуктов от различных поставщиков

Соответствующие спецификации регламентируют использование одномодового, многомодового волокна, а также витой пары UTP cat.5 на коротких расстояниях (до 25 м). Стандартизация системы передачи Gigabit Ethernet по неэкранированной витой паре на расстояния до 100 м требовала разработки специального помехоустойчивого кода для чего был создан отдельный подкомитет P802.3ab. 28 июня 1999 г. был принят соответствующий стандарт.

Архитектура стандарта Gigabit Ethernet

На рисунке 4-50 показана структура уровней Gigabit Ethernet. Как и в стандарте Fast Ethernet, в Gigabit Ethernet не существует универсальной схемы кодирования сигнала, которая была бы идеальной для всех физических интерфейсов - так, для стандартов 1000Base-LX/SX/CX используется кодирование 8B/10B, а для стандарта 1000Base-T - специальный расширенный линейный код TX/T2. Функцию кодирования выполняет подуровень кодирования PCS, размещенный ниже средонезависимого интерфейса GMII. Рисунок 4-50. Структура уровней стандарта Gigabit Ethernet, GII-интерфейс и трансивер Gigabit Ethernet



4.5.4. Fibre Channel

Известно, что производительность микропроцессоров рабочих станций удваивается каждые 18 месяцев. Растет объем их памяти. Растет сложность приложений, особенно в части графики, видео- и аудиосредств. Все это предъявляет растущие требования к скорости как ввода/вывода данных процессора, так и к скорости сетевого взаимодействия приложений. Канал ввода/вывода (I/O-канал) обеспечивает взаимодействие типа «точка-точка» на коротких расстояниях. Как правило, логика управления им не очень сложна. Канал считается очень надежным и обеспечивает передачу данных между процессором и периферийным устройством. Сетевое взаимодействие предполагает передачу данных на большие расстояния, чем в случае

канала ввода/вывода, форматы данных здесь могут быть весьма замысловатые. Это соединение использует различное программное обеспечение, реализующее надлежащие протоколы передачи, разные типы передач данных и т.д. Управление передачей и взаимодействием у этих соединений достаточно сложное. Fibre Channel сочетает в себе преимущества канальных и сетевых технологий. Он призван объединить в себе простоту и скорость I/O-канала с гибкостью и возможностями установления сетевых соединений на основе протоколов. Такое сочетание позволит разработчикам систем объединять традиционные подключения периферии, сетевые методы передачи данных и управления соединениями, методы соединения процессоров, используемые при создании кластеров, в единые мультипротокольные интерфейсы. Fibre Channel обеспечивает шесть независимых классов услуг (каждый класс представляет определенную стратегию обмена информацией), которые облегчают решение широкого диапазона прикладных задач:

Класс 1	Соединение с коммутацией каналов по схеме точка-точка (end-to-end) между портами типа n_port. Класс удобен для аудио- и видеоприложений, например, видеоконференций. После установления соединения используется вся доступная полоса пропускания канала. При этом гарантируется, что кадры будут получены в том же порядке, в каком они были отправлены.
Класс 2	Обмен без установления соединения с коммутацией пакетов, гарантирующий доставку данных. Так как соединение не устанавливается, порт может взаимодействовать одновременно с любым числом портов типа n_port, получая и передавая кадры. Здесь не гарантируется, что кадры будут доставлены в том же порядке, в каком были переданы (за исключением случаев соединения «точка-точка» или «кольцо с арбитражем»). В этом классе допустимы схемы управления потоком «буфер-буфер» и «точка-точка». Класс характерен для локальных сетей, где время доставки данных не является критическим.
Класс 3	Обмен дейтаграммами без установления соединения и без гарантии доставки. Схема управления потоком - «буфер-буфер». Применяется для каналов SCSI.
Класс 4	Обеспечивает выделение определенной доли пропускной способности канала с заданным значением качества обслуживания (QoS). Работает только с топологией структура (fabric), где соединяются два порта типа n_port. При этом формируются два виртуальных соединения, обслуживающих встречные потоки данных. Пропускная способность этих соединения может быть разной. Как и в классе 1, здесь гарантируется порядок доставки кадров. Допускается одновременное соединение более чем с одним портом типа n_port. Используется схема управления потоком «буфер-буфер».
Класс 5	Регламентирующие документы находятся в процессе подготовки.
Класс 6	Предусматривает групповое обслуживание в рамках топологии типа структура (fabric).

Сетевой уровень: проблемы построения сетевого уровня (Сервис, внутренняя организация сетевого уровня). Алгоритмы маршрутизации (принцип оптимальности, маршрутизация по наикратчайшему пути, маршрутизация лавиной, маршрутизация с анализом потока).

Основное назначение сетевого уровня состоит в получении пакетов от всех источников и передача их по назначению. Передача по назначению может состоять из нескольких этапов, потребовать нескольких маршрутизаторов. Это, согласитесь, более сложная задача, чем у канального уровня - передать кадр с одного конца провода на другой. Для реализации своего назначения сетевой уровень должен знать топологию

транспортной подсети и выбрать подходящий путь в ней. Выбирая маршрут, он должен позаботиться, чтобы этот маршрут не привел к перегрузкам некоторых линий и маршрутизаторов. Наконец, если источник и получатель принадлежат разным сетям, то задача сетевого уровня, принимая во внимание различия между этими сетями, обеспечить корректную передачу данных из одной сети в другую.

Сервис сетевого уровня разрабатывался в следующих целях:

- Сервис должен быть независимым от технологии передачи, используемой в среде передачи данных.
- Транспортный уровень должен быть независим от числа узлов, типа и топологии транспортной подсети.
- Адрес на сетевом уровне, доступный на транспортном уровне, должен иметь унифицированную форму по всей сети.

Ориентация на соединение

Представители лагеря Internet считают, что задача сетевого уровня - передвигать биты туда-сюда и ничего более. С их точки зрения транспортная среда ненадежна по определению, вне зависимости от того, как она построена. Поэтому хосты должны восполнять изъяны транспортной среды, т.е. контролировать ошибки и управлять потоками. Отсюда следует, что сервис на сетевом уровне не должен быть ориентирован на соединения, с примитивами типа SEND_PACKET, RECEIVE_PACKET. Никакой проверки упорядоченности пакетов, управления потоком, перегрузками здесь не должно быть. Это все должны делать хосты. Каждый пакет должен нести полный адрес назначения, поскольку пакеты отправляются абсолютно независимо. Представители другого лагеря - телефонные компании, основываясь на своем столетнем опыте эксплуатации телефонных сетей, считают, что сетевой уровень должен быть надежным и ориентированным на соединения. С их точки зрения соединения должны обладать следующими свойствами:

- Прежде чем передача начнется, передающая сторона должна установить соединение с равнозначной сущностью на принимающей стороне. При этом ей должен быть сообщен специальный идентификатор, который используется в течение всей передачи.
- Когда соединение установлено, два процесса начинают переговоры о параметрах, качестве и стоимости предоставляемого сервиса.
- Передача происходит в двух направлениях, а пакеты посылаются в определенном порядке.
- Управление потоком предоставляется автоматически, чтобы избежать переполнения на противоположной стороне.

Сервис, ориентированный на соединение, предполагает, что эта сложность приходится на сетевой уровень, т.е. на транспортную среду. Сервис без соединений - на транспортный уровень, а стало быть - на хост. Защитники сервиса без соединения говорят, что стоимость вычислительных средств падает, а их мощность растет, так что нет причин не нагрузить хост. В то же время подсеть - это всеобщая инвестиция, и часто модернизировать подсеть вряд ли будет возможно. Так что она должна оставаться неизменной как можно дольше. Кроме этого, для многих приложений скорость доставки важнее, чем ее аккуратность. Сторонники сервиса, ориентированного на соединение, считают, что большинство пользователей не хотят гонять сложные транспортные протоколы на своих машинах. То, что им надо, так это надежный сервис, а таковой могут предоставить соединения на сетевом уровне. Более того, многие приложения, такие как передача звука и изображения в реальном масштабе времени, легче связывать с соединениями на сетевом уровне, чем с сетевым уровнем без соединений. Так каким же должен быть сетевой уровень - надежным, ориентированным на соединения, или ненадежным без соединений? Два ответа на этот вопрос предоставляют Internet и ATM. В Internet сетевой уровень действует без соединений и предполагается ненадежным. В ATM - с соединениями и надежный.

Естественный вопрос: как Internet работает над ATM? Сначала на уровне ATM устанавливается соединение между источником и получателем, а потом над этим соединением работает TCP/IP, как это показано на рисунке 5-1. Однако здесь очень много избыточности и ненужного дублирования. Так, например, ATM-уровень гарантирует, что пакеты доставляются точно в том порядке, в каком они отправлялись источником,

тем не менее, на уровне TCP происходит проверка последовательности пакетов и переупорядочение в соответствии с RFC 1577.

Рисунок 5-1. Работа TCP/IP над ATM

Внутренняя организация сетевого уровня

С точки зрения внутренней организации сетевой уровень делится на ориентированный на соединения и без соединений. В первом случае соединение называют виртуальным каналом, по аналогии с физическим каналом в телефонных сетях. Во втором случае о пакетах говорят как о дейтаграммах, по аналогии с телеграммами.

Идея виртуального канала – избежать маршрутизации для каждого пакета. Маршрут устанавливается один раз при установлении виртуального канала между отправителем и получателем и в дальнейшем не меняется до тех пор, пока передача не закончится. Подсеть запоминает выбранный маршрут. После окончания передачи, когда соединение разрывается, виртуальный канал уничтожается. При подходе без соединения каждый пакет маршрутизируется независимо. Разные пакеты могут следовать разными маршрутами. Продвижение по разным маршрутам может требовать разное время. Вследствие такой организации подсеть более надежна, способна гибко реагировать на ошибки и перегрузки. Позже мы вернемся к обсуждению всех pro и contra этих двух подходов. Каждый маршрутизатор в сети, ориентированной на виртуальные каналы, должен помнить, какие каналы проходят через него. У каждого маршрутизатора есть таблица виртуальных каналов. Каждый пакет должен иметь дополнительное поле, где хранится номер виртуального канала. Когда пакет приходит к маршрутизатору, то, зная линию, по которой он пришел, и номер виртуального канала, указанный в пакете, по таблице маршрутизатор устанавливает, по какой линии надо отправить пакет далее. При установлении соединения номер виртуального канала выбирается из числа неиспользуемых в данный момент на данной машине. Так как каждая машина выбирает номер канала независимо, то этот номер имеет лишь локальное значение. Заметим, что каждый процесс должен указать ожидаемое время освобождения виртуального канала. В противном случае возникнут проблемы с принятием решения при освобождении виртуального канала: может быть, одна из машин на маршруте «зависла». Итак, при использовании виртуальных каналов транспортной среде предстоит немало работы. В случае дейтаграмм никакой таблицы виртуальных каналов в каждом маршрутизаторе иметь не надо. Вместо этого у них есть таблица, в которой указано, какую линию надо использовать, чтобы доставить пакет по тому или иному адресу. Такая таблица нужна и при виртуальных каналах, когда устанавливается соединение. У каждой дейтаграммы должен быть полный адрес доставки. В больших сетях этот адрес может быть достаточно большим (десятки байт). Когда пакет поступает, маршрутизатор по таблице и адресу определяет, по какой линии надо отправить эту дейтаграмму, и посылает ее туда.

Алгоритмы маршрутизации

Основной задачей сетевого уровня является маршрутизация пакетов. Пакеты маршрутизируются всегда, независимо от того, какую внутреннюю организацию имеет транспортная среда - с виртуальными каналами или дейтаграммную. Разница лишь в том, что в первом случае этот маршрут устанавливается один раз для всех пакетов, а во втором - для каждого пакета. В первом случае говорят иногда о маршрутизации сессии, потому что маршрут устанавливается на все время передачи данных пользователя, т.е. сессии.

Алгоритм маршрутизации - часть программного обеспечения сетевого уровня. Он отвечает за определение, по какой линии отправлять пакет дальше. Вне зависимости от того, выбирается маршрут для сессии или для каждого пакета в отдельности, алгоритм маршрутизации должен обладать рядом свойств: корректностью, простотой, устойчивостью, стабильностью, справедливостью и оптимальностью. Если корректность и простота комментариев не требуют, то остальные свойства надо разъяснить.

Алгоритм маршрутизации должен быть устойчивым, т.е. сохранять работоспособность независимо ни от каких сбоев или отказов в сети, изменений в ее топологии (отключение хостов, машин транспортной подсети, разрушения каналов и т.п). Алгоритм маршрутизации должен адаптироваться ко всем таким изменениям, не требуя перезагрузки сети или остановки абонентских машин.

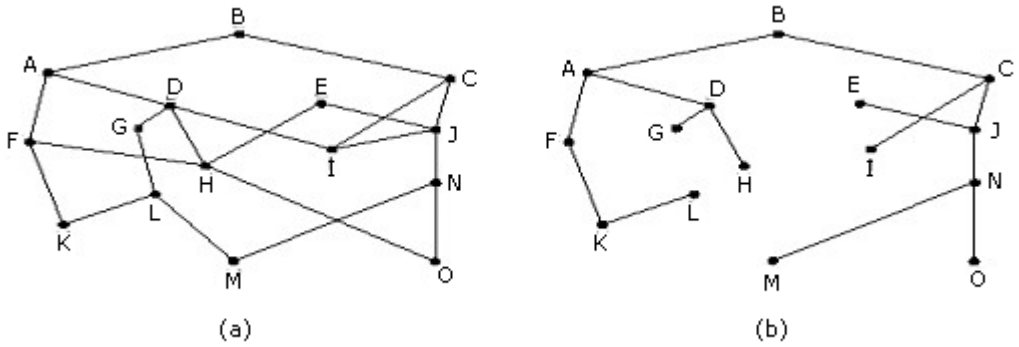
Стабильность алгоритма - также весьма важный фактор. Существуют алгоритмы маршрутизации, которые никогда не сходятся к какому-либо равновесному состоянию, как бы долго они ни работали. Это означает, что адаптация алгоритма к изменениям в конфигурации транспортной среды может оказаться весьма продолжительной. Более того, она может оказаться сколь угодно долгой.

Справедливость значит, что все пакеты, вне зависимости от того, из какого канала они поступили, будут обслуживаться равномерно, никакому направлению не будет отдаваться предпочтение, для всех абонентов будет всегда выбираться оптимальный маршрут. Надо отметить, что справедливость и оптимальность часто могут вступать в противоречие друг с другом. Прежде чем искать компромисс между оптимальностью и справедливостью, мы должны решить, что является критерием оптимизации. Один из возможных критериев - минимизация средней задержки пакета. Другой - максимизация пропускной способности сети. Однако эти критерии конфликтуют. Согласно теории массового обслуживания, если система с очередями функционирует близко к своему насыщению, то задержка в очереди увеличивается. Как компромисс, во многих сетях минимизируется число переходов между маршрутизаторами - один такой переход мы будем называть скачком (hop). Уменьшение числа скачков сокращает маршрут, а следовательно, сокращает задержку, а также минимизирует потребляемую пропускную способность при передаче пакета. Алгоритмы маршрутизации можно разбить на два больших класса: адаптивные и неадаптивные. Неадаптивные алгоритмы не принимают в расчет текущую загрузку сети и состояние топологии. Все возможные маршруты вычисляются заранее и загружаются в маршрутизаторы при загрузке сети. Такая маршрутизация называется статической маршрутизацией. Адаптивные алгоритмы, наоборот, определяют маршрут, исходя из текущей загрузки сети и топологии. Адаптивные алгоритмы различаются тем, где и как они получают информацию (локально от соседних маршрутизаторов или глобально от всех), когда они меняют маршрут (каждые T секунд, когда меняется нагрузка, когда меняется топология), какая метрика используется при оптимизации (расстояние, число скачков, ожидаемое время передачи).

Принцип оптимальности

Прежде чем мы приступим к рассмотрению конкретных алгоритмов маршрутизации, сформулируем принцип оптимальности. Этот принцип утверждает, что если маршрутизатор J находится на оптимальном пути между маршрутизаторами I и K, то оптимальный маршрут между J и K принадлежит этому оптимальному пути. Это так, поскольку существование между J и K оптимального маршрута, отличного от части маршрута между I и K, противоречил бы утверждению об оптимальности маршрута между I и K. Следствием из принципа оптимальности является утверждение, что все маршруты к заданной точке сети образуют дерево с корнем в этой точке. Это дерево называется деревом захода, оно проиллюстрировано на рисунке 5-5.

Рисунок 5-5. Дерево захода



Поскольку дерево захода - это дерево, то там нет циклов, поэтому каждый пакет будет доставлен за конечное число шагов. На практике все может оказаться сложнее. Маршрутизаторы могут выходить из строя, и, наоборот, появляться новые, каналы могут выходить из строя, разные маршрутизаторы могут узнавать об этих изменениях в разное время и т.д. и т.п.

Маршрутизация по наикратчайшему пути

Наше изучение алгоритмов маршрутизации мы начнем со статического алгоритма, широко используемого на практике в силу его простоты. Идея этого алгоритма состоит в построении графа транспортной среды, где вершины - маршрутизаторы, а дуги - линии связи. Алгоритм находит для любой пары маршрутизаторов, а точнее абонентов, подключенных к этим маршрутизаторам, наикратчайший маршрут в этом графе. В общем случае веса на дугах могут быть функциями от расстояния, пропускной способности канала, среднего трафика, стоимости передачи, средней длины очереди в буфере маршрутизатора к данному каналу и других факторов. Изменяя весовую функцию, алгоритм будет вычислять наикратчайший путь в смысле заданной метрики.

Маршрутизация лавиной

Другим примером статического алгоритма может служить следующий алгоритм: каждый поступающий пакет отправляют по всем имеющимся линиям, за исключением той, по которой он поступил. Ясно, что если ничем не ограничить число повторно генерируемых пакетов, то их число может расти неограниченно. Время жизни пакета ограничивают областью его распространения. Для этого в заголовке каждого изначально генерируемого пакета устанавливается счетчик переходов. При каждой пересылке этот счетчик уменьшается на единицу. Когда он достигает нуля, пакет сбрасывается и далее не посылается. В качестве начального значения счетчика выбирают наихудший случай, например, диаметр транспортной подсети. Другим приемом, ограничивающим рост числа дублируемых пакетов, является отслеживание на каждом маршрутизаторе тех пакетов, которые через него однажды уже проходили. Такие пакеты сбрасываются и больше не пересылаются. Для этого каждый маршрутизатор, получая пакет непосредственно от абонентской машины, помечает его надлежащим числом. В свою очередь, каждый маршрутизатор ведет список номеров, сгенерированных другим маршрутизатором. Если поступивший пакет уже есть в списке, то этот пакет сбрасывается. Для предотвращения безграничного роста списка вводят ограничительную константу k. Считается, что все номера, начиная с k и далее, уже встречались. Несмотря на кажущуюся неуклюжесть, этот алгоритм применяется, например, в распределенных базах данных, когда надо параллельно обновить данные во всех базах одновременно. Этот алгоритм всегда находит наикратчайший маршрут за самое короткое время, поскольку все возможные пути просматриваются параллельно.

Маршрутизация на основе потока

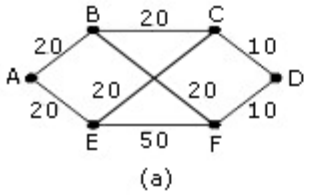
Алгоритмы, которые мы рассматривали до сих пор, принимали в расчет только топологию транспортной среды и никак не учитывали ее загрузку. Хотя, например, в том случае, когда наикратчайший маршрут перегружен, очевидно, лучше воспользоваться пусть более длинным, но менее загруженным маршрутом. Здесь мы рассмотрим статический алгоритм маршрутизации на основе потока, который учитывает как топологию, так и загрузку транспортной подсети. В некоторых сетях трафик между каждой парой узлов

известен заранее и относительно стабилен. Например, в случае взаимодействия сети торгующих организаций со складом. Время подачи отчетов, размер и форма отчетов известны заранее. В этих условиях, зная пропускную способность каналов, можно с помощью теории массового обслуживания вычислить среднюю задержку пакета в канале. Тогда нетрудно построить алгоритм, вычисляющий путь с минимальной задержкой пакета между двумя узлами.

Для реализации этой идеи нам нужно о каждой транспортной среде заранее знать следующее:

- топологию
- матрицу трафика F^{ij}
- матрицу пропускных способностей каналов C^{ij}
- алгоритм маршрутизации

Рисунок 5-8. (a) Топология ТС с пропускной способностью в Кбит/сек.; (b) Матрица с трафиком в пакетах/сек. и маршрутом



(a)

Назначение

Источник	A	B	C	D	E	F
A		9 AB	4 ABC	1 ABFD	7 AE	4 AEF
B	9 BA		8 BC	3 BFD	2 BFE	4 BF
C	4 CBA	8 CB		3 CD	3 CE	2 CEF
D	1 DFBA	3 DFB	3 DC		3 DCE	4 DF
E	7 EA	2 EFB	3 EC	3 ECD		5 EF
F	4 FEA	4 FB	2 FEC	4 FD	5 FE	

(b)

На рисунке 5-8 показан пример: (a) – топология транспортной среды с пропускной способностью каналов в Кбит/сек., (б) – матрица, где для каждой пары узлов (i, j) указан средний размер трафика в пакетах в секунду и маршрут для этого трафика. В таблице 5-9 показан итоговый трафик для некоторых пар соседних вершин. Предполагается здесь, что трафик на каждой линии симметричен, т.е. трафик X к Y идентичен трафику от Y к X. В этой таблице также показаны среднее число пакетов в секунду (μC_i), при средней длине пакета $1/\mu$=800 бит. В последнем столбце указана средняя величина задержки, вычисленная по формуле

$$T = \frac{1}{\mu C - \lambda}$$

Имея эти данные, нетрудно построить алгоритм вычисления наикратчайшего пути с точки зрения весов на дугах графа. Если трафик изменится по какой-либо причине, то достаточно перевычислить таблицу, не меняя алгоритма.

Сетевой уровень: проблемы построения сетевого уровня (Сервис, внутренняя организация сетевого уровня). Алгоритмы маршрутизации (маршрутизация по вектору расстояния, маршрутизация по состоянию канала, иерархическая маршрутизация, маршрутизация для мобильного узла, маршрутизация при вещании, маршрутизация для группы).

Маршрутизация по вектору расстояния

Алгоритм маршрутизации по вектору расстояния устроен следующим образом: у каждого маршрутизатора в транспортной подсети есть таблица расстояний до каждого маршрутизатора, принадлежащего подсети. Периодически маршрутизатор обменивается такой информацией со своими соседями и обновляет

информацию в своей таблице. Каждый элемент таблицы состоит из двух полей: первое - номер канала, по которому надо отправлять пакеты, чтобы достичь нужного места, второе - величина задержки до места назначения. Величина задержки может быть измерена в разных единицах: числе переходов, миллисекундах, длине очереди на канале и т.д. Фактически в протоколе использовалась версия алгоритма, где эту задержку определяли не на основе пропускной способности канала, а на основе длины очереди к каналу. Каждые T секунд маршрутизатор шлет своим соседям свой вектор задержек до всех маршрутизаторов в подсети. В свою очередь, он получает такие же вектора от своих соседей. Кроме этого, он постоянно замеряет задержки до своих соседей. Поэтому, имея вектора расстояний от соседей и зная расстояние до них, маршрутизатор всегда может вычислить кратчайший маршрут до определенного места в транспортной среде. Рассмотрим, как маршрутизатор J с помощью этой таблицы вычислит маршрут до G. J знает, что он может достичь A за 8 мсек., A объявляет, что от него до G 18 мсек. Таким образом, J может достичь G за 26 мсек. через A. Аналогично можно подсчитать, что достичь G через I, H и K можно за 41 (31+10), 18 (6+12) и 37 (31+6) мсек. соответственно. Наилучшее значение – 18, поэтому это и есть наилучший маршрут.

Проблема счетчика до бесконечности

Алгоритм маршрутизации по вектору расстояния теоретически работает хорошо, но у него есть один недостаток: он очень медленно реагирует на разрушения каналов в транспортной среде. Информация о появлении хорошего маршрута в подсети распространяется более или менее быстро, а вот данные о потере, разрушении какого-то маршрута распространяются не столь быстро. Рассмотрим пример на рисунке 5-11 (a). В нем показана линейная транспортная среда. Пусть изначально маршрутизатор A не работал. Поэтому у всех маршрутизаторов в подсети расстояние до него было равно ∞. Пусть в какой-то момент времени A был включен. По истечении определенного времени маршрутизаторы начнут обмениваться векторами, и B узнает об A. Еще через один обмен векторами об A узнает C, и т.д. Таким образом, информация о новом маршруте будет распространяться линейно шаг за шагом, за каждый обмен векторами. Если самый длинный маршрут в подсети имеет длину N, то потребуется N обменов векторами, пока информация о новом маршруте дойдет до самого удаленного узла в подсети.

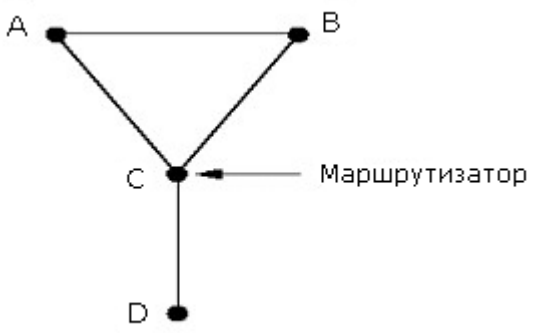
Теперь рассмотрим обратную ситуацию на рисунке 5-11(b). Здесь изначально все маршрутизаторы и каналы были работоспособны. Пусть в какой-то момент времени канал между A и B оказался разрушен. B перестает видеть A, но C говорит B: «Не беспокойся, у меня есть маршрут до A». При этом B не подозревает, что маршрут от C до A идет через него же. Маршрутизаторы D и E своих таблиц не меняют. Их расстояния до A на единицу больше, чем у C. На втором обмене C увидит, что оба его соседа достигают A за 3 единицы. C выбирает одного некоторым случайным образом, увеличив значение 3 на единицу. Плохая весть будет распространяться медленно, пока счетчики задержек не примут значения бесконечности для данной сети. Только после этого станет ясно, что A не достижим ни через C, ни через D, ни через E. Сколько времени на это потребуется, зависит от конкретного значения бесконечности в данной подсети.

Разделение направлений (Split Horizon Hack)

Одним из решений этой проблемы является следующий прием. Алгоритм работает так, как было описано, но при передаче вектора по линии, по которой направляются пакеты для маршрутизатора X, т.е. по которой достижим маршрутизатор X, расстояние до X указывается как бесконечность. Если теперь рассмотреть то, как будет работать подсеть на рисунке 5-11(b), то там проблем возникать не будет. Действительно, когда A «упадет», при первом же обмене B это обнаружит, но C также будет слать B вектор, согласно которому A не достижимо (∞). На следующем обмене C увидит, что A недостижим из обоих его соседей, и также отметит его как недостижимый узел.

Однако и в алгоритме разделения направлений есть «дыры». Рассмотрим подсеть на рисунке 5-12. Если линия между C и D будет разрушена, то C сообщит об этом A и B. Однако A знает, что у B есть маршрут до D, а B знает, что такой маршрут есть и у A. И опять мы «сваливаемся» в проблему бесконечного счетчика.

Рисунок 5-12. Случай, при котором разделение направлений не помогает



Маршрутизация по состоянию канала

Основная идея построения этого алгоритма проста и состоит из пяти основных шагов:

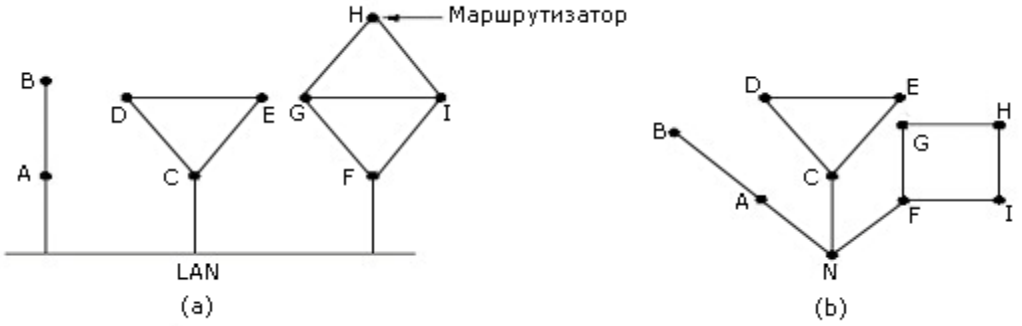
1. Определить своих соседей и их сетевые адреса.
2. Измерить задержку или оценить затраты на передачу до каждого соседа.
3. Сформировать пакет, где указаны все данные, полученные на шаге 2.
4. Послать этот пакет всем другим маршрутизаторам.
5. Вычислить наикратчайший маршрут до каждого маршрутизатора.

Топология и все задержки оцениваются экспериментально и сообщаются всем узлам. После этого можно использовать, например, алгоритм Дейкстры для вычисления наикратчайшего маршрута. Теперь рассмотрим подробнее эти пять шагов.

1. Определение соседей

При загрузке маршрутизатор прежде всего определяет, кто его соседи. Для этого он рассылает по всем своим линиям точка-точка специальный пакет HELLO. В ответ все маршрутизаторы отвечают, указывая свое уникальное имя. Имя маршрутизатора должно быть уникальным в сети, чтобы избежать неоднозначностей. Если же два и более маршрутизатора соединены одним каналом, как на рисунке 5-13 (a), то этот канал в графе связей представляют отдельным, искусственным узлом (рисунок 5-13 (b)).

Рисунок 5-13. Определение соседей



2. Оценка затрат

Оценка затрат до каждого соседа происходит с помощью другого специального пакета ECHO. Это пакет рассылается всем соседям, при этом замеряется задержка от момента отправки этого пакета до момента его возвращения. Все, кто получает такой пакет, обязаны отвечать незамедлительно. Такие замеры делают несколько раз и вычисляют среднее значение. Таким образом, длина очереди к каналу не учитывается. Здесь есть одна тонкость: учитывать загрузку в канале или нет? Если учитывать, то задержку надо замерять от момента поступления пакета в очередь к каналу. Если не учитывать, то от момента, когда пакет достиг

головы очереди. Есть доводы, как в пользу учета нагрузки, так и против такого учета. Учитывая нагрузку, мы можем выбирать между двумя и более каналами с одинаковой пропускной способностью, получая лучшую производительность. Однако можно привести примеры, когда ее учет вызывает проблемы. Здесь два одинаковых по производительности канала CF и EI соединяют две сети. Если на одном из них нагрузка меньше, то предпочтительнее будет другой, что через некоторое время приведет к его перегрузке и предпочтительнее должен стать первый. Если нагрузка не учитывается, то этой проблемы не возникает.

3. Формирование пакета состояния канала

После того как измерения выполнены, можно сформировать пакет о состоянии каналов. На рисунке 5-14 показаны пакеты для примера сети. В пакете указаны: отправитель, последовательное число, возраст (назначение этих полей станет ясно позднее), список соседей и задержки до них. Формирование таких пакетов не вызывает проблем. Основной вопрос - когда их формировать? Периодически, с каким-то интервалом, или по особому событию, когда в транспортной подсети произошли какие-то существенные изменения?

4. Распространение пакетов состояния каналов

Наиболее хитрая часть этого алгоритма – как надежно распространить пакеты о состоянии каналов (СК-пакеты)? Как только СК-пакет получен и включен в работу, маршрутизатор будет его использовать при определении маршрута. При неудачном распространении СК-пакетов разные маршрутизаторы могут получить разное представление о топологии транспортной среды, что может приводить к возникновению циклов, недостижимых машин и другим проблемам. Сначала рассмотрим базовый алгоритм, потом некоторые его усовершенствования. СК-пакеты распространяются методом лавины, т.е. СК-пакет рассылается всем соседям, те, в свою очередь, своим соседям, и т.д. Однако, чтобы не потерять контроль и не вызвать неограниченное дублирование СК-пакетов, каждый маршрутизатор ведет счетчик последовательных номеров СК-пакетов, которые он сгенерировал. Все маршрутизаторы запоминают пары «маршрутизатор, последовательное число», которые они уже встречали среди полученных СК-пакетов. Если поступивший СК-пакет содержит пару, которая еще не встречалась маршрутизатору, то он отправляет этот СК-пакет всем своим соседям, за исключением того, от которого он его получил. Если он уже встречал такой пакет, то пакет сбрасывается и никуда не дублируется. У этого алгоритма есть несколько проблем, но все они разрешимые. Первая: размер поля последовательных номеров пакетов. Если оно будет недостаточно длинное, то его переполнение приведет к повтору номеров, что, в свою очередь, приведет к некорректной работе всего алгоритма. Решением является достаточно большое поле, например, 32-разрядное. При таком поле, если обмен СК-пакетами происходит раз в секунду, то потребуется 137 лет, чтобы возникло переполнение. Вторая проблема: если маршрутизатор «упал» по какой-либо причине и потерял последовательность использованных последовательных номеров, то неясно, как ее восстановить. Третья – если в результате передачи возникнет ошибка в одном бите, например, вместо 4 получим пакет с номером 65540, то все пакеты с 5 по 65540 будут сбрасываться как устаревшие, поскольку текущий номер - 65540. Для решения этих проблем используется поле «возраст» СК-пакета. Там устанавливается некоторая величина, которая уменьшается периодически на единицу каждую секунду. Когда она достигнет нуля, пакет сбрасывается. В целях сокращения числа рассылаемых СК-пакетов, когда такой пакет поступает, его не сразу дублируют и отправляют. Сначала его помещают в специальную область задержки. Там он находится некоторое время. Если за это время придет другой пакет от того же источника, то пакеты сравниваются. Если нет различий между ними, то вновь пришедший сбрасывается, если есть, то последний пришедший дублируется и отправляется другим, а первый сбрасывается. Все СК-пакеты передаются с уведомлением. Флаг отправления означает, что соответствующий СК-пакет должен быть послан по указанной линии. Флаг уведомления указывает на то, что по соответствующей линии должно прийти подтверждение.

Наличие дубликатов СК-пакетов легко распознать по состоянию флагов отправки.

5. Вычисление нового пути

Когда маршрутизатор получил полный комплект СК-пакетов, он может построить топологию транспортной среды и, например, локально запустить алгоритм Дейкстры для вычисления наикратчайшего пути. В системе, где есть n маршрутизаторов с k линий у каждого, каждый маршрутизатор должен иметь достаточно памяти, чтобы хранить необходимую информацию о сети. При больших n эта величина может стать существенной. Кроме этого, в сетях, где число маршрутизаторов достигает десятков или сотен тысяч, проблемы, вызванные сбоем одного из них, могут оказаться весьма серьезными. Например, если из-за сбоя в маршрутизаторе послан неправильный СК-пакет или неправильно оценено состояние канала, то в дальнейшем это может привести к проблемам. Маршрутизация по состоянию каналов широко используется в реальных сетях. Например, в протоколе OSPF, который мы будем подробно рассматривать позже. Другим важным примером может служить протокол IS-IS, который был изначально предложен фирмой DEC, а позже адаптирован ISO. IS-IS широко используется в Интернете. Поскольку OSPF появился позже IS-IS, он вобрал в себя все усовершенствования, сделанные для IS-IS. Основное различие между ними в том, что IS-IS может работать с разными протоколами сетевого уровня, что делает его весьма привлекательным при маршрутизации между разнотипными сетями, чего не может делать OSPF.

Иерархическая маршрутизация

По мере роста транспортной среды размер таблиц, т.е. затраты памяти, время процессора на обработку этих таблиц, пропускная способность каналов, затрачиваемая на передачу служебной информации, может превысить разумные пределы. Таким образом, дальнейший рост сети, когда каждый маршрутизатор знает все о каждом другом маршрутизаторе, будет невозможен. Решение этой проблемы – иерархия сетей, подобно иерархии коммутаторов в телефонной сети. Нетрудно видеть, что таблица маршрутизации во втором случае резко сократилась. Однако за эту экономию приходится платить эффективностью маршрутизации. При построении иерархии возникает сразу несколько вопросов. Один из них: сколько уровней должно быть в иерархии при заданном размере сети? Например, если наша транспортная среда содержит 920 маршрутизаторов, то без иерархии таблица каждого будет иметь 920 строк. Если транспортную среду разбить на 40 регионов, то размер таблиц будет равен 23 строки для маршрутизации внутри региона плюс 40 для маршрутизации между регионами. Если использовать трехуровневую иерархию и объединить регионы в кластеры, то при 5 кластерах, по 8 регионов с 23 маршрутизаторами в каждом у таблицы будет 23 входа для внутрикластерной маршрутизации, плюс 7 входов для межкластерной, плюс 5 входов для межрегиональной маршрутизации. Итого 35 входов. Клейнрок и Камоун показали, что оптимальное число уровней иерархии в транспортной среде при N узлах будет равняться lnN, при e*lnN строках в таблице маршрутизатора.

Маршрутизация для мобильного узла

Миллионы людей в наши дни путешествуют, находятся в командировках. Многим из них просто необходимо иметь доступ к своей электронной почте, файловой системе. Так мы приходим к проблеме мобильного узла, которую мы уже отмечали. Это относительно новая проблема, но, несмотря на это, она стоит довольно остро. На рисунке 5-17 показана модель WAN с мобильным узлом. Всех пользователей мы можем разделить на две большие группы. Стационарные - это большая группа, их компьютеры подключены к сети стационарными средствами (проводами, кабелями) и редко меняют свое место положение. Другая группа постоянно меняет свое местоположение и стремится поддерживать связь с сетью. Этих пользователей мы будем называть мобильными. Предполагается, что в сети каждый пользователь имеет постоянное домашнее местоположение, которое никогда не меняется. Проблема маршрутизации в этих условиях заключается в том, чтобы посылать пакеты мобильному пользователю через его домашнее местоположение. При этом то, где находится сам пользователь, не имеет значения. Вся WAN на рисунке 5-17 разбивается на области. В каждой области есть агент визитеров, который знает о всех мобильных пользователях в своей области. В свою очередь, в каждой области есть домашний агент, который знает обо всех стационарных пользователях в своей области, которые в настоящий момент путешествуют. Как только мобильный узел подключается к местной, локальной сети, он регистрируется у агента визитеров. Эта процедура примерно выглядит так:

1. Периодически агент визитеров рассылает по своей области пакет, где указано местоположение этого агента и его адрес. Если мобильный узел, подключившись к сети, долго не видит такого пакета, он рассылает свой пакет с просьбой агенту визитеров объявить свои координаты.
2. Мобильный узел регистрируется у агента визитеров, указывая свое текущее местоположение, домашнее местоположение и определенную информацию, связанную с безопасностью передаваемых данных.
3. Агент визитеров обращается через сеть к домашнему агенту домашнего местоположения визитера, указывая, что один из его пользователей сейчас находится в его области, передавая конфиденциальную информацию, которая должна убедить домашнего агента, что это действительно его пользователь пытается соединиться с ним.
4. Домашний агент изучает конфиденциальные данные, время связи. Если эти данные соответствуют той информации, что есть у домашнего агента об этом пользователе, он дает добро на связь.
5. Агент визитеров, получив подтверждение от домашнего агента, заносит данные о мобильном узле в свои таблицы и регистрирует его.

В идеале пользователь, покидая область визита, должен закрыть свою временную регистрацию. Однако, как правило, закончив сеанс связи, пользователь просто выключает свой компьютер и все. Поэтому, если по прошествии некоторого времени пользователь не объявился вновь, агент визитеров считает его покинувшим область.

Рассмотрим теперь, что происходит, когда кто-то посылает сообщения мобильному узлу (рисунок 5-18). Пакет поступает на адрес домашнего местоположения пользователя, где его перехватывает домашний агент. Домашний агент инкапсулирует этот пакет в свой пакет, который он отправляет по адресу агента визитеров той области, откуда последний раз был сеанс связи с пользователем. Одновременно с этим домашний агент посылает сообщение отправителю пакета, чтобы он все последующие пакеты мобильному узлу инкапсулировал в сообщениях, направляемых по адресу агента визитеров. Такой механизм инкапсулирования одних пакетов в другие называется туннелированием, и мы его подробно рассмотрим позднее. Здесь мы обрисовали лишь в общих чертах основную схему работы. Конкретных схем существует множество, которые различаются разными аспектами. Прежде всего тем, как распределяется работа между маршрутизаторами и хостами, какой уровень в стеке протоколов хоста отвечает за реализацию соответствующих протоколов. Во-вторых, есть схемы, где маршрутизаторы запоминают информацию о местонахождения мобильных узлов и могут вмешиваться в диалог между агентом визитеров и домашним агентом, по-разному маршрутизируя трафик. В некоторых схемах мобильный узел получает некоторый уникальный адрес, в других это адрес агента, который отвечает за маршрутизацию всего трафика мобильных узлов. Кроме этого, схемы различаются разным уровнем безопасности передаваемой информации.

Маршрутизация при вещании

В некоторых приложениях возникает потребность переслать одно и то же сообщение всем машинам. Например, прогноз погоды, биржевые сводки, новости и т.д. Такой режим передачи называется вещанием. Есть несколько способов реализации такого режима. Первый способ: источник знает, кому надо послать, и генерирует столько сообщений, сколько получателей. Это одно из самых плохих решений. Оно не требует никаких специальных средств, однако весьма накладно. Тратится не только пропускная способность каналов, но и память – ведь где-то кому-то надо хранить весь лист рассылки. Метод лавины – другое решение. Однако, как мы уже видели, он затратен для каналов «точка-точка». Он слишком сильно расходует пропускную способность. Третий подход – маршрутизация множественной доставки. Здесь каждый пакет должен иметь либо лист рассылки, либо карту рассылки. Каждый маршрутизатор, получив такой пакет, отправляет и дублирует его в соответствии с картой рассылки. Четвертый подход основан на использовании дерева захода, либо любого другого подходящего дерева связей. Дерево захода позволяет избежать циклов и ненужного дублирования пакетов. Каждый маршрутизатор дублирует пакет вдоль линий, соответствующих дереву захода, кроме той, по которой пакет пришел. В этом подходе очень рационально используется пропускная способность каналов, генерируется абсолютный минимум пакетов при рассылке. Однако каждый маршрутизатор где-то должен иметь дерево захода, соответствующее случаю. Пятый подход основан на неявном использовании дерева связей. Когда пакет поступает, маршрутизатор проверяет: если он поступил по линии, которая используется для отправления пакетов источнику вещательного пакета, то вещательный пакет дублируется и рассылается по всем линиям, кроме той, по которой пакет пришел. Если нет, то он

сбрасывается. Этот метод называется пересылкой вдоль обратного пути. На рисунке 5-19 показан пример работы этого алгоритма. На рисунке 5-19 (a) показана топология транспортной среды. На рисунке 5-19 (b) показано дерево захода для вершины I, часть (c) показывает, как работает этот алгоритм. Сначала в вершине I было сгенерировано и разослано 4 пакета. Во всех четырех вершинах (F, H, J, N), куда поступили эти пакеты, они поступили с предпочтительного для I направления. Из восьми пакетов, сгенерированных на следующем этапе, дереву захода только пять поступили по предпочтительному для I направлению. На третьем этапе из шести сгенерированных пакетов только три поступили по предпочтительному направлению (G, D, N поступили дубликаты). После того, как пять этапов пройдены и сгенерированы 23 пакета, рассылка прекращается. Достоинством этого метода является простота и легкость в реализации.

Маршрутизация при групповой передаче

Этот вид передачи используют, когда надо обеспечить взаимодействие группы взаимосвязанных процессов, разбросанных по сети. Такие ситуации часто встречаются в распределенных базах данных. Если группа велика по сравнению с размерами сети, то можно воспользоваться методами вещания. Однако если ее размер мал относительно размера всей сети, то такой подход будет неэкономичен. Кроме того, если рассылаемая в группе информация конфиденциальная, то алгоритмы вещания не подходят. Этот вид маршрутизации называют групповой маршрутизацией. В случае обмена информации в группе, кроме алгоритма групповой маршрутизации, нужны алгоритмы управления группой, которые должны обеспечивать средства для реконфигурации группы: включение новых членов, удаление старых и т.п. Однако эти проблемы не затрагивают алгоритм групповой маршрутизации, поэтому мы их здесь рассматривать не будем.

Алгоритм групповой маршрутизации, как правило, основан на дереве связей. Каждый маршрутизатор в транспортной среде вычисляет дерево связей, охватывающее все другие маршрутизаторы. Для обрезания дерева связей используются разные алгоритмы. Наиболее простой основан на применении алгоритма маршрутизации на основе состоянии канала. В этом случае каждый маршрутизатор имеет полную конфигурацию транспортной среды. Сокращение дерева связей начинается от вершин – членов группы и разворачивается в сторону корня, удаляя все маршрутизаторы, которые не ведут к членам группы. Существуют и другие подходы.

Сетевой уровень: проблемы построения сетевого уровня (Сервис, внутренняя организация сетевого уровня). Алгоритмы управления перегрузками на сетевом уровне (Основные принципы управления перегрузками, методы предотвращения перегрузок: формирование трафика, спецификация потока, управление перегрузками в сетях с виртуальными каналами; методы устранения перегрузок: подавляющие пакеты, сброс трафика; управление перегрузками при вещании).

Когда в транспортной среде находится в одно и тоже время слишком много пакетов, ее производительность начинает падать. Перегрузка может возникнуть в силу нескольких причин. Например, если сразу несколько потоков, поступающих по нескольким входным линиям, устремятся на одну и ту же выходную линию. Очередь на этой линии может расти бесконечно, и пакеты начнут посылаться повторно, так как они слишком долго будут находиться в очереди. Если буфер маршрутизатора переполнится, то пакеты начнут теряться. Увеличение памяти в этом случае вряд ли исправит положение. Пакеты долго будут находиться в памяти, и отправители начнут их дублировать. Перегрузки могут случаться и из-за недостаточной скорости процессора. Если процессор будет не в состоянии справиться своевременно с рутинными задачами (размещения пакета в буфере, корректировка таблиц и т.п.), то даже при наличии линий с достаточной пропускной способностью очередь будет расти. Аналогичная картина может случиться при быстром процессоре, но медленном канале и наоборот. Таким образом, источник проблемы - несбалансированность производительности компонентов системы. Перегрузки имеют тенденцию к самостоятельному росту и ухудшению ситуации. Если у маршрутизатора не хватает памяти буфера, то он начинает сбрасывать пакеты. Отправитель, не получая пакеты, начинает их повторять снова и снова, усугубляя положения получателя. Надо различать управление перегрузками сети и управление потоком. Перегрузка - это глобальная проблема в сети. Управление перегрузками - это такая организация потоков в транспортной среде, при которой потоки соответствуют пропускной способности подсети и не превышают ее. Это глобальная проблема в сети, затрагивающая поведение всех хостов и всех маршрутизаторов. Управление потоком возникает между парой взаимодействующих машин. Это локальная проблема, касающаяся двух взаимодействующих машин. Ее

решение гарантирует, что быстрый отправитель сообщений не «завалит» нерасторопного получателя. Здесь яркими примерами могут быть: один быстрый компьютер передает файл в 1 Гб более медленному компьютеру через сеть с пропускной способностью 1 Тбит/сек. со скоростью 1 Гбит/сек. Ясно, что здесь не будет перегрузки, хотя быстрый компьютер может создать такой поток пакетов, что он захлестнет медленный. В тоже время, если в сети с линиями на 1 Мбит/сек. и 1000 компьютеров хотя бы половина машин начнет передавать файлы со скоростью 100 Кбит/сек. другой половине, то ясно, что перегрузки не избежать. Часто управление перегрузкой и управление потоком путают из-за того, что и там и там применяют одинаковые приемы, например, направляют источникам специальные пакеты, тормозящие нарастание потоков.

Основные принципы управления перегрузками

В терминологии теории управления все методы управления перегрузками в сетях можно разбить на две большие группы: с открытым контуром управления и закрытым контуром управления. Методы с открытым контуром предполагают, что все продумано и предусмотрено заранее в конструкции системы, и если нагрузка находится в заданных пределах, то перегрузки не происходит. Если же нагрузка начинает превышать определенные пределы, то заранее известно, когда и где начнется сброс пакетов, в каких точках сети начнется перепланировка ресурсов, и т.п. Главное, что все эти меры будут приниматься вне зависимости от текущего состояния сети.

Решения, основанные на закрытом контуре, используют обратную связь. Эти решения включают три этапа:

- Наблюдение за системой для определения, где и когда началась перегрузка
- Передача данных туда, где будут предприняты надлежащие меры
- Перестройка функционирования системы для устранения проблемы

При наблюдении за системой используются разные метрики для определения перегрузки. Основными среди них являются:

- процент пакетов, сброшенных из-за нехватки памяти в буферах
- средняя длина очередей в системе
- число пакетов, для которых наступил time_out и для которых были сделаны повторные передачи
- средняя задержка пакета при доставке и среднее отклонение задержки при доставке пакета

Следующий шаг при использовании обратной связи - передать информацию о перегрузке туда, где что-то может быть сделано, чтобы исправить положение. Например, маршрутизатор, обнаруживший перегрузку, может направить сообщение о перегрузке всем источникам сообщений. Ясно, что это увеличит нагрузку в сети, причем именно в тот момент, когда это менее всего желательно. Однако есть и другие возможности. Например, в каждом пакете зарезервировать специальный бит перегрузки, и если какой-то маршрутизатор обнаружил перегрузку, то он устанавливает этот бит, тем самым сообщая другим о ней (вспомним структуру кадра во Frame Relay). Другое решение напоминает прием, используемый некоторыми радиостанциями: направлять несколько автомашин по дорогам, чтобы обнаруживать пробки, а затем сообщать о них по радиоканалам, предупреждая другие машины, призывая их пользоваться объездными путями. По аналогии с этим решением в сети рассылаются специальные пробные пакеты, которые проверяют нагрузку, и если где-то обнаружена перегрузка, то о ней сообщается всем и происходит перенаправление пакетов так, чтобы обогнуть перегруженные участки. Алгоритмы управления перегрузками подразделяются, как мы уже сказали, на решения с открытым и решения с закрытым контуром. Решения с открытым контуром, в свою очередь, делятся на две группы: воздействующие на источники и воздействующие на получателей. Решения с закрытым контуром - с явной обратной связью и неявной обратной связью. Явная обратная связь предполагает, что источнику посылается специальный пакет, который информирует его о перегрузке. Неявная обратная связь основана на том, что источник сам определяет факт перегрузки на основе своих локальных наблюдений за трафиком, например, по величине задержки на поступление уведомления о доставке пакета. Появление перегрузки означает, что нагрузка превысила, возможно временно, ресурсы системы или некоторой ее части. Есть два выхода из этого положения: увеличить ресурсы и сократить нагрузку. Увеличить ресурсы чаще всего невозможно. Тогда остается только сокращение нагрузки. Для этого

есть несколько способов: отказать некоторым пользователям в сервисе, ухудшить сервис всем или некоторым пользователям, заставить пользователей планировать свои потоки определенным образом.

Методы, предотвращающие перегрузки

Рассмотрение методов, предотвращающих перегрузки, начнем с методов для систем с открытым контуром. Эти методы ориентированы на минимизацию перегрузок при первых признаках их проявлений, а не на борьбу с перегрузками, когда они уже случились. Основные факторы, влияющие на перегрузки на канальном, сетевом и транспортном уровнях, перечислены в таблице 5-22.

Таблица 5-22. Факторы, влияющие на перегрузки

Уровень	Факторы
Транспортный	Повторная передача Порядок передачи бит Уведомления Управление потоком Значение timeout
Сетевой	Виртуальные каналы vs. дейтаграммы внутри подсети Очередность пакетов и сервисы Сброс пакета Алгоритм маршрутизации Управление временем жизни пакетов
Канальный	Повторная передача Порядок передачи бит Уведомления Управление потоком

Начнем с канального уровня. Вызвать перегрузку может повторная пересылка кадров. Если у источника сообщений часто возникает time_out и он начинает повторно передавать пакет, то тем самым он лишь усугубляет положение. Близко к этому фактору стоит нарушение порядка следования пакетов при передаче. Если получатель часто сбрасывает пакеты, поступившие не в надлежащем порядке от источника, то их повторная передача будет также усугублять перегрузку. Организация рассылки уведомлений также влияет на перегрузку. Если уведомление происходит немедленно и специальными пакетами, то это увеличивает трафик и следовательно может привести к перегрузкам. Если для уведомления используются пакеты с сообщениями (прием piggybacking), то возможны time_out из-за отсутствия уведомлений вовремя и, как следствие, повторные пересылки пакетов, что может привести к перегрузкам. В то же время жесткая схема управления потоком (небольшое окно) сдерживает нарастание трафика и предотвращает появление перегрузок. На сетевом уровне выбор схемы работы - с виртуальными соединениями или дейтаграммами - влияет на появление перегрузок. На этом уровне большинство методов борьбы с перегрузками ориентированы на виртуальные соединения. Методы управления очередями, организация очередей: одна общая на входе или одна общая на выходе; по одной на каждую входную линию или на каждую выходную; по одной очереди на каждую входную и выходную - все это влияет на появление перегрузок. Выбор метода сброса пакетов также влияет на перегрузки. Правильная маршрутизация, равномерно использующая каналы в транспортной среде, позволяет избежать перегрузки. Методы, регулирующие время жизни пакета в сети, также влияют на образование перегрузок. Если пакет долго блуждает в сети, прежде чем будет принято решение о его сбросе, то это плохо, так как увеличивает трафик и может привести к перегрузке. Если

поторопиться, то преждевременный сброс пакета может привести к повторным передачам, что опять-таки увеличит нагрузку. На транспортном уровне возникают те же самые проблемы, что и на канальном, однако определить величину time_out намного сложнее. Дело в том, что оценить время передачи через СПД много сложнее, чем время передачи по каналу «точка-точка» между двумя маршрутизаторами. Если оно будет слишком большим, то это снизит вероятность перегрузки, но повлияет на производительность из-за длительного ожидания поступления пакета. Если сделать его коротким, то появятся лишние пакеты.

5.3.3. Формирование трафика

Одной из основных причин перегрузки является нерегулярный, взрывообразный трафик в сети. Если бы он был равномерным, то перегрузок можно было бы избежать. Один из методов с открытым контуром, часто используемым особенно в АТМ-сетях, - метод формирования трафика (shaping - т.е. придание формы), когда скорость передачи пакетов контролируется и регулируется. Формирование трафика регулирует среднюю скорость передачи данных так, чтобы сделать его по возможности гладким. Следует обратить внимание, что протокол скользящего окна, который мы рассматривали при изучении канального уровня, лишь регулирует объем данных, передаваемых за один раз, но не скорость передачи. Здесь же речь идет именно о скорости передачи. Когда виртуальное соединение устанавливается, то пользователь договаривается с транспортной средой о форме трафика. Если пользователь обеспечивает договоренную форму трафика, то транспортная среда обеспечивает ему доставку трафика с определенной скоростью. Для таких приложений, как передача видео- и аудиоданных в реальном времени, это очень важно. Здесь уместно вспомнить организацию работы СПД Frame Relay. Когда пользователь и транспортная среда договариваются о форме трафика, то они приходят к соглашению не только о форме трафика, но также и о том, что произойдет, если эта форма будет нарушена пользователем. Это соглашение называется соглашением о трафике. Использование техники формирования трафика и соглашения о трафике легче всего реализовать при использовании виртуальных соединений, чем в случае дейтаграмм. В случае дейтаграмм эти идеи могут быть применены к соединениям на транспортном уровне.

5.3.3.1. Алгоритм текущего ведра

Идея этого алгоритма показана на рисунке 5-23 (а). Ведро может наполняться с любой скоростью, но вытекать из него вода будет со строго определенной скоростью. Если вода будет поступать слишком быстро, то ее часть будет переливаться через края и пропадать. Скорость истечения воды из ведра зависит только от размера отверстия в днище. Каждая станция, подключенная к сети, имеет подобие текущего ведра в своем интерфейсе. Не важно, сколько процессов посылает пакеты в сеть. Если буфер переполнен, то пакеты будут сбрасываться в соответствии с соглашением о трафике. Это не что иное, как сервер с постоянной скоростью обслуживания. В качестве регулятора скорости поступления пакетов можно использовать системные часы. В этом случае устанавливается предел числа пакетов, которые процесс может направить в сеть за один промежуток времени. Этот прием дает хорошие результаты, когда пакеты имеют фиксированную длину, как в АТМ. В случае пакетов переменной длины это соглашение ограничивает количество байтов, направляемых в сеть. Например, если разрешается за один промежуток послать 2048 байт, то это может быть два пакета по 1024 или 4 пакета по 512 байт. Если же пакет больше чем 2048, например, 2560 байт, то он должен ждать следующего временного промежутка. На рисунке 5-24 дан пример использования алгоритма текущего ведра со счетчиком байтов. Если имеется буфер С на 1 Мбит и скорость истечения равна 2 Мбит/сек., то при всплеске в 1 Мбит в течение 40 мсек. мы легко справимся с таким трафиком. При этих условиях даже не важно, с какой скоростью будут поступать этот 1 Мбит, главное чтобы этот всплеск не растянулся более чем на 500 мсек. На рисунке 5-24 (с) - (f) показана работа алгоритма текущего ведра при разных скоростях входного потока.

5.3.3.2. Алгоритм ведра с маркерами

Алгоритм текущего ведра позволяет сгладить трафик, убрать нерегулярность. Однако в целом ряде приложений бывает полезно разрешить, при всплеске трафика и наличии необходимых ресурсов, ускорить на некоторое время передачу пакетов в сеть. Один из алгоритмов, позволяющих это сделать, - алгоритм ведра с маркерами. Рисунок 5-25 иллюстрирует этот алгоритм. Идея его заключается в том, что вместе с пакетами в ведро поступают маркеры. Пакеты из ведра уходят в сеть только при наличии соответствующего количества

маркеров. Таким образом, можно накапливать маркеры и кратковременно ускорять передачу пакетов в сеть. Другое отличие алгоритма ведра с маркером - при переполнении буфера хосту будет временно запрещено передавать пакеты. Здесь опять существуют разные варианты в зависимости от длины пакетов, правила работы со счетчиком маркеров и т.д. Для реализации алгоритма ведра с маркерами нужна лишь переменная, значение которой увеличивается каждые ΔT сек. и уменьшается с каждым посланным пакетом. В случае пакетов переменной длины значение этой переменной увеличивается на k байтов каждые ΔT сек. и уменьшается на длину каждого посланного пакета. Нетрудно рассчитать длительность всплеска при передаче на основе уравнения

$$C + \rho S = MS$$

где C – объем буфера, S – длительность всплеска, ρ – скорость поступления маркера, M – максимальная скорость «вытекания» данных.

Таким образом, $S = C/(M - \rho)$.

Рисунок 5-24 (d-f) иллюстрирует эту функцию для случая C = 250 Кбит, M = 25 Мбит/сек. и ρ = 2 Мбит/сек.

5.3.4. Спецификация потока

Формирование трафика эффективно тогда, когда отправитель, получатель и среда передачи заранее договорились о форме трафика. Это соглашение называется спецификацией потока. Она представляет собой структуру данных, которая определяет как форму выходного трафика, так и качество сервиса, необходимого приложению. Эта спецификация применима как к пакетам, передаваемым по виртуальным каналам, так и к дейтаграммам.

В таблице 5-26 показан пример спецификации потока. Левый столбец определяет характеристики выходного потока. Правый определяет то, что приложение ожидает от СПД-среды.

Таблица 5-26. Пример спецификации потока

Характеристики выходного потока	Желаемый сервис
Максимальный размер пакета (байт)	Чувствительность к потерям (байт)
Пропускная способность ведра с маркерами (байт/сек.)	Интервал между потерями (мксек.)
Размер ведра с маркерами (байт)	Чувствительность к потерям пакетов (пакеты)
Максимальная скорость передачи (байт/сек.)	Минимальная задержка (мксек.)
	Максимальное различие в задержке (мксек.)
	Качество гарантии

Управление перегрузками в сетях с виртуальными каналами

До сих пор мы рассматривали методы управления перегрузками, основанные на открытом контуре. Это значит, что данные методы скорее стараются предотвратить появление перегрузок, чем, обнаружив перегрузку, принять меры к ее устранению. Здесь мы рассмотрим только один метод устранения уже возникшей перегрузки в сетях с виртуальными каналами - динамический контроль доступа. В последующих двух разделах мы рассмотрим приемы для управления перегрузками, применимые в любых СПД-средах. Прием, который широко используется, чтобы сдержать уже возникшую перегрузку и не дать положению ухудшиться - это контроль на входе. Идея очень проста - если обнаружена перегрузка, то все, что способствует увеличению трафика, запрещено. Прежде всего, запрещается создание новых виртуальных соединений. Таким образом, запрещается создание новых соединений на транспортном уровне. Этот метод хотя и грубоват, но прост в реализации и хорошо апробирован в телефонных сетях. Другой подход разрешает установку новых виртуальных соединений, но только при наличии не перегруженных маршрутов, т.е. таких маршрутов, которые не пересекаются с перегруженными участками, даже если такие маршруты далеко не

оптимальны. Третий подход уже упоминался: хост и транспортная среда договариваются перед установкой виртуального соединения о форме трафика, объеме передаваемых данных, качестве сервиса и т.п. После этого транспортная среда резервирует необходимое количество ресурсов, необходимых ей для выполнения этих соглашений. Это резервирование может происходить постоянно, а может быть сделано только при возникновении перегрузок. Плата за резервирование - неоптимальное использование пропускной способности каналов.

Подавляющие пакеты

Теперь рассмотрим приемы, используемые как в средах с виртуальными каналами, так и в средах с дейтаграммами. Каждый маршрутизатор может контролировать степень загрузки своих выходных линий и другие ресурсы. Например, он может периодически вычислять степень загруженности своих выходных линий. Всякий раз, когда степень загруженности при очередном вычислении оказывается выше некоторого порога, эта линия переводится в состояние предупреждения. Каждый пакет, маршрутизируемый через такую линию, вызывает генерацию подавляющего пакета, направляемого отправителю маршрутизируемого пакета. При этом в пакете отправителя проставляется определенный разряд, предотвращающий генерацию подавляющих пакетов другими маршрутизаторами в дальнейшем. Когда отправитель получает подавляющий пакет, он сокращает интенсивность своего трафика на определенную величину. Поскольку пакеты, направляемые одному и тому же получателю, могут маршрутизироваться по-разному, то отправитель вправе ожидать несколько подавляющих пакетов. В течение определенного времени отправитель будет игнорировать подавляющие пакеты, поступающие с направления получателя. По истечении этого периода времени отправитель ожидает появления подавляющих пакетов в течение следующего интервала. Если появился хоть один подавляющий пакет, то линия перегружена и отправитель ждет. Если в течение очередного интервала не поступило ни одного подавляющего пакета, то отправитель может увеличить интенсивность трафика. Существует много вариантов этого алгоритма. Так, например, можно использовать не только загруженность линии, но и длину очереди, заполненность буфера и параметры спецификации трафика. У методов на основе подавляющих пакетов есть один недостаток. Если есть несколько отправителей, работающих через одну и ту же выходную линию, то при определенных условиях маршрутизатор всем им пошлет подавляющие пакеты. Однако, так как отправители независимы, то один, например, сократит трафик значительно, тогда как другие лишь незначительно. Это приведет к несправедливому использованию пропускной способности канала между ними. Для предотвращения такой ситуации был предложен алгоритм справедливого чередования. Суть его состоит в том, что для каждого отправителя у выходной линии строится своя очередь. Отправка пакетов из этих очередей происходит по кругу. Поэтому, если кто-то из отправителей незначительно сократит трафик, то это лишь увеличит скорость роста его очереди. И у этого алгоритма есть недостаток: если один отправитель использует длинные пакеты, а другой - короткие, то последний получит меньшую долю пропускной способности линии. Для борьбы с этой несправедливостью в алгоритм обслуживания очередей вносят модификацию: пакеты из очередей передаются побайтно, а не весь пакет сразу. Другие модификации алгоритма чередования связаны с установкой приоритетов между очередями, что дает большую гибкость в обслуживании отправителей. Так, если есть очереди для сервера и для клиента, то естественно обслуживать очередь сервера быстрее. Представленные здесь алгоритмы с подавляющими пакетами плохо работают в высокоскоростных сетях и на больших расстояниях. Дело в том, что пока подавляющий пакет дойдет до отправителя, пройдет много времени и отправитель успеет «напихать» в сеть много пакетов. Для исправления этой ситуации была предложена его модификация – алгоритм с подавлением по скачкам. Суть его в том, что как только обнаружится перегрузка на выходной линии и маршрутизатор отправит подавляющий пакет, то ближайший маршрутизатор, получивший этот подавляющий пакет, сократит трафик к маршрутизатору, пославшему подавляющий пакет. И так скачок за скачком трафик будет быстро падать. Естественно, этот прием увеличит нагрузку на буфера маршрутизаторов, сокращающих трафик, но обеспечит быструю реакцию на возникающую перегрузку. На рисунке 5-29 дан пример алгоритма с подавлением скачками. На этом рисунке хорошо видно, что, применяя технику подавляющих пакетов «в лоб», мы достигнем сокращения нагрузки за 7 скачков (a). Техника подавляющих пакетов по скачкам позволяет добиться того же самого эффекта за пять скачков (b).

Сброс нагрузки

Когда ни один из упомянутых выше приемов не срабатывает, маршрутизатор может применить «тяжелую артиллерию» – сброс нагрузки. Было бы слишком примитивно предполагать, что маршрутизатор, при возникновении перегрузки просто начинает сбрасывать пакеты. Идея метода и его название пришли из области передачи электроэнергии. Там при возникновении перегрузок в сети, начинают отключать отдельные группы потребителей, чтобы сохранить работоспособность системы.

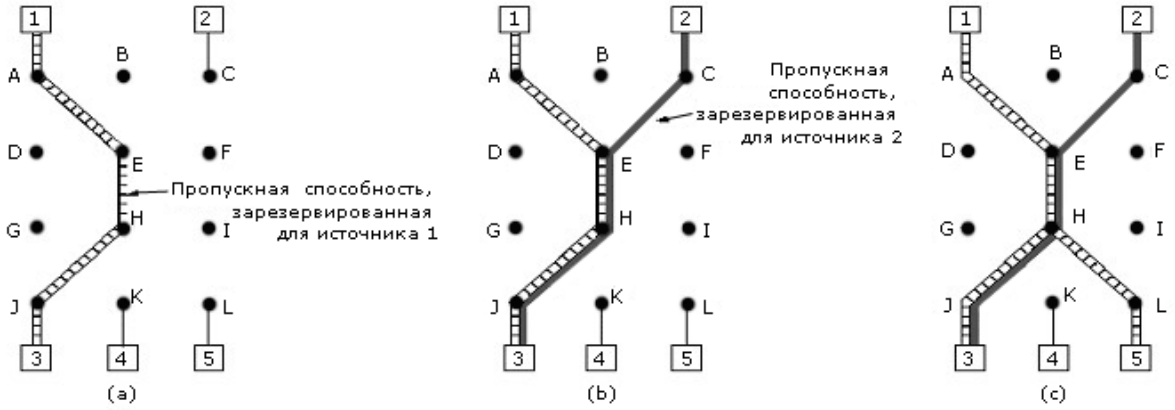
Маршрутизатор может сбрасывать пакеты, исходя из информации о приложении, пославшем эти пакеты. Если, например, передается файл, то старые пакеты, т.е. расположенные ближе к концу файла, сбрасывать лучше, поскольку приложение может потребовать перепослать все пакеты, начиная с пропущенного пакета. В этом случае, чем старее пакет, тем меньше придется перепосылать. Есть приложения, для которых все наоборот. Лучше сбрасывать новые пакеты, т.е. ближе к началу передачи, чем старые. В общем случае подход на основе сброса нагрузки предполагает определенное взаимодействие между приложением и маршрутизатором. Например, при передаче изображений в целях компрессии сначала посылают всю картинку, а потом лишь ее изменения. Ясно, что потеря одного из изменений лишь ухудшит изображение на некоторое время, в то время, как потеря картинки будет означать потерю изображения вовсе. Для обеспечения такого взаимодействия с приложением вводят приоритеты среди пакетов. Это позволяет маршрутизатору минимизировать потери для приложения, когда маршрутизатор вынужден сбрасывать пакеты. Для того, чтобы приложение не злоупотребляло приоритетными пакетами, можно увязать приоритет пакетов с величиной оплаты за трафик. Чем больше приоритетных пакетов, тем выше стоимость передачи. Приоритеты можно использовать также в целях формирования трафика. Например, при использовании алгоритма ведра с маркерами, если пакет пришел, а маркеров нет, можно применить прием, когда пакет все же будет передан, но низким приоритетом. Приоритеты используются, например, в протоколах Frame Relay. Пока величина трафика находится в заранее оговоренных пределах, все пакеты идут с определенным приоритетом. При пиковых нагрузках, превосходящих номинальную величину, приоритеты пакетов начинают падать в зависимости от времени. Если увеличение размера трафика продолжается недопустимо долго, то вскоре приоритеты пакетов достигнут минимума, и при первых же признаках перегрузки их начнут сбрасывать.

Управление перегрузками при групповой передаче

Все алгоритмы управления перегрузками, которые мы до сих пор рассматривали, относились к случаю, когда был один источник и один получатель. В этом разделе мы рассмотрим случай управления нагрузкой при групповой передаче, т.е. когда есть несколько источников и несколько получателей. Примером могут служить системы кабельного телевидения. Например, в системе может быть несколько источников телепередач (станций вещания), и зрители могут по желанию переключаться с одной станции на другую. Аналогичная ситуация может иметь место в системе видеоконференций, когда слушатели могут переключаться с одной конференции на другую. Во многих подобных приложениях группы могут возникать и изменяться по составу динамически. В этих условиях прием, когда источник сообщений резервирует заранее необходимые для передачи ресурсы, не работает эффективно. Источнику сообщений придется при каждом изменении группы генерировать дерево связей. В случае кабельного телевидения, когда группы содержат миллионы зрителей, такой подход не годится. Одно из возможных решений для подобных случаев было предложено в 1993 году – это RSVP-протокол (Resource reSerVation Protocol – протокол резервирования ресурсов). Он позволяет нескольким отправителям передавать сообщения группам получателей, отдельным получателям переходить из группы в группу, оптимизировать использование пропускной способности каналов, избегая перегрузок. В простейшей форме этот протокол для групповой маршрутизации использует дерево связей так, как мы уже рассматривали ранее. Каждой группе приписана группа адресов. При отправке пакета отправитель помещает в него весь список адресов группы. После этого стандартный алгоритм групповой маршрутизации строит дерево связей, покрывающее все адреса группы. Собственно маршрутизация не является частью RSVP. Чтобы понять действие алгоритма RSVP, рассмотрим пример. На рисунке 5-30 (a) показана сеть. Пусть 1 и 2 – групповые отправители, а 3, 4 и 5 – групповые получатели. То, что множество отправителей и получателей не пересекаются, сделано для простоты. Дерево связей для групповой рассылке от 1 дано на рисунке 5-30 (b), а от 2 – на рисунке 5-30 (c).

Чтобы избежать перегрузок, любой получатель в группе шлет надлежащему отправителю резервирующее сообщение. Это сообщение с помощью алгоритма пересылки вдоль обратного пути, рассмотренного в разделе 5.2.9, движется к отправителю и вызывает резервирование необходимой пропускной способности на каждом узле, через который оно проходит. Если при прохождении очередного узла ему не удается зарезервировать необходимую пропускную способность, то получателю направляется отказ в установлении соединения.

Рисунок 5-31. Управление перегрузками при групповой передаче



На рисунке 5-31 (a) показан путь резервирования между получателем 3 и отправителем 1. Как только канал между ними установлен, получатель 3 может получать поток пакетов от 1. Рассмотрим теперь, что произойдет, если получатель 3 захочет также получать данные от отправителя 2. Для этого резервирующее сообщение проложит второй путь, показанный на рисунке 5-31 (b). Предположим теперь, что получатель 5 также захотел подключиться к отправителю 2 (рисунок 5-31 (c)). Он запустит резервирующее сообщение, которое свободно пройдет до узла H, а там будет обнаружено, что ранее уже были зарезервированы ресурсы для доставки трафика от узла 2 до узла H. Здесь возможны нюансы. Например, если требования к качеству сервиса у 5 и 3 разное, то резервироваться ресурсы будут по максимуму. Кроме этого, при резервировании получатель может указывать не только несколько источников, от которых он хотел бы получать информацию, но также то, является ли создаваемый канал временным (вплоть до того, на какое время он создается), или он должен долго оставаться открытым. Маршрутизаторы могут использовать эту информацию для оптимизации планирования ресурсов, в особенности при разделении каких-либо ресурсов между несколькими получателями. Благодаря такой стратегии этот протокол может поддерживать динамику внутри групп.

Сетевой уровень: проблемы построения сетевого уровня (Сервис, внутренняя организация сетевого уровня). Межсетевое взаимодействие (соединение виртуальных каналов, межсетевая передача без соединений, тунелирование, межсетевая маршрутизация, фрагментация, Firewall).

Теперь мы перейдем к рассмотрению случаев, когда соединение возникает между СПД-средами с разной архитектурой. Существование разных сетей - явление временное, или оно отражает некоторые объективные тенденции? Хотя Unix работает с TCP/IP, тем не менее, существует SNA от IBM, NCP/IPX от Novel, AppleTalk и т.д. Безусловно, будут появляться новые технологии, стандарты, устройства и программное обеспечение, их реализующее. Все эти инновации надо будет как-то связывать с существующими сетями.

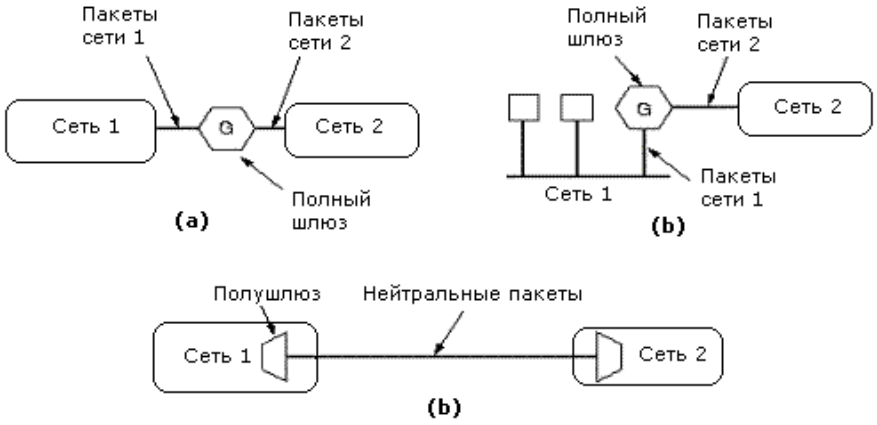
Название средства, соединяющего сети между собой, зависит от того, на каком уровне это происходит.

1. Уровень 1: репитер копирует биты из одного кабельного сегмента в другой.
2. Уровень 2: мост передает пакеты канального уровня из одной ЛВС в другую.
3. Уровень 3: мультипротокольный маршрутизатор передает пакеты между сетями с разной архитектурой.
4. Уровень 4: транспортный шлюз соединяет байтовые потоки на транспортном уровне.
5. Над уровнем 4: прикладной шлюз соединяет приложения в разных сетях.

Напомним, что термин «шлюз» мы используем для обозначения устройства, соединяющего сети с разной архитектурой. Репитер - устройство, обеспечивающее усиление и очистку сигнала. На MAC-уровне трансивер обеспечивает передачу сигнала в пределах 500 метров. Репитер обеспечивает передачу на 2,5 км. Мост способен хранить и маршрутизировать пакеты на канальном уровне. Он получает канальный пакет целиком и решает, по какой линии его передать дальше. Мультипротокольные маршрутизаторы функционально примерно то же самое, что и мосты, но работают на сетевом уровне. Они получают пакеты сетевого уровня и определяют, куда их передать. Однако при этом разные каналы могут принадлежать разным сетям, использующим разные протокольные стеки. Поэтому мультипротокольному маршрутизатору, кроме задачи маршрутизации, приходится решать и задачу сопряжения форматов пакетов на сетевом уровне в сетях с разной архитектурой.

На рисунке 5-33 показаны разные схемы включения шлюза.

Рисунок 5-33. Схемы включения шлюза: (a) Полный шлюз между двумя WAN; (b) Полный шлюз между LAN и WAN; (c) Два полушлюза



Чем различаются сети

В таблице 5-34 перечислены основные различия, которые могут встречаться на сетевом уровне. Эти различия усложняют межсетевое взаимодействие.

Таблица 5-34. Различия сетей на сетевом уровне

Признак	Возможные значения
Предоставляемый сервис	Ориентированные vs. неориентированные на соединение
Протоколы	IP, IPX, CLNP, AppleTalk, DECnet и т.д.
Адресация	Прямая (802) vs. иерархическая
Групповое вещание	Есть/нет (также транслирование)
Размер пакетов	У каждой сети свой максимальный размер
Качество сервиса	Есть/нет; много различных видов
Действия в случае ошибок	Надежная, упорядоченная или неупорядоченная доставка
Управление потоком	«Скользящее окно», регулирование скорости, другие способы или неуправляемый поток
Управление перегрузками	Протокол «текущего ведра», подавляющие пакеты и т.д.
Безопасность	Правила защиты информации, шифрование и т.д.
Параметры	Разные значения тайм-аута, характеристики потока и т.д.
Оплата	За время соединения, за количество пакетов, побайтно или без оплаты

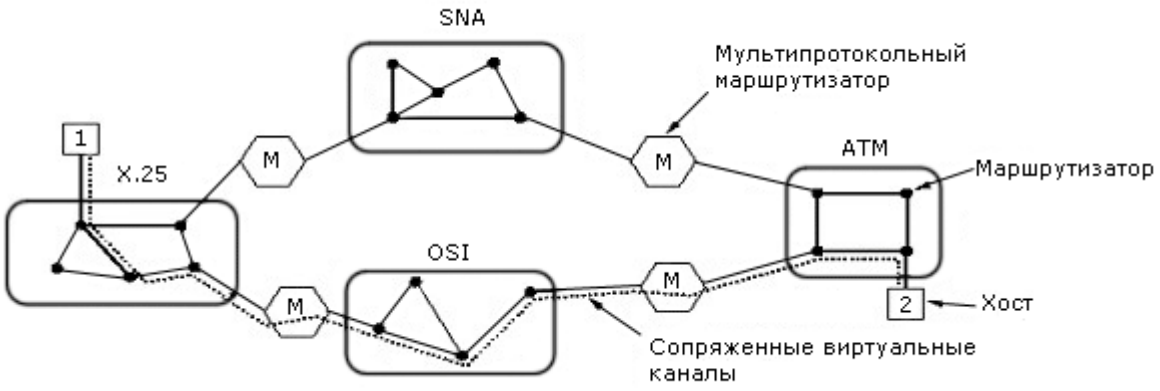
Когда пакет, отправленный из одной сети, должен пройти несколько разных сетей, прежде чем достичь нужную сеть, приходится решать множество проблем, возникающих на интерфейсах между сетями. Например, если пакеты были отправлены по виртуальному соединению, а одна из транзитных сетей не

поддерживает такие соединения, то порядок следования пакетов может быть нарушен, к чему ни отправитель, ни получатель могут быть не готовы. Также может потребоваться согласование функциональности протоколов на одноименных уровнях. Если функциональность одного протокола не может быть выражена в терминах функциональности другого, то такое согласование может оказаться невозможным. Проблемы могут возникнуть с адресацией при переходе из одной сети в другую, групповой рассылкой и т.д. Много проблем доставляет различие в максимальной длине пакетов, используемой в разных сетях. Другим источником трудностей является различие в качестве услуг в разных сетях. Например, когда надо передать пакет из сети, поддерживающей ограничение на время передачи пакета, через сеть, где нет никаких гарантий на максимальную величину задержки пакета при передаче. В разных сетях по-разному решаются вопросы управления перегрузками и потоками, обнаружения и исправления ошибок, что также может служить источником проблем.

5.4.2. Сопряжение виртуальных каналов

Есть два общих приема для межсетевого взаимодействия: сопряжение, ориентированное на соединения подсетей с виртуальными каналами, и взаимодействие подсетей через дейтаграммы. На рисунке 5-35 показана модель сопряжения виртуальных каналов. Абонентская машина одной сети устанавливает виртуальное соединение не только внутри своей сети, но и в другой сети, вплоть до получателя. Внутри своей сети соединение прокладывается по правилам этой сети вплоть до мультипротокольного маршрутизатора, ближайшего к сети получателя. Затем от этого маршрутизатора до получателя - по правилам сети получателя. (Рассмотреть прохождение пакетов вдоль соединения.)

Рисунок 5-35. Межсетевое взаимодействие с использованием сопряжения виртуальных каналов



На рисунке показано решение с использованием полного шлюза. Однако такое же решение возможно и с полушлюзом. Это решение хорошо работает для сетей с примерно одинаковыми характеристиками.

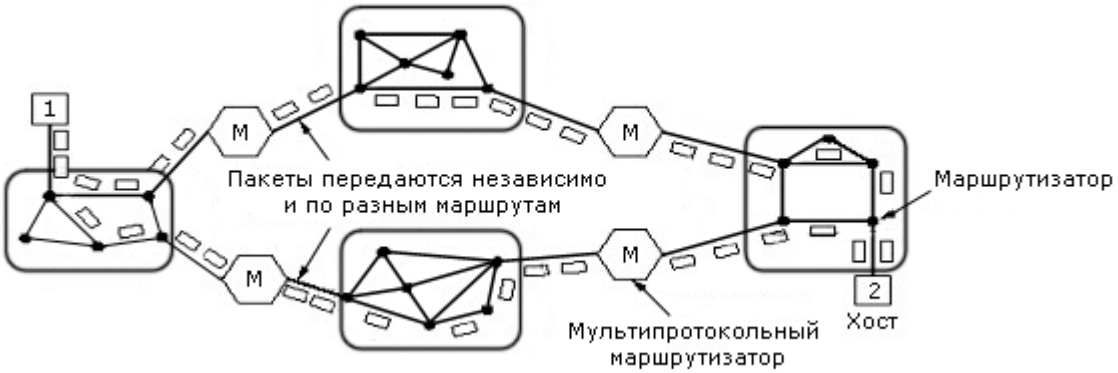
Достоинства сопряжения виртуальных каналов: буферы можно резервировать заранее, порядок пакетов сохраняется, проще управлять повторной передачей из-за задержки, короткие заголовки пакетов.

Недостатки сопряжения виртуальных каналов: хранение таблицы сопряжения, сложности в изменении маршрута при перегрузках, требование высокой надежности маршрутизаторов вдоль сопряжения.

5.4.3. Межсетевое взаимодействие без соединений

На рисунке 5-36 показано решение на основе соединения сетей на уровне дейтаграмм. При этом подходе единственный сервис, который сетевой уровень предоставляет транспортному – «впрыскивание» дейтаграмм в транспортную среду. Дальше приходиться надеяться на удачу. Такое сопряжение сетей возможно, если соединяемые транспортные среды используют одни и те же или очень близкие сетевые протоколы. Вспомним проблемы мостов между подуровнями 802.x.

Рисунок 5-36. Межсетевое взаимодействие без соединений



Проблемы возникают с адресацией. Различия в адресации могут быть столь велики, что сопряжение станет невозможным. Например, в TCP/IP используется 32-разрядный адрес, а в OSI - десятичный номер, подобный телефонному. Выход - распространять каждую адресацию на все машины в мире. Однако, очевидно, что такой подход не работает.

Другой выход - создать универсальный пакет, который понимали бы разные сети, - тоже не работает. Всех уговорить признать один формат как универсальный невозможно.

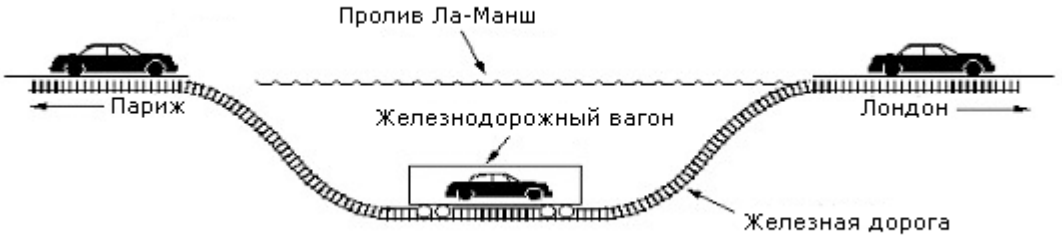
Основное достоинство дейтаграммного подхода - он может использоваться между сетями, которые не поддерживают виртуальных соединений. Число таких сетей весьма велико.

5.4.4. Туннелирование

В общем случае проблема межсетевого соединения весьма сложна. Однако есть достаточно распространенный случай, для которого есть решение. Это соединение двух одинаковых сетей через третью. Например, так, как показано на рисунке 5-37. Решение проблемы межсетевого соединения в этом случае дает применение техники туннелирования. Рисунок 5-38 разъясняет прием туннелирования на примере автомобиля. Суть этого приема состоит в том, что пакет из одной сети упаковывается в кадр промежуточной сети. Затем он передается через промежуточную сеть на канальном уровне. При достижении кадром сети назначения кадр распаковывается, пакет передается на сетевой уровень и движется дальше.

Рисунок 5-37. Транспортировка пакета методом туннелирования

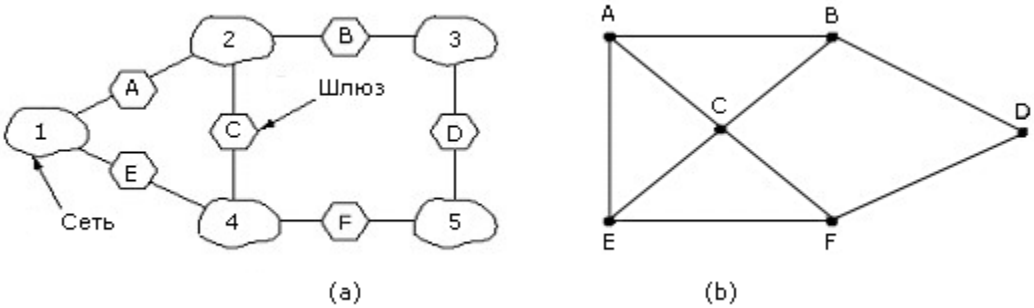
Рисунок 5-38. Транспортировка автомобиля через туннель под Ла-Маншем



5.4.5. Межсетевая маршрутизация

Маршрутизация на межсетевом уровне происходит примерно так же, как на сетевом, но с некоторыми дополнительными сложностями. Рассмотрим пример на рисунке 5-39. На этом рисунке шесть мультипротокольных маршрутизаторов соединяют пять сетей.

Рисунок 5-39. Пример межсетевой маршрутизации



Имея граф соединений этих маршрутизаторов между собой, можно применять уже известные нам алгоритмы маршрутизации: по вектору расстояния или по состоянию канала. Так мы приходим к двум уровням маршрутизации: внутреннему межшлюзовому протоколу и внешнему межшлюзовому протоколу. Поскольку каждая сеть в определенном смысле автономна, то для нее часто используют термин - автономная система.

Главная сложность, отличающая внутрисетевую маршрутизацию от межсетевой - государственные границы. Здесь возникают различия в законах разных стран, различия в оплате трафика, принятые на территориях разных стран, и т.д.

5.4.6. Фрагментация

В каждой сети есть свой максимальный размер пакетов. Это ограничение имеет несколько причин:

1. Аппаратура (например, максимальный TDM-слот)
2. Операционная система (все буфера по 512 байтов)
3. Протоколы (например, размер поля длины пакета)
4. Совместимость с некоторыми национальными и международными стандартами
5. Стремление сократить ошибку, вызываемую повторной передачей
6. Желание предотвратить длительный захват канала одним пакетом

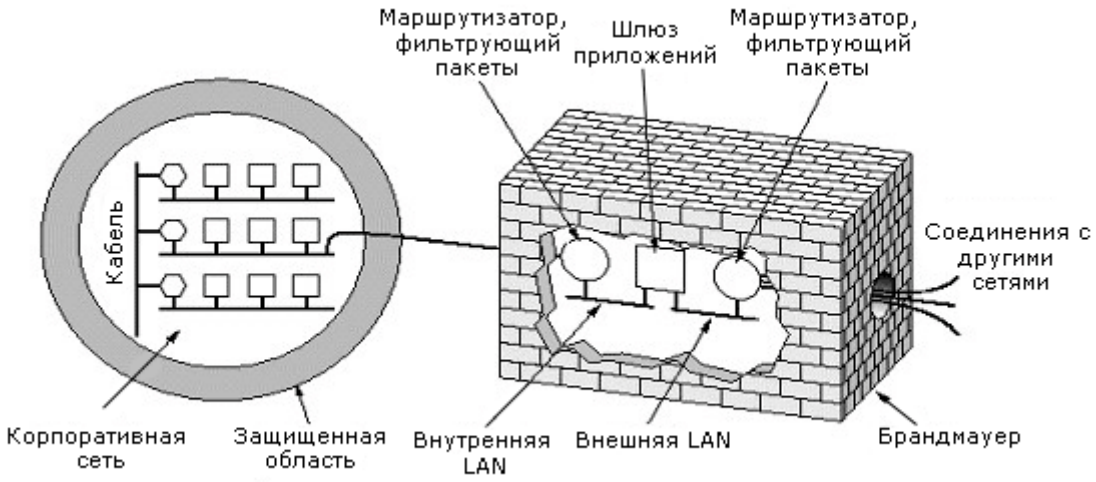
Максимальный размер пакета колеблется от 48 байтов в ATM-сети до 65515 байтов в IP-сети (у протоколов более высоких уровней он еще больше). Очевидно, первая же проблема возникает при попытке передать большой пакет через сеть, у которой максимальный размер пакета меньше. Одно из решений - проложить маршрут для таких пакетов так, чтобы избежать подобной ситуации. Однако что делать, в такой сети расположен получатель? Единственное решение - разрешить шлюзу разбивать пакет на фрагменты и отправлять каждый фрагмент независимо. В этом случае возникает проблема сборки фрагментов. Есть два

подхода к тому, как это осуществлять. Первый - делать фрагменты столь малыми, что любая сеть на их пути будет прозрачна для них. Когда поступает большой пакет, его разбивают на малые и все пакеты отправляют на один и тот же выходной шлюз, где они собираются снова в большой пакет. В такой фрагментации есть трудности: как узнать, что все фрагменты достигли выходного шлюза, как выбирать маршрут для фрагментов, накладные расходы на разбиение и сборку пакета из фрагментов. Другой подход - разбив пакет на фрагменты, рассматривать каждый из них как обычный пакет. Сборка фрагментов происходит только в узле назначения. Однако при таком подходе каждый хост должен уметь собирать пакеты из фрагментов.

Firewall

Способность любого компьютера соединяться, где бы он ни был, с любым другим компьютером - благо для пользователя, но сущее наказание для службы безопасности любой организации. Здесь, кроме угрозы потери информации, есть угроза притока всякой гадости типа вирусов, червей и прочих цифровых паразитов. Позднее мы о них поговорим подробнее. Надо заметить, что, согласно результатам исследований, 50% опасности таится вне сети, а 50% - внутри, среди сотрудников. Итак, нужен механизм, который бы различал «чистые» биты от «нечистых». Один способ - шифровать данные. Так поступают при передаче данных. Со способами шифрования мы познакомимся позднее. Но шифрование бессильно против вирусов, хакеров и прочей нечисти. Одним из средств борьбы с ними служат брандмауеры (firewall). Барьер - современная форма крепостного рва. Компания может иметь сколь угодно сложную сеть, объединяющую много локальных сетей. Однако весь трафик в сеть и из этой сети направляется только через один шлюз, где происходит проверка пакета на соответствие определенным требованиям. Если пакет не удовлетворяет этим требованиям, то он не допускается в сеть или из нее. На рисунке 5-42 показана организация брандмауера.

Рисунок 5-42. Устройство брандмауера



Барьер состоит из двух маршрутизаторов, фильтрующих пакеты, и шлюза приложений. Фильтры содержат таблицы сайтов, от которых можно принимать и которым можно передавать пакеты. Шлюз приложений ориентирован на конкретные приложения. Пример - шлюз для электронной почты. Этот шлюз анализирует поле данных и принимает решение, сбросить пакет или нет. Набор таких шлюзов полностью зависит от политики информационной безопасности конкретной организации. Чаще всего это шлюз электронной почты и WWW.

Сетевой уровень в Интернет: IP, IPv6, адресация, протоколы ARP, RARP

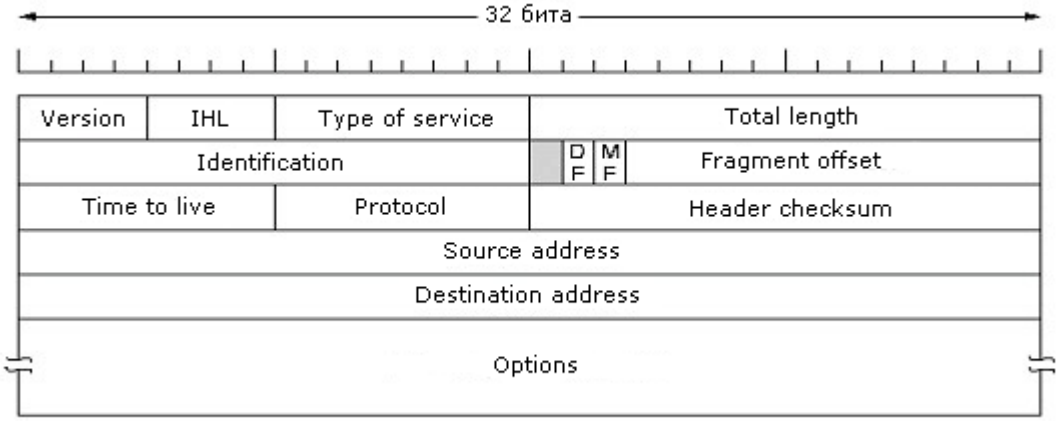
Интернет представляет собой объединение подсетей, которые называются автономными системами. Автономные системы – это подсеть, охватывающая единую территорию, находящаяся под единым административным управлением и имеющая единую политику маршрутизации по отношению ко всем остальным сетям. В Интернете нет какой-либо регулярной, специально предусмотренной структуры подсетей. Он образован из соединения большого числа подсетей, среди которых можно выделить несколько

остовых (backbone). К этим остовым сетям подключены региональные сети, к которым подключены локальные сети организаций. На рисунке 5-43 показана схема соединения таких остовых сетей. Соединяет все автономные системы вместе IP-протокол. В отличие от других протоколов сетевого уровня, этот протокол с самого начала создавался для объединения сетей. Его целью было наилучшим образом передавать дейтаграммы от одной машины к другой, где бы эти машины ни находились. Как мы уже отмечали, подсеть в Интернете реализует сервис без соединений и работает следующим образом. Транспортный уровень получает поток данных и делит их на дейтаграммы. Дейтаграммы могут быть от 64К до 1500 байт. Они передаются через подсети в Интернет и, если необходимо, делятся на более короткие. Когда все дейтаграммы достигают места назначения, они собираются в исходные дейтаграммы на сетевом уровне и передаются на транспортный уровень, где и восстанавливается исходный поток данных.

IP-протокол

На рисунке 5-44 показан заголовок IP-пакета (дейтаграммы). Он имеет обязательную часть размером 20 байт и может быть расширен до 60. Дейтаграмму передают, начиная с поля Version.

Рисунок 5-44. Заголовок IP



- Version - указывает версию протокола.
- IHL - длина заголовка в 32-разрядных словах (минимум 5, максимум 15, что соответствует 60 байтам).
- Type of service - вид необходимого сервиса. Здесь возможны различные комбинации скорости и надежности: например, передача голоса, аккуратная доставка строки битов, файла и т.п.
- Identification - позволяет отличать фрагменты одной и той же дейтаграммы.
- DF – признак управления фрагментацией. Если он равен 1, то фрагментация невозможна.
- Total length - указывает общую длину дейтаграммы, включая заголовок и поле данных. Максимальная длина 65535 байт.
- Identification – указывает, какой дейтаграмме принадлежит очередной поступивший фрагмент. Все фрагменты одной дейтаграммы имеют в этом поле одно и то же значение.
- MF – содержит единицу только у последнего фрагмента дейтаграммы. Это поле позволяет отличить последний фрагмент от всех остальных.
- Fragment offset – указывает, где в дейтаграмме располагается данный фрагмент. Длина всех фрагментов, кроме последнего, должна быть кратна 8 байтам. Поскольку поле имеет 13 разрядов, то на одну дейтаграмму максимально может быть 8192 фрагментов.
- Time to live – время жизни пакета. Максимальное значение этого поля - 255, измеряется в секундах. Очень часто здесь используется счетчик скачков.
- Protocol - показывает, какому процессу на транспортном уровне передать собранную дейтаграмму (TCP, UDP и т.д.).
- Header checksum - контрольная сумма, которая охватывает только заголовок.
- Source address, Destination address – идентифицируют машину отправителя и получателя в сети.
- Options - предусмотрено для расширения возможностей протокола. Оно делится, в свою очередь, на следующие поля:

- Security – указывает уровень секретности передаваемой информации. Маршрутизатор может на основании значения этого поля запретить определенные маршруты, например, если они пролегают через ненадежные регионы.
- Strict source routing – здесь указан полный маршрут в виде списка IP-адресов. Это поле используется в алгоритме маршрутизации от источника, а также в критических ситуациях, например, когда таблица маршрутизации по какой-то причине оказалась испорченной.
- Loose source routing – список маршрутизаторов, через которые фрагмент обязан пройти. Он может пройти и через другие маршрутизаторы, но данные обязательно должны принадлежать его маршруту.
- Record route – указывает маршрутизаторам на необходимость заносить в поле свои адреса. Это позволяет проследить, как шел фрагмент.
- Time stamp – вместе с предыдущим полем указывает маршрутизаторам на необходимость записывать не только свои адреса, но и время, когда фрагмент проходил через них. Это поле очень полезно при отладке алгоритмов маршрутизации.

5.5.2. IP-адресация

Каждая машина в Интернете имеет уникальный IP-адрес. Он состоит из адреса сети и адреса машины в этой сети. Все IP-адреса имеют длину 32 разряда. На рисунке 5-45 показаны форматы IP-адресов. Если машина подключена к нескольким сетям, то в каждой сети у нее будет свой IP-адрес.

Рисунок 5-45. Форматы IP-адресов

Все адреса разделяются на классы. Всего есть пять классов адресов: A, B, C, D, E. Классы A позволяет адресовать до 126 сетей по 16 миллионов машин в каждой, B - 16382 сетей по 64K машин, C - 2 миллиона сетей по 256 машин, D - предназначены для групповой передачи, E - зарезервированы для развития. Адреса выделяет только организация NIC - Network Information Center. Несколько адресов, показанных на рисунке 5-46, имеют специальное назначение. Адрес из одних нулей используется при загрузке машины.

5.5.3. Подсети

Все машины одной сети должны иметь одинаковый номер сети в своем адресе. Это приводит к целому ряду проблем. По мере роста сети приходиться менять класс адреса. Появление новых адресов приводит к проблеме модификации таблиц маршрутизации и распространению информации о новых адресах повсюду. Перенос машины из одной сети в другую требует изменения маршрутизации. Эти изменения происходят не сразу, и пока они не будут выполнены, все сообщения пойдут по старому адресу.

Решением этих проблем является разделение сети на части, которые извне видны как единое целое, но внутри каждая часть имеет свой адрес. Эти части называются подсети. Мы уже отмечали в главе 1, что термин подсеть в Интернете имеет особый смысл. Итак, подсеть - это часть сети, невидимая извне. Изменение адреса подсети или введение новой подсети не требует обращения в NIC или изменений какой-либо глобальной

базы данных. Чтобы понять, как используется адрес подсети, надо проследить, как маршрутизатор использует IP-адрес. Есть две таблицы: некоторое количество IP-адресов вида «сеть, 0» и некоторое количество IP-адресов вида «эта_сеть, адрес машины». Первая показывает, как достичь интересующей сети. Вторая - как достичь узла внутри сети. Когда поступает IP-пакет, маршрутизатор ищет его адрес доставки в таблице маршрутизации. Если этот адрес – адрес другой сети, то пакет передают дальше тому маршрутизатору, который отвечает за связь с этой сетью. Если это адрес в локальной сети, то маршрутизатор направляет пакет прямо по месту назначения. Если адреса нет в таблице, то маршрутизатор направляет пакет специально выделенному по умолчанию маршрутизатору, который должен разобраться с этим случаем с помощью более подробной таблицы. Из этого описания видно, что алгоритм маршрутизации имеет дело только с сетями или локальными машинами, а не с парами «сеть, узел». Такая организация алгоритма позволяет существенно сократить размер таблиц в маршрутизаторах. С появлением подсети структура адресов меняется. Теперь записи в таблице имеют форму «эта_сеть, подсеть, 0» и «эта_сеть, эта_подсеть, машина». Таким образом, маршрутизатор подсети в данной локальной сети знает, как достичь любой подсети в данной локальной сети и как найти конкретную машину в своей подсети. Все что ему нужно – это знать маску подсети. С помощью логической операции «&» маршрутизатор выделяет адрес подсети с помощью маски, показанной на рисунке 5-47. По своим таблицам он определяет, как достичь нужной подсети или (если это локальная подсеть данного маршрутизатора) как достичь конкретной машины.

5.5.4. Протоколы управления межсетевым взаимодействием

В Интернете, кроме IP-протокола, который используется для передачи данных, есть несколько протоколов управления, используемых на сетевом уровне, таких как ICMP, ARP, RARP, BOOTP, которые мы рассмотрим последовательно.

5.5.4.1. Internet Control Message Protocol

Управление функционированием Интернета происходит через маршрутизаторы с помощью протокола ICMP (RFC 792). Этот протокол обеспечивает доставку сообщений любой машине, имеющей IP-адрес (хосту), от маршрутизаторов и других хостов в сети. Этот протокол обеспечивает обратную связь при возникновении проблем при передаче. Протокол ICMP использует протокол IP и доставка его дейтаграмм не более надежна, чем любой IP-дейтаграммы в сети. Сообщение destination unreachable покрывает множество случаев: от случая, когда маршрутизатор не знает, как достигнуть нужной подсети или хоста, до случая, когда дейтаграмма при доставке должна быть фрагментирована, но установлен флаг, который запрещает это делать. Сообщение time exceeded посылает маршрутизатор, если он обнаружил дейтаграмму с истекшим времени жизни. Хост генерирует такое сообщение, если он не успел завершить сборку дейтаграммы до истечения времени ее жизни. Синтаксические или семантические ошибки в заголовке IP-дейтаграммы вызывают появление сообщения parameter problem. Сообщение source quench обеспечивает средство управления потоком. Маршрутизатор или хост-получатель высылает этот пакет хосту-отправителю, если необходимо понизить скорость передачи. Сообщения этого типа будут генерироваться до тех пор, пока скорость поступления дейтаграмм от отправителя не достигнет нужной хосту-получателю величины. Это сообщение система может использовать для предотвращения перегрузки. Оно возникает всякий раз, когда маршрутизатор вынужден сбросить дейтаграмму из-за переполнения своего буфера.

Сообщение redirect позволяет маршрутизатору отправить рекомендацию о лучшем маршруте и впредь посылать дейтаграммы с определенным адресом через другой маршрутизатор. Сообщения echo request и echo reply обеспечивают механизм проверки работоспособности объектов в сети. Получатель сообщения echo request обязан ответить сообщением echo reply, причем с теми же параметрами, что и в echo request. Сообщения timestamp request и timestamp reply обеспечивают механизм для измерения и изменения параметров временной задержки в Интернете. Этот механизм необходим, например, для работы алгоритма маршрутизации по состоянию канала.

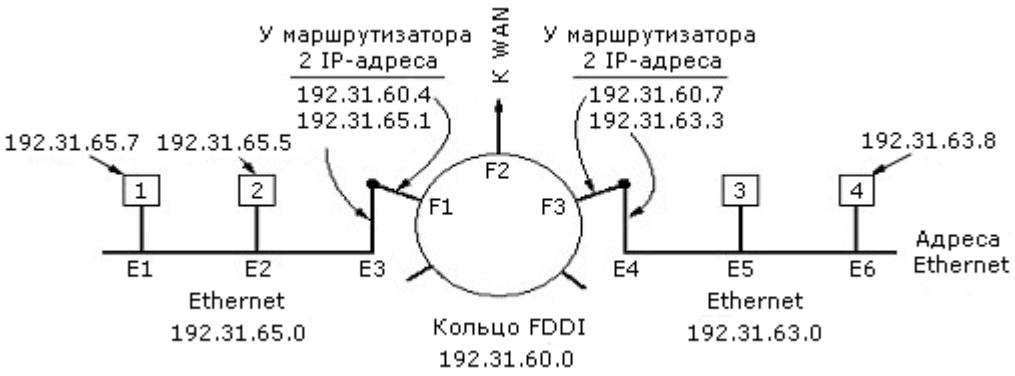
5.5.4.2. Address Resolution Protocol – протокол определения адреса

Хотя каждая машина в Интернете имеет уникальный IP адрес, и даже не один, но при передаче пакета через сеть от этого мало пользы, так как канальный уровень не понимает IP адресов. Как правило, машина

подключена к ЛВС через сетевую карту, которая понимает только ЛВС адреса канального уровня, например, Ethernet-адрес. Этот адрес имеет 48 разрядов. Сетевая карта знает только такие адреса и ничего об 32-разрядных IP.

Как отобразить 32-разрядный IP-адрес в адреса канального уровня, например, Ethernet-адрес? Для объяснения воспользуемся рисунком 5-49.

Рисунок 5-49. Три объединенных сети класса C: две Ethernet-сети и кольцо FDDI



Когда машина 1 посылает сообщение машине 2, то через DNS (Domain Name Service – службу имен домена – это приложение мы будем рассматривать в главе 7) определяется IP-адрес места назначения. Далее, для отображения IP-адреса в Ethernet-адрес, в подсеть посылается запрос, у кого такой IP-адрес. Машина с указанным адресом шлет ответ. Протокол, который реализует рассылку запросов и сбор ответов - ARP-протокол. Практически каждая машина в Интернете использует этот протокол.

Теперь рассмотрим случай, когда обращение идет в другую сеть. Здесь два решения - есть определенный маршрутизатор, который принимает все сообщения, адресованные определенной сети или группе адресов - proxy ARP. Этот маршрутизатор знает, как найти адресуемую машину. Другое решение - выделенный маршрутизатор, который управляет маршрутизацией удаленного трафика. Машина определяет, что обращение идет в удаленную сеть, и шлет сообщение на этот маршрутизатор.

5.5.4.3. Reverse Address Resolution Protocol (RARP) – обратный протокол определения адреса

Иногда возникает обратная проблема - известен Ethernet-адрес, но какой IP-адрес ему соответствует? Эта проблема возникает, например, при удаленной загрузке бездисковой станции. Как эта станция определит свой и соседние IP-адреса?

Станция посылает запрос к RARP-серверу: "Мой Ethernet-адрес такой то, кто знает соответствующий IP-адрес?" RARP-сервер отлавливает такие запросы и шлет ответ.

У этого протокола есть один существенный недостаток – пакеты с одним и тем же запросом рассылаются всем, что увеличивает накладные расходы. Для устранения этого недостатка был предложен протокол BOOTP. В отличие от RARP, BOOTP использует UDP-сообщения, которые рассылаются только маршрутизаторам. Этот протокол также используется в бездисковых станциях, у которых в памяти прошит IP-адрес выделенного маршрутизатора.

Протоколы внутренней и внешней маршрутизации (RIP, OSPF, BGP)

OSPF - внутренний протокол маршрутизации шлюзов

Интернет состоит из сетей, управляемых разными организациями. Каждая такая сеть использует внутри свои алгоритмы маршрутизации и управления и называется автономной системой. Наличие стандартов позволяет

преодолеть различия во внутренней организации автономных систем и обеспечить их совместное функционирование. Алгоритмы маршрутизации, применяемые внутри АС, называются внутренними протоколами шлюзов. Алгоритмы маршрутизации, применяемые для маршрутизации между АС, называются внешними протоколами шлюзов. Изначально в качестве внутреннего протокола шлюзов использовался протокол по вектору расстояния (RIP). Этот протокол работал хорошо, пока автономная система была небольшой. Однако по мере роста АС он начинал работать все хуже и хуже. Проблемы «счетчика до бесконечности» и медленная сходимость не получили удовлетворительного решения. В 1979 году он был замещен протоколом маршрутизации по состоянию каналов. В 1988 году инженерный комитет Internet принял решение о разработке нового алгоритма маршрутизации. Этот алгоритм, названный OSPF - Open Shortest Path First, стал стандартом в 1990 году (RFC 1247).

5.5.5.1. Требования к протоколу

На основе имеющегося опыта был составлен длинный список требований к протоколу. Прежде всего, алгоритм должен быть опубликован в открытой литературе (отсюда «open»). Во-вторых, он не должен быть собственностью какой-либо компании. В-третьих, он должен уметь работать с разными метриками: расстоянием, пропускной способностью, задержкой и т.п. Он должен быть динамическим, т.е. реагировать на изменении в топологии сети автоматически и быстро.

В-четвертых, он должен поддерживать разные виды сервиса, поддерживать маршрутизацию для трафика в реальном времени одним способом, а для других типов трафика - другим. В IP-пакете есть поле Type of service, которое не использовалось существующими в то время протоколами. В-пятых, он должен обеспечивать балансировку нагрузки и при необходимости разделять потоки по разным каналам. Все предыдущие протоколы использовали только один канал - наилучший. В-шестых, он должен поддерживать иерархию. К 1988 году Интернет стал столь большим, что ни один маршрутизатор был уже не в состоянии хранить всю топологию. Поэтому новый протокол должен быть сконструирован так, чтобы по нему мог бы работать не один маршрутизатор. В-седьмых, должна быть усилена безопасность маршрутизаторов для защиты от студентов, которые развлекались тем, что подсовывали маршрутизаторам неверную информацию о маршрутах. Наконец, надо было позаботиться о том, чтобы позволить маршрутизаторам общаться с помощью туннелирования.

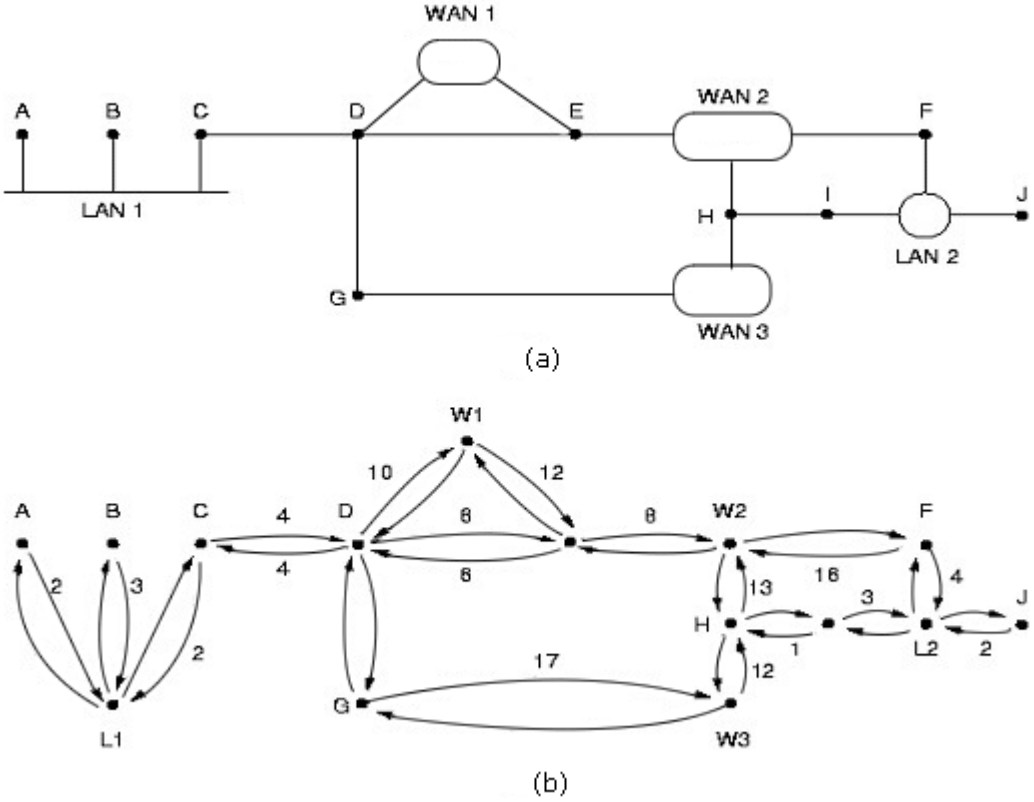
5.5.5.2. Виды соединений и сетей в OSPF

OSPF поддерживает три вида соединений и сетей:

1. Точка-точка между двумя маршрутизаторами
2. Сети с множественным доступом и вещанием (большинство ЛВС)
3. Сети с множественным доступом без вещания (например, региональные сети с коммутацией пакетов)

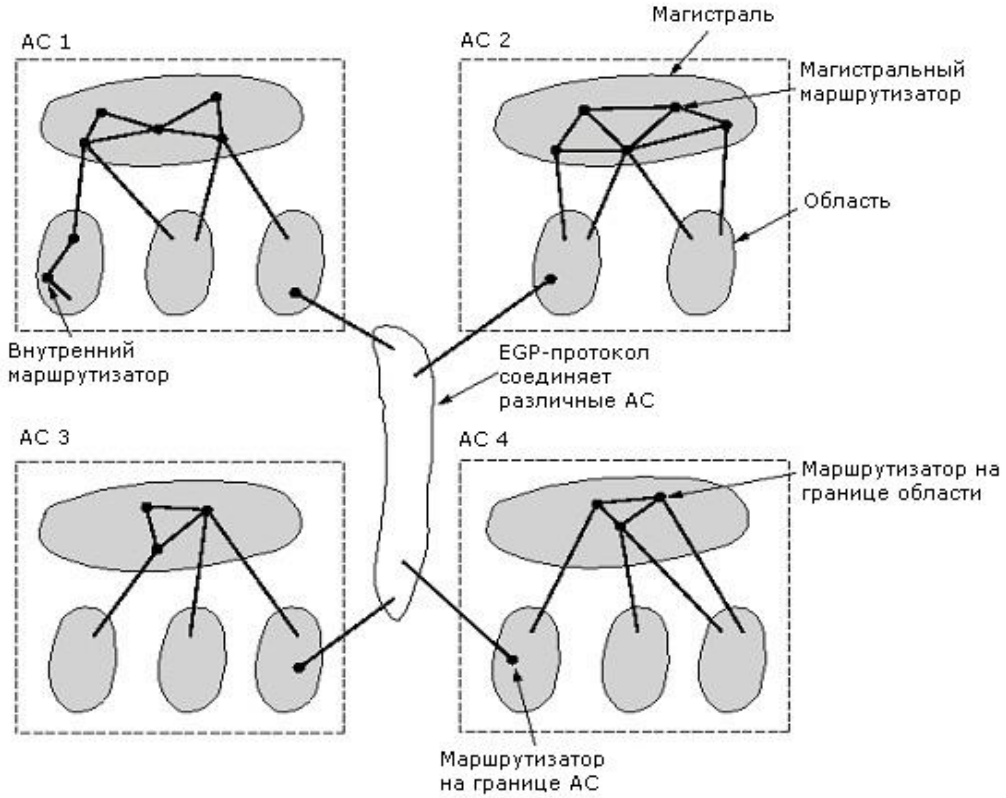
На рисунке 5-50 (a) показаны все три вида сетей. Отметим, что хосты не играют никакой роли в OSPF. OSPF абстрагируется от конкретных сетей, маршрутизаторов и хостов в форме ориентированного графа, каждая дуга в котором имеет вес, представляющий собой задержку, расстояние и т.п. В этом графе находится кратчайший путь на основе весов дуг. Последовательный канал между узлами представляют две дуги, которые могут иметь разный вес. Сеть с множественным доступом представляет узел, соединенный с маршрутизаторами этой сети дугами с весом 0, часто опускаемыми на рисунках. На рисунке 5-50 (b) показан такой граф.

Рисунок 5-50. Три вида сетей в OSPF и их представление в виде графа



Многие АС сами по себе представляют большие сети. OSPF позволяет разбивать их на области, где каждая область - это либо сеть, либо последовательность сетей. Области не пересекаются. Есть маршрутизаторы, которые не принадлежат никакой области. Область - обобщение понятия подсети. Вне области ее топология не видна. Каждая АС имеет остовую область, называемую «областью 0». Все области АС соединяются с остовой, возможно через туннелирование. Поэтому можно из одной области попасть в другую через остовую область. Туннель представлен в графе дугой с весом. Любой маршрутизатор, соединенный с двумя или более областями, - часть остовой области. Как и в других областях, топология остовой области не видна извне. Внутри области у каждого маршрутизатора одинаковая база данных состояний каналов и одинаковый алгоритм наикратчайшего пути. Задача маршрутизатора - вычислить наикратчайший путь до другого маршрутизатора этой области, включая маршрутизатор, соединенный с остовой областью. Маршрутизатор, соединенный с двумя областями, должен иметь две базы данных и выполнять два алгоритма наикратчайшего пути независимо. Чтобы поддерживать разные типы сервисов, OSPF использует несколько графов, один с разметкой относительно задержки, другой - относительно пропускной способности, третий - относительно надежности. Хотя все три требуют соответствующих вычислений, но зато мы получаем три маршрута, оптимизированных по задержке, пропускной способности и надежности. Во время функционирования возникают три вида маршрутов: внутри области, между областями и между АС. Внутри области вычислить маршрут просто - им будет наикратчайший до маршрутизатора получателя. Маршрутизация между областями всегда выполняется в три этапа: от источника до остовой области, от остовой до области назначения, внутри области назначения. Этот алгоритм навязывает звездообразную топологию OSPF: остовая область – центр, ось, остальные области – лучи, спицы. Пакеты маршрутизируются без изменений, как есть, за исключением случая, когда область получателя соединена с остовой областью туннелем. Рисунок 5-51 показывает эти три вида маршрутов.

Рисунок 5-51. Связи внутри области, между областями и между разными АС



5.5.5.3. Маршрутизаторы в OSPF

OSPF различает четыре класса маршрутизаторов:

1. Внутренний, целиком внутри одной области
2. Пограничный, соединяющий несколько областей
3. Остовый, принадлежащий остовой области
4. Пограничный, соединенный с маршрутизаторами других АС

Эти классы могут пересекаться. Например, все пограничные маршрутизаторы – остовые; маршрутизатор из остовой области, но не на ее границе - внутренний. Примеры этих классов маршрутизаторов показаны на рисунке 5–51. Когда маршрутизатор загружается, он рассылает сообщение «Hello» всем своим соседям - на линиях «точка-точка», группам маршрутизаторов в ЛВС с множественным доступом, чтобы получить информацию о своем окружении.

В OSPF маршрутизаторы обмениваются данными не со своими соседями, а со смежными маршрутизаторами. Это не одно и то же. Маршрутизатор не общается со всеми маршрутизаторами, например, внутри ЛВС, а лишь с тем, который объявлен выделенным маршрутизатором. Этот выделенный маршрутизатор смежен всем другим. У выделенного маршрутизатора есть дублер, который имеет ту же информацию, что и основной. Периодически в ходе нормальной работы каждый маршрутизатор рассылает всем своим смежным маршрутизаторам сообщение LINK STATE UPDATE. В этом сообщении он передает информацию о состоянии своих линий и их стоимости в разных метриках для базы данных топологии соединений. Это сообщение в целях надежности идет с подтверждением. Каждое такое сообщение имеет номер, который позволяет определить, несет ли пришедшее сообщение новую информацию по сравнению с той, что есть в его базе, или старую. Маршрутизаторы рассылают эти сообщения, когда у них появляются новые линии, разрушаются старые или меняется стоимость линии. DATABASE DESCRIPTION – сообщение, содержащее состояние всех каналов в базе данных отправителя. Сравнивая свои значения с теми, что у отправителя, получатель может определить, у кого наиболее свежая информация. Используя сообщение LINK STATE REQUEST, маршрутизатор может в любой момент запросить информацию о любой линии у другого маршрутизатора. Наиболее свежая информация распространяется другим. Все сообщения передаются как IP-пакеты. Все типы сообщений показаны в таблице 5-52. Маршрутизаторы в остовой области делают все, что было описано выше, а также обмениваются информацией с пограничными маршрутизаторами, чтобы уметь

вычислять наилучший маршрут от любого маршрутизатора остовой области до любого другого маршрутизатора.

5.5.6. BGP - внешний протокол маршрутизации шлюзов

Для маршрутизации между АС используется BGP - протокол пограничных шлюзов. Его предшественником был протокол EGP. Однако с ростом Интернета протокол EGP перестал удовлетворять требованиям к протоколу внешней маршрутизации. Основное отличие BGP от OSPF проистекает из различия в целях. При маршрутизации внутри АС основная цель - наикратчайший маршрут. При маршрутизации между АС надо учитывать также ряд условий, вызванных политикой конкретной АС. Например, какая-то АС может не допускать маршрутизацию через себя ни для какой другой АС (у них нет другого кратчайшего пути - это ваши проблемы); может разрешать лишь определенным. Типичными примерами таких ограничений могут быть:

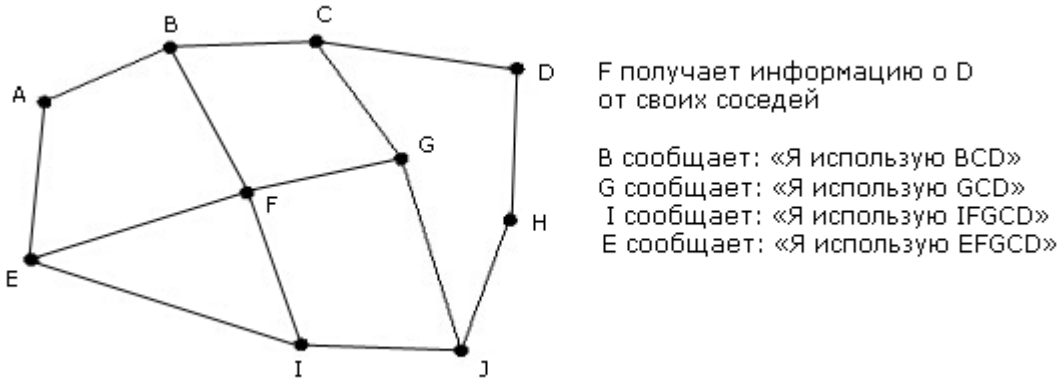
1. Трафик не должен проходить через определенные АС.
2. Маршрут, начинающийся в министерстве обороны России, никогда не должен проходить через Чечню.
3. Трафик через Украину может проходить, только если нет другого маршрута.
4. Трафик из IBM никогда не должен проходить через АС Microsoft.

Такие правила вручную вводятся в каждый BGP-маршрутизатор. С точки зрения BGP-маршрутизатора весь мир состоит из BGP-маршрутизаторов, соединенных между собой. Два BGP-маршрутизатора соединены, если у них есть общая сеть. Сети делятся на три категории по степени интереса направления трафика через сеть. Первая - тупиковые сети, они никуда не ведут. У них только одна точка соединения в BGP-графом. Они не могут использоваться для транзита. Сети с множественными соединениями могут использоваться для транзита, если допускают его. Транзитные сети предназначены для транзита трафика, возможно с некоторыми ограничениями.

Два BGP-маршрутизатора взаимодействуют через TCP-соединение. Это обеспечивает надежность передачи информации и скрывает все подробности от сетей, через которые она проходит.

BGP - это протокол на основе вектора расстояний. Однако вместо стоимости для каждого места в сети он хранит конкретный маршрут. Своим соседям он передает не вектор расстояний, а те маршруты, которые он использует. На рисунке 5-53 показан пример.

Рисунок 5-53. (a) Сеть из BGP-маршрутизаторов; (b) Информация, получаемая F



BGP-протокол легко решает проблему «счета до бесконечности». Предположим, что маршрутизатор G или линия FG отказали. Тогда F получит от своих соседей три оставшихся маршрута до D. Поскольку маршруты IFGCD и EFGCD проходят через F, то он их отбросит и воспользуется FBCD.

Определение BGP-протокола дано в RFC 1654 и RFC 1268.

Групповая адресация в Интернете

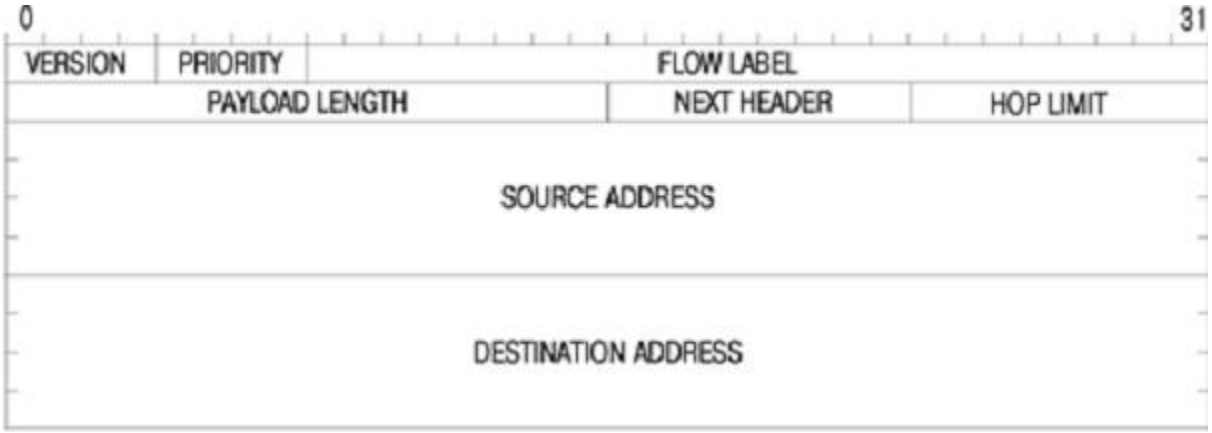
Обычно в IP-сетях один отправитель взаимодействует с одним получателем. Однако в ряде приложений бывает полезным одно и то же сообщение передать сразу нескольким получателям. Примеры: поддержка обновления данных в реплицируемых базах данных, передача биржевой информации сразу нескольким брокерам, поддержка телеконференций. В IP-сетях групповая адресация поддерживается с помощью адресов класса D, в нем 28 разрядов для адресации группы, т.е. можно адресовать 250 миллионов групп. При передаче сообщения группе делается все возможное, чтобы каждый член группы получил сообщение, однако это не гарантируется. Поддерживается два типа групповых адресов: постоянные и временные.Временные группы должны создаваться специальным образом и специальным образом удаляться. Каждый процесс на машине может запросить аппаратуру присоединиться к определенной группе или покинуть ее. Когда последний процесс на машине покинет группу, то эта группа более не представлена на этом хосте. Каждый хост следит, какие группы на нем представлены.

Групповая адресация реализуется специальным групповым маршрутизатором, который может размещаться отдельно от обычного маршрутизатора. Раз в минуту каждый групповой маршрутизатор рассылает через канальный уровень запрос всем хостам ЛВС указать, каким группам принадлежат их процессы. Эти запросы и ответы на них регулируются IGMP-протоколом (Internet Group Management Protocol), который очень похож на ICMP. Он описан в RFC 1112.

IPv6

Появление новой версии протокола IP (IPv6, в настоящее время используется IPv4) обусловлено целым рядом причин. Одна из основных - стремительный рост всемирной сети Интернет. Фундаментальным принципом построения сетей на основе протокола IP, необходимым для правильной маршрутизации и доставки пакетов, является уникальность сетевых адресов, т.е. каждый IP-адрес может принадлежать только одному устройству. На сегодняшний день остались невыделенными около 1 400 000 000 адресов из возможных 4 294 967 296, то есть примерно 30%, чего должно хватить на несколько лет, а может быть и более. Дефицит адресов пока выражается в основном в том, что, по выражению одного из сетевых гуру, адрес класса A не смог бы получить и сам Господь Бог. Таких адресов может существовать всего 128 (формат: 0, адрес сети - 7 бит, адрес хоста - 24 бита), но каждый из них содержит 16 777 216 адресов. Однако появившиеся в последнее время новые устройства для доступа в Интернет и развитие цифрового телевидения, которое собирается превратить каждый телевизор в интернет-устройство, могут быстро исчерпать имеющиеся запасы неиспользованных адресов. Если в компьютерных сетях для выхода в Интернет могут применяться технологии типа NAT (Network Address Translation, — преобразование сетевого адреса), при которой для взаимодействия с окружающей средой используется всего несколько уникальных адресов, предоставляемых, возможно, провайдером, а внутри локальной сети адресация может быть достаточно произвольной, то для сетевого телевизора этот способ не подходит, так как каждому устройству требуется свой уникальный адрес.

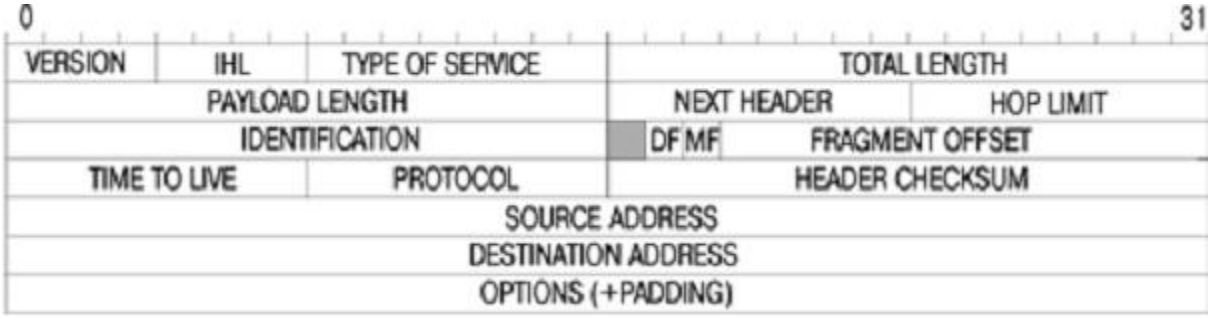
Рисунок 5-54. Заголовок пакета IPv6



Кроме всего прочего, новые возможности предъявляют к протоколам сетевого уровня, каковым является IP, совершенно новые требования в части легкости получения и смены адресов, полностью автоматического

конфигурирования (представьте себе домохозяйку, настраивающую DNS своего телевизора). Если новый протокол не появится своевременно, то фирмы-провайдеры начнут внедрять свои собственные, что может привести к невозможности гарантированного соединения «всех со всеми». Открытый протокол, удовлетворяющий требованиям необходимого адресного пространства, легкости конфигурирования и маршрутизации, способный работать совместно с имеющимся IPv4, поможет сохранить способность к соединению между собой любых устройств, поддерживающих IP, при наличии новых возможностей, которые основаны на анализе использования IPv4. Кроме того, остается еще одна проблема: уникальность адреса вовсе не означает, что устройство будет правильно функционировать. Адреса нужны в первую очередь не для того, чтобы «всех пересчитать», а для правильной маршрутизации при доставке пакетов. Таким образом, для беспрепятственного роста Интернета необходимо не только наличие свободных адресов, но и определенная методика их выделения, позволяющая решить проблему масштабируемости. Сведение к минимуму накладных расходов на маршрутизацию является сегодня одной из основных проблем, и ее важность будет возрастать в дальнейшем по мере роста Сети. Просто присвоить устройству адрес недостаточно, необходимо еще обеспечить условия для правильной маршрутизации с минимальными накладными расходами.

Рисунок 5-55. Заголовок пакета IPv4 (для сравнения)



В настоящее время только одна известная технология, а именно, иерархическая маршрутизация, позволяет за счет приемлемых технических издержек обеспечить доставку пакетов в сети размерами с Интернет. Технология иерархической маршрутизации заключается в разбиении всей сети на более мелкие подсети, маршрутизация в которых производится самостоятельно. Подсети, в свою очередь, могут разбиваться на еще более мелкие, и т.д. В результате образуется древовидная структура, причем в качестве узлов выступают маршрутизаторы, а в качестве листьев - оконечные устройства-хосты. Путь, который проделывает пакет, передаваемый от одного листа до другого, может быть длиннее, чем при иной топологии, но зато он всегда может быть рассчитан с наименьшими издержками. Некоторую аналогию можно провести с телефонными номерами — первым идет код страны, за ним код города, а затем собственно номер, состоящий, в свою очередь, из кода АТС и собственно номера абонента. История нового протокола восходит к концу 1992 года. Именно тогда IETF (Internet Engineering Task Force — рабочая группа по технической поддержке Интернет) приступила к анализу данных, необходимых для разработки нового протокола IP. К концу 1994 года был утвержден рекомендательный стандарт и разработаны все необходимые для реализации протокола вспомогательные стандарты и документы. IPv6 является новой версией старого протокола, разработанной таким образом, чтобы обеспечить совместимость и «мягкий» переход, не приуроченный к конкретной дате и не требующий одновременных действий всех участников. По некоторым прогнозам, совместное существование двух протоколов будет продолжаться до десяти и более лет. Учитывая то обстоятельство, что среди выделенных типов адресов IPv6 имеется специальный тип адреса, эмулирующий адрес IPv4, можно ожидать относительно спокойного перехода, не сопровождающегося крупными неудобствами и неприятностями. Фактически на одном компьютере могут работать оба протокола, каждый из которых подключается по мере необходимости.

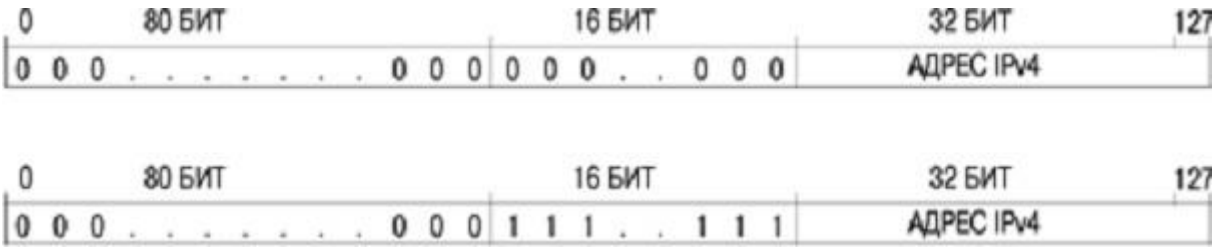
Рисунок 5-56. Provider-Based Unicast Address - адресация единственного абонента, основанная на адресе провайдера

Однако использование старых адресов не является выходом из положения, поэтому протокол IPv6 предусматривает специальные возможности по присвоению новых адресов и их замене без вмешательства (или при минимальном вмешательстве) персонала. Для этого предусмотрена привязка к компьютеру не IP-адреса, а интерфейса. Сам же интерфейс может иметь несколько адресов, принадлежащих к трем категориям: действительный, прошлый, недействительный. При замене адреса «на лету» новый адрес становится действительным, а старый — прошлым. Все вновь осуществляемые соединения производятся при помощи действительного адреса, но уже имеющиеся продолжаются по прошлому адресу. Через некоторое время, которое может быть выбрано достаточно большим, чтобы гарантировать полный разрыв всех соединений по прошлому адресу, он переходит в категорию недействительных. Таким образом, практически гарантируется автоматическая замена адреса без участия персонала. Для полностью гарантированной автоматической замены адреса потребовалось бы внесение изменений в протоколы TCP и UDP, которые не входят в состав IP. Замена адресов осуществляется двумя способами — явным и неявным. Явный способ использует соответствующим образом доработанный протокол DHCP. Неявный способ не требует наличия сервера DHCP, а использует адрес подсети, получаемый от соседей и мостов. В качестве адреса хоста используется просто MAC-адрес хоста, т.е. адрес, используемый на канальном уровне. Этот способ, при всем своем изяществе, по понятным причинам не может присваивать адреса, совместимые с IPv4, и поэтому в переходный период его применение будет ограничено. К сожалению, механизм выделения новых адресов не затрагивает таких аспектов, как обновление базы данных DNS, адресов серверов DNS, конфигурации маршрутизаторов и фильтров, а также тех приложений «клиент-сервер», которые используют привязку к адресу, что делает полную замену адресов локальной сети не менее трудоемким мероприятием, чем при применении IPv4.

Рисунок 5-57. Локальные адреса для использования в пределах сегмента (вверху) и локальной сети (внизу)

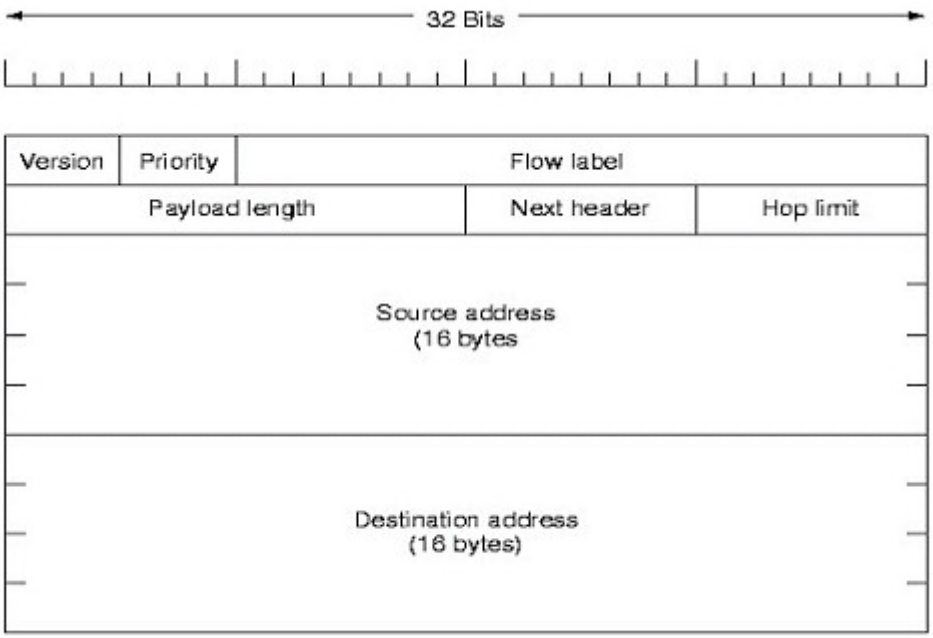
Протокол IPv6 предполагает также значительные улучшения при работе в локальной сети. Единый протокол NDP (Neighbor Discovery Protocol - протокол распознавания соседей) заменяет используемые в IPv4 протоколы ARP, ICMP и значительно расширяет их функциональные возможности. Вместо широковещательных пакетов канального уровня протокола ARP используются групповые сообщения (multicast), то есть адресованные всем членам подсети, причем не на канальном, а на сетевом уровне, что должно значительно снизить широковещательный трафик, являющийся бичом локальных сетей Ethernet. Усовершенствованы функции протокола ICMP, что облегчает работу разных подсетей в одном физическом сегменте. Включен механизм распознавания неисправных маршрутизаторов, что позволяет повысить устойчивость к сбоям оборудования. В дополнение к имевшимся ранее двум типам адресации - Unicast и Multicast (доставке уникальному получателю или группе получателей) - добавлен третий, Anycast, при котором осуществляется доставка любому получателю из группы. Существенное отличие нового протокола от старого заключается в том, что длина адресной части составляет 128 бит — в четыре раза больше, чем 32 бита у IPv4. Чтобы представить эту величину, достаточно сказать, что на каждом квадратном метре поверхности суши и моря можно разместить примерно 6,7x1023 адресов. Из заголовка пакета IP изъяты как некоторые неиспользуемые поля, что позволило сократить издержки, связанные с их обработкой, и уменьшить размер заголовка (он длиннее, чем у IPv4, всего в два раза, несмотря на учетверенный размер адресной части).

Рисунок 5-58. Адреса, построенные на основе IPv4: (a) Совместимый; (b) Для устройств, не поддерживающих IPv6



На рисунке 5-59 показана структура заголовка пакета в формате Ipv6.

Рисунок 5-59. Заголовок пакета IPv6



Первым идет четырехбитное поле Version (Версия), его значение равно 6. Следующее поле - Priority (Приоритет) - длиной 8 бит используется для установки приоритета пакета. Приоритет увеличивается с ростом значения этого поля. Значения 0...7 используются для пакетов, время доставки которых не лимитировано, например, значение 1 рекомендуется использовать для новостей, 2 — для почты, 7 — для служебного трафика (SNMP, маршрутизирующие протоколы). Значения 8...15 используются для пакетов, задержка доставки которых нежелательна, например аудио и видео в реальном времени. Далее следует поле Traffic Class, первоначально называвшееся «Flow Label», длиной 20 бит. Оно служит для идентификации последовательности пакетов. Его значение присваивается при помощи генератора случайных чисел и имеет одинаковую величину у всех пакетов данной последовательности. Следующее поле - Payload Length - содержит размер данных, следующих за заголовком, в байтах и имеет длину 16 бит. Следом расположено поле Next Header, идентичное по назначению полю Protocol протокола IPv4 и использующее те же значения. Восьмибитное поле Hop Limit аналогично по назначению полю Time to Live. Оно устанавливается источником согласно разумным предположениям о длине маршрута, а затем уменьшается на 1 при каждом прохождении через маршрутизатор. При снижении значения поля до нуля пакет снимается, как «заблудившийся». Последними идут поля адресов источника и приемника длиной 128 бит (16 байт) каждое. Адреса в стандарте IPv6 имеют более сложную структуру, чем в предыдущем, при этом используются префиксы разной длины. В настоящее время для непосредственного использования предназначено около 15% адресов, остальные 85% зарезервированы для распределения в будущем. Специальные типы адресов предназначаются для более гибкого использования. Provider-Based Unicast Address (Выделяемый провайдером уникальный адрес) служит для глобальной связи. Он состоит из префикса 010, Registry Id, идентифицирующего организацию, зарегистрировавшую провайдера; Provider Id, идентифицирующего провайдера; Subscriber Id, идентифицирующего организацию-клиента, и собственно адреса. Адреса для локального использования (Link Local Use и Site Local Use) предназначены для применения внутри одного сегмента или одной организации, т.е. пакеты с такими адресами не маршрутизируются за границы текущего

сегмента или локальной сети соответственно. Они могут быть использованы, например, при автоматическом присвоении адресов. Для выхода в глобальную сеть может быть использована подстановка адресов по типу NAT. Если под заполнители-нули выделено достаточно места, то организация, ранее не имевшая соединения с Интернетом, может легко провести замену адресов на глобальные путем конкатенации REGISTRY.

Рисунок 5-60. Адресация группы абонентов (Multicast Address)

ID + PROVIDER ID + SUBSCRIBER ID и локального адреса. К специальным типам адресов также относятся адреса, совместимые с IPv4. Первый тип относится к «совместимым» адресам, которые предназначены для туннелирования пакетов IPv6 через существующую инфраструктуру IPv4. Второй тип адресов отображает на IPv6 подмножество адресов IPv4 для тех устройств, которые не поддерживают новый протокол. Широковещательный адрес благодаря использованию полей Flags и Scope может также использоваться более гибко. В четырехбитном поле Flags пока используется только младший бит для указания, является ли данный адрес постоянным и выделенным соответствующими организациями, ответственными за выдачу адресов, или используется единовременно. Поле Scope используется для ограничения области распространения широковещательных пакетов. Значения этого поля приведены в таблице 2. При рассмотрении возможностей, предоставляемых новым протоколом, может возникнуть вопрос, а зачем он все-таки нужен? Большинство функций либо уже имеются в IPv4, либо могут быть реализованы путем доработки соответствующих протоколов. Так, автоматическое выделение адресов производится при помощи протокола DHCP, адресный барьер преодолевается при помощи протокола NAT и т.д. и т.п. Однако разработка всех необходимых заплаток для протокола IPv4 потребовала бы не меньших (а то и больших) усилий, чем создание нового протокола «с чистого листа». Разумеется, лист был не совсем чистым, поскольку вопросы совместимости и совместной работы обоих протоколов имелись в виду с самого начала проектирования. В конце концов, Интернет все равно пришел бы к кризису дефицита адресов, так что заблаговременная разработка и постепенное внедрение протокола IPv6 были более чем уместны.

Транспортный уровень: сервис, примитивы, адресация, установление соединения, разрыв соединения, управление потоком и буферизация, мультиплексирование, восстановление разрывов.

Транспортный протокол - это центральный протокол во всей иерархии протоколов. Именно он обеспечивает надежную передачу данных в сети от одного абонента к другому. Здесь мы подробно рассмотрим организацию, сервис, протоколы и производительность на транспортном уровне.

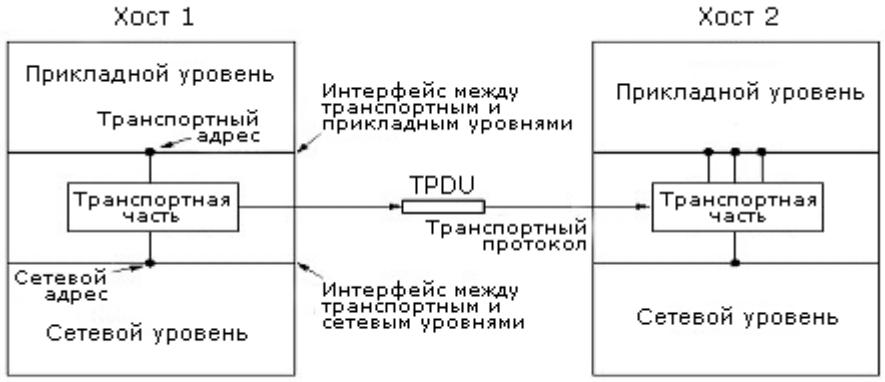
Сервис

В этом разделе мы рассмотрим, какие виды сервиса транспортный уровень предоставляет прикладному, как характеризуется качество предоставляемого сервиса, как обеспечивается доступ из прикладного уровня к транспортному, как выглядит интерфейс транспортного уровня со смежными уровнями.

1. Сервис для верхних уровней

Основная цель транспортного уровня - обеспечить эффективный, надежный и дешевый сервис для пользователей на прикладном уровне. Достижение этой цели - задача сервиса, предоставляемого сетевым уровнем. То, что выполняет работу транспортного уровня, называется транспортным агентом. Транспортный агент может располагаться в ядре операционной системы, в отдельном процессе пользователя, в библиотеке сетевого приложения или на карте сетевого интерфейса. В некоторых случаях оператор сети может предоставлять надежный транспортный сервис, при котором транспортный агент располагается на специальной интерфейсной машине на границе транспортной среды, к которой подключены абонентские машины. На рисунке 6-1 показано взаимное расположение сетевого, транспортного и прикладного уровней.

Рисунок 6-1. Сетевой, транспортный и прикладной уровень



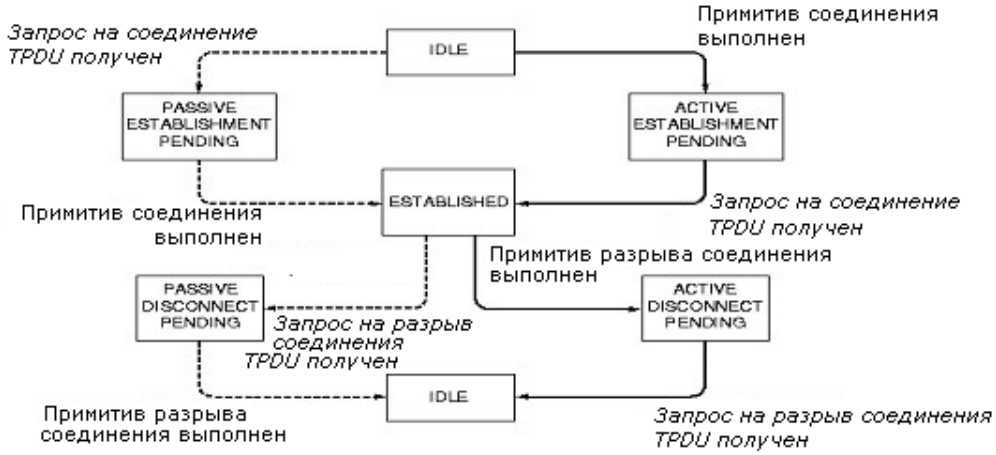
Подобно сетевому уровню, транспортный уровень также может поддерживать два вида сервиса - ориентированный на соединения и без соединений. Транспортный сервис, ориентированный на соединение, имеет много общего с аналогичным сетевым сервисом. Адресация и управление потоком также схожи на обоих уровнях. Возникает вопрос: если сервис сетевого уровня столь схож с сервисом транспортного, то зачем выделять два разных уровня? Причина состоит в том, что сетевой уровень - это часть транспортной среды, которой управляет оператор. Что будет, если сетевой уровень предоставляет ненадежный сервис, ориентированный на соединения? Предположим, что он часто теряет пакеты. Что делать, если маршрутизатор время от времени отказывает? У пользователя такой транспортной среды отсутствуют средства для решения этих проблем, в случае их возникновения. Нужно поверх сетевого пустить еще один уровень, который позволит исправлять качество сервиса сетевого уровня. Если транспортному уровню придет сообщение, что соединение на сетевом уровне неожиданно было разорвано, то он может установить новое сетевое соединение, с помощью которого выяснить, что произошло, какие данные были переданы, а какие нет, и т.п. Задача транспортного уровня в том, чтобы сделать сервис транспортного уровня для прикладного более надежным, чем сетевого для транспортного. Другое важное свойство транспортного уровня - прикладная программа, опираясь на транспортный сервис, становится независимой от сети и может работать в сети с любым сетевым сервисом. И, наконец, с транспортным сервисом работает прикладная программа, а с сетевым – транспортный уровень. Поэтому интерфейс с транспортным уровнем должен быть дружественным, удобным и эффективным. В силу приведенных доводов первые четыре уровня называют поставщиками транспортного сервиса, а все, что выше четвертого, - пользователями транспортного сервиса.

Примитивы транспортного уровня

Примитивы транспортного уровня позволяют пользователю получить доступ к транспортному сервису. Транспортный сервис аналогичен сервису сетевого уровня. Однако между ними существует одно различие - сетевой сервис по природе своей ненадежен. Задача транспортного сервиса как раз обеспечить надежную доставку сообщений. Два процесса, соединенные между собой, ничего не должны знать о том, как физически они соединены. Один помещает данные на вход транспортного уровня, другой получает их. Задача транспортного уровня скрыть и от получателя и от отправителя все детали передачи, исправления ошибок и т.п. Теоретически транспортный сервис может быть как ориентированным на соединения, так и нет. Однако дейтаграммный транспортный сервис - это редкость, поэтому мы будем рассматривать транспортный сервис, ориентированный на соединения. Другое важное различие между сетевым и транспортным сервисами состоит в том, кто их использует. Сетевой сервис использует транспортный сервис, а вот транспортный использует пользователь, т.е. прикладные программы. Поэтому транспортный сервис должен быть ориентирован на пользователя, удобным и простым в использовании. Общее представление о примитивах транспортного сервиса дает таблица 6-3. Этот пример содержит основные виды примитивов для установления соединения, передачи данных и разрыва соединения, что вполне достаточно для многих приложений. Использование этих примитивов может быть продемонстрировано следующим образом. Сервер приложения выполняет примитив LISTEN, в результате чего он блокируется до поступления запросов от клиентов. Клиент для установления соединения выполняет примитив CONNECT. Транспортный агент на стороне клиента блокирует клиента и посылает серверу пакет с запросом на установление соединения. Напомним, что транспортные агенты обмениваются пакетами, которые имеют специальное название - Transport Protocol Data Unit (TPDU). По примитиву CONNECT транспортный агент на стороне клиента шлет

CONNECTION REQUEST TPDU. Транспортный агент сервера, видя, что сервер заблокирован по LISTEN, разблокирует сервер и посылает CONNECTION ACCEPTED TPDU. После этого транспортное соединение считается установленным и начинается обмен данными с помощью примитивов SENT и RECEIVE. По окончании обмена соединение должно быть разорвано. Есть два варианта разрыва соединения: симметричный и асимметричный. Асимметричный разрыв предполагает, что для разрыва соединения одна из сторон посылает DISCONNECT TPDU. Получив этот TPDU, другая сторона считает соединение разорванным. При симметричном разрыве каждое направление закрывается отдельно. Когда одна сторона посылает DISCONNECT TPDU, это значит, что с ее стороны больше данных не будет. На рисунке 6-5 показана диаграмма состояний при установлении и разрыве соединения. (Переходы, выделенные курсивом, вызываются прибытием пакета. Пунктирной линией обозначена последовательность состояний сервера, сплошной - клиента.)

Рисунок 6-5. Диаграмма состояний при установлении и разрыве соединения



В таблице 6-6 показан другой набор примитивов, так называемые сокеты Беркли. В этом наборе два основных отличия от того, что мы только что рассмотрели. Первые четыре примитива выполняются сервером в том порядке, как они указаны в таблице. Примитив SOCKET создает новую точку подключения к серверу, резервирует для нее место в таблице транспортного агента. Параметры обращения определяют формат адреса, тип желаемого сервиса, протокол и т.д. По примитиву BIND сервер выделяет сокету адрес. Причина, по которой адрес выделяется не сразу, в том, что некоторые процессы сами управляют своими адресами, которые жестко закреплены за ними. Второе – LISTEN - не блокирующий примитив. Он выделяет ресурсы и создает очередь, если несколько клиентов будут обращаться за соединением в одно и то же время. Примитив ACCEPT - блокирующий в ожидании запроса на соединение. Когда клиент выполняет примитив CONNECT, он блокируется своим транспортным агентом, и запускается процесс установления соединения. Когда он закончится, клиента разблокируют, и начинается обмен данными с помощью примитивов SEND и RECEIVE. Разрыв соединения здесь симметричен, т.е. соединение считается разорванным, если обе стороны выполнили примитив CLOSE.

6.2. Элементы транспортного протокола

Транспортный сервис реализует транспортный протокол, который используют транспортные агенты. Транспортный протокол в чем-то схож с канальным. Однако между ними несколько различий:

1. Они работают в разных средах (см. рисунок 6-7).

2. Процессы на канальном уровне взаимодействуют непосредственно через физическую среду, поэтому процедура установления соединения много проще.

3. Среда, в которой работает транспортный протокол, использует память, которая может терять свое содержимое.

4. Количество соединений, которое может возникать на транспортном уровне, намного больше, чем количество соединений на канальном уровне, что создает дополнительные проблемы для буферизации и управления потоком.

Рисунок 6-7. (a) Среда канального уровня; (b) Среда транспортного уровня

Транспортный протокол должен решать следующие проблемы:

1. Как адресовать прикладной процесс, с которым надо установить соединение?
2. Как корректно установить соединение? Пакеты могут теряться. Как отличить пакеты нового соединения от повторных пакетов, оставшихся от старого?
3. Как корректно разрывать соединение?

6.2.1. Адресация

Проблема адресации состоит в том, как указать, с каким удаленным прикладным процессом надо установить соединение. Обычно для этого используется транспортный адрес, по которому прикладной процесс может слушать запросы на соединение. Вместо него мы будем здесь использовать термин TSAP - Transport Service Access Point. Аналогичное понятие существует и на сетевом уровне - IP-адрес - NSAP для сетевого уровня. Из этой иллюстрации не ясно лишь, как прикладной процесс на машине 1 узнает, что интересующий его сервер подключен к TSAP 122 на машине 2? Одно из возможных решений - если данный сервер всегда подключен к TSAP 122, и все процессы об этом знают. Такое решение хорошо работает для часто используемого сервиса с длительным периодом активности, но как быть прикладным процессам пользователя, которые активизируются спорадически на короткое время? Одно из решений, используемых в операционной системе Unix, показано на рисунке 6-9. Оно называется протоколом установления начального соединения. На каждой машине есть специальный сервер процессов, который представляет все процессы, исполняемые на этой машине. Этот сервер слушает несколько TSAP, куда могут поступить запросы на TCP-соединение. Если нет свободного сервера, способного выполнить запрос, то соединение устанавливается с сервером процессов, который переключит соединение на нужный сервер, как только он освободится.

Рисунок 6-9. Пользовательский процесс на хосте 1 устанавливает соединение с сервером времени

Однако есть случаи, когда этот подход с сервером процессов не работает. Не всегда можно запускать сервер сервиса по требованию пользователя. Например, файловый сервер. Он должен существовать всегда. Решение в этом случае - сервер имен. Пользователь устанавливает соединение с сервером имен, для которого TSAP известен, и передает ему имя сервиса. В ответ сервер имен шлет надлежащий TSAP. Клиент разрывает соединение с сервером имен и устанавливает его по полученному адресу. Пусть пользователь узнал TSAP, но как он узнает, на какой машине этот TSAP расположен, какой сетевой адрес надо использовать? Ответ заключается в структуре TSAP-адреса, где указана вся необходимая информация. Этот адрес имеет иерархическую структуру.

6.2.2. Установление соединения

Проблема установления транспортного соединения сложна потому, что пакеты могут теряться, храниться и дублироваться на сетевом уровне. Типичный пример - установление соединения с банком для перевода денег

с одного счета на другой. Из-за перегрузки в сети или по какой-либо другой причине может произойти большая задержка. Тогда по time_out активная сторона вышлет еще один запрос. Пакеты-дубли могут вызвать повторное соединение и вторичный перевод денег. Как быть? Одно из возможных решений - временное TSAP. После того, как оно использовано, TSAP с таким адресом более не возникает. При этом решении не работает модель с сервером процессов (рисунок 6-9), когда определенные TSAP имеют фиксированные адреса. Другое решение - каждому транспортному соединению сопоставлять уникальный номер. Когда соединение разрывается, этот номер заносится в специальный список. К сожалению, этот список может расти бесконечно. Кроме этого, в случае сбоя машины он может быть потерян, и тогда... Альтернативой может быть ограничение времени жизни пакетов. Достичь этого можно тремя путями:

1. ограничением конструкции подсети
2. установкой счетчиков скачков в каждом пакете
3. установкой временной метки на каждом пакете

Заметим, что последний метод требует синхронизации маршрутизаторов в сети. На практике нам надо обеспечить, чтобы стали недействительными не только сами пакеты, но и уведомления о них. Это значит, что надо ввести величину T – множитель для максимального, реального времени жизни пакета в сети. Его конкретное значение зависит от конкретного протокола и позволяет немного увеличить реальное время жизни так, чтобы по его истечении в сети не осталось ни самого пакета, ни уведомления о нем. При ограничении времени жизни пакета можно построить безопасный способ установления соединения. Этот метод был предложен Томлинсоном (Tomlinson). Его идея состоит в следующем. Все машины в сети оснащены таймерами. Таймер работает даже в случае сбоя машины, т.е. он абсолютно надежен. Каждый таймер - двоичный счетчик достаточно большой разрядности, равной или превосходящей разрядность последовательных чисел, используемых для нумерации пакетов. При установлении соединения значения нескольких младших разрядов этого таймера берутся в качестве начального номера пакета. Главное, чтобы последовательности номеров пакетов одного соединения не приводили к переполнению счетчика и его обнулению. Эти номера можно также использовать для управления потоком в протоколе скользящего окна. Проблема возникает, когда машина восстанавливается после сбоя. Транспортный агент не знает в этот момент, какое число можно использовать для очередного номера. Чтобы избежать повторного использования порядкового номера, который уже был сгенерирован перед сбоем машины, вводится специальная величина по времени, которая образует область запрещенных. Необходимость введения этой области демонстрирует следующий пример. Предположим для простоты, что в момент x начальный номер будет x, и что максимальное время жизни пакета равно 60 сек. Пусть в t=30 сек. был послан пакет с номером 80, после чего наступил сбой в ее работе. Пусть в момент t=60 машина была восстановлена после сбоя, и в момент t=70 появился пакет с номером 80. Однако старый пакет с номером 80 все еще существует, так как он будет жить до t=90. Поэтому для каждого момента времени вводят область запрещенных номеров. Машина, восстановленная после сбоя, не может выбирать номера из этой запретной зоны. Поэтому после восстановления следует подождать T сек., пока все ранее посланные пакеты не перестанут существовать. На практике поступают иначе, чтобы не тратить впустую эти T сек. Строят кривую скорости генерации номеров n=at, тогда после сбоя надо выбирать номера по формуле: n=at+T. Проблема номеров может возникать по двум причинам. Либо потому, что машина генерирует слишком быстро пакеты и соединения, либо потому, что делает это слишком медленно. Чем больше разрядность счетчика последовательных номеров, тем дальше отодвигается момент попадания в запретную область. Другая нетривиальная проблема - надежное установление соединения: пакеты ведь могут пропадать. Для ее решения Томлинсон предложил процедуру «троекратного рукопожатия» (three-way handshake), которая проиллюстрирована на рисунке 6-11. (CR и ACK обозначают соответственно CONNECTION REQUEST и CONNECTION ACCEPTED.) Эта процедура предполагает, что машина 1 шлет запрос на установление соединения под номером x. Машина 2 шлет подтверждение на запрос x, но со своим номером y. Машина 1 подтверждает получение подтверждения с номером y.

6.2.3. Разрыв соединения

Разрыв соединения, как уже было сказано, может быть асимметричным или симметричным. Асимметричный разрыв может привести к потере данных .Симметричный разрыв каждая сторона проводит самостоятельно, когда она передала весь имеющийся объем данных. Однако определить этот факт не всегда просто. Здесь есть одна проблема, которая называется проблемой двух армий (см. рисунок 6-13). Суть этой проблемы в

следующем. Пусть есть две противоборствующие армии, скажем, A и B. Армия A представлена двумя группировками, между которыми расположена армия B. Суммарно ресурсы A превосходят ресурсы B, и, если обе группировки A ударят по B, то A победит. Дело лишь за тем как договорится, чтобы обе группировки ударили одновременно. Имеется сложность – гонец от A должен пройти через территорию, контролируемую B. Пусть группировка №1 шлет гонца с донесением, в котором указано время атаки. Вопрос, выслав гонца, может ли армия №1 выступать? Конечно, нет! Если гонец не доставил донесение, то атака будет отбита, ресурсы потрачены, и B победит. Выход из создавшегося положения – дождаться гонца от армии №2 с подтверждением. Пусть гонец от армии №2 прибыл. Можно ли наступать? Опять нельзя! Не получив подтверждения, что гонец доставил подтверждение, армия №2 не может выступить. Этот процесс ожидания подтверждений можно продолжать сколь угодно долго.

Рисунок 6-13. Проблема двух армий

Внимательно изучив проблему разрыва соединения, мы придем к выводу, что ни одна армия не начнет атаки до тех пор, пока не получит подтверждения на подтверждение, и так до бесконечности. На самом деле, можно доказать, что нет протокола, который безопасно разрешает эту ситуацию. На рисунке 6-14 показаны четыре сценария разрыва соединения: нормальный случай с «тройным рукопожатием», с потерей последнего подтверждения, с потерей ответа и с потерей ответа и последующих данных. Обычно эту проблему решают, фиксируя число попыток разрыва.

6.2.4. Управление потоком и буферизация

Теперь, рассмотрев, как устанавливают соединение, обратимся к тому, как им управляют. Прежде всего, рассмотрим управление потоком. Проблема управления потоком на транспортном уровне в чем-то аналогична проблеме управления потоком на канальном уровне. Различия в том, что у маршрутизатора число каналов невелико, в то время как на транспортном уровне соединений может быть очень много. Канальный протокол сохранял пакеты как на стороне отправителя, так и на стороне получателя до тех пор, пока они не будут подтверждены. Если у нас есть 64 соединения и поле «время жизни» пакета занимает 4 разряда, то нам потребуется суммарная емкость буферов на 1024 TPDU-пакетов. Число буферов можно сократить, если есть информация о надежности сетевого уровня или о наличии буфера у получателя. На транспортном уровне отправитель сохраняет все пакеты на случай, если какой-то из них придется посылать вторично. Если получатель знает об этом, то он может иметь лишь один пул буферов для всех соединений, и, если пришел пакет и ему нет буфера в пуле, то он сбрасывается, в противном случае сохраняется и подтверждается. Если сетевой уровень не надежный, то на транспортном уровне отправитель вынужден сохранять все отправленные пакеты до тех пор, пока они не будут подтверждены. При надежном сетевом сервисе, наоборот, отправителю нет нужды сохранять отправленные пакеты, если он уверен, что у получателя всегда есть буфер для сохранения полученного TPDU. Если такой уверенности нет, то ему придется сохранять пакеты. Однако и в первом и во втором случае возникает проблема размера буфера. При фиксированной длине буфера естественно организовывать пул буферов одного размера. Однако при переменной длине пакетов проблема становится много сложнее. Если размер буфера устанавливать по максимальной длине пакета, то мы столкнемся с проблемой фрагментации, т.е. неэффективного использования пространства. Если по минимальной длине, то один пакет придется пересылать как несколько, с дополнительными накладными расходами. Можно установить схему динамического согласования размера буфера при установлении соединения. Оптимальное соотношение между буферизацией на стороне отправителя или на стороне получателя зависит от типа трафика. Для низкоскоростного, нерегулярного трафика буферизацию лучше делать на обоих концах. В общем случае вопрос о количестве буферов лучше всего решать динамически. Здесь надо только позаботиться о решении проблемы потери управляющих пакетов. Другую проблему представляет согласование доступного числа буферов и пропускная способность сетевого уровня. Дело в том, что пропускная способность транспортной среды между двумя определенными хостами ограниченна, и если поток между ними превысит пропускную способность транспортной среды, то возникнет перегрузка. Эту проблему лучше всего решать динамически с помощью управляющих сообщений. Механизм управления потоком должен прежде всего учитывать пропускную способность подсети, а уже потом - возможности буферизации. Располагаться этот механизм будет на стороне отправителя, чтобы предотвращать накопление большого числа неподтвержденных сообщений.

6.2.5. Мультиплексирование

Потребность в мультиплексировании нескольких потоков одного уровня на одном соединении, виртуальном канале, физической линии на других уровнях возникает постоянно. Эта проблема возникает и на транспортном уровне. Например, если пользователь за терминалом установил транспортное соединение и отошел попить кофе, то транспортное соединение продолжает поддерживаться, под него резервируется буферное пространство, пространство в таблице маршрутизации и т.д. В целях удешевления стоимости транспортных соединений можно отобразить несколько транспортных соединений на одно сетевое. Такое отображение называется нисходящим мультиплексированием. В некоторых случаях, наоборот, в целях увеличения пропускной способности по отдельным транспортным соединениям можно отобразить транспортное соединение на несколько сетевых и по каждому сетевому иметь свое скользящее окно. Тогда, быстро исчерпав возможности одного оконного буфера, можно переключиться на другое сетевое соединение и продолжить передачу по нему. В этом случае мы получим канал, пропускная способность которого равна сумме пропускных способностей отдельных каналов на сетевом уровне. Такое мультиплексирование называется восходящим. **Восстановление после сбоев .** Восстановление после сбоев мы будем рассматривать в предположении, что транспортный агент целиком располагается на абонентской машине. Восстановление сетевого уровня достаточно просто. Если сетевой уровень предоставляет дейтаграммный сервис, то транспортный уровень знает, как исправлять подобные ситуации. При сервисе, ориентированном на соединение, транспортный уровень восстановит потерянное соединение и постарается в диалоге с транспортным агентом на другой стороне выяснить, что успели передать, а что нет. Проблема становится сложнее, когда надо восстанавливать работоспособность машины, включая и транспортный уровень. Рассмотрим случай, когда транспортный сервер взаимодействует с клиентами. Предположим, сервер упал и старается восстановить функционирование. Прежде всего, ему надо узнать у клиента, какое TPDU было последним неподтвержденным, и попросить повторить его. В свою очередь, клиент может находиться в одном из двух состояний: S1 – есть неподтвержденное TPDU, либо S0 – все TPDU подтверждены. Казалось бы, все просто. Однако рассмотрим проблему внимательнее. Сервер, получив TPDU, либо сначала шлет подтверждение, а затем записывает полученное TPDU в буфер приложения, либо сначала записывает, а потом шлет подтверждение. Если сервер упал, послав подтверждение, но до того, как он осуществил запись, то клиент будет находиться после восстановления сервера в состоянии S0, хотя подтвержденное TPDU потеряно. Пусть, наоборот, сервер сначала записал TPDU, а потом упал. Тогда после сбоя сервер найдет клиента в состоянии S1 и решит, что надо повторить неподтвержденное TPDU. В результате получим повторное TPDU.

Можно формально показать, что эта проблема только средствами транспортного уровня не решается. Надо, записав TPDU, информировать об этом приложение и только после этого слать подтверждение. При восстановлении надо опрашивать не только клиента на транспортном уровне, но и приложение.

Транспортный уровень в Internet (TCP, UDP). Сервис TCP, протокол, заголовок сегмента, управление соединениями, стратегия передачи, управление перегрузками, управление таймерами. Протокол UDP, беспрводной TCP и UDP. Способы ускорения обработки TPDU.

Транспортные протоколы в Internet: TCP и UDP .

В Internet есть два основных транспортных протокола: TCP - ориентированный на соединение и UDP - не ориентированный на соединение. Поскольку сервис, реализуемый протоколом UDP - это практически сервис, реализуемый протоколом IP, с добавлением небольшого заголовка, то основное внимание здесь мы уделим протоколу TCP.

TCP (Transmission Control Protocol) - специально созданный протокол для надежной передачи потока байтов по соединению «точка-точка» через ненадежную сеть. TCP был сознательно разработан так, чтобы он мог адаптироваться к условиям и особенностям разных сетей, устойчиво и эффективно функционировать в условиях internet (нескольких сетей). На каждой машине, поддерживающей TCP, есть TCP-агент, который располагается либо в ядре ОС, либо в процессе пользователя, который управляет TCP-потоками и доступом к сервису IP-протокола. TCP получает поток данных от прикладного процесса, дробит их на сегменты не более чем по 65 Кбайт (на практике не более 1,5 Кбайт) и отправляет их как отдельные IP-пакеты. Поскольку IP-уровень не гарантирует доставку каждого пакета, то в задачу TCP входит определение потерь и организация повторной передачи потерянного. Поскольку на сетевом уровне в Internet соединения не поддерживаются, то сегменты могут поступать к получателю в неправильном порядке и задача TCP - восстановить этот порядок.

6.3.1. Модель сервиса TCP

Доступ к TCP-сервису происходит через сокет. Сокет состоит из IP-адреса хоста и 16-разрядного локального номера на хосте, называемого порт. Сокеты создаются как отправителем, так и получателем. Порт - это TSAP для TCP. Каждое соединение идентифицируется парой сокетов, между которыми оно установлено. Один и тот же сокет может быть использован для разных соединений. Никаких дополнительных виртуальных соединений не создается. Порты с номерами до 256 зарезервированы для стандартного сервиса. Например, если надо обеспечить FTP-передачу файла, то надо соединяться через 21-й порт, где находится FTP-демон. Для TELNET - через 23-й порт. Полный список таких портов можно найти в RFC 1700. Все TCP-соединения - дуплексные, т.е. передача идет независимо в оба направления. TCP-соединение поддерживает только соединение «точка-точка». Не существует TCP-соединений от одного ко многим. TCP обеспечивает поток байтов, а не поток сообщений. Напомним, это значит, что границы сообщений не поддерживаются автоматически в потоке. После того, как приложение передало данные TCP агенту, эти данные могут быть отправлены сразу на сетевой уровень, а могут быть буферизованы, как поступить - решает TCP-агент. Однако в ряде случаев надо, чтобы они были отправлены сразу, например, если эти данные представляют собой команду для удаленной машины. Для этого в заголовке TCP-пакета есть флаг PUSH. Если он установлен, то это говорит о том, что данные должны быть переданы немедленно.

Наконец, последняя возможность TCP-сервиса, которую здесь стоит упомянуть - срочные данные. Если для данных установлен флаг URGENT в заголовке, то все данные после этого по данному соединению передаются сразу и не буферизуются. Когда срочные данные поступают к месту назначения, то получателю передают их немедленно.

6.3.2. Протокол TCP

Каждый байт в TCP соединении имеет 32-разрядный номер. В сети с пропускной способностью 10 Мбит/сек. потребуется не менее часа, чтобы исчерпать все номера с 0 до 2^32. Эти номера используются как для уведомления, так и в механизме управления окнами. TCP-агенты обмениваются сегментами данных. Каждый сегмент имеет заголовок от 20 байтов и более (по выбору) и тело переменной длины. Один сегмент может включать байты от разных отправителей, а может включать часть данных одного. Какой длины может быть тело, решает TCP-агент. Длину сегмента ограничивают два фактора. Во-первых, длина сегмента не должна превышать максимальную длину IP-пакета - 64 Кбайт. Во-вторых, каждая сеть имеет максимальную единицу передачи - MTU (maximum transfer unit), и каждый сегмент должен помещаться в MTU. В противном случае маршрутизаторам придется применять фрагментацию. При этом возрастают накладные расходы на передачу в сети, так как каждый фрагмент оформляется как самостоятельный пакет (с 20-байтным заголовком). Основным протоколом, который используется TCP-агентом, является протокол скользящего окна. Это значит, что каждый посланный сегмент должен быть подтвержден. Одновременно с отправлением сегмента взводится таймер. Подтверждение придет либо с очередными данными в обратном направлении, если они есть, либо без данных, но с подтверждением. Подтверждение будет иметь порядковый номер очередного ожидаемого получателем сегмента. Если таймер исчерпается прежде, чем придет подтверждение, то сегмент посылается повторно. Несмотря на кажущуюся простоту, TCP-протокол достаточно сложен и должен решать следующие основные проблемы:

- восстанавливать порядок сегментов
- убирать дубликаты сегментов, в каком бы виде они не поступали
- определять разумную задержку для time_out для подтверждений в получении сегмента
- устанавливать и разрывать соединения надежно
- управлять потоком
- управлять перегрузками

6.3.3. Заголовок сегмента в TCP

Заголовок сегмента в TCP показан на рисунке 6-24. Максимальная длина раздела данных – 65 495 байтов.

- Поля Source port и Destination port указывают сокеты на стороне отправителя и получателя соответственно.

- Sequence number и Acknowledgement number содержат порядковый номер ожидаемого байта и следующего ожидаемого, а не последнего полученного байта.

- 6-битное поле флагов.

    - Бит Urg используется вместе с полем Urgent pointer, которое указывает на начало области срочных данных.
    - ACK - 1, если поле Acknowledgement number используется, в противном случае – 0.
    - PSH - 1, если отправитель просит транспортного агента на стороне получателя сразу передать эти данные приложению и не буферизовать их.
    - RST – используется, чтобы переустановить соединение, которое по какой-либо причине стало некорректным. Получение пакета с таким флагом означает наличие проблемы, с которой надо разбираться.
    - SYN – 1, при запросе на соединение. Флаг ACK указывает на наличие или отсутствие подтверждения. SYN=1 ACK=0 – запрос на соединение, SYN=1 ACK=1 – подтверждение соединения.
    - FIN - запрос на разрыв соединения. У отправителя нет больше данных.

- Поле Window size используется алгоритмом управления окном.
- Поле Options используется для установления возможностей, не предусмотренных стандартным заголовком. Например, здесь часто указывается максимальный размер поля данных, допустимый по данному соединению.

6.3.4. Управление соединениями в TCP

Как уже было сказано, установление TCP-соединения происходит по протоколу трехкратного рукопожатия. Флаги SYN и ASK в заголовке сегмента используются для реализации примитивов CONNECTION REQUEST и CONNECTION ACCEPTED. Флаг RST используется для реализации примитива REJECT. Это означает, что указанные выше примитивы вызывают посылку TCP-пакета с установленным соответствующим флагом.

На рисунке 6-17 (a) показаны состояния при установлении соединения. Когда приходит запрос на соединение по определенному порту, транспортный агент проверяет, есть ли процесс, который выполнил примитив LISTEN на этом порту. Если такой процесс есть, то ему передается управление. Если такого процесса нет, то в ответ идет отказ от установления соединения. Случай (b) показывает ситуацию, когда два хоста одновременно пытаются установить соединение между двумя одинаковыми сокетами (коллизия). Поскольку каждое соединение идентифицируется парой сокетов, то будет установлено только одно из соединений. Таймер для последовательных номеров сегментов тактируется с частотой 4 мксек., максимальное время жизни пакета - 120 сек. Напомним, что начальный номер сегментов никогда не равен нулю, по соображениям, приведенным ранее. Для генерации последовательных номеров сегментов используют механизм логических часов. TCP-соединение, как уже говорилось, - дуплексное, т.е. в каждом направлении данные передаются независимо и соединение разрывается независимо по каждому направлению. Поэтому лучше всего представлять его как два симплексных соединения. Если в очередном сегменте флаг FIN=1, то в этом направлении данных больше не будет. При получении подтверждения для этого сегмента соединение в этом направлении считается разорванным. В другом направлении передача может продолжаться сколь угодно долго. Если подтверждения на первый FIN нет в течение двух интервалов жизни пакетов, то по time-out соединение считается разорванным. Противоположная сторона также по истечении этого периода времени узнает, что никто от нее не ждет ответа. В таблице 6-18 и на рисунке 6-19 представлена процедура установления и разрыва соединения в виде диаграммы конечного автомата.

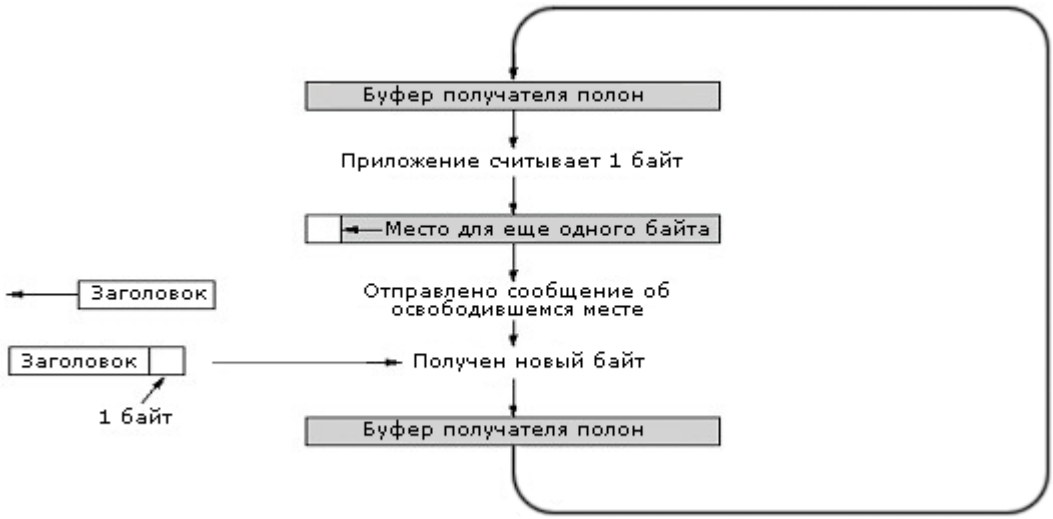
6.3.5. Стратегия передачи в TCP

Управление окнами в протоколе TCP, как в управлении потоком на канальном уровне, не связано прямо с поступлением подтверждений. Предположим, что у получателя есть буферы в 4096 байт, как показано на рисунке 6-20. Если отправитель послал сегмент в 2048 байт, то получатель, получив и подтвердив этот сегмент, будет показывать окно в 2048 байт до тех пор, пока приложение не возьмет часть данных из полученного сегмента в 2048 байт. Отправитель посылает следующие 2048 байт. Теперь размер окна равен 0

байт. Когда поле WIN=0, отправитель может послать сегмент в двух случаях. Первый - если это данные URGENT. Например, когда требуется убить процесс на удаленной машине. Второй - если это однобайтовый сегмент. Это может потребоваться, чтобы заставить получателя показать текущее состояние буфера, что очень важно, так как позволяет обойти тупик. Заметим, что протокол TCP не требует от агента-отправителя сразу передавать сегмент, как только данные поступили от приложения. Эту свободу TCP использует, чтобы повысить свою производительность. Рассмотрим, к примеру, удаленный редактор TELNET. Если передавать по сети каждое движение пользователя мышкой или нажатие им клавиши на клавиатуре, то обмен будет очень не эффективным. На передачу одного символа будет приходиться передача сегмента в 160 байт. Поэтому часто при реализации протокола TCP вводят специальную задержку на посылку подтверждения и состояния окна, чтобы дать отправителю накопить буфер для отправки. Другую стратегию предложил Нагл (Nagle) – если работа идет с приложением, которое генерирует однобайтные сообщения, то надо первый байт послать, а все остальные буферизовать до тех пор, пока не придет подтверждение на посланный байт. Все буферизованные байты нужно послать одним сегментом, после чего буферизовать все байты, пока не придет подтверждение на посланный сегмент. Алгоритм Нагла работает хорошо. Однако есть приложения, где его следует отключить, – X-Windows. Здесь перемещения мыши по экрану надо пересылать сразу без буферизации. Другая проблема, которая может существенно понизить производительность протокола TCP – т.н. «синдром дурацкого окна» (рисунок 6-21). Он возникает, когда приложение на стороне отправителя передает TCP-агенту данные большими блоками, а приложение на стороне получателя читает данные побайтно! В этой ситуации может произойти следующее. Буфер на стороне получателя полон и отправитель знает об этом. Приложение-получатель считывает один байт из буфера. TCP-агент получателя радостно шлет сообщение о доступном буфере в один байт. Отправитель обязан послать один байт. После чего буфер получателя опять полон, и т.д.

Кларк предложил запретить сообщать получателю в таких случаях об освободившемся месте на один байт. Получателя принуждают ждать, пока либо не освободится размер, равный максимальной длине сегмента, объявленной при установлении соединения, либо не освободится половина буфера. Отправитель также должен стараться избегать крохотных сегментов, а посылать большие. У TCP-агента получателя также есть средства улучшить производительность соединения. Например, можно запретить приложению использовать примитив READ до тех пор, пока у агента есть данные в буфере. Конечно, такое решение увеличит время отклика, но для неинтерактивных приложений это не страшно. Другая проблема для получателя, понижающая производительность соединения, - нарушение порядка поступления сегментов. Например, если поступили сегменты 0, 1, 2, 4, 5, 6, 7, получатель может подтвердить сегменты 0-2, забуферизовать сегменты 4-7. Тогда отправитель по time-out перешлет сегмент 3, после чего получатель подтвердит 4-7 сегменты.

Рисунок 6-21. «Синдром дурацкого окна»



6.3.6. Управление перегрузками в TCP

Здесь мы рассмотрим, как протокол TCP борется с перегрузками. В основе всех методов лежит принцип сохранения количества пакетов: не посылать новый, пока старый не покинет сеть, т.е. не будет доставлен. Основная идея очень проста - при возникновении перегрузки не посылать новых пакетов. В протоколе TCP это реализуется динамически с помощью механизма окон. Прежде всего, протокол TCP обнаруживает перегрузку по росту числа time_out. Если эта величина превышает некоторый предел, являющийся параметром протокола, то это фиксируется как перегрузка. Причин может быть две – шум в канале и сброс пакетов маршрутизатором. Различить их сложно. В наши дни каналы достаточно надежные, так что актуальной остается вторая причина. На рисунке 6-22 дана иллюстрация перегрузок. Перегрузки возникают по двум причинам: нехватка буфера на стороне получателя – недостаточная емкость получателя (a); перегрузка внутри сети – недостаточная емкость сети (b). В Internet эти ситуации различаются как внутренняя емкость сети и емкость получателя. Поэтому каждый отправитель поддерживает два окна - обычное окно отправителя и окно перегрузки. Каждое показывает количество байтов, которое отправитель может послать. Фактически отправляемое количество байтов - минимум из этих двух величин. Сначала окно перегрузки полагают равным размеру максимального сегмента для данного соединения. Если сегмент успешно (без time_out) был передан, то окно перегрузки увеличивается вдвое. Это увеличение будет происходить до тех пор, пока либо не наступит time_out и произойдет возврат к предыдущему значению, либо размер окна перегрузки не достигнет размера окна получателя. Этот алгоритм называется slow start - медленный старт. Другой параметр управления перегрузками в Internet – порог (threshold). Алгоритм медленного старта при возникновении перегрузки устанавливает этот параметр равным половине длины окна перегрузки, а окно перегрузки - равным размеру максимального сегмента. Окно перегрузки растет экспоненциально до тех пор, пока не сравняется с порогом, после чего оно растет линейно, пока не достигнет размера окна получателя. На этом рост прекращается до первой перегрузки. Работа этого алгоритма показана на рисунке 6-23.

6.3.7. Управление таймером в TCP

Протокол TCP использует несколько таймеров для управления передачей. Наиболее важный из них - таймер повторной передачи. Этот таймер устанавливают, когда отправляют сегмент. (Напомним, что так мы называем TPDU-пакет.) Если подтверждение пришло до исчерпания этого таймера, то его останавливают и сбрасывают. Если таймер исчерпан, то сегмент посылают повторно. Здесь основная проблема - как удачно выбрать величину time_out: временной интервал, по истечении которого сегмент надо передать повторно. Как мы знаем, эта проблема встречается и на других уровнях. Однако на транспортном уровне она имеет особенность, которая заключается в следующем. На канальном уровне дисперсия величины задержки подтверждения имеет ярко выраженный максимум (см. рисунок 6-24 (a)). Другими словами, ее разброс невелик. Величину time_out на этом уровне устанавливают чуть больше ожидаемой величины прихода подтверждения. На транспортном уровне функция распределения величины задержки подтверждения носит более гладкий характер, чем на канальном уровне (см. рисунок 6-24 (b)). Поэтому предсказать величину времени, которая нужна для передачи данных от источника до получателя и передачи подтверждения от получателя до источника, очень трудно. Заведомо эта величина не должна быть постоянной в силу гладкости функции распределения. В основе используемого в протоколе TCP алгоритма, предложенного Якобсоном в 1988 году, лежит специальная переменная RTT для получения оптимального значения величины time_out (Round Trip Time), значение которой постоянно модифицируется. В этой переменной хранится наименьшее время подтверждения. При каждой передаче сегмента замеряется величина задержки подтверждения M. Если при очередной передаче подтверждение поступило прежде, чем наступил time_out, значение переменной RTT немного уменьшают, в противном случае - увеличивают по формуле:

$RTT = \lambda RTT + (1- \lambda)M$, где $\lambda=0,87$.

Однако, даже зная величину RTT, определить величину ожидания оказалось непросто. Якобсон предложил вычислять величину D - отклонения между ожидаемой величиной задержки и измеренной по формуле:

$D = \lambda D + (1-\lambda) |RTT - M|$.

Кроме этого, Якобсон показал как, зная D, вычислить величину time_out по формуле:

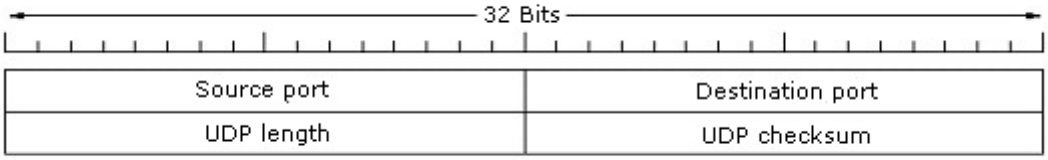
time_out = RTT + 4*D.

Позднее Филл Карн (Phill Karn) модифицировал эту формулу для случая повторно передаваемых сегментов. Действительно, когда поступает подтверждение для сегмента, то не ясно, то ли это подтверждение для первой передачи, толи для последней. Филла интересовал вопрос вычисления величины RTT для передачи данных по протоколам TCP/IP по КВ-радиоканалу. В результате экспериментов он показал, что для повторно передаваемых сегментов эффективно просто удваивать величину ожидания до тех пор, пока подтверждение не поступит с первого раза. Другой важный таймер в протоколе TCP - таймер настойчивости. Он позволяет бороться со следующего типа тупиками. Когда получатель посылает сообщение с нулевым размером окна, отправитель останавливает передачу и ждет сообщения об изменении размера окна. Наконец, получатель послал это сообщение, а оно было потеряно. Все ждут. Чтобы избежать такой ситуации, используют таймер настойчивости. Если он исчерпан, то отправитель шлет сообщение получателю, напоминая ему о проблеме размера буфера. Еще один важный таймер - таймер функционирования. Если по какой-либо причине по соединению долго не посылали сообщений, то надо проверить, функционирует ли оно. Когда этот таймер исчерпан, то соответствующая сторона шлет другой стороне запрос: «Жива ли ты?» Если ответа не поступает, то соединение считается разорванным.

6.3.8. Протокол UDP

Internet поддерживает также транспортный протокол без соединений - UDP (User Data Protocol). Протокол UDP (User Datagram Protocol) предназначен для обмена дейтаграммами между процессами компьютеров, входящих в единую сеть с коммутацией пакетов. В качестве протокола нижнего уровня UDP-протокол использует IP. Протокол UDP предоставляет прикладным программам возможность отправлять сообщения другим приложениям, используя минимальное количество параметров протокола. Этот протокол не обеспечивает достоверность доставки пакетов, защиты от дублирования данных или от сбоев в передаче. За исключением параметров приложения - номеров портов отправителя и получателя пакета, UDP практически ничего не добавляет к IP-дейтаграмме.

Рисунок 6-25. Заголовок UDP



Протокол UDP намного проще, чем TCP, и полезен в ситуациях, когда мощные механизмы обеспечения надежности протокола TCP не требуются или будут только помехой для решения определенного рода задач, например, аутентификации пользователей. Структура UDP-заголовка показана на рисунке 6-25.

- Source Port (16 бит). Порт отправителя. Это поле может содержать номер порта, с которого был отправлен пакет, когда это имеет значение (например, когда отправитель ожидает ответа). Если это поле не используется, оно заполняется нулями.
- Destination Port (16 бит). Порт назначения - это порт компьютера, на который пакет будет доставлен.
- Length (16 бит). Поле длины. Длина (в байтах) этой дейтаграммы, включая заголовок и данные. (Минимальное значение этого поля равно 8).
- Checksum (16 бит). Поле контрольной суммы. Контрольная сумма UDP-пакета представляет собой побитное дополнение 16-битной суммы 16-битных слов (аналогично TCP). В вычислении участвуют: данные пакета, заголовок UDP-пакета, псевдозаголовок (информация от IP-протокола), поля выравнивания по 16-битной границе (нулевые).

Преимущество протокола UDP состоит в том, что он требует минимум установок и параметров для соединения двух процессов между собой. Этот протокол используется при работе Серверов Доменов (Name Servers), протокола TFTP (Trivial File Transfer), при работе с SNMP-протоколом и построении систем

аутентификации. Идентификатор UDP в IP-заголовке - число 17. Более подробное описание протокола UDP можно найти в RFC-768.

6.3.9. TCP и UDP в беспроводных коммуникациях

Теоретически TCP не должен зависеть от того, над какой средой он работает – оптической или беспроводной. Однако на практике дело обстоит иначе. TCP-протокол тщательно оптимизировали при разных предположениях, которые не выполняются в беспроводной среде. Так, например, предполагалось, что сетевая среда достаточно надежна и рост числа time_out – это результат перегрузки, а не потери пакетов.

В беспроводной среде потеря пакета - дело частое. Поэтому применение протокола TCP «в лоб» с протоколом Якобсона медленного старта лишь усугубит положение. Так, например, если потери составляют 20%, то при пропускной способности канала в 100 пакетов в секунду фактически будем иметь лишь 80 пакетов. Согласно алгоритму медленного старта, при увеличении числа time_out надо понизить скорость, скажем до 50 пакетов в секунду, что приведет к фактической скорости 40 пакетов. Поэтому, если увеличилось число отказов в обычном канале, надо сбросить скорость, а если это число возросло в беспроводной среде, надо не понижать скорость, а слать пакеты повторно. Так что без знаний о среде принять решение трудно. Основным источником проблем является то, что в беспроводной среде соединения часто неоднородные. На рисунке 6-26 показан пример. Была предложена модификация TCP – разбить такое соединение на два так, чтобы каждое стало однородным. Есть и другие решения, связанные с модификацией не самого TCP, а канального уровня для базовых станций.

6.4. Вопросы производительности

Производительность вычислительных сетей и машин один из основных показателей их эффективности. Сегодня настройка производительности сетей и систем больше искусство, чем наука. Многое из того, что здесь будет сказано – результат практики. Мы уже много внимания уделили вопросам производительности при рассмотрении, например, сетевого уровня. Однако вопросы производительности сети в целом, как системы, относятся к транспортному уровню и будут рассмотрены здесь. В последующих пяти разделах мы рассмотрим пять основных аспектов производительности сетей:

1. Проблемы производительности
2. Измерение производительности
3. Влияние организации сетей на производительность
4. Быстрая обработка TPDU
5. Протоколы для высокопроизводительных сетей

6.4.1. Проблемы производительности в сетях

Одна из таких причин – перегрузки из-за несбалансированности ресурсов и нагрузки. Если на маршрутизаторы наваливается больше нагрузки, чем они могут переработать, возникает перегрузка. Перегрузки были достаточно подробно рассмотрены ранее. Производительность может падать из-за структурной несбалансированности ресурсов. Например, если персональную машину подключить к гигабитному каналу, то она будет захлебываться от наплыва пакетов из-за несбалансированности скорости процессора и скорости канала. Перегрузка может быть вызвана синхронно, как реакция на некоторые действия в сети. Это так называемые синхронные перегрузки. Например, если сегмент TPDU содержит неверный параметр (номер порта или процесса), то в ответ пойдет сообщение об ошибке. Если такой пакет получили сотни или тысячи машин, то в ответ последует ураган сообщений. Этой проблеме был подвержен протокол UDP до тех пор, пока не было внесено изменение в протокол, разрешающее или запрещающее слать подтверждение. Другой пример – перезагрузка машин в сети после сбоя питания. Все машины разом ринутся на RARP-сервер и файл-сервер за надлежащей информацией. В результате произойдет коллапс серверов. Другая причина – несоответствующая настройка системы. Например, если машины в сети имеют достаточно мощные процессоры и достаточно памяти, но под буфера в системе памяти выделено мало. В результате пакеты будут теряться из-за переполнения буферов. Аналогично, если планировщик процессов в операционной системе не дает достаточно высокий приоритет процессу обработки TPDU, то пакеты будут теряться. Другой параметр настройки – время time_out. Если time_out на подтверждение слишком короток, то

много будет повторных посылок, если велик – скорость передачи упадет. Еще один пример - время ожидания попутного сообщения, с которым можно отправить подтверждение о полученном пакете. Если значение этого параметра мало, то будет много дополнительных пакетов-уведомлений. Если велико, то получатель может генерировать запросы по time_out и скорость передачи упадет. Появление гигабитных сетей принесло новые причины потери производительности. Пусть мы хотим передать пакет из Москвы во Владивосток. У нас есть линия на 1 Гбит/сек., а у получателя - буфер на 64 Кбайт. Задержка на передачу в одном направлении по оптоволоконному каналу будет около 40 мсек. Через 500 мксек. все 64 Кбайт будут в канале, и отправитель будет ждать подтверждения. В результате пропускная способность канала будет использована на 1,25%! Дело в том, что буфер у получателя надо устанавливать равным произведению пропускной способности канала на величину задержки. А на практике он должен быть даже чуть больше. Получатель по какой-то причине может не сразу отреагировать на поступившие данные.

6.4.2. Измерение производительности в сети

Когда пользователи сети обнаруживают падение производительности их приложений, они идут к администратору сети с жалобами. Последний обязан выяснить, что случилось, и принять необходимые меры. Типичная последовательность действий при исправлении производительности сети такова:

1. Измерить надлежащие параметры сети и производительность
2. Постараться понять, что происходит
3. Изменить один параметр

Эти шаги надо повторять до тех пор, пока либо не удастся повысить производительность, либо не станет ясно, что имеющимися ресурсами этого сделать нельзя.

Измерения можно проводить в разных местах и разными способами. Основная идея всех измерений состоит в том, чтобы запустить какую-то активность и измерить, как долго она продолжается, какие события ее сопровождают. Измерение длительности и сбор информации о событиях таят много подвохов. Ниже перечислены лишь некоторые из них.

- Количество испытаний должно быть достаточно велико.
- Выборка испытаний должна быть представительной.
- Надо учитывать разрешающую способность часов.
- Ничего неожиданного во время измерений происходить не должно.
- Кэш-память может разрушить ваши измерения.
- Нужно четко осознавать, что вы измеряете.
- Надо быть очень осторожным при экстраполяции результатов.

Проведя измерения при определенной нагрузке, надо быть очень осторожным при их экстраполяции. Во многих случаях предположение о линейной экстраполяции может быть неверным

6.4.3. Правила, улучшающие производительность

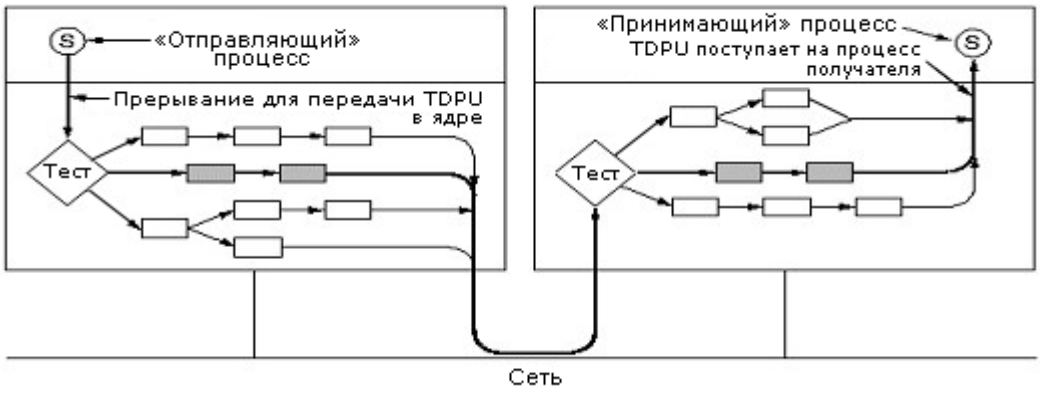
Повышать производительность существующей сети можно лишь в определенных пределах. Куда большие возможности для этого есть при проектировании сети. Ниже перечислены некоторые правила, сформулированные исключительно на опыте создания многих сетей.

- Правило 1: Скорость процессора важнее, чем скорость сети.
- Правило 2: Понижай число пакетов, чтобы сократить накладные расходы
- Правило 3: Минимизируй переключение контекста.
- Правило 4: Минимизируй число копий.
- Правило 5: Увеличение пропускной способности не сократит задержку.
- Правило 6: Лучше избегать перегрузок, чем восстанавливаться после них.
- Правило 7: Избегайте наступления time_out.

6.4.4. Быстрая обработка TPDU

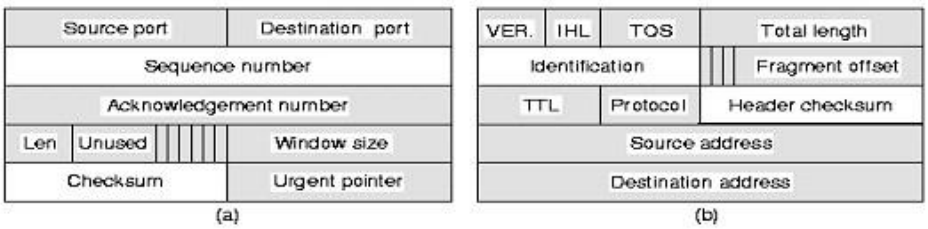
Вывод из вышесказанного один – основным препятствием для быстрой работы сети является программное обеспечение стека протоколов. Здесь мы рассмотрим некоторые способы ускорения работы этого программного обеспечения. Затраты на обработку TPDU делятся на затраты на обработку каждого TPDU и затраты на обработку каждого байта. Оба вида затрат могут и должны быть сокращены. Для ускорения обработки собственно TPDU можно использовать следующую идею. Основную долю TPDU-сегментов обрабатывают в режиме Established (см. рисунок 6-19). Поэтому важно максимально ускорить обработку TPDU в этом режиме. Для этого надо уметь быстро различать этот нормальный случай от остальных специальных, например, разрыва соединения. Рассмотрим пример. Для простоты будем предполагать, что транспортный агент расположен в ядре операционной системы (см. рисунок 6-28), хотя те же идеи применимы и для других случаев, например, когда транспортный агент является частью прикладного процесса или библиотечной функцией. На стороне отправителя прикладной процесс через программное прерывание передает TPDU транспортному агенту в ядре. Агент с помощью проверок определяет, во-первых, какой случай имеет место: нормальный – отправка TPDU или специальный – разрыв соединения, во-вторых, что оправляется регулярное TPDU, а не специальное, и в-третьих, что окно получателя имеет достаточный размер. Если все условия выполнены, то может быть запущен ускоренный процесс отправки.

Рисунок 6-28. Быстрая обработка TDPU



В нормальной ситуации заголовки последовательных TPDU сегментов почти одинаковы. Чтобы воспользоваться преимуществом этого факта, транспортный агент сохраняет прототип заголовка у себя при запуске процедуры быстрой обработки. Обычно это делается максимально быстро на регистровом буфере. Те поля TPDU, которые меняются, переписываются в буфере. Затем указатель на TPDU и указатель на тело данных передаются сетевому агенту на сетевой уровень. Там может быть применена та же схема, после чего сетевой агент передаст пакет на канальный уровень. Рассмотрим, как эта идея работает в случае TCP/IP. На рисунке 6-29 (a) показан заголовок TPDU. Светлым тоном выделены поля, которые меняют свои значения от сегмента TPDU к сегменту TPDU. Пять слов прототипа TPDU копируются в буфер, вычисляется контрольная сумма, увеличивается порядковый номер. IP-процедура, в свою очередь, копирует пять слов прототипа (см. рисунок 6-29 (b)) своего заголовка, заполняя соответствующими данными поля «Identification» и «Checksum».

Рисунок 6-29. (a) TCP-заголовок; (b) IP-заголовок



Теперь рассмотрим, что происходит на стороне получателя (правая часть рисунка 6-28). Прежде всего, транспортный агент на стороне получателя должен найти запись о соединении для поступившего сегмента TPDU. Для TCP-протокола эта запись может храниться в хеш-таблице. Ключом к этой таблице может служить информация о портах отправителя и получателя и их IP-адресах. Другой подход к поиску записи о соединении предложил Кларк – использовать последнюю использованную. Как показала практика, эта эвристика работает хорошо. После этого выполняются проверки, чтобы убедиться, что мы имеем дело с нормальным случаем, т.е. нет попытки разрыва соединения, нет URGENT-флага, и т.п. Если мы в состоянии Established, данные копируются приложению, при этом вычисляется контрольная сумма. Если она правильная, то корректируется запись о соединении, формируется подтверждение о получении и посылается отправителю. Описанная здесь в общих чертах схема носит название предвидения заголовка. Она используется во многих реализациях. Две другие области ускорения – управление буферизацией и таймерами. Основная идея ускорения при управлении буферизацией – избегать излишнего копирования. Управление таймерами состоит в том, что, хотя таймер устанавливается для каждого TPDU, срабатывает он лишь для немногих TPDU. Общая схема, оптимизирующая работу с таймерами, заключается в следующем. Записи о таймерах связываются в список. В очередном элементе списка указывают, сколько тактов от срабатывания предыдущего таймера должно пройти, чтобы сработал текущий. Поэтому, если есть три таймера, которые должны сработать в моменты 3, 10 и 12, то список будет выглядеть, как 3, 7, 2 соответственно. При такой организации достаточно корректировать при каждом такте не все таймеры, а только первую запись в списке. Другой прием, оптимизирующий работу с таймерами, называется колесо времени. Он использует массив (см. рисунок 6-30), длина которого пропорциональна максимальной длине временного интервала, который может возникнуть при работе. Каждый элемент в этом массиве соответствует одному такту часов.

Безопасность и способы защиты данных в сетях ЭВМ: методы шифрования. Обычное шифрование. Рассеивание и перемешивание. Два основных принципа шифрования. Алгоритмы с секретными ключами (Алгоритм DES, Раскрытие DES). Алгоритмы с открытыми ключами.

Проблема безопасности сети очень многогранна и охватывает широкий спектр вопросов. Большую их часть можно разделить на следующие группы:

1. Секретность

   - Конфиденциальность – только санкционированный доступ к информации (никто не может прочесть ваши письма без вашего ведома).
   - Целостность - только санкционированное изменение информации (никто без вашего разрешения не может изменить данные о вашем банковском счете).

2. Идентификация подлинности пользователей

   - Имея с кем-то дело через сеть, вы должны быть уверены, что это тот, за кого он себя выдает (если вы получили сообщение от налоговой инспекции уплатить определенную сумму денег, вы должны быть уверены, что это не шутка).

3. Идентификация подлинности документа

   - Получив через сеть электронную версию документа, как определить, что он подлинный и не был фальсифицирован?

4. Надежность управления

   - Несанкционированное использование ресурсов (если вы получите счет за телефонные переговоры, которые вы не делали, вам это вряд ли понравится).
   - Обеспечение доступности ресурсов для авторизованных пользователей.

На канальном уровне данные могут быть зашифрованы на одной машине и расшифрованы на другой. Об этом шифре верхние уровни могут ничего не знать. Однако, поскольку пакет дешифруется на каждом маршрутизаторе, то в памяти маршрутизатора он может стать предметом атаки. Тем не менее, при передаче данных этот метод, называемый шифрованием канала, часто применяется в сетях. На сетевом уровне распространенным решением является брандмауер (firewall). Напомним, что это средство, которое позволяет фильтровать как входящие, так и исходящие пакеты на сетевом уровне (см. главу 5). На транспортном уровне проблему секретности данных при передаче решают шифрованием всех сегментов транспортного соединения. Однако в сети до сих пор нет удовлетворительного решения проблемы идентификации пользователя и идентификации документа.

1. Обычное шифрование

Стандартная схема шифрования такова (см. рисунок 7-2). Исходный текст, называемый также открытым текстом (plain text), обрабатывают специальной функцией со специальным параметром, называемым ключом. В результате этой обработки получают так называемый шифр-текст (ciphertext), или криптограмму. Злоумышленник аккуратно копирует все шифр-тексты. Однако, в отличие от получателя, у него нет ключа, и он не может быстро прочесть сообщение. Иногда злоумышленник может не только копировать сообщение, но позже отправлять свои, имитируя настоящего отправителя, чьи сообщения он копировал. Такого злоумышленника называют активным. Искусство создания шифра называют криптографией, а вскрытия – криптоанализом. Обе эти дисциплины образуют криптологию. Один из способов повышения надежности шифра - шифрование на основе ключа. Ключ - относительно короткая строка текста, которая используется при шифровании и расшифровке сообщений. При этом, сама процедура, алгоритм шифрования могут быть известными. Тогда вся схема шифрования всем хорошо известна, менять ничего не надо, надо лишь время от времени изменять ключи. Основа секретности – ключ. Его длина – один из основных вопросов разработки. Рассмотрим, как пример, комбинационный цифровой замок. Все знают, что его ключ – последовательность цифр. Для замка из двух цифр надо перебрать 100 комбинаций, для 3 – 1000, и т.д. Длина ключа определяет объем работы, который надо проделать криптоаналитику, чтобы вскрыть шифр. Этот объем растет экспоненциально в зависимости от длины ключа. Секретность достигается не за счет сложности алгоритма, а за счет длины ключа. С точки зрения криптоаналитика проблема дешифровки возникает в трех вариантах:

- есть только шифрограмма
- есть шифрограмма и само сообщение
- есть фрагменты исходного сообщения и их шифрограммы

1. Шифрование замещением

Все приемы шифрования исторически делились на шифрование замещением и шифрование перестановкой. Шифрование замещением состоит в том, что буква или группа букв замещается другой буквой или группой букв.

2. Шифрование перестановкой

Шифрование перестановкой состоит в изменении порядка букв без изменения самих букв. Один из таких методов – шифрование по столбцам. Выбираем ключ – последовательность неповторяющихся символов. Символы в этой последовательности нумеруются в соответствии с их местом в алфавите. Для раскрытия этого типа шифров криптоаналитик, прежде всего, должен убедиться, что он имеет дело с шифрованием перестановкой. Для этого он должен подсчитать частоту встречаемости букв в шифре. Если она соответствует частоте букв в языке, то это означает, что он имеет дело именно с перестановкой. Намек на порядок столбцов могут дать устойчивые буквосочетания в языке.

3. Одноразовые подложки

Построить нераскрываемый шифр достаточно просто. Выберем случайным образом битовую строку – это будет ключ. Текст представим как битовую строку. Разделим эту строку по длине ключа. Выполним над полученными фрагментами строки операцию «исключающее ИЛИ» (EXCLUSIVE OR). Полученные фрагменты объединим в строку. Новая строка и есть криптограмма. Этот метод, называемый методом

одноразовой подложки, имеет ряд недостатков. Трудно запомнить ключ. Его где-то надо записать или носить с собой. Это делает метод уязвимым. Например, набор одноразовых подложек можно записать на CD и закамуфлировать под запись последних хитов. Объем шифруемых данных по этому методу ограничен длиной ключа. Метод также очень чувствителен к потере символов при передаче. Правда, используя возможности компьютеров, этот метод вполне можно применять в ряде случаев.

4. Рассеивание и перемешивание

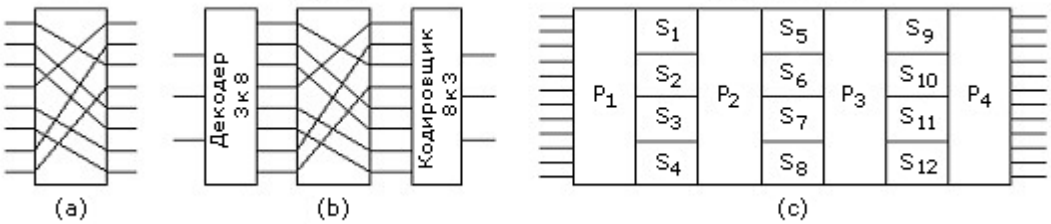
Основная угроза раскрытия текста при криптоанализе состоит в высокой избыточности естественного языка. Например, частотные характеристики встречаемости букв, устойчивые буквосочетания, приветствия и т.д. В связи с этим Шеннон предложил два основных криптографических метода: рассеивание и перемешивание. Цель рассеивания состоит в перераспределении избыточности исходного языка на весь исходный текст. Этот прием может быть реализован как перестановкой по некоторому правилу, так и замещением. Последнее, например, достигается тем, что замещающая комбинация зависит не только от замещаемой буквы, но от ей предшествующих букв. Цель перемешивания состоит в том, чтобы сделать зависимость между ключом и шифр-текстом настолько сложной, насколько это возможно. Криптоаналитик на основе анализа шифр-текста не должен получать сколь-нибудь полезной информации о ключе. Этот метод как раз реализуется с помощью перестановок. Следует учитывать, что применение порознь ни рассеивания, ни перемешивания не дает желаемого результата. Их совместное использование делает криптосистему намного более стойкой.

Два основных принципа шифрования

Первый: все шифруемые сообщения должны иметь избыточность, т.е. информацию, которая не нужна для понимания сообщения. Эта избыточность позволит нам отличить нормально зашифрованное сообщение от подсунутого. Например, если в заказах на поставку после имени заказчика стоит 3 байтовое поле заказа (2 байта – код продукта и 1 байт количество), зашифрованное с помощью ключа, то злоумышленник, прихватив с работы справочник заказчиков, может устроить компании «веселую жизнь». Для этого он сгенерирует от имени заказчиков из справочника заявки, где последние 3 байта - случайные числа. Если же длину заявки сделать не 3, а, например, 12 байтов, где первые 9 байтов - 0, и шифровать эту избыточную запись, то уже случайными числами здесь обойтись трудно, и подделку легко распознать. Однако такая избыточность имеет недостаток, эта избыточность может служить для криптоаналитика дополнительной информацией. Например, если в первом случае догадка о ключе и применение этого ключа к записи не дает криптоаналитику дополнительной информации о правильности ключа, то во втором случае, если в результате применение ключа-догадки, мы получим запись из 9 нулей с последующими данными, то это уже будет дополнительной информацией о правильности догадки. Второй – надо позаботиться о специальных мерах от активного злоумышленника, который может копировать, а потом пересылать модифицированные копии. Например, временная метка позволит обнаружить сообщения, которые где-то были задержаны по непонятным причинам.

Алгоритмы с секретными ключами

Рисунок 7-4. Основные составляющие шифрования: (a) P-схема; (b) S-схема; (c) Результат



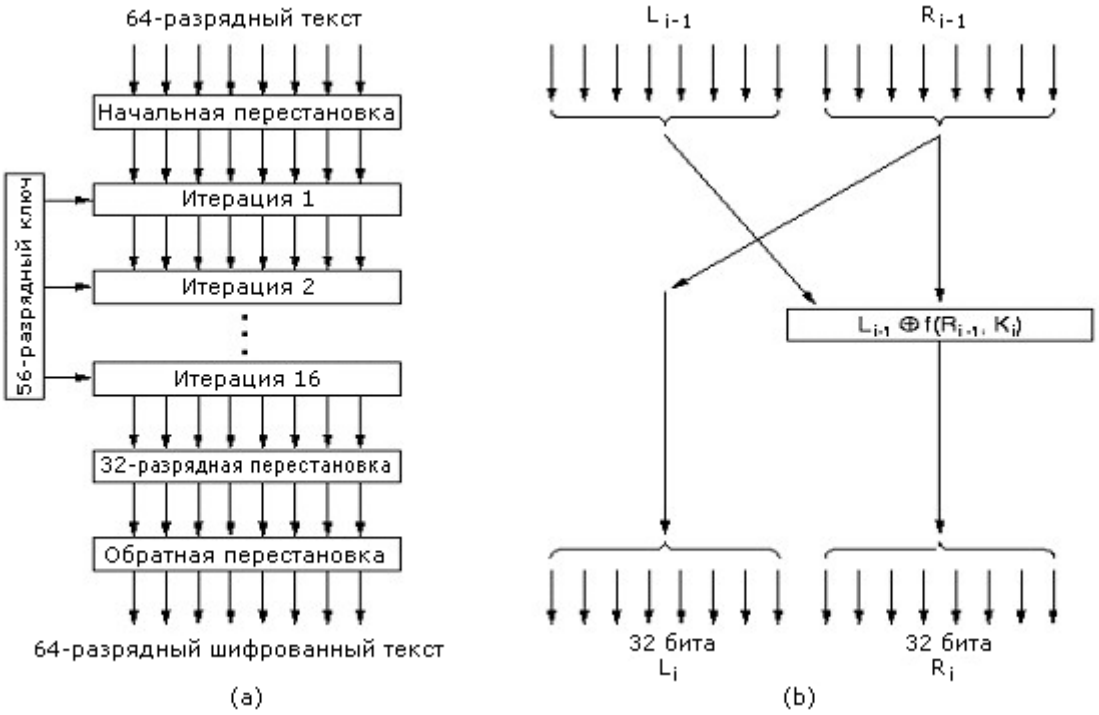
Перестановка и замещение реализуются простыми схемами, показанными на рисунке 7-4. P- и S-схемы могут объединяться в сложные каскады. В этом случае выход становится очень сложной функцией входа. На этом рисунке P-схема выполняет перестановку над словом из 8 разрядов. Схема S действует несколько сложнее,

выполняя операцию замещения. Она кодирует трехразрядное слово одной из 8 линий на выходе, устанавливая ее в 1. Затем схема P переставляет эти 8 разрядов, после чего S-схема выполняет замещение 8 на 3.

1. Алгоритм DES

В январе 1977 правительство США приняло стандарт в области шифрования (Data Encryption Standard), созданный на базе разработки фирмы IBM. Разница между этими разработками состояла в том, что DES предполагал 56-разрядный ключ, а у IBM он 128-разрядный. На рисунке 7-5 показана схема этого алгоритма.

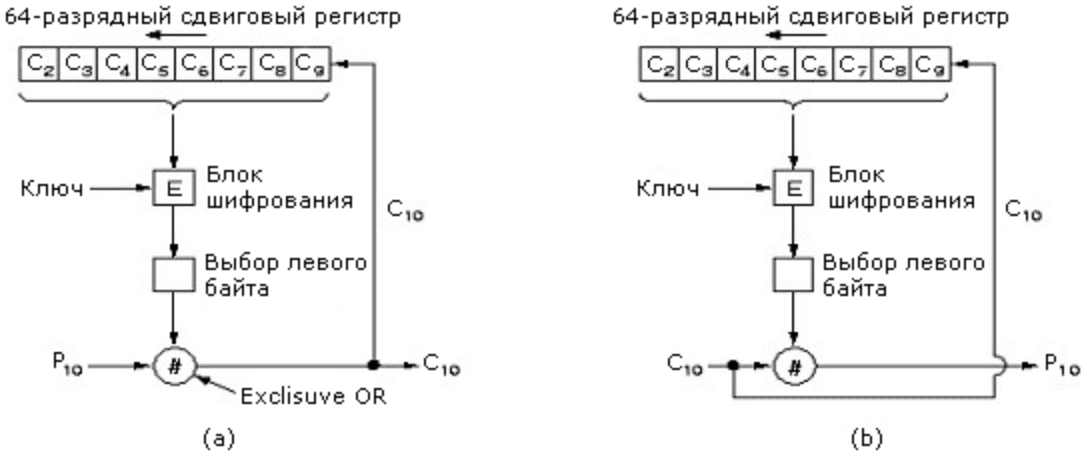
Рисунок 7-5. Стандарт DES: a) Общая схема; b) Действие одной итерации



Алгоритм состоит из 19 этапов. На первом этапе исходный текст разбивается на блоки по 64 бит каждый. Над каждым блоком выполняется перестановка. Последний этап является инверсией первой перестановки. Предпоследний этап состоит в обмене местами 32 самых левых битов с 32 самыми правыми битами. Из исходного ключа с помощью специального преобразования строят 16 частных ключей для каждого промежуточного этапа алгоритма со второго по семнадцатый. Все промежуточные этапы схожи. У них два входа по 32 бита. Правые 32 бита входа копируют на выход, как левые 32 бита. Над каждым битом из правых 32 битов выполняется преобразование, которое состоит из EXCLUSIVE OR с соответствующим левым битом и значением специальной функции над правым битом и частным ключом данного этапа. Вся сложность алгоритма заключена в этой функции. Функция состоит из четырех шагов. На первом строят 48-разрядный номер E как расширение 32-разрядного Ri-1 в соответствии с определенными правилами дублирования и перестановки. Над полученным номером и ключом данного шага выполняется операция EXCLUSIVE OR – это второй шаг. Результат разбивается на 8 групп по 6 бит в каждой. Каждая группа пропускается через свой S-box с четырьмя выходами каждый. На последнем шаге 8x4 бит пропускаются через P-box. На каждой из 16 итераций используется свой ключ. До начала итераций к исходному ключу применяют 56-разрядную перестановку. Перед каждой итерацией ключ разбивают на две части по 28 разрядов каждая. Каждую часть циклически сдвигают влево на число разрядов, равное номеру итерации. Ki получают как 48-разрядную выборку из 56-разрядного ключа, полученного циклическими сдвигами с дополнительной перестановкой. У предложенного алгоритма есть два недостатка. Он представляет собой моноалфавитное замещение с 64-разрядным символом. Всегда, когда одни и те же 64 разряда исходного текста подают на вход, одни и те же 64 разряда получают на выходе. Это свойство может использовать криптоаналитик. Одни и те же поля исходного текста попадут в одни и те же места шифра. Этим можно воспользоваться. Пример с премиями.

Один сотрудник может приписать себе премию другого, если знает взаимное расположение соответствующих полей в исходной записи. Второй недостаток – для начала шифрования надо иметь сразу весь 64-разрядный блок исходного текста. Это не совсем удобно, когда приходится иметь дело с интерактивными приложениями. Первый недостаток устраняется модификацией алгоритма DES, в которой очередной блок исходного текста через операцию EXCLUSIVE OR смешивается с шифром предыдущего блока. Для первого блока используется специальный инициирующий блок, который передается вместе с шифром. Для устранения второго недостатка используется т.н. обратная связь по шифру (рисунок 7-8

Рисунок 7-8. Обратная связь по шифру



Раскрытие DES

Существует много способов атаковать шифр. Все они основаны на распараллеливании перебора множества возможных ключей. Основной вывод - DES не является надежным шифром и его нельзя использовать для ответственных документов.. Эти рассуждения наталкивают на идею увеличения длины ключа за счет двукратного применения DES с разными ключами К1 и К2. Однако в 1981 году Хеллман и Меркл обнаружили, что простое использование двух ключей не дает надежной схемы. Дело в том, что применение дешифрации к зашифрованному тексту дает шифр, который получается после применения первого ключа к исходному тексту. Эту мысль поясняют следующие формулы:

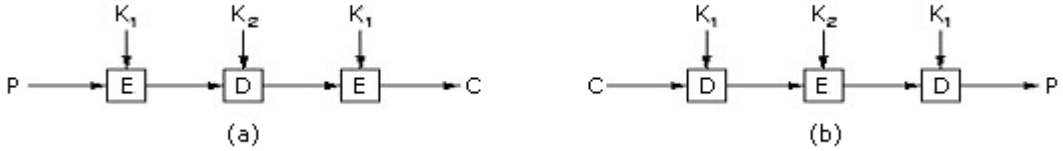
Ci = EK2(EK1(Pi)); DK2(Ci) = EK1(Pi)

В этом случае можно предложить следующую процедуру взлома:

1. Вычислить все возможные применения функции E к шифруемому тексту.
2. Вычислить все возможные дешифрации зашифрованного текста однократным применением дешифрирующей функции.
3. Отсортировать полученные таблицы и искать совпадающие строки.
4. Полученная пара номеров строк – пара ключей.
5. Проверить эту пару на совпадение шифрования; если неудачный результат, продолжить с шага 1.

Тройное шифрование совершенно меняет дело. На рисунке 7-9 показана модификация схемы шифрования с двумя ключами в три этапа - EDE-схема. Ее никому еще не удалось вскрыть. Она была положена в основу международного стандарта. Здесь может возникнуть два вопроса. Первый: почему в этой схеме используются 2, а не 3 ключа. Второй: почему используется схема EDE, а не EEE? Ответ на первый вопрос состоит в том, что двух ключей более чем достаточно для большинства применений. Использование схемы EDE вместо EEE связано с особенностями организации алгоритма DES.

Рисунок 7-9. Тройное шифрование с помощью алгоритма DES



Надо отметить, что было предложено много других алгоритмов шифрования.

Хорошо известным является международный алгоритм шифрования данных IDEA. Он был разработан специалистами из Швейцарии, и в нем используют 128-разрядный ключ. Подобно DES, этот алгоритм разбивает исходный текст на 64-разрядные блоки, над которыми производят определенные итерации, каждая из которых имеет параметры. В результате на выходе получают 64-разрядный блок, как показано на рисунке 7-10 (а). На каждой итерации значение каждого выходного бита зависит от всех входных битов, поэтому достаточно 8 итераций, а не 19, как в DES.

Алгоритмы с открытыми ключами

Идея алгоритмов шифрования с открытыми ключами была предложена в 1976 году Диффи и Хеллманом и состоит в следующем. Пусть у нас есть алгоритмы E и D, которые удовлетворяют следующим требованиям:

- D(E(P))=P
- Чрезвычайно трудно получить D из E.
- E нельзя вскрыть через анализ исходных текстов.

Алгоритм шифрования E и его ключ публикуют или помещают так, чтобы каждый мог их получить, алгоритм D также публикуют, чтобы подвергнуть его изучению, а вот ключи к последнему хранят в секрете. В этом случае взаимодействие двух абонентов A и B будет выглядеть следующим образом. Пусть A хочет послать B сообщение P. A шифрует EB(P), зная алгоритм и открытый ключ для шифрования. B, получив EB(P), использует DB с секретным ключом, т.е. вычисляет DB(EB(P))=P. Никто не прочтет P кроме A и B, т.к. по условию алгоритм EB не раскрываем по условию, а DB не выводим из EB.

Примером такого алгоритма является алгоритм RSA. Общая схема этого алгоритма такова:

1. Выберем два больших (больше 10100) простых числа p и q.
2. Вычислим n=pxq и z=(p-1)x(q-1).
3. Выберем d, относительно простое к z.
4. Вычислим e такое, что exd=1 mod z.

Разбиваем исходный текст на блоки P так, чтобы каждый блок, как число не превосходил n. Для этого выбираем наибольшее k такое, чтобы P=2k<n. Вычисляем C=Pe(mod n), чтобы зашифровать сообщение P. Для расшифровки вычисляем P=Cd(mod n).

Для шифрования нам нужны (e, n) – это открытый ключ, для расшифровки (d, n) – это закрытый ключ. Можно доказать, что для любого P в указанном выше диапазоне функции шифрования и дешифрования взаимообратные. Безопасность этого метода основана на высокой вычислительной сложности операции разложения на множители больших чисел. Так, например, разложение на множители 200-разрядного числа потребует 4 миллиардов лет.

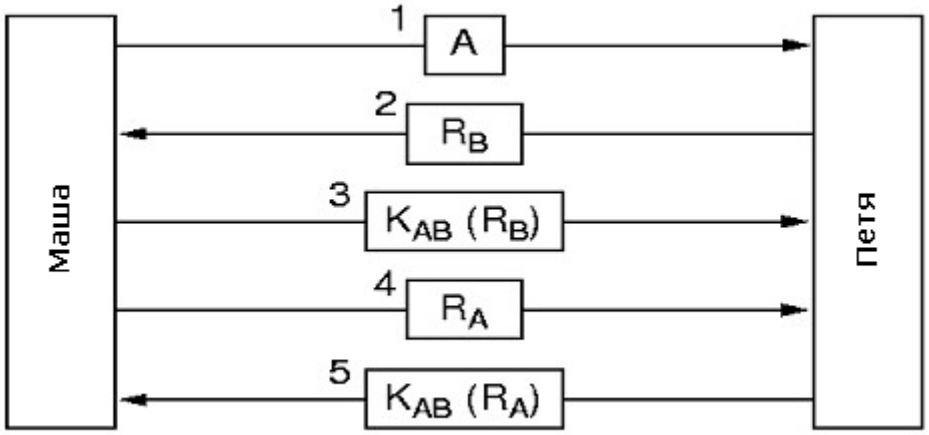
Безопасность и способы защиты данных в сетях ЭВМ: протоколы установления подлинности документов и пользователей (аутентификация на основе закрытого разделяемого ключа, установка разделяемого ключа, проверка подлинности через центр раздачи ключей, установление подлинности протоколом Цербер, установление подлинности, используя шифрование с открытым ключом). Электронная подпись (подпись с секретным ключом, подпись на основе открытого ключа). Сокращение сообщения.

Протоколы установления подлинности (аутентификации) позволяют процессу убедиться, что он взаимодействует с тем, с кем должен, а не с тем, кто лишь представляется таковым. Очень часто путают проверку прав на выполнение тех или иных операций с аутентификацией. (В первом случае имеем авторизацию.) Аутентификация отвечает на вопрос: как убедиться, что вы взаимодействуете именно с нужным процессом. Если, например, к серверу процесс обратился с запросом удалить файл x.old и объявил себя процессом Вася, то сервер должен убедиться, что перед ним действительно Вася и что Вася имеет право делать то, что просит. Ключевым конечно является первый вопрос, ответ на второй вопрос – это дело просмотра таблицы. Общая схема всех протоколов аутентификации такова: сторона А и сторона В начинают обмениваться сообщениями между собой или с Центром раздачи ключей (ЦРК). ЦРК - всегда надежный партнер. Протокол аутентификации должен быть устроен так, что даже если злоумышленник перехватит сообщения между А и В, то ни А, ни В не спутают друг друга со злоумышленником.

7.1.5.1. Аутентификация на основе закрытого разделяемого ключа

Основная идея первого протокола аутентификации, так называемого «протокола ответ по вызову», состоит в том, что одна сторона посылает некоторое число (вызов), другая сторона, получив это число, преобразует его по определенному алгоритму и отправляет обратно. Посмотрев на результат преобразования и зная исходное число, инициатор может судить, правильно сделано преобразование или нет. Алгоритм преобразования является общим секретом взаимодействующих сторон. Будем предполагать, что стороны А и В имеют общий секретный ключ KAB. Этот секретный ключ взаимодействующие стороны как-то установили заранее, например, по телефону. Описанная выше процедура показана на рисунке 7-12.

Рисунок 7-12. Двусторонняя аутентификация с запросом и подтверждением



На этом рисунке:

A,B - идентификаторы взаимодействующих сторон

Ri - вызов, где индекс указывает, кто его послал

Kj - ключ, индекс которого указывает на его владельца

Есть несколько общих правил построения протоколов аутентификации (протокол проверки подлинности или просто подлинности):

1. Инициатор должен доказать, кто он есть, прежде чем вы пошлете ему какую-либо важную информацию.

2. Инициатор и отвечающий должны использовать разные ключи.

3.Инициатор и отвечающий должны использовать начальные вызовы из разных непересекающихся множеств.

7.1.5.2. Установка разделяемого ключа

До сих пор мы предполагали, что А и В имеют общий секретный ключ. Рассмотрим теперь, как они могут его установить. Например, они могут воспользоваться телефоном. Однако, как В убедится, что ему звонит именно А, а не злоумышленник? Можно договориться о личной встрече, куда принести паспорт и прочее, удостоверяющее личность. Однако есть протокол, который позволяет двум незнакомым людям установить общий ключ даже при условии, что за ними следит злоумышленник.

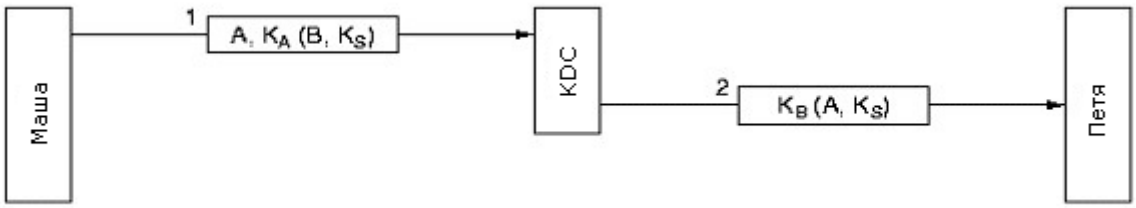
Это протокол обмена ключом Диффи-Хеллмана. Его схема показана на рисунке 7-15. Прежде всего, А и В должны договориться об использовании двух больших простых чисел n и g, удовлетворяющих определенным условиям. Эти числа могут быть общеизвестны. Затем, А выбирает большое число, скажем x, и хранит его в секрете. То же самое делает В. Его число – y. Злоумышленник следит за всем этим. Единственно, что мешает ему вычислить x и y, – это то, что не известно алгоритма с приемлемой сложностью для вычисления логарифма от модуля для простых чисел. Однако и у этого алгоритма есть слабое место. Рисунок 7-16 показывает его. Такой прием называется чужой в середине.

7.1.5.3. Проверка подлинности через центр раздачи ключей

Договориться с незнакомцем об общем секрете можно, но вряд ли это следует делать сразу. Кроме этого, общение с n людьми потребует хранения n ключей, что для общительных или популярных личностей может быть проблемой.

Другое решение можно получить, введя надежный центр раздачи ключей (KDC). Его использование иллюстрирует рисунок 7-17.

Рисунок 7-17. Проверка подлинности через центр раздачи ключей



Идея этого протокола состоит в следующем. А выбирает ключ сессии KS. Используя свой ключ KA, А шлет в центр KDC запрос на соединение с В. Центр KDC знает В и его ключ KB. С помощью этого ключа KDC сообщает В ключ сессии KS и информацию о тех, кто хочет с ним соединиться. Однако решение с центром KDC имеет изъян. Пусть злоумышленник как-то убедил А связаться с В и скопировал весь обмен сообщениями. Позже он может воспроизвести этот обмен за А и заставить В действовать так, как если бы с В говорил А! Этот способ атаки называется атака подменой. Против такой атаки есть несколько средств. Одно из них – временные метки. Это решение, однако, требует синхронизации часов. Поскольку в сети всегда есть расхождение в показаниях часов, то надо будет задать определенный интервал, в течение которого допустимо считать сообщение верным. Злоумышленник может использовать прием атаки подменой в течение этого интервала. Другое решение - использование разовых меток. Однако при этом каждая из сторон должна помнить все разовые метки, использованные ранее. Это обременительно. Кроме этого, если список использованных разовых меток будет утерян по каким-либо причинам, то весь метод перестанет работать. Можно комбинировать решения разовых меток и временных меток. Более тонкое решение установления подлинности дает многосторонний протокол «вызов-ответ». Хорошо известным примером такого протокола является протокол Нидхема-Шредера, вариант которого показан на рисунке 7-18. Вначале А сообщает KDC, что он хочет взаимодействовать с В. KDC сообщает ключ сессии, разовую метку RA, шифруя сообщение

ключом А. Разовая метка защищает А от подмены. Теперь, имея ключ сессии, А начинает обмен сообщениями с В. RA2 и RB – разовые метки, защищающие А и В от подмен.

Хотя этот протокол в целом надежен, но и при его использовании есть небольшая опасность. Если злоумышленник раздобудет все-таки старый ключ сессии, то он сможет подменить сообщение 3 старым и убедить В, что это А! На рисунке 7-19 приведена схема исправленного протокола, который предложили Отвей и Риис. В этой модификации KDC следит, чтобы R было одним и тем же в обеих частях сообщения 2.

7.1.5.4. Установление подлинности протоколом «Цербер»

Протокол установления подлинности «Цербер» используется многими действующими системами. Он представляет собой вариант протокола Нидхема-Шредера и был разработан в MIT для безопасного доступа в сеть (предотвращения несанкционированного использования ресурсов сети). В нем используется предположение, что все часы в сети хорошо синхронизованы.

Протокол «Цербер» предполагает использование кроме рабочей станции А еще трех серверов:

- Сервер установления подлинности (СП) – проверяет пользователей на этапе login.
- Сервер выдачи квитанций (СВБ) – идентификация квитанции.
- Сервер В – тот кто должен выполнить работу, необходимую А.

СП аналогичен KDC и знает секретный пароль для каждого пользователя. СВБ выдает квитанции, которые подтверждают подлинность заказчиков работ.

Теперь А может обращаться непосредственно к В с этим ключом. Это взаимодействие сопровождается временными метками, чтобы защититься от подмены. Если позднее А понадобится работать с сервером С, то А должен будет повторить сообщение 3, но указать там сервер С. Поскольку сеть может быть очень большой, то нельзя требовать, чтобы все использовали один и тот же СП. Сеть разбивают на области, в каждой из которых - свои СП и СВБ, которые взаимодействую между собой.

7.1.5.5. Установление подлинности с шифрованием с открытым ключом

Установить взаимную подлинность можно с помощью шифрования с открытым ключом. Пусть А и В уже знают открытые ключи друг друга. Они их используют, чтобы установить подлинность друг друга, а затем используют шифрование с секретным ключом, которое на несколько порядков быстрее. Единственным слабым местом этого протокола является предположение, что А и В уже знают открытые ключи друг друга. Обмен такими ключами уязвим для атаки типа «чужой в середине». Ривст и Шамир предложили протокол, защищенный от атаки «чужой в средине». Это так называемый протокол с внутренним замком. Его идея заключается в том, чтобы передавать сообщения в два этапа: сначала только четные биты, затем нечетные.

Электронная подпись

Подлинность многих юридических, финансовых и прочих документов устанавливается наличием подписи уполномоченного лица. Поскольку есть способы отличить фотокопии от подлинника, то фотокопии не рассматриваются. Такая же проблема возникает для документов в электронной форме.

Проблема электронного аналога для ручной подписи весьма сложна. Нужна система, которая позволяла бы одной стороне посылать «подписанный» документ другой стороне так, чтобы:

1. Получатель мог удостовериться в подлинности отправителя.
2. Отправитель позднее не мог отречься от документа.
3. Получатель не мог подделать документ.

Первое требование важно, например, при взаимодействии с банком, чтобы убедиться, что тот, кто проводит операцию, действительно является владельцем счета. Второе требование нужно в случаях, когда, например,

клиент запросил закупить тонну золота, цена которого после этого на бирже неожиданно упала. У клиента может возникнуть соблазн отказаться от своей заявки. Третье требование предотвращает ситуации типа следующей: цена на золото в предыдущем примере неожиданно подскочила, тогда у банка может появиться соблазн изобразить, что клиент просил купить не тонну, а, скажем, килограмм золота.

1. Подпись с секретным ключом

Одно из решений проблемы электронной подписи – наделить полномочиями третью сторону, которую знают все, которая знает всех и которой верят все. Назовем ее «Большой Брат» (Big Brother - BB). Что произойдет, если A позже откажется от посланного сообщения? B суде B предъявит сообщение A, KBB (A, t, P), которое будет отправлено BB как непререкаемому авторитету. СД расшифрует своим ключом эту запись и все увидят A, t, P. Единственная слабость такого решения – злоумышленник может скопировать диалог между отправителем и получателем через BB и позже его повторить. Механизм временных меток позволяет уменьшить эту проблему. Кроме этого, сохранение последних RA позволяет B заметить их повторное использование.

7.1.6.2. Подпись на основе открытого ключа

Недостаток вышеописанного решения в том, что все должны доверять СД, который может читать сообщения. Кандидатами на его роль может быть правительство, банк, нотариус. Здесь есть два недостатка. Оба основаны на том, что схема работает до тех пор, пока сторона A либо умышленно не рассекретила свой ключ, либо не изменила его в одностороннем порядке. При наступлении судебного случая B предъявляет P и DA(P), так как он не знает закрытый ключ A, то он не мог подделать DA(P). При этом должно быть EA(DA(P))=P, в чем суд легко может убедиться.

Если A обращается в суд, то есть P и EA(P), что легко сопоставить с тем, что есть у B. Однако, если A заявит, что у него украли ключи, а сам тайно передаст их, либо сменит их, не сообщив об этом B, то в последнем случае текущий EA будет не применим к тому DA(P), который предъявит B. Здесь надо сопоставлять даты передачи сообщения и смены ключей.

7.1.6.3. Сокращение сообщения

Рассмотренные методы электронной подписи критикуют за то, что они подменяют задачу установления подлинности задачей соблюдения секретности. Зачастую нужно только установление подлинности. Кроме этого, шифрование – операция медленная. Поэтому часто желательно просто поставить подпись, чтобы удостоверить подлинность сообщения. Ниже рассматривается метод, который не требует шифрования всего сообщения. Эта схема основана на идее однозначной функции перемешивания, которая по сообщению вычисляет битовую строку фиксированной длины. Эта функция называется сокращение сообщения (MD). Она обладает тремя важными свойствами:

1. Для заданного P вычислить MD(P) просто.
2. Имея MD(P), невозможно восстановить P.
3. Никто не сможет подобрать таких два сообщения, что MD от них будут одинаковыми.

Этот метод можно применять как с закрытым ключом, так и с открытым. В случае закрытого ключа, как в схеме на рисунке 7-22, надо пятый элемент в сообщении от BB к B заменить на KBB(A, t, MD(P)). «Большой брат» подписывает не все сообщение, а лишь его сокращение. В случае возникновения спора B легко докажет в суде, что он получил именно сообщение P, так как нет другого сообщения, которое даст такое же MD(P). Сторона A также легко докажет свою правоту, так как MD вычисляет BB. В случае открытого ключа этот метод также работает. Его схема показана на рисунке 7-24. Здесь сторона A шифрует своим ключом не все сообщение, а лишь его сокращение. При этом сторона B защищена от злоумышленника, так как если он подменит сообщение, то сторона B, получив сообщение и вычислив его MD, сравнит полученное MD и обнаружит подмену.

Служба DNS. Организация, функционирование и основные протоколы почтовой службы в Internet. Протокол FTP.

Несмотря на то, что любая машина в Интернете имеет уникальный IP-адрес, который представляет собой число (см. главу 5), люди предпочитают работать с именами. Очень непросто разговаривать, используя машинную адресацию (как бы это звучало: “192.112.36.5 обещает вскоре…”?), еще труднее запомнить эти адреса. Поэтому компьютерам в Интернете для удобства пользователей были присвоены собственные имена. Все интернет-приложения позволяют пользоваться системными именами вместо числовых адресов. Все эти имена зафиксированы в специальной распределенной базе данных DNS, которая поддерживает иерархическую систему имен для идентификации абонентских машин или узлов в сети Интернет.

7.2.2. Поиск адреса по доменному имени

Теперь, после того как мы узнали, как соотносятся домены и создаются имена, познакомимся с тем, как использовать эту замечательную систему. Она работает автоматически. Нам не надо разыскивать адрес, соответствующий имени или подавать специальную команду для его поиска (в UNIX – команда nslookup). Все компьютеры в Интернете способны пользоваться доменной системой. Когда используют имя, например, www.lvk.cs.msu.su, надо преобразовать его в адрес. Для этого приложение начинает запрашивать помощь у DNS-серверов. Эти приложения обладают соответствующей базой данных, в число обязанностей которых входит обслуживание такого рода запросов. DNS-сервер начинает обработку имени с его правого конца и двигается по нему влево, т.е. сначала производится поиск адреса в самой верхней группе иерархии, потом постепенно поиск опускается по иерархии, тем самым сужая область поиска. Однако с целью сокращения поиска, на первом шаге опрашивается локальный узел DNS. В последнем случае местный сервер обращается к корневому серверу. Это сервер, который знает адреса серверов имен высшего уровня (самых правых в имени), в нашем случае это уровень государств (ранга домена su). У него запрашивается адрес компьютера, ответственного за зону su. Местный DNS-сервер связывается с этим сервером, расположенным на вершине иерархии, и запрашивает у него адрес сервера, ответственного за домен msu.su. Теперь уже запрашивается сервер, отвечающий за домен msu, потом опрашивается сервер домена cs, затем – lvk, и у него запрашивается адрес рабочей машины www. Как уже было сказано, для повышения эффективности поиск начинается не с самого верха, а с наименьшего домена, в который входите и вы, и компьютер, имя которого вы запросили. Например, если ваш компьютер имеет имя cmc.cs.msu.su, то опрос начнется (если имя не выяснится сразу) не со всемирного сервера, чтобы узнать адрес сервера группы su, а сразу с группы su, что сокращает поиск и по объему, и по времени.

7.2.3. Серверы имен

Должно быть ясно, что нет и не может быть единого сервера, содержащего всю базу DNS. Чтобы сделать базу распределенной, все пространство имен доменов разбивают на непересекающиеся зоны. Границы зоны определяет администратор зоны. Каждая зона покрывает часть дерева доменов, в нее входят сервера имен этих доменов. Обычно в каждой зоне есть основной сервер зоны и несколько вспомогательных серверов имен. Часто из соображений надежности сервер зоны располагают вне зоны. Весь процесс поиска IP-адреса по имени домена, описанный в разделе 7.2.2., реализуют сервера имен. Если запрос относится к юрисдикции того сервера имен, к которому обратились, т.е. запрашиваемый домен находится в ведении данного сервера имен, тогда этот сервер генерирует ответ, содержащий записи всех ресурсов, соответствующих запросу. Этот ответ считается авторитетным, т.е. содержащаяся в нем информация считается a priori верной. Если запрос относится к удаленному домену, то сервер имен генерирует запрос к соответствующему удаленному серверу имен. Однако прежде чем обратиться к удаленному серверу имен, обращающийся сервер посмотрит записи ресурсов в своей кэш-памяти. Записи в кэш-памяти не являются авторитетными. Время актуальности содержащейся в них информации, определяет поле времени жизни (см. назначение поля времени жизни в разделе 7.2.4.).

7.2.4. Записи ресурсов

С каждым доменом связано множество ресурсов, отнесенных к этому домену. Эти записи хранятся в базе DNS. Когда происходит обращение к DNS с каким-либо именем, в ответ приходит не только IP-адрес, но и запись о ресурсах, соответствующих указанному имени. Запись ресурса состоит из пяти полей (см. рисунок

7-27). Вот эти поля: «Имя домена» (Domain name), «Время жизни» (Time to live), «Класс» (Class), «Тип» (Type), «Источник полномочий» (SOA - Start Of Authority). В поле «Имя домена» указано имя домена, к которому относится эта запись. При обращении к базе DNS с таким ключом в ответ поступают все записи, у которых в этом поле указано заданное имя. Поля «Время жизни» указывает интервал времени в секундах, в течение которого поля этой записи не меняются. Например, 86 400 - это число секунд в сутках. Если в этом поле указано такое число, то это значит, что запись меняется не чаще одного раза в сутки. В третьем поле «Класс» указано IN, если ресурс, к которому относится эта запись, является ресурсом Интернета. Здесь могут быть и другие значения, но они встречаются редко.

7.4. Электронная почта

7.4.1. Архитектура и сервис

Архитектура почтовой системы состоит из двух основных компонентов - агента пользователя и агента передачи сообщений. Первый отвечает за интерфейс с пользователем, составление и отправку сообщений. Второй – за доставку сообщения от отправителя к получателю.

7.4.2. Агент пользователя

Агент пользователя – обычно программа, имеющая множество команд для получения, составления сообщения и ответа на сообщение, а также для работы с почтовым ящиком. Некоторые агенты используют командную строку, некоторые - графический интерфейс.

7.4.2.1. Отправка почты

Чтобы послать сообщение, пользователь должен предоставить адрес назначения, само сообщение и другие параметры, например, приоритетность, секретность и т.п. Для создания сообщения может быть использован любой текстовый процессор либо текстовый редактор, встроенный в агент пользователя. Все параметры должны быть заданы в формате, который понимает и с которым может работать агент пользователя. Большинство агентов пользователя ожидает адрес назначения в формате DNS: mailbox@location.

7.4.2.2. Чтение почты

Прежде чем агент пользователя (далее АП) что-либо высветит на экране при загрузке, он просмотрит почтовый ящик на предмет того, что нового туда поступило. Затем он высветит на экране его содержимое с краткой аннотацией каждого сообщения. В таблице 7-49 показан пример содержимого почтового ящика в том виде, как это предоставляет АП. В простых почтовых АП высвечиваемые поля встроены в АП, в развитых – пользователь сам определяет, что показывать, а что нет. После того как содержимое ящика показано пользователю, последний может выполнять любую из имеющихся команд.

7.4.3. Формат сообщений

Рассмотрим формат самого сообщения. Начнем с формата по RFC 822, а потом перейдем к мультимедийному расширению этого документа.

7.4.4. Передача сообщений

Основная задача системы передачи почтовых сообщений – надежно доставить сообщение от отправителя к получателю. Самый простой способ сделать это – установить транспортное соединение и по нему передавать почту.

7.4.4.1. SMTP (Simple Mail Transfer Protocol) – Простой протокол передачи почты

В Internet почта передается следующим образом. Машина отправитель устанавливает TCP-соединение с 25-м портом машины получателя. На 25 порту находится почтовый демон, который работает по протоколу SMTP. Он принимает соединение и распределяет поступающие сообщения по почтовым ящикам машины-получателя. SMTP - это простой ASCII-протокол. После установления соединения машина-отправитель

работает как клиент, а машина-получатель – как сервер. Сервер шлет текстовую строку, идентифицирующую его и готовность принимать почту. Если он не готов принимать почту, то клиент разрывает соединение и повторяет всю процедуру позже. Если сервер подтвердил свою готовность, то клиент сообщает, от кого и кому предназначено очередное сообщение. Если сервер подтвердил наличие получателя, то он дает команду клиенту, и сообщение передается без контрольных сумм и подтверждений, так как TCP-соединение обеспечивает надежный поток байтов. Если сообщений несколько, то все они передаются. Обмен по соединению происходит в обоих направлениях. У SMTP протокола, несмотря на то что он хорошо описан в RFC 821, есть несколько проблем. Первая – длина сообщения не может превосходить 64 Кбайт. Другая – time out. Если время ожидания подтверждения у отправителя и получателя не согласовано, то один будет разрывать соединение, не дождавшись, тогда как другой просто очень загружен. Третья – почтовый ураган. Пусть машина-получатель имеет лист рассылки, где указана машина-отправитель, и наоборот. Тогда отправка сообщения по листу рассылки вызовет бесконечно долгие обмены сообщениями между этими машинами. Для преодоления этих проблем в RFC 1425 был описан протокол ESMTP. Клиент вначале шлет команду EHLO, если она отвергается сервером, то это означает, что сервер работает по SMTP.

7.4.4.2. Почтовые шлюзы

Протокол SMTP хорош, когда обе машины находятся в Internet. Однако это не всегда так. Многие компании в целях сетевой защиты соединяют свои сети через надлежащие средства, либо используют другие протоколы. Например, отправитель или получатель могут использовать протокол X.400. Такая ситуация показана на рисунке 7-55. Отправитель передает сообщение шлюзу, тот его буферизует и позднее передает получателю. Звучит просто, но на деле все сложнее. Первая проблема – соответствие адресов. Вторая – соответствие конвертов и заголовков. Третья – соответствие тела сообщения. Например, в случае, если тело содержит аудиофайл, а на стороне получателя с ним работать не умеют. Или отправитель поставил следующее условие: если передача сообщения не пройдет по почтовому соединению, то нужно повторить его по факсу, а получатель не умеет работать с факсом. Однако для простых неструктурированных ASCII-сообщений SMTP-шлюз – решение проблемы.

7.4.4.3. Доставка получателю

До сих пор мы предполагали, что машина пользователя может и отправлять сообщения, и получать их. Однако часто машина пользователя – это персональный компьютер или ноутбук, которая время от времени связывается с почтовым сервером, чтобы отправить или получить почту. Как это происходит? Простой протокол для изъятия почты из удаленного почтового ящика – POP3 (Post Office Protocol – RFC 1225). Он позволяет входить в удаленную систему и выходить из нее, передавать письма и принимать их. Главное - он позволяет забирать почту с сервера и хранить ее на машине пользователя.

Более сложный протокол IMAP – Interactive Mail Access Protocol (RFC 1064). Он позволяет одному и тому же пользователю заходить с разных машин на сервер, чтобы прочесть или отправить почту. Это по существу удаленное хранилище писем. Он, например, позволяет получать доступ к письму не только по его номеру, но и по содержанию.

Третий часто используемый протокол – DMSP (Distributed Mail System Protocol RFC 1056). Этот протокол не предполагает, что пользователь работает все время с одной и той же почтовой службой. Пользователь может обратиться к серверу и забрать почту на свою локальную машину, после чего разорвать соединение. Обработав почту, он может ее отправить позже, когда будет установлено очередное соединение.

Важными почтовыми сервисами являются:

- Фильтры
- Пересылка поступающей почты на другие адреса
- Демон отсутствия
- Почтовый робот (программа, анализирующая входящие письма и отвечающая на них)

7.4.5. Конфиденциальность почты

Пославший почту, естественно, предполагает, что ее никто не читает кроме адресата. Однако если об этом специально не позаботиться, то гарантировать этого нельзя. Далее мы рассмотрим две широко распространенных безопасных почтовых системы - PGP и PEM.

PGP – Pretty Good Privacy

PGP – дословно: «вполне хорошая конфиденциальность» – разработка одного человека - Фила Зиммермана (Phil Zimmermann, 1995). Это полный пакет безопасности, который включает средства конфиденциальности, установления подлинности, электронной подписи, сжатия, и все это в удобной для использования форме.PGP использует алгоритмы шифрования RSA, IDEA и MD5. PGP поддерживает компрессию передаваемых данных их секретность, электронную подпись и средства управления доступом к ключам. Схема работы PGP показана на рисунке 7-56. На этом рисунке DA, DB - личные (закрытые) ключи A и B соответственно, а EA, EB – их открытые ключи. Отметим, что секретный ключ для IDEA строится автоматически в ходе работы PGP на стороне A и называется ключом сессии - KM, который затем шифруется алгоритмом RSA с открытым ключом пользователя B. Также следует обратить внимание на то, что медленный алгоритм RSA используется для шифрования коротких фрагментов текста: 128-разрядного ключа для MD5 и 128-разрядного ключа для IDEA.

PEM – почтовая служба с повышенной конфиденциальностью

PEM имеет статус Internet-стандарта (RFC 1421, 1424). Сообщения, пересылаемые с помощью PEM, сначала преобразуются в каноническую форму. В этой форме соблюдены соглашения относительно спецсимволов типа табуляции, последовательных пробелов и т.п. Затем сообщение обрабатывается MD5 или MD2, шифруется с помощью DES (56-разрядный ключ) и передается с помощью кодировки base64. Передаваемый ключ защищается либо с помощью RSA, либо с помощью DES по схеме EDE.

7.5. Протокол передачи файлов - FTP

FTP (File Transfer Protocol, Протокол передачи данных) - это один из первых и все еще широко используемых интернет-сервисов. Первые спецификации FTP относятся к 1971 году. С тех пор FTP претерпел множество модификаций и значительно расширил свои возможности.

Протокол FTP предназначен для решения следующих задач:

- разделение доступа к файлам на удаленных абонентских машинах
- прямое или косвенное использования ресурсов удаленных компьютеров
- обеспечение независимости клиента от файловых систем удаленных абонентских машин
- эффективная и надежная передача данных

FTP - это протокол прикладного уровня, который, как правило, использует в качестве транспортного протокола TCP. FTP не может использоваться для передачи конфиденциальных данных, поскольку не обеспечивает защиты передаваемой информации и передает между сервером и клиентом открытый текст. FTP-сервер может потребовать от FTP-клиента аутентификации (т.е. при подсоединении к серверу FTP-пользователь должен будет ввести свой идентификатор и пароль). Однако и пароль, и идентификатор пользователь будут переданы от клиента на сервер открытым текстом.

7.5.1. Модель работы FTP

Простейшая модель работы протокола FTP представлена на рисунке 7-59, где введены следующие обозначения:

- «User Interface» - пользовательский интерфейс работы с FTP.

- «User-PI» - интерпретатор команд пользователя (User Protocol Interpretator). Этот объект взаимодействует с «Server-PI», чтобы обмениваться командами управления передачей данных по каналу передачи команд и с «User-DTP» - модулем, который осуществляет непосредственную передачу данных по каналу передачи данных.
- «User-DTP» - модуль, осуществляющий обмен данными (User Data Transfer Process) между клиентом и сервером FTP по каналу передачи данных на основании команд модуля «User-PI». Этот объект взаимодействует с файловой системой пользователя и объектом «Server-DTP».
- «Server-PI» - модуль управления обменом данных со стороны сервера (Server Protocol Interpretator) по каналу передачи команд.
- «Server-DTP» - модуль, осуществляющий обмен данными со стороны сервера (Server Data Transfer Process) по каналу передачи данных
- «Сервер FTP» - модуль, осуществляющий работу FTP-сервера. Он состоит из модуля управления передачей - «Server-PI» и модуля, осуществляющего передачу, - «Server-DTP».
- «Пользователь FTP» - модуль клиента FTP. Он состоит из модуля управления передачей - «User-PI» - и модуля, осуществляющего передачу - «User-DTP».

FTP поддерживает сразу два канала соединения - канал передачи команд (и статусов их обработки) и канал передачи данных. Канал передачи данных может использоваться для передачи как в одном, так и в другом направлении, кроме того, он может закрываться и открываться по командам управляющих модулей в процессе работы. Канал передачи команд открывается с установлением соединения и используется только для передачи команд и ответов их обработки.

Алгоритм работы протокола FTP состоит в следующем:

1. Сервер FTP использует в качестве управляющего соединение на TCP порт 21, который всегда находится в состоянии ожидания соединения со стороны FTP-клиента.

2. После того как устанавливается управляющее соединение модуля «User-PI» с модулем сервера - «Server-PI», клиент может отправлять на сервер команды. FTP-команды определяют параметры соединения передачи: роли участников соединения (активный или пассивный), порт соединения (как для «User-DTP», так и для «Server-DTP»), тип передачи, тип передаваемых данных, структуру данных и управляющие директивы, обозначающие действия, которые пользователь хочет совершить, например, сохранить, считать, добавить или удалить данные или файл.

3. После того как согласованы все параметры канала передачи данных, один из участников соединения, который является пассивным (например, клиентский модуль «User-DTP»), становится в режим ожидания открытия соединения на заданный для передачи данных порт. После этого активный модуль (например, «Server-DTP») открывает соединение и начинает передачу данных.

4. После окончания передачи данных соединение между «Server-DTP» и «User-DTP» закрывается, но управляющее соединение «Server-PI» - «User-PI» остается открытым. Пользователь, не закрывая сессии FTP, может еще раз открыть канал передачи данных, передать необходимую информацию и т.д.

FTP может использоваться не только при передаче файлов между клиентом и сервером, но и между двумя FTP-серверами, ни один из которых не расположен на локальном хосте пользователя. Для этого пользователь сначала устанавливает управляющие соединения с двумя FTP-серверами, а затем устанавливает между ними канал передачи данных. В этом случае управляющая информация передается через модуль «User-PI», но данные транслируются через канал «Server1-DTP» - «Server2-DTP».

Основу передачи данных FTP составляет механизм установления соединения между соответствующими портами и механизм выбора параметров передачи. Каждый участник FTP-соединения должен поддерживать 21 порт передачи данных по умолчанию. По умолчанию «User-DTP» использует тот же порт, что и для передачи команд (обозначим его «U»), а «Server-DTP» использует порт номер (L-1), где L-управляющий порт. Однако, как правило, участниками соединения используются порты передачи данных, выбранные для

них «User-PI», поскольку из управляющих процессов, участвующих в соединении, только он может изменить порты передачи данных как у «User-DTP», так и у «Server-DTP». Пассивная сторона соединения должна до того, как будет подана команда начать передачу, слушать свой порт передачи данных. Активная сторона, подающая команду к началу передачи, определяет направление перемещения данных. После того как соединение установлено, между «Server-DTP» и «User-DTP» начинается передача. Одновременно по каналу «Server-PI» - «User-PI» передаются уведомления о получении данных. Протокол FTP требует, чтобы управляющее соединение было открыто, пока по каналу обмена данными идет передача. Сессия FTP считается закрытой только после закрытия управляющего соединения. Как правило, сервер FTP ответственен за открытие и закрытие канала передачи данных. Сервер FTP должен самостоятельно закрыть канал передачи данных в следующих случаях:

1. Сервер закончил передачу данных в формате, который требует закрытия соединения.
2. Сервер получил от пользователя команду прервать соединение.
3. Пользователь изменил параметры порта передачи данных.
4. Было закрыто управляющее соединение.
5. Возникли ошибки, при которых невозможно возобновить передачу данных.

FTP-протокол имеет двух «младших братьев»: TFTP - Trivial FTP и SFTP - Simple FTP.

Управление сетями. Протокол SNMP. Вопросы производительности сетей.

SNMP – протокол управления сетью

Когда сеть из компьютеров охватывает небольшие пространства - несколько комнат, этаж – то в случае возникновения каких-то неполадок можно обойти все помещения и проверить работоспособность оборудования и программного обеспечения каждого устройства в сети. Когда сеть охватывает большие территории и включает оборудование, принадлежащее нескольким разным организациям, то так уже не поступишь. Например, много времени уйдет на согласование доступа в помещения, а при больших территориях – просто чтобы обойти все устройства и найти плохо работающее. Поэтому, когда Интернет стал охватывать большие территории, потребовались адекватные средства для управления сетью. Под словом «управление» в данном случае мы будем понимать возможность оперативно получать информацию о каждом устройстве в сети: о его работоспособности, состоянии, истории его функционирования. Позднее на понятии управления в сети мы остановимся подробнее. В 1990 году была опубликована первая версия протокола Simple Network Management Protocol (SNMP v.1). Эта версия описана в RFC 1155 и RFC 1157. В этих документах было описано систематическое наблюдение за сетью (какая информация могла накапливаться, как и где) и управление ею (как и какие параметры работы устройств сети можно было менять). Этот протокол получил широкое распространение и был реализован практически во всех устройствах, используемых в сетях. Опыт использования протокола SNMP v.1 выявил ряд недостатков. Например, в первой версии недостаточно были проработаны вопросы безопасности. При опросе устройств явно указывался пароль, глядя на который, устройство аутентифицировало запрашивающее устройство. Во второй версии протокола SNMP v.2 (RFC 1441-1452) была введена криптографическая защита механизма аутентификации. Далее будет кратко описана именно вторая версия протокола SNMP.

7.3.1. Модель управления

Предполагается, что сеть состоит из четырех категорий компонентов:

- Станции управления
- Управляемые устройства
- Управляющая информация
- Протокол управления

Управление сетью осуществляют станции управления. Это компьютеры, на которых выполняются процессы, собирающие и накапливающие информацию об управляемых устройствах в сети. Сбор этой информации происходит по запросу от управляющей станции управляемому устройству. Запросы, передача и другие действия выполняются с помощью команд протокола SNMP. На управляемых устройствах работают специальные SNMP-агенты (далее для краткости просто «агенты»), которые выполняют команды, передаваемые с помощью SNMP-протокола, фиксируют определенный набор параметров функционирования управляемого устройства. Управляемым устройством может быть маршрутизатор, мост, рабочая станция, устройство печати – любое устройство, где может работать SNMP-агент. Каждый агент поддерживает локальную базу данных MIB (Management Information Base), где хранится информация о состоянии агента, история его функционирования и переменные, характеризующие работу устройства, где функционирует агент.

7.3.2. SMI – структура управляющей информации

В сети используется аппаратура сотен различных производителей. Естественно агент должен формировать данные о функционировании управляемого устройства в некотором унифицированном виде, например, по составу, способу представления независимо от того, кто изготовил это устройство. В соответствии с терминологией, принятой в стандарте протокола SNMP, переменную, в которой агент накапливает информацию, будем называть объектом. Все объекты собраны в группы, определяемые стандартом, а группы – в модули. Чтобы все объекты имели единые правила идентификации, поступают следующим образом. Строят дерево стандарта, в котором отражают иерархию понятий, используемых в стандарте. Это дерево стандарта является поддеревом дерева стандартов. На рисунке 7-30 показана часть такого дерева. На первом ярусе этого дерева расположены названия организаций, имеющих право выпускать международные стандарты – это ISO и МКТТ (теперь МСТ – международный союз телекоммуникаций). Есть в этом дереве и место для Интернета – ярус 4. На последнем ярусе указаны группы. В скобках рядом с именем каждой группы указано количество объектов в группе. Объекты имеют тип, определяемый через Object Identifier. Все объекты в этом дереве могут быть заданы указанием пути в дереве. SMI в определенном смысле представляет собой язык для определения структур данных, представляющих объекты в базе данных MIB. В таблице 7-31 указаны типы данных, используемые для определения объектов, отслеживаемых протоколом SNMP.

7.3.4. Протокол SNMP

SNMP-протокол определяет пять типов сообщений, которыми обмениваются станция управления и устройство. Все они показаны на рисунке 7-44.

- get-request - получить значение одной или нескольких переменных.
- get-next-request - получить одну или несколько переменных, следующих после указанной переменной.
- set-request - установить значение одной или нескольких переменных.
- get-response - выдать значение одной или нескольких переменных. Это сообщение возвращается агентом станции управления в ответ на операторы get-request, get-next-request и set-request.
- оператор trap - уведомить станцию управления, когда что-либо произошло с агентом.

Первые три сообщения отправляются от станции управления к устройству, а последние два - от устройства к станции управления. Так как четыре из пяти SNMP-сообщений реализуются простой последовательностью «запрос-ответ», в SNMP-протоколе используют UDP-протокол. Это означает, что запрос от станции управления может не дойти до устройства, а отклик от устройства – до станции управления. В этом случае будет задействован механизм тайм-аута и повторная передача. Станция управления отправляет все три запроса на UDP-порт 161. Устройство устанавливает ловушки (программные прерывания trap) на UDP-порт 162. Так как используются два разных порта, одна и та же система может выступать и как станция управления, и как устройство. На рисунке 7-45 показан формат пяти SNMP-сообщений, инкапсулированных в UDP-дейтаграмму. При взаимодействии между станцией управления и устройством используют пароль.

Пароль - это 6-символьная строка, которую передавали в SNMP v.1 в открытом виде. В операторах get, get-next и set станция управления устанавливает идентификатор запроса (request ID), который возвращается устройством в сообщении get-response. Это повышает безопасность при взаимодействии. Это поле также позволяет станции управления выдать несколько запросов одному или нескольким устройствам, а затем отсортировать полученные отклики. Статус ошибки (error status) - это целое число, которое возвращается агентам и указывает на ошибку.