Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Ответы на билеты по курсу

«Введение в компьютерные сети»

Авторы:
Белов Н. А., 327 группа,
Богомолов Д. А., 327 группа,
Кислых Д. М., 324 группа,
Теплова С. Е., 327 группа

Москва, 2014

Благодарность

1. Кочетков А. А., 324 группа,
2. Лукьяненко С. Ю., 324 группа,
3. Мадорский К. М., 325 группа,
4. Малахов Д. П., 328 группа,
5. Мешков Г. С., 320 группа,
6. Орпанен И. С., 320 группа,
7. Романов А. С., 327 группа,
8. Туаев А. А., 327 группа,
9. Тузикова А. В., 325 группа,
10. Филимонов В. В., 324 группа,
11. Холопкин С. А., 328 группа,
12. Шишков И. Д., 327 группа,
13. Шевелев А. В., 324 группа.

Содержание

1 Основные движущие силы развития информационных технологий.

Состояние и направление развития информационных технологий определяют три основные составляющие: **микроэлектроника**, **телекоммуникации** и **инженерия программного обеспечения**.

1946 – год создания первого компьютера ENIAC. Д. Моучли, Д. Эккерт: вес 27 т, 18’000 электронных ламп, 1’500 реле, потребляла около 150 кВт энергии.

1947 – точечный транзистор. В.Шокли, Д. Барден и У. Бретейн.

1971 – первый микропроцессор Intel 4004: частота 108 кГц, 2’300 транзисторов.

1978 – микропроцессор Intel 8086: частота 5 МГц, 29’000 транзисторов.

2001 – микропроцессор Pentium 4: частота 1,7 ГГц, число транзисторов 42 млн.

С 1971 г. тактовая частота процессоров Intel возросла в 28 тыс. раз, т.е. со 108 кГц до 3 ГГц, а среднее число транзисторов в одном процессоре выросло в 350 тыс. раз.

Тенденция увеличения числа транзисторов на кристалле была сформулирована в 1965 Г. **Муром** (одним из основателей Intel) в виде закона, который носит его имя: **количество транзисторов в интегральной схеме с минимальной ценой удваивается каждые 18 месяцев**.

И сам компьютер, и его интерфейс с человеком подверглись существенным изменениям. Персональный компьютер впервые появился в 1981, и за следующие 20 лет он совершил революцию: до этого он был доступен только специалистам, а теперь он «вошел» почти в каждый дом, с ним работают школьники, он стал мобильным. В настоящее время в мире функционирует более 1,5 млрд ПК. Отдельный класс устройств на основе микропроцессоров составляют встроенные системы. Это весьма широкий класс устройств: от бытовых приборов до сложных технических систем. Уже в 2002 г. устройства, не являющиеся персональными компьютерами, составляли около 50% от всех устройств доступа в Интернет, а к 2005 г. число таких устройств превысило число ПК. К 2005 г. на одного жителя Европы и Северной Америки в среднем приходилось около 20 микропроцессов, размещенных в бытовых приборах. Сегодня микропроцессор можно встретить везде: в кроссовках, кастрюлях, автомобилях, самолетах, на кораблях.

В настоящее время, с одной стороны, компьютеры объединяют в сети, оборудуя их необходимыми средствами телекоммуникации, а с другой стороны, традиционные средства коммуникации, такие как телефон, превращают из простого средства передачи голоса в изощренные средства интерактивного беспроводного взаимодействия.

В области телекоммуникаций действует **закон Гилдера**, выявивший на основании статистических данных следующую тенденцию: **пропускная способность телекоммуникационных каналов удваивается каждые шесть месяцев**. Современная система передачи данных, или просто кабель, за секунду способна пропустить столько данных, сколько в 1997 г. пропускал весь Интернет. Так, например, в 1997 г. Ethernet обладал пропускной способностью в 100 Мбит, а сегодня это 100 Гбит.

В первые десятилетия компьютерной истории профессионального программирования как такового не было. Программирование рассматривалось как кодирование. Если в качестве примера посмотреть на первые учебники по программированию, то они представляли собой сборники профессиональных рецептов, как ввести число, строку символов, и т.д. Программирование было ремеслом, а не видом индустриальной деятельности. Программы в массе своей не были продуктами.

В 1970-е гг. пришло понимание того, что программы как продукт инженерной деятельности имеют беспрецедентную в истории человеческой цивилизации сложность, а следовательно, нельзя каждый раз разработку программной системы начинать с нуля. Поэтому активно стали развиваться методы программировании, сновная идея которых состояла в использовании при создании программных систем ранее написанных компонентов – кирпичиков, в которых аккумулировался бы опыт предшествующих разработок и корректность работы которых не надо было бы каждый раз обосновывать.

Однако реализация этой идеи потребовала решения череды очень непростых задач, осознание которых во многих случаях приходило не сразу. Так, например, эти компоненты должны были работать в разных операционных средах, быть многократно используемыми в разных программных контекстах, и т.д.

2 Кто, как и для чего использует Сеть: интранет, B2B, B2C, B2G.

В сфере бизнеса можно выделить области применения сетей в следующих направлениях:

- интранет – использование сети для управления и производственных нужд внутри предприятия;
- B2B (Business To Business) – взаимодействие с другими предприятиями, которые, в свою очередь, можно подразделить на взаимодействие с предприятиями-заказчиками, взаимодействие с предприятиями-поставщиками и создание виртуальных предприятий;
- B2C (Business To Customer) – взаимодействие предприятия с конечными пользователями их продукции;
- электронное правительство, которое, в свою очередь, разделяется на B2G (Business To Goverment) – взаимодействие предприятия с государством и G2C (Government To Citizen) – взаимодействие государства с гражданами.

Интранет – это средства, обеспечивающие и регулирующие доступ сотрудников компании к внутрикорпоративным средствам коммуникации, информационным и вычислительным сервисам. Упрощенно интранет – это внутренняя корпоративная сеть, построенная на интернет-технологиях. Не все пользователи находятся в зоне сети интранет. Некоторые из них подключаются к корпоративным информационным ресурсам извне.

Интранет обеспечивает для всех сотрудников единый способ доступа к информации, единую унифицированную среду работы, единый формат документов. Такой подход позволяет сотрудникам наиболее эффективно использовать накопленные корпоративные знания, оперативно реагировать на происходящие события, где бы они ни находились, а предприятию в целом предоставляет новые возможности организации своего бизнеса.

Важными характеристиками интранета являются открытость и масштабируемость. Интранет должен позволять наращивать функциональность и интегрировать информационные прикладные системы организации. Это свойство позволяет предприятию развивать информационные системы эволюционным путем по мере возникновения необходимости.

B2B – условное обозначение набора услуг, которые одна фирма может оказать другой, используя сеть Интернет. На сегодня сформировались два основных направления в этой сфере: интернет-биржи и интернет-консалтинг. Интернет-биржи являются несколько расширенным представлением обычных бирж с тем только отличием, что за счет больших возможностей по отображению визуальной информации с их помощью стала реальной торговля не только классическими биржевыми товарами, такими как нефть, зерно, металлы, но и некоторыми стандартными видами товарной продукции (оборудованием, комплектующими, компьютерной техникой и т.д.)

Интернет-консалтинг в Сети обеспечивает, в первую очередь, поиск и обработку информации из самой Сети (поиск поставщиков и потребителей, мониторинг, анализ рынков и т.д.). Многие электронные биржи позволяют не только заключать сделки, но и планировать, а также управлять поставками. Управление поставками очень важно, так как позволяет регулировать стоимость конечного продукта. Основной формой взаимодействия в сфере B2B является сотрудничество.

B2C – условное обозначение набора услуг, которые фирма может оказать клиенту (конечному потребителю), используя возможности, предоставляемые Сетью. Типичными представителями данного направления применения Сети являются интернет-магазины. Безусловно, к этому классу приложений относятся банковские услуги, разнообразные информационные и справочные услуги, ориентированные на конечного потребителя, туристические услуги, сетевые библиотеки игр, книг, фильмов, услуги по дистанционному обучению. Клиент может оформить заказ и отследить его выполнение, получая необходимые уведомления о прохождении определенного этапа изготовления, вплоть до даты доставки товара. Развитие данного вида предприятий предопределено резким снижением затрат. Остаются только затраты на склад и службу доставки.

Концепция электронного правительства (Electronic Government) была провозглашена в 1997 г. в США на самом высоком правительственном уровне. Цель программы была заявлена как снижение издержек

при финансировании деятельности госаппарата и повышение открытости и прозрачности органов управления на основе внедрения технологии электронной коммерции. Роль электронной коммерции в организации работы государственных органов двояка. С одной стороны, это снижение издержек и экономия средств налогоплательщиков на содержание и финансирование деятельности госаппарата (B2G). С другой стороны, это повышение открытости и прозрачности органов управления, обеспечение свободного доступа граждан ко всей необходимой государственной информации (G2C). Разделение новой индустрии на секторы B2G и G2C чисто функциональное. Однако в обоих секторах необходимую инфраструктуру обеспечивает Сеть. В первую очередь электронными могут стать самые очевидные функции государства – сбор налогов, регистрация транспортных средств, регистрация патентов, выдача лицензий, получение необходимой информации, заключение договоров и оформление поставок необходимых государственному аппарату материалов, оборудования. В результате сокращается объем бумажной работы, и проведение необходимых процедур значительно ускоряется. То, для чего раньше требовалось долгое стояние граждан в очередях, общение с правительственными чиновниками, а также производство и перемещение большою количества бумажных документов будет происходить теперь за несколько минут. B2G может стать эффективным оружием в борьбе с коррупцией.

3 Модели сетевого взаимодействия OSI ISO и TCP/IP. Базовая модель взаимодействия сетевых приложений.

Модель OSI имеет уровневую организацию. Она включает в себя **семь уровней**: физический, канальный, сетевой, транспортный, сессии, представления и прикладной.

Ключевыми в этой модели являются понятия сервиса, интерфейса и протокола. Под сервисом понимают услуги, которые нижерасположенный уровень оказывает по запросам вышерасположенного. В этой модели нижерасположенный уровень свои услуги может предоставлять только вышерасположенному уровню. Интерфейс определяет формирование и передачу запроса на услугу. Активные элементы уровня, т.е. элементы, которые могут сами совершать действия, в отличие от элементов, над которыми совершают действия, называются активностями. Активности могут быть программными и аппаратными. Активности одного и того же уровня на разных машинах называются равнозначными, или одноименными. Активности уровня n + 1 являются пользователями сервиса, создаваемого активностями уровня n, которые, в свою очередь, называются поставщиками сервиса. Сервис может быть разного качества. Правила и соглашения по установлению соединения, его поддержанию и обмену данными по нему между активностями, расположенными на одинаковом уровне на разных машинах, называется протоколом. Доступ к сервису в модели OSI осуществляется через так называемые точки доступа к сервису – SAP (Service Access Points), каждая из которых имеет уникальный адрес. Взаимодействие между двумя соседними уровнями в этой модели можно описать следующим образом: активность на уровне n + 1 передает интерфейсную единицу данных – IDU (Interface Data Unit) активности на уровне n через SAP (рис. 1.3). IDU состоит из сервисной единицы данных – SDU (Service Data Unit) – и управляющей информации. SDU передается далее по сети равнозначной сущности, а затем - - на уровень n + 1. Управляющая информация необходима нижерасположенному уровню, чтобы правильно передать SDU, но она не является частью передаваемых данных. Чтобы передать SDU по сети нижерасположенному уровню, может потребоваться разбить ее на части. При этом каждая часть снабжается заголовком и концевиком и передается как самостоятельная единица данных протокола PDU (Protocol Data Unit). Заголовок в PDU используется протоколом при передаче. В нем указывается, какой PDU содержит управляющую информацию, а какой – данные, порядковый номер PDU и т.д. Формально сервис можно описать в терминах примитивных операций, или примитивов, с помощью которых пользователь или какая-либо активность получает доступ к сервису.

Модель TCP/IP.

Рассмотрим другую эталонную модель, прототипом для которой послужил прародитель Интернета – сеть ARPA. С самого начала эта сеть задумывалась как объединение нескольких разных сетей. Одной из основных целей этого проекта была разработка унифицированных способов соединения сетей для

создания систем передачи данных, обладающих высокой живучестью. Так появилась модель TCP/IP, получившая название по именам двух основных протоколов: протокола управления передачей – TCP (Transmission Control Protocol) и межсетевого протокола – IP (Internet Protocol). Связь в этой сети должна поддерживаться до тех пор, пока источник информации и получатель информации работоспособны. Архитектура сети ARPA не должна была ограничивать приложения, начиная от простой передачи файлов до передачи речи и изображения в реальном времени.

Модель TCP/IP включает в себя **3 основных уровня**:

- межсетевой уровень,
- транспортный уровень,
- уровень приложений.

В модели TCP/IP нет уровней сессии и представления, поскольку необходимость в них была неочевидна для ее создателей.

Модели TCP/IP и OSI имеют много общего. Обе эти модели имеют уровневую организацию и поддерживают понятие стека протоколов. Назначение их уровней примерно одинаковое. Все уровни этих моделей от транспортного и ниже используют протоколы для поддержки взаимодействия типа точка–точка, не зависящего от организации СПД, а все уровни выше транспортного ориентированы на приложения. Наибольшее значение модели OSI методологическое: в ней явно определены и четко выделены понятия сервиса, интерфейса, протокола, уровня. В модели TCP/IP нет столь же четкого выделения этих понятий. В ней понятие протокола оторвано от остальных частей, и в ней нет единой, хорошо продуманной, концепции построения. Этот факт является следствием того, как создавались эти модели.

4 Основные принципы организации и функционирования Интернета.

В 1969 году в США была создана компьютерная сеть ARPAnet, объединяющая компьютерные центры министерства обороны и ряда академических организаций. Эта сеть была предназначена для узкой цели: главным образом для изучения того, как поддерживать связь в случае ядерного нападения и для помощи исследователям в обмене информацией. По мере роста этой сети создавались и развивались многие другие сети. Еще до наступления эры персональных компьютеров создатели ARPAnet приступили к разработке программы Internetting Project («Проект объединения сетей»). Успех этого проекта привел к следующим результатам. Во-первых, была создана крупнейшая в США сеть internet (со строчной буквы i). Во-вторых, были опробованы различные варианты взаимодействия этой сети с рядом других сетей США. Это создало предпосылки для успешной интеграции многих сетей в единую мировую сеть. Такую «сеть сетей» теперь всюду называют Internet (в отечественных публикациях широко применяется и русскоязычное написание – Интернет).

В настоящее время на десятках миллионов компьютеров, подключенных к Интернету, хранится громадный объем информации (сотни миллионов файлов, документов и т.д.) и сотни миллионов людей пользуются информационными услугами глобальной сети.

Глобальные сети (Wide Area Network, WAN) – это сети, предназначенные для объединения отдельных компьютеров и локальных сетей, расположенных на значительном удалении (сотни и тысячи километров) друг от друга. Глобальные сети объединяют пользователей, расположенных по всему миру, используя при этом самые разнообразные каналы связи.

Современный Интернет позволяет пользователю:

- общаться с людьми, находящимися в любой точке земного шара,
- быстро и комфортно отыскивать любую необходимую информацию,
- публиковать для всеобщего сведения данные, которые он хотел бы сообщить всему миру.

В действительности Internet не просто сеть. Это структура, объединяющая обычные сети. Internet – это «сеть сетей».

Internet – метасеть, состоящая из многих сетей, которые работают согласно протоколам семейства TCP/IP, объединены через шлюзы и используют единое адресное пространство и пространство имен.

Среду передачи данных в Internet нельзя рассматривать только как паутину проводов или оптоволоконных линий. Оцифрованные данные пересылаются через **маршрутизаторы**, которые соединяют сети и с помощью сложных алгоритмов выбирают наилучшие маршруты для информационных потоков (рис.1).

Глобальная сеть включает подсеть связи, к которой подключаются локальные сети, отдельные компоненты и терминалы (средства ввода и отображения информации) (рис. 2).

Компьютеры, за которыми работают пользователи-клиенты, называются **рабочими станциями**, а компьютеры, являющиеся источниками ресурсов сети, предоставляемых пользователям, называются **серверами**. Такая структура сети получила название **узловой**.

Надежность функционирования глобальной сети обеспечивается избыточностью линий связи: как правило, серверы имеют более двух линий связи, соединяющих их с Интернетом. Основу, «каркас» Интернета составляют более ста миллионов серверов, постоянно подключенных к сети.

Рис. 1: Схема взаимодействия в сети Интернет.

Инфраструктура Интернет (рис.2):

1. Магистральный уровень (система связанных высокоскоростных телекоммуникационных серверов).
2. Уровень сетей и точек доступа (крупные телекоммуникационные сети), подключенных к магистрали.
3. Уровень региональных и других сетей.
4. ISP – интернет-провайдеры.
5. Пользователи.

К **техническим ресурсам** сети Интернет относятся компьютерные узлы, маршрутизаторы, шлюзы, каналы связи и др.

Рис. 2: Инфраструктура сети Интернет.

В основу архитектуры сетей положен **многоуровневый принцип передачи сообщений**.

Формирование сообщения на самом верхнем уровне модели ISO/OSI – **уровне приложений**. Затем последовательно проходит **все уровни** системы **до** самого **нижнего**, где и передается по каналу связи адресату. По мере прохождения каждого из уровней системы **сообщение трансформируется, разбивается на сравнительно короткие части**, которые снабжаются дополнительными заголовками. В этом узле сообщение проходит **от нижнего уровня к верхнему**, снимая с себя заголовки. В результате адресат принимает сообщение **в первоначальном виде**.

Протоколом передачи данных называется соглашение, устанавливающее, каким образом должна осуществляться передача данных из компьютера в компьютер и как можно распознавать и устранять ошибки, которые могут при этом возникать. И для того, чтобы осуществилась идея неограниченной коммуникации между компьютерами Интернет, используется один и тот же протокол TCP/IP. Он состоит из набора протоколов, каждый из которых выполняет различные задачи.

- TCP, UDP транспортные протоколы, управляющие передачей данных между машинами;
- IP, ICMP, RIP протоколы маршрутизации. Они обрабатывают адресацию данных, обеспечивают фактическую передачу данных;
- DNS, ARP протоколы поддержки сетевого адреса обеспечивают идетификацию машины с уникальным номером и именем;
- FTP, TELNET протоколы прикладных сервисов. Это программы, которые пользователь использует для получения доступа к различным услугам и др.

Протоколы семейства TCP/IP реализуют всевозможные сервисы (услуги) Интернет.

WWW

Популярнейший из них – World Wide Web (сокращенно WWW или Web), его еще называют Всемирной паутиной. Представление информации в WWW основано на возможностях гипертекстовых ссылок. Гипертекст - это текст, в котором содержаться ссылки на другие документы. Это дает возможность при

просмотре некоторого документа легко и быстро переходить к другой связанной с ним по смыслу информации, которая может быть текстом, изображением, звуковым файлом или иметь любой другой вид, принятый в WWW. При этом связанные ссылками документы могут быть разбросаны по всему земному шару.

Многочисленные пересекающиеся связи между документами WWW компьютерной паутиной охватывают планету – отсюда и название. Таким образом, пропадает зависимость от местонахождения конкретного документа.

Gopher-система

Эта система является предшественником WWW и сейчас утрачивает свое значение, хотя пока и поддерживается в Интернет. Это информационные серверы, на которых содержаться документы академической направленности и большие текстовые файлы. Просмотр информации на Gopher-сервере организуется с помощью древовидного меню, аналогичного меню в приложениях Windows или аналогично дереву каталогов (папок) файловой системы. Меню верхнего уровня состоит из перечня крупных тем, например, экономика, культура, медицина и др. Меню следующих уровней детализируют выбранный элемент меню предыдущего уровня. Конечным пунктом движения вниз по дереву (листом дерева) служит документ аналогично тому, как конечным элементом в дереве каталогов является файл.

Электронная почта

Следующий вид сервиса Интернет – электронная почта, или E-mail. Она предназначена для передачи в сети файлов любого типа. Одни из главных ее преимуществ – дешевизна и быстрота. Электронная почта является исторически первой информационной услугой компьютерных сетей и не требует обязательного наличия высокоскоростных и качественных линий связи. Любой пользователь Интернета может получить свой «почтовый ящик» на одном из почтовых серверов Интернета (обычно на почтовом сервере провайдера), в котором будут храниться передаваемые и получаемые электронные письма. У электронной почты есть преимущества перед телефонной связью. Телефонный этикет очень строг. Есть множество случаев, когда нельзя позвонить человеку по соображениям этикета. У электронной почты требования намного мягче. По электронной почте можно обратиться к малознакомому человеку или очень занятому человеку. Если он сможет, то ответит. Чтобы электронное письмо дошло до адресата, оно, кроме текста послания, обязательно должно содержать электронный адрес получателя письма. Адрес электронной почты записывается по определенной форме и состоит из двух частей: имя_пользователя@имя_сервера Имя_пользователя имеет произвольный характер и задается самим пользователем; имя_сервера жестко связано с выбором пользователем сервера, на котором он разместил свой почтовый ящик.

Пример, ivanov@kyaksa.net

В нашем классе имя пользователя – это имя компьютера, например, pc01, pc02 и т.д. имя сервера: server, поэтому электронный адрес компьютера в локальной сети класса: pc01@server

Чтобы отправить электронное письмо, отправитель должен подключиться к Интернету и передать на свой почтовый сервер сообщение. Почтовый сервер сразу же отправит это письмо через систему почтовых серверов Интернет на почтовый сервер получателя, и оно попадет в его почтовый ящик. Однако получатель получит письмо только после того, как соединится с Интернетом и «скачает» почту из своего почтового ящика на собственный локальный компьютер.

Телеконференции UseNet

Телеконференции UseNet представляют собой электронные форумы. Пользователи Интернет посылают туда свои сообщения, в которых высказываются по определенной теме. Сообщения поступают в специальные дискуссионные группы - телеконференции, при этом каждое мнение становится доступным для всех участников конкретной группы. Уже сегодня UseNet имеет более 20 000 телеконференций, посвященных различным темам: компьютерам, рецептам, вопросам генной инженерии и многому другому.

Протокол передачи файлов FTP

Протокол передачи файлов FTP используется для переписывания файлов с дистрибутивными копиями программ с удаленных серверов на Ваш компьютер. В зависимости от своих прав (обычный пользователь или др.) Вы можете производить те или иные действия по отношению к удаленному серверу (в большинстве случаев это копия имеющейся на нем информации).

Telnet

Программа Telnet была разработана для обеспечения дистанционного доступа к удаленному компьютеру в Интернет. При этом компьютер пользователя выступает в качестве терминала, подключенного к большому компьютеру. В отличие от компьютеров, терминалы не обладают собственными вычислительными возможностями. Они только обеспечивают доступ к какому - то компьютеру благодаря имеющимся у них монитору и клавиатуре. В качестве примера можно привести системы в аэропортах, на вокзалах, где Вы можете получить информацию о билетах, рейсах и т.п.

5 Модели IP, TCP, UDP и ICMP сервисов в Интернете.

Модель сервиса IP

Internet Protocol («межсетевой протокол») – маршрутизируемый протокол сетевого уровня стека TCP/IP. Именно IP стал тем протоколом, который объединил отдельные компьютерные сети во всемирную сеть Интернет. Неотъемлемой частью протокола является адресация сети.

IP объединяет сегменты сети в единую сеть, обеспечивая доставку пакетов данных между любыми узлами сети через произвольное число промежуточных узлов (маршрутизаторов). Он классифицируется как протокол третьего уровня по сетевой модели OSI. IP не гарантирует надёжной доставки пакета до адресата – в частности, пакеты могут прийти не в том порядке, в котором были отправлены, продублироваться (приходят две копии одного пакета), оказаться повреждёнными (обычно повреждённые пакеты уничтожаются) или не прийти вовсе. Гарантию безошибочной доставки пакетов дают некоторые протоколы более высокого уровня – транспортного уровня сетевой модели OSI, – например, TCP, которые используют IP в качестве транспорта.

В современной сети Интернет используется IP четвёртой версии, также известный как IPv4. В протоколе IP этой версии каждому узлу сети ставится в соответствие IP-адрес длиной 4 октета (4 байта). При этом компьютеры в подсетях объединяются общими начальными битами адреса. Количество этих бит, общее для данной подсети, называется маской подсети (ранее использовалось деление пространства адресов по классам – A, B, C; класс сети определялся диапазоном значений старшего октета и определял число адресуемых узлов в данной сети, сейчас используется бесклассовая адресация).

В настоящее время вводится в эксплуатацию шестая версия протокола – IPv6, которая позволяет адресовать значительно большее количество узлов, чем IPv4. Эта версия отличается повышенной разрядностью адреса, встроенной возможностью шифрования и некоторыми другими особенностями. Переход с IPv4 на IPv6 связан с трудоёмкой работой операторов связи и производителей программного обеспечения и не может быть выполнен одномоментно. На середину 2010 года в Интернете присутствовало более 3000 сетей, работающих по протоколу IPv6. Для сравнения, на то же время в адресном пространстве IPv4 присутствовало более 320 тысяч сетей, но в IPv6 сети гораздо более крупные, нежели в IPv4.

- предотвращает «зацикливание» пакетов;
- фрагментирует пакеты если они слишком длинные;
- использует контрольную сумму, чтобы сократить возможность доставки в неправильное место назначение;
- две версии:

  IPv4 с 32 битным адресом

  IPv6 с 128 битным адресом
- позволяет добавлять новые опции к заголовку;
- работает над любой физической средой;
- над IP работают различные транспортные протоколы;
- поверх транспорта работают прикладные протоколы;

- IP используется всегда для передачи пакетов между различными сетями;
- очень простой сервис.

Свойство	Поведение
Datagram	Индивидуально маршрутизируемый скачками пакет
Ненадежный	Пакет может быть сброшен
По возможности (best effort)	...но только при необходимости
Без соединения	Нет состояния у потока Последовательность пакетов может быть нарушена

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 до 256 номера зарезервированы для стандартных сервисов, которые постоянно активны и готовы к работе. Например, для обеспечения FTP-передачи файла соединение должно выполняться через 21-й порт, где находится FTP-демон, а для TELNET - через 23-й порт. Полный список таких портов можно найти в RFC 1700.

Все TCP-соединепня — дуплексные, т.е. передача по ним происходит независимо в оба направления. TCP-соединение поддерживает только соединение точка—точка. Не существует TCP-соединений типа «от одного ко многим».

TCP-агент поддерживает поток байтов, а не поток сообщений. Напомним, это означает, что гранины сообщений не поддерживаются автоматически в потоке. Например, если по TCP-соединению передается текст, разбитый на страницы, то TCP-агент не будет каким-либо образом обозначать конец каждой страницы.

После передачи приложением данных TCP-агенту, эти данные могут быть отправлены сразу на сетевой уровень, а могут быть буферизованы. Как поступить в этом случае решает TCP-агент. Однако в ряде случаев бывает необходимо, чтобы данные были отправлены сразу, например если эти данные представляют собой команду для удаленной машины. Для этого в заголовке TCP-сегмента имеется флаг PUSH, и если он установлен, то это говорит TCP-агенту о том, что данные должны быть переданы немедленно.

Наконец, если в заголовке TPDU-сегмента установлен флаг URGENT, то TCP-агент передает такой сегмент незамедлительно. Когда срочные данные поступают к месту назначения, то их передают получателю немедленно.

Свойство	Поведение
Поток байт	*Сервис доставки байт*
Надежная доставка	*1. Ack подтверждает доставку* *2. Контрольная сумма выявляет ошибки передачи (охватывает весь сегмент)* *3. Последовательные номера позволяют обнаружить пропущенные данные* *4. Управление потоком предотвращает получателя от «захлебывания»*
Сохранение последовательности	*Данные доставляют приложению в том же порядке, в каком они передавались*
С установкой соединения	*Процедура 3-х кратного рукопожатия*
Управление перегрузкой	*Управление перегрузками в сети*

Рис. 3: Модель сервиса TCP

Модель сервиса UDP

UDP – транспортный протокол без соединений, предназначенный для обмена дейтаграммами между процессами на абонентских машинах, входящих в единую сеть с коммутацией пакетов, который описан в RFC 768. В качестве протокола нижнего уровня в UDP-протоколе используется протокол IP.

Протокол UDP предоставляет прикладным программам возможность отправлять сообщения другим приложениям, используя минимальное число параметров протокола. При этом протокол UDP не обеспечивает достоверность доставки пакетов, защиту 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.

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. Его используют многие приложения, например почтовый протокол SMTP, протокол виртуального терминала Telnet, протокол всемирной паутины HTTP, протокол передачи файлов FTP, протокол управления сетью SNMP, протокол службы доменных имен DNS и многие другие.

Биты	0 - 15	16 - 31
0-31	Порт отправителя (Source port)	Порт получателя (Destination port)
32-63	Длина датаграммы (Length)	Контрольная сумма (Checksum)
64-...	Данные (Data)	

Рис. 4: Структура UDP-заголовка

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, т.е. 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, которая вычисляется так же как и в протоколе TCP, включая псевдозаголовок. Контрольная сумма UDP-пакета представляет собой побитовое дополнение 16-битовых суммы 16-битовой слов (аналогично TCP). В вычислении участвуют данные пакета, заголовок UDP-пакета, псевдозаголовок (информация от IP-протокола), поля выравнивания по 16-битовой границе (нулевые).

Преимущество протокола UDP состоит в том, что он требует минимум установок параметров для соединения двух процессов между собой. Этот протокол используется при работе серверов доменов (Name Servers), при работе протокола TFTP (Trivial File Transfer), при работе с SNMP-протоколом и при построении систем аутентификации. Идентификатор UDP в IP-заголовке – число 17. Более подробное описание протокола UDP приведено в RFC 768.

Свойство	Поведение
Сервис без соединения	*Соединение не устанавливают* *Пакеты могут приходить в произвольном порядке*
Дейтаграммы самодостаточны	
Ненадежная доставка	1. *Отсутствуют подтверждения (Аск)* 2. *Контрольная сумма охватывает только заголовок* 3. *Отсутствуют средства выявления пропущенных данных или данных, нарушающих порядок отправки.* 4. *Нет управления потоком*

Рис. 5: Модель сервиса UDP

Модель сервиса ICMP

Управление функционированием Интернета на сетевом уровне происходит через маршрутизаторы с помощью протокола ICMP (Internet Control Message Protocol), описанного в RFC 792. Этот протокол обеспечивает доставку сообщений любой машине, имеющей IP-адрес, от маршрутизаторов и других хостов в сети. С помощью этого протокола реализуют обратную связь для решения проблем, возникающих при передаче. Он также выявляет и рассылает сообщения о десятках событий. Для доставки своих сообщений протокол ICMP использует пакеты IP-протокола. Приведем наиболее важные сообщения.

Сообщение **destination unreachable** охватывает множество случаев, например, случай, когда маршрутизатор не знает, как достигнуть необходимой подсети или хоста, или случай, 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. Это сообщение также генерирует хост, если он не успел завершить обработку IP-пакета до истечения времени его жизни. (Далее слова «пакет», «IP-пакет», «дейтаграмма» будем понимать как синонимы, если специально не оговорено что-либо другое.)

Синтаксические или семантические ошибки в заголовке IP-пакета вызывают появление сообщения **parameter problem**.

Сообщение **source quench** обеспечивает управление потоком. Маршрутизатор или хост-получатель высылает этот пакет хосту-отправителю, если последнему необходимо понизить скорость передачи. Другими словами, это пример подавляющего пакета. Сообщения такого типа будут генерироваться до тех пор, пока скорость поступления пакетов от отправителя не достигнет значения, необходимого хосту-получателю. Это сообщение система может использовать для предотвращения перегрузки, поскольку оно возникает всякий раз, когда маршрутизатор вынужден сбросить пакет из-за переполнения своего буфера.

Сообщение **redirect** позволяет маршрутизатору отправить рекомендацию о лучшем маршруте и впредь посылать пакеты с определенным IP-адресом через другой маршрутизатор.

Сообщения **echo request** и **echo reply** позволяют проверить работоспособность хостов в сети: получатель сообщения echo request обязан ответить сообщением echo reply, причем с теми же параметрами, что и в echo request.

Сообщения **time-stamp request** и **time-stamp reply** позволяют измерять временную задержку в Интернете на сетевом уровне. Этот механизм необходим, например, для работы алгоритма маршрутизации по состоянию канала.

Свойство	**Поведение**
Сообщение о состоянии	Самодостаточное сообщение об ошибке или исключительном состоянии
Ненадежный	Сервис без соединения и подтверждения

ICMP предоставляет информацию о сетевом уровне хостам и маршрутизаторам. ICMP работает над IP и относится к транспортному уровню. ping и traceroute реализованы с помощью ICMP.

6 Понятия имени и адреса в Интернете.

Домен – это область пространства иерархических имен сети Интернет, которая обслуживается набором серверов доменных имен (DNS) и централизованно администрируется. Домен идентифицируется именем домена.

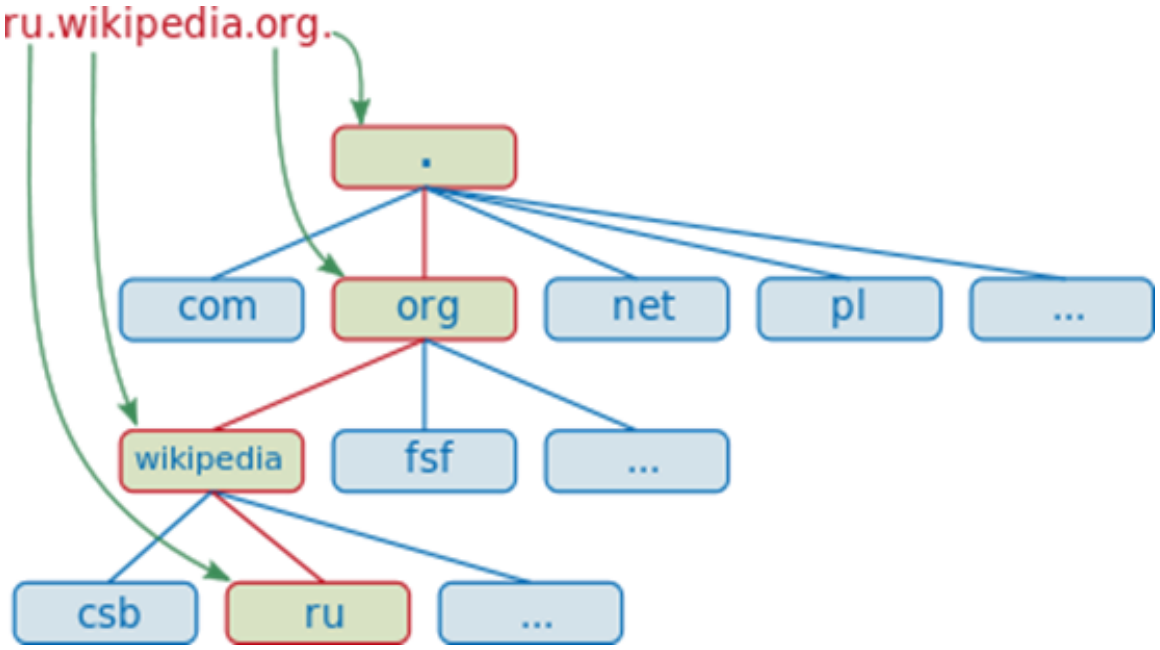
Доменное имя (domain name) – это адрес сетевого соединения, который идентифицирует владельца адреса.

Регистрация доменов – представляет собой занесение информации о домене и его администраторе в центральную базу данных с целью обеспечения уникальности использования домена, а также получения прав на администрирование домена администратором. Услуга по регистрации домена считается оказанной с момента занесения информации в базу данных. Регистрация домена действует в течение одного года, считая с момента регистрации домена.

DNS обладает следующими характеристиками:

- **Распределённость администрирования.** Ответственность за разные части иерархической структуры несут разные люди или организации.
- **Распределённость хранения информации.** Каждый узел сети в обязательном порядке должен хранить только те данные, которые входят в его зону ответственности, и (возможно) адреса корневых DNS-серверов.
- **Кеширование информации.** Узел может хранить некоторое количество данных не из своей зоны ответственности для уменьшения нагрузки на сеть.
- **Иерархическая структура**, в которой все узлы объединены в дерево, и каждый узел может или самостоятельно определять работу нижестоящих узлов, или делегировать (передавать) их другим узлам.
- **Резервирование.** За хранение и обслуживание своих узлов (зон) отвечают (обычно) несколько серверов, разделённые как физически, так и логически, что обеспечивает сохранность данных и продолжение работы даже в случае сбоя одного из узлов.

DNS важна для работы Интернета, так как для соединения с узлом необходима информация о его IP-адресе, а для людей проще запоминать буквенные (обычно осмысленные) адреса, чем последовательность цифр IP-адреса.



Составные части доменного адреса называются **сегментами** и образуют иерархическую систему. Самый последний (крайний правый) сегмент, называемый **доменом верхнего уровня**, определяет принадлежность компьютера к сети той или иной страны и состоит обычно из двух букв, например .su – Советский Союз, .ru – Россия.

В США традиционно используется другая система – тематическая. В этой системе домен верхнего уровня состоит из трех букв и обозначает принадлежность владельца адреса к одному из следующих классов: .com – коммерческие сайты, .edu – образовательные организации, .org – прочие организации.

Доменные адреса компьютеров предназначены для людей. Когда один узел пытается отыскать в Интернете другой, он пользуется иным типом адреса – так называемым IP-адресом (IP – Internet Protocol – межсетевой протокол). Если доменный адрес можно сравнить с именем человека, то IP-адрес – это его «номер телефона», который только и дает реальную возможность связаться с ним.

IP-адрес похож на доменный адрес тем, что также состоит из сегментов, образующих иерархическую систему. Число сегментов в IP-адресе всегда равно четырем, а сами сегменты представляют собой не строки символов, а числа в диапазоне от 0 до 255 (в десятичной записи). Кроме того, в IP-адресах иерархическая лестница спускается слева направо, а не справа налево, как в доменных адресах. Это означает, что два компьютера – соседа по Интернету будут, скорее всего, различаться последним сегментом своих IP-адресов и первым сегментом доменных адресов.

Пример: 194.105.195.17 и 147.115.3.27 представляют два IP адреса.

Для определения по доменному адресу IP-адреса на специальных узлах Сети имеются **Таблицы соответствия**. Такие узлы называются **серверами DNS** (Domain Name Service, «служба доменных имен»). Компьютеру должен быть известен адрес хотя бы одного такого сервера. Если этот сервер не будет знать IP-адрес узла, который вам нужен, он обратится к другим, ближайшим к нему серверам системы DNS, т. е к своим соседям, и так далее. Когда нужный IP-адрес будет, наконец, найден на одном из серверов DNS, его тут же перешлют на ваш компьютер.

7 Способ коммутации потоков данных в Интернете. Виды задержек передачи данных при пакетной коммутации.

Выделяют два основных способа коммутации потоков данных: коммутацию каналов и коммутацию пакетов.

Схематично коммутацию каналов можно описать следующим образом: прежде чем приемник и передатчик начнут обмен данными, между ними посредством коммутации создается канал из отдельных каналов. Коммутированный канал сохраняется до тех пор, пока будут передаваться данные и до поступления команды разрыва соединения. Для создания соединения сигнал вызова должен пройти от точки

возникновения до места назначения и быть подтвержден сигналом об успешном создании соединения. Ярким примером такого способа коммутации потоков данных является телефонная сеть. В телефонных сетях время соединения исчисляется секундами, а при удаленных звонках – это время доходит до минуты, поскольку прежде чем соединение возникнет, сигнал вызова должен проложить маршрут до места назначения, а это требует времени. Однако для многих компьютерных приложений такая большая задержка во времени может оказаться неприемлемой. При этом если соединение установлено, то нет опасности, что во время разговора вы услышите сигнал «занято» из-за нехватки свободных линий у какого-либо коммутатора.

Для коммутации потоков данных посредством коммутации каналов характерно следующее:

- сохранение порядка передаваемых данных, т.е. соблюдение принципа «раньше посланный – раньше будет получен»;
- наличие огромного опыта создания и эксплуатации;
- наличие хорошо развитой инфраструктуры.

В то же время этому способу коммутации потоков данных присущ ряд недостатков:

- неэффективное использование ресурсов, т.е. скорость передачи определяют в этом случае приемник и передатчик, а не возможности коммутированного канала;
- низкая надежность, т.е. достаточно разрушить либо один из промежуточных каналов, составляющих коммутированный канал, либо один из узлов коммутации, и весь коммутированный канал будет разрушен;
- медленная установка соединения.

Альтернативой коммутации каналов является коммутация сообщений. Изначально этот метод использовался при передаче телеграмм. Сообщение получали на узле коммутации целиком, затем целиком передавали по каналу, ведущему к абоненту. И так от оператора к оператору, от одного узла коммутации к другому, пока сообщение не приходило к адресату. В этом случае не требуется создавать соединение заранее. Однако при таком способе передачи необходимо обеспечить на каждом узле коммутации необходимое количество памяти для буферизации любого сообщения, сколь угодно большого.

Для преодоления этого недостатка был предложен метод коммутации пакетов, при котором сообщение разбивается на фрагменты фиксированной длины. Эти фрагменты называются пакетами. Пакеты одного сообщения передаются от одного узла коммутации к другому, пока они не достигнут места назначения. При этом каждый пакет можно передавать независимо от других. Поскольку пакет имеет фиксированную длину, абонент не может монополизировать линию. Одну и ту же линию могут разделять пакеты разных пользователей. Другим достоинством коммутации пакетов является конвейерность: второй пакет можно отправить, не дожидаясь, когда первый достигнет места назначения. Послав второй пакет, можно начать передачу третьего и т.д.

Основные различия между коммутацией каналов и коммутацией пакетов заключаются в следующем:

- при коммутации каналов создается соединение, пропускная способность которого полностью резервируется за двумя абонентами, вне зависимости от того, какая пропускная способность реально им требуется. При коммутации пакетов физическая линия может использоваться пакетами разных абонентов. Следует иметь в виду, что, поскольку при коммутации пакетов не происходит жесткого закрепления канала, то резкое увеличение потока пакетов в узле коммутации может привести к его 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 каждый пакет следует своим маршрутом, и принцип «ранее посланный – будет раньше принят» уже не выполняется, например хотя бы потому, что у разных пакетов маршруты могут быть разной длины. Кроме того, пакеты могут быть искажены при передаче либо вовсе потеряны, поэтому сохранение исходного порядка пакетов в сообщении получателю не гарантируется;

- коммутация каналов абсолютно прозрачна для абонентов. Они могут пересылать данные в любой кодировке и любом формате. При коммутации пакетов формат и способ кодировки пакетов заданы заранее и определяются 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.

Признак	Коммутация каналов	Коммутация пакетов
Выделенный канал передачи	Есть	Нет
Пропускная способность	Фиксированная	Динамическая
Неиспользуемая пропускная способность	Возможна	Невозможна
Передача с буферизацией пакетов	Нет	Есть
Единый путь для всех пакетов	Есть	Нет
Установление соединения	Требуется	Не требуется
Возможность перегрузки	При установлении соединения	На любом пакете
Оплата	За время соединения	За переданные пакеты

Сквозная задержка состоит из трех компонентов:

1. Задержка распространения.
2. Задержка пакетизации.
3. Задержка в очереди в буфере маршрутизатора.

Сквозная задержка (end-to-end delay) – время от момента когда послан первый бит до момента когда придет последний бит.

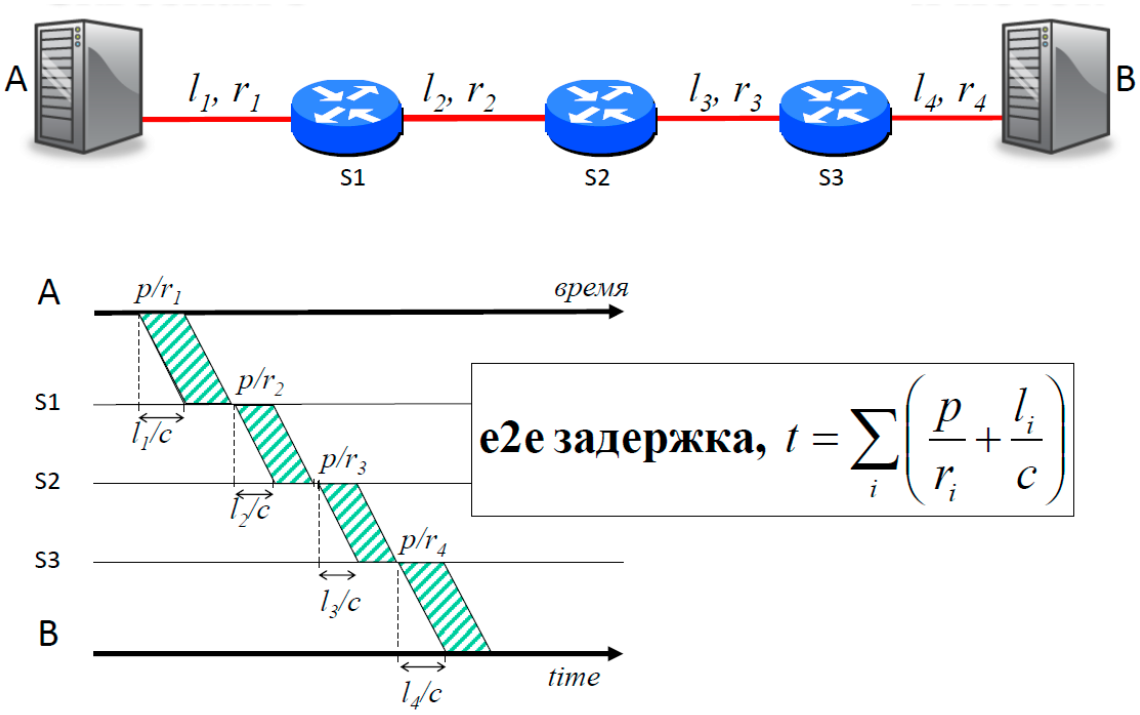
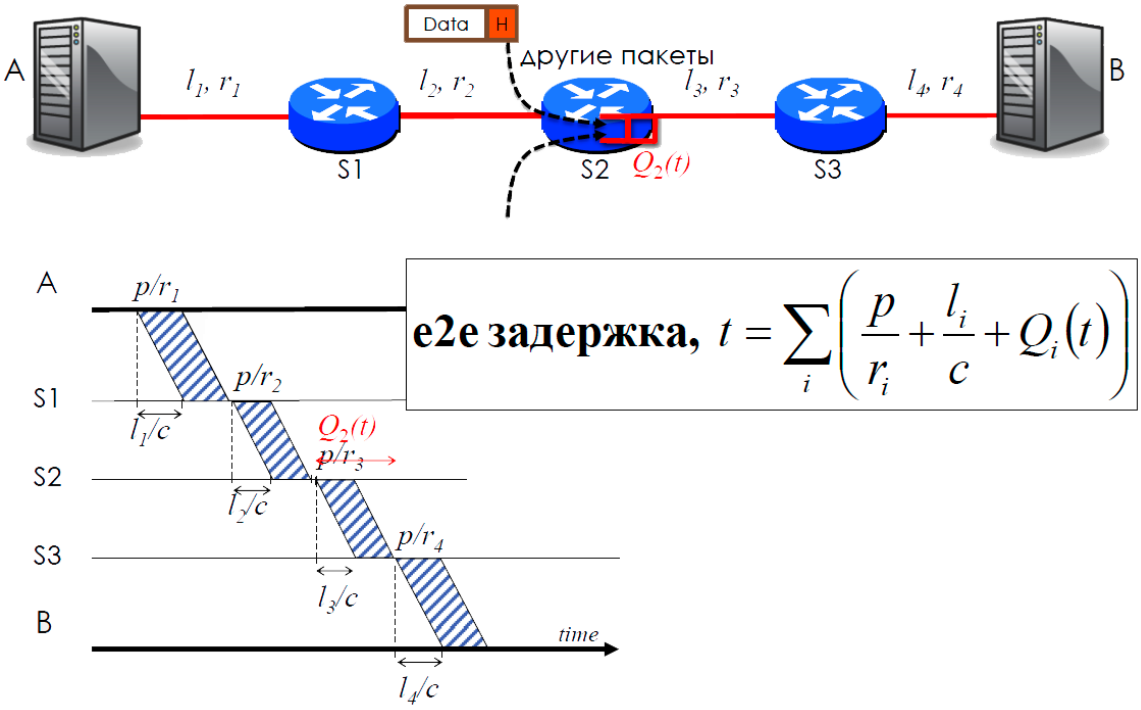


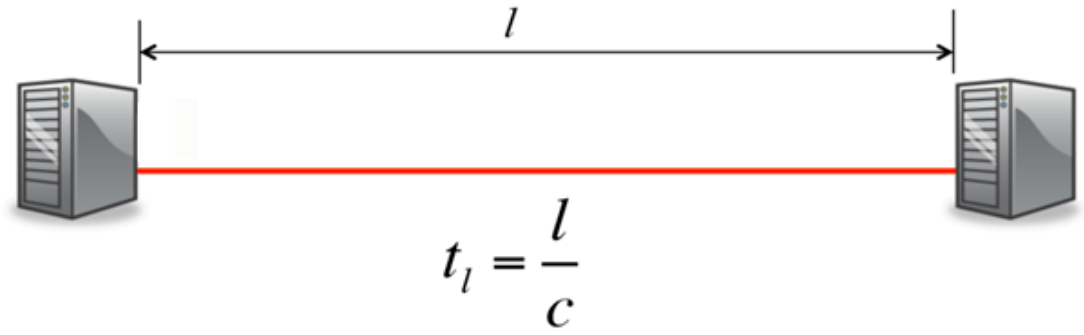
Рис. 6: один поток



e2e задержка, $t = \sum_i \left(\frac{p}{r_i} + \frac{l_i}{c} + Q_i(t) \right)$

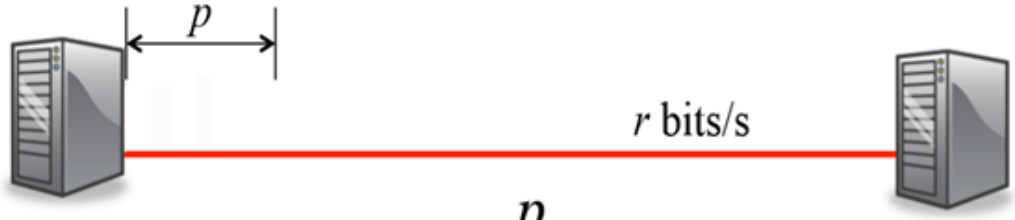
Рис. 7: не один поток

Задержка распространения t_l – время распространения одного бита по каналу со скоростью с.



$$t_l = \frac{l}{c}$$

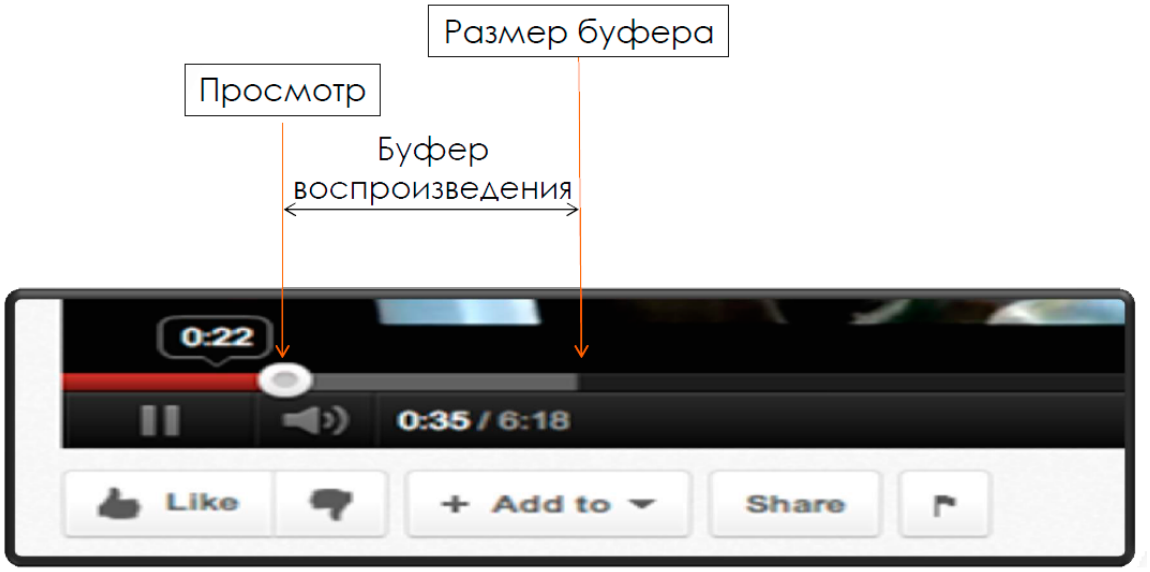
Задержка пакетизации t_p – время, за которое биты с первого до последнего переданы в канал.

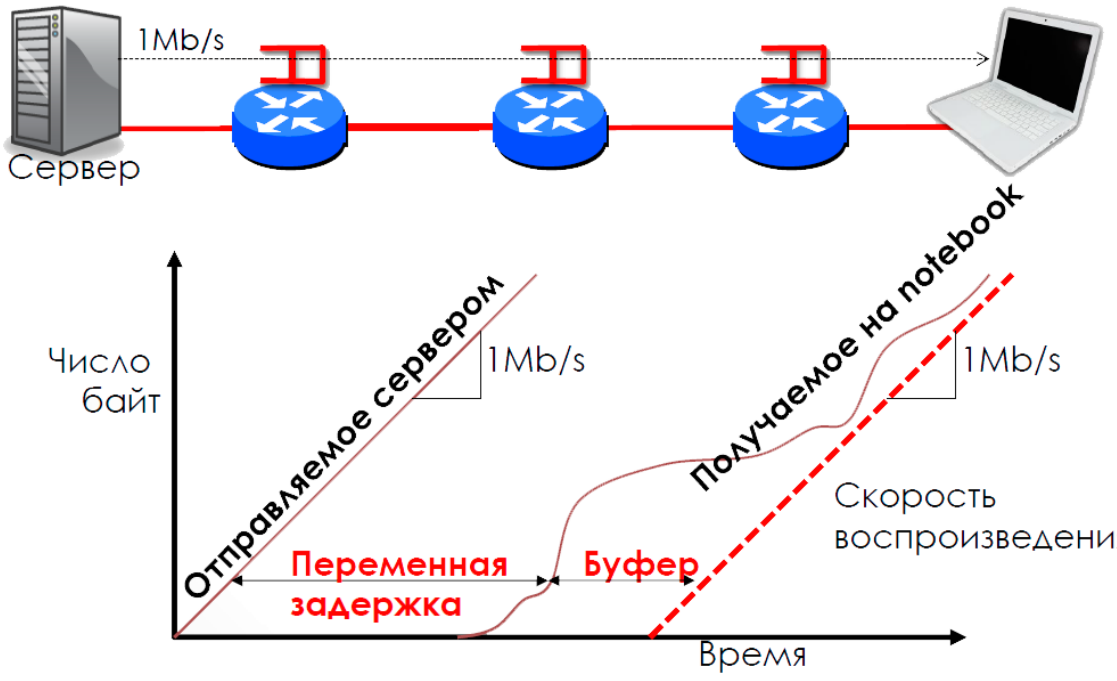


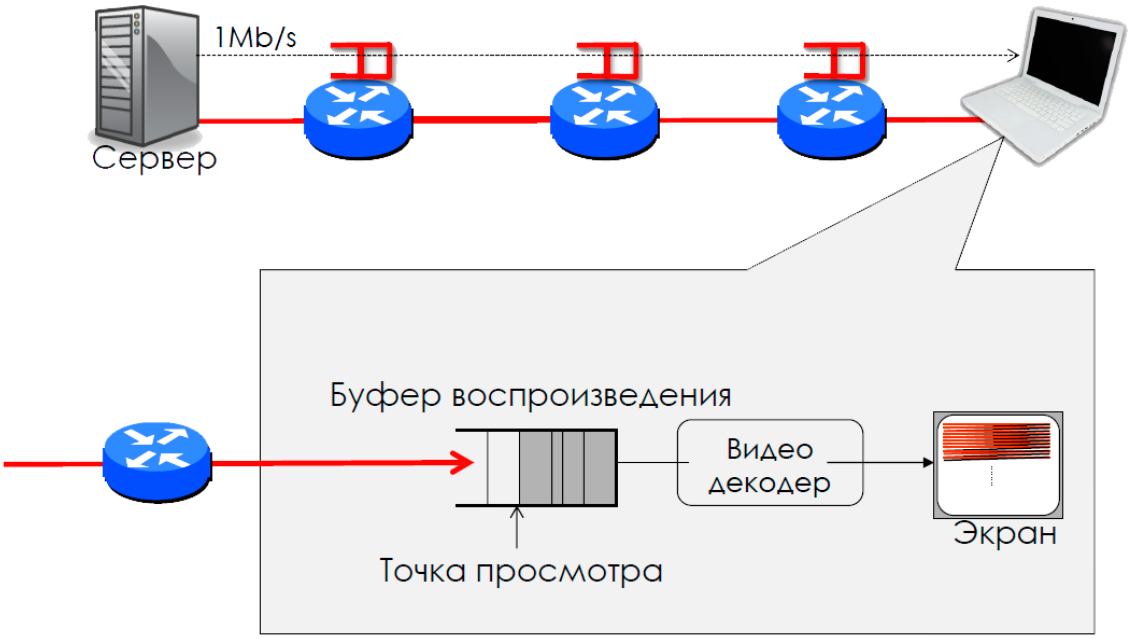
$$t_p = \frac{p}{r}$$

8 Буферизация воспроизведения.

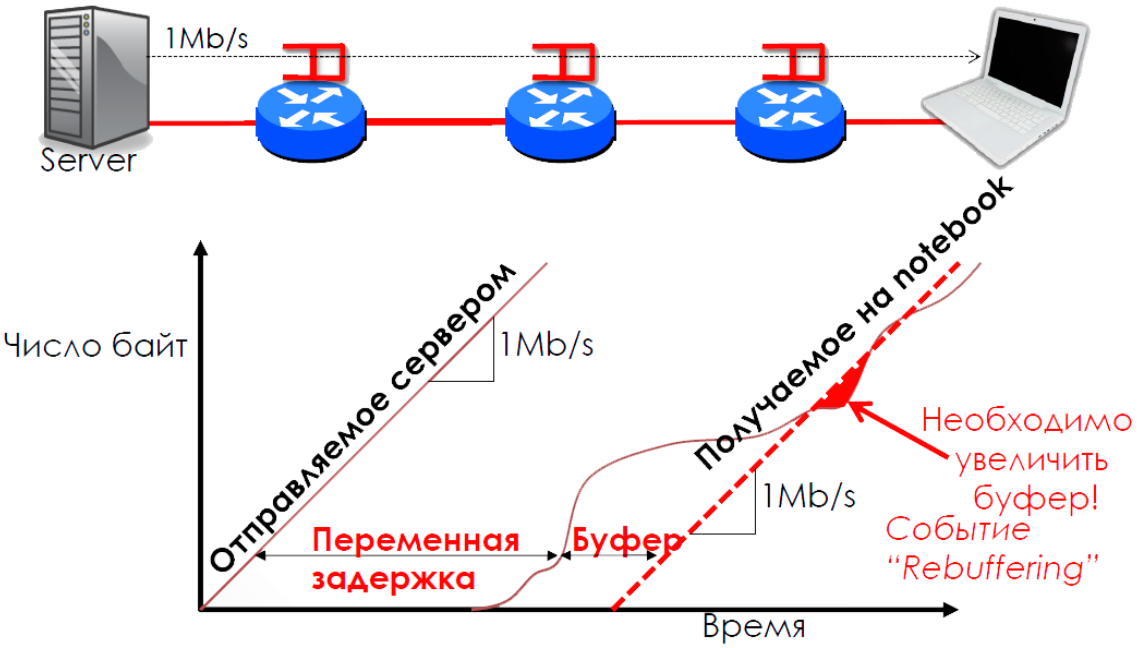
При коммутации пакетов e2e задержка – величина изменчивая. Буферизация позволяет сгладить эти изменения. Можно делать буфер воспроизведения сразу большим, но тогда будет большая задержка на старте. Поэтому приложение должно оценивать задержку, устанавливать размер буфера воспроизведения, и изменять этот размер при изменении задержки. Приложения реального времени (YouTube, Skype) используют буферизацию воспроизведения для сглаживания задержек в очередях.



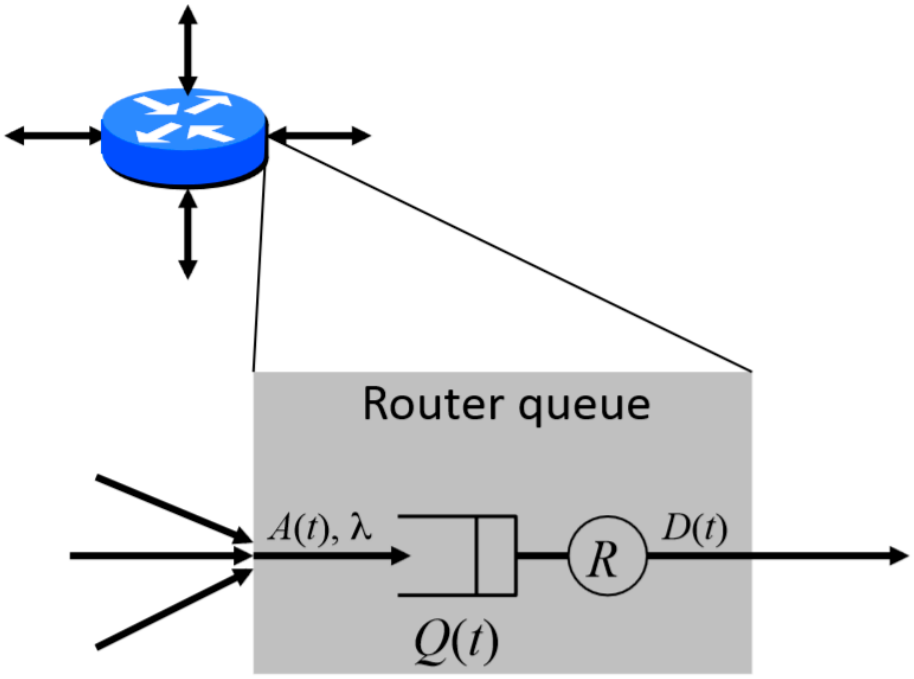




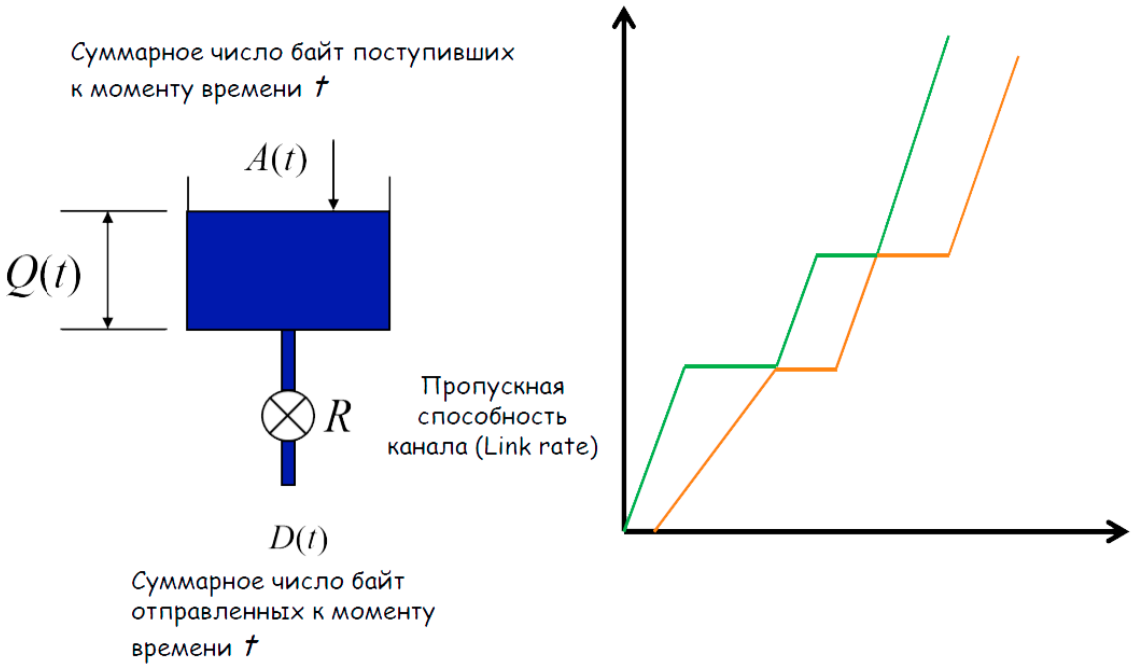
При малом размере буфера:

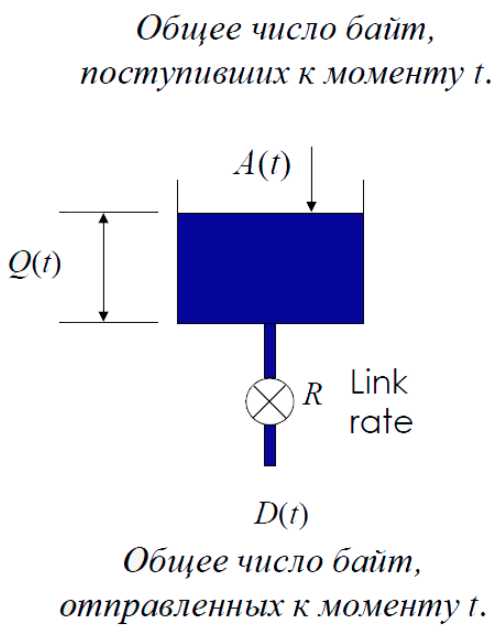


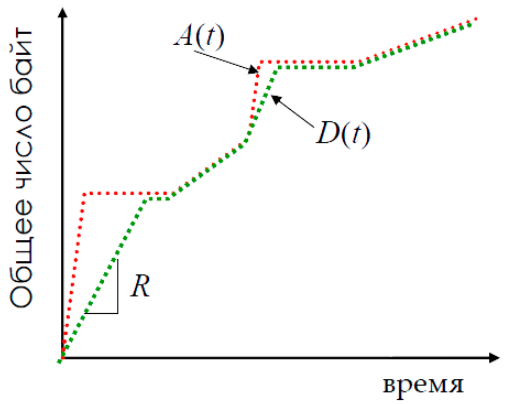
9 Простая модель очереди



Properties of $A(t)$, $D(t)$:
- $A(t)$, $D(t)$ are non-decreasing
- $A(t) >= D(t)$

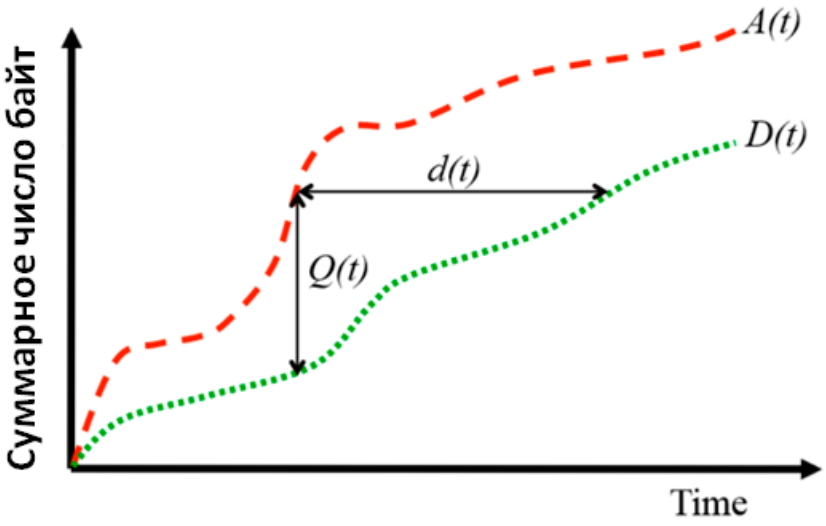






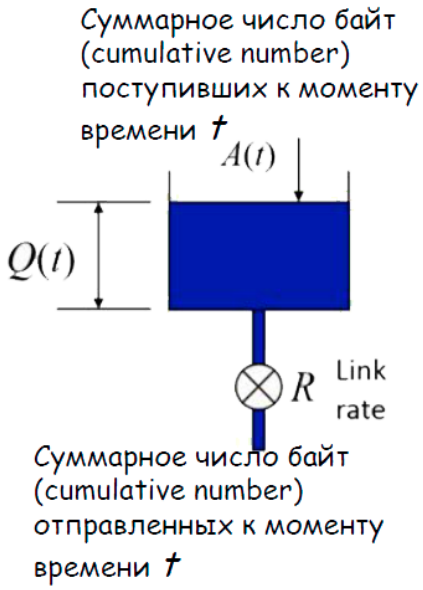
Свойства $A(t)$, $D(t)$:

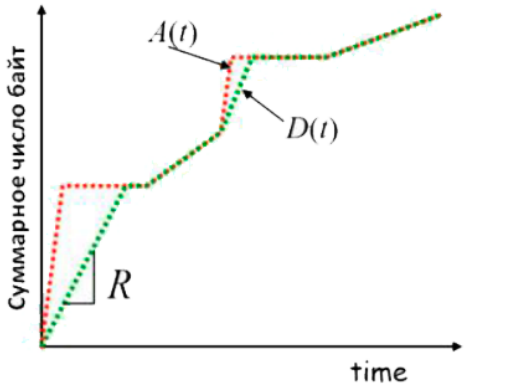
- $A(t)$, $D(t)$ неубывающие
- $A(t) >= D(t)$



Длина очереди: $Q(t) = A(t) - D(t)$.

Рис. 8: Задержка в очереди d(t) – время, которое байт, поступивший в момент t, пробыл в очереди, при условии что дисциплина очереди FIFO.





Свойства функций A(t), D(t):
- A(t), D(t) - неубывающие
- A(t) ≥ D(t)

10 Модели с очередями: свойства очередей.

Из википедии, ниже по лекциям.

Поток заявок **однороден**, если все заявки равноправны и рассматриваются только моменты времени поступления заявок, т.е. факты заявок без уточнения деталей каждой конкретной заявки.

Поток **без последействия**, если число событий любого интервала времени $(t, t+x)$ не зависит от числа событий на любом другом непересекающемся с данным интервале времени. Это значит, что число требований, поступающих в данный отрезок времени, не зависит от числа требований, обслуженных в предыдущем промежутке времени.

Поток заявок **стационарен**, если вероятность появления n событий на интервале времени $(t, t+x)$ не зависит от времени t, а зависит только от длины x этого участка.

Однородный стационарный поток без последействий является простейшим или потоком Пуассона. Число n событий такого потока, выпадающих на интервал длины x, распределено по Закону Пуассона:

$$P(n,x) = \frac{(\lambda x)^n e^{-\lambda x}}{n!},$$

где λ — интенсивность, среднее число требований, поступивших на обслуживание в единицу времени.

На практике условия простейшего потока не всегда строго выполняются. Часто имеет место нестационарность процесса: в различные часы дня и различные дни месяца поток требований может меняться. Существует также наличие последействия, когда количество требований на отпуск товаров в конце месяца зависит от их удовлетворения в начале месяца. Наблюдается и явление неоднородности, когда несколько клиентов одновременно пребывают на склад за материалами. Однако в целом пуассоновский закон распределения с достаточно высоким приближением отражает многие процессы массового обслуживания.

Наличие пуассоновского потока требований можно определить статистической обработкой данных о поступлении требований на обслуживание. Одним из признаков закона распределения Пуассона является равенство математического ожидания случайной величины и дисперсии этой же величины.

Одной из важнейших характеристик обслуживающих устройств, которая определяет пропускную способность всей системы, является время обслуживания. Время обслуживания одного требования t_{obs} — случайная величина, которая может изменятся в большом диапазоне. Она зависит как от стабильности работы самих обслуживающих устройств, так и от различных параметров, поступающих в систему. Случайная величина t_{obs} полностью характеризуется законом распределения, который определяется на основе статистических испытаний.

На практике чаще всего принимают гипотезу о показательном законе распределения времени обслуживания. Он имеет место тогда, когда плотность распределения резко убывает с возрастанием времени t. Например, когда основная масса требований обслуживается быстро, а продолжительное обслуживание встречается редко. Наличие показательного закона распределения времени обслуживания устанавливается на основе статистических наблюдений. При показательном законе распределения времени обслуживания вероятность события, что время обслуживания продлится не более чем t, равна:

$$P_{obs}(t) = 1 - e^{-\nu t},$$

где ν — интенсивность обслуживания одного требования одним обслуживающим устройством, которая определяется из соотношения:

$$\nu = \frac{1}{t'_{obs}},$$

где t'_{obs} — среднее время обслуживания одного требования одним обслуживающим устройством.

Следует заметить, что если закон распределения времени обслуживания показательный, то при наличии нескольких обслуживающих устройств одинаковой мощности закон распределения времени обслуживания несколькими устройствами будет также показательным.

$$P_{obs}(t) = 1 - e^{-\nu n t}$$

Важным параметром системы массового обслуживания (СМО) является коэффициент загрузки α, который определяется как отношение интенсивности поступления требований к интенсивности обслуживания ν:

$$\alpha = \frac{\lambda}{\nu},$$

где λ — интенсивность поступления требований в систему, ν — интенсивность обслуживания одного требования одним обслуживающим устройством.

Из вышеизложенных соотношений получаем, что $\alpha = \lambda t'_{obs}$. Произведение $\lambda t'_{obs}$ показывает количество требований, поступающих в систему обслуживания за среднее время обслуживания одного требования одним устройством. Загрузка α - это среднее число заявок, приходящих за среднее время обслуживания одной заявки.

Для СМО с ожиданием количество обслуживаемых устройств n должно быть строго больше коэффициента α. В противном случае число поступающих требований будет больше суммарной производительности всех обслуживающих устройств и очередь будет неограниченно расти. Для СМО с отказами и смешанного типа это условие может быть ослаблено, для эффективной работы этих типов СМО достаточно потребовать, чтобы минимальное количество обслуживаемых устройств было не меньше коэффициента загрузки $n \geq \alpha$.

По лекциям.

Обычно процесс поступления пакетов сложен и трудно предсказуем, поэтому его часто моделируют случайным процессом. Изучается все это в теории массового обслуживания.

Свойства:

1. Нерегулярность увеличивает задержку.
2. Детерминизм сокращает задержку (при случайном поступлении в среднем приходится ждать дольше, чем при регулярном).
3. Формула Литтла:

Формула Литтла

$$L = \lambda d$$

L - среднее число заявок в системе (в очереди + в обслуживании)
λ - средняя скорость поступления заявок в секунду
d - средне время пребывания заявки в системе (в очереди + в обслуживании)
Это свойство верно если ни одна заявка не теряется/сбрасывается

4. **Пуассоновский процесс** – поступление пакетов не Пуассоновский процесс, но такие процессы, как поступление Web-запросов, новых потоков могут быть описаны, как Пуассоновские процессы:

Пуассоновский процесс

- *Процесс поступления является Пуассоновским если:*
  - *Вероятность поступления k заявок за t секунд*

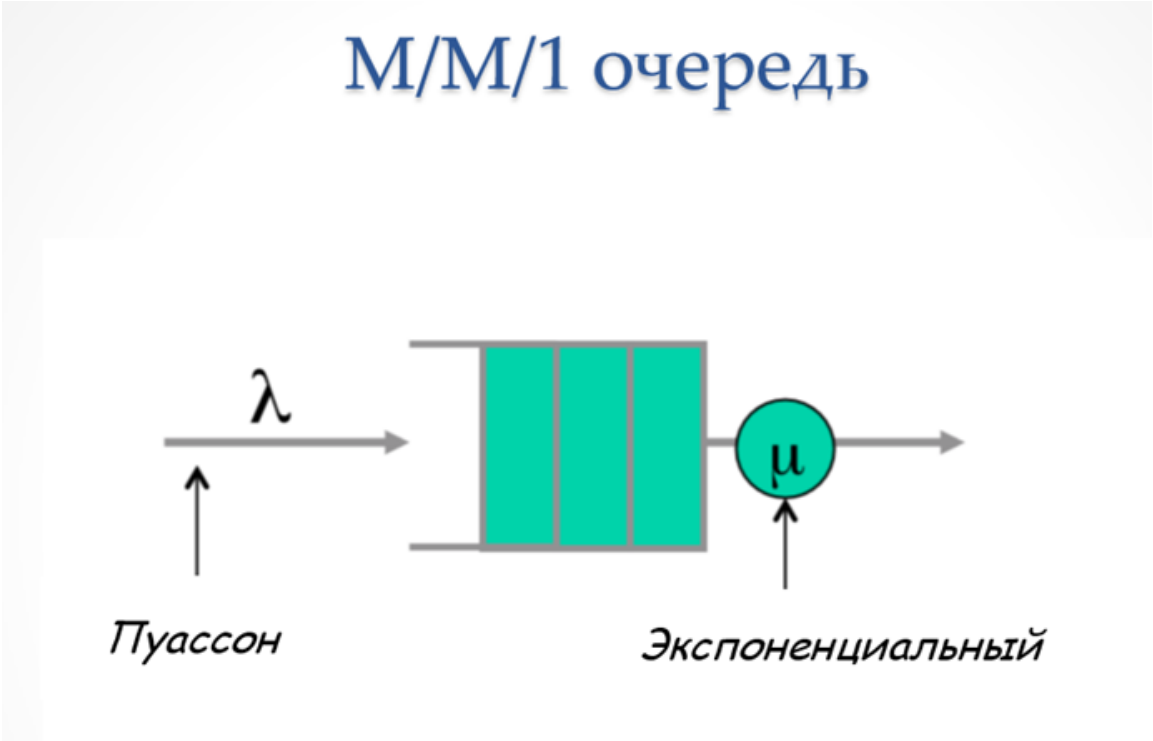
$$P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$$

  - *Интервалы между последовательными поступлениями независимы (нет регулярности)*
- *Тогда число заявок поступивших за время t* **Свойство 4**

$$E = \lambda t$$

- *Пуассоновские процессы хорошо моделируют многие случайные процессы (телефонные звонки, распад частиц, шумы в электрических цепях и т.д.)*
- *Удобный математический аппарат*

5. M/M/1 очередь – простая модель очереди:



11 Как устроен и работает пакетный коммутатор.

Из википедии, ниже по лекциям.

Коммутация пакетов — способ доступа нескольких абонентов к общей сети, при котором информация разделяется на части небольшого размера — пакеты, которые передаются в сети независимо друг от друга. Коммутаторы такой сети имеют внутреннюю буферную память для временного хранения пакетов, что позволяет сглаживать пульсации трафика на линиях связи.

Сетевой коммутатор — устройство, предназначенное для соединения нескольких узлов в пределах одного или нескольких сегментов сети. Он работает на канальном уровне модели OSI. Коммутатор передаёт данные непосредственно получателю, кроме случаев широковещательного трафика и трафик для устройств, для которых неизвестен исходящий порт коммутатора. Это повышает производительность и безопасность сети, избавляя остальные сегменты сети от необходимости и возможности обрабатывать данные, которые им не предназначаются.

Принцип работы. Коммутатор хранит в ассоциативной памяти таблицу коммутации, в которой указывается соответствие MAC-адреса узла порту коммутатора. При включении коммутатора эта таблица пуста, и он работает в режиме обучения. В этом режиме поступающие на какой-либо порт данные передаются на все остальные порты коммутатора. При этом коммутатор анализирует фрейм и, определив MAC-адрес хоста-отправителя, заносит его в таблицу на определенное время. Если на один из портов коммутатора поступит фрейм, предназначенный для хоста, MAC-адрес которого уже есть в таблице, то этот фрейм будет передан только через порт, указанный в таблице. Если MAC-адрес хоста-получателя не ассоциирован с каким-либо портом коммутатора, то фрейм будет отправлен на все порты, за исключением того, с которого он был получен. Со временем коммутатор строит таблицу для всех активных MAC-адресов и в результате трафик локализуется.

Существует три способа коммутации, каждый из которых — это комбинация таких параметров, как время ожидания и надёжность передачи:

- **С промежуточным хранением.** Коммутатор читает всю информацию в кадре, проверяет его на отсутствие ошибок, выбирает порт коммутации и после этого посылает в него кадр.
- **Сквозной.** Коммутатор считывает в кадре только адрес назначения и после выполняет коммутацию. Этот режим уменьшает задержки при передаче, но в нём нет метода обнаружения ошибок.

- **Бесфрагментный или гибридный.** Этот режим является модификацией сквозного режима. Передача осуществляется после фильтрации фрагментов коллизий: первые 64 байта кадра анализируются на наличие ошибки и при её отсутствии кадр обрабатывается в сквозном режиме.

Задержка, связанная с принятием коммутатором решения, добавляется к времени, которое требуется фрейму для входа на порт коммутатора и выхода с него, и вместе с ним определяет общую задержку коммутатора.

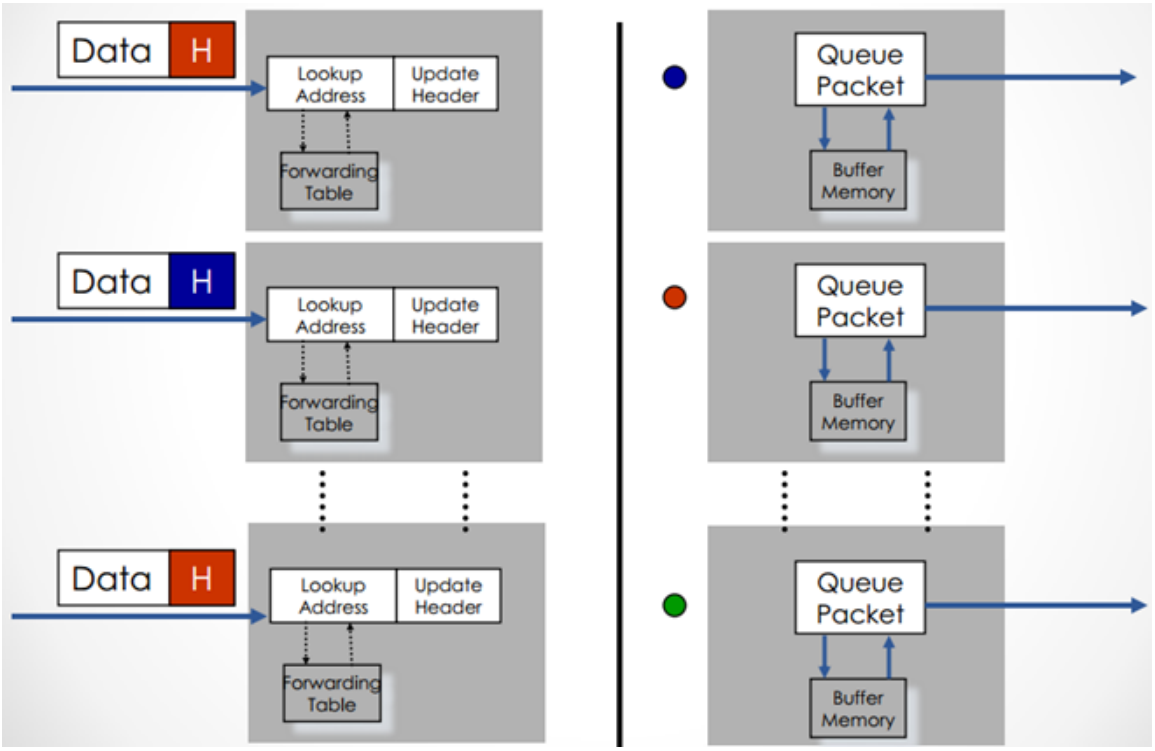
Для временного хранения фреймов и последующей их отправки по нужному адресу коммутатор может использовать буферизацию. Буферизация может быть также использована в том случае, когда порт пункта назначения занят. Буфером называется область памяти, в которой коммутатор хранит передаваемые данные.

Буфер памяти может использовать два метода хранения и отправки фреймов: буферизация по портам и буферизация с общей памятью. При буферизации по портам пакеты хранятся в очередях, которые связаны с отдельными входными портами. При этом возможна ситуация, когда один фрейм задерживает всю очередь из-за занятости порта его пункта назначения. При буферизации в общей памяти все фреймы хранятся в общем буфере памяти, который используется всеми портами коммутатора.

По лекциям.
Lookup Address – фаза просмотра адреса.
Forwarding Table – анализ заголовка пакета с определением выходной линии.
Update Header – модификация заголовка сетевого уровня (TTL, контрольная сумма).

Ethernet коммутатор.

1. Проверяет заголовок каждого прибывающего кадра.
2. Если адрес DA есть в таблице коммутации, то кадр передают на надлежащий выходной порт.
3. Если адрес DA нет в таблице, кадр рассылается по всем портам, кроме того на который пришел.
4. Когда придет ответ на разосланный пакет, то по его адресу отправителя мы узнаем , куда надо направлять пакеты с такими адресами получателей.



Интернет маршрутизатор.

1. Если Ethernet DA поступившего кадра есть Ethernet адрес маршрутизатора, то принять кадр, иначе сбросить его.
2. Просмотреть поля IP version и длина дейтаграммы.
3. Сократить поле TTL, пересчитать контрольную сумму IP заголовка.
4. Проверить TTL на 0.
5. Если IP DA есть в таблице маршрутизации, переслать на надлежащий выходной порт для следующего скачка (hop).
6. Найти Ethernet DA для следующего маршрутизатора.
7. Построить новый Ethernet кадр и отправить его.

Базовые операции коммутатора.

1. **Поиск адреса:** как адрес ищется в таблице маршрутизации?
2. **Коммутация:** как пакет пересылают на надлежащий выходной порт?

Поиск адреса: Ethernet.

1. Адреса хранятся в хэш-таблице.
2. Ищем в хэш-таблице точное совпадение.

Ethernet DA	Действие
0xA8B72340E678	Передать на порт 7
0xB3D22571053B	Передать на порт 3
.....	.....

Рис. 9: Таблица Ethernet адресов.

Поиск адреса: IP.
Ищут **совпадение по самому длинному префиксу, а не точное совпадение**.

IP DA	Действие
127.43.57.99	Передать на 56.99.32.16
123.66.44.x	Передать на 22.45.21.126
76.9.x.x	Передать на 56.99.32.16
.....	.....

Рис. 10: Таблица IP адресов в маршрутизаторе.

Троичная Ассоциативная память.

Двоичная АП – простейший тип ассоциативной памяти, который использует слова поиска данных, состоявшие полностью из единиц и нулей. В троичной АП добавляется третье значение для сравнения «X» или «не важно», для одного или более битов в сохраненном слове данных, добавляя таким образом большей гибкости поиску. Например, в троичной АП могло бы быть сохранено слово «10XX0», которое выдаст совдпадение на любое из четырех слов поиска «10000», «10010», «10100», или «10110». Добавление гибкости к поиску приходит за счет увеличения цены двоичной АП, поскольку внутренняя ячейка памяти должна теперь закодировать три возможных состояния вместо двух. Это дополнительное состояние обычно осуществляется добавлением бита маски «важности»(«важно»/«не важно») к каждой ячейке памяти. Голографическая ассоциативная память обеспечивает математическую модель для интегрированного ассоциативного воспоминания бита «не важно», используя комплекснозначное представление.

Поиск адреса <совпадение, действие>.

Обобщение поиска и коммутации в коммутаторах, маршрутизаторах и т.п.

- По сути, Ethernet коммутаторы и маршрутизаторы выполняют одинаковые действия.
- Поиск адреса в коммутаторе и маршрутизаторе происходит по-разному.

Совпадение	Действие
IP DA = X	Передать на порт 7
EthDA=Y & IP DA = Z	Сброс пакета

Организация очередей на входе.

Примером такого варианта размещения может служить организация буферов на входных портах неблокирующей структуры с пространственным разделением, например баньяновидной сети Бэтчера. К его недостаткам можно отнести опасность возникновения блокировки в начале очереди. Если две одновременно поступившие ячейки направляются на один и тот же выходной порт, одна из них попадет во входной буфер и будет препятствовать прохождению следующих за ней ячеек, снижая тем самым пропускную способность коммутатора. Решением проблемы является значительное увеличение производительности коммутационного поля с пространственным разделением или замена дисциплины «пришедший первым обслуживается первым» (FIFO) на другую, скажем «пришедший первым обслуживается в случайном порядке» (FIRO).

Организация очередей на выходе.

Этот тип буферизации используется в выходных портах структуры с разделяемой шиной. Он оптимален с точки зрения производительности и задержек, но требует применения дополнительных средств для организации одновременной множественной доставки ячеек на любой выходной порт. Таким образом, либо выходные буферы должны функционировать с достаточно высокой скоростью, либо на каждом выходном порте требуются несколько буферов. Оба решения ограничивают производительность и масштабируемость устройства: в первом случае — из-за необходимости существенно повысить внутреннее быстродействие коммутатора, а во втором – буферов.

Пакетный коммутатор: заключение.

Пакетный коммутатор выполняет две базовые операции:

- Поиск соответствия в таблице коммутации.
- Передачу на надлежащий выходной порт.

Самый простой и самый медленный коммутатор использует буферизацию на выходе с минимальной задержкой пакета.

Высокоскоростные коммутаторы используют буферизацию на входе с виртуальными очередями на выходах для увеличения пропускной способности.

12 Коммутация пакетов: приоритеты, веса и гарантированная скорость потока

Рассмотрим пакетный коммутатор с буферизацией на выходе, как гарантию того, что пакеты из буфера будут обслужены с определенной скоростью.

Пакетные коммутаторы работают по принципу FIFO, однако поступающие пакеты могут быть разной длины, а значит, при равномерной работе, окажется, что кто-то получит бОльшую долю пропускной способности.

Если пакеты имеют равный размер, то время задержки каждого пакета не более, чем размер буфера/пропускную способность канала.

Задание строгих приоритетов нарушает справедливость: на каждом цикле работы сети есть пакеты с высоким приоритетом, то они обслуживаются в первую очередь. При этом пакеты с низким приоритетом оказываются «задавлены», в то время как с высоким даже не знают о них. Это плохо.

Взвешенная справедливая очередь (англ. Weighted fair queuing, WFQ) — механизм планирования пакетных потоков данных с различными приоритетами. Его целью является регулировать использования одного канала передачи данных несколькими конкурирующими потоками. В данном случае под потоком понимается очередь пакетов данных.

Это обобщение алгоритма честных планировщиков (англ. Fair Queuing) (FQ). Оба планировщика имеют отдельные FIFO-очереди для каждого потока данных. Так, если канал со скоростью R используется для N потоков, то скорость обработки каждого из них будет R/N при использовании честного планировщика.

Честный планировщик с приоритетными коэффициентами позволяет регулировать долю каждого потока. Если имеется N активных потоков, с приоритетами $w_1, ..., w_N$, то i-ый поток будет иметь скорость: $\frac{Wwi}{\sum\limits_{i=1}^{N} wi}$.

- FIFO очередь – нет приоритетов, не гарантирована скорость.
- Строгие приоритеты: высокоприоритетный трафик «не видит» низкоприоритетного трафика в сети. Полезно, когда высокоприоритетного трафика ограниченное количество.
- Waited Fair Queuing (WFQ) позволяет каждому потоку обеспечить гарантированный сервис, планируя их в порядке bit-by-bit finishing time.

13 Коммутация пакетов: гарантирование задержки

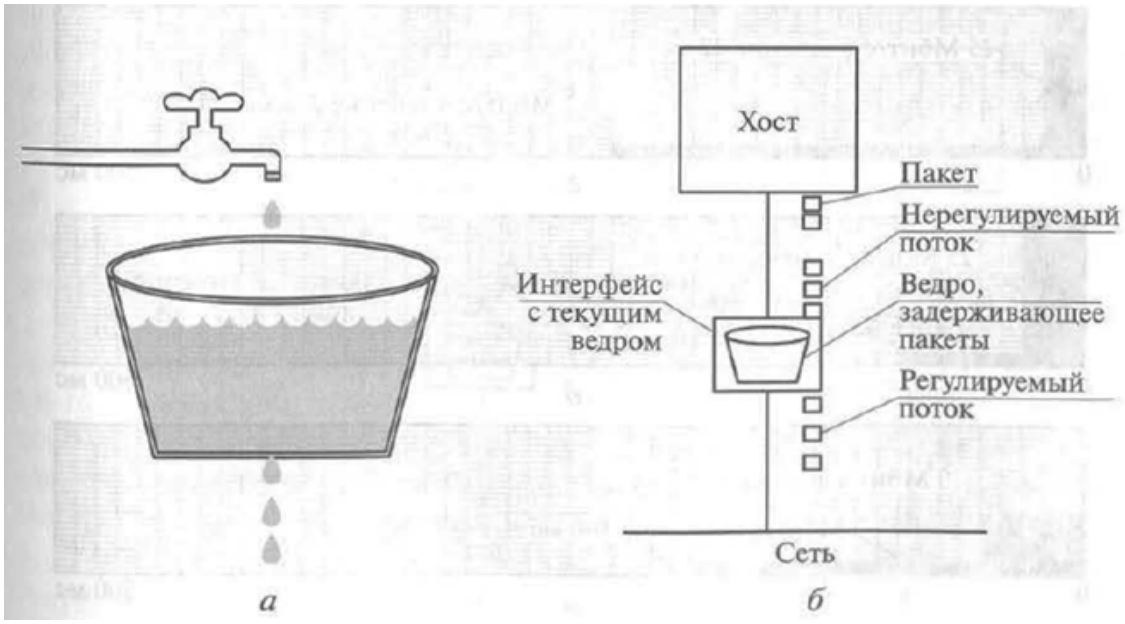
Основные допущения:

- Пакеты не теряются.
- FIFO обслуживание.

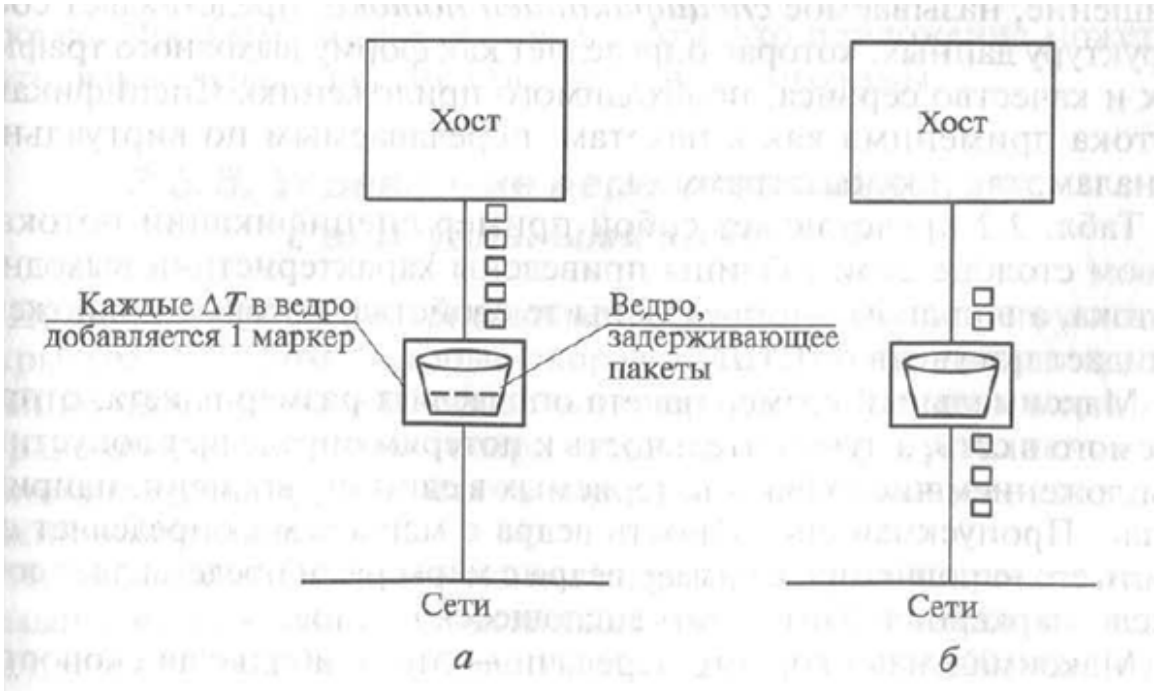
Поскольку мы не можем управлять процессом поступления, его можно ограничить. Пусть число бит, которые могут поступить за период t ограничены $\sigma + \rho t$.Где, например, $\sigma = B$, $\rho = R1$ – размер канала, тогда мы не столкнемся с проблемой переполнения буфера **(для конкретного случая)**.

Одной из причин перегрузок является неравномерный трафик. Если бы этого не было, перегрузок можно было бы избежать. Поэтому используются механизмы формирования трафика, например, **алгоритм текущего ведра**.

Каждая станция, подключенная к сети имеет в своем интерфейсе буфер, подобный ведру и сбрасывающий пакеты при переполнении. Для регулирования скорости поступления пакетов можно, например, использовать системные часы и установить предел числа пакетов, которые можно направить в сеть в промежуток времени. Если пакеты имеют переменную длину – можно ограничивать число байтов, поступающих в сеть.



Иногда бывает полезно ускорить передачу пакетов в сеть, тогда используют **алгоритм текущего ведра с маркерами**. Вместе с пакетами в ведро поступают маркеры, а пакеты выходят только при наличии определенного числа маркеров. Тогда можно накапливать маркеры и кратковременно ускорять передачу пакетов в сеть. Особенность – при переполнении буфера маршрутизатору временно будет запрещено передавать пакеты в сеть.



$C + \rho S = MS$, тогда $S = \frac{C}{M-\rho}$, где S – длительность временного увеличения трафика на входе, ρ – скорость поступления маркеров Б/с, – максимальная скорость входного трафика Б/с, – емкость корзины

в байтах.

Несмотря на то, что технически это возможно, лишь некоторые сети могут управлять e2e задержкой. Причины:

1. Слишком сложно и хлопотно.
2. В большинстве сетей комбинация прогнозирования и приоритетов дает вполне приемлемые результаты.

e2e задержка, $\tau = \sum_i (\frac{p}{r_i} + \frac{l_i}{c} + Q_i(t))$.

- Если мы знаем длину очереди и дисциплину ее обслуживания, то мы можем ограничить величину задержки в ней.
- Выбрав длину очереди, и, используя WFQ, мы можем определить скорость обслуживания.
- Поэтому самое главное не допустить сброса пакетов. Для этого можно использовать текущий буфер.
- Таким образом, мы можем ограничить величину e2e задержки.

14 Управление потоком при пакетной коммутации

В компьютерных сетях неизбежны потери пакетов данных, в частности, из-за переполнения буферной памяти хотя бы одного из узлов, расположенных на пути от источника к приёмнику, включая последний. Такие потери, связанные с переполнениями, в дальнейшем именуются **перегрузками узлов сети**.

Существует множество способов предотвращения и устранения перегрузок; эти способы, в большинстве своем, основаны на управлении потоками данных. Особое место занимает обслуживание пакетов с учетом их приоритетов.

Способ 1. Управление потоком данных регулировкой длительности пауз между пакетами.

Прототип. В процессе передачи данных приемник замечает устойчивую нехватку прибывающих пакетов (например, отслеживая их порядковые номера) и посылает источнику данных управляющий пакет, содержащий команду XOFF приостановки потока данных. Адрес источника данных известен приемнику, так как поступающие к нему пакеты данных содержат информацию об адресах отправителя и адресата. При этом также посылаются запросы на повторную передачу потерянных пакетов.

Получив команду XOFF, источник данных полностью прекращает передачу пакетов и возобновляет ее либо через некоторое время, оговоренное в протоколе обмена данными, либо после получения от приемника команды XON возобновления передачи.

Недостатки:

1. Поток либо есть, либо его нет. Запаздывание выполнения команд может привести к неоправданному простаиванию передатчика и периодическому возникновению новых перегрузок, при которых некоторая часть пакетов, в том числе, принадлежащих другим потокам, будет утеряна.
2. При длительной перегрузке приемник пересылает передатчику серию одинаковых команд приостановки потока, что засоряет канал связи большим количеством повторяющихся служебных пакетов.
3. Команды приостановки работы передатчика формируются приемником только в том случае, когда число отвергнутых в связи с переполнением буфера пакетов достаточно велико.
4. Приостановки работы передатчика увеличивают среднюю и максимальную задержки передачи пакетов по трассе, что может снизить заданные в контракте между пользователем и провайдером показатели качества обслуживания канала.

Решение. Предлагаемое решение в значительной степени устраняет перечисленные недостатки благодаря плавной и «опережающеий события» регулировке скорости передачи данных источником. Скорость регулируется изменением длительности пауз между пакетами: чем больше паузы, тем ниже скорость передачи данных, и наоборот. Отметим, что наличие паузы не означает, что сигнал в линии связи отсутствует – сигнал присутствует постоянно, но в нем нет флаговых кодов, обозначающих начало пакета, либо наоборот – передается непрерывный поток этих кодов.

Способ 2. Управление потоком данных оповещением источника пакетов о причинах перегрузки.
Прототип. В ответ на каждый пакет или на группу пакетов приемник посылает источнику ответные пакеты ACK или NACK. Ответ ACK подтверждает успешный прием, ответ NACK служит запросом повторной передачи одного пакета или группы пакетов. Если источник данных при повышении скорости передачи пакетов или на некоторой фиксированной скорости начинает получать чрезмерное число запросов повторной передачи, то, вероятнее всего, по крайней мере, один из узлов трассы вошел в режим перегрузки. В этом случае источник данных резко снижает скорость передачи пакетов или (и) увеличивает их длину, чтобы уменьшить долю передаваемых в потоке данных служебных битов, образующих заголовки. В дальнейшем источник данных постепенно либо методом случайных проб и ошибок повышает скорость передачи данных, продвигаясь к допустимой верхней границе с учетом некоторого разрешенного запаса повышения скорости. Такой способ называют **«медленным стартом»**.

Рассмотренный способ управления потоком данных не предотвращает предстоящую потерю пакетов, а позволяет реагировать только на свершившийся факт перегрузки промежуточного узла сети или приемника данных. В этом состоит его **основной недостаток**.

Другой прототип. Идея состоит в том, чтобы вовремя предупредить источник данных A об угрозе перегрузки одного или нескольких узлов вдоль трассы AB распространения пакетов данных D. Таким предупреждением служит бит Z, входящий в состав заголовка ответного пакета ACK или NACK.

Решение. Поставленная задача решается расширением одноразрядного признака Z до нескольких битов.

Узел может испытывать перегрузку по крайней мере по одной из следующих причин:

1. Сужение полосы (пропускной способности) канала AB из-за появления «узкого места».
   - Увеличился до значительного уровня ранее малозаметный конкурирующий поток данных по маршруту M4M2M3, который использует тот же канал M2M3, что и маршрут AB. В результате узел M2 перераспределил полосу этого канала в ущерб маршруту AB.
   - Узел M2 изменил тип модуляции сигнала в канале M2M3, снизив скорость передачи в связи с ухудшением соотношения «сигнал/шум» в этом канале.
2. Процессор узла M2 по тем или иным причинам перестал справляться с объемом работы по анализу заголовков пакетов, следующих по маршруту AB.

Первая и вторая причины надвигающейся перегрузки отображаются соответственно кодами $Z = 01$ и $Z = 10$, отсутствие опасности перегрузки соответствует коду $Z = 00$, обе причины одновременно вызывают формирование кода $Z = 11$.

Способ 3. Управление потоком данных с компенсацией инерционности контура обратной связи.
Плавность управления достигается дроблением серий пакетов и более интеллектуальным алгоритмом формирования команд XON и XOFF возобновления и прекращения передачи потока.

15 Заголовок IP, TCP. Фрагментация.

Заголовок TCP.
Каждый сегмент данных имеет заголовок >= 20 байт. Поля заголовка:

1. **Source port** и **Destination port**, указывающие сокеты на стороне отправителя и получателя;
2. **Sequence number** и **Acknowledgement number** – порядковый номер ожидаемого байта и следующего ожидаемого байта, а не последнего полученного байта;
3. 6-битовое **поле флагов**, содержащее следующие биты:
   - **URG** (исп. вместе с полем Urgent pointer), указывает на начало области срочных данных;
   - **ACK** = 1 если поле Acknowledgement number используется, 0 иначе;
   - **PSH** = 1 если отправитель просит транспортного агента не буферизовать данные на стороне получателя;
   - **RST**, используемый для переустановки некорректного соединения (получили этот флаг - значит, есть ошибка)
   - **SYN** = 1 при запросе на соединение. Флаг ACK указывает наличие или отсутствие подтверждения. При SYN = 1 и ACK = 0 — запрос на соединение, а при SYN = 1 и ACK = 1 — подтверждение соединения;
   - **FIN**, используемый при запросе на разрыв соединения (получили этот флаг - значит, у отправителя нет больше данных).
4. **Window size**, используемое алгоритмом управления окном. получатель указывает, сколько байтов может послать отправитель, начиная с последнего подтвержденного байта;
5. **Options**, используемое для возможностей нестандартных заголовков. Например, максимальный размер поля данных, допустимый по данному соединению;
6. **Checksum**, используемое для обнаружения ошибок при передаче. Оно охватывает заголовок, данные и псевдозаголовок.

Псевдозаголовок содержит 32-разрядные IP-адреса отправителя и получателя, номера протокола (для TCP-6) и число байтов в TCP-сегменге, включая заголовок. Для повышения эффективности в RFC 1323 появился параметр «Масштаб окна», задаваемый при установке соединения. Позволяет увеличить поле «Размер окна» до 14 разрядов влево, увеличивая при этом размер окна до 1 Гбайт.

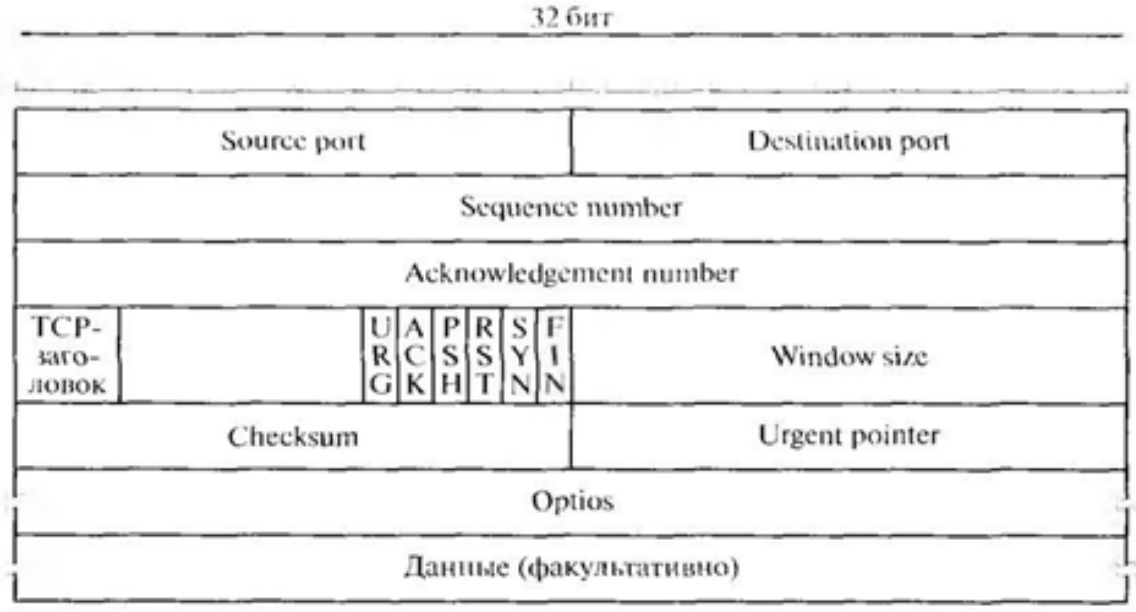


Рис. 3.8. Структура заголовка сегмента TCP-протокола

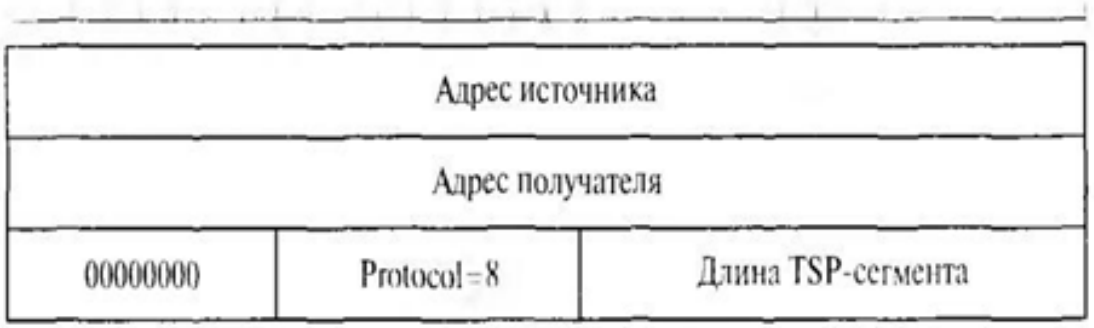


Рис. 3.9. Структура псевдозаголовка, включаемого в контрольную сумму TCP-сегмента

Заголовок IP.
Имеет обязательную часть размером 20-60 байт. Поля заголовка:

- **Version** – версия IP-протокола;
- **IHL** – длина заголовка, который может содержать от 5 до 60 32-разрядных слов;
- **Type of service** – вид необходимого сервиса. (скорость, надежности, приоритет);
- **Total length** – указывает общую длину пакета, вкл. заголовок и поле данных. <= 65 535 б.
- **Identification** – какому пакету принадлежит очередной поступивший фрагмент. Все фрагменты одного и того же пакета имеют в этом поле одно и то же значение;
- **DF** = 1, если фрагментация невозможна;
- **MF** = 0 у последнего фрагмента пакета, 1 у остальных;
- **Fragment offset** – указывает, где в пакете располагается данный фрагмент пакета. Длина всех фрагментов, кроме последнего, должна быть кратна 8 байт. Поскольку поле имеет 13 разрядов, то у одного пакета максимально может быть 8 192 фрагментов. Таким образом, длина минимального фрагмента равна 8 байт;
- **Time to live** – время жизни пакета. <= 255 сек. Часто здесь используется счетчик скачков;
- **Protocol** – показывает, какому протоколу на транспортном уровне передать собранную дейтаграмму (TCP, UDP и т.д.);
- **Header checksum** – контрольная сумма, охватывающая только заголовок;
- **Source address, Destination address** – идентифицируют машину отправителя и получателя на сетевом уровне;
- **Options** – необязательная часть заголовка, предусмотренная для расширения возможностей протокола (Security – уровень секретности: чтобы маршрутизатор мог запретить определенные маршруты, Strict source routing – полный маршрут в виде списка IP-адресов, например, когда таблица маршрутизации оказалась испорченной, Loose source routing – список маршрутизаторов, через которые фрагмент обязан пройти, Time stamp – вместе с полем Record route указывает маршрутизаторам на необходимость записывать не только свои адреса, но и время – удобно для отладки).

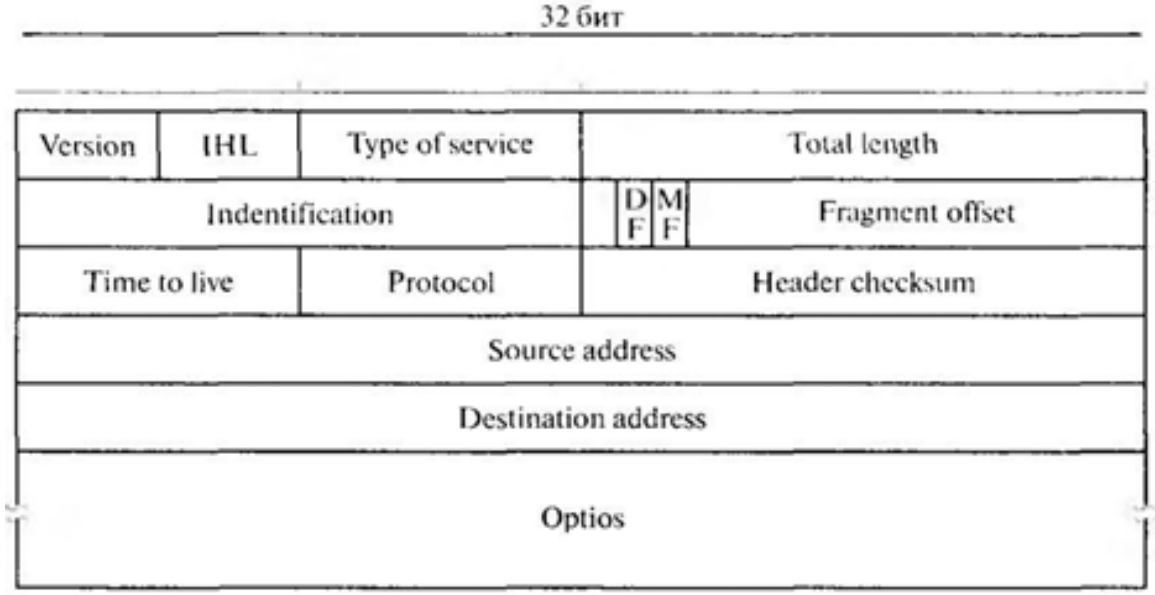


Рис. 2.23. Заголовок IP-пакета

Фрагментация. (Структура текста: проблема – решение.)

В каждой транспортной среде существует свой максимальный размер пакетов, опеределяемый аппаратурой, операционной системой, протоколами, стандартами, желанием предотвратить длительный захват канала.

Проблема*: попытка передать большой пакет через сеть, у которой макс. размер пакета меньше.

Решение*: разрешить шлюзу фрагментацию пакета и независимую передачу фрагментов.

Отсюда – **проблема**** сборки фрагментов.

2 решения:**

1. Делать фрагменты столь малыми, чтобы любая сеть на их пути была для них прозрачна. Тогда все фрагменты проходят через один и тот же выходной шлюз, где и собираются в один пакет. **Проблемы:** как узнать, что все фрагменты достигли выходного шлюза; как выбирать маршрут для фрагментов; как сократить накладные расходы на разбиение и сборку пакета из фрагментов.

2. Разбить пакет на фрагменты и рассматривать каждый из них как независимый отдельный пакет. При этом сборка фрагментов происходит только в узле на значения. **Проблемы:** каждый маршрутизатор должен уметь собирать пакеты из фрагментов; резко возрастают накладные расходы на передачу, так как каждый фрагментированный пакет будет нести свой заголовок.

\+ Еще одна **Проблема***:** нумерация фрагментов, принадлежащих разным пакетам – она должна обеспечивать не только восстановление исходною пакета, но и восстановление утерянных или поврежденных при передаче фрагментов.

Решение*:** определение минимального размера фрагмента, который бы проходил через любую транспортную среду между отправителем и получателем.

16 Методы обнаружения ошибок при передаче сетевого трафика.

Добавление контрольной суммы в пакеты IP, TCP. Возможна быстрая аппаратная реализация, просто, но ненадежно — слабая защита от ошибок (только одиночные ошибки). IP, UDP и TCP используют один и тот же алгоритм комплементарной контрольной суммы:

- Установить поле $checksum = 0$.
- Найти сумму всех 16-разрядных слов в пакете.

- Установить разряд четности.
- Контрольная сумма должна быть такой, чтобы сумма всего пакета, включая контрольную сумму была бы 0xffff.

Также можно использовать коды для обнаружения/исправления ошибок, например, код Хемминга.

CRC — Cyclic Redundancy Check. Отправитель и получатель договариваются о конкретном генераторе полиномов $G(x)$ степени r (коэффициенты при старшем члене и при младшем члене должны быть равны 1). Для вычисления контрольной суммы блока из m бит надо чтобы обязательно $m > r$. Добавить контрольную сумму к передаваемому блоку, рассматриваемому как полином $M(x)$ так, чтобы передаваемый блок с контрольной суммой был кратен $G(x)$. Когда получатель получает блок с контрольной суммой, он делит его на $G(x)$. Если есть остаток, то при передаче были ошибки.

Алгоритм вычисления контрольной суммы:

- Добавить r нулей в конец блока так, что он теперь содержит $m + r$ разрядов и соответствует полиному $(x^r)M(x)$.
- Разделить по модулю 2 полином $(x^r)M(x)$ на $G(x)$.
- Вычесть остаток (длина которого всегда не более r разрядов) из строки, соответствующей $(x^r)M(x)$, по модулю 2. Результат и есть блок с контрольной суммой.

Данный метод позволяет обнаруживать одиночные ошибки, групповые ошибки длинной не более r и нечетное число отдельных ошибок. Существует три международных стандарта на вид полинома $G(x)$:

- CRC-12 $= x^{12} + x^{11} + x^3 + x^2 + x + 1$.
- CRC-16 $= x^{16} + x^{15} + x^2 + 1$.
- CRC-CCITT $= x^{16} + x^{12} + x^5 + 1$.

Стандарт CRC-12 используется для передачи символов из шести разрядов, а CRC-16 и CRC-CCITT — из восьми. Стандарты CRC-16 и CRC-CCITT ловят одиночные, двойные ошибки, групповые ошибки длиной не более 16 и нечетное число изолированных ошибок.

MAC — Message Authentication Code. Не столь устойчиво к ошибкам, как CRC. Защищает от злоумышленников. Используются криптографические функции для вычисления контрольной суммы $m = \text{MAC}(M, s)$, $|m| < |M|$.

- Если M известно, s — секретно, то возможно проверить $m = \text{MAC}(M, s)$, $|m| < |M|$.
- Если s не известно, то получить m практически не возможно.
- Если известно m, то практически не возможно вычислить M, даже зная MAC.

17 Протокол TCP: установка и разрыв соединения.

Установка соединения. В протоколе TCP-соединения устанавливаются с помощью «тройного рукопожатия». Чтобы установить соединение, одна сторона (например, сервер) пассивно ожидает входящего соединения, выполняя примитивы LISTEN и ACCEPT, либо указывая конкретный источник, либо не указывая его.

Другая сторона (например, клиент) выполняет примитив CONNECT, указывая IP-адрес и порт, с которым он хочет установить соединение, максимальный размер TCP-сегмента и, по желанию, некоторые данные пользователя (например, пароль). Примитив CONNECT посылает TCP-сегмент с установленным битом SYN и сброшенным битом ACK и ждет ответа.

Когда этот сегмент прибывает в пункт назначения, TCP-сущность проверяет, выполнил ли какой-нибудь процесс примитив LISTEN, указав в качестве параметра тот же порт, который содержится в поле

Порт получателя. Если такого процесса нет, она отвечает отправкой сегмента с установленным битом RST для отказа от соединения.

Если какой-либо процесс прослушивает какой-либо порт, то входящий TCP-сегмент передается этому процессу. Последний может принять соединение или отказаться от него. Если процесс принимает соединение, он отсылает в ответ подтверждение.

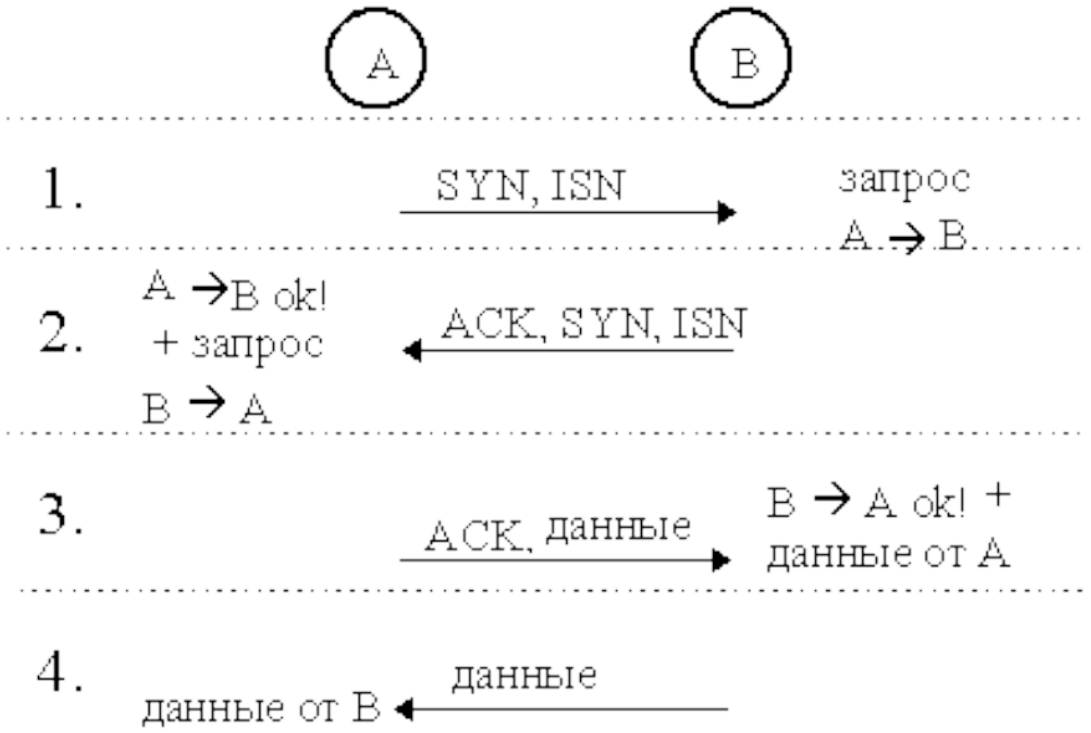
Столкновение вызовов (коллизия): когда два хоста одновременно пытаются установить соединение между двумя одинаковыми сокетами. Будет установлено только одно соединение, так как пара конечных точек однозначно идентефицирует соединение.

Разрыв соединения. Хотя TCP-соединения являются дуплексными, чтобы понять, как происходит их разъединение, лучше считать их парами симплексных соединений. Каждое симплексное соединение разрывается независимо от своего напарника. Чтобы разорвать соединение, любая из сторон может послать TCP-сегмент с установленным в единицу битом FIN, что означает, что у него больше нет данных для передачи. Когда этот TCP-сегмент получает подтверждение, это направление передачи закрывается.

Тем не менее, данные могут продолжать передаваться неопределенно долго в противоположном направлении. Соединение разрывается, когда оба направления закрываются. Обычно для разрыва соединения требуются четыре TCP-сегмента: по одному с битом FIN и по одному с битом ACK в каждом направлении. Первый бит ACK и второй бит FIN могут также содержаться в одном TCP-сегменте, что уменьшит количество сегментов до трех.

Как при телефонном разговоре, когда оба участника могут одновременно попрощаться и повесить трубки, оба конца TCP-соединения могут послать FIN-cегменты в одно и то же время. Они оба получают обычные подтверждения, и соединение закрывается. По сути, между одновременным и последовательным разъединениями нет никакой разницы.

Чтобы избежать проблемы двух армий, используются таймеры. Если ответ на посланный FIN-сегмент не приходит в течение двух максимальных интервалов времени жизни пакета (120 секунд), отправитель разрывает соединение. Противоположная сторона также по истечении этого периода времени узнает, что никто от нее не ждет ответа.



18 Явление перегрузки и основные методы борьбы с ней.

(Том 2, стр. 52-56.)

Когда в транспортной среде находится в одно и то же 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. Если буфер маршрутизатора переполнится, то пакеты начнут теряться. 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. 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.

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.

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:

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

Методы:

1. Схема управления потоком (небольшое окно) сдерживает нарастание трафика и предотвращает появление перегрузок.
2. 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.
3. Выбор метода сброса пакетов также влияет на перегрузки. Правильная маршрутизация, равномерно использующая каналы в транспортной среде, позволяет избежать перегрузки.
4. Методы, регулирующие время жизни пакета в сети, также влияют на образование перегрузок.

19 Перегрузка: AIMD в случае одного потока и в случае нескольких потоков

(Слайды лекций – week_07.)

Additive-increase/multiplicative-decrease (AIMD) алгоритм является алгоритмом управления перегрузками с обратной связью. AIMD сочетает линейный рост окна с экспоненциальным сокращением, когда происходит сброс пакета.

Пусть W – размер окна, тогда:

- Если пакет **получен**, то: $W = W + \frac{1}{W}$.
- Если пакет **сброшен**, то: $W = \frac{W}{2}$.

AIMD в случае одного потока:

1. Окно увеличивают, сокращают в соответствии с AIMD, можно определить как много байт канал еще может вместить.

2. Пилообразное поведение графика размера окна от времени – это нормальная форма динамики.
3. Скорость отправки постоянная: $R = \frac{W}{RTT}$. **(RTT – Round-trip time)**, если есть достаточно буферного пространства (R * RTT > W).

AIMD в случае нескольких потоков:

1. Окно увеличивают, сокращают в соответствии с AIMD.
2. В «узком месте» будут скапливаться пакеты разных потоков.
3. Скорость отправки меняется в зависимости от размера окна.
   Доля теряемых пакетов: $p = \frac{1}{A}$, где $A = \frac{3}{8}W_{max}^2$, $R = \frac{A}{\frac{W_{max}}{2}RTT}$.
   $R = \sqrt{\frac{3}{2}}\frac{1}{RTT\sqrt{p}}$.
4. AIMD очень чувствителен к частоте потери пакетов.
5. AIMD ущемляет потоки с большим RTT.

20 Управление передачей в TCP: алгоритм Tahoe

Используемые значения:

- **MSS** – максимальный размер сегмента.
- **CWND** – окно перегрузки.
- **ssthresh** – порог медленного старта.
- **RTT** – время круговой задержки.

Стратегия TCP Tahoe заключается в следующем:

1. Используя медленный старт, быстро «нащупать» доступную пропускную способность сети.
2. Приблизившись к насыщению сети, перейти в режим предотвращения перегрузки.

Медленный старт.
Устанавливаем $CWND = MSS$.
Затем отправитель передает в сеть пакеты в количестве, равном CWND. Получение пакетов подтвердится через время, равное круговой задержке (RTT). В каждом случае, когда подтверждение о получении сегмента приходит до срабатывания таймера повторной передачи, отправитель увеличивает окно перегрузки (CWND) на MSS.
В результате, получится экспоненциальный рост числа пакетов в сети.
Пример:

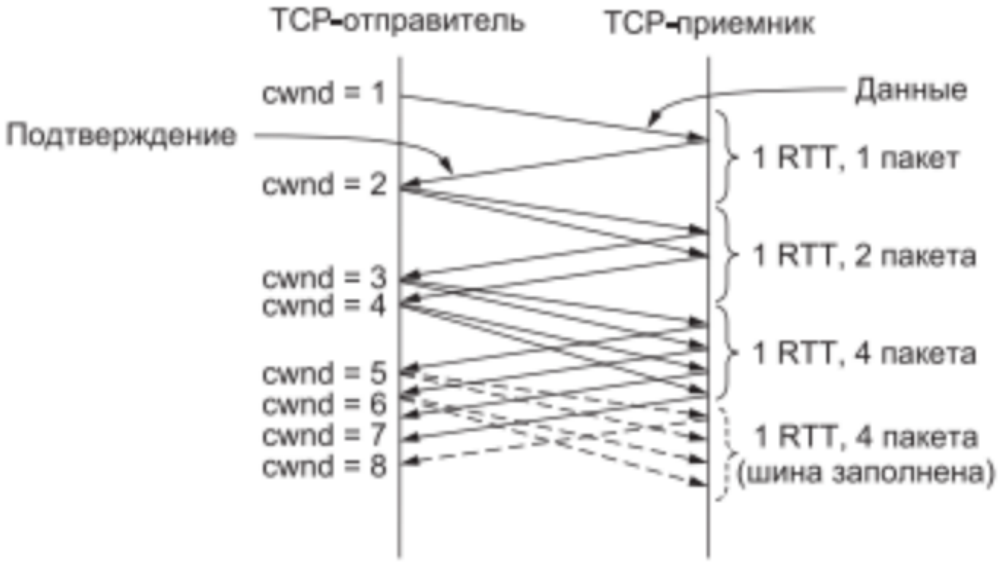


Рис. 6.38. Медленный старт с начальным окном перегрузки в один сегмент

Поскольку алгоритм медленного старта приводит к экспоненциальному росту, то в какой-то момент в сеть будет отправлено слишком много пакетов за короткое время. Это приведет к образованию очередей на маршрутизаторах. Когда очереди переполняются, происходит потеря пакетов. Чтобы такого не происходило, используют порог медленного старта (ssthresh). Изначально устанавливается произвольное высокое значение, не превышающее размер окна управления потоком, чтобы оно никак не ограничивало возможности соединения. Используя алгоритм медленного старта, TCP продолжает увеличивать окно перегрузки, пока не произойдет тайм-аут или размер окна перегрузки не превысит пороговое значение (или не заполнится окно получателя). При обнаружении потери пакета (например, в ситуации тайм-аута) $ssthresh = \frac{CWND}{2}$, где CWND – размер окна перегрузки, при котором произошла потеря пакета, окно перегрузки устанавливается равным MSS. И алгоритм медленного старта повторяется.

Предотвращение перегрузки.

Когда CWND становится равным или превышает ssthresh, происходит переход с медленного старта на режим предотвращения перегрузки. В этом режиме значение CWND увеличивается на $\frac{MSS^2}{CWND}$ для каждого из подтвержденных о доставке $\frac{CWND}{MSS}$ пакетов. Т.е. за каждый RTT окно перегрузки увеличивается на MSS. Получаем линейный рост для размера окна перегрузки. Смысл данного режима заключается в том, что значение окна перегрузки «удерживается» в области максимально возможных значений (где сеть загружена, но не происходит потеря пакетов).

Пример (аддитивное увеличение = режим предотвращения перегрузки):

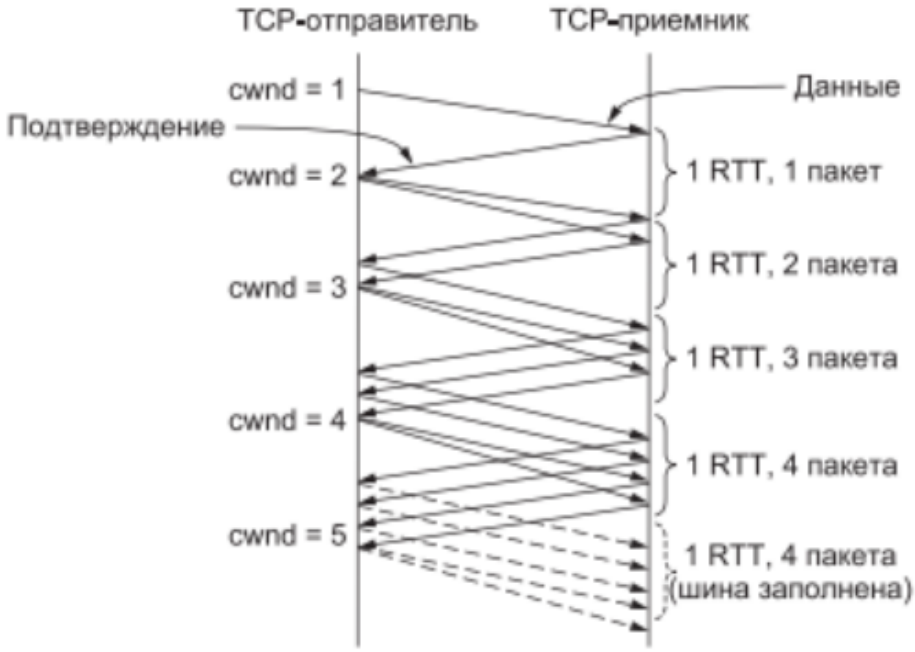


Рис. 6.39. Аддитивное увеличение при начальном окне размером один сегмент

В TCP Tahoe есть три типа сигналов (по лекциям Смелянского):

1. **Рост числа уведомлений** – передача данных идет хорошо.
2. **Повторные уведомления** – где произошла задержка/потеря данных.
3. **Time_out** – что-то работает не так как надо.

Повторные уведомления.

Для отправителя существует быстрый способ узнать, что один из его пакетов потерян. По мере того как пакеты, следующие за потерянным пакетом, прибывают на приемник, они инициируют отправку подтверждений, которые приходят к отправителю.

Все они имеют один и тот же номер подтверждения и называются **повторными уведомлениями**. Каждый раз, когда отправитель получает дубликат подтверждения, естьвероятность, что другой пакет уже пришел, а потерянный – нет.

Так как пакеты могут следовать разными путями, они могут приходить в неправильном порядке. В таком случае дубликаты подтверждений не будут означать потерю пакетов. Однако в Интернете такое случается достаточно редко. Если в результате прохождения пакетов по разным путям порядок пакетов все же нарушается, он нарушается не сильно. Поэтому в TCP условно считается, что три повторных уведомления сигнализируют о потере пакета. Также по номеру подтверждения можно узнать, какой пакет утерян (это следующий по порядку пакет).

Оценка time_out.

Основана на RTT измерении, которое критично для оценки time_out:

- Если слишком коротко, то впустую тратим ресурсы сети на повторные передачи.
- Если слишком длинный, то зря тратим ресурсы на ожидание.

При измерении RTT сталкиваемся с трудностями, связанными с тем, что RTT меняется очень динамично и сильно зависит от загрузки сети.

В Tahoe оценка time_out производится так (лекции Смелянского):

1. r – значение RTT из здравого смысла.

2. m – измерение RTT для последнего подтвержденного пакета.
3. Вычисляем взвешенное среднее: $g = ag + (1 - a)m$, где $a \approx 0,25$.
4. Ошибка: $e = m - r$, где m – измерение для последнего ack.
5. Вычисляем: $r = r + ge$.
6. Измеряем вариацию: $v = v + g(|e| - v)$.
7. Time_out = $r + bv$, где $b = 4$.
8. Экспоненциально увеличиваем time_out в случае перегрузки.

Пример работы TCP Tahoe:

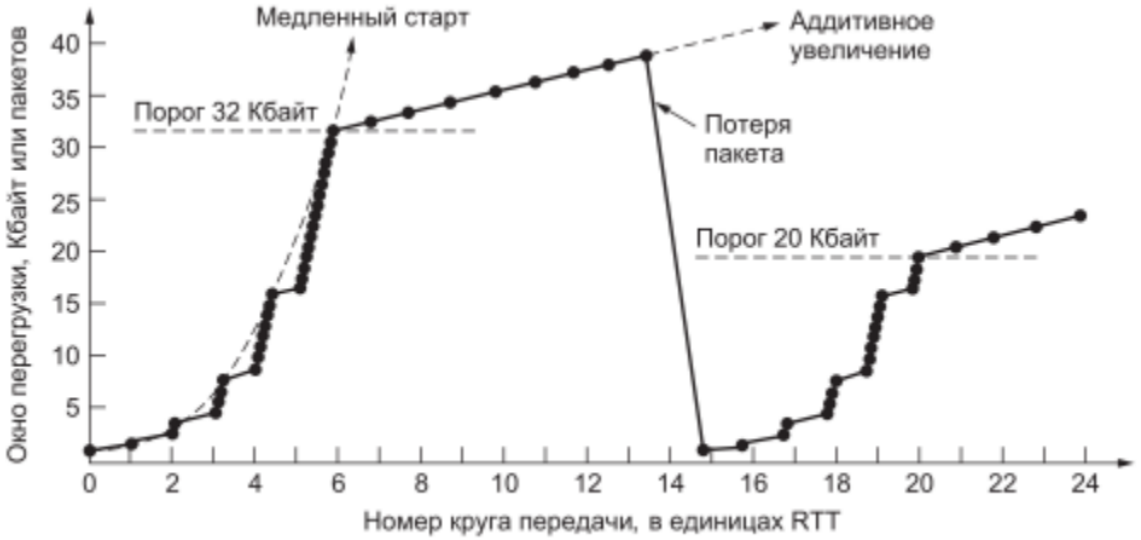


Рис. 6.40. Медленный старт и последующее аддитивное увеличение в TCP Tahoe

21 Управление передачей в TCP: алгоритм Reno

В TCP Reno поведение на сигнал time_out совпадает с поведением TCP Tahoe. Отличается обработка тройного уведомления (по трем повторным уведомлениям в TCP сообщается о потере пакета):

1. Устанавливается порог медленного старта $ssthresh = \frac{CWND}{2}$, где CWND – размер окна перегрузки, при котором произошла потеря пакета.
2. Значение $CWND = \frac{CWND}{2}$ (быстрое восстановление).
3. Повторно пересылает пропущенный сегмент (быстрая пересылка данных, не дожидаясь time_out).
4. Остается в фазе предотвращения перегрузки.

Быстрое восстановление.
Это временный режим, позволяющий не останавливать учет скорости прихода подтверждений в тот момент, когда порогом медленного старта становится текущий размер окна или его половина (во время быстрой повторной передачи). Для этого считаются дубликаты подтверждений (включая те три, которые инициировали быструю повторную передачу) до тех пор, пока число пакетов в сети не снизится до нового порогового значения. На это уходит примерно половина круговой задержки. Начиная с этого момента для каждого полученного дубликата подтверждения отправитель может передавать в сеть новый пакет. Через один круг после быстрой повторной передачи получение потерянного пакета подтвердится.

В этот момент дубликаты подтверждений перестанут приходить сплошным потоком, и алгоритм выйдет из режима быстрого восстановления. Окно перегрузки станет равным новому порогу медленного старта и начнет увеличиваться линейно. В итоге этот метод позволяет избежать медленного старта в большинстве ситуаций, за исключением случаев установления нового соединения и возникновения таймаутов. Последнее может произойти, если теряется более чем один пакет, а быстрая повторная передача не помогает.

Пример работы TCP Reno:

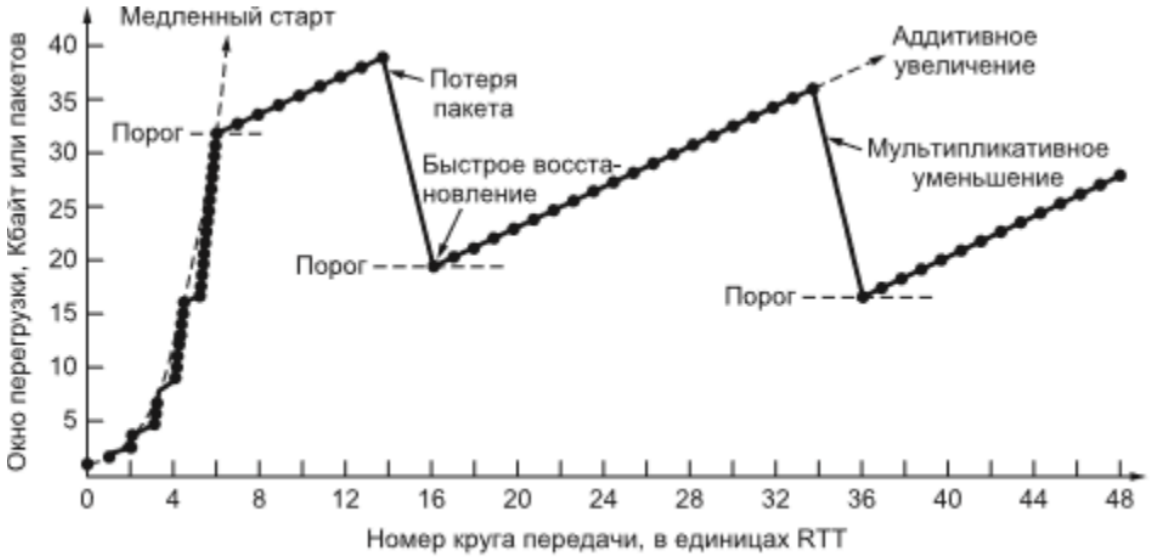


Рис. 6.41. Быстрое восстановление и пилообразный график для TCP Reno

TCP New Reno.
По time_out поведение такое же, как и у Tahoe/Reno. В фазе быстрого восстановления:

1. При входе в эту фазу – запомнить последний неподтвержденный пакет.
2. При каждом повторном уведомлении – увеличить CWND на MSS.
3. Когда последний пакет подтвержден:
   (a) Вернуться в фазу предотвращения перегрузки.
   (b) Восстановить размер CWND до того размера, который оно имело до входа в фазу быстрого восстановления.
4. Начать отправку новых пакетов пока находимся в фазе быстрого восстановления.

22 Маршрутизация в Интернет: основные подходы и маршрутизация по вектору расстояния.

Общие сведения.
Основной задачей сетевого уровня является маршрутизация пакетов. Пакеты маршрутизируются всегда, т.е. независимо от того, какую внутреннюю организацию имеет транспортная среда: с виртуальными каналами или дейтаграммную. Разница состоит лишь в том, что в первом случае этот маршрут устанавливается один раз для всех пакетов, а во втором – для каждого пакета отдельно. Первый случай называют иногда **маршрутизацией сессии**, поскольку маршрут устанавливается на все время передачи данных пользователя, т.е. на время сессии.

Алгоритм маршрутизации реализует программное обеспечение маршрутизатора на сетевом уровне, т.е. он отвечает за определение, по какой из линий, доступных маршрутизатору, отправлять пакет дальше. При этом независимо от выбора маршрута (для сессии или для каждого пакета в отдельности) алгоритм маршрутизации должен обладать **следующими свойствами**: корректностью, простотой, устойчивостью, стабильностью, справедливостью и оптимальностью.

Корректность – свойство алгоритма маршрутизации, определяющее, что при любых обстоятельствах этот алгоритм либо найдет маршрут для доставки пакета адресату, либо выдаст сообщение о невозможности его доставки. Третьего варианта быть не может. При этом крайне желательно, чтобы алгоритм также сообщил о причинах невозможности доставки пакета.

Простота – свойство, определяющее вычислительную сложность алгоритма маршрутизации: чем она меньше, тем алгоритм проще, и тем меньше ресурсов маршрутизатора тратится на решение задачи маршрутизации.

Устойчивость – свойство алгоритма маршрутизации сохранять работоспособность независимо от каких-либо сбоев, отказов в системе передачи данных или транспортной среде, а также изменений топологии (отключения хостов или машин транспортной среды, разрушения каналов и т.п.). Алгоритм маршрутизации должен адаптироваться ко всем таким изменениям, не требуя при этом перезагрузки транспортной среды или остановки абонентских машин.

Стабильность – весьма важное свойство алгоритма маршрутизации. Существуют алгоритмы, которые никогда не приводят к какому-либо определенному маршруту, как бы долго они ни работали. Это означает, что адаптация алгоритма к изменениям в топологии иди конфигурации транспортной среды мажет оказаться весьма продолжительной, и более того, она может оказаться сколь угодно долгой.

Справедливость – свойство, означающее, что все пакеты независимо оттого, из какого канала они поступили, будут обслуживаться маршрутизатором равномерно, т.е. никакому направлению не будет отдаваться предпочтение, для всех абонентов будет всегда вы-бираться оптимальный маршрут.

Следует отметить, что справедливость и оптимальность часто могут вступать в противоречие друг с другом при неудачном выборе критерия оптимизации. На рис. 2.1 приведен пример такого противоречия. Например если в качестве критерия оптимизации выбрать расстояние, а трафики между A и A', B и B', C и C' уже заполнили канал между X и X', то вместо наикратчайшего маршрута между X и X' потребуется выбирать какой-то другой маршрут, который не будет оптимальным по критерию наикратчайшего расстояния. При этом ситуация коренным образом изменится, если выбрать в качестве критерия оптимизации время задержки при доставке.

Прежде чем искать компромисс между оптимальностью и справедливостью, необходимо решить, что является **критерием оптимизации маршрута**. Один из возможных критериев – средняя задержка пакета (обратите внимание, что именно средняя задержка).

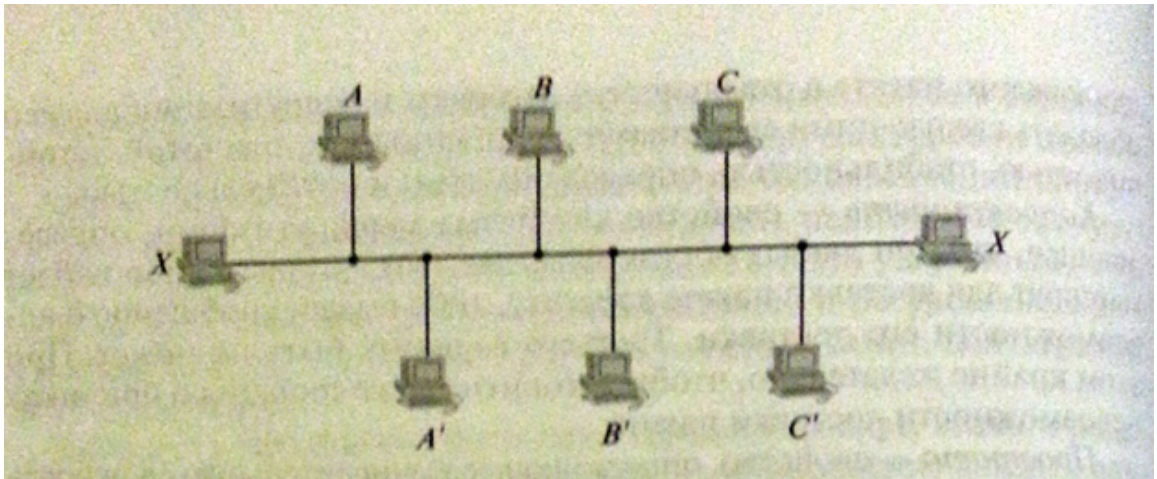


Рис. 2.1. Пример противоречия между справедливостью и оптимальностью

Другой критерий – пропускная способность транспортной среды. 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 называется **скачком**, или **переходом** (hop). Уменьшение числа скачков сокращает маршрут, а следовательно, сокращает задержку и минимизирует необходимую пропускную способность СПД для передачи пакета.

Алгоритмы маршрутизации можно разбить на два больших класса: **адаптивные** и **неадаптивные**. Неадаптивные алгоритмы не принимают в расчет текущую загрузку сети и ее текущую топологию. Все возможные маршруты вычисляются заранее и загружаются в маршрутизаторы при загрузке сети. Такая маршрутизация называется статической.

Адаптивные алгоритмы, наоборот, определяют маршрут исходя из текущей загрузки и топологии транспортной среды. Адаптивные алгоритмы различаются способом получения информации (локально от соседних маршрутизаторов или глобально от всех маршрутизаторов), временем изменения маршрута (через каждые T секунд либо только когда изменяется нагрузка, либо когда изменяется топология) и метрикой, используемой при оптимизации (расстояние, число скачков, ожидаемое время передачи и т.н.).

Свойство оптимального пути.

Обоснуем одно важное предположение о свойстве оптимального маршрута, которое будет использоваться в дальнейших рассуждениях. Это свойство состоит в том, что если маршрутизатор J находится на оптимальном пути между маршрутизаторами I и K (рис. 2.2, а), 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, противоречило бы утверждению об оптимальности маршрута между I и K. Дело в гом, что если представить маршрут от I до K, как от I и J (назовем его S_1) и от J и K (назовем его S_2) и если между J и K имеется маршрут лучше, чем S_2, например S_3, то маршрут S_1S_2 не может быть лучшим. Взяв конкатенацию маршрутов S_1S_3, мы получим лучший маршрут, чем маршрут S_1S_2. Следствием из этого свойства является утверждение, что все маршруты к заданной точке транспортной среды образуют дерево с корнем в этой точке. Это дерево, называемое **деревом захода**, показано на рис. 2.2, б.

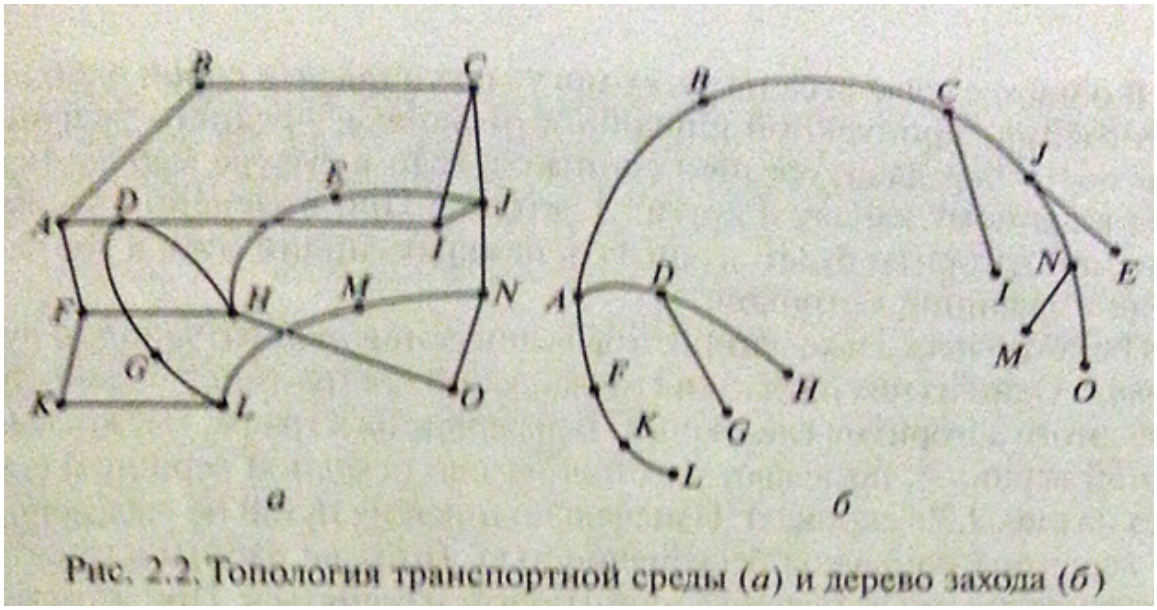


Рис. 2.2. Топология транспортной среды (*а*) и дерево захода (*б*)

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

Маршрутизация по вектору расстояния.

Во всех современных транспортных средах используется динамическая маршрутизация, а не статическая. Один из наиболее популярных алгоритмов динамической маршрутизации – маршрутизация по вектору расстояния. Этот алгоритм, построенный на идеях алгоритма Беллмана–Форда для нахождения

наикратчайшего пути и алгоритма Форда–Фолкерсона, определяющего максимальный поток в графе, изначально использовался в сети ARPA и используется в настоящее время в протоколе RIP (Routing IP), а также в сетях Novell, AppleTalk и маршрутизаторах Cisco.

Алгоритм маршрутизации по вектору расстояния работает следующим образом: у каждого маршрутизатора в транспортной среде есть таблица расстояний до всех других маршрутизаторов, принадлежащих этой транспортной среде. Периодически каждый маршрутизатор обменивается этой информацией со своими соседями и обновляет информацию в своей таблице. Каждый элемент этой таблицы включает в себя два поля: первое – номер канала, по которому следует отправлять пакеты, чтобы достичь нужного места, второе – значение задержки до места назначения, которое может измеряться в разных единицах: числе скачков, миллисекундах, длине очереди к каналу 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.

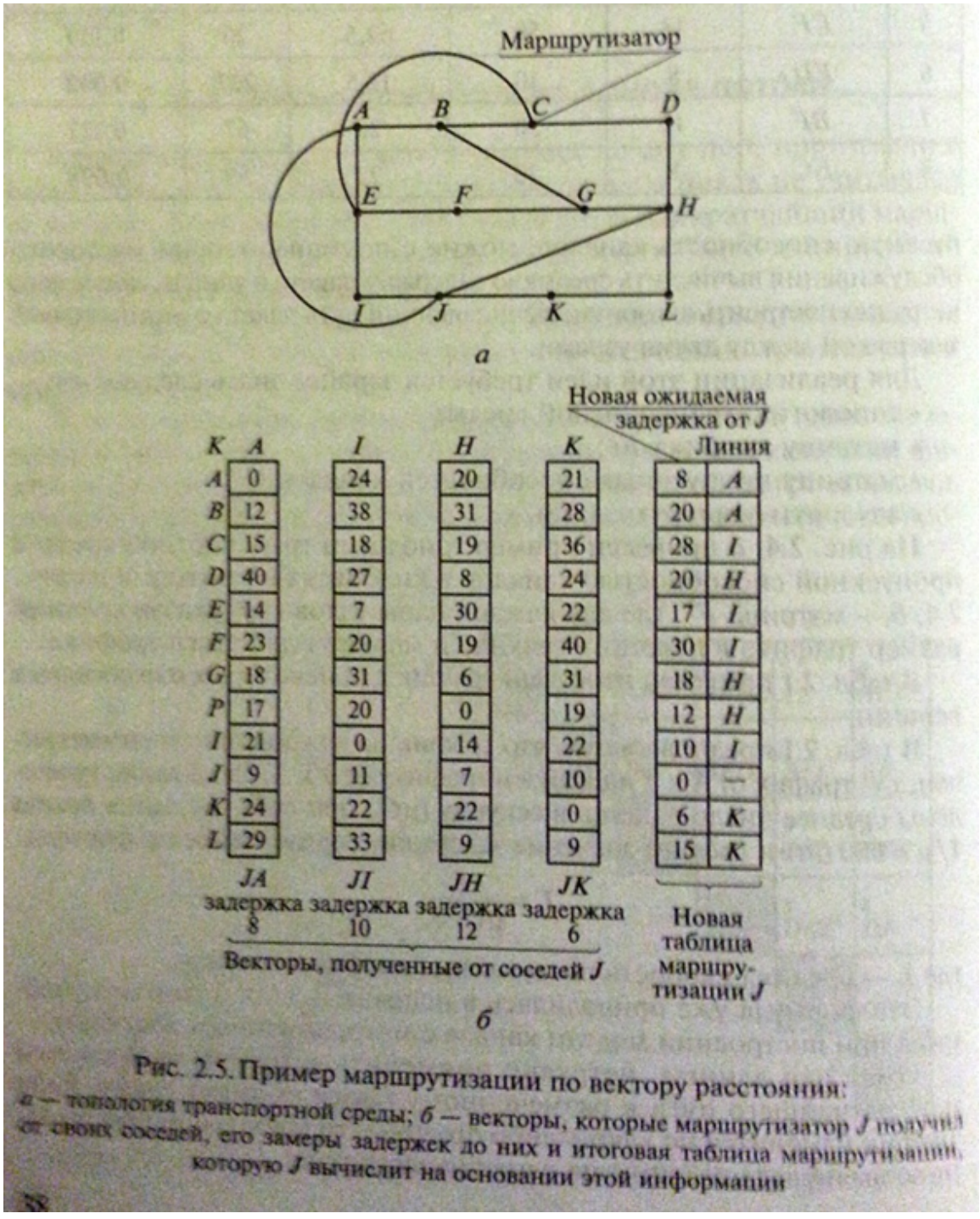


Рис. 2.5. Пример маршрутизации по вектору расстояния:
а — топология транспортной среды; *б* — векторы, которые маршрутизатор *J* получил от своих соседей, его замеры задержек до них и итоговая таблица маршрутизации, которую *J* вычислит на основании этой информации

Каждые T секунд маршрутизатор шлет своим соседям свой вектор задержек до всех маршрутизаторов в транспортной среде. В свою очередь, он получает такие же векторы от своих соседей. Кроме

того, он постоянно замеряет задержки до своих соседей, следовательно, имея векторы расстояний от соседей и зная расстояние до них, маршрутизатор всегда может вычислить наикратчайший маршрут до определенного места в транспортной среде.

На рис. 2.5 приведен пример маршрутизации по вектору расстояния.

Рассмотрим, как маршрутизатор J с помощью итоговой таблицы маршрутизации вычислит маршрут до G. Маршрутизатор J знает, что он может достичь A за 8 мс, маршрутизатор A объявляет, что от него до G 18 мс. Таким образом, J может достичь G за 26 мс через A. Аналогично можно подсчитать, что достичь G через I, H и K можно соответственно за 41 (31 + 10), 18 (6 + 12) и 37 (31 + 6) мс. Наилучшее полученное значение – 18, следовательно, этот маршрут и является наилучшим по критерию скорости доставки.

Проблема бесконечного счетчика задержки.

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 транспортной среде распространяется более или менее быстро, а вот данные о потере, разрушении какого-то маршрута распространяются значительно медленнее.

Рассмотрим пример транспортной среды с линейной топологией. Показанный на рис. 2.6, а. Пусть изначально маршрутизатор A не работал, поэтому у всех маршрутизаторов расстояние до него было равно ∞. Пусть в какой-то момент времени A включился. 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 при каждом обмене векторами. Если самый длинный маршрут в транспортной среде имеет длину N, то потребуется N обменов векторами, пока информация о новом маршруте дойдет до самого удаленного узла.

Теперь рассмотрим обратную ситуацию, показанную на рис. 2.6, б, т.е. когда изначально все маршрутизаторы и каналы работоспособны.

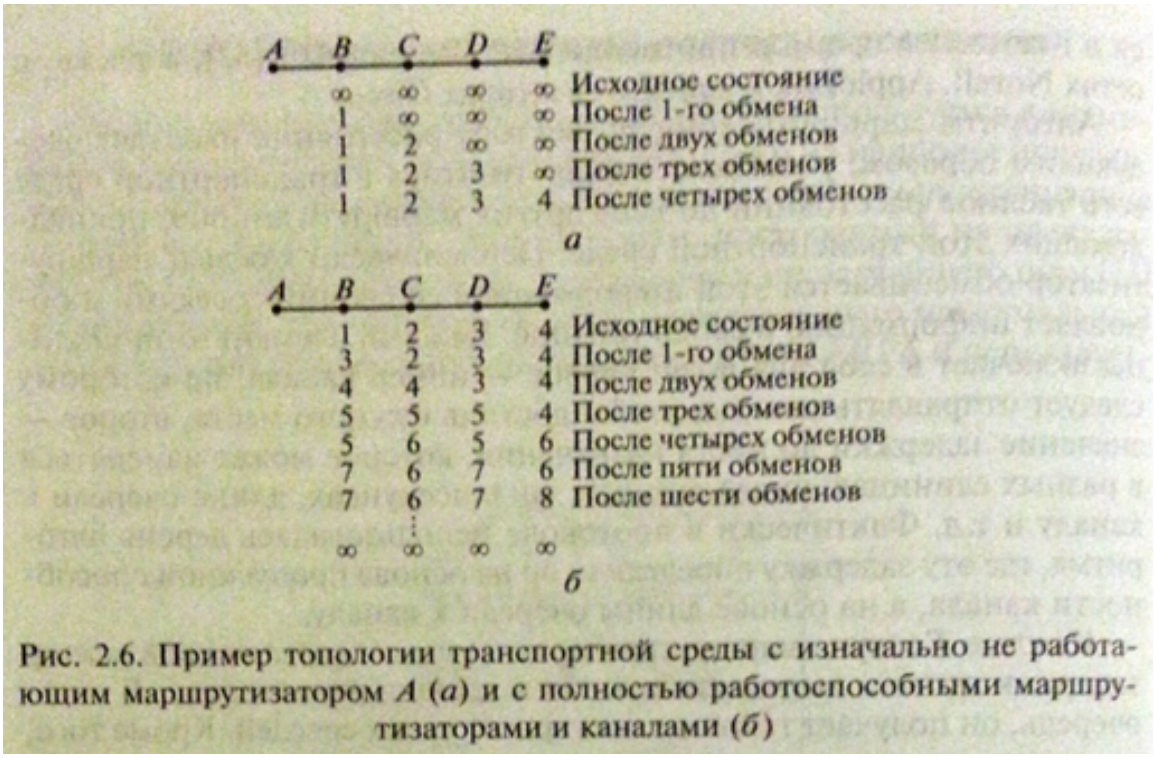


Рис. 2.6. Пример топологии транспортной среды с изначально не работающим маршрутизатором *A* (*а*) и с полностью работоспособными маршрутизаторами и каналами (*б*)

Пусть в какой-то момент времени канал между A и B оказался разрушен. В этом случае B перестает видеть A, но C говорит B: «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 изменяют. Их расстояния до A на единицу больше, чем у C. При втором обмене C увидит, что оба его соседа достигают A за 3 единицы. Некоторым случайным образом C выбирает одного соседа, увеличив значение 3 на единицу. Плохая

весть будет распространяться медленно, пока счетчики задержек не примут некоторого очень большого значения, практически бесконечного для данной сети. Только после этого станет ясно, что A недостижим ни через C, ни через D, ни через E. Сколько времени на это потребуется зависит от конкретного значения бесконечности в данной транспортной среде.

Разделение направлений (Split Horizon Hack).

Одним из решений проблемы бесконечного счетчика задержки является следующее. Алгоритм маршрутизации по вектору расстояния работает так, как было описано ранее, 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. Теперь у алгоритма маршрутизации по вектору расстояния (см. рис. 2.6, б) проблемы бесконечного счетчика не возникнет. Действительно, когда маршрутизатор A «упадет», при первом же обмене B это обнаружит, но C также будет посылать вектор B, согласно которому A недостижим (∞). При следующем обмене C увидит, что A недостижим и через B, и через D, и также отметит его как недостижимый узел.

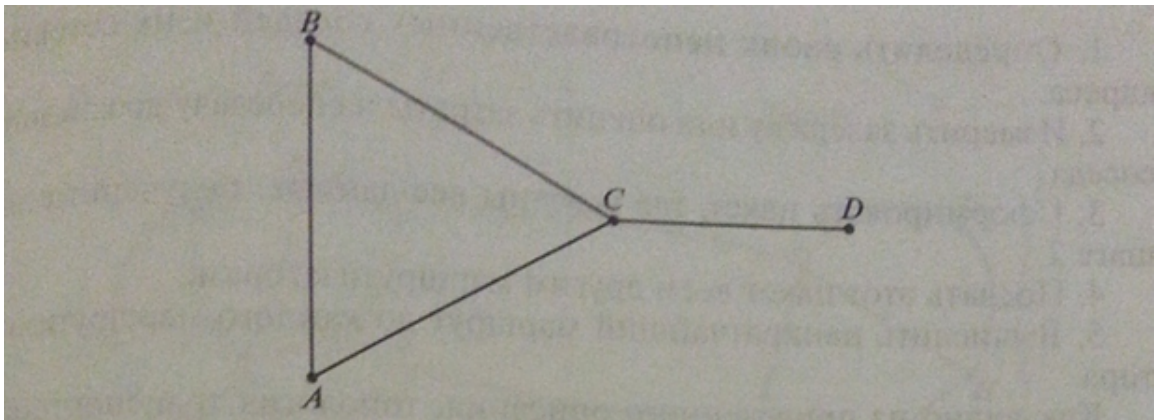


Рис. 2.7. Пример, когда алгоритм разделения направлений не помогает: *A ... D* — маршрутизаторы

Однако и в алгоритме разделения направлений есть «дыры». Рассмотрим пример, показанный на рис. 2.7. 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.

23 Маршрутизация в Интернет: основные подходы и маршрутизация по состоянию канала.

См. **«общие сведения»** и **«свойство оптимального пути»** в вопросе №22.

Маршрутизация по состоянию канала.

Алгоритм маршрутизации по вектору расстояний использовался в сети ARPANET до 1979 г., после чего он был заменен по двум основным причинам. Первая причина – это то, что в нем никак не учитывалась пропускная способность канала, поскольку задержка в основном определялась длиной очереди. Пока основная масса каналов имела пропускную способность 56 Кбит/с, это было не страшно. Однако, когда появились каналы с пропускной способностью 256 Кбит/с и 1,5 Мбит/с, это стало недостатком. Вторая причина – это медленная сходимость алгоритма при изменениях в транспортной среде. По этим причинам был создан новый алгоритм маршрутизации по состоянию канала.

Идею алгоритма маршрутизации по состоянию канала можно описать в виде пяти основных шагов, которые должен выполнить каждый маршрутизатор в транспортной среде:

1. Определить своих непосредственных соседей и их сетевые адреса.

2. Измерить задержку иди оценить затраты на передачу до каждого соседа.

3. Сформировать пакет, где укатаны все данные, полученные на шаге 2.

4. Послать этот пакет всем другим маршрутизаторам.

5. Вычислить наикратчайший маршрут до каждого маршрутизатора.

Как видно из приведенного описания, топология транспортной среды и все задержки в СПД оцениваются всеми узлами экспериментально. После этого можно использовать, например, алгоритм Дейкстры для вычисления наикратчайшего маршрута. Теперь рассмотрим подробнее эти пять шагов.

Определение соседей.

При загрузке маршрутизатор, прежде всего, определяет, кто его непосредственные соседи. Для этого он рассылает по всем подсоединенным к нему каналам специальный пакет HELLO, и все маршрутизаторы отвечают, указывая свое уникальное имя. Имя маршрутизатора должно быть уникальным в сети, чтобы избежать неоднозначностей.

Если же два и более маршрутизаторов соединены каналом с множественным доступом, как на рис. 2.8 а, то этот канал в графе связей представляют отдельным искусственным узлом (рис. 2.8, б).

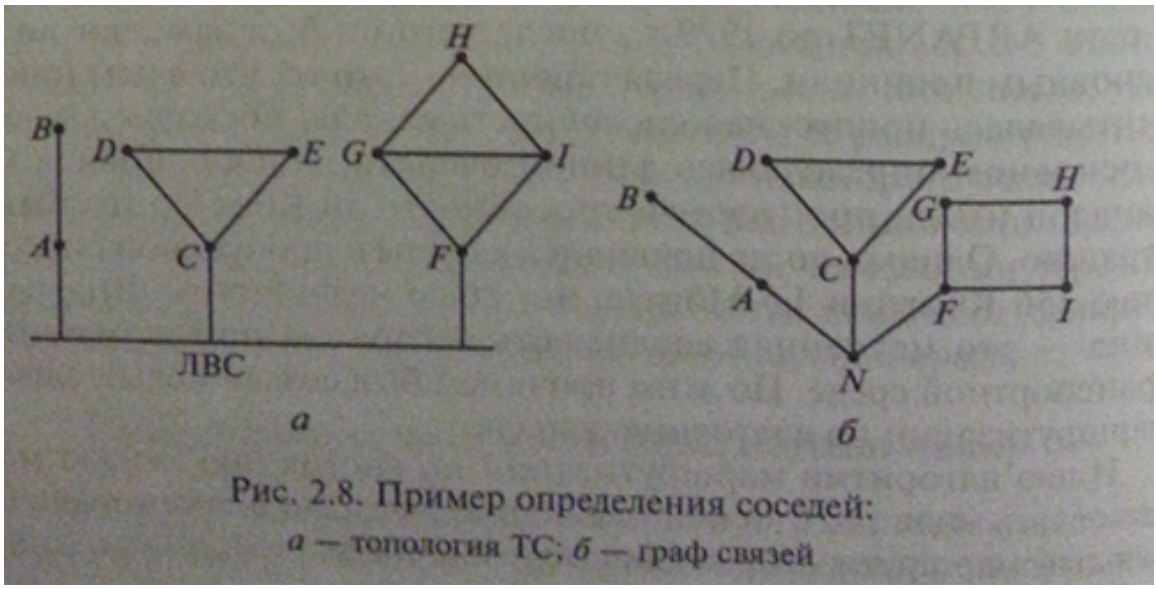


Рис. 2.8. Пример определения соседей:
а — топология ТС; *б* — граф связей

Оценка затрат.

Оценка затрат до каждого соседа происходит с помощью другого специального пакета ECHO, который рассылается **всем соседям**, при этом замеряется задержка от момента отправки этого пакета до момента его возвращения. Все маршрутизаторы, которые получают такой пакет, обязаны отвечать незамедлительно, отправляя пакет обратно. Такие замеры делают несколько раз и вычисляют их среднее значение. Таким образом, длина очереди к каналу не учитывается.

Здесь необходимо понять: следует учитывать загрузку канала или нет? Если учитывать, то задержку надо замерять от момента поступления пакета в очередь к каналу, а если не учитывать, то – от момента, когда пакет достиг головы очереди. Есть доводы и за учет загрузки, и против ее учета. При учете загрузки можно выбирать между двумя и более каналами с одинаковой пропускной способностью, обеспечивая лучшую производительность.

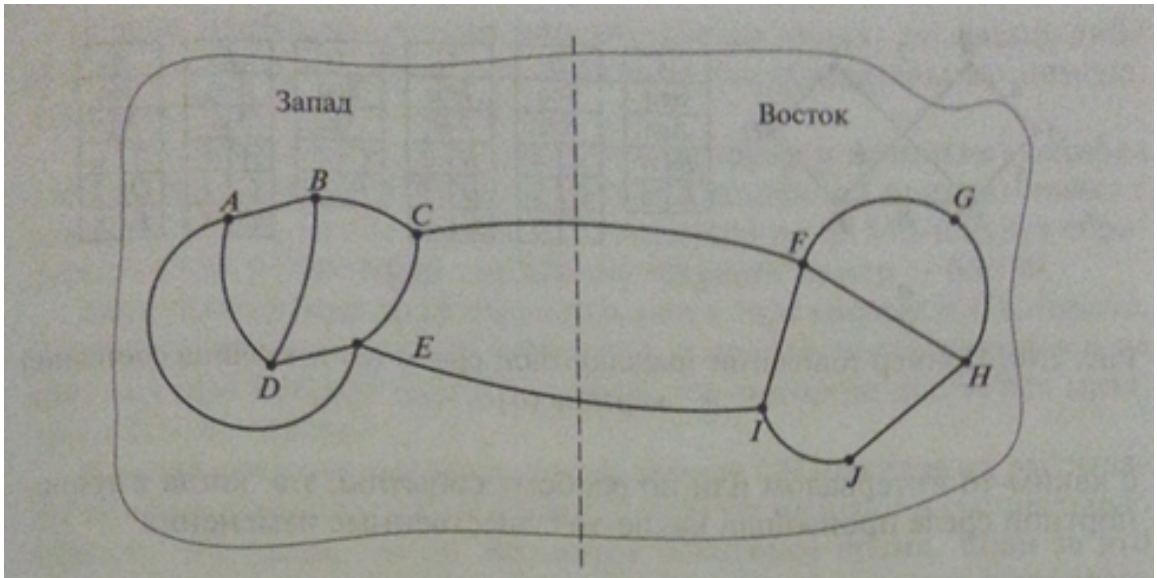


Рис. 2.9. Пример учета загрузки канала при маршрутизации по состоянию

Однако можно привести примеры, когда учет загрузки вызывает проблемы. Например, рассмотрим рис. 2.9, где два одинаковых по производительности канала CF и EI соединяют две транспортные среды. Когда загрузка одного из этих каналов меньше, то он предпочтительнее другого, что через некоторое время приводит к его перегрузке, и предпочтительнее становится другой канал. Если загрузка не учитывается, то такой проблемы не возникает.

Формирование пакетов состояний каналов.

После выполнения всех указанных измерений можно формировать пакеты состояний каналов.

На рис. 2.10, б показаны пакеты состояний каналов для примера транспортной среды, приведенной на рис. 2.10, а.

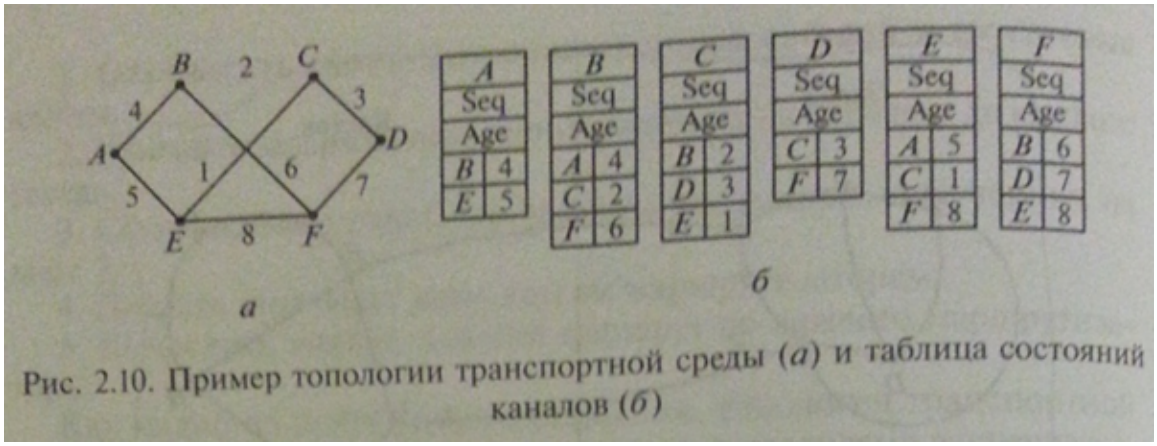


Рис. 2.10. Пример топологии транспортной среды (*а*) и таблица состояний каналов (*б*)

В пакетах состояний указываются: отправитель, последовательное число, возраст (назначение этих полей станет ясно позднее), список соседей и задержки до них. Формирование таких пакетов не вызывает проблем. Основной вопрос здесь, когда их формировать: периодически с каким-то интервалом или по особому событию, т.е. когда в транспортной среде произошли какие-то существенные изменения?

Распространение пакетов состояния каналов.

Наиболее хитрая часть этого алгоритма – надежное распространение пакетов состояний каналов (далее СК-пакетов). Как только СК-пакет получен и включен в работу, маршрутизатор начинает использовать его при определении маршрута. При неудачном распространении СК-пакетов разные маршрутизаторы могут получить разное представление о топологии транспортной среды, что может привести к возникновению циклов, появлению недостижимых машин и другим проблемам.

Рассмотрим сначала базовый алгоритм, в котором СК-пакеты распространяются методом лавины, т.е. СК-пакет рассылается всем соседям, которые, в свою очередь, пересылают его своим соседям и т.д. Однако чтобы не потерять контроль и не вызвать неограниченного дублирования 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 соседа, от которого он его получил. Если маршрутизатор уже встречал такой пакет, то этот пакет сбрасывается и никуда не дублируется.

У этого алгоритма есть несколько проблем, но все они разрешимые.

Первая проблема – размер поля последовательных номеров пакетов. Если это поле будет недостаточно велико, то его переполнение приведет к повтору номеров, а следовательно, к некорректной работе всего алгоритма. Решением здесь является использование достаточно большого поля, например 32-разрядного. В этом случае если обмен СК-пакетами происходит раз в секунду, потребуется 137 лет, чтобы возникло переполнение.

Вторая проблема – если маршрутизатор «упал» по какой либо причине и потерял уже использованные последовательные номера, то неясно, как их, восстановить.

Третья проблема – если в результате передачи возникнет ошибка в одном бите, например вместо пакета с номером 4 получим пакет с номером 65540, то вес пакеты с 5-го номера по 65540-й будут сбрасываться как устаревшие, поскольку текущий номер – 65540.

Для решения этих проблем используется поле «Возраст» СК-пакета, в котором устанавливается некоторое значение, уменьшающееся па единицу при каждом скачке, и, когда это значение достигнет нуля, пакет сбрасывается.

В целях сокращения числа рассылаемых СК-пакетов их рассылают не сразу. Сначала полученный СК-пакет помещают в специальную область задержки, где он находится некоторое время. Если за это время придет другой пакет от того же источника, то эти пакеты сравниваются, и если они одинаковые, то вновь пришедший пакет сбрасывается, если же они различаются, то последний пришедший пикет дублируется и отправляется другим маршрутизаторам, а первый пакет сбрасывается. Все СК-пакеты передаются с уведомлением.

Структура данных, используемая маршрутизатором B из примера, приведенного на рис. 2.10, а, показана на рис. 2.11 11. Здесь каждая строка таблицы соответствует последнему поступившему, но не до конца обработанному СК-пакету. При этом для каждого канала маршрутизатора B указано с помощью флага, был ли соответствующий СК-пакет отправлен и подтвержден. Флаг отправления означает, что соответствующий СК-пакет должен быть послан по указанному каналу. Флаг уведомления указывает, что по соответствующей линии должно прийти подтверждение.

			Флаги отправления			Флаги уведомления			
Источник	Номер	Возраст	*A*	*C*	*F*	*A*	*C*	*F*	Данные
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Рис. 2.11. Буфер пакетов для маршрутизатора *B* из примера, приведенного на рис. 2.10, *a*

На рис. 2.11 СК-пакет от маршрутизатора A прибыл и он должен быть переслан в C и в F и подтвержден для A. Аналогичная ситуация с СК-пакетом, поступившим от F. Существенно отличается ситуация

с маршрутизатором E, от которого поступило два СК-пакета: один по маршруту EAB, а другой по маршруту EFB. Следовательно, СК-пакет должен быть послан только в C, а вот уведомления должны быть посланы и в A, и в F. Наличие дубликатов СК-пакетов легко распознать по состоянию флагов отправки.

Вычисление нового пути.

Когда маршрутизатор получил полный комплекс СК-пакетов, он может построить топологию транспортной среды и, например, локально запустить алгоритм Дейкстры для вычисления наикратчайшего пути.

В системе, где есть n маршрутизаторов, имеющих по k линий, у каждого из этих маршрутизаторов должно быть достаточно памяти для хранения необходимой информации о сети. При больших значениях n этот объем памяти может стать существенным. Кроме того, в сетях, где число маршрутизатором достигает десятков или сотен тысяч, проблемы, вызванные сбоем одного из них, могут оказаться весьма серьезными. Например, в случае если из-за сбоя маршрутизатор послал неправильный СК-пакет или неправильно оценил состояние канала, в дальнейшем это может создать большие проблемы.

Маршрутизация по состоянию каналов широко используется в реальных сетях. Например, в протоколе OSPF, который будет подробно рассматриваться далее, и в протоколе IS-IS, который был изначально предложен фирмой DEC, а позже адаптирован ISO. Протокол IS-IS широко используется в Интернете. Поскольку OSPF появился позже IS-IS, он вобрал в себя все усовершенствования, сделанные для IS-IS, а основное различие между ними заключается в том, что IS-IS может работать с разными протоколами сетевого уровня, что делает его весьма привлекательным при маршрутизации между различными сетями, чего не может делать протокол OSPF.

24 Маршрутизация в Интернет: структура Интернета, понятие автономной системы, протокол внешней маршрутизации BGP.

Фактически, Интернет состоит из множества локальных и глобальных сетей, принадлежащих различным компаниям и предприятиям, работающих по самым разнообразным протоколам, связанных между собой различными линиями связи, физически передающих данные по телефонным проводам, оптоволокну, через спутники и радиомодемы. Структура Интернет напоминает паутину, в узлах которой находятся компьютеры, связанные между собой линиями связи. Узлы Интернет, связанные высокоскоростными линиями связи, составляют базис Интернет. Оцифрованные данные пересылаются через маршрутизаторы, которые соединяют сети с помощью сложных алгоритмов, выбирая маршруты для информационных потоков.

Каждый компьютер в Интернет имеет свой уникальный адрес – **IP-адрес**. Этот номер может быть постоянно закреплен за компьютером или же присваиваться динамически – в тот момент, когда пользователь соединился с провайдером, но в любой момент времени в Интернет не существует двух компьютеров с одинаковыми IP-адресами.

Доменное имя – это уникальное имя, которое данный поставщик услуг избрал себе для идентификации. Для преобразования имени в адрес копьютер посылает запрос серверу DNS, начиная с правой части доменного имени и двигаясь влево.

Данные в Интернет пересылаются не целыми файлами, а небольшими блоками, которые называются пакетами. Каждый пакет содержит в себе адреса компьютеров отправителя и получателя, передаваемые данные и порядковый номер пакета в общем потоке данных. Благодаря тому, что каждый пакет содержит все необходимые данные, он может доставляться независимо от других, и довольно часто случается так, что пакеты добираются до места назначения разными путями. А компьютер-получатель затем выбирает из пакетов данные и собирает из них тот файл, который был заказан.

Порт – это число, которое добавляется к адресу компьютера, которое указывает на программу, для которой данные предназначены.

В Интернет используются не просто доменные имена, а **универсальные указатели ресурсов** URL (Universal Resource Locator).

URL включает в себя:

- метод доступа к ресурсу, т.е. протокол доступа (http, gopher, WAIS, ftp, file, telnet и др.);
- сетевой адрес ресурса (имя хост-машины и домена);
- полный путь к файлу на сервере.

Сервер в сети Интернет – это компьютер, обеспечивающий обслуживание пользователей сети: разделяемый доступ к дискам, файлам, принтеру, системе электронной почты. Обычно сервер - это совокупность аппаратного и программного обеспечения.

Сайт – обобщенное название совокупности документов в Интернет, связанных между собой ссылками.

Шлюз (gateway) – это компьютер или система компьютеров со специальным программным обеспечением, позволяющая связываться двум сетям с разными протоколами.

Домашняя страница – это персональная Web-страница конкретного пользователя или организации.

Автономная система (AS) в интернете – это система IP-сетей и маршрутизаторов, управляемых одним или несколькими операторами, имеющими единую политику маршрутизации с Интернетом.

Протокол BGP используется для передачи информации о внутренних маршрутах между автономными системами. Протокол BGP может быть использован для определения различных типов маршрутов:

- **Inter-autonomous system routing** – маршруты, которые соединяют данную автономную систему с одной или несколькими другими автономными системами.
- **Intra-autonomous system routing** – протокол может быть использован для определения маршрута внутри автономной системы, в том случае, когда несколько маршрутизаторов участвуют в процессе определения маршрута BGP.
- **Pass-through autonomous system** – протокол может быть использован для определения маршрутов, которые проходят через автономную систему, которая не участвует в процессе BGP.

Для обеспечения информационного обмена маршрутизаторы BGP используют сообщения стандартной формы. Для передачи этих сообщений в протоколе BGP предусматривается использование транспортного протокола TCP. Сообщения BGP передаются в следующих случаях:

- Начало сеанса (**Open**).
- Для периодической проверки состояния соседа (**Keep Alive**).
- При изменении содержания таблицы маршрутов автономной системы (**Update**).
- При возникновении аварийной ситуации (**Notification**).

Формат сообщения BGP.

Каждое сообщение BGP состоит из заголовка и последующих специфических полей:

MARKER	
MARKER	
MARKER	
MARKER	
LENGTH	TYPE

В поле **LENGTH** помещается размер сообщения (вместе с заголовком), выраженный в байтах.

В поле **TYPE** помещается код сообщения в соответствии со следующей таблицей:

TYPE	Сообщение
1	OPEN
2	UPDATE
3	NOTIFICATION
4	KEEPALIVE

В поле **маркера** может быть помещена информация, которая необходима для выполнения операции аутентификации абонента. Если установление подлинности абонента не требуется, маркер формируется значениями – все «1».

OPEN – первое сообщение, которое должно быть передано маршрутизатором BGP после установления соединения TCP.

UPDATE используется для представления маршрута соседнему маршрутизатору BGP. Это сообщение одновременно может быть использовано для уничтожения маршрутов, которые перестали существовать.

25 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).

Любая информация может передаваться с помощью электромагнитных импульсов (сигналов). В зависимости от среды передачи и организации СПД используются либо аналоговые, либо цифровые сигналы.

Любой сигнал можно рассматривать либо как функцию времени, либо как функцию частоты (как композицию составляющих сигналов – гармоник). Важной характеристикой сигнала является **ширина его полосы**, которая покрывает весь спектр частот гармоник, составляющих сигнал. Чем шире эта полоса, тем больше информационная емкость сигнала. При создании любой СПД приходится искать компромисс между четырьмя основными факторами: шириной полосы сигнала, скоростью передачи сигналов, уровнем шумов и искажений сигнала, допустимым уровнем ошибок при передаче.

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

Характеристику канала, определяющую спектр частот, которые физическая среда, из которой сделана линия связи, образующая канал, пропускает без существенного понижения мощности сигнала, называют **полосой пропускания**.

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.

Сигналы могут иметь непрерывную или дискретную форму. В первом случае говорят об **аналоговом сигнале**, во втором – о **цифровом**.

Большое значение имеет количество уровней, которое может иметь сигнал. Чем больше число уровней сигнала, тем больше информации можно передать за один переход с уровня на уровень.

Процесс передачи также может иметь аналоговую или цифровую формы. **Аналоговая передача** предполагает непрерывное изменение параметров передачи. **Цифровая** — резкое, дискретное изменение параметров передаваемого сигнала или импульса.

Максимальную скорость, с которой канал способен передавать данные, называют **пропускной способностью** канала или **битовой скоростью**.

Теорема Найквиста (взаимосвязь между пропускной способности канала и шириной его полосы пропускания): $V_{max\_data\_rate} = 2Hlog_2M$ бит/сек, где $V_{max\_data\_rate}$ — максимальная скорость передачи, H — ширина полосы пропускания канала, выраженная в Гц, M – количество уровней сигнала.

Шум в канале – отношение мощности полезного сигнала к мощности шума: $\frac{S}{N}$. Измеряется в децибелах: $10log_{10}(\frac{S}{N})$ dB.

Теорема Шеннона: максимальная скорость передачи данных по каналу с шумом равна: $Hlog_2(1+\frac{S}{N})$ бит/сек, где $\frac{S}{N}$ – соотношение сигнал-шум в канале.

Сигнальная скорость или **скорость модуляции** – скорость изменения значения сигнала. Измеряется в бодах. Если скорость изменения значения сигнала b бод, 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.

Среды передачи.

Назначение физического уровня – передать данные в виде потока бит от одной машины к другой. Для передачи можно использовать разные физические среды. Каждая из сред имеет свои уникальные характеристики, такие как:

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

Кроме них важно учитывать также такие характеристики как, например, достоверность передачи, затухание, помехоустойчивость и т.д.

Магнитные носители.

Магнитная лента или магнитный диск в сочетании с обычным транспортным средством могут быть прекрасной физической средой передачи данных. Это так особенно там, где высокая пропускная способность и низкая стоимость передачи в расчете на один бит – ключевые факторы.

Витая пара.

Для многих приложений нужен оперативный обмен информацией. Самой старой и все еще используемой средой передачи является витая пара. Витая пара состоит из двух медных изолированных проводов, один из которых обвит вокруг другого. Вьющийся провод предназначен для устранения взаимного влияния между соседними витыми парами. Витая пара широко используется в телефонии. Особенно между абонентами и местной АТС, линии из витой пары могут иметь протяженность до нескольких километров без промежуточного усиления. Витые пары объединяются в многопарные кабели. Витая пара может быть использована для передачи как цифрового, так и аналогового сигналов. Пропускная способность зависит от толщины линий и расстояния. Скорость в несколько мегабит в секунду вполне достижима с помощью соответствующих методов передачи. На коротких расстояниях была достигнута скорость до 1 Г бит/сек. На больших расстояниях скорость передачи не превышает 4 М бит/сек.

Кабели категории 3 содержат по четыре витые пары с невысокой плотностью навивки, и имеют полосу пропускания до 16 МГц. Кабель категории 5 имеет тоже четыре пары, но с более плотной навивкой, что позволяет достичь более высоких скоростей, и имеют полосу пропускания 100 МГц.

Коаксиальные кабели.

Подобно витой паре у коаксиального кабеля есть два проводника. Центральный проводник представляет собой медный проводник, окруженный изолятором. Эта конструкция помещается внутри второго цилиндрического проводника, который обычно представляет собой сплетенную плотную металлическую сетку. Все это закрывается плотным защитным слоем пластика. Обычно толщина коаксиала от 1 до 2.5 см. У коаксиала полоса пропускания шире и характеристики по затуханию сигнала лучше, чем у витой пары. Поэтому эти кабели применяют на больших расстояниях и по ним могут передавать одновременно несколько потоков данных от разных компьютеров. Коаксиальные кабели используют для передачи как аналоговых, так и цифровых сигналов. Основными ограничениями скорости и расстояния передачи 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. 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 при высокой помехозащищенности.

Второй вид коаксиальных кабелей используют в телевидении и называют высокочастотным кабелем.

В двух кабельных системах прокладывается сразу два кабеля: один кабель используется для входящего потока, а второй для исходящего.

Оптоволокно.

Для использования оптической связи нужен источник света, светопроводящая среда, детектор, преобразующий световой поток в электрический. На одном передающем конце волоконнооптической линии находится источник света, световой импульс от этого источника проходит по тонкому светопроводящему волокну и попадает на детектор, который 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. Количество рассеиваемой энергии зависело от угла падения светового луча на стенки шнура.

При углах больше некоторого критического угла, называемого углом полного внутреннего отражения вся энергия луча отражается обратно внутрь.

Если сделать силиконовый шнур толщиной близкой к длине волны источника света, то этот шнур будет работать, как провод для тока, без потерь на внутреннее отражение. По такому одномодовому шнуру можно передавать со скоростью в несколько Гбит/сек на сотню километров без промежуточного усиления.

Поскольку можно испускать несколько лучей 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 мы получаем многомодовый шнур.

Оптоволокно делают из стеклоподобного материала, которое в свою очередь делают из песка и других широко распространенных материалов.

Затухание оптического сигнала в стекле зависит от длины волны источника света. Затухание измеряется в dB по формуле $10log_{10}\frac{Tp}{Rp}$, где Tp — мощность передаваемого сигнала, Rp — мощность полученного сигнала.

Для передачи используются три полосы с длинами волн 0,85, 1,30 и 1,55 мкм. Две последние обладают тем замечательным свойством, что их затухание составляет менее 5% на километр. Длина волны в 0,85 мкм имеет большее затухание, но хороша тем, что лучше соответствует возможностям лазерных источников света. У всех трех полос ширина полосы пропускания от 25’000 до 30’000 ГГц.

Другую проблему при использовании оптоволокна представляет дисперсия: исходный световой импульс по мере распространения теряет начальную форму и размеры. 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**.

В заключение будет полезно сравнить 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.

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.

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

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 отображается в несколько единичных сигналов. В простейшем случае это соответствие имеет однозначный характер: один бит — один сигнал.

Если все единичные сигналы имеют одинаковую полярность, то говорят, что **сигнал униполярный**.

Скорость передачи данных — это количество бит в секунду, которые передают с помощью сигналов. Эту скорость также называют **битовой скоростью**. **Продолжительность** или **длина бита** — это интервал времени, который нужно передатчику, чтобы испустить надлежащий единичный сигнал. При скорости передачи данных R бит/сек, длина бита равна $\frac{1}{R}$.

Прежде всего, приемник должен быть строго настроен на длину бита. Он должен уметь распознавать начало и конец передачи каждого бита. Уметь распознавать уровень сигнала: низкий или высокий. Например, эти задачи решаются измерением уровня сигнала в середине длины бита и сравнением результата измерения с пороговым значением. Из-за шума на линии при этом могут возникать ошибки.

Есть три важных фактора влияющие на правильность передачи: уровень шума, скорость передачи данных и ширина полосы пропускания канала. Есть еще один фактор, влияющий на передачу данных: это способ представления (кодировки) данных на физическом уровне.

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

- **Ширина спектра сигнала:** чем меньше высокочастотных составляющих в сигнале, тем уже ширина полосы пропускания может быть при передаче. Важным также является отсутствие постоянной составляющей (приводит к наличию постоянного тока между приемником и передатчиком, что крайне нежелательно). Чем шире спектр, тем сильнее искажения.

- **Синхронизация между приемником и передатчиком:** приемник должен точно определять начало и конец битового интервала. На небольших расстояниях можно использовать дополнительную линию синхронизации – тактирующая схема (часы) выдает строго через определенные промежутки синхроимпульсы. Другое решение этой проблемы состоит в создании самосинхронизирующихся кодов.

- **Обнаружение ошибок.**

- **Чувствительность к шуму:** за счет надлежащих ухищрений в схеме кодировки данных можно добиться очень высокой производительности при передаче даже при наличии очень высокого уровня шума.

- **Стоимость и скорость.**

Потенциальный код NRZ:
0 – высокий потенциал.
1 – низкий потенциал.

Биполярный код NRZI:
0 – нет перепада уровня сигнала в начале битного интервала.
1 – перепад уровня сигнала в начале интервала.

Биполярный код AMI:
0 – отсутствие сигнала.
1 – положительный или отрицательный потенциал, обратный по отношению к потенциалу в предыдущий период.

Манчестерский код:
0 – переход с высокого на низкий потенциал в середине интервала.
1 – переход с низкого на высокий потенциал в середине интервала.

Потенциальный код 2B1Q:
Использует 4 уровня сигналов, значение уровня определяется значением пары битов данных.

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

Потенциальный NRZ код.
Основным недостатком этого кода является отсутствие синхронизации. На длинных последовательностях нулей или единиц потенциал на линии не меняется, и может произойти рассинхронизация между приемником и передатчиком, что приведет к ошибкам. Если исключить возможность появления длинных последовательностей 0 или 1 (например, использовать специальные устройства скремблеры), то этот метод может быть весьма эффективен.

Модификацией NRZ кода и хорошим примером дифференциального кодирования является **NRZ-I код**. Идея дифференциальных кодов состоит в том, чтобы кодировать не абсолютное значение текущего бита, а разницу значений между предыдущим битом и текущим. В случае NRZ-I кода 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 NRZ-I кода по отношению NRZ коду является большая устойчивость к шуму.

Биполярный код AMI.
У этого метода есть несколько существенных преимуществ по сравнению с NRZ кодами. Во-первых, в случае длительной последовательности 1 рассинхронизации не происходит. Каждая единица сопровождается изменение потенциала устойчиво распознаваемом 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, четко определенное правило чередования уровней позволяет обнаруживать единичные ошибки.

Биполярные импульсные 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. В дифференциальном Манчестерском коде сигнал может менять свой уровень дважды в течении 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.

Преимущества биполярных импульсных методов:

- самосинхронизация,
- отсутствие постоянной составляющей,
- отсутствие 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 NRZ и AMI кодов, а спектр сигнала в два раза уже. Поэтому с помощью 2B1Q кода можно по одной и той же линии предавать 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.

В общем случае соотношение между битовой и сигнальной скоростью определяется формулой $D = \frac{R}{b}$, где D – сигнальная скорость, R – битовая скорость в бит/сек, b – количество бит на единичный сигнал.

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

Преобразование аналоговых данных в цифровой сигнал можно представить как преобразование аналоговых данных в цифровую форму. Этот процесс называют **оцифровкой данных**. Выполнив его, мы можем передать цифровые данные цифровым или аналоговым сигналом. 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).

Импульсно-кодовая модуляция (ИКМ) основана на следствии из теоремы Найквиста, которое утверждает, что если изменять 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.

Другой альтернативой ИКМ является **метод дельта модуляции**. На исходную непрерывную функцию, представляющую аналоговый сигнал, накладывают ступенчатую функцию. Значения этой ступенчатой функции меняются на δ на каждом шаге квантования по времени T_s. Замена исходной функции на эту дискретную, ступенчатую функцию интересно тем, что поведение последней носит двоичный характер. На каждом шаге значение ступенчатой функции либо увеличивается на δ, будем представлять этот случай 1, либо сокращается на δ – случай 0.

Процесс передачи в случае 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, придается 1, в противном случае 0. Таким образом, ступенчатая функция всегда меняет свое значение.

У метода Дельта модуляции есть два параметра: величина шага δ и частота замеров или шаг квантования. Выбор шага δ – это баланс между ошибкой квантования и ошибкой перегрузки по крутизне.

Когда исходный сигнал изменяется достаточно медленно, то возникает только ошибка квантования, чем больше δ, тем больше эта ошибка. Если же сигнал изменяется резко, то рост ступенчатой функции может отставать. Это вид ошибки растет с уменьшением δ.

Положение можно улучшить, увеличив частоту замеров, но это увеличит битовую скорость на линии.

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

Примером передачи данных в цифровой форме с помощью аналоговых сигналов является использование телефонных сетей для передачи цифровых данных. Телефонные сети были созданы для передачи и коммутации аналоговых сигналов в голосовом диапазоне частот от 300 до 3400 Гц.

Модем (МОдулятор–ДЕМодулятор) – прибор, который преобразует цифровой сигнал в аналоговый и наоборот в надлежащем диапазоне частот.

Аналоговая модуляция заключается в преобразовании одного или нескольких параметров из трех основных параметров несущего сигнала: амплитуды, частоты и фазы.

Есть три основных метода модуляции для преобразования цифровых данных в аналоговую форму:

- амплитудная модуляция,
- частотная модуляция,
- фазовая модуляция.

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 $S(t) = \begin{cases} Acos(2\pi f_c t) & \text{двоичная 1} \\ 0 & \text{двоичный 0} \end{cases}$, где $Acos(2\pi f_c t)$ – несущий сигнал с амплитудой A. Метод амплитудной модуляции не очень эффективен по сравнению с другими методами, т.к. он очень чувствителен к шумам.

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: $S(t) = \begin{cases} Acos(2\pi f_1 t) & \text{для 1} \\ Acos(2\pi f_2 t) & \text{для 0} \end{cases}$, где $f_c = f_1 - \Delta = f_2 + \Delta$, где Δ — сдвиг по частоте.

Частотную модуляцию чаще всего применяют в радиомодемах на частотах от 3 МГц до 30 МГц, а также в высокочастотных кабелях локальных сетей.

Фазовая модуляция состоит в представлении цифровых данных сдвигом фазы несущего сигнала. Для дифференциальной фазовой модуляции получаем: $S(t) = \begin{cases} Acos(2\pi f_c t + \pi) & \text{для 1} \\ Acos(2\pi f_c t) & \text{для 0} \end{cases}$.

Эффективность использования полосы пропускания можно существенно повысить, если единичный сигнал будет кодировать несколько бит. В общем случае скорость модуляции $D = \frac{R}{b} = \frac{R}{log_2 L}$, где D – скорость модуляции (сигнальная скорость), R – битовая скорость (скорость передачи данных), L – число разных уровней единичных сигналов, b – число бит на единичный сигнал.

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

Потребность возникает при использовании радио каналов. **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.

Три способа модуляции: амплитудная модуляция, частотная и фазовая.

При амплитудной модуляции форма результирующего сигнала определяется формулой: $S(t) = [1 + n_a x(t)]cos(2\pi f_c t)$, где f_c – частота несущей, n_a – индекс модуляции, который определяют как отношение амплитуды исходного сигнала к амплитуде несущего сигнала.

В наших обозначениях: $m(t) = 1 + n_a x(t)$.

Форма результирующего сигнала при частотной модуляции определяется следующим выражением: $S(t) = A_c cos(2\pi f_c t + n_f m(t))$, где n_f – индекс частотной модуляции.

Сигнал, получаемый фазовой модуляцией, определяет соотношение: $S(t) = A_c cos(2\pi f_c t + n_p m(t))$, где n_p – индекс частотной модуляции.

Хотя все эти три вида модуляции порождают сигнал S(t), спектр которого симметричен относительно f_c, но в случае амплитудной модуляции он проще по составу. В случае частотной и фазовой модуляций требуется, в общем случае, более широкая полоса пропускания.

Широко распространенным случаем аналоговой модуляции является метод квадратичной амплитудной модуляции QAM. Именно этот метод используется в асимметричных цифровых линиях – ADSL. Метод QAM – это комбинация амплитудной и фазовой модуляций. Идея этого метода состоит в том, что можно по одной и той же линии послать одновременно два разных сигнала с одинаковой несущей частотой, но сдвинутых по фазе друг относительно друга на $90°$. Каждый сигнал генерируется методом амплитудной модуляции.

30 Физические среды передачи данных. Беспроводная связь (электромагнитный спектр, радиопередача, микроволновая передача, видимое излучение). Протоколы MACA.

Назначение физического уровня – передавать данные в виде потока битов от одной машины к другой. При этом для передачи данных можно использовать различные физические среды, каждую из которых характеризуют следующие параметры:

- ширина полосы пропускания,
- пропускная способность,
- задержка сигнала,
- стоимость,
- сложность прокладки,
- сложность обслуживания.

Кроме перечисленных, физическую среду характеризуют и другие параметры, например:

- достоверность передачи,
- затухание,
- помехоустойчивость.

Магнитная лента или **магнитный диск** в сочетании с обычным транспортным средством (автомашиной, железной дорогой и т.п.) могут быть прекрасной физической средой для передачи данных. Это так, особенно в случае, если высокая пропускная способность и низкая стоимость передачи в расчете на один бит являются ключевыми факторами.

Витая пара – два медных изолированных провода, один из которых обвит вокруг другого. Вьющийся провод предназначен для устранения взаимного влияния между соседними витыми парами.

Витая пара широко используется в телефонии. Протяженность линии из витой пары может составлять несколько километров без промежуточного усиления. В России в городских условиях средняя длина абонентской линии около 3,5 км.

Витая пара может быть использована для передачи как цифровых, так и аналоговых сигналов. Ее пропускная способность зависит от толщины используемых проводов и длины линии. На коротких расстояниях (до сотни метров) может достигаться скорость до 1 Тбит/с, на больших расстояниях (несколько километров) скорость передачи не превышает 4 Мбит/с.

Витые пары объединяются в многопарные кабели. Кабель категории 3 содержит четыре витые пары с невысокой плотностью навивки и имеет ширину полосы пропускания до 16 МГц. Наиболее часто используются кабели категории 5, который тоже состоит из четырех пар, но имеет более плотную навивку, что позволяет достигать более высоких скоростей передачи и ширину полосы пропускания 100 МГц.

Коаксиальные кабели.

У коаксиального кабеля есть два проводника. Центральный проводник представляет собой толстый медный провод, окруженный изолятором. Эта конструкция помещается внутри второго цилиндрического проводника, который обычно представляет собой плетеную плотную металлическую сетку. Все это закрывается плотным защитным слоем пластика. Толщина коаксиального кабеля составляет от 1 до 2,5 см. У такого кабеля шире полоса пропускания и характеристики по затуханию сигнала лучше, чем у витой пары. Коаксиальные кабели работают на частотах от 1 до 500 МГц, поэтому их применяют на больших расстояниях и по ним могут передаваться одновременно несколько потоков данных от разных компьютеров.

Коаксиальные кабели используют для передачи как аналоговых, так и цифровых сигналов. Основными ограничителями скорости и расстояния при передаче без усиления в этих кабелях являются затухание сигнала, тепловой и интермодуляционный шумы. Когда всю полосу пропускания кабеля разбивают на более узкие полосы и каждую такую полосу используют как отдельный канал, на границах таких каналов возникает интермодуляционный шум.

Оптоволокно.

Для создания оптической связи требуется источник света с постоянной длиной волны, светопроводящая среда и детектор, преобразующий световой поток в электрический. На одном конце оптоволоконной линии имеется передатчик – источник света, световой импульс от которого проходит по светопроводящему волокну и попадает на детектор, расположенный на другом конце этой линии и преобразующий этот импульс в электрический.

Одна из основных проблем при создании оптоволоконных систем состояла в том, чтобы не дать световому пучку рассеяться через боковую поверхность силиконового шнура. Количество рассеиваемой энергии зависит от утла падения светового луча на стенки шнура. При углах больше некоторого критического угла, называемого углом полного внутреннего отражения, вся энергия луча отражается обратно внутрь.

Силиконовый шнур, имеющий толщину, близкую к длине волны источника света, работает без потерь на внутреннее отражение. По такому **одномодовому** шнуру можно передавать данные со скоростью несколько гигабит в секунду на сотню километров без промежуточного усиления.

Поскольку можно испускать несколько лучей разной длины волны так, чтобы они попадали на границы шнура под углом больше, чем угол полного внутреннего отражения, следовательно, по одному шнуру можно пускать несколько лучей. При этом каждый луч, как говорят, имеет свою моду. Так получается **многомодовый** шнур.

Оптоволокно изготавливают из стеклоподобного материала. Затухание оптического сигнала в стекле зависит от длины волны источника света. На практике для передачи сигнала используются три полосы для волн длиной 0,85, 1,30 и 1,55 мкм. Волны длиной 1,30 и 1,55 мкм обладают тем замечательным свойством, что их затухание составляет менее 5% на километр. Волны длиной 0,85 мкм имеют большее затухание, но они лучше соответствуют возможностям лазерных источников света. Ширина полосы пропускания во всех трех случаях составляет от 25000 до 30000 ГГц.

Другой проблемой при использовании оптоволокна является дисперсия – потеря по мере распространения исходными световыми импульсами начальных форм и размеров. При этом искажения также

зависят от длины волны. Одно из возможных решений этой проблемы – увеличение расстояния между соседними сигналами. Однако это сокращает скорость передачи. К счастью, исследования показали, что при придании сигналу некоторой специальной формы дисперсионные эффекты почти исчезают и сигнал можно передавать на тысячи километров. Сигналы, имеющие такую специальную форму, называются **силитонами**.

Оптоволоконный кабель имеет сердечник, состоящий из сверхпрозрачного оптоволокна и изоляционного покрытия. В одномодовом кабеле толщина сердечника составляет 8 – 10 мкм, а в многомодовом 50 – 100 мкм. Сердечник имеет оптическое покрытие из стекловолокна с низким коэффициентом рефракции, сокращающего потери света через его границы, и защитное покрытие из пластика. Соединяют его с помощью специальных коннекторов, механически прижимая один край к другому, либо сваркой. При этом в точке соединения теряется от 5 до 20% мощности сигнала.

Для передачи используются два вида источников света: светодиод (LED) и полупроводниковый лазер, которые обладают разными свойствами. С помощью специальных интерферометров эти источники света можно настроить на требуемую длину волны. На принимающем конце устанавливается фотодиод, время срабатывания которого 1 не, что ограничивает максимальную скорость передачи значением 1 Гбит/с.

Активное подключение содержит промежуточный усилитель электрического сигнала. Фотодиод преобразует оптический сигнал в электрический, который усиливается и передается компьютеру либо транслируется дальше с помощью лазера или светодиода.

1. Ширина полосы пропускания у оптоволокна несравнимо больше, чем у медного кабеля, что позволяет достигать скоростей передачи в сотни гигабит в секунду на расстояниях в десятки километров

2. Оптоволокно компактнее и имеет меньшую массу.

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

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

5. Чем меньше используется репитеров, тем дешевле система передачи и меньше источников ошибок. С этой позиции оптоволоконные системы достигли большего совершенства. Среднее расстояние между репитерами у них в разы больше, чем у коаксиального кабеля и витой пары.

Беспроводная связь востребована для мобильных вычислительных средств и там, где прокладка любого кабеля затруднительна либо невозможна (горы, старые здания), либо если требуется быстрое создание коммуникации.

Электромагнитный спектр.

В вакууме электромагнитная волна распространяется со скоростью света ($= 3 * 10^8$ м/с). В медном проводнике эта скорость составляет 2/3 от скорости в вакууме. Обозначим f – частоту, λ – длину волны. Фундаментальное соотношение между f, c и λ имеет вид f * λ = c.

При определенных условиях волны распространяются 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.

Для передачи информации из всего спектра частот используют только следующие диапазоны: радиодиапазон, микроволновый, инфракрасный, видимый и частично ультрафиолетовый. Диапазоны рентгеновского излучения, гамма-излучения и большая часть ультрафиолетового, включающие в себя большие частоты, а следовательно, предпочтительные для передачи, требуют, однако, использования сложной аппаратуры для генерации и модуляции, сигналы в них плохо преодолевают препятствия и, что самое главное, они опасны для живой материи.

Количество данных, передаваемых электромагнитной волной, определяется ее шириной, т.е. спектром частот гармоник, составляющих эту волну. При определенных условиях на низких частотах можно закодировать несколько бит данных на 1 Гц, но на высоких частотах это число можно довести до 40 бит. Следовательно, по кабелю с полосой пропускания 500 МГц можно передавать данные со скоростью несколько гигабит в секунду. Учитывая широкую полосу пропускания оптоволоконного кабеля, становится ясно, почему оптоволокно столь привлекательно для сетей ЭВМ.

Задав некоторую полосу длин волн, получим полосу частот, откуда затем найдем скорость передачи для этой полосы частот. Чем шире полоса частот, тем выше битовая скорость, что следует из формулы, связывающей ширину полосы пропускания и битовую скорость передачи.

На практике чаще всего используются узкочастотные полосы передачи.

При широкочастотной передаче, используемой в основном военными и спецслужбами, частота несущей волны изменяется по определенному закону в диапазоне полосы. Перехватить такую передачу можно только в случае, если известен закон изменения частоты несущей.

Радиопередача.

Радиоволны распространяются на большие расстояния, легко преодолевая преграды. Поскольку радиоволны распространяются во всех направлениях, то принимающая и передающая антенны не требуют дополнительной настройки и регулирования взаимного расположения.

Свойства радиоволн зависят от их частоты. На низких частотах, т.е. длинных волнах, они прекрасно преодолевают препятствия, но мощность сигнала падает пропорционально $1/r^3$, где r – расстояние цо источника. На высоких частотах радиоволны распространяются но прямой, но хуже преодолевают препятствия.

На любых частотах радиоволны чувствительны к помехам от электрических устройств. В силу перечисленных причин лицензирование, т.е. право использования частот в радиодиапазоне, находится под жестким контролем государства.

Длинные и средние волны могут огибать поверхность Земли и распространяться на большие расстояния. Короткие полны хотя и поглощаются земной поверхностью, но за счет отражения от ионосферы также могут распространяться на большие расстояния.

Микроволновая передача.

Частоты свыше 10 МГц представляют собой 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 при этом должны быть тщательно ориентированы в пространстве по отношению друг к другу. Такая направленность позволяет, построив цепочку ретрансляторов, передавать сигнал на большие расстояния.

Микроволны не проходят сквозь здания так же хорошо, как низкочастотные волны. Кроме того, из-за рефракции в нижних слоях атмосферы они могут отклоняться от прямого направления. При этом увеличивается задержка сигнала и нарушается передача. Помимо того передача на этих частотах зависит и от погоды: при повышении влажности (дождь, туман и т.п.) ширина полосы пропускания резко сужается, растет шум, сигнал рассеивается.

Для увеличения пропускной способности увеличивают частоту, но начиная с частоты 8 ГГц, волны поглощаются водой и, в частности, дождем. Единственный выход из положения в этом случае – изменить маршрут передачи, и обойти область дождя.

Одно из главных достоинств микроволнового диапазона – не требуется прокладка никакой линии. Достаточно получить права на небольшие площадки земли (в сотню квадратных метров) для установки башен-ретрансляторов через каждые 50 км.

Несколько частотных полос в диапазоне 2’400 – 2’484 ГГц, например, инфракрасные волны, можно использовать свободно без специального разрешения. Однако в разных странах могут использоваться и другие дополнительные диапазоны, например, в США помимо указанного диапазона используются диапазоны 902 – 928 МГц и 5 725 – 5 850 ГГц.

Инфракрасные и миллиметровые волны.

Инфракрасное излучение и излучение в миллиметровом диапазоне используются на небольших расстояниях в блоках дистанционного управления. Основной недостаток таких излучений – они не проходят через преграды. Например, для инфракрасного излучения лист бумаги – непреодолимое препятствие. Однако этот недостаток одновременно является и преимуществом, поскольку такое излучение в одной комнате не интерферирует с подобным излучением в другой комнате. На использование этих частот не надо также получать разрешение, т. е. это прекрасный канал для передачи данных внутри помещений на небольших расстояниях.

Видимое излучение. Видимый диапазон также используется для передачи сигналов. Обычно источником света в этом случае является лазер. Монохромное когерентное излучение легко фокусируется. Однако смог, загрязнение атмосферы, дождь или туман портят дело. Передачу такого излучения способны нарушить даже конвекционные потоки воздуха на крыше, возникающие в жаркий день, которые вызывают дрожание луча вокруг приемника, ухудшая качество передачи.

Протоколы MACA и MACAW. Одним из первых протоколов, разработанных для беспроводных локальных сетей, является MACA (Multiple Access with Collision Avoidance – множественный доступ с предотвращением столкновений) (Karn, 1990). Идея, лежащая в основе этого протокола, заключается в том, что отправитель заставляет получателя передать короткий кадр, чтобы окружающие станции могли услышать эту передачу и воздержаться от действий на время, требуемое для приема большого информационного кадра. Протокол MACA проиллюстрирован на рисунке.

Рассмотрим ситуацию, в которой станция A передает станции B. Станция A начинает с того, что посылает станции B кадр RTS (Request To Send – запрос на передачу), как показано на рисунке а. Этот короткий кадр (30 байт) содержит длину кадра данных, который последует за ним. Затем станция B отвечает кадром CTS (Clear To Send – разрешение передачи), как показано на рисунке б. Кадр CTS также содержит длину информационного кадра (скопированную из кадра RTS). Приняв кадр CTS, станция A начинает передачу.

Теперь посмотрим, как реагируют станции, которые слышат передачу одного из этих кадров. Любая станция, которая слышит кадр RTS, находится близко к станции A и поэтому должна хранить молчание, пока кадр CTS не будет принят станцией A. Станции, слышащие кадр CTS, находятся вблизи от станции B, следовательно, должны воздержаться от передачи, пока станция B не получит кадр данных, длину которого они могут узнать из кадра CTS.

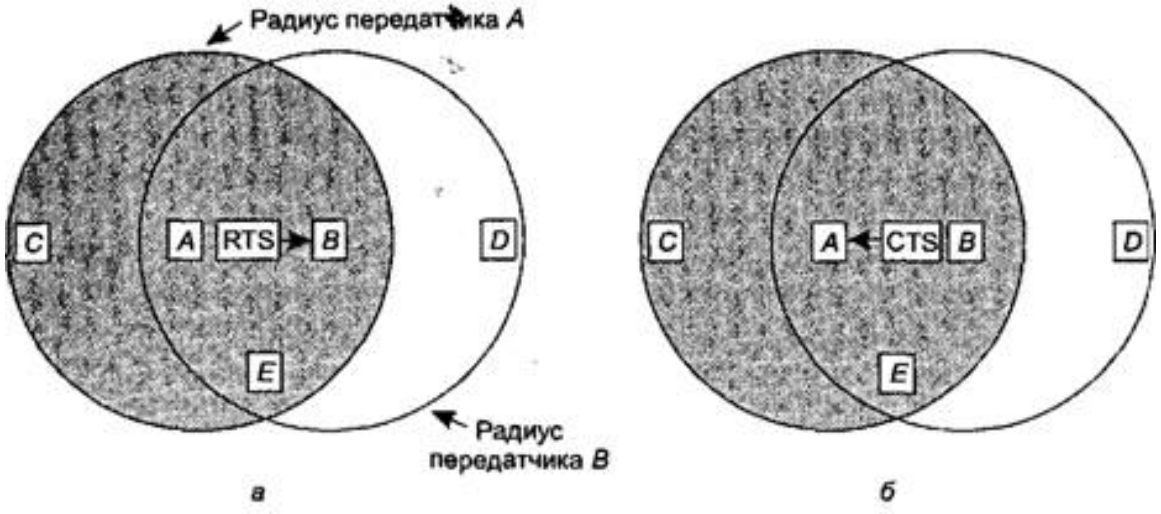


Рис. 11: Протокол MACA: станция A посылает кадр RTS станции B (а); станция B отвечает кадром CTS станции A (б).

На рисунке станция C находится в зоне станции A, но не входит в зону станции B. Поэтому она слышит кадр RTS, передаваемый станцией A, но не слышит кадр CTS, которым отвечает станция B. Поскольку она не интерферирует с кадром CTS, она не обязана воздерживаться от передачи в то время,

пока пересылается информационный кадр. Станция D, напротив, находится близко от станции B, но далеко от станции A. Она не слышит кадра RTS, но слышит кадр CTS, а это означает, что она находится вблизи станции, собирающейся принять кадр с данными. Поэтому ей нельзя вести передачу, пока этот кадр не будет передан. Станция E слышит оба управляющих сообщения и так же, как и станция Д должна хранить молчание, пока не будет завершена передача информационного кадра.

Несмотря на все меры предосторожности, конфликты все равно могут произойти. Например, станции B и C могут одновременно послать кадры RTS станции A. При этом кадры столкнутся и не будут приняты. В этом случае передатчики, не услышав кадр CTS в установленный срок, ждут случайное время и после этого повторяют попытку. Алгоритм выдержки времени, использующийся в случае конфликта, называется двоичным экспоненциальным откатом, и мы изучим его, когда будем рассматривать сеть Ethernet.

Основываясь на изучении модели протокола MACA, Бхаргаван (Bharghavan) сотоварищи в 1994 году осуществили тонкую настройку протокола MACA, чтобы улучшить его производительность. Новый протокол был назван MACAW (MACA for Wireless – MACA для беспроводных сетей). Для начала исследователи заметили, что без подтверждений на уровне передачи данных потерянные кадры не передавались повторно, пока их нехватку не обнаруживал транспортный уровень. Для решения этой проблемы был введен кадр подтверждения (ACK), которым получатель отвечал на каждый успешно принятый кадр данных. Кроме того, было использовано свойство протокола CSMA – станции научились прослушивать эфир и воздерживаться от передачи кадра RTS, если рядом уже кто-то передавал такой же кадр той же станции. Также было решено связать алгоритм выдержки времени (в случае конфликта) не с отдельной станцией, а с потоком данных, то есть с парой станций «источник – приемник». Это изменение протокола очень улучшило его. Наконец, был добавлен механизм обмена между станциями информацией о перегрузке. Кроме того, алгоритм выдержки времени в случае конфликта был несколько смягчен, что улучшило производительность системы.

31 Канал с множественным доступом: систем Aloha, модель работы и оценка пропускной способности.

(Том 1, стр. 142.)

Основной вопрос для каналов с множественным доступом данных: как определить, кому из абонентов, запросивших канал, предоставить право пользоваться им. Протоколы для решения данной проблемы относятся к подуровню канального уровня, который называется **подуровнем доступа к среде** (MAC – Medium Access Control).

Aloha.

Была разработана в 1970 году Абрамсоном из университета Гавайи. Система состояла из наземных радиостанций, работающих на одной частоте и связывающей острова между собой. Идея ее конструкции заключалась в том, чтобы позволить в вещательной среде любому количеству пользователей неконтролируемо использовать один и тот же канал.

См. вопрос №36.

32 Семейство протоколов IEEE 802.11. Система передачи данных WiFi: принципы организации, структура кадра, алгоритм функционирования.

(Стр. 107, 158, 162.)

В IEEE 802.11 описаны семейство стандартов, определяющих функционирование беспроводных локальных сетей.

Физический уровень. Первой и ключевой технологией стандарта 802.11 является технология расширения спектра передачи методом **прямой последовательности** (Direct Sequence Spread Spectrum – DSSS). Использование DSSS позволяет беспроводным интерфейсам передавать данные со скоростью от 1 до 2 Мбит/с.

Идея метода. Пусть имеется канал с широкой полосой пропускания. Разобьем его на полосы. Каждому значению бита сопоставим определенный кот с длиной, равной числу полос, на которые разбили канал. Теперь будем передавать каждый бит, параллельно передавая его код, причем каждый элемент кода(чип) в своей полосе. Такой способ передачи позволяет эффективнее использовать полосу пропускания канала, и он более надежен по сравнению с традиционным способом передачи.

Канальный уровень. Минимально сеть wifi может содержать всего два устройства. В этом случае организуется выделенная есть, в которую входят все беспроводные интерфейсы этих устройств. Данное соединение можно сравнить с соединением типа точка-точка в проводной связи.

Для решения задачи совместимости в сети устанавливается выделенный узел – точка доступа, представляющая собой устройство, имеющее проводной интерефейс для подключения к проводной сети, а также антенну, образующую вокруг себя зону покрытия точки доступа. Радиус покрытия точки доступа от 90 до 150м.

Когда абонентское устройство включается внутри сети, оно начинает прослушивать эфир в поисках совместимого устройства, с которым оно могло бы взаимодействовать. Этот этап называется **сканированием**. При активном сканировании генерируется широковещательный запрос от абонентского устройства, обязательно включающий в себя идентификатор сети (SSID), к которой он хочет присоединиться. Когда запрос достигает точки доступа, имеющий запрашиваемый идентификатор, эта сеть генерирует ответ на запрос. При **пассивном сканировании** абонентское устройство слушает эфир и ожидает появления кадров-маяков, которые периодически рассылаются точками доступа. Когда абонентское устройство получает кадр-маяк, в котором указан SSID, оно пытается присоединиться к указанной сети. Пассивное сканирование – постоянный процесс, при котором устройства могут присоединиться к точке доступа или отсоединяться по мере изменения мощности радиосигнала.

Структура кадра. В wifi определены 3 типа кадров: контрольные, управляющие и кадры данных. Структрура совпадает с IEEE802.3, за исключением некоторых отличий: размер поля данных равен 1500 байт, максимальный размер кадра составляет 2346 байт.

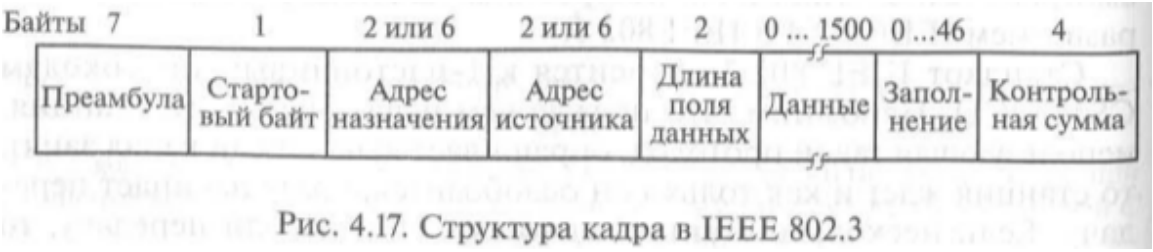


Рис. 4.17. Структура кадра в IEEE 802.3

33 Принципы организации и функционирования семейства протоколов IEEE 802.3, оценка производительности.

Стандарт 802.3 относится к 1-настойчивым протоколам CSMA/CD. Напомним, что прежде, чем начать передачу, станция, использующая такой протокол, опрашивает канал. Если канал занят, то станция ждёт и как только он освободится, сразу начинает передачу. Если несколько станций одновременно начали передачу, то возникает коллизия и передача тут же прекращается. Станции ожидают некоторый случайный промежуток времени, и всё начинается сначала.

Структура кадра в IEEE 802.3 показана на рисунке. Кадр начинается с преамбулы – 7 байт вида 10101010, которая в манчестерском коде на скорости 10 МГц обеспечивает интервал времени 5.6 мкс для синхронизации приёмника и передатчика. Затем следует стартовый байт 10101011, обозначающий начало передачи.

Хотя стандарт IEEE 802.3 допускает двух- и шестибайтовые адреса назначения, для 10Base используются только шестибайтовые. Нуль в старшем бите адреса получателя указывает на обычный адрес, а

единица – это признак группового адреса. Групповой адрес позволяет обращаться сразу к нескольким станциям одновременно. Адрес получателя, состоящий из одних единиц, – вещательный адрес, т.е. этот кадр должны получить все станции в сети.

Адресация обеспечивает также возможность различения локального и глобального адресов. На то, какой адрес используется, указывает 46-й бит. Если этот бит равен 1 – это локальный адрес, который устанавливает сетевой администратор, и вне данной СПД этот адрес смысла не имеет. Глобальный адрес устанавливает IEEE, гарантируя при этом, что нигде в мире нет второго такого адреса. С помощью 46 бит можно получить $7 * 10^{13}$ глобальных адресов.

Поле данных в кадре может занимать от 0 до 1500 байт. Поле данных длиной 0 создаёт проблему для обнаружения коллизий, поэтому IEEE 802.3 предписывает, что кадр не может быть короче 64 байт. Если длина поля данных недостаточна, то поле Заполнение компенсирует нехватку длины. Этот приём называется расширением носителя.

Ограничение длины кадра связано со следующей проблемой. Если кадр короткий, то станция может закончить передачу прежде чем начало этого кадра достигнет самого отдалённого получателя. В этом случае станция может пропустить коллизию, ошибочно считая, что кадр доставлен благополучно. В IEEE 802.3 (при 2,5 км и четырёх репитерах) минимальное время обнаружения коллизии равно 51,2 мкс, что соответствует 64 байт. Например, при скорости 1 Гбит и длине сегмента 2,5 км она должна будет равна 6'400 байт.

Последнее поле в структуре кадра – это контрольная сумма, которая формируется с помощью CRC-кода.

Байты	7	1	2 или 6	2 или 6	2	0...1500	0...46	4
Структура	Преамбула	Стартовый байт	Адрес назначения	Адрес источника	Длина поля данных	Данные	Заполнение	Контрольная сумма

Теперь рассмотрим, как определяется случайная задержка при возникновении коллизий. На этапе состязаний время разбивается на слоты длиной, соответствующей удвоенному времени распространения сигнала до самой удалённой станции (2T). В IEEE 802.3, как уже указывалось, это время равно 51,2 мкс.

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 снова. Тогда выбор будет происходить среди чисел $0, 2^i$, 1, где i – порядковый номер очередной коллизии.

После 10-ти коллизий число слотов достигает 1023 и далее не увеличивается. После 16 коллизии Ethernet-контроллер фиксирует ошибку и сообщает о ней более высокому уровню стека протоколов.

Этот алгоритм, называемый алгоритмом двоичной экспоненциальной задержки, позволяет динамически подстраиваться под число конкурирующих станций. Если для каждой коллизии случайный интервал составляет 1023, то вероятность повторной коллизии двух станций была бы пренебрежимо мала, однако среднее время ожидания разрешения коллизии в этом случае равнялась бы сотням слотов. Если бы случайный интервал составлял постоянно 0 или 1, то при наличии 100 станций разрешение коллизии потребовало бы года, т.к. в этом случае 99 станций должны были бы случайно выбрать, например 0, и лишь одна – 1.

Рассмотрим производительность протоколов стандарта IEEE 802.3 при условии плотной и постоянной нагрузки. Пусть имеется k станций, всегда готовых к передаче. В целях упрощения анализа при коллизиях 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 того, что некоторая станция захватит канал в этом слоте: $A = kp(1-p)^{k-1}$.

Вероятность A достигает максимума при $p = \frac{1}{k}$, $A \to \frac{1}{e}$, при $k \to \infty$. Вероятность того, что период состязаний будет составлять j слотов, равна $A(1-A)^{j-1}$, откуда среднее число слотов в состязаниях определяется в виде: $\sum_{j=0}^{\infty} jA(1-A)^{j-1} = \frac{1}{A}$.

Т.к. каждый слот имеет длительность 2, то средний интервал состязаний $\omega = \frac{2T}{A}$. При $p = \frac{1}{k}$ средний интервал состязаний $\omega \leq 2Te$, что примерно равно 5,4T. Если передача кадра средней длины занимает

m секунд, то при условии работы большого числа станций, постоянно имеющих кадры для передачи, эффективность канала равна $\frac{m}{m+\frac{2T}{A}}$.

Из этой формулы видно, что чем длиннее кабель, т.е. больше T, тем хуже эффективность канала, т.к. растёт длительность периода состязаний. При T = 51,2 мкс, что соответствует 2,5 км при четырёх репитерах и скорости передачи 10Мбит/с, минимальный размер кадра составляет 512 бит, или 64 байта. Хотя с ростом длины кадра эффективность канала растёт, время задержки кадра в системе также увеличивается.

34 Проблемы передачи данных на канальном уровне. Сервис, предоставляемый сетевому уровню. 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.

Основная задача канального уровня – обеспечить сервис сетевому уровню. Назначение этого сервиса – помочь передать данные от процесса, на сетевом уровне одной машины, процессу, на сетевой уровень другой машины.

Канальный уровень может обеспечивать различные классы сервиса. Однако, есть три общие класса 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 прием переданного кадра должен подтверждаться, что до начала передачи должно устанавливаться соединение, которое после передачи должно разрываться. Если в результате помех на физическом уровне кадр будет потерян, то никаких попыток на канальном уровне его восстановить не будет.

Следующий класс сервиса – уведомление без соединения. В этом классе получение каждого посланного кадра должно быть подтверждено. Если подтверждения не пришло в течение определенного времени, то кадр должен быть послан опять. Этот класс сервиса используется в ненадежной физической среде передачи, например, беспроводной.

Наиболее сложный класс сервиса на канальном уровне – сервис с уведомлением и соединением. Этот класс сервиса предполагает, что до начала передачи между машинами устанавливается соединение, и данные передаются по этому соединению. Каждый передаваемый кадр нумеруется и канальный уровень гарантирует, что кадр будет обязательно получен и только один раз, а все кадры будут получены в надлежащей последовательности. При сервисе без соединения этого гарантировать нельзя потому, что потеря уведомления о получении кадра приведет к его пересылке так, что может появиться несколько идентичных кадров.

При сервисе с уведомлением и соединением передача разбивается на три этапа. На первом этапе устанавливают соединение: на обеих машинах инициируют счетчики, отслеживающие какие кадры были приняты, а какие нет. На втором этапе передают один или несколько кадров. На третьем – соединение разрывают: переменные, счетчики, буфера и другие ресурсы, использованные для поддержки соединения, освобождаются.

Разбиение на кадры.

Сервис, создаваемый канальным уровнем для сетевого, опирается на сервис, создаваемый физическим уровнем. На физическом уровне протекают потоки битов. Посланное количество битов не обязательно равно принятому, значение посланного бита так же не обязательно равно принятому. Поэтому нужны специальные усилия на канальном уровне по обнаружению и исправлению ошибок.

Типовой подход к решению этой проблемы – разбиение потока битов на кадры, подсчет контрольной суммы для каждого кадра при посылке данных. При приеме контрольная сумма вычисляется для каждого кадра заново и сравнивается с той, что храниться в кадре. Если они различаются, то это признак ошибки передачи. Канальный уровень должен принять меры к исправлению ошибки, например, сбросить плохой кадр, послать сообщение об ошибке тому, кто прислал этот кадр.

Один из способов разбиения потока битов на кадры – делать временную паузу между битами разных кадров. Однако, в сети, где нет единого таймера, нет гарантии, что эта пауза всегда будет одинаковой или, наоборот, не появятся новые паузы.

Другие методы:

1. Счетчик символов. В начале каждого кадра указывается, сколько символов в кадре. При приеме число принятых символов подсчитывается опять. Этот метод имеет существенный недостаток: счетчик символов может быть искажен при передаче. Тогда принимающая сторона не сможет обнаружить границы кадра. Даже обнаружив не совпадение контрольных сумм, принимающая сторона не сможет сообщить передающей какой кадр надо переслать, сколько символов пропало. Этот метод ныне используется редко.

2. Вставка специальных стартовых и конечных символов. Метод построен на вставке специальных символов. Обычно для этого используют последовательность DLE STX для начала кадра и DLE ETX для конца кадра. При этом методе, если даже была потеряна граница текущего кадра, можно найти границу следующего. Для этого надо просто искать ближайшую последовательность DLE STX или DLE ETX. Здесь есть одна опасность: при передаче чисел или программы в объектном коде такие последовательности могут уже содержаться в передаваемых данных. Для решения этой проблемы используется прием экранирования: каждая последовательность DLE просто дублируется в передаваемых данных. Поэтому, если при приеме есть два последовательных DLE, то один удаляется. Основным недостатком этого метода является то, что он жестко связан с размером байта и кодировкой ASCII.

3. Вставка стартовых и концевых битов. Метод состоит в том, что каждый кадр начинается и заканчивается специальным флаг-байтом: 01111110. Посылающая сторона, встретив последовательно 5 единиц, обязательно вставит 0. Принимающая сторона, приняв 5 последовательных единиц, обязательно удалит следующий за ними 0. Если в передаваемых данных встретиться конфигурация флаг-байта, то она будет преобразована в конфигурацию 011111010.

4. Комбинация этих методов. Например, счетчик символов с одним из выше перечисленных методов. Тогда, если число символов в кадре совпадает с кодировкой границы кадра, кадр считается переданным правильно.

Обнаружение ошибок.

Если нам нужен сервис с подтверждением и с соединением. 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 там сообщение об ошибке, то переданные кадры надо передать заново. Однако, возможны

случаи когда из-за ошибок в канале кадр исчезнет целиком. Для решения этой проблемы на канальном уровне вводят таймеры. Таймер это счетчик, который увеличивает или уменьшает свое значение на единицу автоматически, при получении тактирующего импульса. Если отправитель не получит подтверждение 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. Для решения этой проблемы каждому кадру присваивают порядковый номер. С помощью этого номера получатель может обнаружить дубли.

Управление потоком.

Другой важной проблемой, которая решается на канальном уровне является управление потоком. Суть этой проблемы в том, что отправитель будет слать кадры столь часто, что получатель не будет успевать их обрабатывать. Для борьбы с такими ситуациями вводят управление потоком. Это управление предполагает обратную связь между отправителем и получателем, которая позволяет им урегулировать такие ситуации. Есть 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 кадров. Отправитель, после того как передаст это число кадров, должен приостановить передачу и спросить получателя опять как много кадров тот может принять.

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. Ну, и, наконец, данные в канале не теряются и не искажаются.

Симплекс старт-стопный протокол. Тоже самое, но без условия обработки поступающих данные сколь угодно быстро. Остальное выполнено: канал абсолютно надежный, трафик однонаправленный. Основная проблема здесь как предотвратить ситуацию, когда отправитель «заваливает» данными получателя. Решением такой проблемы может быть введение специальных, коротких служебных сообщений. Получатель, получив один или несколько кадров, отправляет отправителю специальный, короткий кадр, означающий, что отправитель может передавать следующий. Это, так называемый, старт-стопный протокол.

Симплексный протокол для канала с шумом. Основная проблема здесь состоит в том, что кадр с подтверждением о получении может потеряться целиком. Поскольку проблема различения стоит для кадров m и m + 1, то достаточно одного разряда. 0 для только что посланного кадра и 1 для кадра, посланного повторно. Все кадры, не содержащие корректной нумерации, просто сбрасываются при приеме.

35 Проблемы передачи данных на канальном уровне. Сервис, предоставляемый сетевому уровню. Обнаружение и исправление ошибок (Коды исправляющие ошибки, Коды обнаруживающие ошибки).

Обнаружение и исправление ошибок.

В разных средах характер ошибок разный. Ошибки могут быть одиночные, а могут возникать группами, сразу несколько. Недостатком групповых ошибок является то, что их труднее обнаруживать и исправлять, чем одиночные.

Коды с исправлением ошибок.

Для надежной передачи кодов было предложено два основных метода. Первый – внести избыточность в форме дополнительных битов в передаваемый блок данных так, чтобы, анализируя полученный

блок, можно было бы указать, где и какие возникли искажения. Это, так называемые, **коды с исправлением ошибок**. Второй – внести избыточность, но лишь настолько, чтобы анализируя полученные данные, можно было сказать, есть в переданном блоке ошибки или нет. Это, так называемые, **коды с обнаружением ошибок**.

Пусть данные занимают m разрядов и мы добавляем r разрядов, как контрольные разряды. Нам надо передать слово длины n (n = m + r), которое называют **n-битовым кодословом**.

Количество разных битов в двух кодословах называется **расстоянием Хемминга**.

Если мы хоти обнаруживать d ошибок, то надо чтобы кодослова отстояли друг от друга на расстояние d + 1. Если мы хотим исправлять ошибки, то надо, чтобы кодослова отстояли друг от друга на 2d + 1.

Простым примером кода с обнаружением одной ошибки является **код с битом четности**.

Оценим минимальное количество контрольных разрядов, необходимое для исправления одиночных ошибок. Пусть у нас есть код из m бит сообщения и r контрольных бит. Каждое из 2^m правильных сообщений имеет n неправильных кодослов на расстоянии 1. Таким образом, с каждым из 2^m кодослов связано n + 1 кодослов. Так как общее число кодослов 2^n, то (n+1)$2^m \leq 2^n$, учитывая, что n = m + r, получаем: $(m + r + 1) \leq 2^r$.

Для заданного m, эта формула дает предельно минимальное число контрольных разрядов, необходимое для исправления единичных ошибок. Этот теоретический предел достижим при использовании метода, предложенного Хеммингом. Идея его в следующем: все биты, номера которых есть степень 2 (1, 2, 4,8 , 16 и т.д.) – контрольные, остальные – биты сообщения. Каждый контрольный бит отвечает за четность группы битов, включая себя. Один и тот же бит может относиться к разным группам. Значение бита сообщения определяется по значениям контрольных битов. Чтобы определить какие контрольные биты контролируют бит в позиции k, надо представить значение k по степеням двойки. Получив кодослово, получатель устанавливает специальный счетчик в ноль. Затем он проверят каждый контрольный бит на предмет правильности четности. Если четность нарушена, то порядковый номер этого бита заноситься в счетчик. Если после этой проверки счетчик ноль, то все в порядке. Если нет, то он содержит номер неправильного разряда. Код Хемминга может исправлять только одиночные ошибки. Однако есть прием, который позволяет распространить идеи Хемминга на случай групповых ошибок.

Коды обнаруживающие ошибки.

Применение техники четности «в лоб» в случае групповых ошибок не даст нужного результата. Однако ее можно скорректировать. Пусть нам надо передать n слов по k бит. Расположим их в виде матрицы n на k. Для каждого из n столбцов вычислим бит четности и разместим их в дополнительной строке. Получившаяся матрица затем передается по строкам. По получению матрица восстанавливается и если нарушен хоть один бит, то весь блок передается повторно. Этот метод позволяет обнаружить одиночный пакет ошибок длины n. Против групповых ошибок длины n+1 он бессилен.

Поэтому на практике применяют другую технику, которая называется **полиномиальным кодом** или **циклическим избыточным кодом** (Cyclic Redundancy Code) или **CRC кодом**.

CRC коды построены на рассмотрении битовой строки как строки коэффициентов полинома. k битовая строка – коэффициенты полинома степени k - 1. Самый левый бит строки – коэффициент при старшей степени.

Например, строка 110001 представляет полином $x^5 + x^4 + x^0$.

Полиномиальная арифметика выполняется по модулю 2, т.е. сложение и вычитание происходят без переноса разрядов. Так, что обе это операции эквивалентны EXCLUSIVE OR. Деление выполняется как обычно в двоичной системе с той лишь разницей, что вычитание выполняется по модулю два.

Использование полиномиальных кодов при передаче заключается в следующем. Отправитель и получатель заранее договариваются о конкретном генераторе полиномов G(x), у которого коэффициенты при старшем члене и при младшем члене должны быть равны 1. Пусть степень G(x) равна r. Для вычисления контрольной суммы блока из m бит надо, чтобы обязательно m > r. Идея состоит в том, чтобы добавить контрольную сумму к передаваемому блоку, рассматриваемому как полином M(x) так, чтобы передаваемый блок с контрольной суммой был кратен G(x). Когда получатель получает блок с контрольной суммой, он делит его на G(x). Если есть остаток, то были ошибки при передаче.

Алгоритм вычисления контрольной суммы (здесь r степень G(x)):

1. Добавить r нулей в конец блока так, что он теперь содержит m+r разрядов и соответствует полиному x^rM(x);

2. Разделить по модулю 2 полином x^rM(x) на G(x);

3. Вычесть по модулю 2 остаток (длина которого всегда не более r разрядов) из строки, соответствующей x^rM(x). Результат и есть блок с контрольной суммой (назовем его T(x)).

Этот метод позволяет отлавливать одиночные ошибки. Групповые ошибки длины не более r. Нечетное число отдельных ошибок. Идея обоснования этих утверждений состоит в следующем. Ошибки при передаче означают в терминах полиномов, что мы получим после передачи не T(x), а T(x) + E(x). Если степень E(x) меньше степени G(x), то остаток от деления никогда не будет равен 0. Степень, количество и вид термов в G(x) определяет вид выявляемых ошибок этим методом.

Существует три международных стандарта на вид G(x):

- CRC-12 = $x^12 + x^11 + x^3 + x^2 + x + 1$,
- CRC-16 = $x^16 + x^15 + x^2 + 1$,
- CRC-CCITT = $x^16 + x^12 + x^5 + 1$.

CRC-12 используется для передачи символов из 6 разрядов. Два остальных – для 8 разрядных. CRC-16 и CRC-CCITT ловят одиночные, двойные ошибки, одиночные пакеты ошибок длины не более 16 и нечетное число изолированных ошибок с вероятностью 99,997%.

36 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).

Статические методы доступа к каналу.

При рассмотрении методов множественного доступа к каналу, естественно, возникает идея разделить весь канал на несколько подканалов и каждому потенциальному абоненту предоставить свой подканал - мультиплексирование, или уплотнением канала. Имеется два основных подхода к мультиплексированию: использование частотного (FDM) и временного (TDM) разделения канала. При частотном разделении весь диапазон частот полосы пропускания канала разбивается на поддиапазоны, которые называются подканалами. По каждому подканалу выполняется передача независимо от того, что происходит в других каналах. При временном разделении используется вся полоса пропускания канала для каждого абонента, но при этом время передачи делится на слоты по числу потенциальных абонентов, и каждому из них выделяется свой интервал времени (слот) для передачи. Частотное разделение хорошо работает в условиях, когда число абонентов небольшое и фиксированное и каждый из них обеспечивает плотную загрузку канала. При этом каждому абоненту выделяется своя полоса частот, которую он использует независимо от других. Статическое разделение канала на подканалы является неэффективным решением проблемы доступа при предположении о постоянстве числа абонентов в среднем и нерегулярном трафике у абонентов.

Базовая модель динамического предоставления доступа к каналу.

Пять основных предположений, составляющих основу моделей сетей ЭВМ, в которых в качестве СПД используется канал с множественным доступом:

1. Станции. Модель состоит из N независимых станций (компьютеров, телефонов, факс-машин и т. п.). На каждой станции работает пользователь или программа, генерирующие кадры для передачи. Предполагаем, что если кадр сгенерирован, то станция блокируется, и новый кадр не появится, пока не будет передан первый кадр. Это означает, что станции независимы, и на каждой из них работает только одна программа или один пользователь, генерирующие 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 одинаковые, но программно можно устанавливать станциям приоритеты.

3. Коллизии. Если во время передачи кадра одной станцией другая станция начала передачу своего кадра, то такой случай будем называть коллизией. Предполагаем, что любая станция может обнаружить коллизию и что кадры, разрушенные при коллизии, должны быть посланы повторно позднее. Кроме коллизий других ошибок передачи нет.

4. Время. Возможны две модели времени – непрерывная и дискретная:

   - непрерывное время. Передача кадра может начаться в любой момент. В сети нет единых часов, которые разбивают время на слоты. Другими словами, время является непрерывной функцией, отображающей интересующие нас действия в сети на множество вещественных чисел;
   - дискретное время. Все время работы канала разбивается на одинаковые интервалы, называемые слотами, В слоте может оказаться нуль кадров, если это слот ожидания, один кадр, если в этом слоте передача кадра прошла успешно, и несколько кадров, если в этом слоте произошла коллизия.

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

   - с обнаружением несущей. Станция прежде чем использовать канал всегда определяет, занят он или нет с помощью несущей – сигнала определенной формы. Когда канал не занят, по нему все время передается такой сигнал, а если канал занят, то сигнал в нем отличается от несущей, и станция не начинает передачу;
   - при отсутствии несущей. Станция ничего не знает о состоянии канала, пока не начнёт использовать его. Она сразу начинает передачу и лишь в ходе передачи обнаруживает коллизию, т.к. сигнал, который она «увидит» в канале, будет отличаться от того сигнала, который станция передала в канал.

Говоря о **динамическом доступе**, подразумевают, что отсутствует какая-либо фиксированная политика предоставления доступа к каналу для передачи в отличие от статических методов доступа. При этом любая станция может запросить доступ к каналу в любой момент времени, а методы, доступа лишь определяют правила удовлетворения этих запросов.

Методы множественного доступа ALOHA.

Система состояла из наземных радиостанций, работающих на одной частоте и связывающих острова между собой. Идея ее конструкции заключалась в том, чтобы позволить в вещательной среде любому количеству пользователей неконтролируемо использовать один и тот же канал.

Чистая ALOHA: любой пользователь, желающий передать сообщение, сразу пытается это сделать. Благодаря тому, что в вещательной среде у него всегда есть обратная связь, т.е. он может определить, пытался ли кто-то еще передавать сообщение на его частоте, отправитель может установить возникновение конфликта при передаче.

Обратная связь в среде ЛВС происходит практически мгновенно. Отправитель при этом должен слушать среду передачи до тех пор, пока последний бит его сообщения не достигнет самого отдаленного получателя. Обнаружив конфликт, отправитель ожидает некоторый случайный отрезок времени,

после чего повторяет попытку передачи. Интервал времени на ожидание должен быть случайным, иначе конкуренты, повторяя попытки передачи вызовут коллизию снова. Системы, в которых пользователи конкурируют за получение доступа к общему каналу, называются **системами с состязаниями**.

Неважно, когда произошел конфликт, оба кадра считаются испорченными и должны быть переданы повторно. Контрольная сумма, защищающая данные в кадре, не позволяет различать разные случаи наложения кадров.

Назовем **временем кадра** время, необходимое на передачу кадра стандартной фиксированной длины. Обозначим это время t. Предположим, что число пользователей неограниченно и все они порождают кадры по закону Пуассона со средним числом N кадров за t. Это означает, что вероятность события, при котором будет порождено n кадров за время t, можно записать в виде:

$$P[k] = \frac{G^k e^{-G}}{k!},$$

$$P[n] = \frac{\lambda^n e^{-\lambda}}{n!}, \lambda = N.$$

Поскольку при N > 1 очередь на передачу будет только расти и все кадры будут страдать от коллизий, предположим, что 0 < N < 1. Также предположим, что вероятность за время кадра сделать к попыток передачи, как новых, так и ранее не переданных из-за коллизий кадров, распределяется по закону Пуассона со средним значением G. Понятно, что при этом должно выполняться соотношение $G \geq N$, иначе очередь будет расти бесконечно. При слабой загрузке ($N \sim 0$) будет мало передач, а, следовательно, и коллизий, поэтому G N. При высокой загрузке должно выполняться соотношение G > N. При этом пропускная способность канала (S) будет равна числу кадров, которые надо передать, умноженному на вероятность успешной передачи. Если обозначить P0 вероятность отсутствия коллизий при передаче кадра, то можно записать $S = GP_0$. Рассмотрим внимательно, сколько времени требуется 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), отправитель узнает об этом только через время, равное $t_0 + 2t$. Вероятность появления к кадров при передаче кадра с распределением Пуассона поэтому вероятность, что появится 0 кадров, равна $^{-G}$. За двойное время кадра среднее число кадров равно 2G, откуда $P_0 = e^{-2G}$, а так как $S = GP_0$ то пропускная способность канала $S = Ge^{-2G}$.

Максимальная пропускная способность достигается при G = 0,5 и при $S = \frac{1}{2e}$, что составляет примерно 18% номинальной пропускной способности системы.

Слотированная ALOHA.

Модификация чистой ALOHA, в которой все время работы канала разделяется на слоты. Размер слота при этом должен быть 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. Откуда $S = Ge^{-G}$.

Максимум пропускной способности слотированной ALOHA наступает при G = 1, где S = $S = \frac{1}{e}$, т.е. составляет около 37% , что вдвое больше, чем у чистой ALOHA.

Рассмотрим как G влияет на пропускную способность системы. Для этого подсчитаем вероятность успешной передачи кадра за k попыток. Так как e^{-G} – это вероятность отсутствия коллизии при передаче, то вероятность того, что кадр будет передан ровно за k попыток, можно записать в виде $P_k = e^{-G}(1 - e^{-G})^{k-1}$.

Среднее ожидаемое число повторных передач: $E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1 - e^{-G})^{k-1} = e^G$.

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

Протоколы множественного доступа с обнаружением несущей.

Протоколы, реализующие идею начала передачи только после определения, занят канал или нет, называются **протоколами с обнаружением несущей** – CSMA (Carrier Sensitive Multiple Access).

Настойчивые и ненастойчивые CSMA-протоколы.

Если канал занят, то станция ждет, а как только он освободился, пытается сразу начать передачу. Если при этом произошла коллизия, станция ожидает случайный промежуток времени и все начинает сначала. Этот протокол называется **настойчивым CSMA-протоколом первого уровня** или **1-настойчивым CSMA-протоколом**, поскольку 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 существует вероятность того, что, как только одна станция начала передачу, другая станция также стала готова передавать. Если вторая станция проверит состояние канала прежде чем до нее дойдет сигнал от первой станции о том, что она заняла канал, то вторая станция сочтет канал свободным и начнет передачу. В результате возникает коллизия. Чем больше время задержки сигнала, тем больше вероятность такого случая и тем хуже производительность канала.

Однако даже если время задержки сигнала будет равно нулю коллизии все равно могут возникать. Например, если во время передачи готовыми к передаче оказались две станции. В этом случае они подождут, пока ранее начатая передача будет закончена, а затем 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.

Другим вариантом CSMA-протокола является **ненастойчивый CSMA-протокол**. Основное отличие его от предыдущего состоит в том, что готовая к передаче станция опрашивает канал. Если канал свободен, то она начинает передачу. Если же канал 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 сигнала для станции, но общая эффективность протокола возрастает. И, наконец, настойчивый CSMA-протокол уровня p. Который применяется квотированным каналам. Когда станция готова к передаче, она опрашивает канал. Если канал он свободен, то она с вероятностью p передает свой кадр и с вероятностью q = 1 - p ждет следующего слота. Так станция действует, пока не передаст кадр. Если во время передачи происходит коллизия, станция ожидает случайный промежуток времени и опрашивает канал снова. Если при опросе он опять оказывается занятым, станция ждет начала следующего слота, и весь алгоритм повторяется.

CSMA-протокол с обнаружением коллизий.

Протоколы этого класса широко используются в локальных сетях. Модель их работы следующая. В момент времени t_0 одна станция заканчивает передачу очередного кадра, а все другие станции, у которых имеется кадр для передачи, начинают передачу. Естественно, в этом случае происходят коллизии, которые быстро обнаруживаются посредством сравнения отправленного сигнала с тем сигналом, который есть на линии. Обнаружив коллизию, станция сразу прекращает передачу на случайный промежуток 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).

Рассмотрим подробнее алгоритм состязаний. Определим, сколько времени станции, начавшей передачу, требуется, чтобы обнаружить коллизию. Обозначим t время распространения сигнала до самой удаленной станции на линии. Для коаксиального кабеля длиной в 1 км t = 5 мкс. В этом случае минимальное время для определения коллизии будет равно 2t. Следовательно, станция не может быть уверена, что она захватила канал до тех пор, пока не убедится, что в течение 2t секунд не было коллизий. Поэтому мы будем рассматривать период состязаний как слотированную систему ALOHA со слотом 2t секунд на один бит. Захватив канал, станция может далее передавать кадр с любой скоростью.

Обнаружение коллизий – это аналоговый процесс, поэтому, чтобы обнаруживать их, необходимо использовать специальные кодировки на физическом уровне.

37 Протокол EEE 802.3 и Ethernet (кабели, способ физического кодирования, алгоритм вычисления задержки, MAC подуровень, структура кадра).

Стандарт IEEE 802.3 и Ethernet.

Стандарт IEEE 802.3 относится к 1-настойчивым протоколам CSMA/CD для локальных сетей. 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.

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.

При использовании **манчестерского кода** весь период передачи бита разбивается на два равных интервала. При передаче 1 передается высокий сигнал в первом интервале и низкий во втором. При передаче 0 – наоборот. Такой подход имеет переход в середине передачи каждого бита, что позволяет синхронизироваться приемнику и передатчику. Недостатком такого подхода является то, что пропускная способность канала падает вдвое по сравнению с прямым кодированием. При использовании **дифференциального манчестерского кода** при передаче 1 в начале передачи нет различия в уровне с предыдущим интервалом передачи, т.е. нет перепада в уровне в начале каждого интервала, а при передаче 0 – есть. Этот способ кодирования обладает лучшей защищенностью, чем просто манчестерский код, но требует более сложного оборудования

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.

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 (2t). 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, 2i, 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.

Если передача кадра средней длины занимает m сек, то при условии большого числа станций, постоянно имеющих кадры для передачи, эффективность канала равна $\frac{m}{m+\frac{2t}{A}}$, где A – вероятность, что станция захватит канал, 2t – длительность слота. Хотя с ростом длины кадра эффективность канала растет, время задержки кадра в системе также увеличивается.

Название	Тип кабеля	Максимальная длина сегмента	Кол-во узлов на сегмент	Преимущества	Подключение
10Base5	Толстый коаксиал	500 м	100	Подходит для магистралей	Трансивер на кабеле соединяется с компьютером трансиверным кабелем. Его длина не должна превосходить 50 метров. Он состоит из 5 витых пар.
10Base2	Тонкий коаксиал	200 м	30	Самый дешевый	Т-образное соединение BNC коннектором. Трансивер расположен на контроллере в компе.
10Base-T	Витая пара	100 м	1024	Простое обслуживание	Подключение витухой к хабу.
10Base-F	Оптоволокно	2000 м	1024	Идеально для соединения зданий	

Шина с маркером 802.4.

Физически шина с маркером имеет линейную или древовидную топологию. Логически станции объединены в кольцо, 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.

38 Физические среды передачи данных. Организация физического уровня: СПД, коллизии в Ethernet.

Назначение физического уровня – передавать данные в виде потока битов от одной машины к другой. При этом для передачи данных можно использовать различные физические среды, каждую из которых характеризует следующие параметры:

- Ширина полосы пропускания.
- Пропускная способность.
- Задержка сигнала.
- Стоимость.
- Сложность прокладки.
- Сложность прокладки.
- Сложность обслуживания.
- Достоверность передачи.
- Затухание.
- Помехоустойчивость.

Магнитная лента или магнитный диск:

- + Высокая пропускная способность.
- + Стоимость передачи в расчёте на один бит.
- + Надежность.
- - Огромная задержка.

Витая пара – два медных изолированных провода, один из которых обвит вокруг другого. Этот второй вьющийся провод предназначен для устранения взаимного влияния между соседними витыми парами. Широко используется в телефонии. Протяжённость линии из витой пары между абонентами и автоматизированной телефонной станцией может составлять несколько километров без промежуточных усилений. Витая пара может быть использована для передачи как цифровых, так и аналоговых сигналов. Её пропускная способность зависит от толщины используемых проводов и длины линии. Скорость в несколько мегабит вполне достижима с помощью соответствующих методов передачи. Витые пары объединяют в многопарные кабели. Кабель категории 3 содержит четыре витые пары с невысокой плотностью навивки и имеет ширину полосы пропускания до 16МГц. Наиболее часто используется кабели категории 5, который тоже состоит из четырёх пар, но имеет более плотную навивку, что позволяет достигать более высоких скоростей передачи и ширину полосы пропускания 100МГц.

Как и у витой пары, у коаксиального кабеля есть два проводника. Однако устроены они иначе, что позволяет существенно увеличить ширину полосы пропускания. Центральный проводник представляет собой толстый медный провод, окружённый изолятором. Эта конструкция помещается внутри второго цилиндрического проводника, который обычно представляет собой плетённую плотную металлическую сетку. Всё это закрывается плотным защитным слоем пластика. Обычно толщина коаксиального кабеля составляет от 1 до 2.5 см, поэтому монтировать и прокладывать его сложнее, чем витую пару, однако у такого кабеля шире полоса пропускания и характеристики по затуханию сигнала лучше, чем у витой пары. Основным ограничителями скорости и расстояния при передачи без усиления в этих кабелях

являются затухания сигнала, тепловой и интермодуляционный шумы. Когда всю полосу пропускания кабеля разбивают на более узкие и каждую такую полосу используют как отдельный канал, на границах таких каналов возникает интермодуляционный шум.

Для создания оптической связи требуется источник света и постоянной длиной волны, светопроводящая среда и детектор, преобразующий световой поток в электрический. На одном конце оптоволоконной линии имеется передатчик – источник света, световой импульс от которого проходит по светопроводящему волокну и попадает на детектор, расположенный на другом конце этой линии и преобразующий этот импульс в электрический. Одна из основных проблем при создании оптоволоконных систем состояла в том, чтобы не дать световому пучку рассеяться через боковую поверхность силиконового шнура. Количество рассеиваемой энергии зависит от угла падения светового луча на стенки шнура. При углах больше некоторого критического угла, называемого углом полного внутреннего отражения, вся энергия луча отражается обратно внутрь. Оказалось, что силиконовый шнур, имеющий толщину, близкую к длине волны источника света, работает как провод для тока, т.е. без потерь на внутреннее отражение. По такому одномодовому шнуру можно передать данные со скоростью несколько гигабит в секунду на сотню километров без промежуточного усиления. Поскольку можно испускать несколько лучей разной длины волны так, что они попадали на границы шнура под углом больше, чем угол полного внутреннего отражения, следовательно, по одному шнуру можно пускать несколько лучей. При этом каждый луч, как говорят, имеет свою моду. Так получается многомодовый шнур. Другой проблемой при использовании оптоволокна является дисперсия – потеря по мере распространения исходными световыми импульсами начальных форм и размеров. При этом искажения также зависят от длины волны. Одно из возможных решений этой проблемы – увеличение расстояния между соседними сигналами. Однако это сокращает скорость передачи. К счастью, исследования показали, что при придании сигналу некоторой специальной формы дисперсионные эффекты почти исчезают и сигнал можно передавать на тысячи километров. Сигналы, имеющие такую форму, называются силитонами.

Преимущества относительно коаксиального:

- Ширина полосы пропускания больше.
- Более компактное и имеет меньшую массу.
- Затухание сигнала существенно меньше.
- Не восприимчиво к внешним электромагнитным излучения.

Кабели в стандарте IEEE 802.3.

Первым появился так называемый толстый Ethernet(10Base5). Коаксиальный кабель жёлтого цвета с отметками через каждые 2.5 м, указывающими, где можно производить подключение. Подключение выполняется через специальные розетки, которые монтируются прямо на кабеле. В эти розетки встраивался специальный прибор – трансивер, отвечающий за обнаружение несущей частоты и коллизий. Когда трансивер обнаруживает коллизию, он посылает специальный сигнал по кабелю, гарантирующий, что другие трансиверы услышат эту коллизию. Кабель обеспечивает пропускную способность 10 Мбит/с, а максимальная длина равна 500м(10Base5).

Вторым появился 10Base2. Это более простой в употреблении коаксиальный кабель с простым подключением через BNC-коннектор, представляющий собой Т-образное соединение коаксиальных кабелей. Его сегмент не должен превышать 200м и объединять более 30 машин.

Решение проблемы поиска обрыва, частичного повреждения кабеля или плохого контакта в коннекторе привели к созданию совершенно иной кабельной конфигурации на основе витой пары. В этом случае каждая машина соединяется витой парой со специальным устройством – хабом. Такой способ обозначается 10Base-T.

Трансиверы в 10Base5 размещаются прямо на кабеле и соединяются с компьютером трансиверным кабелем, длина которого не может превышать 50м. Трансиверный кабель состоит из пяти витых пар. Две из них используются для передчи данных к компьютеру и от него, две служат для передачи управляющей информации в обе стороны, а пятая – для подачи питания на трансивер.

Трансиверный кабель подключается к контроллеру в компьютере через интерфейс AUI. В контроллере имеется специальная микросхема, отвечающая за приём кадром и их отправку, проверку и формирование контрольной суммы. В некоторых случаях эта микросхема отвечает и за управление буферами на канальном уровне, очередью буферов на отправку и обеспечивает прямой доступ к памяти машины, а также решает другие вопросы доступа к сети.

В 10Base2 трансивер располагается на контроллере, и каждая машина должна иметь свой индивидуальный трансивер.

В 10Base-T трансивера нет вовсе. Машины соединяются хабом с витой парой, длина которой не должна превышать 100м. Вся электроника сосредоточена в нём.

Последний используемый в стандарте IEEE 802.3 тип кабеля – оптоволокно 10Base-F, которое относительно дорогое, но обеспечение низкого уровня шума и большая длина одного сегмента являются его преимуществом. Для увеличения длины сегментов в этом стандарте используются репитеры – устройства физического уровня, отвечающие за очистку, усиление и передачу сигнала. Репитеры не могут отстоять друг от друга более чем на 2.5 км, и на одном сегменте их не может быть более четырёх.

39 Протокол LLC уровня управления логическим каналом (IEEE802.2). Структура кадров в протоколе IEEE802.2.

Основное назначение протокола LLC(стандарт 802.2) – обеспечение требуемого качества услуг системы передачи данных посредством передачи своих кадров либо дейтаграммным способоом, либо с помощью процедур с установлением соединения и восстановлением кадров, а также обеспечение независимости вышерасположенных уровней стека протоколов от конкретного типа физической среды канала с множественным доступом.

Этот протокол занимает уровень между сетевыми протоколами и протоколами уровня MAC. Протоколы сетевого уровня передают через межуровневый интерфейс данные для протокола LLC: свой пакет (например пакет IP, IPX или NetBEUI), адресную информацию об узле назначения, а также требования к качеству транспортных услуг, которые этот протокол должен обеспечить. Он помещает протокол верхнего уровня в свой кадр, который дополняется необходимыми служебными полями. Далее, через межуровневый интерфейс передает свой кадр вместе с адресной информацией об узле назначения соответствующему протоколу уровня MAC, который упаковывает кадр LLC в свой кадр.

В основу LLC положен HDLC, который широко используется в территориальных сетях. Спецификация IEEE 802.2 несколько отличается от этого стандарта. Сначала подуровень LLC в технологиях фирм-изноготовителей сетевого оборудования не выделялся в самостоятельный подуровень – его функции были частью общих функций канального уровня.

Из-за больших различий в функциях протоколов фирменных технологий, которые можно отнести к LLC, на этом уровне пришлось ввести три типа процедур:

1. **LLC1** – сервис без установления соединения и без подтверждения.
2. **LLC2** – сервис с установлением соединения и подтверждением.
3. **LLC3** – сервис без установления соединения, но с подтверждением.

Этот набор процедур является общим для всех методов доступа к среде, определенных стандартами 802.3 – 802.5.

Сервис без установления соединения и без подтверждения LLC1 дает пользователю средства для передачи данных с минимумом издержек. Обычно, этот вид сервиса используется тогда, когда такие функции как восстановление данных после ошибок и упорядочивание данных выполняются протоколами вышележащих уровней, поэтому нет нужды дублировать их на уровне LLC.

Сервис с установлением соединений и с подтверждением LLC2 дает пользователю возможность установить логическое соединение перед началом передачи любого блока данных и, если это требуется, выполнить процедуры восстановления после ошибок и упорядочивание потока этих блоков в рамках

установленного соединения. Протокол LLC2 во многом аналогичен протоколам семейства HDLC(LAP-B, LAP-D, LAP-M), которые применяются в глобальных сетях для обеспечения надежной передачи кадров на зашумленных линиях.

В некоторых случаях (например, при использовании сетей в системах реального времени, управляющих промышленными объектами), когда временные издержки установления логического соединения перед отправкой данных неприемлемы, а подтверждение корректности приема переданных данных необходимо, базовый сервис без установления соединения и без подтверждения не подходит. Для таких случаев предусмотрен дополнительный сервис, называемый сервисом без установления соединения, но с подтверждением LLC3.

Чаще всего в локальных сетях используются протоколы LLC1(например, в стеке TCP/IP). Это объясняется тем, что кабельные каналы локальных сетей обеспечивают низкую вероятность искажений бит и потери кадров. Поэтому, использование повышающего надежность обмена протокола LLC2 часто приводит к неоправданной избыточности, только замедляющей общую пропускную способность стека коммуникационных протоколов. Тем не менее, иногда протокол LLC2 применяется и в локальных сетях. Так, этот протокол используется стеком SNA в том случае, когда мэйнфремы или миникомпьютеры IBM взаимодействуют через сети Token Ring. Протокол LLC2 используется также компанией Hewlett-Packard в том случае, когда принтеры подключается к сети Ethernet непосредственно, с помощью встроенных сетевых адаптеров.

Структура кадров.

По своему назначению все кадры уровня LLC (называемые в стандарте 802.2 блоками данных - Protocol Data Unit, PDU) подразделяются на три типа: **информационные**, **управляющие** и **ненумерованные**:

1. **Информационные кадры** предназначены для передачи информации в процедурах с установлением логического соединения и должны обязательно содержать поле информации. В процессе передачи информационных блоков осуществляется их нумерация в режиме скользящего окна.

2. **Управляющие кадры** предназначены для передачи команд и ответов в процедурах с установлением логического соединения, в том числе запросов на повторную передачу искаженных информационных блоков.

3. **Ненумерованные кадры** предназначены для передачи ненумерованных команд и ответов, выполняющих в процедурах без установления логического соединения передачу информации, идентификацию и тестирование LLC-уровня, а в процедурах с установлением логического соединения - установление и разъединение логического соединения, а также информирование об ошибках.

Все типы кадров уровня LLC имеют единый формат:

Флаг 01111110	DSAP (адрес точки входа службы назначения)	SSAP (адрес точки входа службы источника)	CTRL (управляющее поле)	DATA (данные)	Флаг 01111110

Они содержат четыре поля:

1. Адрес точки входа сервиса назначения (Destination Service Access Point, DSAP).

2. Адрес точки входа сервиса источника (Source Service Access Point, SSAP).

3. Управляющее поле (Control, CTRL).

4. Поле данных (Data).

Обрамляются двумя однобайтовыми полями «Флаг», имеющими значение «01111110». Флаги используются на MAC-уровне для определения границ кадра. (Отметим, что формат кадров LLC, за исключением поля адреса точки входа сервиса источника, соответствует формату кадра HDLC, а также одного из вариантов протокола HDLC – протокола LAP-B, используемого в сетях X.25).

Поле данных кадра LLC предназначено для передачи по сети пакетов протоколов верхних уровней - IP, IPX, AppleTalk, DECnet, в редких случаях - прикладных протоколов, когда те не пользуются сетевыми протоколами, а вкладывают свои сообщения непосредственно в кадры канального уровня. Поле данных может отсутствовать в управляющих кадрах и некоторых ненумерованных кадрах.

Поле управления (один байт) используется для обозначения типа кадра данных – информационный, управляющий или ненумерованный. Кроме этого, в этом поле указываются порядковые номера отправленных и успешно принятых кадров, если подуровень LLC работает по процедуре LLC2 с установлением соединения. Формат поля управления полностью совпадает с форматом поля управления кадра LAP-B.

Поля DSAP и SSAP позволяют указать, какой сервис верхнего уровня пересылает данные с помощью этого кадра. Программному обеспечению узлов сети при получении кадров канального уровня необходимо распознать, какой протокол вложил свой пакет в поле данных поступившего кадра, для того, чтобы передать извлеченный из кадра пакет нужному протоколу для последующей обработки. Например, в качестве значения DSAP и SSAP может выступать код протокола IPX или же код протокола покрывающего дерева Spanning Tree.

40 Сетевые коммутаторы: организация, основные функции, принципы функционирования. Обучающийся коммутатор канального уровня.

Сетевой коммутатор представляет собой устройство с несколькими портами, к которым можно подключать сегменты с множественным доступом, например сегменты 802.3. на основании таблицы коммутании, расположенной в памяти коммутатора, кадры с входного порта коммутатор передает на надлежащий выходной порт, поэтому трафик на отдельных сегментах существенно ниже, чем на коммутаторе в целом. Коммутаторы работают на более высоких скоростяз, чем мосты и функционально болеее гибкие.

По сравнению с мостами у коммутаторов больше портов. Достаточно распространены коммутаторы с 24 и 48 портами со скоростями 10 и 100 Мбит/с. Коммутаторы на крупных предприятиях могут поддерживать сотни портов. У них больше размер буферного пространства для сохранения принимаемых кадров, что весьма полехно, особенно если элементы сети перегружены. Возможна поддержка локальных СПД с КМП с разными скоростями.

Есть разные режимы коммутации между сетевыми портами:

1. Коммутация без буферизации кадров (получив 6-байтовый MAC-адрес получателя, коммутатор сразу занимает требуемый выходной порт и начинает передачу. Обнаржения ошибок нет, если произойдет коллизия, то кадр будет потерян).

2. Коммутация с буферизацией кадров (в буфере сохраняется весь кадр, при сохранении коммутатор анализирует и находит ошибки, убедившись в отсутствии коллизии на входе, передает кадр а выходной порт).

3. Коммутация с исключением фрагментов (обеспечивает малое время задержки. Получив 64-битный заголовок кадра, коммутатор занимает требуемый выходной порт и начинает передачу. Обнаружения ошибок нету, т.к. гарантируется что из источника считалось достаточное количество байт чтобы обнаружить коллизию еще до пересылки).

Последний вариант является компромисным между большим временем задержки и гарантией целостности (при коммутации с буферизацией) и небольшим временем задержки и риском потери кадра (без буферизации). На практике это не столь важно, т.к. несущественное уменьшение времени задержки

при коммутации без буферизации возмещается незначительными колебаниями времени ожидания при коммутации с буферизацией.

Для каждого порта коммутатор формирует таблицу MAC-адресов, связанных с сегментом, подключенным к порту коммутатора. Затем коммутатор использует их при принятии решения о дальнейших операциях с кадрами: фильтрация, пересылка или лавинная рассылка.

Когда на порт поступает кадр, коммутатор сравнивает MAC-адрес адресата с адресами в таблицах. Если адрес получателя кадра находится в том же сегменте сети, что и отправитель, коммутатор сбрасывает кадр. Этот процесс называется фильтрацией, и сего помощью коммутаторы могут значительно уменьшить трафик между сегментом сети. Если адрес получателя находится в другом сегменте, коммутатор пересылает кадр на порт, к которому подключен соответствующий сегмент.

Если у коммутатора нет записи об адресе получателя, то он передаст кадр всем портам, кроме того порта, с которого кадр был получен. Устройство-получатель отвечает на штроковещательгую рассылку специальным кадром по адресу отправителя. Коммутатор вводит искомый адрес получателя и номер соответствующего порта коммутатора в таблицу MAC-адресов. Теперь коммутатор сможет пересылать кадры между отправителем и получателем без широковещательной рассылки.

41 Виртуальные сети на основе протокола IEEE 802.1Q. Реализация Router-on-a-stick.

В виртуальных сетях, основанных на стандарте IEEE 802.1Q, информация о принадлежности передаваемых Ethernet-кадров к той или иной виртуальной сети встраивается в сам передаваемый кадр. Таким образом, стандарт IEEE 802.1 Q определяет изменения в структуре кадра Ethernet, позволяющие передавать информацию о VLAN по сети. Для этого к кадру Ethernet добавляют метку Tag длиной 4 байт и получают кадры, называемые кадрами с метками – Tagged frame, где дополнительные биты содержат информацию о принадлежности кадра Ethernet к виртуальной сети и о его приоритете.

Добавляемая к кадру метка включает в себя двухбайтовое поле TPID (Tag Protocol Identifier) и двухбайтовое поле TCI (Tag Control Information). Поле TCI состоит из полей Priority (3 бита, задает восемь возможных уровней приоритета кадра), CFI (Canonical Format Indicator, 1 бит, зарезервировано для обозначения кадров СПД других типов, передаваемых по магистрали Ethernet, для кадров Ethernet всегда равно 0) и V1D (VLAN ID, 12 бит, идентификатор виртуальной сети).

Изменение формата кадра Ethernet приводит к тому, что сетевые устройства, не поддерживающие стандарт IEEE 802.1Q (называемые Tag-unaware), не могут работать с кадрами, в которые вставлены метки, а подавляющее большинство этих устройств (в частности, сетевые Ethernet-контроллеры конечных узлов сети) в настоящее время не поддерживают этот стандарт. Следовательно, для обеспечения совместимости с устройствами, поддерживающими стандарт IEEE 802.1Q (Tag-aware-устройств а), коммутаторы стандарта IEEE 802.10 должны поддерживать как традиционные Ethernet-кадры, т. е. кадры без меток (Untagged), так и кадры с метками (Tagged). Для обеспечения поддержки различных типов трафиков и образования внутреннего трафика коммутатора из пакетов типа Tagged кадры на принимаемом и передающем портах коммутатора должны преобразовываться в соответствии с определенными правилами.

Правила входного порта. По отношению к трафику каждый порт коммутатора может быть как входным, так и выходным. После принятия кадра входным портом коммутатора решение о его дальнейшей обработке принимается на основании определенных правил входного порта (Ingress rules). Поскольку принимаемый кадр может быть как типа Tagged, так и типа Untagged, то правила входного порта определяют, какие типы кадров должны приниматься портом, а какие отфильтровываться. По умолчанию для всех коммутаторов правилами входного порта устанавливается возможность приема кадров обоих типов. Если правилами входного порта определено, что он может принимать кадр типа Tagged, в котором имеется информация о принадлежности к конкретной виртуальной сети (VID), то этот кадр передается без изменения. Если же этими правилами определена возможность работы с кадрами типа Untagged, в которых не содержится информация о принадлежности к виртуальной сети, то прежде всего такой

кадр преобразуется входным портом коммутатора к типу Tagged. Чтобы такое преобразование стало возможным, каждому порту коммутатора присваивается уникальный идентификатор PVID (Port VLAN Identifier), определяющий принадлежность порта к конкретной виртуальной сети внутри коммутатора (по умолчанию все порты коммутатора имеют одинаковый идентификатор PVID = 1). Для преобразования кадра типа Untagged к типу Tagged его необходимо дополнить меткой VTD. Значение поля VID входного Untagged-кадра устанавливается равным значению PVID входящего порта, т.е. все входящие Untagged-кадры автоматически приписываются к той виртуальной сети внутри коммутатора, к которой принадлежит входной порт.

Правила продвижения пакетов. После того как все входящие кадры отфильтрованы, преобразованы или оставлены без изменения в соответствии с правилами входящего порта, решение об их передаче к выходному порту основывается на определенных правилах продвижения пакетов (Forwarding Process).

Пакеты могут передаваться только между портами, ассоциированными с одной виртуальной сетью. Порты с одинаковыми идентификаторами внутри одного коммутатора ассоциируются с одной виртуальной сетью. Если же виртуальная сеть строится на базе одного коммутатора, то идентификатора порта PV1D, определяющего его принадлежность к виртуальной сети, вполне достаточно.

Стандарт IEEE 802.1Q задумывался для обеспечения построения масштабируемой инфраструктуры виртуальных сетей, включающей в себя множество коммутаторов. При этом для расширения сети за пределы одного коммутатора наличие одних идентификаторов портов недостаточно, поэтому каждый порт может быть ассоциирован с несколькими виртуальными сетями, имеющими различные идентификаторы VID.

Если адрес назначения пакета соответствует порту коммутатора, принадлежащему к той же виртуальной сети, что и сам пакет (могут совпадать VID пакета и V1D порта или VID пакета и PVID порта), то такой пакет может быть передан. Если же передаваемый кадр принадлежит к виртуальной сети, с которой выходной порт никак не связан (VID пакета не соответствует PVID или VID порта), то кадр не может быть передан и отбрасывается.

Правила выходного порта.

После того как кадры внутри коммутатора переданы на выходной порт, их дальнейшее преобразование зависит от правил выходного порта (Egress rules). Правилами выходного порта (правилом контроля метки – Tag Control) определяется, следует ли преобразовывать кадры Tagged к формату Untagged.

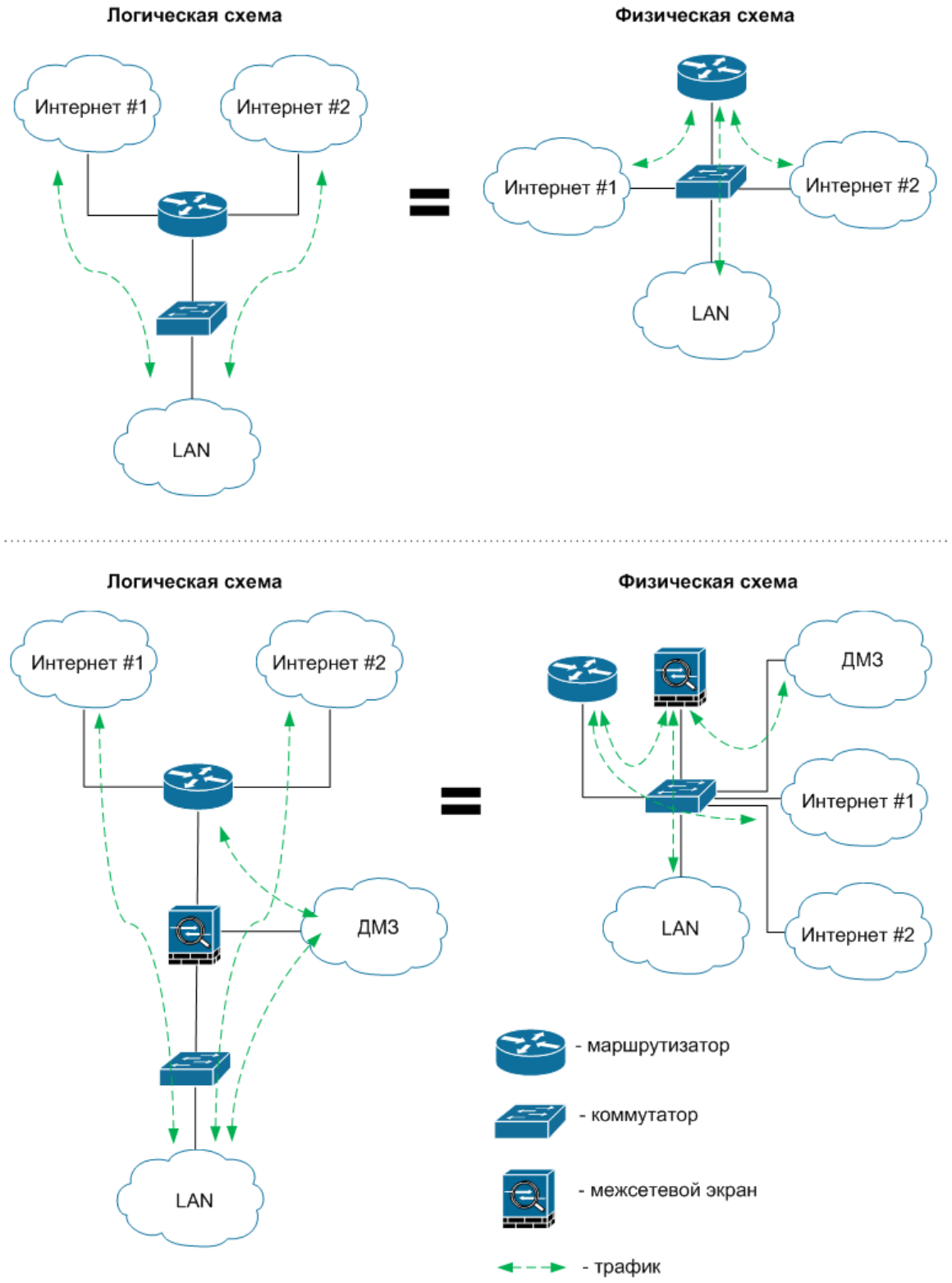
Каждый порт коммутатора может быть сконфигурирован как Tagged Port или Untagged Port. Если выходной порт коммутатора определен как Tagged Port, то исходящий трафик будет создаваться кадрами типа Tagged с информацией о принадлежности к виртуальной сети. Выходной порт в этом случае не изменяет тип кадров, оставляя их такими же, какими они были внутри коммутатора. К такому порту может подсоединяться только устройство, совместимое со стандартом IEEE 802.1Q, например коммутатор или сервер с сетевой картой, поддерживающей работу с виртуальными сетями данного стандарта.

Если же выходной порт коммутатора определен как Untagged Port, то все исходящие кадры преобразуются к типу Untagged, т.е. из них удаляется дополнительная информация о принадлежности к виртуальной сети. К такому порту можно подключать любое сетевое устройство.

Router-on-a-Stick.

Подобному тому, как коммутатор может разделить локальную сеть на множество VLAN, так и маршрутизатор может использовать один физический интерфейс для создания подмножества логических виртуальных интерфейсов и обеспечить маршрутизацию данных, видео или голоса между ними.

В качестве наглядного примера, на схемах я хочу продемонстрировать некоторые возможные сценарии, которые можно реализовать с помощью одного физического порта и подмножества виртуальных интерфейсов на маршрутизаторе или межсетевом экране Cisco.



Как мы видим, для решения поставленной задачи требуется коммутатор, желательно 3-го уровня. Коммутатор должен обладать достаточной пропускной способностью, чтобы снизить потенциальные задержки передачи пакетов в случае возникновения больших объемов трафика. Если это модульный коммутатор, тогда желательно обзавестись резервным блоком питания и резервным управляющим процессором для него.

Кроме очевидных преимуществ подобной архитектуры, есть также некоторые недостатки, одним из которых является увеличенная в несколько раз нагрузка на отдельно взятый физический порт устройства. Но бывают ситуации, когда обойтись без виртуальных интерфейсов нельзя. Так, например, если немного отвлечься от темы, то невозможно выстроить отказоустойчивую связку двух межсетевых экранов в режиме Active/Passive, если не подключить каждый из них одним физическим линком к коммутатору, а вторым объединив их между собой для обмена служебными данными. В случае выхода из строя одного

межсетевого экрана, его место займет второй с идентичной конфигурацией.

42 Сетевые коммутаторы. Маршрутизация по соединяющему дереву (протокол STP).

Сетевой коммутатор (далее просто коммутатор) представляет собой устройство с несколькими портами, к которым можно подключать сегменты каналов с множественным доступом (далее КМД), например сегменты 802.3. На основании таблицы коммутации, расположенной в памяти коммутатора, кадры с входного порта коммутатор передает на надлежащий выходной порт, поэтому трафик на отдельных сегментах существенно ниже, чем на коммутаторе в целом. Коммутаторы работают на более высоких скоростях, чем мосты, и функционально являются более гибкими.

По сравнению с мостами у коммутаторов больше портов. Достаточно распространены коммутаторы с 24 и 48 портами со скоростями соответственно 10 и 100 Мбит/с. Коммутаторы на крупных предприятиях могут поддерживать сотни портов. У коммутаторов больше размер буферного пространства для сохранения принимаемых кадров, что весьма полезно, особенно если элементы сети перегружены. В зависимости от стоимости коммутатора возможна поддержка локальных сегментов СПД с КМД с разными скоростями: 10 Мбит/с, 100 Мбит/с, 1 Гбит/с или 10 Гбит/с.

Для коммутации данных между сетевыми портами коммутаторы используют один из следующих методов:

- **Коммутация без буферизации кадров.** При использовании этого метода коммутатор, получив 6-байтовый MAC-адрес получателя, сразу занимает требуемый выходной порт и начинает передачу. Обнаружения ошибок при этом не происходит. Если на входе произойдет коллизия, то кадр будет потерян;

- **Коммутация с буферизацией кадров.** При использовании этого метода коммутатор сохраняет в буфере весь кадр, причем во время сохранения кадра коммутатор анализирует его и производит обнаружение ошибок. После этого, убедившись в отсутствии коллизии на входе, он передает кадр;

- **Коммутация с исключением фрагментов.** Коммутация без буферизации кадров обеспечивает малое время задержки. Получив 64-байтовый заголовок кадра, коммутатор занимает требуемый выходной порт и начинает передачу. Обнаружения ошибок при этом не происходит. Коммутация с исключением фрагментов гарантирует, что из источника считывается достаточное число байтов, чтобы обнаружить коллизию до пересылки.

Коммутация с исключением фрагментов – это метод, обеспечивающий компромисс между большим временем задержки с гарантией передачи кадра при коммутации с буферизацией, и небольшим временем задержки с риском потери кадра при коммутации без буферизации кадров. На практике разница между коммутацией с буферизацией и без буферизации кадров неважна, поскольку несущественное уменьшение времени задержки при коммутации без буферизации, возмещается незначительными колебаниями времени ожидания при коммутации с буферизацией.

Для каждого порта коммутатор формирует таблицу MAC-адресов, связанных с сегментом, подключенным к порту коммутатора. Затем коммутатор использует эти MAC-адреса при принятии решения о дальнейших операциях с кадрами: фильтрация, пересылка или лавинная рассылка.

Когда на порт поступает кадр, коммутатор сравнивает MAC-адрес адресата с адресами в таблицах. Если MAC-адрес получателя кадра находится в том же сегменте сети, что и отправитель, коммутатор сбрасывает кадр. Этот процесс называется фильтрацией, и с его помощью коммутаторы могут значительно уменьшить трафик между сегментами сети. Если MAC-адрес получателя кадра находится в другом сегменте, коммутатор пересылает кадр на порт, к которому подключен соответствующий сегмент.

Если у коммутатора нет записи об адресе получателя, то он передаст кадр всем портам, кроме того порта, с которого кадр был получен. Устройство-получатель отвечает на широковещательную рассылку специальным кадром по адресу отправителя. Коммутатор вводит искомый MAC-адрес получателя и

номер соответствующего порта коммутатора в таблицу MAC-адресов. Теперь коммутатор может пересылать кадры между отправителем и получателем без широковещательной рассылки.

Коммутируемые СПД КМД – это самый распространенный в настоящее время тип СПД для локальных сетей. Сегодня цена за порт на коммутаторе уменьшилась на столько, что концентраторы и мосты больше не рассматриваются при принятии решения о покупке сетевого оборудования. Коммутаторы позволяют структурировать трафик, т. е. разбивать его на фрагменты по определенному признаку, чаще всего по физическому расположению абонентских машин пользователей. Такая организация позволяет каждой группе обращаться к устройствам в сети, например серверам, с меньшей вероятностью возникновения коллизий и повышает общую производительность сети.

Маршрутизация по соединяющему дереву (протокол STP).

Основное предназначение STP (Spanning Tree Protocol) – удаление петель в топологии сети и автоматическое управление топологией сети с дублирующими каналами. Действительно, если сетевое оборудование связано для надежности избыточным числом соединений, то без принятия дополнительных мер, кадры будут доставляться получателю в нескольких экземплярах, что приведет к сбоям и возникновению широковещательных штормов (broadcast storm) в случае отправки широковещательных кадров. Следовательно, в каждый момент времени должен быть задействован только один из параллельных каналов, но при этом необходимо иметь возможность переключения при отказах или физическом изменении топологии. С этой задачей может вручную справиться администратор, однако более элегантным и экономичным решением, освобождающим от необходимости круглосуточного мониторинга состояния системы человеком, является использование STP.

Для своей работы STP строит граф, называемый также «деревом», создание которого начинается с корня (root). Корнем становится одно из STP-совместимых устройств, выигравшее выборы. Каждое STP-совместимое устройство (это может быть коммутатор, маршрутизатор или другое оборудование, но для простоты далее мы будем называть такое устройство мостом) при включении считает, что оно является корнем. При этом оно периодически посылает на все свои порты специальные блоки данных – Bridge Protocol Data Units (BPDU). Адрес получателя в пакетах, несущих BPDU, является групповым, что обеспечивает его пропуск неинтеллектуальным оборудованием.

В данном случае под адресом понимается MAC-адрес, так как протокол STP функционирует на уровне управления доступом к среде передачи (Media Access Control, MAC). Из этого также следует, что все дальнейшие рассуждения о STP и его уязвимостях не привязаны к какому-то одному методу передачи, т. е. в равной мере относятся к Ethernet, Token Ring и т. д.

Получив очередной BPDU от другого устройства, мост сравнивает полученные параметры со своими и, в зависимости от результата, перестает или продолжает оспаривать статус корня. В результате корнем становится устройство с наименьшим значением идентификатора моста (Bridge ID). Последний представляет собой комбинацию MAC-адреса и заданного для моста приоритета. Очевидно, что в сети с единственным STP-совместимым устройством оно и будет корнем.

Выбранный корень, или назначенный корневой мост (Designated Root Bridge, в соответствии с терминологией стандарта), не несет никакой дополнительной нагрузки – он всего лишь служит отправной точкой для построения топологии.

Для всех остальных мостов в сети определяется корневой порт (Root Port), т.е. ближайший к корневому мосту порт. От других портов, соединенных с корневым мостом непосредственно или через другие мосты, он отличается своим идентификатором – комбинации из его номера и задаваемым администратором «веса».

На процесс выборов влияет и стоимость пути до корня (Root Path Cost) – она складывается из стоимости пути до корневого порта данного моста и стоимости путей до корневых портов мостов по всему маршруту до корневого моста. Эта стоимость может определяться временем передачи по линку, а также стоимостью портов, через которые происходит передача. Стоимость портов может задаваться вручную администратором.

Помимо выделенного корневого моста в STP вводится логическое понятие назначенного моста (Designated Bridge) – владелец этого статуса считается главным в обслуживании данного сегмента локальной сети. Статус назначенного моста также выборный и может переходить от одного устройства к другому.

Аналогичным образом вводится логическое понятие выборного назначенного порта (Designated Port, он обслуживает данный сегмент сети), а для него – понятие соответствующей стоимости пути (Designated Cost).

После окончания всех выборов наступает фаза стабильности, характеризуемая следующими условиями.

1. В сети только одно устройство считает себя корнем, а остальные периодически анонсируют его как корень.

2. Корневой мост регулярно посылает на все свои порты пакеты с BPDU. Интервал времени, через который происходит рассылка, называется интервалом приветствия (Hello Time).

3. В каждом сегменте сети имеется единственный назначенный порт, через который происходит обмен трафиком с корневым мостом. Он имеет наименьшее значение стоимости пути до корня по сравнению с другими портами в сегменте. При равенстве этой величины в качестве назначенного выбирается порт с наименьшим идентификатором порта (MAC-адрес порта и его приоритет).

4. BPDU принимаются и отправляются STP-совместимым устройством на всех его портах, даже на тех, которые были «отключены» в результате работы STP. Однако BPDU не принимаются на портах, которые были «отключены» администратором.

5. Каждый мост осуществляет пересылку (Forwarding) пакетов только между корневым портом и назначенными портами соответствующих сегментов. Все остальные находятся в блокированном состоянии (Blocking).

Как следует из последнего пункта, STP управляет топологией путем изменения состояния портов, которое может принимать следующие значения:

- **Блокирован (Blocking).** Порт заблокирован, однако, в отличие от пользовательских кадров, кадры с пакетами STP (BPDU) принимаются и обрабатываются;

- **Ожидает (Listening).** Первый этап подготовки к состоянию пересылки. В отличие от пользовательских кадров, кадры с пакетами STP (BPDU) принимаются и обрабатываются. Обучения не происходит, так как в этот период в таблицу коммутации может попасть недостоверная информация;

- **Обучается (Learning).** Второй этап подготовки к состоянию пересылки. Кадры с пакетами STP (BPDU) принимаются и обрабатываются, а пользовательские кадры мост принимает для построения таблицы коммутации, но не пересылает данные;

- **Передает (Forwarding).** Рабочее состояние портов, когда передаются как кадры с пакетами STP, так и кадры пользовательских протоколов.

- **Отключен (Disabled).** Порт не предпринимает никакой активности.

43 Сетевой уровень: проблемы построения сетевого уровня. Алгоритмы маршрутизации: иерархическая маршрутизация, маршрутизация при вещании, групповая маршрутизация.

Сервис сетевого уровня должен удовлетворять следующим требованиям:

- быть независимым от технологии передачи, используемой в системе передачи данных, топологии и каких-либо других параметров СПД;

- транспортный уровень не должен зависеть от числа узлов и топологии транспортной среды.

Проблемы:

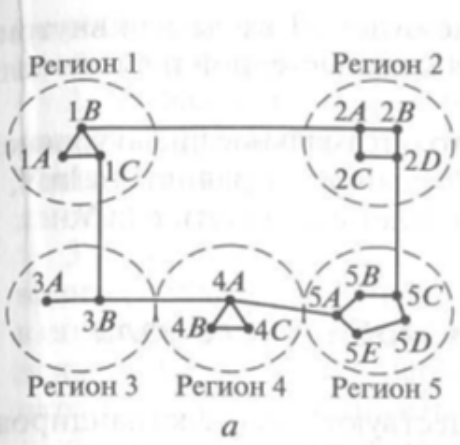
1. Где поместить основную вычислительную сложность: сервис, ориентированный на соединение, предполагает, что эта сложность должна приходиться на сетевой уровень (маршрутизаторы и СПД), а сервис без соединений – на транспортный уровень (абонентская машина или хост). Сторонникам сервиса, ориентированного на соединение, утверждают, что лучше использовать надёжный сервис, который могут предоставить соединения на сетевом уровне (многие приложения легче связывать с соединениями на сетевом уровне, чем с сетевым уровнем, без соединений). Защитники сервиса без соединений говорят, что стоимость вычислительных средств падает, а их мощность растёт, следовательно, нет причин не нагружать хост, в то же время, транспортная среда должна оставаться неизменной как можно дольше (для многих приложений скорость доставки важнее, чем её аккуратность).

2. Внутренняя организация сетевого уровня может быть ориентированной на соединение (виртуальный канал) и не ориентированном на соединении (о пакетах говорят как о дейтаграммах). Идея виртуального канала – избежать маршрутизации для каждого пакета: маршрут выбирается один раз при установлении виртуального канала между отправителем и получателем и не изменяется до конца передачи. Транспортная среда запоминает выбранный маршрут. После окончания передачи виртуальный канал уничтожается. При выборе сервиса без соединения каждый пакет маршрутизируется независимо (разные пакеты могут следовать разными маршрутами), вследствие такой организации транспортная среда становится более надёжна, так как способна гибко реагировать на ошибки и перегрузки, хотя для продвижения по разным маршрутам может потребоваться различное время. При использовании виртуальных каналов их поддержка полностью обеспечивается транспортной средой, а в случае использования дейтаграмм никакой таблицы виртуальных каналов не требуется (у них есть таблица с указаниями какую линию надо использовать, чтобы доставить пакет по заданному адресу, хотя эта таблица необходима при использовании виртуальных каналов, когда устанавливается соединение), но каждая дейтаграмма должна иметь полный адрес доставки.

3. Выбор внутренней организации транспортной среды требует компромисса между пропускной способностью и памятью маршрутизатора, между временем установки соединения и временем разбора адреса доставки. Виртуальные каналы имеют преимущества в борьбе с перегрузками, но слабоустойчивы к сбоям.

Иерархическая маршрутизация – маршрутизация с иерархией сетей подобной иерархии коммутаторов в телефонной сети.

Пример:



а

Назначение	Линия	Переходы
1A	—	—
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1B	5
5D	1C	5
5E	1C	5

б

Назначение	Линия	Переходы
1A	—	—
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

в

Рис. 2.12. Иерархическая маршрутизация:

а — сеть с двухуровневой иерархией из пяти регионов; *б* — полная таблица маршрутизации; *в* — иерархическая таблица

На рис. 2.12, *а* приведен пример сети с двухуровневой иерархией из пяти регионов, где каждый из регионов соединен с двумя другими регионами. На рис. 2.12, *б* показана полная таблица маршрутизации для маршрутизатора 1*A* без иерархии, а на рис. 2.12, *в* — та же таблица при двухуровневой иерархии. Нетрудно увидеть, что таблица маршрутизации во втором случае резко сократилась. Однако за эту экономию приходится платить эффективностью маршрутизации. Например, наилучший маршрут от 1*A* к 5*C* проходит через регион 2, но в данном случае он пройдет через регион 3, поскольку большинство машин в регионе 5 могут быть эффективнее достигнуты через регион 3.

Клейнрок и Камоун показали, что оптимальное число уровней иерархии в транспортной среде при N узлах будет равняться: lnN, а число строк в таблице маршрутизатора будет составлять: $elnN$.

Маршрутизация при вещании.

В некоторых приложениях возникает потребность пересылки одного и того же сообщения всем машинам, такой режим передачи называется вещанием. Есть несколько реализаций этого режима:

1. Источник знает, кому и что надо послать, и генерирует столько сообщений, сколько имеется получателей.

2. Использование метода лавины.

3. Маршрутизация множественной доставки: каждый пакет должен иметь либо лист рассылки (содержит адреса получателей), либо маршрут рассылки (содержит последовательность маршрутизаторов).

4. Использование дерево захода либо любого другого подходящего дерева связей.

5. Пересылка вдоль обратного пути (неявное использование дерева связей): при поступлении пакета маршрутизатор проверяет по какой линии он поступил, если пакет 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 пришёл, если пакет поступил не по этой линии, то пакет сбрасывается.

Маршрутизация при групповой передаче.

Групповая передача используется, когда надо обеспечить взаимодействие группы взаимосвязанных процессов, разбросанных по сети. Алгоритм групповой маршрутизации основывается на дереве связей (каждый маршрутизатор в транспортной среде вычисляет дерево связей, охватывающее все остальные маршрутизаторы). Самый простой алгоритм для редукции дерева связей основывается на применении алгоритма маршрутизации по состоянию канала (каждый маршрутизатор имеет полную конфигурацию транспортной среды). Редукция дерева связей начинается от вершин (члены группы) и разворачивается в сторону корня, удаляя все маршрутизаторы, которые не ведут к членам группы.

44 Сетевой уровень в Интернет: адресация, протокол IPv4, протоколы ARP, RARP, DHCP.

Каждая машина в Internet имеет уникальный IP адрес. Он состоит из адреса сети и адреса машины в этой сети. Все IP адреса имеют длину 32 разряда. 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 000 машин, C – 2 миллиона сетей по 256 машин, D – групповая передача, E зарезервирован для развития. Адреса выделяет только NIC – Network Information Center.

Подсети.

Все машины одной сети должны иметь одинаковый номер сети в адресе. По мере роста сети приходиться менять класс адреса. Перенос машины из одной сети в другую требует изменения маршрутизации. 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. Подсеть – это часть сети, не видимая извне. Изменение адреса подсети или введение новой подсети не требует обращения в NIC или изменений какой-либо глобальной базы данных.

Адресация.

Есть две таблицы. Первая показывает как достичь интересующей сети. Вторая – как достичь узел внутри сети. Когда поступает IP пакет, маршрутизатор ищет его адрес доставки в таблице маршрутизации. Если это 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. 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, маршруизатор подсети в данной локальной сети знает, как достичь любую подсеть в данной локальной сети, и как найти конкретную машину в своей подсети. Все что ему нужно – это знать маску подсети. С помощью логической операции И маршрутизатор выделяет адрес подсети с помощью маски. По своим таблицам он определяет как достичь нужной подсети или, если этого локальная подсеть данного маршрутизатора, как достичь конкретной машины.

Протоколы управления межсетевым взаимодействием.

Internet Control Message Protocol.

Управление функционированием Internet происходит через маршрутизаторы с помощью протокола ICMP. Этот протокол выявляет и рассылает сообщения о десятках событий.

Address Resolution Protocol – протокол определения адреса.

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 на адреса канального уровня, например, Ethernet адрес. Для отображения IP адреса на 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 Internet имеет этот протокол. Для того чтобы узнать адрес в другой сети есть 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.

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

Иногда возникает обратная проблема – известен Ethernet адрес, какой IP адрес ему соответствует. Он 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 запросов рассылаются всем, увеличивает накладные расходы. Для устранения этого недостатка был предложен протокол BOOTP. В отличии от RARP, BOOTP использует UDP сообщения, которые рассылаются только маршрутизаторам

DHCP (англ. Dynamic Host Configuration Protocol – протокол динамической настройки узла) – сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Данный протокол работает по модели «клиент-сервер». Для автоматической конфигурации компьютер-клиент на этапе конфигурации сетевого устройства обращается к так называемому серверу DHCP, и получает от него нужные параметры. Сетевой администратор может задать диапазон адресов, распределяемых сервером среди компьютеров. Это позволяет избежать ручной настройки компьютеров сети и уменьшает количество ошибок. Протокол DHCP используется в большинстве сетей TCP/IP.

Протокол DHCP предоставляет три способа распределения IP-адресов:

- **Ручное распределение.** При этом способе сетевой администратор сопоставляет аппаратному адресу (для Ethernet сетей это MAC-адрес) каждого клиентского компьютера определённый IP-адрес. Фактически, данный способ распределения адресов отличается от ручной настройки каждого компьютера лишь тем, что сведения об адресах хранятся централизованно (на сервере DHCP), и потому их проще изменять при необходимости.

- **Автоматическое распределение.** При данном способе каждому компьютеру на постоянное использование выделяется произвольный свободный IP-адрес из определённого администратором диапазона.

- **Динамическое распределение.** Этот способ аналогичен автоматическому распределению, за исключением того, что адрес выдаётся компьютеру не на постоянное пользование, а на определённый срок. Это называется арендой адреса. По истечении срока аренды IP-адрес вновь считается свободным, и клиент обязан запросить новый (он, впрочем, может оказаться тем же самым). Кроме того, клиент сам может отказаться от полученного адреса.

Некоторые реализации службы DHCP способны автоматически обновлять записи DNS, соответствующие клиентским компьютерам, при выделении им новых адресов. Это производится при помощи протокола обновления DNS, описанного в RFC 2136.

Протокол DHCP является клиент-серверным, то есть в его работе участвуют клиент DHCP и сервер DHCP. Передача данных производится при помощи протокола UDP, при этом сервер принимает сообщения от клиентов на порт 67 и отправляет сообщения клиентам на порт 68.

45 Бесклассовая адресация на сетевом уровне. Сетевая маска переменной длины и суммирование адресов.

Бесклассовая адресация.

Популярность Интернета обернулась против него: не стало хватать адресов. Проблема состоит в том, что адреса выделяются классами, при этом многие организации не используют всего диапазона адресов выделенного им класса. Проблемой также является взрывообразный рост таблиц маршрутизации. Маршрутизатор не должен знать о каждой машине в сети, но должен знать о каждой сети. Кроме того, многие из алгоритмов маршрутизации требуют, чтобы маршрутизаторы периодически обменивались своими таблицами (чем больше таблицы, тем больше шансов допустить ошибки при передаче). Протокол **CIDR** (Classless InterDomain Routing) основан на идее, что сети класса С почти не использовались, а их более 2 млн. Поэтому по запросу организации можно выделить несколько последовательных сетей этого класса таким образом, чтобы охватить требуемое число машин. Например, если организация подаёт заявку на 2000 машин, можно выделить ей восемь последовательных сетей класса С, т.е. 2048 машин.

При этом мир был поделён на 4 зоны:

194.0.0.0 ... 195.255.255.255 — Европа;
198.0.0.0 ... 199.255.255.255 — Северная Америка;
200.0.0.0 ... 201.255.255.255 — Центральная и Южная Америка;
202.0.0.0 ... 203.255.255.255 — Азия и Тихий Океан.

Таким образом, каждая зона получила 32 млн. адресов для раздачи пользователям, 320 млн. адресов класса С были зарезервированы на будущее.

46 Сетевой уровень в Интернет: адресация, протокол IPv6.

Из-за того что распределение адресов ipv4 становится невозможным связи с ростом числа машин. Была поставлена задача придумать новый протокол такой, чтобы он удовлетворял требованиям:

1. Работа с миллиардами машин, даже при не эффективном распределении адресов;
2. Сократить размер таблиц маршрутизации;
3. Упростить протоколы, чтобы сделать маршрутизацию быстрее;
4. Обеспечить более высокую безопасность, чем существующий IP;
5. Обратить больше внимания на тип сервиса, особенно для приложений реального времени;
6. Расширить групповую адресацию, разрешив описание группы;
7. Разрешить роуминг для хоста без изменения его адреса;
8. Позволить эволюцию протоколов в будущем;
9. Разрешить совместное существование как старых таки новых протоколов.

В 1993 году был опубликован протокол **SIPP** – Simple Internet Protocol Plus, который был принят как IPv6.

IPv6 имеет механизмы совместимости с IPv4:

- **Двойной стек.** На каждом узле сети, который работает с IPv6 и которому требуется взаимодействие с IPv4-хостами, устанавливается стек протокола IPv4, и ему выделяется IPv4-адрес. Естественно, после этого данный узел может «общаться» с другими хостами, работающими с разными версиями протокола IP. Главным преимуществом принципа двойного стека является относительная простота. К сожалению, недостатков у него гораздо больше.

- **Application Level Gateway.** Application Level Gateway (ALG) переводится с английского как «шлюз прикладного уровня». Этот механизм предполагает, что для каждого приложения, работающего с Интернетом, создается специальная утилита, которая преобразовывает весь входящий трафик этого приложения из IPv4 в IPv6, а весь исходящий – из IPv6 в IPv4. Таким образом, сам узел работает по новой версии протокола, а весь трафик передается по старой.

- **Туннелирование.** Механизм туннелирования используется для частичного решения проблемы совместимости. Он не может применяться для связи IPv6-узлов с IPv4-хостами. Туннелирование предназначено для организации связи между IPv6-узлами или сетями посредством существующей среды передачи данных, поддерживающей только версию протокола IPv4. Суть этого механизма заключается в следующем. Между сетями или хостами, работающими с IPv6, с помощью специального программного обеспечения создается «туннель». Пакеты информации, попадая на один конец этого «туннеля», преобразовываются. Это преобразование заключается в инкапсулировании IPv6-пакетов в пакеты стандарта IPv4, которые затем пересылаются на «выход». На втором конце «туннеля» происходит обратный процесс. Из IPv4-пакетов извлекаются пакеты стандарта IPv6, которые затем обрабатываются маршрутизаторами как обычно.

- **Транслятор IP/ICMP.** Это самый современный механизм, обеспечивающий как совместимость между IPv4 и IPv6-узлами, так и связь между собой IPv6-сетей посредством среды передачи данных IPv4. В нем используются специальные типы адресов (0000:0000:0000:0000:0000:0000:XXXX:XXXX и 0000:0000:0000:0000:0000:FFFF:XXXX:XXXX), которые есть в новой версии протокола. Для реализации этого механизма необходимо специальное программное обеспечение – транслятор. Он устанавливается на границе IPv6-сети. Весь приходящий извне трафик в формате IPv4 принимается именно этим транслятором, который преобразовывает его согласно стандарту IPv6 и отправляет узлам сети, каждому из которых присвоен IPv4-транслированный адрес. То же самое происходит и с исходящим трафиком. Он проходит через транслятор, который преобразовывает его согласно протоколу IPv4 и отправляет адресатам.

Отличия IPv6:

1. Более длинный адрес – 16 байт. Это решает одну из главных задач – неограниченное расширение Internet.

2. Заголовок стал проще (всего 7 полей), что ускорило обработку и маршрутизацию.

3. Он лучше поддерживает варианты в заголовке, что делает работу с ним более гибкой, позволяя опускать не нужные поля и вводить необходимые.

4. Серьезно улучшена безопасность протокола. Идентификация и конфиденциальность - ключевые возможности нового IP.

5. Существенно улучшена работа с типом сервиса, особенно учитывая возрастающий multimedia трафик.

Существуют различные типы адресов IPv6: **одноадресные** (Unicast), **групповые** (Anycast) и **многоадресные** (Multicast).

Адреса типа Unicast хорошо всем известны. Пакет, посланный на такой адрес, достигает в точности интерфейса, который этому адресу соответствует.

Адреса типа Anycast синтаксически неотличимы от адресов Unicast, но они адресуют группу интерфейсов. Пакет, направленный такому адресу, попадёт в ближайший (согласно метрике маршрутизатора) интерфейс. Адреса Anycast могут использоваться только маршрутизаторами.

Адреса типа Multicast идентифицируют группу интерфейсов. Пакет, посланный на такой адрес, достигнет всех интерфейсов, привязанных к группе многоадресного вещания.

47 Транспортный уровень: сервис, примитивы, адресация, установление соединения, разрыв соединения, управление потоком и буферизация, восстановление последовательности сегментов.

Транспортный протокол обеспечивает надежную передачу данных от одного абонента в сети другому.

Сервис для верхних уровней.

Основная цель транспортного уровня – обеспечить эффективный, надежный и дешевый сервис для пользователей на прикладном уровне. Достижение этой цели обеспечивает сервис, предоставляемый сетевым уровнем. То, что выполняет работу транспортного уровня, называется транспортным агентом. Подобно сетевому уровню, транспортный уровень так же может поддерживать два вида сервиса, ориентированный на соединения и без соединений. 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. Идея транспортного уровня в том, чтобы сделать сервис транспортного уровня более надежным, чем сетевого. Другое важное соображение в том, что прикладная программа, опираясь на транспортный сервис, становится независимой от сети и может работать в сети с любым сервисом.

Качество сервиса.

Транспортный уровень позволяет пользователю определить желаемые, допустимые и минимальные значения для различных параметров, характеризующих качество сервиса, в момент установки соединения. Далее транспортный уровень сам будет решать, сможет ли он с помощью сетевого сервиса удовлетворить запросы пользователя и до какой степени.

Примеры параметров качества:

- **Connection establishment delay** – задержка на установку соединения определяет время между запросом на установку соединения и подтверждением о его установлении.
- **Connection establishment failure probability** – вероятность что соединение не будет установлено за время, равное задержке на установку соединения.
- **Throughput** – пропускная способность транспортного соединения определяет количество байт пользователя, передаваемых за секунду.
- **Transit delay** – задержка на передачу определяет время от момента, когда сообщение ушло с машины отправителя, до момента, когда оно получено машиной получателем.
- **Residual error ration** – доля ошибок при передаче. Теоретически этот параметр должен быть равен 0, если транспортный уровень надежно передает сообщение. На практике это не так.
- **Protection** – этот параметр позволяет определить уровень защиты передаваемых данных от не санкционированного доступа третьей стороной.
- **Priority** – приоритет позволяет пользователю указать степень важности для него разных соединений.
- **Resilience** – устойчивость определяет вероятность разрыва транспортным уровнем соединения в силу своих внутренних проблем или перегрузки.

Если требуемое качество недостижимо, то транспортный уровень сразу сообщает об этом пользователю, даже не обращаясь к получателю сообщения.

Примитивы транспортного уровня.

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 пакет с запросом на установление соединения серверу.

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 сторона больше данных не будет. На рис.6-5 показана диаграмма состояний при установлении и разрыве соединения.

Другой набор примитивов, так называемые, сокеты Беркли. Примитив SOCET создает новую точку подключения, резервирует для нее место в таблице транспортного агента. По примитиву BIND сервер выделяет сокету адрес. 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.

Элементы транспортного протокола.

Транспортный протокол должен решать следующие проблемы:

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

Адресация.

Проблема адресации состоит в том, как указать с каким удаленным прикладным процессом надо установить соединение? Для этого используется TSAP – Transport Service Access Point. Аналогичное понятие существует и на сетевом уровне – IP адрес – SAP для сетевого уровня.

Это решение хорошо работает для часто используемого сервиса с длительным периодом активности, но в случае с временными приложениями нужно как-то запустить их и узнать их TSAP. Для этого используется протокол установления начального соединения. На каждой машине есть специальный сервер процессов, который как бы 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.

Другой пример использовать сервер имён. 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.

Установление соединения.

Пакеты могут теряться, храниться и дублироваться на сетевом уровне. И из-за задержек приходить позже. Одно из возможных решений – временное TSAP. После того, как оно использовано, TSAP с таким адресом более не возникает. Другое решение – каждому транспортному соединению сопоставлять

уникальный номер. Когда соединение разрывается, этот номер заносится в специальный список. Другой подход – ограничить время жизни пакетов.

На практике нам надо обеспечить, чтобы умерли не только сами пакеты, но и уведомления о них.

Безопасный способ установления соединения. Все машины в сети оснащены таймерами. Каждый таймер двоичный счетчик достаточно большой разрядности, равной или превосходящей разрядность последовательных чисел, используемых для нумерации пакетов. При установлении соединения несколько младших разрядов этого таймера берут 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 Для того, чтобы избежать повторного использования порядкового номера, который уже был сгенерирован перед сбоем машины, вводится специальная величина по времени, которая образует запрещенную область последовательных номеров.

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.

Троекратное рукопожатие 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.

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.

Управление потоком и буферизация.

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. Если - по минимальной, 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 лучше всего решать вопрос о количестве буферов динамически. Здесь надо только позаботиться о решении проблемы потери управляющих пакетов.

Другую проблему представляет согласование доступного числа буферов и пропускная способность сетевого уровня. 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 должен на стороне отправителя, чтобы предотвращать накопление большого числа не подтвержденных сообщений.

Мультиплексирование.

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. Такое мультиплексирование называется восходящим.

Восстановление после сбоев.

Надо восстанавливать работоспособность машины, включая и транспортный уровень. Предположим, сервер упал и старается восстановить функционирование. Прежде всего, ему надо узнать у клиента, какое TPDU было последним не подтвержденным и попросить перепослать его. В свою очередь клиент может находиться в одном из двух состояний: S1 – есть не подтвержденное TPDU, либо S0 – все TPDU подтверждены.

Если сервер упал, послав подтверждение, но до того, как он осуществил запись, то клиент будет находится после восстановления сервера в состоянии S0, хотя подтвержденное TPDU потеряно. Пусть наоборот сервер сначала записал TPDU, а потом упал. Тогда сервер после сбоя найдет клиента в состоянии S1 и решит, что надо перепослать не подтвержденное TPDU. В результате получим повторное TPDU.

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.

48 Транспортный уровень в Интернет (TCP, UDP). Сервис TCP, протокол, заголовок сегмента, управление соединениями, стратегия передачи, управление перегрузками, управление таймерами. Протокол UDP.

TCP – ориентированный на соединение, **UDP** – не ориентированный на соединение.

TCP (Transmission Control Protocol) – специально созданный протокол для надежной передачи потока байтов по соединению «точка-точка» через ненадежную сеть. 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.

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.

Порты с номерами до 256 зарезервированы для стандартного сервиса (FTP, SSH и т.д.). Все TCP-соединения – дуплексные, т.е. передача идет независимо в оба направления. TCP-соединение поддерживает только соединение «точка-точка».

TCP обеспечивает поток байтов, а не поток сообщений.

После того, как приложение передало данные TCP агенту, эти данные могут быть отправлены сразу на сетевой уровень, а могут быть буферизованы, в заголовке TCP-пакета есть флаг PUSH. Если он установлен, то это говорит о том, что данные должны быть переданы немедленно. 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.

Каждый байт в TCP соединении имеет 32-разрядный номер. TCP-агенты обмениваются сегментами данных. Каждый сегмент имеет заголовок от 20 байтов и более (по выбору) и тело переменной длины.

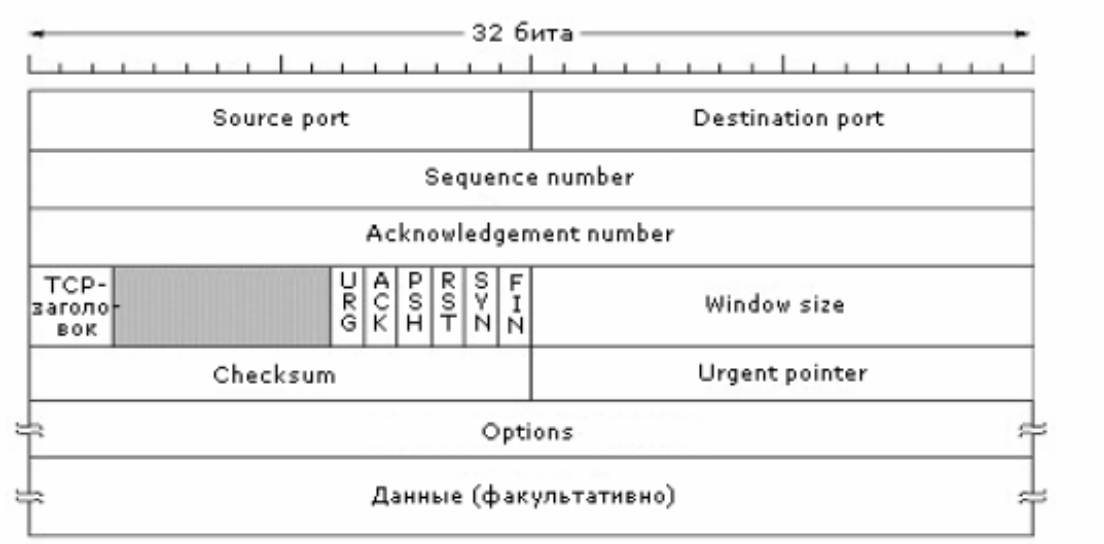
длина сегмента не должна превышать максимальную длину IP-пакета – 64 Кбайт. Во-вторых, каждая сеть имеет максимальную единицу передачи – **MTU** (maximum transfer unit), и каждый сегмент должен помещаться в MTU.

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.

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.

Заголовок TCP.



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.

Биты:

- **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.

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.

Таймер для последовательных номеров сегментов тактируется с частотой 4 мксек., максимальное время жизни пакета - 120 сек.

Перегрузки.

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.

В основе используемого в протоколе TCP алгоритма выбора времени таймаута, 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 хитрой формуле.

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.

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.

Заголовок UDP.

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).

Сервис, реализуемый протоколом UDP – это практически сервис, реализуемый протоколом IP, с добавлением небольшого заголовка.

49 Безопасность и способы защиты данных в сетях ЭВМ: методы шифрования. Рассеивание и перемешивание - два основных принципа шифрования. Алгоритмы с секретными ключами.

Безопасность информации – это состояние информации, характеризующееся ее защищенностью от внутренних и внешних угроз, т.е. от нарушения конфиденциальности, целостности, доступности, а также от незаконного тиражирования, которые наносят материальный и моральный ущерб владельцу или пользователю этой информации.

Угроза безопасности информации – это потенциально возможное преднамеренное или непреднамеренное происшествие, которое может оказать нежелательное воздействие на саму систему в сети ЭВМ, а также привести к потере безопасности информации, хранящейся в ней.

Уязвимость сети ЭВМ – это некая характеристика сети, которая делает возможным возникновение угрозы безопасности информации. **Атака на сеть ЭВМ** – это действие, предпринимаемое злоумышленником, заключающееся в поиске и использовании уязвимости сети.

Три группы проблем:

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 и документов**.
3. **Надежность управления или доступность ресурсов и сетевых услуг:**
   - несанкционированное использование ресурсов (если вы получите счет за телефонные переговоры, которые вы не вели, вам это вряд ли понравится);
   - обеспечение доступности ресурсов для авторизованных пользователей.

Прежде чем обсуждать проблемы обеспечения безопасности следует рассмотреть основные виды работы с информацией в сети: передачу, обработку, хранение и представление.

В каком месте стека протоколов должна располагаться защита информации в сети? Одного такого места не существует: защита возможна на каждом уровне сети.

Например, на физическом уровне для контроля доступа к каналу можно поместить кабель в опечатанную трубу, заполненную газом под давлением. Тогда любая попытка просверлить эту трубу приведет к падению давления газа, срабатыванию датчика давления и включению сигнала тревоги. На канальном

уровне данные могут быть зашифрованы на одной машине, а расшифрованы на другой, и об этом шифре верхние уровни могут ничего не знать – шифрование канала, часто применяется в сетях.

На сетевом уровне распространенным решением является наличие брандмауера.

На транспортном уровне проблему секретности данных при передаче решают шифрованием всех сегментов транспортного соединения и применением так называемых сеансовых шлюзов.

Обычное шифрование. Исходный текст, называемый также открытым текстом, обрабатывают по определенному алгоритму со специальным параметром (ключом). При этом сам алгоритм шифрования может быть хорошо известен, а менять требуется только ключи. В результате этой обработки получают шифр-текст, или криптограмму. Так как у Злоумышленника нет ключа, быстро прочесть сообщение он не может, для этого ему необходимо вскрыть шифр, т.е. узнать алгоритм и ключ, использованные при шифровании этого сообщения.

Как проверить устойчивость алгоритма шифрования к взлому? Для этого алгоритм публикуют. Публикуя алгоритм шифрования, его автор даром получает консультации многих исследователей в этой области. Если ни один из них в течение пяти лет не объявит, что он вскрыл алгоритм, то такой алгоритм можно считать вполне надежным.

Активный злоумышленник – не только копирует сообщение, но и отправляет свои сообщения, имитируя настоящего отправителя. Искусство создания шифра называют **криптографией**, а искусство его вскрытия – **криптоанализом**. Вместе эти дисциплины образуют **криптологию**.

Шифрование замещением состоит в том, что буква или группа букв замещается другой буквой или группой букв из того же самого либо из другого алфавита (моноалфавитное и полиалфавитное замещение). **Шифр Юлия Цезаря** – замена каждой буквы в слове третьей буквой, следующей за ней в алфавите: а $\rightarrow$ г, б $\rightarrow$ д, и т.д. Это так называемое моноалфавитное замещение, где ключом является 33-буквенная строка, соответствующая алфавиту. Здесь возможны 33! ключа. Даже если на проверку одного ключа компьютер будет тратить 1 мкс, то на расшифровку уйдет около 1013 лет.

Алфавитов шифрограммы может быть несколько, и они могут изменяться по определенному правилу, зависящему от ключа. Чтобы прочесть сообщение, необязательно тупо перебирать все возможные варианты ключей. Найти необходимый ключ быстрее можно, используя знание частотных характеристик 151 языка: частоты встречаемости отдельных букв, двухбуквенных буквосочетаний, трехбуквенных сочетаний и т.д. Для этого надо подсчитать частоту букв в шифр-тексте и попытаться сопоставить наиболее часто встречающимся буквы в шифре с буквами, наиболее часто встречающимися в языке. Затем найти устойчивые буквосочетания и т.д. Следовательно, здесь большое значение имеют дополнительные сведения о шифрограмме: на каком языке написано исходное сообщение, его длина, типичные приветствия в данном языке и т.д. Чем длиннее сообщение, тем представительнее будет выборка для его анализа по частоте встречаемости букв и буквосочетаний.

Шифрование перестановкой состоит в изменении порядка набора букв без изменения самих букв. Для примера рассмотрим метод шифрования по столбцам. Выбираем ключ – последовательность неповторяющихся символов, которые нумеруем в соответствии с их местом в алфавите. Шифруемый текст размещается по строкам. Длина строки – длина ключа. В результате получаем массив, где столбцы нумеруются в соответствии с номером символа в ключе. Каждому столбцу соответствует символ ключа, который имеет определенный номер. Упорядочим столбцы по возрастанию этих номеров: сначала выпишем все символы первого столбца, затем символы второго столбца и т.д.

Для раскрытия этого шифра криптоаналитик прежде всего должен убедиться, что имеет дело с шифрованием перестановкой, Для этого он должен подсчитать частоту встречаемости букв в шифре, и если она соответствует частотным характеристикам языка, то это означает, что это именно метод перестановки. Намек на порядок столбцов могут дать устойчивые буквосочетания, имеющиеся в языке.

Алгоритмы с секретными ключами.

Если раньше алгоритм был простой, а вся сложность шифрования заключалась в ключе, то теперь, наоборот, стараются алгоритм делать как можно изощреннее, чтобы криптоаналитик, получив как угодно много зашифрованного текста, не смог из него ничего извлечь.

Все алгоритмы шифрования с позиции использования ключа подразделяются на **алгоритмы с секретным ключом** и **алгоритмы с открытым ключом**. Алгоритмы с секретным ключом используют один

ключ и для шифрования, и для дешифрования, поэтому их часто называют симметричными алгоритмами шифрования. Этот ключ является строжайшим секретом, известным только тому, кто шифрует сообщение, и тому, кто это сообщение расшифровывает. Слабым местом этих шифров является этап установки общего секретного ключа.

DES.

Одним из широко известных шифров с секретным ключом является шифр DES (Data Encryption Standard). Создан на базе разработки фирмы IBM, был принят как стандарт в области шифрования в январе 1977 г. правительством США. Алгоритм DES состоит из 19 этапов. На первом этапе исходный текст разбивается на блоки по 64 бит каждый, и над каждым блоком выполняется перестановка. Последний этап является инверсией первой перестановки. Предпоследний этап заключается в обмене местами 32 самых левых битов и 32 самых правых битов. Для этапов со 2-го по 17-й с помощью специального преобразования исходного 56-разрядного ключа строятся 16 частных ключей, которые используются для преобразования данных. У алгоритма DES имеется два недостатка. Во-первых, он представляет собой моноалфавитное замещение с 64-разрядным символом, а всегда при подаче одних и тех же 64 разрядов исходного текста на вход, те же самые 64 разряда получают на выходе. DES сохраняет структуру сообщения, т. е. одни и те же поля исходного текста попадут в одни и те же места шифр-текста, чем может воспользоваться злоумышленник. Зная структуру исходного сообщения и длину его полей, он просто переставит необходимые поля в шифр-тексте, чтобы несанкционировано изменить сообщение. Во-вторых, для начала шифрования надо иметь сразу весь 64-разрядный блок исходного текста, а это не совсем удобно, если речь идет об интерактивных приложениях. Кроме того, эти 64 разряда надо накапливать в открытом виде, что делает схему шифрования уязвимой. Тройное шифрование (EDE-схема).

AES.

Шифруемые данные представляют в виде двухмерных байтовых массивов размером 4 на 4 байт. Все операции производятся над отдельными байтами массива, независимо над столбцами и строками.

На каждой итерации алгоритма выполняются следующие преобразования массива:

1. **Операция Sub Bytes**, представляющая собой замену каждого байта массива данных в соответствии со специальной таблицей кодирования.

2. **Операция Shift Rows**, представляющая собой циклический сдвиг влево всех строк массива данных за исключением нулевой. Сдвиг i-й строки массива (для i = 1, 2, 3) производится на i байт.

3. **Операция MixColumns**, выполняющая умножение каждого столбца массива данных, который рассматривается как полином степени 2^8, на фиксированный полином $(x) = x^3+x^2+x+2$. Умножение выполняется по модулю $x^4 + 1$.

4. **Операция Add Round Key**, преобразующая массив данных с расширенным ключом итерации (наложение ключа).

Процедура получения расширенного ключа для каждой итерации такова: над i-м столбцом массива данных (i = 0...3) побитово выполняется логическая операция XOR с расширенным ключом W_{4r+I}, где r – номер текущей итерации алгоритма, начиная с 1. Количество итераций R алгоритма зависит от длины ключа. Перед первой итерацией алгоритма AES выполняется предварительное наложение расширенного ключа $W_0...W_3$ на открытый текст первых четырех слов итерации с помощью операции AddRoundKey. Последняя итерация отличается от предыдущих тем, что в ней нет операции MixColumns.

В алгоритме AES используются ключи шифрования трех фиксированных размеров: 128, 192, и 256 бит. Цель процедуры расширения ключа состоит в формировании необходимого числа слов расширенного ключа для их использования в операции AddRoundKey. Дешифрация выполняется посредством применения обратных операций в обратной последовательности.

50 Безопасность и способы защиты данных в сетях ЭВМ: методы шифрования. Рассеивание и перемешивание - два основных принципа шифрования. Алгоритмы с открытыми ключами.

См. предыдущий билет.

Алгоритмы с открытыми ключами.
Пусть имеются алгоритмы шифрования E и дешифрования D которые удовлетворяют следующим требованиям:

- D(E(P)) равно исходному тексту P,
- чрезвычайно трудно получить D, зная E,
- E нельзя вскрыть через анализ исходных текстов.

Алгоритм шифрования E и его ключ (**открытый ключ**), публикуют или помещают таким образом, чтобы каждый мог их получить. Алгоритм D также публикуют, чтобы подвергнуть его изучению и проверке на стойкость, а вот ключ к нему хранят в секрете. Это **секретный (или закрытый) ключ**.

Взаимодействие двух абонентов A и B происходит следующим образом. Пусть A хочет послать B текст P. Абонент A шифрует текст $E_B(P)$, зная алгоритм и открытый ключ абонента B для шифрования. Абонент B, получив текст $E_B(P)$ и использовав секретный ключ и алгоритм D_B, вычисляет $D_B(E_B(P)) = P$. Никто не прочтет текст P кроме абонентов A и B, так как по условию алгоритм E_B нераскрываем, а алгоритм D_B нельзя вывести из E_B.

Пример: алгоритм RSA.
Общая схема этого алгоритма следующая:

1. Выберем два больших (больше 10^100) простых числа p и q.
2. Вычислим n = p * q и z = (p - 1) (q - 1).
3. Выберем простое d, взаимно простое по отношению к z.
4. Найдем e, удовлетворяющее условию (e * d) mod z = 1.

Разобьем исходный текст на блоки длиной p таким образом, чтобы каждый блок как число не превосходил n. Для этого выберем наибольшее k, при котором выполняется условие $p = 2^k < n$. Шифр сообщения p получим, вычислив шифрованный текст $C = -p^e(modn)$. Для расшифровки найдем $P = C^d(modn)$.

Для шифрования требуются числа e, n, представляющие собой открытый ключ, а для дешифрования – числа d, n – закрытый ключ.

Безопасность, обеспечиваемая применением этого метода шифрования, основана на высокой вычислительной сложности операции разложения на простые множители больших чисел. Один из основных недостатков алгоритма RSA – медленная работа.

51 Информационная безопасность: основные задачи. Протоколы установления подлинности на основе закрытого ключа, протокол Деффи-Хелмана. Электронная подпись. Профиль сообщения.

Протоколы установления подлинности (аутентификации) позволяют процессу убедиться, что он взаимодействует с тем, с кем должен, а не с тем, кто лишь представляется таковым. Авторизация — проверка прав на выполнение тех или иных операций.

Если, например, к серверу обратился процесс с запросом удалить файл x.dat и объявил себя процессом «Вася», то сервер должен убедиться, что перед ним действительно Вася (аутентификация) и что Вася имеет право делать то, что просит (авторизация).

Общая схема всех протоколов аутентификации следующая: сторона A и сторона B начинают обмениваться сообщениями между собой или с центром раздачи ключей (ЦРК). При этом предполагается, что ЦРК — всегда надежный партнер, т.е. его нельзя фальсифицировать. Протокол аутентификации должен быть устроен таким образом, чтобы даже в случае перехвата злоумышленником сообщения между A и B, ни A, ни B не спутают друг друга со злоумышленником.

Аутентификация на основе закрытого разделяемого ключа.

Протокол ответа по вызову: одна сторона посылает некоторое число (вызов), а другая сторона, получив это число, преобразует его по определенному алгоритму и отправляет обратно. Увидев результат преобразования и зная исходное число, инициатор может определить, правильно сделано преобразование или нет. Алгоритм преобразования является общим секретом взаимодействующих сторон.

Идея атаки состоит в том, чтобы «заставить» Петю дать некоторое число, после чего на третьем шаге подсунуть ему это же число как свой вызов. На этот вызов Петя, согласно протоколу, ответит преобразованным ключом. В результате злоумышленник получит и число, и ключ, что ему и надо, чтобы выдать себя за Машу.

Существует несколько общих правил построения протоколов аутентификации:

- Инициатор передачи должен доказать, кто он есть, прежде чем вы пошлете ему какую-либо важную информацию.
- Инициатор и отвечающий должны использовать разные ключи.
- Инициатор и отвечающий должны использовать начальные вызовы из разных непересекающихся множеств.

Установка общего закрытого ключа. Протокол Диффи–Хеллмана.

Прежде всего, Маша и Петя должны договориться об использовании двух больших простых чисел n и g, удовлетворяющих определенным условиям, причем эти числа могут быть общеизвестны. Затем Маша выбирает большое число, например x, и хранит его в секрете, а Петя выбирает число y и также держит его в секрете. Маша посылает Пете сообщение вида (n, g, $g^x \ mod \ n$), а Петя отвечает сообщением вида ($g^y \ mod \ n$). Теперь Маша выполняет операцию $(g^y \ mod \ n)^x$, а Петя — операцию $(g^x \ mod \ n)^y$. Теперь они имеют общий ключ $(g^{xy} \ mod \ n)$. Злоумышленник следит за всем этим, и единственное, что мешает ему вычислить x и y, это то, что неизвестен алгоритм с приемлемой сложностью вычисления логарифма от модуля для простых чисел. Слабое место – «чужой в цепочке».

Проверка подлинности через центр раздачи ключей.

Идея протокола проверки подлинности через ЦРК состоит в следующем. Маша выбирает ключ сессии K_S. Используя свой ключ K_A, Маша посылает в ЦРК запрос на соединение с Петей. ЦРК знает Петю и его ключ K_B. С помощью этого ключа ЦРК сообщает Пете ключ сессии K_S и информацию о том, кто хочет с ним соединиться.

Однако это решение уязвимо к атакам подменой. Пусть злоумышленник как-то убедил Машу связаться с Петей и скопировал весь обмен сообщениями. Позже он может воспроизвести этот обмен за Машу и заставить Петю действовать так, как если бы с ним говорила Маша.

Существует несколько решений:

- Использование временных меток.
- Использование разовых меток. Можно комбинировать использование разовых и временных меток.
- Смена ключей для каждой новой транзакции, для чего необходимо иметь заранее согласованный список одноразовых ключей. Ключ повторно использован быть не может. Наиболее часто используемое решение.

Электронная цифровая подпись.

Подлинность многих юридических, финансовых и прочих документов устанавливается наличием подписи уполномоченного лица. Имеются способы, позволяющие отличить фотокопии от подлинника. Подпись на документе — это факт, подтверждающий, что лицо, подписавшее документ, либо является автором документа, либо знакомо с документом. Проблема создания электронного аналога ручной подписи:

- Получатель должен иметь возможность удостовериться в подлинности отправителя.
- Отправитель не должен иметь возможность отречься от документа.
- Получатель не должен иметь возможность подделать документ.

Подпись с секретным ключом. Одно из решений проблемы электронной подписи — наделить полномочиями третью сторону, которую знают все, которая знает всех и которой верят все — «Сердечный друг» (СД). Единственная слабость такого решения заключается в том, что злоумышленник может скопировать диалог между отправителем и получателем через СД и позже его повторить. Механизм временных меток позволяет ослабить эту проблему. Кроме того, сохранение последних ключей позволяет Пете заметить их повторное использование. Недостаток: необходимость доверять сердечному другу.

Подпись с открытым ключом. Предположим $E(D(P)) = P$ дополнительно к $D(E(P)) = P$ (этим свойством обладает алгоритм шифрования RSA). В этой схеме имеются два недостатка. Оба основаны на том, что схема работает до тех пор, пока Маша либо умышленно не рассекретит свой ключ, либо не изменит его в одностороннем порядке. При наступлении судебного случая Петя предъявляет сообщение P и $D_A(P)$, а так как он не знает закрытый ключ Маши, то, значит, не мог подделать $D_A(P)$. При этом должно $E_A(D_A(P)) = P$, в чем суд легко может убедиться.

Если Маша обращается в суд, т. е. предъявляет сообщение P и открытый ключ $E_A(P)$, это легко сопоставить с тем, что есть у Пети. Однако если Маша заявит, что у нее украли ключи, а сама тайно передаст их либо сменит, не сообщив об этом Пете, то в последнем случае текущий открытый ключ E_A будет неприменим к тому закрытому ключу $D_A(P)$, который предъявит Петя. При этом надо сопоставить даты передачи сообщения и смены ключей.

Профиль сообщения.

Недостаток этих методов в том, что они подменяют задачу установления подлинности задачей шифрования, когда зачастую необходимо только установление подлинности, а шифрование — медленная операция. Рассмотрим метод, который не требует шифрования всего сообщения. Он основан на использовании хэш-функции, которая по сообщению вычисляет битовую строку фиксированной длины. Эта функция, называемая профилем сообщения (Message Digest — MD), обладает тремя свойствами:

- У функции MD нет обратной функции.
- Для заданного сообщения P вычислить функцию MD(P) просто.
- Имея MD(P), невозможно восстановить P.
- Никто не сможет подобрать два таких сообщения, MD от которых будут одинаковыми.

Этот метод можно применять как с закрытым, так и с открытым ключом.

52 Информационная безопасность: контроль доступа и защита от компьютерных атак. Межсетевые экраны и их виды. Системы обнаружения и предотвращения компьютерных атак (метод аномалий и метод злоупотреблений).

Межсетевые экраны.

При построении интранета используется хорошо известная идея построения крепости: все, что находится внутри крепостных стен, дружественно, т.е. это среда, которой можно доверять до определенной степени, а все, что вне крепостных стен, – враждебно.

Число точек входа-выхода в интранете строго ограничено и все информационные потоки, проходящие через эти точки, строго проверяют. Точки входа-выхода образуют так называемый периметр сети организации. Только эти точки являются выходами корпоративной сети в Интернет.

Чистилище – это средство, позволяющее ограничить число точек входа в сеть на сетевом уровне и обеспечить в этих точках жесткий контроль входящих и исходящих пакетов. Весь трафик в интранет и из интранета направляется только через один шлюз, где происходит проверка пакета на соответствие определенным требованиям. Если пакет не удовлетворяет этим требованиям, го он не допускается в сеть или не выпускается из нее.

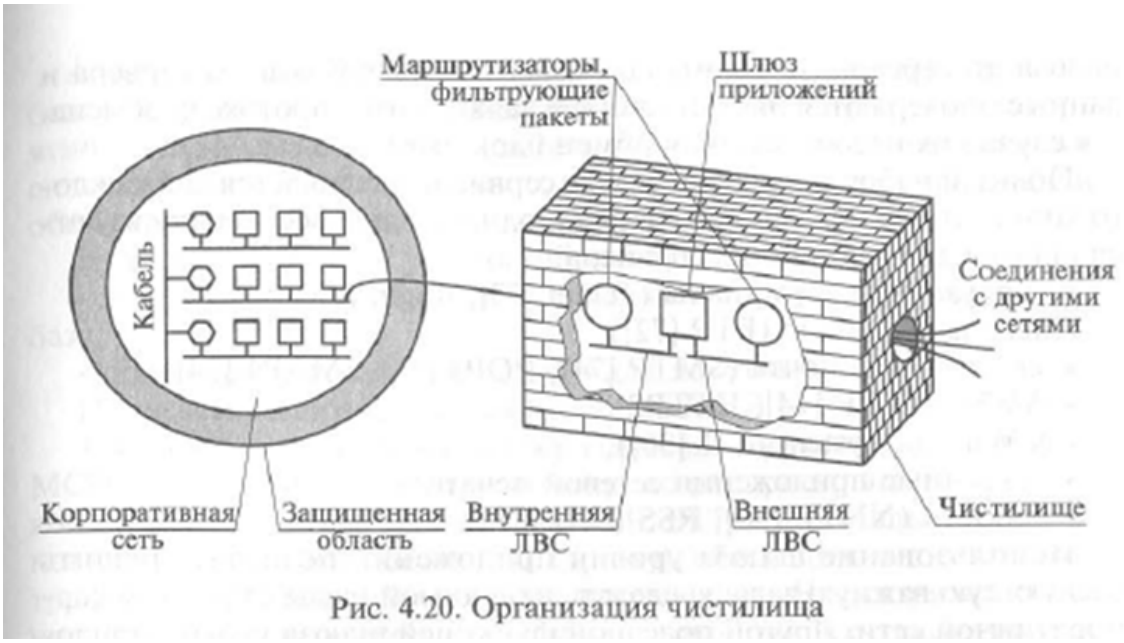


Рис. 4.20. Организация чистилища

На рис. 4.20 показана организация чистилища, состоящего из двух маршрутизаторов, фильтрующих пакетов и шлюза приложений. Фильтры содержат таблицы , где перечислены марш рутизаторы и абонентские машины , от которых можно принимать и которым можно передавать пакеты. Шлюз приложений ориентирован на конкретные приложения.

Межсетевые экраны могут также использоваться для разграничения доступа к внутренним сетевым ресурсам компании между различными ее подразделениями. Функциями межсетевого экрана являются сокрытие внутренней структуры сети, состава ресурсов сети и блокирование доступа к отдельным ресурсам.

МСЭ может быть реализован в виде программно-аппаратного комплекса на базе специализированной операционной системы, а может быть программной частью ОС общего назначения и ее ядра.

Все межсетевые экраны подразделяются на три типа:

- пакетные фильтры;
- шлюзы уровня соединения;
- шлюзы уровня приложений.

Все три типа МСЭ могут одновременно находиться на одном устройстве. Принцип их функционирования прост: МСЭ проверяет заголовок PDU соответствующего уровня на соответствие его параметров заранее введенным в МСЭ правилам. Если заголовок PDU несоответствует хотя бы одному правилу, то МСЭ такой PDU не пропускает.

Пакетные фильтры, являющиеся наиболее простым типом межсетевых экранов, работают на сетевом уровне. Описание правил прохождения пакетов для МСЭ выполняется в виде Поля «Протокол», «Порт

источника». «Порт назначения» относятся к правилам МСЭ транспортного уровня. Поле «Действие» может принимать значения пропустить или отбросить PDU, а флага здесь те же, что и и заголовке IP -пакета.

Шлюз уровня соединения выполняет анализ заголовков сегментов в рамках каждого транспортного соединения (например , TCP-соединения). Шлюз уровня соединения принимает решение для соединения целиком.

Шлюзы уровня приложений, устанавливаемые между клиентами и серверами конкретных приложений, анализируют сессии соответствующих протоколов – SMNP , FTP, HTTP и т.д. Соединение между клиентом и сервером дефакто разбито на два независимых соединения: от клиента до шлюза и от шлюза до сервера. Все команды клиента серверу, ответы сервера и данные проверяются на соответствие заявленному протоколу обмена, и в случае их несоответствия обмен блокируется. Полный набор поддерживаемых сервисов различается для каждого конкретного межсетевого экрана, однако чаще всего используются шлюзы для следующих сервисов:

- виртуальные терминалы (Telnet, rlogin);
- передача файлов (FTP);
- электронная почта (SMTP, POP3, IMAP4);
- WWW (HTTP, HTTPS);
- X Window System;
- различные приложения сетевой печати;
- новости (NNTP, RSS) и т.д.

Использование шлюза уровня приложений позволяет решить следующую важную задачу: сделать невидимой извне структуру корпоративной сети. Другой полезной функцией шлюза уровня приложений является обеспечение возможности аутентификации пользователей корпоративной сети централизованно на шлюзе и разграничения их полномочии доступа к сетевым ресурсам предприятия в соответствии с сто политикой безопасности.

При описании правил доступа используются следующие параметры: название сервиса, имя пользователя, допустимый временной диапазон использования сервиса, IP-адреса компьютеров, с которых можно пользоваться сервисом, и схемы аутентификации. Шлюзы прикладного уровня позволяют обеспечит ь наиболее высокий уровень защиты, поскольку взаимодействие с внешним миром здесь реализуется через небольшое число прикладных программ, полностью контролирующих весь входящий и выходящий трафик.

Достоинствами пакетных фильтров являются:

- невысокая стоимость;
- высокая скорость обработки трафика (низкая вычислительная сложность);
- небольшая задержка при прохождении пакетов.

Недостатки пакетных фильтров следующие:

- каждый пакет анализируется вне контекста соединения и сетевого трафика;
- диапазон параметров фильтрации ограничен полями заголовков протоколов IP, TCP и UDP;
- аутентификацию с использованием IP-адреса можно обмануть, используя подмену IP-адресов (IP spoofing);

- отсутствие аутентификации на пользовательском уровне.

Преимуществами шлюзов уровня приложений являются:

- отсутствие непосредственного сетевого соединения между клиентом и сервером, что позволяет скрыть внутреннюю структуру сети от внешнего мира;
- наличие защиты на уровне приложений позволяет осуществлять большое число дополнительных проверок, снижая тем самым вероятность ее взлома с использованием уязвимостей программного обеспечения (ошибок в программах или в их настройке, позволяющих обойти механизмы защиты, аутентификации и т.д.).

Недостатки шлюзов уровня приложений следующие:

- высокие вычислительная сложность и нагрузка на аппаратную базу;
- производительность ниже, чем для пакетных фильтров.

Приведем уязвимости и слабые места МСЭ.

1. Сложность защиты новых сетевых сервисов.
2. Возможность обхода межсетевого экрана. Межсетевые экраны не могут защитить ресурсы корпоративной сети в случае неконтролируемого использования в ней модемов, мобильных телефонов и беспроводны х точек доступа.
3. Незащищенность от вредоносного программного обеспечения и компьютерных атак.

СОА.

Сложность и высокая стоимость разработки защищенны х систем, свойства безопасности которых доказаны формально, обусловили появление и развитие направления информационной безопасности, связанного с обнаружением нарушений безопасности информационных систем.

Принято разделять методы обнаружения атак на методы обнаружения аномалий и методы обнаружения злоупотреблений. Методы обнаружения аномалий используют описание нормального поведения наблюдаемых объектов в сети, и любое отклонение от нормального поведения считается аномальным – нарушением. Методы обнаружения злоупотреблений используют описание запрещенных действий объектом в сети, например описание известных атак, и если наблюдаемое поведение некоторого объекта сети совпадает с описанием запрещенного, то действие объекта блокируют.

Методы обнаружения злоупотреблений используются в больш инстве современных коммерческих систем (Cisco IPS, ISS RealSccure, NFR). В этих системах каждая известная сетевая атака представлена сигнатурой – шаблоном значений полей заголовков сетевых пакетов и содержимого их полей данных, сопровождающих атаку. Основной недостаток методов обнаружения злоупотреблений состоит в том, что они не способны обнаруживать новые атаки, как и антивирусны е системы, т. е. если описание вируса отсутствует в базе данных системы, то она не будет реагировать на этот вирус.

Существуют два основных типа систем обнаружения атак (СОА): узловые и сетевые. Узловая СОА располагается на отдельном узле и отслеживает признаки атак на узел. Сетевая СОА находится на отдельном узле сети, через который проходит весь сетевой трафик или строго определенная его часть в данном сегменте.

Узловые СОА.

Узловая СОА состоит из сенсоров и анализаторов. Задача сенсоров заключается в регистрации событий, важных с позиции безопасности на защищаемом узле: вход-выход пользователя в систему, изменения в составе программного и аппаратного обеспечения узла, взаимодействие с другими узлами сети

и т.п. Анализаторы узловой СОА получают от сенсоров данные о событиях безопасности и выявляют в них признаки атак.

Выделяют три типа узловых СОА: **анализаторы журналов**; **анализаторы системных вызовов**; **анализаторы поведения приложений**.

Анализаторы журналов работают с журналами приложений, безопасность которых является критичной, а также с журналами операционной системы. Если такой анализатор встретит запись, соответствующую некоторому шаблону в своей базе, то он зарегистрирует событие нарушения безопасности.

Анализаторы журналов реагируют на событие уже после того, как оно произошло. Таким образом, журнал будет содержать сведения о том, что проникновение в систему состоялось. В большинстве случаев анализаторы журналов не способны предотвратить осуществляемую атаку на систему.

Анализаторы системных вызовов следят за вызовами между приложениями и операционной системой. Сенсоры узловой СОА данного типа размещаются между операционной системой и приложениями и «подменяют» стандартные системные вызовы ОС, а часто и функции стандартных библиотек, таких как libc, библиотека сокетов и т.д. Когда приложению требуется выполнить систем ны й вызов, параметры вызова попадают в «обертку», создаваемую узловой СОА, и анализируются в соответствии с набором заданных шаблонов. Шаблоны различаются для каждого типа приложения и для каждой известной атаки на узел.

Анализаторы системных вызовов отличаются от анализаторов журналов тем, что часто они могут предотвращать развитие атаки в том случае, если являются частью операционной системы и могут разрешать или запрещать выполнение системного вызова. Если приложение генерирует вызов, который может быть использован, например для атаки на переполнение буфера, то датчик може т блокироват этот вызов и сохранить систему в состоянии информационной безопасности.

Анализатор поведения приложений оперирует описаниями нормального (разрешенного) поведения приложений, например в виде профилей разрешенных для приложений системных вызовов ОС.

Анализатор наблюдает за поведением приложений, и в случае появления «запрещенного» вызова выполнение приложения принудительно останавливается.

Сетевые СОА.

Сетевая СОА располагается на узле, через который проходит большая часть трафика определенной части сети.

В случае с коммутируемой СПД существуют два варианта подключения датчиков сетевых СОА: через специальный порт, отслеживающий весь трафик, проходящий через сетевой коммутатор (SPAN-порт), и чередубликатор трафика (network tap) . В настоящее время большинство сетевых СОА используют для обнаружения атак методы обнаружения злоупотреблений.

Сетевые СОА обладают следующими преимуществами по сравнению с узловыми:

- сетевую СОА можно полностью скрыть в сети, и злоумышленник не будет знать о том. что за ним ведется наблюдение;
- один сенсор сетевой СОА может использоваться для мониторинга трафика в сегменте сети с большим числом систем-целей, потенциальных для атак;
- сетевая СОА может перехваты ватт, содержимое всех пакетов, направляющихся на определенный узел.

Недостатки сетевых СОА:

- отсутствие адаптивности к неизвестным атакам, сигнатуры которых отсутствуют в базе СОА;
- сетевая СОА не может определить, была ли атака успешной (так как она не видит результата атаки);
- как правило, сетевая СОА не может просматривать зашифрованный трафик.

В настоящее время получают развитие гибридные системы, которые объединяют в себе узловые и сетевые сенсоры, сохраняя все достоинства соответствующих систем.

Пример использования СОА для контроля политики безопасности

Политика безопасности	Сетевая СОА	Узловая СОА
Обнаружение атак	Весь трафик, поступающий на потенциально атакуемые системы (межсетевые экраны, веб-серверы, серверы приложений и т.д.)	Неудачные попытки входа. Попытки соединения. Удачный вход с удаленных систем. Запуск запрещенных приложений
Предотвращение атак	То же	То же
Сбор доказательств	Содержимое всего трафика, формируемого на системе-цели или атакующей системе	Все успешные подключения, исходящие от атакующей системы. Все неудачные соединения от атакующих систем. Все нажатия клавиш из интерактивных сеансов на атакующих системах

53 Служба DNS: основные функции, структуры данных, принципы функционирования.

Каждая машина в Интернете должна иметь IP-адрес. Однако оперировать числовыми IP-адресами неудобно, поэтому рассмотрим, как в Интернете можно использовать символьные имена вместо IP-адресов и как пользователь на абонентской машине может узнать IP-адреса других абонентских машин, зная их имена.

Все Интернет-приложения позволяют при обращении к узлам сети вместо числовых адресов использовать имена, зафиксированные в специальной распределенной базе данных DNS, которая поддерживает иерархическую систему имен для идентификации абонентских машин или узлов в сети Интернет. Такой способ адресации на прикладном уровне называется символьной адресацией. Аналогия с почтовой службой.

Проблемы: никакие компьютеры, включенные в сеть, не могут иметь одинаковых имен; преобразование имен в числовые адреса.

Когда Интернет был невелик, иметь дело с именами было довольно просто. Организация NIC создала регистратуру. Можно было послать запрос, и в ответ получить список имен и адресов – host file, этот файл регулярно рассылался всем машинам в сети. Все имена были простыми и уникальными. Компьютер просматривал файл и подставлял вместо имени реальный числовой адрес. По мере развития и расширения Интернета стало очевидно, что требуется распределенная оперативная система, работающая на новом принципе. Такая система была создана, и ее назвали доменной системой имен – DNS (Domain Name Service), а способ адресации в этой системе – способом адресации по доменному принципу. Также эту систему иногда называют региональной системой наименований.

Структура региональной системы имен.

Доменная система имен – это метод, при котором в сетевой группе выделяется абонентская машина, отвечающая за назначение имен машинам в группе и обладающая полнотой информации о всех именах машин группы и их IP-адресах. При этом группы первого уровня могут быть объединены в группы второго уровня, группы второго уровня – в группы третьего уровня и т.д., причем ни одна группа не

может входить в две и более групп. Каждая группа в такой иерархии называется доменом. Чтобы указать путь к интересующей нас машине, достаточно перечислить имена от самого верхнего домена до самого нижнего, содержащего интересующую нас машину. Домены в имени отделяют друг от друга точками. В имени может быть различное число доменов. Первым в имени стоит название абонентской машины – реального компьютера с IP-адресом. Это имя создано и поддерживается группой, к которой он относится. Группа входит в более крупное подразделение, которое, в свою очередь, является частью национальной сети.

Все пространство доменов распределено на зоны. Имена зон можно условно подразделить на организационные и географические. Организационные зоны: com, edu, gov, mil, net, org. В организационных зонах обычно размещаются непосредственно домены организаций. Каждая страна имеет свой географический домен из двух букв.

В доменном имени слева в конце цепочки доменных имен должно быть указано имя абонентской машины. Это имя может быть собственным или функциональным. Имена собственные каждый придумывает в меру своей фантазии. Имена функциональные вытекают из функций, выполняемых машиной: www – сервер HTTP (WWW); ftp – FTP-сервер; ns, nss, dns – сервер DNS (Name); mail – почтовый сервер; relay – почтовый сервер обмена; proxy – соответствующий proxy-сервер.

Доменная группа может создавать или изменять любые принадлежащие ей имена.

Поиск адреса по доменному имени.

Доменная система работает автоматически, т.е. нам не надо разыскивать адрес, соответствующий имени, или подавать специальную команду для его поиска. Все компьютеры в Интернете могут пользоваться доменной системой.

Когда используют имя, например zodiac.pe, его необходимо преобразовать в IP-адрес. Для этого приложение формирует запрос к DNS-серверу, где работает DNS-служба. Эта служба – приложение, обладающее соответствующей базой данных, с помощью которой оно обслуживает такого рода запросы.

Обработка имени DNS-сервером выполняется справа налево, т.е. сначала производится поиск адреса в самой верхней группе иерархии, а затем он постепенно опускается по иерархии, тем самым сужая область поиска. В целях сокращения поиска сначала опрашивается локальный узел DNS. При этом возможны три варианта ответов:

- Местный сервер знает адрес, потому что этот адрес содержится в его базе данных.
- Местный сервер знает адрес, потому что кто-то недавно уже запрашивал его, и он сохранил у себя в кэш-памяти этот адрес. Когда запрашивается адрес, DNS-сервер придерживает его у себя в кэш-памяти некоторое время на случай, если кому-нибудь потребуется тот же адрес, что повышает эффективность системы.
- Местный сервер адрес не знает. В этом случае запускают ранее описанную процедуру опроса DNS-серверов доменов, указанных в имени справа налево.

Серверы имен.

Нет и не может быть единого сервера, содержащего базу DNS, охватывающую весь Интернет (из-за вопросов безопасности, надёжности и производительности).

Чтобы сделать базу распределенной, все пространство имен доменов разбивают на непересекающиеся зоны. Границы каждой зоны определяет ее администратор. Каждая такая зона покрывает часть дерева доменов, и в нее входят серверы имен этих доменов. Обычно в каждой зоне есть основной сервер имен зоны и несколько вспомогательных серверов имен. Часто из соображений надежности сервер зоны располагают вне этой зоны.

Весь процесс поиска IP-адреса по имени домена реализуют серверы имен. Если запрос относится к юрисдикции того сервера имен, к которому обратились, т. е. запрашиваемый домен находится в ведении данного сервера имен, то этот сервер генерирует ответ, содержащий записи всех ресурсов, соответствующих запросу, и этот ответ считается авторитетным, т.е. содержащаяся в нем информация считается

apriori 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. При этом записи в кэш-памяти не являются авторитетными. Время актуальности содержащейся в них информации определяет поле времени жизни.

Записи ресурсов.

С каждым доменом связано множество ресурсов, отнесенных к этому домену. Записи об этих ресурсах хранятся в базе DNS. Когда происходит обращение к DNS-серверу с каким-либо именем, в ответ приходит не только IP-адрес, но и запись о ресурсах, соответствующих указанному имени.

Запись о каждом ресурсе состоит из пяти полей: **«Имя домен»** (Domain name), **«Время жизни»** (Time to live), **«Класс»** (Class), **«Тип»** (Type), **«Источник полномочий»** – SOA (Start Of Authority).

В поле **«Имя домена»** указывается 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.

В поле **«Время жизни»** указывается интервал времени в секундах, в течение которого значение этого поля считается неизменным.

В поле **«Класс»** указывается значение IN, если ресурс, к которому относится эта запись, является ресурсом Интернета. Здесь могут быть указаны и другие значения, но они встречаются редко.

В поле **«Источник полномочий»** указывается имя источника информации о зоне сервера имен (об этом сервере будет сказано далее). Также здесь указывается адрес электронной почты администратора сервера имен и другая служебная информация. Если в этом поле указано значение A, то это означает, что в следующем поле указан IP-адрес этого ресурса. При указании в этом поле значения MX за ним следует имя машины, которая может получать почту для данного домена.

Записи типа **NS** указывают на серверы имен, относящиеся к домену верхнего уровня. Эта информация необходима при пересылке почты в другие домены.

Записи типов **CNAME** и **PTR** позволяют создавать псевдонимы. Например, человек может иметь несколько адресов электронной почты, но все они будут относиться к одному почтовому ящику.

Запись типа **HINFO** позволяет определять тип машины и операционной системы соответствующего ресурса.

Замечания по региональной системе имен.

Доменная служба имен указывает на ответственного за поддержку имени, но ничего не сообщает о владельце компьютера, т. е. где эта машина находится географически (несмотря на коды стран).

Понятия доменного имени и сети не связаны. Часто доменные имена и сети перекрываются, и жестких связей между ними нет, т.е. две машины одного домена могут не принадлежать к одной сети.

У машины может быть много имен. В частности, это верно для машин, предоставляющих какие-либо службы, которые в будущем могут быть помещены на другую машину. Когда эти службы будут перемещены, то имя, под которым такая машина выступала в качестве их сервера, будет передано новой машине-серверу вместе с услугами. При этом для внешних пользователей ничего не изменится, т.е. они будут продолжать пользоваться этой службой, запрашивая ее по имени, независимо от того, какой компьютер на самом деле реализует ее. Имена, по смыслу относящиеся к службе и называемые каноническими, в Интернете встречаются довольно часто.

54 Организация, функционирование и основные протоколы почтовой службы в Интернет.

Поначалу возможности электронной почты сводились к передаче файлов с одним ограничением: первая строка файла должна была содержать адрес получателя. Со временем этого оказалось недостаточно в силу следующих обстоятельств:

- посылать одно и то же сообщение сразу нескольким получателям было неудобно,

- сообщение не имело внутренней структуры, что усложняло его обработку на машине,
- отправитель никогда не знал, получено сообщение или нет,
- невозможно перенаправить свои сообщения кому-то другому,
- интерфейс пользователя был неудобен, поскольку пользователь должен был от работы в редакторе файлов переходить в систему отправки файлов,
- было невозможно отправить в одном и том же сообщении и текст, и голос, и видео.

Архитектура почтовой системы включает в себя два основных компонента: агента пользователя (отвечает за интерфейс с пользователем, составление и отправку сообщений) и агента передачи сообщений (отвечает за доставку сообщения от отправителя к получателю).

Обычно почтовая система поддерживает пять базовых функций:

1. **Композиция.** Обеспечивает создание сообщений и ответов. Хотя для формирования тела сообщения может использоваться любой текстовый редактор, система автоматизирует заполнение многочисленных полей заголовка сообщения. Например, подстановка адреса при формировании ответа.
2. **Передача.** Обеспечивает передачу сообщения от отправителя к получателю без вмешательства пользователей.
3. **Отчет о доставке.**
4. **Визуализация сообщения.** Выполняет перекодировку сообщения, изменение формата и т.д.
5. **Размещение,.** Определяет, что делать с сообщением: уничтожить после (до) прочтения или, если сохранить, то где. Поиск интересующего сообщения, перенаправление сообщения, повторное прочтение ранее полученного сообщения относятся также к данной функции.

Кроме указанных обязательных функций в большинстве почтовых систем имеется и ряд других функций. Например если пользователь уехал, он может перенаправить поступающие в его отсутствие сообщения куда-либо еще. Во многих системах пользователь может создавать так называемые внутренние почтовые ящики для поступающих сообщений; создавать лист рассылки, по которому одно и то же сообщение будет разослано всем его участникам; сортировать сообщения по определенным директориям в зависимости от их характеристик и многое другое.

Ключевой функцией всех современных почтовых систем является разделение почтового отправления на конверт сообщения и собственно сообщение. Система доставки использует только конверт, содержащий всю необходимую ей информацию о сообщении: адрес назначения, приоритет, секретность, требование об уведомлении и т.д. Сообщение внутри конверта имеет заголовок и тело. Заголовок содержит всю необходимую информацию о теле для агента пользователя, а тело предназначено исключительно для пользователя.

Агент пользовател.

Агент пользователя – это обычно программа уровня приложений, способная выполнять определенный набор команд для получения, написания и композиции сообщения и ответа на сообщение, а также для работы с почтовым ящиком. При этом некоторые агенты используют командную строку, а некоторые – графический интерфейс.

Отправка почты. Чтобы послать сообщение, пользователь должен предоставить адрес назначения, само сообщение и другие его параметры, например приоритетность, секретность и т.п. Для создания сообщения может быть использован любой 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, где location – доменное имя, mailbox – ресурс в самом внутреннем (левом) домене в доменном имени.

Агент пользователя также поддерживает лист рассылки, который позволяет рассылать одно и то же сообщение сразу нескольким пользователям. Причем сообщение размножается необязательно самим агентом, а там, где поддерживается лист рассылки.

Чтение почты. Прежде чем агент пользователя (далее АП) что-либо высветит на экране при загрузке, он просмотрит почтовый ящик на предмет новых поступлений и высветит на экране его содержимое с краткой аннотацией каждого сообщения. В простых почтовых АП высвечиваемые поля встроены в АП, а в развитых – пользователь сам определяет, что показывать, а что нет (эта информация содержится в файле user profile).

Формат сообщений.

Формат RFC 822. Сообщение состоит из простейшего конверта (описанного в RFC 821), полей заголовка, пустой строки и тела сообщения. Каждая строка заголовка – это строка ASCII-текста, содержащая название поля, двоеточие и какое-то значение. Стандарт 822 не различает четко заголовок и конверт. В современных почтовых системах это различие более четкое, и агент пользователя имеет дело с заголовком, а агент передачи – с конвертом, формируемым на основе заголовка.

Формат MIME (Multipurpose Internet Mail Extension). Когда Интернет только начинал развиваться, почтовые системы способны были передавать только текстовые сообщения на английском языке в формате ASCII. Для этих целей RFC 822 было достаточно. В наши дни этих возможностей уже недостаточно. Необходимо, чтобы почтовая система умела работать:

- с сообщениями на европейских языках (на французском, немецком и т.д.),
- с сообщениями не в латинском алфавите (на русском, арабском и т.д.),
- с сообщениями вне алфавита (на японском, китайском),
- с сообщениями, содержащими не только текст (звук, видео, графику).

Решение: MIME – многоцелевое расширение почтовой службы в Интернете. Основная идея – расширение RFC 822 в целях структурирования тела сообщения и введения правил кодировки ASCII-сообщений. Введение MIME повлияло на программы доставки и отправки сообщений. В формате MIME определены пять новых заголовков.

Заголовок **MIME-Version** сообщает агенту пользователя, что он имеет дело с MIME-сообщением и какая версия MIME используется.

Заголовки **Content-Description** и **Content-Id** характеризуют сообщение. Например, второй заголовок можно использовать для фильтрации сообщений.

Заголовок **Content-Transfer-Encoding** определяет подготовку сообщения для передачи через сеть, для чего используются четыре основных схемы. Простейшая схема применяется для передачи ASCII-текста – 7 бит на символ (для учета национальных алфавитов используется схема 8 бит на символ) при условии, что длина строки не превышает 1000 символов в строке. Для корректной передачи двоичных данных (например, исполняемого кода программ) используется схема base64 encoding, которая разбивает сообщение на блоки по 24 бит. Каждый блок разбивается на четыре группы по 6 бит каждая. Для сообщений, которые являются «почти» ASCII-сообщениями с небольшими исключениями, используется схема quoted-printable-encoding.

Можно также указать и какую-то особую схему в поле **Content-Transfer-Encoding**.

В поле заголовка **Content-Type** указывается тип сообщения.

Передача сообщений.

Основная задача системы передачи почтовых сообщений – надежная доставка сообщения от отправителя к получателю. Самым простым способом в этом случае является использование простого протокола передачи почты – SMTP (Simple Mail Transfer Protocol).

В Интернете почта передается следующим образом. Машина-отправитель устанавливает TCP-соединение с 25-м портом машины-получателя, на котором находится почтовый демон, работающий по

протоколу SMTP. Этот порт принимает соединение и распределяет поступающие сообщения по почтовым ящикам машины-получателя. После установки соединения машина-отправитель работает как клиент, а машина-получатель – как сервер. Сервер посылает текстовую строку, идентифицирующую его и готовность принимать почту. Если он не готов принимать почту, то клиент разрывает соединение и повторяет всю процедуру позднее. Если сервер подтвердил свою готовность принимать сообщение, то клиент сообщает, от кого и кому оно предназначено. 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.

Недостатки SMTP-протокола:

1. Длина сообщения не может превосходить 64 Кбайт.

2. Наличие time-out. Если время ожидания подтверждения у отправителя и получателя не согласовано, то один будет разрывать соединение, не дождавшись, тогда как другой просто будет очень загружен.

3. Возможность возникновения почтового урагана. 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, по которому клиент сначала посылает команду EHLO, и если она отвергается сервером, это означает, что сервер работает по SMTP.

Почтовые шлюзы. Протокол SMTP хорош, когда обе машины находятся в Интернете. Однако это не всегда так. Многие компании в целях сетевой защиты соединяют свои сети через надлежащие средства либо используют другие протоколы. В этом случае отправитель передает сообщение шлюзу, тот его буферизует и позднее передает получателю. Проблемы:

1. Соответствие адресов.

2. Соответствие структур конвертов и заголовков.

3. Соответствие структуры тела сообщения.

Для простых неструктурированных ASCII-сообщений SMTP-шлюз способен решить такие проблемы.

Доставка получателю. Самый простой протокол для изъятия почты из удаленного почтового ящика – **POP3 (Post Office Protocol)**, описанный в RFC 1225, позволяет входить в удаленную систему и выходить из нее, передавать письма и принимать их, а главное – он позволяет забирать почту с сервера и хранить ее на машине пользователя.

Более сложный протокол – **IMAP (Interactive Mail Access Protocol)**, описанный в RFC 1064, позволяет одному и тому же пользователю заходить с разных машин на сервер, чтобы прочесть или отправить почту. Сервер в этом случае, по существу являющийся удаленным хранилищем писем, позволяет, например, получать доступ к письму не только по его номеру, но и по содержанию.

Важными почтовыми сервисами являются:

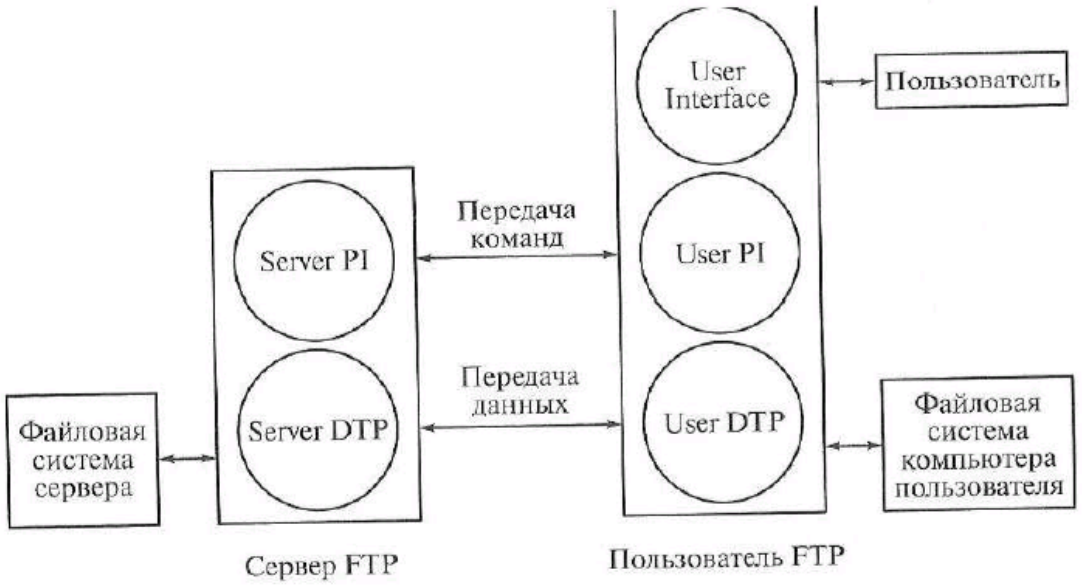
- фильтры (спам-фильтры, сортировщик почты и т.д.) (программа procmail в UNIX-системах),
- возможность пересылки поступающей почты на другие адреса,
- демон отсутствия (в UNIX-системах настраивается утилитой vacation),
- почтовый робот (программа, анализирующая входящие письма и отвечающая на них).

55 Служба FTP: организация, протокол.

Протокол передачи файлов (File Transfer Protocol – FTP) предназначен для решения следующих задач:

- разделение доступа к файлам на удаленных абонентских машинах;
- прямое или косвенное использование 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. Его нельзя использовать для передачи конфиденциальных данных, поскольку он не обеспечивает защиты передаваемой информации и передает между сервером и клиентом открытый текст. Сервер FTP может потребовать от клиента FTP аутентификации (т.е. при установлении соединения с сервером FTP-пользователь должен будет ввести свой идентификатор и пароль). Однако и пароль, и идентификатор пользователя будут переданы от клиента на сервер открытым текстом.



- **User Interface** – пользовательский интерфейс работы с FTP;
- **User PI (User Protocol Interpretator)** – интерпретатор команд пользователя. Эта программа взаимодействует и с Server-PI, чтобы обмениваться командами управления передачей данных по каналу передачи команд, и с модулем User DTP, который осуществляет непосредственную передачу данных по каналу передачи данных;
- **User DTP (User Data Transfer Process)** – модуль, осуществляющий обмен данными между клиентом и сервером FTP по каналу передачи данных по командам от модуля User PI. Этот объект взаимодействует с файловой системой пользователя и объектом Server DTP;
- **Server PI (Server Protocol Interpretator)** – модуль управления обменом данных со стороны сервера по каналу передачи команд;
- **Server DTP (Server Data Transfer Process)** – модуль обмена данными со стороны сервера по каналу передачи данных;
- **Сервер FTP** – собственно сервер FTP, который состоит из модуля Server PI управления передачей и модуля Server DTP, осуществляющего передачу;

- **Пользователь FTP** – модуль клиента FTP, состоящий 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 и в одном, и в другом направлениях, 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 получения ответов после их обработки.

Алгоритм работы FTP следующий:

1. Сервер FTP устанавливает в качестве управляющего соединение с портом 21 TCP, который всегда находится в состоянии ожидания соединения со стороны FTP-клиента.

2. После установки управляющего соединения модуля User PI с модулем Server PI, клиент может отправлять на сервер команды. FTP-команды определяют параметры соединения передачи: роли участников соединения (активная или пассивная), 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 иди файл.

3. После согласования всех параметров работы канала передачи данных один из участников соединения, который является пассивным (например, клиентский модуль User DTP), переходит 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.

Протокол FTP можно использовать при передаче файлов не только между клиентом и сервером, но и между двумя FTP-серверами. 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, а данные транслируются через канал Server 1 DTP – Server2 DTP.

Основу передачи данных в протоколе FTP составляют 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 управляющий порт с номером L1. Однако, как правило, участниками соединения используются порты передачи данных, выбранные для них User PI, поскольку из всех управляющих процессов, участвующих в соединении, только он может изменять порты передачи данных как у User DTP, так и у Server DTP.

Пассивная сторона соединения должна до подачи команды начала передачи слушать свой порт передачи данных. Активная сторона, подающая команду на начало передачи, определяет направление перемещения данных.

После установки соединения между Server DTP и User DTP начинается передача. Одновременно по каналу Server PI – User PI передается уведомление о получении данных. Протокол FTP требует, чтобы управляющее соединение было открыто все время 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 закрывать канал передачи данных в следующих случаях:

- сервер закончил передачу данных в формате, требующем закрытия соединения,

- сервер получил от пользователя команду на прерывание соединения,

- пользователь изменил параметры порта передачи данных,
- было закрыто управляющее соединение,
- возникли ошибки, при которых невозможно возобновить передачу данных.

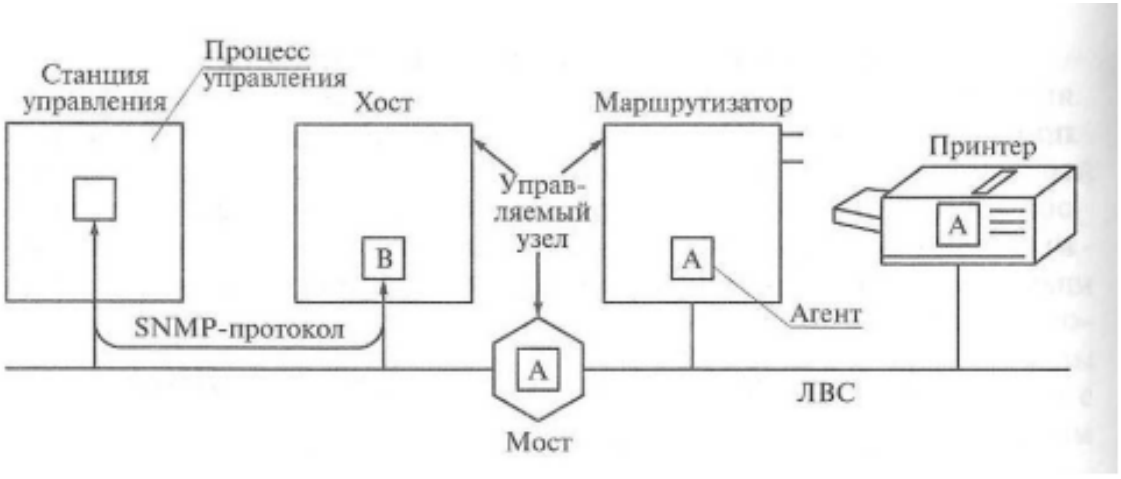
56 Служба управления сетью: организация, протокол SNMP, структура базы данных MIB.

Протокол управления сетью – SNMP.

Когда сеть из компьютеров охватывает небольшие пространства, то в случае возникновения неполадок можно обойти все помещения и проверить работоспособность каждого устройства и его программного обеспечения. Когда сеть охватывает большие территории и включает в себя оборудование, принадлежащее разным организациям, то такой обход уже невозможен. Требуются адекватные средства для управления сетью. В 1990 г. была опубликована первая версия протокола управления сетью (Simple Network Management Protocol – SNMP v.l). В RFC 1155 и RFC 1157 было описано, как организовано систематическое наблюдение за сетью (какая, как и где может накапливаться информация) и управление ею (как и какие параметры работы устройств сети можно изменять). Этот протокол получил широкое распространение и был реализован практически во всех устройствах, используемых в сетях. Ряд недостатков SNMP v.l: например, недостаточно были проработаны вопросы безопасности. Во второй версии протокола SNMP (RFC 1441... 1452) была введена криптографическая защита механизма аутентификации. Далее рассматривается SNMP v.2.

Модель управления, принятая в протоколе SNMP, использует четыре типа сущностей:

- станции управления;
- управляемые устройства;
- управляющая информация;
- протокол управления.



Управляют сетью станции управления, т.е. компьютеры, на которых выполняются процессы, собирающие и накапливающие информацию об управляемых устройствах в сети. Сбор этой информации происходит по запросу от управляющей станции к управляемому устройству. Запросы, передача и другие действия выполняются с помощью команд протокола SNMP.

На управляемых устройствах работают специальные SNMP-агенты, которые выполняют команды, передаваемые с помощью SNMP-протокола, и фиксируют определенный набор параметров функционирования управляемого устройства. Управляемым устройством может быть маршрутизатор, мост, рабочая

станция, устройство печати, т. е. любое устройство, где может работать SNMP-агент. Каждый агент поддерживает локальную базу данных MIB (Management Information Base). В этой базе хранится информация о состоянии агента, история его функционирования и переменные, характеризующие работу устройства, где функционирует агент.

Структура управляющей информации – SMI.

В сети используется аппаратура сотен различных производителей. Естественно, агент должен формировать данные о функционировании управляемого устройства в некотором унифицированном виде, например, по составу или 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 собираются 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 используемых понятий, и это дерево является поддеревом дерева стандартов.

Коллекцию объектов, которыми можно управлять с помощью протокола SNMP, определяет база управляющей информации – MIB. Все объекты этой базы подразделяются на 10 групп, соответствующих 10 узлам, смежных узлу MIB-2 в дереве стандартов.

SMI (Structure of Management Information) – это в определенном смысле язык для определения структур данных, представляющих собой объекты в базе данных MIB.

Управление в сети с помощью протокола SNMP.

SNMP-протокол определяет пять типов сообщений, которыми обмениваются станция управления и управляемое устройство:

- **get-request** – получить значение одной или нескольких переменных;
- **get-next-request** – получить значение одной или нескольких 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**;
- **trap** – уведомить станцию управления, когда что-либо произошло с агентом.

Первые три из этих сообщений использует станция управления, а последние два – управляемое устройство. Так как четыре из пяти SNMP-сообщений реализуются простой последовательностью типа «запрос–ответ», SNMP-протокол использует UDP-протокол. Это означает, что запрос от станции управления может не дойти до управляемого устройства, как и отклик от управляемого устройства – до станции управления. В этом случае будет задействован механизм time-out и выполнена повторная передача.

Станция управления отправляет все три запроса на UDP-порт 161. Управляемое 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 управляемое устройство.

При взаимодействии между станцией управления и управляемым устройством используется пароль, представляющий собой 6-символьную строку, которую в SNMP v.l передавали в открытом виде. В операторах get, get-next и set станция управления устанавливает идентификатор запроса (request ID), который возвращается управляемым устройством в сообщении get-response, что 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.

57 История WWW. Модель сервиса HTTP протокола - запросы, ответы, URL, заголовки. Семантика кодов HTTP-ответов.

Всемирная паутина WWW основывается на использовании гипертекста. Идея создать сеть из документов, расположенных на разных машинах и связанных гиперссылками, сформулирована Т. Бернерс-Ли в 1989 г. Начали использовать Web в январе 1992 г. в Женеве. Т. Бернерс-Ли предложил хранить документы на компьютерах, которые он назвал веб-серверами. Для реализации этой идеи были необходимы специальный протокол, умеющий работать с гипертекстовыми документами, средство описания документов и средство визуализации документа, собранного из отдельных документов, соединенных ссылками. Специальный протокол **HTTP (Hyper Text Transmission Protocol)** используется в Интернете с 1990 г. Для описания документов и связывания их гиперссылками служит язык **HTML (Hyper Text Markup Language)**, а для просмотра документов используются специальные программы – **браузеры**. Способность предоставлять информацию в виде видео, аудио, текста и изображений через стандартный набор элементов графического интерфейса, который не зависит от платформы, делает Web привлекательным ресурсом для всех категорий пользователей.

HTTP является текстовым протоколом. Общий вид HTTP-запроса следующий:

Запрос = Метод SP URI-Запроса SP Версия-HTTP CRLF
Заголовок-Запроса CRLF
Заголовок-Запроса CRLF
Заголовок-Запроса CRLF
CRLF
[Тело запроса]

В HTTP-запросе могут использоваться следующие **методы**: «GET», «HEAD», «PUT», «POST», « DELETE», «LINK», «UNLINK», «CONNECT».

Метод **GET** служит для получения любой информации, идентифицированной URI-запросом, Если URI-запрос ссылается на процесс, выдающий данные, в качестве ответа будут выступать данные, сгенерированные указанным процессом (если они не являются выходными данными процесса), а не код самого процесса. Согласно стандарту HTTP многократное повторение одного и того же запроса GET должно приводить к одинаковым результатам (при условии, что сам ресурс не изменился за время между запросами), что позволяет кэшировать ответы на него.

Метод **HEAD** аналогичен методу GET за исключением того, что клиенту возвращается только заголовок сообщения-ответа (усеченный GET). Этот метод в основном используется для тестирования гиперссылок и проверки доступа к ресурсам.

Метод **PUT** служит для сохранения передаваемого на сервер ресурса с идентификатором URI.

Метод **POST** предназначен для передачи серверу информации, включенной в запрос как дополнительной к ресурсу, указанному в поле URI-запроса. Метод POST был разработан как общий для осуществления следующих целей:

- аннотация существующих ресурсов,
- добавление сообщений в группы новостей, почтовые списки и другие подобные группы статей,
- доставка блоков данных процессам, обрабатывающим данные,
- расширение баз данных через операцию добавления.

В отличие от метода GET при многократном повторении одного и того же запроса с методом POST можно получать разные результаты (например, после каждой отправки комментария в форум будет появляться новая копия этого комментария).

Метод **DELETE** используется для удаления ресурса, определенного идентификатором URI.

Как правило, методы **PUT** и **DELETE** в современных web-серверах запрещены, и управление данными осуществляется только через метод POST.

Web-серверы – обработчики HTTP-запросов.

Протокол HTTP основан на парадигме запрос-ответ. Браузер устанавливает соединение с web-сервером и посылает ему запрос, содержащий метод запроса, URI и версию протокола, за которой следует сообщение в формате MIME, включающее в себя управляющую информацию запроса, информацию о клиенте и может быть тело сообщения. Для адресации на прикладном уровне большинство протоколов используют универсальные идентификаторы ресурса – **URI (Universal Resource Identifier)**. Самые известные примеры идентификатора URI – это URL и URN.

URL – это идентификатор URI, который помимо идентификации ресурса предоставляет еще и информацию о ее местонахождении. Изначально URL предназначался для обозначения мест расположения ресурсов (чаще всего файлов) во Всемирной паутине. Сейчас URL применяется для обозначения адресов почти всех ресурсов Интернета и позиционируется как часть более общей системы идентификации ресурсов URI, а сама аббревиатура URL постепенно уступает место более широкому обозначению URI.

URN – это идентификатор URI, который идентифицирует ресурс в определенном пространстве имен. Например, ISBN 0-395-36341-1 – это URI, указывающий на ресурс (книгу) 0-395-36341-1 в пространстве имен ISBN. В последнее время появилась тенденция говорить просто URI о любой строке-идентификаторе без дальнейших уточнений.

Web-сервер, получив запрос, разделяет его на части и выделяет идентификатор URI запрашиваемого ресурса. Далее по пути, указанному в идентификаторе URI-запроса, в файловой системе сервера находится запрашиваемый объект. В случае если для запрашиваемого объекта не указана программа выполнения (например, файл является HTML-документом или объектом мультимедиа), web-сервер возвращает его в теле HTTP-ответа. Если для запрашиваемого объекта указана программа выполнения (например, файл является скриптом на языке perl, php, ruby или python) либо если объект сам является исполняемым файлом, выполняют следующие действия:

- инициируют выполнение объекта, указанного в URL. Входными данными для этого объекта являются параметры HTTP-запроса. Правила доступа к параметрам HTTP-запроса из кода исполняемого объекта определяются конкретной технологией сопряжения Web-сервера с Web-приложением;
- исполняемый объект динамически генерирует тело HTTP-ответа в зависимости от входных параметров;
- сгенерированный выполненным объектом HTTP-ответ возвращается клиенту и обрабатывается браузером стандартным образом;
- сервер отвечает сообщением, содержащим строку статуса (включая версию протокола и код статуса – успех или ошибка), за которой следует сообщение в формате MIME, включающее в себя информацию о сервере, метаинформацию о содержании ответа и само тело ответа.

Большим шагом в развитии технологий генерации динамического контента был выпуск платформ Java Enterprise Edition (J2EE) и Microsoft .NET. Так же как и в языке Java, среда разработки .NET создает байт-код, предназначенный для исполнения виртуальной машиной. Применение байт-кода позволяет достигать независимости от конкретной среды выполнения (платформы). Перед запуском сборки программной системы в среде исполнения (Common Language Runtime – CLR) ее байт-код преобразуется встроенным в среду JIT-компилятором в машинные коды целевого процессора.

Код состояния HTTP (англ. HTTP status code) – часть первой строки ответа сервера при запросах по протоколу HTTP. Он представляет собой целое число из трёх арабских цифр[1]. Первая цифра указывает на класс состояния. За кодом ответа обычно следует отделённая пробелом поясняющая фраза на английском языке, которая разъясняет человеку причину именно такого ответа. Примеры:

- 201 – Webpage Created.
- 401 – Access allowed only for registered users.
- 507 – Insufficient Storage.

Клиент узнаёт по коду ответа о результатах его запроса и определяет какие действия ему предпринимать дальше. Набор кодов состояния является стандартом, и они описаны в соответствующих документах RFC. Введение новых кодов должно производиться только после согласования с IETF. Тем не менее, известно о двух используемых кодах, не упомянутых в RFC: 449 Retry With. Так же упоминается пояснительная фраза «Reply With» в спецификации по WebDAV в Microsoft Developer Network, введённый Microsoft и 509 Bandwidth Limit Exceeded, введённый в cPanel.

Клиент может не знать все коды состояния, но он обязан отреагировать в соответствии с классом кода. В настоящее время выделено пять классов кодов состояния:

- 1xx – Информационные.
- 2xx – Успех.
- 3xx – Перенаправление.
- 4xx – Ошибка клиента.
- 5xx – Ошибка сервера.

58 NAT: основные функции, принципы функционирования, влияние на приложения.

NAT (Network Address Translation) – в общем случае технология изменения source или destination IP адреса в пакете на какой-то другой. Чаще всего речь идёт о частном случае – napt44, он же inside source NAT with overload, он же PAT (Port Address Translation). Суть этой технологии в том, что меняется не только source IP, но и source port в L4 заголовке, и в специальную таблицу заносится соответствие inside IP/inside port – outside IP/outside port, и эта же таблица используется для трансляции обратного трафика. Всё это позволяет нескольким хостам выходить в Интернет, используя один и тот же реальный IP адрес. Хостам же назначаются т.н. «серые» адреса, которые маршрутизируются только в пределах локальной сети и не являются уникальными в Интернете. У технологии множество минусов, и, по мнению многих, она является костылём, но пока что это практически единственный способ давать пользователям доступ в Интернет в условиях нехватки IP адресов.

Стандарт определяет специальный диапазон серых IP для CGN – 100.64.0.0/10. Несмотря на это, многие используют привычные всем серые IP из другого стандарта, а именнно:

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

Частные IP адреса не маршрутизируются в интернете, поэтому маршрутизатору, который выходит в интернет напрямую, на интерфейс во внутренней сети назначается так же частный IP адрес, а на внешний сетевой интерфейс назначается один или несколько публичных IP адресов. Затем, при помощи протокола NAT происходит преобразование частных IP адресов во внешние.

Существует 3 базовых концепции трансляции адресов: **статическая** (Static Network Address Translation), **динамическая** (Dynamic Address Translation), **маскарадная** (NAPT, NAT Overload, PAT).

Статический NAT – Отображение незарегистрированного IP-адреса на зарегистрированный IP-адрес на основании один к одному. Особенно полезно, когда устройство должно быть доступным снаружи сети.

Динамический NAT – Отображает незарегистрированный IP-адрес на зарегистрированный адрес от группы зарегистрированных IP-адресов. Динамический NAT также устанавливает непосредственное отображение между незарегистрированным и зарегистрированным адресом, но отображение может меняться в зависимости от зарегистрированного адреса, доступного в пуле адресов, во время коммуникации.

Перегруженный NAT (NAPT, NAT Overload, PAT, маскарадинг) – форма динамического NAT, который отображает несколько незарегистрированных адресов в единственный зарегистрированный IP-адрес, используя различные порты. Известен также как PAT (Port Address Translation). При перегрузке каждый компьютер в частной сети транслируется в тот же самый адрес, но с различным номером порта.

Статический NAT.

В этом случае один внутренний адрес преобразуется в один внешний. И при этом все запросы, приходящие на внешний адрес будут транслироваться на внутренний. Словно бы этот хост и является обладателем этого белого IP-адреса.

Что происходит:

1. Узел 172.16.6.5 (внутренняя сеть за NAT) обращается WEB-серверу. Он отправляет IP-пакет, где в качестве адреса получателя стоит 192.0.2.2 (внешняя сеть), а отправителя 172.16.6.5.

2. По корпоративной сети пакет доставляется к шлюзу 172.16.6.1, где и настроен NAT.

3. Согласно настроенной команде, маршрутизатор снимает текущий заголовок IP и меняет его на новый, где в качестве адреса отправителя уже фигурирует белый адрес 198.51.100.2.

4. По большому Интернету обновлённый пакет достигает сервера 192.0.2.2.

5. Тот видит, что ответ надо слать на 198.51.100.2 И подготавливает ответный IP-пакет. В качестве адреса отправителя собственно адрес сервера 192.0.2.2, адрес назначения – 198.51.100.2

6. Пакет обратно летит через Интернет, причём не факт, что тем же путём.

7. На натирующем устройстве указано, что все запросы на адрес 198.51.100.2 нужно перенаправлять на 172.16.6.5. Маршрутизатор снова раздевает спрятанный внутри TCP-сегмент и задаёт новый IP-заголовок (адрес отправителя не меняется, адрес назначения 172.16.6.5).

8. По внутренней сети пакет возвращается инициатору, которому даже и невдомёк, какие чудеса с ним творились на границе.

И так будет с каждым.

При этом если соединение инициируется из Интернета, пакеты автоматически, проходя через натирующее устройство, попадают на внутренний хост.

Такой подход бывает полезным, когда у вас есть сервер внутри вашей сети, к которому необходим полный доступ извне. Разумеется, этот вариант вы не можете использовать, если хотите триста хостов выпустить в Интернет через один адрес. Такой вариант NAT'а никак не поможет сохранить белые IP-адреса, но тем не менее он бывает полезен.

Динамический NAT.

У вас есть пул белых адресов, например, провайдер выделил вам сеть 198.51.100.0/28 с 16-ю адресами. Два из них (первый и последний) – адрес сети и широковещательный, ещё два адреса назначаются на оборудование для обеспечения маршрутизации. 12 оставшихся адресов вы можете использовать для NAT'а и выпускать через них своих пользователей.

Ситуация похожа на статический NAT: один приватный адрес транслируется на один внешний, но теперь внешний не чётко зафиксирован, а будет выбираться динамически из заданного диапазона.

Этот вариант тоже не универсальный, своих 300 пользователей вы так же не сможете выпустить всех в Интернет, если у вас нет 300 внешних адресов. Как только белые адреса исчерпаются, никто новый уже не сможет получить доступ в Интернет. При этом те пользователи, что уже успели отхватить себе внешний адрес, будут работать.

Помимо динамического выделения внешних адресов, этот динамически NAT отличается от статического тем, что без отдельной настройки проброса портов уже невозможно внешнее соединение на один из адресов пула.

Many-to-One.

Этот тип имеет несколько названий: NAT Overload, Port Address Translation (PAT), IP Masquerading, Many-to-One NAT.

Последнее название говорит само за себя: через один внешний адрес выходит в мир много приватных. Это позволяет решить проблему с нехваткой внешних адресов и выпустить в мир всех желающих.

Тут надо бы дать пояснение, как это работает. Как два приватных адреса транслируются в один можно представить, но как маршрутизатор понимает кому нужно переслать пакет, вернувшийся из Интернета на этот адрес?

Всё очень просто:

Предположим, что от двух хостов из внутренней сети приходят пакеты на натирующее устройство. Оба с запросом к WEB-серверу 192.0.2.2.

Данные от хостов выглядят так:

Адрес отправителя	Порт отправителя	Адрес получателя	Порт получателя
172.16.6.5	23761	192.0.2.2	80
172.16.4.5	39800	192.0.2.2	80

Маршрутизатор расчехляет IP-пакет от первого хоста, извлекает из него TCP-сегмент, распечатывает его и узнаёт, с какого порта устанавливается соединение. У него есть внешний адрес 198.51.100.2, на который будет меняться адрес из внутренней сети.

Далее он выбирает свободный порт, например, 11874. И что он делает дальше? Все данные уровня приложений он упаковывает в новый TCP сегмент, где в качестве порта назначения по-прежнему остаётся 80 (именно на него ждёт коннектов WEB-сервер), а порт отправителя меняется с 23761 на 11874. Этот TCP-сегмент инкапсулируется в новый IP-пакет, где меняется IP-адрес отправителя с 172.16.6.5 на 198.51.100.2.

То же самое происходит для пакета от второго хоста, только выбирается следующий свободный порт, например 11875. «Свободный» означает, что он ещё не занят другими такими соединениями.

Данные, которые отправляются в интернет, теперь буду выглядеть так:

Адрес отправителя	Порт отправителя	Адрес получателя	Порт получателя
198.51.100.2	11874	192.0.2.2	80
198.51.100.2	11875	192.0.2.2	80

В свою NAT-таблицу он заносит данные отправителей и получателей:

Локальный адрес отправителя	Локальный порт отправителя	Глобальный адрес отправителя	Глобальный порт отправителя	Адрес получателя	Порт получателя
172.16.6.5	23761	198.51.100.2	11874	192.0.2.2	80
172.16.4.5	39800	198.51.100.2	11875	192.0.2.2	80

Для WEB-сервера – это два совершенно разных запроса, которые он должен обработать каждый индивидуально. После этого он отсылает ответ, который выглядит так:

Адрес отправителя	Порт отправителя	Адрес получателя	Порт получателя
192.0.2.2	80	198.51.100.2	11874
192.0.2.2	80	198.51.100.2	11875

Когда один из этих пакетов доходит до нашего маршрутизатора, тот сопоставляет данные в этом пакете со своими записями в NAT-таблице. Если совпадение найдено, происходит обратная процедура – пакету и TCP сегменту возвращаются его изначальные параметры только в качестве назначения:

Адрес отправителя	Порт отправителя	Адрес получателя	Порт получателя
192.0.2.2	80	172.16.6.5	23761
192.0.2.2	80	172.16.4.5	39800

И теперь пакеты доставляется по внутренней сети компьютерам-инициаторам, которым и невдомёк даже, что где-то с их данными так жёстко обошлись на границе.

Каждое ваше обращение – это отдельное соединение. То есть попытались вы открыть WEB-страницу – это протокол HTTP, использующий порт 80. Для этого ваш компьютер должен установить TCP-сессию с удалённым сервером. Такая сессия (TCP или UDP) определяется двумя сокетами: локальный IP-адрес: локальный порт и удалённый IP-адрес: удалённый порт. В обычной ситуации у вас устанавливается одно соединение компьютер-сервер, в случае же NATa соединения будет как бы два:, маршрутизатор-сервер и компьютер думает, что у него есть сессия компьютер-сервер.

Перенаправление портов.

Иначе говорят ещё проброс портов или mapping.

Когда мы только начали говорить про NAT, трансляция у нас была один-в-один и все запросы, приходящие извне автоматически перенаправлялись на внутренний хост. Таким образом можно было бы выставить сервер наружу в Интернет.

Но если у вас нет такой возможности — вы ограничены в белых адресах, или не хотите выставлять всем пучком портов его наружу, что делать?

Вы можете указать, что все запросы, приходящие на конкретный белый адрес и конкретный порт маршрутизатора, должны быть перенаправлены на нужный порт нужного внутреннего адреса.

Например, может оказаться полезным, если у вас есть два компьютера, к которым нужен доступ по RDP извне. RDP использует порт 3389. Один и тот же порт вы не можете пробросить на разные хосты (при использовании одного внешнего адреса). Поэтому можно сделать так, что чтобы попасть на компьютер 172.16.6.61 вы запускаете RDP-сессию на порт 198.51.100.2:3389, а на 172.16.6.66 — 198.51.100.2:3398. Маршрутизатор сам раскидает всё, куда надо.

Основные функции NAT:

1. Позволяет сэкономить IP-адреса. Собственно для этого он и был создан. Через один адрес, теоретически можно выпустить больше 65000 серых адресов (по количеству портов).

2. Позволяет предотвратить или ограничить обращение снаружи ко внутренним хостам, оставляя возможность обращения изнутри наружу. PAT и динамический NAT является в какой-то степени файрволом, препятствуя внешним соединениям доходить до конечных компьютеров, на которых может не оказаться своего файрвола и антивируса. Дело в том, что если извне на натирующее устройство приходит пакет, который тут не ожидается или не разрешён, он просто отбрасывается.

3. Позволяет скрыть определённые внутренние сервисы внутренних хостов/серверов, а также внутреннюю структуру вашей сети – при трассировке маршрута извне вы не увидите ничего далее натирующего устройства.

Минусы NAT:

1. **Старые протоколы.** Протоколы, разработанные до массового внедрения NAT, не в состоянии работать, если на пути между взаимодействующими хостами есть трансляция адресов. Некоторые межсетевые экраны, осуществляющие трансляцию IP-адресов, могут исправить этот недостаток, соответствующим образом заменяя IP-адреса не только в заголовках IP, но и на более высоких уровнях (например, в командах протокола FTP).

2. **Иллюзия DoS-атаки.** Другая проблема кроется в том, с одного адреса идёт много запросов на один сервер. Многие были свидетелем этого, когда заходишь на какой-нибудь Rapidshare, а он говорит, что с вашего IP уже было соединение, вы думаете, что «врёт, собака», а это ваш сосед уже сосет. По этой же причине бывали проблемы с ICQ, когда сервера отказывали в регистрации.

3. **Нагрузка.** Не очень актуальная сейчас проблема: нагрузка на процессор и оперативную память. Поскольку объём работы довольно велик по сравнению с простой маршрутизацией (это надо не просто глянуть заголовок IP, надо его снять, TCP-заголовок снять, в таблицу занести, новые заголовки прикрутить) в мелких конторах с этим бывают проблемы.

4. **Пиринговые сети.** В NAT-устройствах, не поддерживающих технологию Universal Plug & Play, в некоторых случаях, необходима дополнительная настройка (см. Трансляция порт-адрес) при работе с пиринговыми сетями и некоторыми другими программами, в которых необходимо не только инициировать исходящие соединения, но также принимать входящие.

59 NAT: основные типы.

Классификация NAT. Термин «соединение» использован в значении «последовательный обмен пакетами UDP».

Cone NAT, Full Cone NAT – однозначная (взаимная) трансляция между парами «внутренний адрес: внутренний порт» и «публичный адрес: публичный порт». Любой внешний хост может инициировать соединение с внутренним хостом (если это разрешено в правилах межсетевого экрана).

Address-Restricted cone NAT, Restricted cone NAT – постоянная трансляция между парой «внутренний адрес: внутренний порт» и «публичный адрес: публичный порт». Любое соединение, инициированное с внутреннего адреса, позволяет в дальнейшем получать ему пакеты с любого порта того публичного хоста, к которому он отправлял пакет(ы) ранее.

Port-Restricted cone NAT – трансляция между парой «внутренний адрес: внутренний порт» и «публичный адрес: публичный порт», при которой входящие пакеты проходят на внутренний хост только с одного порта публичного хоста — того, на который внутренний хост уже посылал пакет.

Симметричный NAT (Symmetric NAT) – трансляция, при которой каждое соединение, инициируемое парой «внутренний адрес: внутренний порт» преобразуется в свободную уникальную случайно выбранную пару «публичный адрес: публичный порт». При этом инициация соединения из публичной сети невозможна.

В первых трех типах NATа разные IP-адреса внешней сети могут взаимодействовать с адресом из локальной сети используя один и тот же внешний порт. Четвертый тип для каждого адреса и порта использует отдельный внешний порт.

Полный конус (Full Cone).

При использовании NAT'а работающего по типу полного конуса внешний отображаемый порт открыт для пакетов приходящих с любых адресов. Если кто-то из внешнего Интернета хочет в этот момент отправить пакет клиенту, расположенному за НАТом, то ему нужно знать только внешний порт через который установлено соединение. Например, компьютер за NATом с IP-адресом 10.0.0.1 посылает и получает пакеты через порт 8000, отображающийся на внешний IP-адрес и порт 212.23.21.25:12345, то любой в Интернете может послать пакеты на этот 212.23.21.25:12345, и эти пакеты попадут на клиентский компьютер 10.0.0.1:8000.

Ограниченный конус (Restricted Cone).

NAT, с ограниченным конусом, открывает внешний порт сразу после того как локальный компьютер отправит данные на определенный внешний IP-адрес. Например, если клиент посылает наружу пакет внешнему компьютеру 1, NAT отображает клиента 10.0.0.1:8000 на 212.23.21.25:12345, и внешний компьютер 1 может посылать пакеты назад по этому назначению. Однако, NAT будет блокировать пакеты идущие от компьютера 2, до тех пор пока клиент не пошлет пакет на IP-адрес этого компьютера. Когда он это сделает, то оба внешних компьютера 1 и 2 смогут посылать пакеты назад клиенту, и оба будут иметь одно и то же отображение через НАТ.

Порт ограниченного конуса (Port Restricted Cone).

NAT с портом ограниченного конуса почти идентичен NATу с ограниченным конусом. Только в этом случае, NAT блокирует все пакеты, если клиент предварительно не послал наружу пакет на IP-адрес и порт того компьютера, который посылает пакеты клиенту. Поэтому, если клиент посылает внешнему компьютеру 1 на порт 5060, то NAT только тогда пропустит пакет к клиенту, когда он идет с 212.33.35.80:5060. Если клиент послал наружу пакеты к нескольким IP-адресам и портам, то они могут ответить клиенту на один и тот же отображенный IP-адрес и порт.

Симметричный (Symmetric).

Симметричный NAT кардинально отличается от первых трех в способе отображения внутреннего IP-адреса и порта на внешний адрес и порт. Это отображение зависит от IP-адреса и порта компьютера, которому предназначен посланный пакет. Например, если клиент посылает с адреса 10.0.0.1:8000 компьютеру 1, то он может быть отображен как 212.23.21.25:12345, в тоже время, если он посылает с того же самого порта (10.0.0.1:8000) на другой IP-адрес, он отображается по-другому (212.23.21.25:12346).

Компьютер 1 может отправить ответ только на 212.23.21.25:12345, а компьютер 2 может ответить только на 212.23.21.25:12346. Если любой из них попытается послать пакеты на порт с которого он не получал пакеты, то эти пакеты будут игнорированы. Внешний IP-адрес и порт открывается только тогда, когда внутренний компьютер посылает данные наружу по определенному адресу.

60 Устройство ЦОД. Понятие облачных вычислений. Виртуализация и масштабирование.

В настоящее время, интерес к вопросам построения эффективных Центров Обработки Данных (ЦОД) возрос во всем мире.

Центры обработки данных (ЦОД) – это отказоустойчивая комплексная централизованная система, обеспечивающая автоматизацию бизнес-процессов с высоким уровнем производительности и качеством предоставляемых сервисов. В настоящее время в коммерческой среде наряду с определением Центр обработки данных (ЦОД) используется термин – Дата Центр (ДЦ).

Концепция построения инженерных систем ЦОД.

ЦОД представляет собой технологическое помещение, и в зависимости от направленности обслуживаемой организации, содержит различное «активное» комплектующее оборудование. Корпоративные ЦОД вмещают системы хранения и обработки данных, а так же сетевое и телекоммуникационное оборудование.

В ЦОД операторов связи и сервис – провайдеров располагаются средства связи и другие устройства, необходимые для предоставления абонентских услуг.

Необходимость поддержания благоприятного режима функционирования высокотехнологического современного «активного» оборудования ставит перед создателями инженерных решений проблему организации систем жизнеобеспечения ЦОД.

Гарантом общей работоспособности ЦОД, от которой зависит функционирование всей информационной системы в целом, является правильно организованная инженерная инфраструктура. Современный Центр представляет собой интеллектуальное здание в миниатюре. Все инженерные системы ЦОД интегрированы в единый комплекс, с условием непременного учета возможных будущих изменений.

Обновление телекоммуникационного оборудования происходит значительно быстрее изменений инженерной инфраструктуры. Учитывая это, необходимы решения, обладающие адаптивной структурой, позволяющей с легкостью производить установку/замену оборудования новой конфигурации и расширять инфраструктуру ЦОД.

Основные задачи, решаемые установленными в ЦОД инженерными системами, можно разделить на три группы:

- **Обеспечение функционирования технологического оборудования:** системы электроснабжения, вентиляции и кондиционировании, Структурированные Кабельные Системы (СКС).
- **Защита от технических сбоев:** системы автоматического оповещения и тушения пожара, система автоматизации и диспетчеризации.
- **Защита от несанкционированных действий человека:** охранная сигнализация, видеонаблюдение, контроль доступа.

В связи с повышенной нагрузкой, которая приходится на ЦОД, все комплектующие и используемые при отделке и строительстве материалы должны проходить обязательное тестирование на соответствие международным стандартам.

Типичный дата-центр состоит из:

- **информационной инфраструктуры,** включающей в себя серверное оборудование и обеспечивающей основные функции дата-центра — обработку и хранение информации;
- **телекоммуникационной инфраструктуры,** обеспечивающей взаимосвязь элементов дата-центра, а также передачу данных между дата-центром и пользователями;
- **инженерной инфраструктуры,** обеспечивающей нормальное функционирование основных систем дата-центра.

Инженерная инфраструктура включает в себя: кондиционирование для поддержания температуры и уровня влажности в заданных параметрах; бесперебойное электроснабжение для автономной работы дата-центра в случаях отключения центральных источников электроэнергии; охранно-пожарную сигнализацию и система газового пожаротушения; системы удаленного IP контроля, управления питанием и контроля доступа.

Некоторые дата-центры предлагают клиентам дополнительные услуги по использованию оборудования по автоматическому уходу от различных видов атак. Команды квалифицированных специалистов круглосуточно производят мониторинг всех серверов. Необходимо отметить, что услуги дата-центров сильно отличаются в цене и количестве услуг. Для обеспечения сохранности данных используются системы резервного копирования. Для предотвращения кражи данных, в дата-центрах используются различные системы ограничения физического доступа, системы видеонаблюдения. В корпоративных (ведомственных) дата-центрах обычно сосредоточено большинство серверов соответствующей организации. Оборудование крепится в специализированных стойках и шкафах. Как правило, в дата-центр принимают для размещения лишь оборудование в стоечном исполнении, то есть в корпусах стандартных размеров, приспособленных для крепления в стойку. Компьютеры в корпусах настольного исполнения неудобны для дата-центров и размещаются в них редко.

Cloud computing – технология распределённой обработки данных, при которой некие масштабируемые информационные ресурсы и мощности предоставляются как сервис для многочисленных внешних клиентов посредством Интернет-технологий.

Концепция облачных вычислений включает в себя совокупность следующих понятий:

- **IaaS** (Infrastructure as a Service или «Инфраструктура как сервис») – компьютерная инфраструктура, как правило, представленная в форме виртуализации. Является услугой в рамках концепции облачной обработки данных.
- **PaaS** (Platform as a Service или «Платформа как сервис») – интегрированная платформа для разработки, развертывания, тестирования и поддержки web-приложений. Представлена в виде сервиса на основе концепции «облачные вычисления». Это IaaS + операционная система и ее API.
- **SaaS** (Software as a service или «ПО как сервис») – представляет собой бизнес-модель лицензионного использования ПО, которая подразумевает разработку и поддержку программного обеспечения поставщиком. Заказчикам же предоставляется возможность его платного использования, как правило, посредством Интернета.
- **DaaS** (Desktop as a Service или «Рабочий стол как сервис») – ещё одна бизнес-модель лицензионного использования программного обеспечения, которая представляет собой немного усовершенствованную модель SaaS, в основном предполагающая использование нескольких сервисов одновременно, необходимых для полноценной работы. Впервые была представлена в начале 2000-х годов.

Кроме вышеперечисленных в рамках концепции облачных вычислений распространены также понятия **Data as a service** и **Everything as a service** («Данные как сервис» и «Все как сервис» соответственно). Оба понятия показывают, что посредством всемирной паутины с использованием Cloud Computing можно удовлетворить любые потребности в обработке информации.

Модели развёртывания.

Частное облако (англ. private cloud) – инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей (например, подразделений одной организации), возможно также клиентами и подрядчиками данной организации. Частное облако может находиться в собственности, управлении и эксплуатации как самой организации, так и третьей стороны (или какой-либо их комбинации), и оно может физически существовать как внутри, так и вне юрисдикции владельца.

Публичное облако (англ. public cloud) – инфраструктура, предназначенная для свободного использования широкой публикой. Публичное облако может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций (или какой-либо их комбинации). Публичное облако физически существует в юрисдикции владельца – поставщика услуг.

Общественное облако (англ. community cloud) – вид инфраструктуры, предназначенный для использования конкретным сообществом потребителей из организаций, имеющих общие задачи (например, миссии, требований безопасности, политики, и соответствия различным требованиям). Общественное облако может находиться в кооперативной (совместной) собственности, управлении и эксплуатации одной или более из организаций сообщества или третьей стороны (или какой-либо их комбинации), и оно может физически существовать как внутри, так и вне юрисдикции владельца.

Гибридное облако (англ. hybrid cloud) – это комбинация из двух или более различных облачных инфраструктур (частных, публичных или общественных), остающихся уникальными объектами, но связанных между собой стандартизованными или частными технологиями передачи данных и приложений (например, кратковременное использование ресурсов публичных облаков для балансировки нагрузки между облаками).

Достоинства облачных вычислений:

- снижаются требования к вычислительной мощности ПК (непременным условием является только наличие доступа в интернет);
- отказоустойчивость;
- безопасность;
- высокая скорость обработки данных;
- снижение затрат на аппаратное и программное обеспечение, на обслуживание и электроэнергию;
- экономия дискового пространства (и данные, и программы хранятся в интернете).

Недостатки облачных вычислений :

- зависимость сохранности пользовательских данных от компаний, предоставляющих услугу cloud computing;
- появление новых («облачных») монополистов.

Для обеспечения согласованной работы узлов вычислительной сети на стороне облачного провайдера используется специализированное связующее программное обеспечение, обеспечивающее мониторинг состояния оборудования и программ, балансировку нагрузки, обеспечение ресурсов для решения задачи.

Одним из основных решений для сглаживания неравномерности нагрузки на услуги является размещение слоя серверной виртуализации между слоем программных услуг и аппаратным обеспечением. В

условиях виртуализации балансировка нагрузки может осуществляться посредством программного распределения виртуальных серверов по реальным, перенос виртуальных серверов происходит посредством живой миграции.

В настоящее время **виртуализация** – это очень востребованная технология. Можно выделить следующие варианты использования продуктов виртуализации: консолидация серверов, разработка и тестирование приложений, использование в бизнесе, использование виртуальных рабочих станций.

Консолидация серверов.

В данный момент приложения, работающие на серверах в IT-инфраструктуре компаний, создают небольшую нагрузку на аппаратные ресурсы серверов (в среднем 5–15 процентов). Виртуализация позволяет мигрировать с этих физических серверов на виртуальные и разместить их все на одном физическом сервере, увеличив его загрузку до 60–80 процентов и, повысив тем самым коэффициент использования аппаратуры, что позволяет существенно сэкономить на аппаратуре, обслуживании и электроэнергии.

Установив в компании один производительный и надежный сервер, можно заменить им несколько маломощных машин, при этом общая стоимость владения таким высокопроизводительным сервером, как правило, оказывается ниже. Правда уменьшится и общая надежность системы.

Стоимость оборудования при использовании более дорогого сервера увеличится, но упростится его техническое сопровождение, а следовательно и общая стоимость владения. В конфигурации двух компьютеров с общим RAID-массивом можно максимально уменьшить время простоя системы после сбоя и перенести виртуальную среду на другой компьютер практически мгновенно. При сбое одного компьютера работа приложений может быть тут же восстановлена на другом – потеряны будут только самые оперативные данные, хранившиеся в кэш-памяти остановившейся машины. Таким образом, консолидация позволяет уменьшать затраты, которые вносят максимальную долю в TCO.

С использованием виртуализации упрощается перенос приложений с одного сервера другой. Этот процесс, как правило, можно выполнить безболезненно – штатной операцией перемещения виртуальной среды. В результате, появляется возможность физически передвигать приложения к месту их оптимального использования. Например, если компания имеет центры обработки данных по всему миру, то с помощью виртуализации она сможет перемещать приложения туда, где они наиболее востребованы.

61 Современные проблемы компьютерных сетей. Программно Конфигурируемые Сети (ПКС): структура, принципы функционирования, протокол Open Flow.

Проблемы современных компьютерных сетей.

Одной из основных движущих сил развития информационных технологий являются компьютерные сети и Интернет, как основополагающая инфраструктура. Однако архитектура глобальной сети Интернет устарела. Ее основы закладывалась в конце 60-х – 70-е годы, когда не было мобильных узлов, беспроводной связи (если не считать спутниковой, которая тогда была в начале своего развития). Развитие микропроцессорной техники и телекоммуникаций (закон Мура, закон Гилдера) кардинально изменили положение, роль и значимость компьютерных сетей в обществе. Сегодня количество пользователей компьютерных сетей на основе беспроводных технологий превышает число пользователей с фиксированной связью, число мобильных терминалов, приходящихся на одного пользователя в развитых странах, более трех. Изменилась и сама парадигма организации вычислений в Обществе: на место клиент-серверной организации вычислений пришли Центры Обработки Данных (ЦОД) и облачные вычисления, файловые системы и базы данных трансформировались в Сети Хранения Данных (СХД).

Эти законы за последние 30 лет привели к качественным изменениям, которые требуют пересмотра сетевой архитектуры. Количество и сложность протоколов огромны (на сегодня число активно используемых протоколов и их версий превысило 600); совмещение управления и передачи данных делают контроль и управление работой сети очень сложными, требующими высококвалифицированных специалистов; вопросы безопасности до сих пор не имеют надежных решений. Внесение любых изменений в

средства построения Сетей трудоемко, дорогостояще, длительно по срокам, не возможно без привлечения производителя. Никогда нельзя гарантировать, что программно-аппаратные средства производителя содержат только ту функциональность, которая описана в штатной документации. В компьютерных и телекоммуникационных сетях ситуация может быть еще сложнее – такая функциональность может быть распределенной. Средства построения сетей сегодня являются проприетарными, закрытыми для инноваций со стороны владельцев сетей, академической общественности.

Тренды и потребности рынка.

Согласно данным Cisco Systems, объем трафика в сети Интернет за последние 5 лет возрос в три раза. На сегодня его ежегодный рост составляет около 100%, т.е. удваивается. При этом к 2014 г. около 80% этого трафика будет составлять видеотрафик. Эти цифры говорят о том, что пропускная способность современных каналов связи при существующих методах и средствах управления трафиком в сетях близка к исчерпанию. Существующие темпы роста пропускной способности сети будут не в состоянии удовлетворять растущие потребности пользователей.

Мобильные беспроводные сети сегодня сталкиваются с двумя противоречивыми тенденциями. Увеличение вычислительной мощности мобильных терминалов влечет увеличение вычислительной емкости приложений, работающих на них. Это в свою очередь ведет к увеличению требований к пропускной способности каналов мобильной связи. На сегодня объем мобильного трафика растет в геометрической прогрессии и становится все более разнообразным. По данным Cisco Systems трафик удваивается примерно каждые девять месяцев, что приведет к увеличению нагрузки на несколько порядков в течение ближайших нескольких лет. В то же время на сегодня эффективность доступного спектра частот (т.е. максимальная пропускная способность достижимая на Гц спектра) близка к насыщению. На самом деле, спектральная эффективность 4G LTE PHY приближается к (в пределах $\approx$ 20%) пределу по Шеннону и дальнейшие улучшения, вероятно, будут очень дороги для осуществления и обеспечат лишь ограниченные выгоды.

Для того чтобы справиться с таким нарастанием трафика, беспроводные сети должны иметь более плотное покрытие: Самый верный способ увеличить для каждого пользователя пропускную способность канала – сделать соту небольшой и приблизить к базовой станции мобильного клиента, так как это улучшает связь и уменьшит количество пользователей в соте. По оценке экспертов, плотность базовых станций надо будет увеличить в 20 раз, чтобы справиться с экспоненциальным ростом трафика.

Однако сегодня сетевая архитектура плохо приспособлена для поддержки такого плотного трафика существующей беспроводной инфраструктурой. Во-первых, поскольку невозможно будет равномерно увеличить плотность покрытия в 20 раз и более, то базовые станции придется развертывать там, где это возможно, т.е. хаотично. Однако такой инфраструктурой будет очень сложно управлять, она будет испытывать очень неравномерные нагрузки, взаимные влияния сот и других факторов. И, наконец, плотная инфраструктура стоит очень дорого в развертывании и эксплуатации. Столь масштабные изменения инфраструктуры мобильной связи могут осуществить только очень крупные операторы, но даже им 20-кратное увеличение плотности покрытия может оказаться не по силам.

Итак, проблемы современных компьютерных сетей можно разделить на:

- **Научные.** Сегодня мы не можем контролировать и надежно предвидеть поведение таких сложных объектов, как глобальные компьютерные сети.
- **Социальные.** В повседневной жизни мы все больше и больше полагаемся на Интернет. Однако, безопасность данных, которые мы ему доверяем, включая наши персональные данные, нам не гарантирована, Интернет не устойчив к внешним атакам.
- **Проблемы развития.** В архитектуре современных сетей есть существенные барьеры для введения инноваций, экспериментирования, создания новых сервисов.

Программно-Конфигурируемые Сети: основные идеи.

В отличие от большинства областей техники, промышленность построения компьютерных сетей за последние двадцать лет практически не претерпела существенных изменений, основная парадигма архитектуры компьютерных сетей остается практически неизменной. В результате, сети все еще слишком

дороги, сложны и ими трудно управлять. Это неудовлетворительное состояние дел может измениться из-за двух революционных событий: (1) появление на рынке чрезвычайно усложненного, проприетарного, сетевого оборудования, и (2) появление принципиально нового подхода, называемого программно-конфигурируемыми сетями (ПКС – Software Defined Networks). ПКС-подход обещает сделать все сети дешевле и проще в управлении. Влияние ПКС-подхода будет ощущаться в центрах обработки данных (дата-центрах), корпоративных сетях, WAN, сотовых сетях, а также и в домашних условиях. ПКС-сети возникли из исследований в Стэнфорде и Беркли, и в настоящее время одобрены более чем четырьмя десятками индустриальных компаний через их членство в Фонде Открытых Сетей (Open Network Foundation).

Говоря о ПКС-сетях, акцент делают на плоскость данных – сервисах, связанных с передачей (forwarding) пакетов на основе состояния в каждом коммутаторе, и плоскости управления, которая отвечает за вычисление этого состояния. Хотя плоскость данных стала намного быстрее и более функциональной в течение последних нескольких десятилетий, плоскость управления осталась поразительно примитивной, полагаясь на сложные распределенные алгоритмы маршрутизации и замысловатые инструкции по конфигурированию и настройке сети. Более того, нет никакой единой методологической основы для рассуждений о плоскости управления сетью, поэтому операторы должны полагаться на эвристики и опыт.

Основная идея развиваемого ПКС-подхода состоит в том, чтобы:

1. Отделить управление сетевым оборудованием от управления передачей данных за счет создания специального программного обеспечения, которое может работать на обычном отдельном компьютере и которое находится под контролем администратора сети.

2. Перейти от управления отдельными экземплярами сетевого оборудования к управлению сетью в целом.

3. Создать интеллектуальный, программно-управляемый интерфейс между сетевым приложением и транспортной средой сети.

В основе ПКС сетей лежит представление о компьютерной сети, как сети, имеющей «плоскость данных», которая отвечает за пересылку пакетов на основе состояния в каждом коммутаторе, и «плоскости управления», которая отвечает за вычисление, «планирование» и управление пересылкой. Для реализации этой идеи был разработан открытый протокол Open Flow для управления сетевым оборудованием, не ориентированный на продукты какого-то отдельного поставщика. С помощью этого протокола специалисты сами могут определять и контролировать: кто с кем, при каких условиях и с каким качеством может взаимодействовать в Сети. Все маршрутизаторы и коммутаторы объединяются под управлением Сетевой Операционной Системы (СОС), которая обеспечивает приложениям доступ к управлению сетью и которая постоянно отслеживает конфигурацию средств Сети. В отличие от традиционного толкования термина СОС как операционной системы интегрированной со стеком сетевых протоколов, в данном случае под СОС понимается программная система, обеспечивающая мониторинг, доступ, управление, ресурсами всей сети, а не конкретного узла.

На самом общем уровне ПКС включает в себя только одну распределенную систему, «сетевую операционную систему», которая формирует данные о состоянии всех ресурсов сети и обеспечивает доступ к ним для приложений управления сетью. Эти приложения управления могут быть логически централизованными, работающими на едином графовом представлении сети. Чтобы избежать зависимости от конкретного сетевого оборудования, ПКС использует общие абстракции для пересылки пакетов, которые сетевая операционная система использует для управления сетевыми коммутаторами.

Такой подход должен позволить строго анализировать и рассуждать об управлении сетью – как в настоящее время мы можем действовать и рассуждать о вычислениях и о хранении данных – а не просто полагаться на интуицию и накопленный годами опыт работы. Подобная строгость в рассуждениях и анализе, позволяет в то же время сетям оставаться простыми для понимания и эксплуатации.

62 Протокол Open Flow, организация и принципы работы ПКС коммутатора, маршрутизация в ПКС сетях.

Openflow – протокол управления процессом обработки данных, передающихся по сети передачи данных маршрутизаторами и коммутаторами, реализующий технологию программно-конфигурируемой сети.

Протокол используется для управления сетевыми коммутаторами и маршрутизаторами с центрального устройства — контроллера сети (например, с сервера или даже персонального компьютера). Это управление заменяет или дополняет работающую на коммутаторе (маршрутизаторе) встроенную программу, осуществляющую построение маршрута, создание карты коммутации и т.д.. Контроллер используется для управления таблицами потоков коммутаторов, на основании которых принимается решение о передаче принятого пакета на конкретный порт коммутатора. Таким образом в сети формируются прямые сетевые соединения с минимальными задержками передачи данных и необходимыми параметрами.

Архитектура.

Путь прохождения данных (datapath) состоит из таблицы потоков (flow table) и действий, назначенных для каждой записи в таблице. Сами таблицы могут касаться как Ethernet (или других протоколов канального уровня), так и протоколов вышестоящих уровней (IP, TCP). Точный список действий может меняться, но основные это: форвардинг (пересылка фрагмента данных – пакета, фрейма – в заданный порт), пересылка фрагмента данных на контроллер через безопасный канал для дальнейшего исследования, отбрасывание фрагмента данных (drop). Для устройств, совмещающих openflow и обычную обработку пакетов средствами микропрограммы устройства, добавляется четвёртый тип действия: обработка фрагмента данных обычными средствами. Оборудование, поддерживающее эти четыре действия являются Type0-устройствами.

Устройство OpenFlow состоит, как минимум, из трёх компонент:

- Таблицы потоков (англ. flow table).
- Безопасного канала (англ. secure channel), использующегося для управления коммутатором внешним «интеллектуальным» устройством (контроллером).
- Поддержки протокола OpenFlow protocol, использующегося для управления. Использование этого протокола позволяет избежать необходимости писать программу для управляемого устройства.

Каждая запись в таблице потоков имеет три поля: заголовок PDU (фрагмента данных), который позволяет определить соответствие PDU потоку, действие и поле со статистикой (число байтов и PDU, соответствующее потоку, время, прохождения последнего соответствующего потоку PDU).

Заголовок может состоять из множества полей разного уровня (например, MAC-адресов отправителя и получателя, полей из заголовка IP-пакета, полей из заголовка TCP-сегмента). Следует отметить, что в текущей версии протокола не поддерживается проверка, к примеру, флага SYN в заголовке TCP-сегмента. Каждое поле может иметь особое значение (астериск), означающее соответствие любому значению соответствующего поля в PDU.

Устройства type1, которые будут обеспечивать трансляцию сетевых адресов, поддержку классов и приоритетов, запланированы, но их спецификация пока не определена.

Контроллеры обеспечивают наполнение таблицы потоков, получение пакетов через безопасный канал от устройства. Могут быть реализованы как простейший алгоритм, напоминающий поведение коммутатора, разделяющего пакеты по логическим сетям (VLAN), а могут реализовывать сложную динамическую логику, влияющую на прохождение пакетов исходя из внешних причин (права доступа, загрузка серверов, приоритеты по обслуживанию и т. д.).

Принцип действия.

В большинстве современных коммутаторов Ethernet используются таблицы потоков, которые описывают, как наиболее эффективно доставить пакет от отправителя к пункту назначения. У каждого поставщика таблица потоков своя, однако можно выделить набор функций, общих для большинства

коммутаторов старшего класса, например качество обслуживания и отчетность по трафику. OpenFlow стандартизует этот общий набор функций.

Разделение тракта данных и тракта управления. Как показано на рисунке, OpenFlow отделяет друг от друга функции тракта данных и тракта управления, традиционно реализуемые коммутаторами. Функциональность, относящаяся к тракту данных, по-прежнему выполняется на коммутаторе, но за принятие решений о высокоуровневой маршрутизации в OpenFlow отвечает контроллер, как правило организованный на базе стандартного сервера. Коммутатор и контроллер общаются по протоколу OpenFlow Switching Protocol. Контроллер может, например, приказать коммутаторам ввести в действие правила для потоков сетевого трафика, объясняет профессор Стэнфорда Ник Макьюэн, разработчик OpenFlow. Такие правила могут, в частности, обеспечивать отправку данных по самым быстрым маршрутам или по маршрутам, имеющим минимум транзитных участков.

Интерфейс. OpenFlow предоставляет единый API, с помощью которого администраторы могут программировать работу сети, а также задавать правила маршрутизации пакетов, балансировки нагрузки и управления доступом. В этот API входят два основных компонента: программный интерфейс для контроля пересылки пакетов через сетевые коммутаторы и набор глобальных интерфейсов, на основе которых можно создавать высокоразвитые инструменты управления.

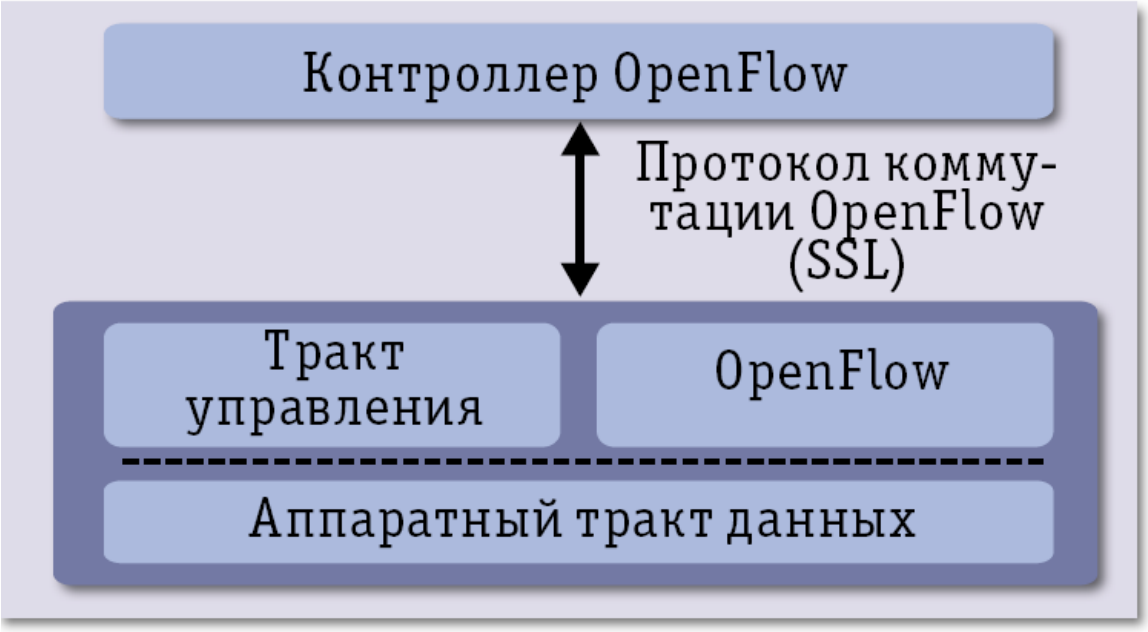


Рис. 12: В сети OpenFlow за принятие решений о высокоуровневой маршрутизации отвечает контроллер, а не локальный коммутатор, как в обычных сетях. Благодаря этому процессорные ресурсы коммутатора можно использовать для более быстрой переадресации пакетов и для решения иных задач. При помощи контроллеров сетевые администраторы могут более эффективно управлять своими сетями.