Билет № 1.

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

Предпосылки возникновения сетей.

1. Появление технически сложных систем.

Новые способы получения энергии открыли новые горизонты в развитии промышленности. Однако создание технически сложных систем, в свою очередь, стало требовать принципиально новых технологий проектирования. Требуется моделирование, специальные методы борьбы со сложностью.

Новые задачи проектирования - новые требования к скоростям расчетов, алгоритмам расчетов. Уже в 40-х годах основным приемом для ускорения ручных расчетов был метод распараллеливания, когда несколько счетчиков работали параллельно над одним и тем же расчетом.

Просто создать объект было мало. В современных условиях его надо уметь поддерживать в работоспособном состоянии, ремонтировать при выходе из строя, что требует, в свою очередь, развития методов и средств управления.

2. Необходимость быстрого получения информации.

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

A. Демографический рост.

B. Территориальная децентрализация населения.

C. Рост числа людей, вовлекаемых в процесс принятия решений.

При развитии данных тенденций отсутствие эффективных методов коммуникации, распределенного доступа к информации, ее автоматического сбора, обработки и хранения тормозили развитие экономики - как на внутригосударственном, так и на межгосударственном уровне.

Таким образом, в XX веке стало ясно: развитие информационных технологий напрямую связано с конкурентоспособностью - как отдельного предприятия, так и государства в целом.

Как следствие первой названной нами причины совершенствуются методы обработки информации: от механических арифмометров, табуляторов с управлением от перфокарт (1928) до ЭВМ. Возможности компьютеров, их размеры, технические характеристики растут с фантастической скоростью. Появляются устройства памяти объемом в несколько терабайт.

Вторая причина обуславливает изменение технологий сбора и передачи информации. Появляются телефонные сети всемирного масштаба, теле- и радиосети, спутники связи.

Одновременно с этим технологии сбора, передачи, обработки и хранения информации начинают сливаться, интегрироваться в единый комплекс. Слияние компьютеров со средствами передачи данных коренным образом изменило представление об организации вычислительных систем. Появились сети ЭВМ. Правда, в первых сетях ЭВМ средства связи использовались именно для соединения между собой ЭВМ. Это не снимало проблему передачи информации их телефонной

сети в сеть ЭВМ или из радио сети в сеть ЭВМ. Возникла проблема интеграции разнородных сетей между собой.

Краткая история развития ЭВМ и методов доступа к ним.

Способы доступа к вычислительным установкам

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

Виды вычислительных установок

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

В нашем курсе под термином сети ЭВМ мы будем понимать множество соединенных между собой автономных машин.

Часто возникает путаница между распределенными системами и сетями ЭВМ. Работая с распределенной системой, пользователь может не иметь ни малейшего представления, на каких процессорах, где, с использованием каких физических ресурсов будет исполняться его программа. В сети, поскольку все машины автономны, пользователь должен делать все явно. Основное различие между этими системами лежит в том, кто и как управляет ресурсами в системе, и, как следствие, в том, как организовано их программное обеспечение. В обоих случаях происходит передача информации. Вопрос в том, кто ее инициирует. В сети - пользователь, в распределенной системе – система.

Современные тенденции

- Различия между ЛВС и РВС по скорости передачи стираются.
- Организация работы с внешней памятью – подсеть в сети.
- Возврат к мэйнфреймам – один слон дешевле десятка тяжеловозов.
- Цифра, звук, видео – интеграция разнородных потоков информации.

Кто и для чего использует сеть ЭВМ.

1. Сети для организаций.

Приведем основные преимущества, которые получают организации, используя сети.

- управление ресурсами
- повышение надежности функционирования предприятия за счет оперативности управления и использования имеющихся ресурсов.
- сокращение затрат на функционирование предприятия – оптимизация бизнес-процессов.
- повышение экономической эффективности за счет гибкой организации работы информационных систем (отсутствие складов, принятие решений)
- средство общения и связи (телеконсультации и конференции, оперативность принятия решений)

• офис в кармане – позволяет сотрудникам получить доступ ко всем устройствам, файлам, базам данных и т.п. вне зависимости от их физического местоположения

• удобства при подготовке персонала (в некоторых крупных западных фирмах стоимость подготовки вновь принятого сотрудника достигает 50 000 долларов).

• управление производством и стратегией развития (ERP-системы ЦБ РФ, FedExp, GM склад, Газпром)

2. Сети для индивидуальных пользователей.

Преимущества использования сетей для индивидуальных пользователей:

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

Сциальное влияние сетей:

- информационная открытость общества
- непрерывность образования
- общедоступность знаний
- взаимодействие с государством
- нанесение ущерба репутации людей
- сеть не знает государственных границ
- использование ресурсов организации в личных целях
- анонимки
- информационная система как наркотик

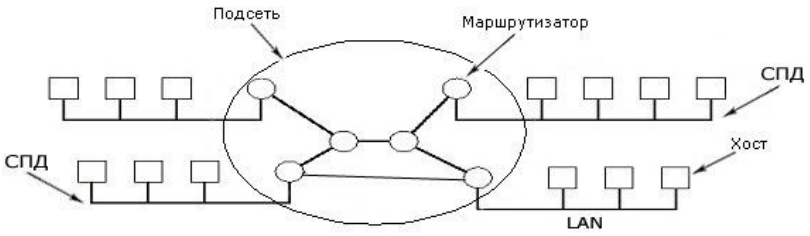
Организация вычислительных сетей.

Все оборудование сети можно разделить на абонентские машины и транспортную среду или транспортную подсеть, которую иногда называют просто подсеть.

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

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

Рисунок 1-1. Сопряжение транспортных сред



Система передачи данных обеспечивает передачу данных между машинами в сети. Эти машины не обязательно абонентские. Система передачи данных состоит из каналов, каналообразующей аппаратуры, коммутирующих элементов (например, коммутаторов).

Каналы передачи данных – это линии связи самой различной природы и каналообразующая аппаратура. Каналы соединяют две или более коммуникационных машины, либо коммутатора. Как уже было сказано, эти машины, обеспечивают коммутацию потоков данных в сети.

Каналообразующая аппаратура обеспечивает интерфейс между линией связи и оконечным устройством, например, А-машиной, К-машиной или коммутатором.

На сегодня отсутствует общепринятая классификация систем передачи данных. Для них, как и для сетей, есть три общепризнанных критерия, по которому их различают, это:

- Способ коммутации потоков данных
- Тип каналов
- Топология системы

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

Коммутация каналов – метод управления потоком данных в реальном времени, для которого характерно следующее:

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

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

- неэффективным использованием ресурсов
- низкой надежностью
- медленным установлением соединения

Коммутация пакетов - способ управления передачей, отличающийся следующими особенностями:

- Высокая скорость установления соединения (передатчик сразу начинает передачу и не ждет физического установления соединения).
- Низкий уровень ошибок в канале.

- Надежность.
- Рациональное использование ресурсов.
- Сильная зависимость времени передачи от загрузки сети.

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

- Каналы «точка-точка». Они соединяют между собой только две машины. Все потоки данных, протекающие по каналу этого типа, доступны лишь этим двум машинам.
- Каналы с множественным доступом. Образуют линию передачи данных, общую для нескольких машин. Короткое сообщение, называемое пакетом, отправленное какой-либо машиной и имеющее специальную структуру, получают все другие машины, подключенные к этой линии. В определенном поле пакета указан адрес получателя. Каждая машина проверяет это поле. Если она обнаруживает в этом поле свой адрес, то она приступает к обработке пакета, в противном случае она просто игнорирует пакет.

Системы передачи данных (СПД) с каналами с множественным доступом, как правило, имеют режим, когда один пакет адресуется всем машинам в сети. Это так называемый режим широкого вещания. Есть в таких сетях режим группового вещания: один и тот же пакет получают машины, принадлежащие к определенной группе в сети.

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

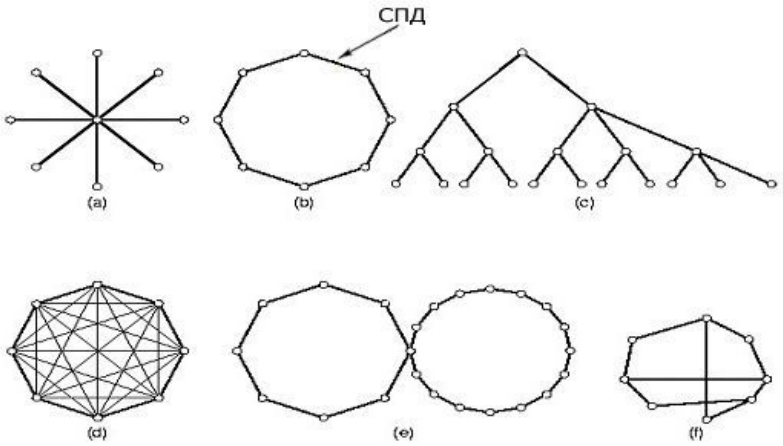
- Статические - временное разделение (time-shearing) канала между машинами; канал простаивает, если машине нечего передавать.
- Динамические - централизованные и распределенные механизмы выделения канала по запросу.

СПД с каналами «точка-точка» соединяют каждую пару машин индивидуальным каналом. Прежде чем пакет достигнет адресата, он может пройти через несколько промежуточных машин. В этих сетях возникает потребность в маршрутизации пакетов. От ее эффективности зависит скорость доставка сообщений, распределение нагрузки в сети.

Системы СПД с каналами с множественным доступом, как правило, используют на географически небольших территориях, СПД с каналами «точка-точка» - для построения крупных сетей, охватывающих большие регионы.

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

Рисунок 1-2. Топологии транспортной среды для соединений «точка-точка»: (a) Звезда; (b) Кольцо; (c) Дерево; (d) Полносвязная; (e) Пересекающиеся кольца; (f) Нестандартная



Абонентские машины обеспечивают работу приложений. Такая машина обязательно содержит средства сопряжения и взаимодействия с СПД. «Чистых» абонентских машин не бывает. А-машина обязательно содержит аппаратуру подключения к каналу и программное обеспечение, организующее прием и передачу данных по этому каналу.

Не следует путать клиент-серверную организацию работы приложений в сети с собственно сетью. В крайнем случае и клиент и сервер могут располагаться на одном и том же компьютере, который будет являться А-машиной в сети. Клиент и сервер - это взгляд на распределение функций в прикладной системе и к сетям, вообще говоря, отношения не имеет.

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

Мосты и шлюзы – средства сопряжения транспортных сред (ТС) на разных уровнях.

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

Шлюз - машина с надлежащим программным обеспечением, обеспечивающая связь между разнородными сетями и необходимое форматирование передаваемых данных.

Множество соединенных сетей называется internet. Примером internet может служить набор LAN, соединенных через WAN. Нельзя путать и internet и Интернет.

Классификация сетей ЭВМ.

На сегодня нет общепризнанной таксономии сетей. Есть два общепризнанных фактора для их различения: технология передачи данных и масштаб. Технология передачи определяется системой СПД.

Масштаб сети - другой критерий для классификации сетей.

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

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

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

Городская вычислительная сеть (MAN - Metropolitan Area Network) охватывает несколько зданий в пределах одного города либо город целиком. Как правило, она поддерживает передачу как данных, так и голоса и иногда объединяется с кабельной телевизионной сетью. Не имеет коммутаторов, базируется на одном или двух кабелях.

Основная причина выделения этой категории сетей состоит в том, что для них был создан специальный стандарт IEEE 802.6 - DQDB - двойная магистраль с распределенной очередью (Distributed Queue Dual Bus). Организация этой СПД показана на рисунке 1-6.

Рисунок 1-6. Архитектура DQDB-сети городского масштаба

Региональная вычислительная сеть (WAN - Wide Area Network) охватывает крупные географические области, такие как страны, континенты.

Транспортная среда таких сетей строится на основе коммутации пакетов с помощью каналов типа «точка-точка». Часто в качестве СПД в таких сетях используют уже существующие системы связи, например, телефонные сети, спутниковые и радиосистемы.

Программное обеспечение сетей ЭВМ.

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

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

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

Уровень n на одной машине непосредственно с уровнем n на другой машине не взаимодействует. Он передает данные нижележащему уровню. Таким образом, возникает как бы два вида взаимодействия: виртуальное – между одноименными уровнями в сети, и непосредственное – между парой соседних уровней.

Между каждой парой уровней есть интерфейс. Интерфейс определяет, какие услуги (сервис) нижележащий уровень должен обеспечивать для верхнего уровня, и с помощью каких примитивов - элементарных операций – верхний уровень может получить доступ к этим услугам. Интерфейс обеспечивает вышележащему уровню доступ к сервису нижележащего уровня.

Набор уровней и протоколов называется архитектурой сети. Описание архитектуры сети должно содержать достаточно информации, чтобы разработчик сетевого программного обеспечения мог разработать надлежащие программы для каждого уровня, а инженер электронщик - надлежащую аппаратуру. Ни вопросы реализации, ни определения интерфейсов не относятся к архитектуре сети.

Конкретный набор протоколов, используемый на конкретной машине, называется стеком протоколов.

При передаче сообщения между уровнями над сообщением выполняются определенные преобразования. Здесь на каждом уровне к сообщению добавляется заголовок. Заголовок содержит управляющую информацию - кому адресовано сообщение, время, дату, порядковый номер и т.д. (На уровне n исходное сообщение уровня n-1 разбивается на два, если длина сообщения уровня n-1 с заголовком уровня n превышает заранее определенную допустимую для передачи на нижележащем уровне длину.)

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

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

Активные элементы уровня, т.е. те, которые могут сами совершать действия, в отличие от тех, над которыми совершают действия, будем называть активностями. Активности могут быть программными и аппаратными. Активности одного и того же уровня на разных машинах будем

называть равнозначными или одноименными активностями. Активности уровня n+1 являются пользователями сервиса, создаваемого активностями уровня n, которые, в свою очередь, называются поставщиками сервиса. Сервис может быть разного качества, например, быстрая и дорогостоящая связь или медленная и дешевая.

Доступ к сервису осуществляется через так называемые точки доступа к сервису - SAP (Service Access Points). Каждая точка доступа к сервису имеет уникальный адрес. Например, телефонная розетка на стене - это точка доступа к сервису АТС. Каждой розетке сопоставлен определенный номер - номер телефона.

Чтобы осуществить обмен информацией между двумя уровнями, нужно определить интерфейс между ними. Типичный интерфейс: активность на уровне n+1 передает IDU (Interface Data Unit - интерфейсную единицу данных) активности на уровне n через SAP. IDU состоит из SDU (Service Data Unit - сервисной единицы данных) и управляющей информации. SDU передается по сети равнозначной сущности, а затем - на уровень n+1. Управляющая информация нужна нижележащему уровню, чтобы правильно передать SDU, но она не является частью передаваемых данных.

Чтобы передать SDU по сети нижележащему уровню, может потребоваться разбить его на части. Каждая часть снабжается заголовком (header) и концевиком, и передается как самостоятельная единица данных протокола - PDU (Protocol Data Unit - единица данных протокола). Заголовок в PDU используется протоколом при передаче. В нем указано, какой PDU содержит управляющую информацию, а какой - данные, порядковый номер и т.д.

Уровни могут предоставлять вышележащим уровням два вида сервисов: ориентированный на соединение и без соединения.

Сервис с соединением предполагает, что между получателем и отправителем сначала устанавливается соединение, и только потом доставляется сервис. Пример - телефонная сеть.

Сервис без соединения действует подобно почтовой службе. Каждое сообщение имеет адрес получателя. В надлежащих точках оно маршрутизируется по нужному маршруту, независимо от других сообщений. При таком сервисе вполне возможно, что сообщение, позже посланное, придет раньше. В сервисе с соединением это невозможно.

Надежный сервис с соединением имеет две разновидности: с сохранением структуры передаваемых данных, например, последовательность сообщений, и просто поток байтов. В первом случае четко различаются границы каждого сообщения. В случае потока байтов получатель не сможет распознать, то ли это 2 сообщения по 1 Мбайт, то ли 1 сообщение в 2 Мбайт, то ли 2048 сообщений по 1 байту.

Другой важной характеристикой качества сервиса является величина задержки в канале. Для некоторых приложений задержки из-за уведомления получения данных неприемлемы. Примерами таких приложений являются цифровая телефонная связь, цифровые видеоконференции.

Пример приложения, не требующего соединения, - электронная почта. Ненадежный сервис (т.е. без уведомления) часто называют дейтаграммным (datagram), по аналогии с телеграммой без уведомления. Однако для тех приложений, где необходима гарантия доставки даже небольшого сообщения, используется дейтаграмм-сервис с подтверждением, подобно телеграмме с уведомлением о получении.

Другой разновидностью дейтаграммного сервиса является сервис «запрос-ответ». Он типичен для взаимодействия между клиентами и сервером.

Формально сервис можно описать в терминах примитивных операций, или примитивов, с помощью которых пользователь или какая-либо активность получает доступ к сервису. С помощью этих примитивов активность на вышележащем уровне сообщает активности на нижележащем уровне,

что необходимо сделать, чтобы вышележащая активность получила нужную услугу (сервис). В свою очередь, нижележащая активность может использовать эти примитивы, чтобы сообщить вышележащей активности о действии, выполненном равнозначной активностью. Примитивы можно разделить на четыре класса, как показано в таблице 1-12.

Таблица 1-12. Примитивы сервиса

Примитив	Значение
Request (Запрос)	Пользователь требует от сервиса каких-либо действий.
Indication (Индикация)	Пользователя информируют о каком-либо событии.
Response (Ответ)	Пользователь требует ответа на какое-либо событие.
Confirm (Подтверждение)	Получен ответ на сделанный ранее запрос.

Большинство примитивов имеет параметры. Услуга может быть либо с подтверждением, либо без подтверждения. При услуге с подтверждением действуют все четыре примитива - request, indication, response, confirm. При услуге без подтверждения используются только два примитива - request и indication.

Сервис и протокол не связаны между собой!

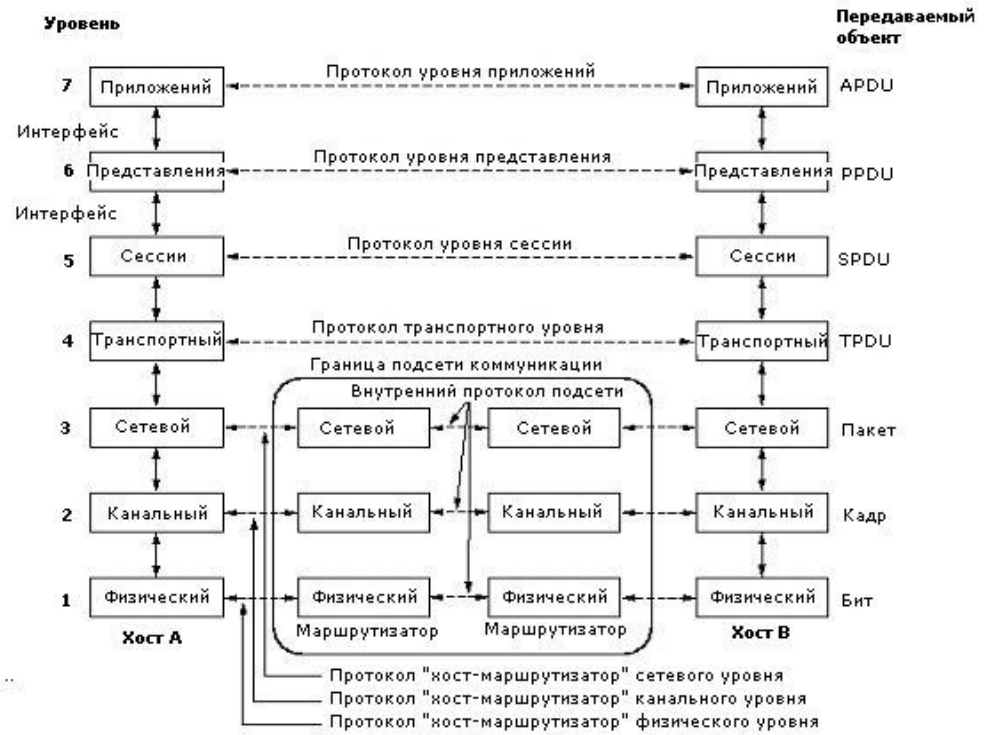
Билет № 2.

Модели сетевого взаимодействия OSI ISO и TCP/IP.

Эталонная модель OSI.

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

Рисунок 1-13. Модель взаимодействия открытых систем (OSI)



Модель МОС имеет семь уровней. Принципы выделения этих уровней таковы:

1. Каждый уровень имеет определенное предназначение.

2. Каждый уровень защищает нижележащий уровень от различий возможных реализаций.

3. Предназначение каждого уровня выбиралось прежде всего так, чтобы для него можно было определить международный стандарт.

4. Границы между уровнями выбирались с целью минимизировать поток информации через интерфейсы.

5. Число уровней выбиралось достаточно большим, чтобы не объединять разные функции на одном уровне, но и достаточно малым, чтобы архитектура не была громоздкой.

Физический уровень отвечает за передачу последовательности битов через канал связи. Одной из основных проблем, решаемых на этом уровне, является то, как гарантировать, что если на одном конце отправили 1, то на другом получили 1, а не 0. На этом уровне также решаются такие вопросы, как: каким напряжением нужно представлять 1, а каким - 0; сколько микросекунд тратится на передачу одного бита; следует ли поддерживать передачу данных в обоих направлениях одновременно; как устанавливается начальное соединение и как оно разрывается; каково количество контактов на физическом разъеме, для чего используется каждый контакт этого разъема. Здесь в основном решаются вопросы механики и электрики.

Основная задача уровня канала данных - превратить несовершенную физическую среду передачи в надежный канал, свободный от ошибок передачи. Эта задача решается разбиением данных отправителя на фреймы (обычно от нескольких сотен до нескольких тысяч байтов), последовательной передачей фреймов и обработкой фреймов уведомления, поступающих от получателя. Поскольку физический уровень не распознает структуры в передаваемых данных, то сохранение этой структуры - целиком и полностью задача канала данных, а именно, на этом уровне нужно определить границы фрейма. Эта задача решается введением специальной последовательности

битов, которая добавляется в начало и в конец фрейма и всегда интерпретируется как границы фрейма.

Помехи на линии могут разрушить фрейм. В этом случае он должен быть передан повторно. Он будет повторен также в том случае, если фрейм уведомления будет потерян. И это уже заботы уровня - как бороться с дубликатами одного и того же фрейма, потерями или искажениями фреймов. Уровень канала данных может поддерживать для сетевого уровня сервис разных классов, разного качества и стоимости.

Другая проблема, возникающая на уровне канала данных (равно как и на других вышележащих уровнях), - как управлять потоком передачи. Например, как предотвратить «захлебывание» получателя? Как сообщить передающему размер буфера для приема передаваемых данных, имеющийся у получателя в этот момент?

Если канал позволяет передавать данные в обоих направлениях одновременно, т.е. если фреймы уведомления для потока от А к В используют тот же канал, что и трафик от В к А, то можно использовать для передачи фреймов уведомлений от В к А фреймы DU от А к В.

В сетях с вещательным способом передачи возникает проблема управления доступом к общему каналу. За это отвечает специальный подуровень канального уровня - подуровень доступа к среде (MAC - Media ACcess).

Основная проблема, решаемая на сетевом уровне, - как маршрутизировать пакеты от отправителя к получателю. Маршруты могут быть определены заранее и прописаны в статической таблице, которая не изменяется. Они могут также определяться в момент установления соединения. Наконец, они могут строиться динамически по ходу передачи в зависимости от загрузки сети.

Если в транспортной подсети циркулирует слишком много пакетов, то они могут использовать одни и те же маршруты, что будет приводить к заторам или перегрузкам. Эта проблема также решается на сетевом уровне.

Поскольку за использование транспортной подсети, как правило, предполагается оплата, то на этом уровне присутствуют функции учета: как много байт (или символов) послал или получил абонент сети. Если абоненты расположены в разных странах, где действуют разные тарифы, то надо должным образом скорректировать цену услуги.

Если пакет адресован в другую сеть, то надо предпринять надлежащие меры: в ней может быть другой формат пакетов, способ адресации, размер пакетов, другие протоколы и т.д. - все эти проблемы решаются на сетевом уровне.

В сетях с вещательной передачей проблемы маршрутизации просты, и этот уровень часто отсутствует.

Основная функция транспортного уровня - принять данные с уровня сессии, разделить, если надо, на более мелкие единицы, передать на сетевой уровень и позаботиться, чтобы все они дошли в целостности до адресата. Все это должно быть сделано эффективно и так, чтобы вышележащий уровень не зависел от того, как именно это было сделано. В нормальных условиях транспортный уровень должен создавать специальное сетевое соединение для каждого транспортного соединения по запросу уровня сессии. Если транспортное соединение требует высокую пропускную способность, то транспортный уровень может потребовать у сетевого уровня создать несколько сетевых соединений, между которыми транспортный уровень буден распределять передаваемые данные. И наоборот, если требуется обеспечить недорогое транспортное соединение, то транспортный уровень может использовать одно и то же соединение на сетевом уровне для нескольких транспортных соединений. В любом случае такое мультиплексирование должно быть незаметным на уровне сессии.

Сетевой уровень определяет, какой тип сервиса предоставить вышележащим уровням и пользователям сети. Наиболее часто используемым сервисом является канал «точка-точка» без ошибок, обеспечивающий доставку сообщений или байтов в той последовательности, в какой они были отправлены. Другой вид сервиса - доставка отдельных сообщений без гарантии сохранения их последовательности или, например, рассылка одного сообщения многим в режиме вещания. В каждом конкретном случае сервис определяют при установлении транспортного соединения.

Транспортный уровень - это уровень, обеспечивающий соединение «точка-точка». Активности транспортного уровня на машине отправителя общаются с равнозначными активностями транспортного уровня на машине получателя. Этого нельзя сказать про активности на нижележащих уровнях. Они общаются с равнозначными активностями на соседних машинах. В этом одно из основных отличий уровней 1-3 от уровней 4-7. Последние уровни обеспечивают соединение «точка-точка». Это хорошо видно на рисунке 1-13.

Многие хост-машины - мультипрограммные, поэтому транспортный уровень для одной такой машины должен поддерживать несколько транспортных соединений. Чтобы определить, к какому соединению относится тот или иной пакет, в его заголовке помещается необходимая информация.

Транспортный уровень также отвечает за установление и разрыв транспортного соединения в сети. Это предполагает наличие механизма именования, что значит, что процесс на одной машине должен уметь указать, с кем в сети ему надо обменяться информацией. Транспортный уровень также должен предотвращать «захлебывание» получателя в случае «очень быстро говорящего» отправителя. Механизм для этого называется управление потоком. Он есть и на других уровнях. Однако, как мы увидим ниже, управление потоком между хостами отличен от управления потоком между маршрутизаторами.

Уровень сессии позволяет пользователям на А-машинах (напомним, что пользователем может быть программа) устанавливать между собой сессии. Сессия позволяет передавать данные, как это может делать транспортный уровень, но, кроме того, этот уровень имеет более сложный сервис, полезный в некоторых приложениях. Например, он может осуществлять вход в удаленную систему, передавать файл между двумя приложениями и т.п.

Один из видов услуг на этом уровне - управление диалогом. Потоки данных могут быть разрешены в обоих направлениях одновременно, либо поочередно в одном направлении. Сервис на уровне сессии будет управлять направлением передачи.

Другой вид сервиса на этом уровне - управление маркером. Для некоторых протоколов недопустимо выполнение одной и той же операции на обоих концах соединения одновременно. Для этого уровень сессии выделяет активной стороне маркер. Операцию может выполнять тот, кто владеет маркером. Другим примером сервиса на этом уровне является синхронизация. Пусть нам надо передать такой файл, что его пересылка займет два часа, между машинами, время наработки на отказ у которых - один час. Ясно, что «в лоб» передачу такого файла средствами транспортного уровня не решить. Уровень сессии позволяет расставлять контрольные точки. В случае отказа одной из машин передача возобновится с последней контрольной точки.

Уровень представления предоставляет решения для часто возникающих проблем, чем облегчает участь пользователей. В основном это проблемы семантики и синтаксиса передаваемой информации. Данный уровень имеет дело с информацией, а не с потоком битов.

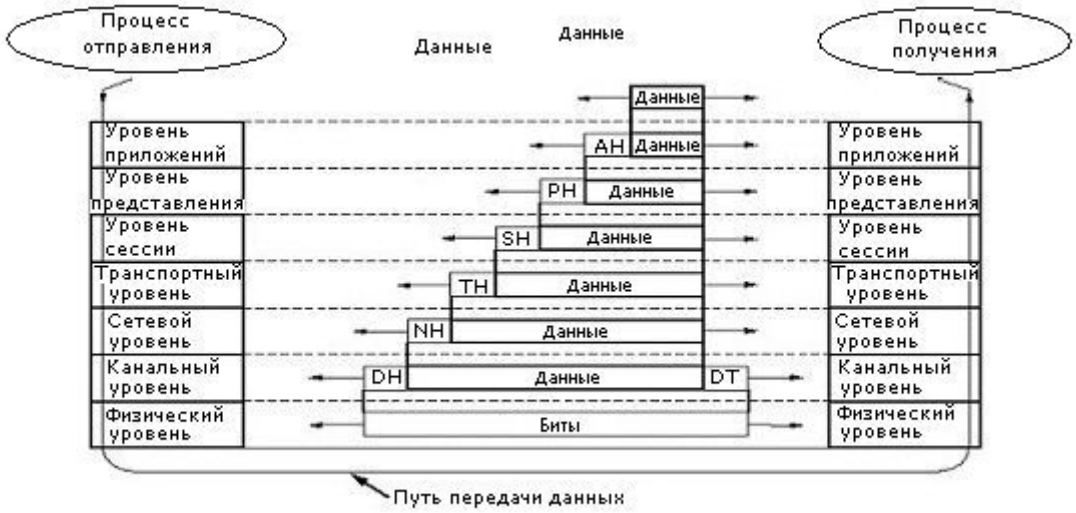
Типичным примером услуги на этом уровне является унифицированная кодировка данных. Уровень представления работает со структурами данных в абстрактной форме, преобразует это представление во внутреннее для конкретной машины и из внутреннего, машинного представления, в стандартное представление для передачи по сети.

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

Есть другой путь: определить сетевой виртуальный терминал и написать для него редактор. Для каждого типа терминала написать программу отображения этого терминала на сетевой виртуальный терминал. Все программное обеспечение для виртуального сетевого терминала расположено на уровне приложений.

На рисунке 1-14 показана последовательность действий при передаче данных в МОС-модели. Хотя данные движутся вертикально, каждый уровень предполагает их горизонтальное передвижение.

Рисунок 1-14. Пример передачи данных в модели МОС



Эталонная модель TCP/IP.

Здесь мы рассмотрим другую эталонную модель, прототипом для которой послужил прародитель всех компьютерных сетей - сеть ARPA.

С самого начала эта сеть задумывалась как объединение нескольких разных сетей. Одной из основных целей этого проекта было разработать унифицированные способы соединения сетей. С появлением спутниковых и радио цифровых каналов связи проблема становилась только актуальнее. Так появилась модель TCP/IP. Свое название она получила по именам двух основных протоколов: TCP - протокол управления передачей (Transmission Control Protocol), и IP - межсетевой протокол (Internet Protocol).

Другой целью проекта ARPA было создание протоколов, не зависящих от характеристик конкретных хост-машин, маршрутизаторов, шлюзов и т.п. Кроме этого, связь должна поддерживаться, даже если отдельные компоненты сети будут выходить из строя во время соединения. Другими словами, связь должна поддерживаться до тех пор, пока источник информации и получатель информации работоспособны. Архитектура сети не должна ограничивать приложения, начиная от простой передачи файлов до передачи речи и изображения в реальном времени.

В силу вышеперечисленных требований выбор организации транспортной среды был очевиден: сеть с коммутацией пакетов с межсетевым уровнем без соединений. Этот уровень называется межсетевым уровнем. Он является основой всей архитектуры. Его назначение - обеспечить доставку пакетов, движущихся в сети независимо друг от друга, даже если получатель принадлежит другой сети. Причем пакеты могут поступать к получателю не в том порядке, в котором они были посланы. Упорядочить их в надлежащем порядке - задача вышележащего уровня.

Межсетевой уровень определяет межсетевой протокол IP и формат пакета. Обратите внимание, что ни протокол, ни формат пакета не являются официальными международными стандартами, в отличие от протоколов эталонной модели МОС. Там большинство протоколов имеют статус международных стандартов.

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

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

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

Второй протокол - UDP (User Datagram Protocol). Это ненадежный протокол без соединения для тех приложений, которые используют свои механизмы фрагментации и управления потоком. Он часто используется для передачи коротких сообщений в клиент-серверных приложениях, а также там, где скорость передачи важнее ее точности.

Над транспортным протоколом располагается уровень приложений. Этот уровень включает следующие приложения: виртуальный терминал - TELNET, передачу файлов - FTP, электронную почту - SMTP. Позднее к ним добавились: служба имен домена - DNS (Domain Name Service), отображающая логические имена хост-машин на их сетевые адреса, протокол для передачи новостей - NNTP и протокол для работы с гипертекстовыми документами во Всемирной паутине - HTTP.

Под межсетевым уровнем в TCP/IP-модели великая пустота. Модель ничего не говорит, что там происходит, кроме того, что хост-машина должна быть связана с сетью через некоторый протокол. Никаких ограничений на этот протокол, равно как и рекомендаций нет.

Сравнение моделей МОС и TCP/IP.

Обе модели имеют много общего. Обе имеют уровневую организацию, поддерживают понятие стека протоколов. Назначение их уровней примерно одинаково. Все уровни от транспортного и ниже используют протоколы для поддержки взаимодействия типа «точка-точка», не зависящего от организации сети. Все уровни выше транспортного ориентированы на приложения.

В модели МОС центральными являются три понятия:

- сервис
- интерфейс
- протокол

Наибольшее методологическое значение этой модели - в четком выделении и разделении этих понятий.

Сервис определяет, что делает уровень, но ничего не говорит, как. Интерфейс уровня определяет для вышележащего уровня доступ к сервису. Протокол определяет реализацию сервиса.

В TCP/IP-модели нет столь же четкого выделения этих понятий. В ней понятие протокола четко «упрятано» и независимо от остальных частей модели. Этот факт есть следствие того, как создавались эти модели. TCP/IP-модель создавалась post factum, а МОС - до того, как появились протоколы. Поэтому понятие протокола там абсолютно не зависит от остальных частей модели. Например, изначально протоколы канального уровня в МОС-модели создавались для соединений «точка-точка». Позднее, когда появились средства типа вещания, на этот уровень были добавлены соответствующие протоколы. Никаких других изменений не последовало.

TCP/IP-модель была создана, когда TCP/IP-стек уже существовал. Поэтому эта модель прекрасно описывала этот стек, но только его, и никакой другой.

Модели имеют разное число уровней. Обе имеют уровень приложений, транспортный уровень и сетевой уровень. Все остальные уровни разные. МОС-модель поддерживает на сетевом уровне как сервис с соединением, так и без соединения. На транспортном уровне этой модели поддерживается сервис только с соединением. В TCP/IP наоборот: сетевой уровень обеспечивает сервис без соединения, но транспортный - как с соединением, так и без.

Недомтатки МОС:

1. Не вовремя.

2. Не технологичны.

3. Трудно реализуемы.

4. Неправильная стратегия.

«Не вовремя»: введение стандарта должно следовать за окончанием исследований, но прежде, чем начнутся крупные вложения в разработку.

Не технологичны:

- Функциональность между семью уровнями распределена неравномерно.
- МСО поспешило за IBM SNA (System Network Architecture).
- Описание модели и ее протоколов очень сложно.
- Некоторые функции, такие как управление потоком, исправление ошибок, адресация, повторяются на каждом уровне.
- Для некоторых функций не ясно, на какой уровень их поместить (виртуальный терминал); шифрование и защита в модели отсутствуют.
- Модель слишком ориентирована на сервис с соединениями и мало внимания уделяет сервису без соединений.
- В модели доминирует связь, практически не отражена взаимосвязь между вычислениями и связью (indication vs. receive). В МОС-модели слушком велико влияние Международного комитета по телефонии и телеграфии (МКТТ).

Трудно реализуемы: первые реализации протоколов МОС были громоздки и неэффективны. Первые реализации TCP/IP были сделаны в университете Беркли в рамках проекта по созданию операционной системы UNIX.

Неправильная стратегия: модель МОС - результат усилий ЕС, европейских министерств и ведомств. Даже правительство США приложило руку. TCP/IP - плод академической среды. Распространение модели МОС шло через правительственные инстанции и государственные структуры, модели TCP/IP - через университеты и научные организации.

Недостатки эталонной модели TCP/IP:

1. В модели нет четкого разграничения понятий «сервис», «интерфейс», «протокол».

2. Модель годится только для описания стека TCP/IP.

3. Уровень «хост-сеть» по существу уровнем не является, это больше интерфейс.

4. В этой модели не разделяются физическая среда передачи и уровень канала данных.

Протоколы TCP и IP разработаны действительно тщательно и эффективно реализованы, чего нельзя сказать о многих других протоколах (протокол виртуального терминала, TELNET)

По существу МОС-модель доказала свою эффективность, как методологический инструмент, стала популярной, чего нельзя сказать о протоколах. С TCP/IP все наоборот - модели по существу нет, зато протоколы получили широкое распространение.

Билет № 3.

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

Сети X.25.

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

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

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

Frame Relay.

Ретрансляция кадров (Frame Relay - FR) - это метод доставки сообщений в сетях передачи данных (СПД) с коммутацией пакетов (в отличие от СПД с коммутацией каналов и сообщений). Первоначально разработка стандарта FR ориентировалась на цифровые сети с интегрированным сервисом (ISDN - Integrated Services Digital Networks), однако позже стало ясно, что FR применим и в других СПД (здесь под данными понимается любое сообщение, представленное в цифровой форме). К числу достоинств метода, прежде всего, необходимо отнести малое время задержки, простой формат кадров, содержащих минимум управляющей информации, и независимость от протоколов верхних уровней эталонной модели МОС.

Эту службу можно рассматривать, как аренду виртуальной линии, по которой можно передавать пакеты длиной до 1600 байт. Можно заказать постоянную виртуальную линию от одного ко многим. Разница между арендуемой физической линией и виртуальной в том, что по физической линии можно гнать данные с максимальной скоростью целый день, по виртуальной средняя скорость будет меньше.

Эта служба предоставляет минимальный сервис. Если фрейм поступил с ошибкой, то он просто сбрасывается. Дело пользователя - определить, какой фрейм пропущен и как его восстановить. В отличие от X.25, FR не поддерживает уведомления о доставке и обычного управления потоком.

В настоящее время разработкой и исследованием стандартов FR занимаются три организации: Frame Relay Forum (FRF) - международный консорциум. Принятый FRF проект рассматривает только спецификации для постоянных виртуальных каналов (PVC) и интерфейса «пользователь-сеть» (UNI). В него не вошли стандарты для коммутируемых виртуальных каналов (SVC) и интерфейса межсетевого взаимодействия.

Высокоскоростной ISDN и ATM.

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

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

Есть много причин, почему данные удобно передавать небольшим пакетами - ячейками:

- Ячейки удобно использовать для управления и передачи разнородных данных - звук, видео, цифра.
- При больших скоростях проще управлять переключением небольших ячеек, чем использовать старую технику мультиплексирования.
- ATM - это технология, ориентированная на соединение: прежде чем передавать данные, устанавливается соединение и лишь потом передаются данные. Доставка данных не гарантируется, но порядок - да.
- ATM-сеть, как любая другая ПД, состоит из каналов и коммутаторов. В настоящее время достигнута скорость 155 Мбит/сек. и 622 Мбит/сек.

Когда ATM появился, основной областью применения этого сервиса считалось видео по заказу. В настоящее время появились и другие приложения, которые также требуют высокой пропускной способности.

Эталонная модель B-ISDN ATM.

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

Рисунок 1-22. Модель ATM

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

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

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

Плоскость пользователя отвечает за транспорт данных, управление потоком, исправление ошибок и другие функции пользователя. Плоскость управления отвечает за управление соединением.

Уровни управления уровнем и плоскостью отвечают за управление ресурсами и координацию межуровневых взаимодействий.

Физический уровень и уровень адаптации имеют по два подуровня. Они показаны в таблице 1-23.

Таблица 1-23. Уровни и подуровни ATM

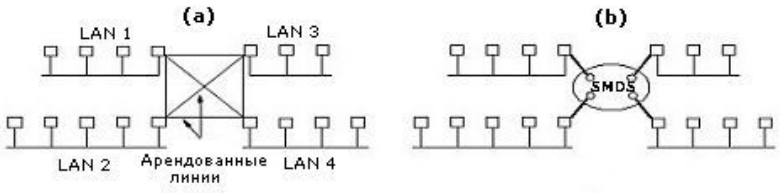
Уровень МОС	Уровень ATM	Подуровень ATM	Функции
3/4	AAL	CS (Convergence Sublayer - подуровень сходимости)	Предоставление стандартного интерфейса (сходимость)
		SAR (Segmentation & Reassembly Sublayer - подуровень сегментации и сборки)	Сегментация и сборка
2/3	ATM		Управление потоком Формирование/извлечение заголовков Управление виртуальным

			каналом/путем (Де)мультиплексирование ячеек
2	Физический	TC (Transmission Convergence Sublayer - подуровень подготовки ячеек)	Разделение передачи ячеек Формирование и проверка контрольной суммы заголовка Формирование ячеек Сборка ячеек в оболочку Формирование кадра
1		PMD (Physical Medium Dependent Sublayer - подуровень среды передачи)	Тактовая синхронизация Физический доступ к сети

SMDS - Мегабитная система передачи данных с коммутацией.

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

Рисунок 1-24. Соединение LAN через SMDS



Преимущества SMDS следующие:

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

Рисунок 1-25. Формат пакета SMDS

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

Таблица 1-26. Возможности разных СПД

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

Билет № 4.

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

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

1. Производительность сети характеризует скорость работы сети. Эта характеристика измеряется в количестве услуг в единицу времени. Под услугой может пониматься пропускная способность - число пакетов, пройденных через сеть за секунду, минуту, час, день. Соответственно говорят о средней, мгновенной, пиковой, минимальной пропускной способности сети. Это может быть выполнение определенной операции – тогда это время реакции. Чаще всего пользователь обращает внимание именно на этот индекс производительности. Он характеризует как скорость работы клиента, так и скорость работы сервера и СПД. Индекс, характеризующий только работу СПД, называется время передачи – время от поступления запроса на вход СПД до появления его на выходе.
2. Надежность. Эта характеристика сети определяет, всегда ли сеть способна выполнять операции и, если операция запущена, то всегда ли она корректно завершится. Есть несколько подходов измерения этой характеристики:

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

3. Безопасность характеризует степень защищенности сети от несанкционированного использования и изменения состояния ее ресурсов:

- ТС

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

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

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

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

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

6. Передача разнородных потоков данных (видео, звук, цифра). Слияние средств вычислений и средств передачи разнородных данных. Здесь основную сложность представляет синхронность передачи.

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

8. Совместимость характеризует способность подключать разное оборудование и программное обеспечение.

Кто, как и для чего вводит стандарты

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

Кто есть кто в мире стандартов для Интернета

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

Билет № 5.

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).

. Важной характеристикой сигнала является ширина его полосы, которая покрывает весь спектр частот гармоник, составляющих сигнал.

Основную проблему построения СПД представляет искажение сигнала при передаче. Это происходит под влиянием нескольких причин, основными из которых являются затухание, неравномерность затухания по частоте, искажение формы, разные виды шумов. Шумы возникают вследствие ряда причин, например таких, как термодинамические свойства проводника, взаимные наводки гармоник, составляющих сигнал, внешние электромагнитные воздействия. В случае аналогового сигнала эти искажения носят случайный характер и приводят к потере информации. В случае цифрового сигнала они приводят к ошибкам передачи.

Характеристику канала, определяющую спектр частот, которые физическая среда канала пропускает без существенного понижения мощности сигнала, называют полосой пропускания канала. Значение «существенного понижения мощности» определяется в конкретных случаях. Обычно падение мощности сигнала считают существенным, если оно составляет более 50% ее начального значения.

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, как мы уже говорили, могут иметь непрерывную или дискретную форму. В первом случае говорят об аналоговом сигнале, во втором - о цифровом

. Аналоговая передача предполагает непрерывное изменение параметров передачи. Цифровая передача - резкое, дискретное изменение параметров передаваемого сигнала или импульса.

Смысл процесса оцифровки состоит в том, что с определенной частотой замеряется уровень сигнала. Результаты замера представляют в виде некоторого кода, который передают с помощью цифрового кодирования

В случае цифровых сигналов это приведет к ошибке передачи, а в случае аналоговых сигналов – к искажению или просто потере сигнала.

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

Теорема Найквиста

max data rate = 2H $\log_2 V$ ит/сек,

где H – ширина полосы пропускания канала, выраженная в Гц, V - количество уровней в сигнале.

Однако теорема Найквиста не учитывает шум в канале, который измеряется как отношение мощности полезного сигнала к мощности шума: S/N Эта величина измеряется в децибелах: 10 $\log_{10}$(S/N) dB. Например, если отношение S/N равно 10, то говорят о шуме в 10 dB, если отношение равно 100, то - 20 dB.

На случай канала с шумом есть теорема Шеннона, по которой максимальная скорость передачи по каналу с шумом равна

H $\log_2$ (1+S/N) бит/сек.,

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

скорость передачи данных зависит от способа представления данных на физическом уровне и сигнальной скорости, или скорости модуляции - скорости изменения значения сигнала. Скорость изменений сигнала в секунду измеряется в единицах, называемых бот. Если скорость изменения значения сигнала b бот, то это не означает, что данные передается со скоростью b бит/сек.

Основой аналоговой передачи является непрерывный сигнал с постоянной частотой, который называют несущим сигналом.

Цифровой сигнал – это дискретная последовательность импульсов по напряжению, каждый из которых имеет ступенчатую форму. Каждый импульс – это единичный сигнал.

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

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

Кроме вышеперечисленных, есть и другие, например:

- достоверность передачи
- затухание
- помехоустойчивость

- и т.д.

Магнитная лента или магнитный диск в сочетании с обычным транспортным средством (автомашина, железная дорога и т.п.) 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.

Хотя вагон с магнитной лентой - это очень дешевый способ передачи, но задержка при передаче очень большая: в лучшем случае часы, обычно сутки. Для многих приложений нужен оперативный обмен информацией. Самой старой и все еще используемой средой передачи со времен появления телефона является ***витая пара***. Витая пара состоит из двух медных изолированных проводов, один из которых обвит вокруг другого. Этот второй, вьющийся провод предназначен для устранения взаимного влияния между соседними витыми парами.

Витая пара широко используется в телефонии. Между абонентами и АТС линии из витой пары могут иметь протяженность до нескольких километров без промежуточного усиления. Например, в России в городских условиях средняя длина абонентской линии равна 1,5 км. Витые пары объединяются в многопарные кабели.

Витая пара может быть использована для передачи как цифровых, так и аналоговых сигналов. Ее пропускная способность зависит от толщины используемых проводов и расстояния. Скорость в несколько мегабит в секунду вполне достижима с помощью соответствующих методов передачи. На коротких расстояниях (до сотни метров) может быть достигнута скорость до 1 Гбит/сек., на больших расстояниях (несколько километров) - не превышает 4 Мбит/сек. Учитывая это, а также низкую стоимость витой пары, она широко используется при создании ЛВС и, скорее всего, будет продолжать использоваться.

Наиболее часто используемыми являются кабели категории 3 и категории 5. Кабель категории 3 содержит по четыре витые пары с невысокой плотностью навивки и имеет полосу пропускания до 16 МГц. Кабель категории 5 имеет тоже четыре пары, но с более плотной навивкой, что позволяет достичь более высоких скоростей, и имеет полосу пропускания 100 МГц.

Как и у витой пары, у ***коаксиального кабеля*** есть два проводника. Однако устроены они иначе, что позволяет существенно увеличить полосу пропускания. 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. Обычно толщина коаксиала от 1 до 2,5 см, поэтому монтировать и прокладывать его сложнее, чем витую пару. Однако у коаксиала полоса пропускания шире и характеристики по затуханию сигнала лучше, чем у витой пары. Коаксиальные кабели работают на частотах от 1 МГц до 500 МГц. Поэтому эти кабели применяют на больших расстояниях и по ним могут передаваться одновременно несколько потоков данных от разных компьютеров.

Такие кабели находят самое широкое применение. Наиболее популярные из них:

- передача телевизионных сигналов, включая системы кабельного телевидения
- передача нескольких телефонных разговоров одновременно на большие расстояния между телефонными станциями, построение ЛВС
- подключение компьютерных периферийных устройств на небольших расстояниях

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

Есть два основных вида коаксиальных кабелей: узкополосный с волновым сопротивлением 50 Ом и широкополосный с волновым сопротивлением 75 Ом.

Узкополосный кабель позволяет достигать скорости в несколько Гбит/сек при длине в 1-2 км и высокой помехозащищенности. При большем расстоянии нужны промежуточные усилители. Эти кабели широко использовались между АТС. Они позволяют передавать более 10000 разговоров одновременно. В настоящее время они заменяются оптоволоконными линиями.

Существенное различие между узкополосным кабелем и широкополосным в том, что широкополосный кабель применяется для передачи аналоговых сигналов на больших расстояниях и, следовательно, требует промежуточных аналоговых усилителей. Эти промежуточные усилители пропускают сигналы только в одном направлении. Поэтому машина, получившая поток битов, не может использовать для ответа тот же путь, по которому поток битов к ней пришел. Для решения этой проблемы есть два вида систем: двухкабельные и однокабельные системы.

В двухкабельных системах прокладывается сразу два кабеля: один кабель используется для входящего потока, а второй для исходящего. Компьютер соединен этими кабелями со специальной головной станцией, которая перебрасывает трафик с одного кабеля на другой, идущий в нужном направлении. В однокабельных системах полоса частот разделяется между входящим и исходящим трафиками. Например, полоса от 5 до 30 МГц служит для входного трафика, а полоса от 40 до 300 МГц – для выходного. Эта граница в каждой стране устанавливается своя. Низкая полоса частот используется для передачи данных от компьютера к головной станции, которая сдвигает их в сторону высоких частот и передает на другие компьютеры.

Коаксиальные кабели активно используют в системах кабельного телевидения.

Волоконнооптические линии - одно из наиболее интенсивно развиваемых направлений средств связи.

Для использования оптической связи нужен источник света, светопроводящая среда и детектор, преобразующий световой поток в электрический. На одном конце волоконнооптической линии находится передатчик - источник света, световой импульс от этого источника проходит по светопроводящему волокну и попадает на детектор, который расположен на другом конце этой линии и преобразует этот импульс в электрический.

Одна из основных проблем создания оптоволоконных систем состояла в том, чтобы не дать световому пучку рассеяться через боковую поверхность силиконового шнура. 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 данные со скоростью несколько Гбит/сек. на сотню километров без промежуточного усиления.

Поскольку можно испускать несколько лучей разной длины волны так, чтобы они попадали на границы шнура под углом, большим угла полного внутреннего отражения, то по одному шнуру можно пускать несколько лучей. Каждый луч, как говорят, имеет свою моду. Так получается многомодовый шнур.

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

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

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

Оптоволоконный кабель состоит из сердечника, состоящего из сверхпрозрачного оптоволокна. В одномодовом кабеле сердечник имеет толщину 8-10 микрон, в многомодовом - около 50 микрон (это примерно толщина человеческого волоса). Сердечник окружен оптическим покрытием: стекловолокном с низким коэффициентом рефракции, сокращающим потери света через границу сердечника. Сверху все покрыто защитным пластиком.

Соединяют такие кабели электрически с помощью специальных коннекторов, механически, прижимая один край к другому, либо сваривая воедино оба конца. Все эти манипуляции приводят к потере от 5 до 20% мощности сигнала в точке соединения.

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

Таблица 2-22. Сравнение свойств светодиода и полупроводникового лазера

Свойство	Светодиод	Полупроводниковый лазер
Скорость передачи	Низкая	Высокая
Дальность передачи	Низкая	Высокая
Модовость	Мультимодовый	Мульти- или одномодовый
Срок службы	Короткий	Долгий
Чувствительность к температурным контрастам	Низкая	Значительная
Стоимость	Низкая	Высокая

С помощью оптоволокна можно строить как LAN, так и сети большего масштаба. Чтобы понять, как решается проблема построения сети из оптоволокна, надо осознать, что сеть типа «кольцо» представляет из себя цепочку соединений типа «точка-точка». Такие соединения могут быть двух видов: пассивное и активное. У пассивного есть светодиод либо лазер, и фотодиод. Принимая сигнал через фотодиод, это соединение передает электрический сигнал компьютеру или транслирует его дальше с помощью светодиода или лазера. Это абсолютно надежное соединение. Выход из строя любого из компонентов не нарушает связь по кольцу, а лишь блокирует работу

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

Кроме кольца, возможны соединения типа пассивной звезды. Все линии, по которым оптический сигнал передается от компьютера, заходят в специальное устройство пассивной звезды, сигналы от них воспринимаются по всем линиям, исходящим из этого устройства и передают к надлежащим приемникам.

Сравнение возможностей медного кабеля и оптоволокна.

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

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

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

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

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

Таблица 2-25. Сравнение характеристик витой пары, коаксиала и оптоволокна

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

Билет № 6.

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

Цифровой сигнал – это дискретная последовательность импульсов по напряжению, каждый из которых имеет ступенчатую форму. Каждый импульс – это единичный сигнал. В общем случае

данные в двоичной форме при передаче кодируются так, что один бит данных может быть отображен в несколько единичных сигналов. В простейшем случае это соответствие имеет однозначный характер: один бит – один единичный сигнал.

Если все единичные сигналы имеют одинаковую полярность (т.е. все положительную или все отрицательную), то говорят, что сигнал униполярный. В противном случае логическую единицу представляют положительным потенциалом, а логический ноль – отрицательным. Скорость передачи данных – это количество бит в секунду, которые передают с помощью сигналов. Эту скорость также называют битовой скоростью.

Продолжительность (длина) бита – это интервал времени, которое нужно передатчику, чтобы испустить последовательность надлежащих единичных сигналов. При скорости передачи данных R бит/сек, длина бита равна 1/R. Напомним, что скорость модуляции или сигнальная скорость измеряется в бот – это скорость изменения уровня сигнала. Очень многое зависит от способа кодировки данных. За одно изменение уровня сигнала можно передать несколько бит данных.

Теперь рассмотрим, какие задачи должен решать приемник при передаче. Прежде всего, приемник должен быть точно настроен на длину бита. Он должен уметь распознавать начало и конец передачи каждого бита, а также уровень сигнала: низкий или высокий. Эти задачи могут решаться 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. Существует 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 меньшая ширина полосы пропускания канала требуется для передачи. Важным также является отсутствие постоянной составляющей, т.е. гармоники с постоянными, не меняющимися параметрами. Появление такой гармоники приводит к наличию постоянного тока между приемником и передатчиком, что крайне нежелательно. Наконец, как мы уже отмечали, ширина спектра влияет на искажение формы сигнала. Чем шире спектр, тем сильнее искажения.
- Синхронизация между приемником и передатчиком. Мы уже отмечали необходимость для приемника точно определять начало и конец битового интервала. На небольших расстояниях, например, внутри компьютера или между компьютером и его периферийными устройствами для этих целей используют дополнительную линию синхронизации. По этой линии специальная тактирующая схема (часы) выдает строго через определенные промежутки синхроимпульсы. Приход такого импульса для приемника означает начало битового интервала. В сетях, на больших расстояниях, это решение не годится по многим причинам. Другое решение этой проблемы состоит в создании так называемых самосинхронизирующихся кодов. Например, перепад в уровне сигнала (фронт) может служить хорошим признаком для приемника о начале битового интервала. Отсутствие фронта между битовыми интервалами существенно усложняет решение проблемы синхронизации, когда в соседних битовых последовательностях надо передать биты с одинаковыми значениями.
- Обнаружение ошибок. Хотя методы обнаружения и исправления ошибок располагаются на канальном уровне, который находится над физическим уровнем, тем не менее, и на физическом уровне весьма полезно иметь такие возможности.
- Чувствительность к шуму. За счет надлежащих ухищрений в схеме кодировки данных можно добиться высокой производительности при передаче даже при наличии очень высокого уровня шума.
- Стоимость и скорость. Несмотря на постоянное удешевление цифровой аппаратуры общая тенденция такова, что увеличение сигнальной скорости с целью увеличения битовой ведет к удорожанию аппаратуры.

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

Потенциальный NRZ-код.

В потенциальной схеме кодирования NRZ (NRZ – Non return to zero – без возврата к нулю на битовом интервале) логическому 0 и логической 1 сопоставлены два устойчиво различаемых потенциала. К достоинствам этого кода следует отнести простоту реализации, устойчивость к ошибкам, достаточно узкий частотный спектр сигнала.

Основным недостатком этого кода является отсутствие синхронизации. На длинных последовательностях нулей или единиц, т.е. когда потенциал на линии не меняется, может произойти рассинхронизация между приемником и передатчиком, что приведет к ошибкам. Однако если исключить возможность появления длинных последовательностей 0 или 1, то этот метод может быть весьма эффективен. Обеспечить отсутствие таких последовательностей могут специальные устройства, называемые скремблеры.

Модификацией NRZ-кода и хорошим примером дифференциального кодирования является код NRZ-I. Идея дифференциальных кодов состоит в том, чтобы кодировать не абсолютное значение текущего бита, а разницу значений между предыдущим битом и текущим. В случае кода NRZ-I если текущий бит – 0, то он кодируется тем же потенциалом, что и предыдущий бит, если текущий бит – 1, то он кодируется другим потенциалом, чем предыдущий. Основным достоинством этого кода по отношению NRZ-коду является большая устойчивость к шуму.

Биполярный код AMI.

Другим примером потенциального кода является метод биполярного кодирования с альтернативной инверсией (Bipolar Alternate Mark Inversion – AMI). В этом методе используются не два уровня сигналов, как в NRZ-методах, а три: положительный, ноль и отрицательный. Значению 0 соответствует нулевой потенциал на линии; значению 1 - либо положительный, либо отрицательный потенциал. При этом потенциал каждой последующей единицы противоположен потенциалу предыдущей.

У этого метода есть несколько существенных преимуществ по сравнению с NRZ-кодами. Во-первых, в случае длительной последовательности единиц рассинхронизации не происходит. Каждая единица сопровождается изменением потенциала, устойчиво распознаваемым приемником. Поскольку каждая единица сопровождается изменением потенциала, то не возникнет постоянной составляющей. Однако длинная последовательность 0 остается проблемой, и требуются дополнительные усилия, которые позволили бы избежать ее появления. Во-вторых, спектр сигнала здесь уже, чем у NRZ-кодов. И, наконец, свойство чередования уровней позволяет обнаруживать единичные ошибки.

С применением надлежащей техники скремблирования биполярные импульсные коды обладают лучшими характеристиками, чем NRZ-коды. Однако это превосходство не бесплатно. Каждый единичный сигнал может иметь один из трех уровней, а поэтому он может нести $\log_2 3 = 1.58$ бит информации, из которых используется только один бит. Поэтому эффективность этого кода ниже. Кроме того, передатчик и приемник для биполярного метода сложнее, чем для NRZ-кодов.

Биполярные импульсные коды.

Существует другая группа методов кодирования, известная как биполярное импульсное кодирование. Здесь мы рассмотрим широко распространенные методы из этой группы: Манчестерский и дифференциальный Манчестерский коды.

В Манчестерском коде данные кодируются фронтами в середине битового интервала. Этим достигаются две цели: синхронизация приемника и передатчика, и передача данных: фронт перехода от низкого потенциала к высокому соответствует 1, а фронт перехода от высокого потенциала к низкому – 0.

В дифференциальном Манчестерском коде сигнал может менять свой уровень дважды в течение битового интервала. В середине интервала обязательно происходит изменение уровня. Этот перепад используется для синхронизации. При передаче 0 в начале битового интервала происходит перепад уровней, при 1 – такой перепад отсутствует.

Все биполярные импульсные методы требуют от одного до двух перепадов уровня сигнала за один битовый интервал. Поэтому их сигнальная скорость в два раза выше, чем у потенциальных кодов. Это означает, что они требуют более широкой полосы пропускания, чем потенциальные коды. Однако у них есть несколько существенных преимуществ:

- самосинхронизация
- отсутствие постоянной составляющей
- обнаружение единичных ошибок

Потенциальный код 2B1Q.

В этом методе каждые два последовательных бита (2B) передаются за один битовый интервал сигнала, который может иметь четыре состояния (1Q). Паре 00 соответствует потенциал -2.5 В, 01 соответствует -0.833 В, 11 – +0.833 В, 10 – +2.5 В.

У этого метода сигнальная скорость в два раза ниже, чем у кодов NRZ и AMI, а спектр сигнала в два раза уже. Поэтому с помощью 2B1Q-кода можно по одной и той же линии передавать данные в два раза быстрее. Однако реализация этого метода требует более мощного передатчика и более сложного приемника, который должен различать не два уровня, а четыре.

Сигнальная скорость.

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

Битовая скорость равна 1/tb, где tb– длина бита. Сигнальная скорость показывает скорость изменения уровня сигнала. Возьмем для примера Манчестерский код. Минимальный размер единичного сигнала равен половине битового интервала. Для последовательности из 0 или 1 будет генерироваться последовательность таких единичных сигналов. Поэтому сигнальная скорость Манчестерского кода равна 2/tb.

В общем случае

$D = R/b,$

где D – сигнальная скорость

R – битовая скорость в бит/сек.

b – количество бит на единичный сигнал

Билет № 7.

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

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). Рассмотрим два основных метода преобразования аналогового сигнала в цифровую форму: импульсно-кодовую модуляцию и дельта-модуляцию.

Импульсно-кодовая модуляция.

Импульсно-кодовая модуляция (ИКМ) основана на следствии из теоремы Найквиста, которое утверждает, что если измерять параметры сигнала f(t) через регулярные интервалы времени с частотой не меньше, чем удвоенная частота самой высокочастотной составляющей сигнала, то полученная серия измерений будет содержать всю информацию об исходном сигнале и этот сигнал может быть восстановлен. Однако надо помнить, что это замеры амплитуды аналогового сигнала. Чтобы преобразовать результаты замера в цифровой код, поступают следующим образом. Весь диапазон всех возможных амплитуд сигналов сначала разбивают, например, на 16 уровней. Каждому уровню сопоставляют двоичный код, который соответствует двоичному представлению номера этого уровня.

Важно иметь в виду, что т.к. каждый из 16 уровней является лишь приближением реального значения амплитуды сигнала, то точное восстановление исходного сигнала будет невозможно. Можно увеличить число уровней до 156, что потребует 8 разрядов (для передачи голоса это будет сравнимо по качеству с аналоговой передачей). Однако заметим, что нам в этом случае придется передавать результаты более 8000 замеров по 8 разрядов каждый, т.е. битовая скорость должна быть не ниже 64 Кбит/сек.

На стороне приемника по полученному цифровому коду восстанавливают аналоговый сигнал. Однако, как мы уже отметили, вследствие «округления» точное восстановление сигнала невозможно. Этот эффект называют ошибкой квантования или шумом квантования. Существуют методы его понижения за счет нелинейных методов квантования.

Дельта-модуляция.

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

Процесс передачи при использовании дельта-модуляции организован следующим образом. В момент очередного замера текущее значение исходной функции сравнивается со значением ступенчатой функции на предыдущем шаге. Если значение исходной функции больше, передается 1, в противном случае – 0. Таким образом, ступенчатая функция всегда меняет свое значение.

У метода дельта-модуляции есть два параметра: величина шага d и частота замеров, или шаг квантования. Выбор шага d – это баланс между ошибкой квантования и ошибкой перегрузки по крутизне (см. рисунок). Когда исходный сигнал изменяется достаточно медленно, то возникает только ошибка квантования, чем больше d, тем больше эта ошибка. Если же сигнал изменяется резко, то скорость роста ступенчатой функции может отставать. Это вид ошибки растет с уменьшением d.

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

Билет № 8.

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

Рассмотрим передачу данных в цифровой форме с помощью аналоговых сигналов. Широко известным примером такой передачи является использование телефонных сетей для передачи цифровых данных. Телефонные сети были созданы для передачи и коммутации аналоговых сигналов в голосовом диапазоне частот от 300 до 3400 Гц. Этот диапазон не совсем подходит для передачи цифровых данных. Поэтому подключить источник таких данных напрямую в телефонную сеть нельзя. Для этого используют специальное устройство - модем (МОдулятор–ДЕМодулятор). Этот прибор преобразует как цифровой сигнал в аналоговый в надлежащем диапазоне частот, так и наоборот: из аналоговой формы в цифровую.

Аналоговая модуляция заключается в управляемом изменении одного или нескольких основных параметров несущего сигнала: амплитуды, частоты и фазы. Есть три основных метода модуляции для преобразования цифровых данных в аналоговую:

- амплитудная модуляция
- частотная модуляция
- фазовая модуляция

Во всех этих случаях спектр гармоник получаемого сигнала сконцентрирован в области частоты несущего сигнала.

В случае амплитудной модуляции двоичные 0 и 1 представлены аналоговым сигналом на частоте несущей, но разной амплитуды. Обычно 0 соответствует сигнал с нулевой амплитудой. Таким образом, при амплитудной модуляции сигнал S(t) имеет вид:

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

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

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

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

S(t) =

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

Рассмотрим пример использования частотной модуляции для полнодуплексной связи по телефонной линии. Напомним, что полнодуплексной называется связь, когда данные можно передавать по каналу одновременно в оба направления. Телефонная линия имеет полосу от 300 Гц до 3400 Гц. Для обеспечения полного дуплекса эта полоса делится на две. По одной полосе с центром в 1170 Гц идет, например, передача, при которой 0 и 1 представлены частотами, сдвинутыми на 100 Гц, а по другой в этом случае идет прием, где 0 и 1 представлены частотами 2025 Гц и 2225 Гц. Обратите внимание, что эти две полосы немного перекрываются, поэтому возможна интерференция сигналов.

Частотная модуляция менее чувствительна к шумам, чем амплитудная. Чаще всего ее 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. Рассмотрим пример дифференциальной фазовой модуляции. В этом примере 0 представлен единичным сигналом той же фазы, что и предыдущий; 1 представлена единичным сигналом, сдвинутым по фазе на 180°. Для дифференциальной фазовой модуляции получаем:

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

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

Различие битовой скорости R бит/сек. и скорости модуляции D бит:

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

где D – скорость модуляции (сигнальная скорость)

R – битовая скорость (скорость передачи данных)

L – число разных уровней единичных сигналов

b – число бит на единичный сигнал

Билет № 9.

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

Аналоговая модуляция цифровых данных возникает там, где нет цифровых каналов. Цифровое кодирование аналоговых данных возникает тогда, когда есть цифровые каналы. Где возникает потребность передавать аналоговые данные с помощью аналоговых сигналов?

Прежде всего, такая потребность возникает при использовании радиоканалов. Если передавать аудиоинформацию в голосовом диапазоне (300 – 3000 Гц), то потребуется антенна диаметром в несколько километров. Модуляция, т.е. объединение исходного сигнала m(t) и несущей частоты f_c, позволяет нужным образом изменять параметры исходного сигнала и тем самым упростить решение ряда технических проблем. Кроме этого, модуляция позволяет использовать методы мультиплексирования или уплотнения.

Три способа модуляции для передачи аналоговых данных в аналоговой форме: 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.

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, $m(t)=1+n_a x(t)$.

Сигнал, получаемый фазовой модуляцией, определяет соотношение:

$$S(t) = A_c \cos(2\pi f_c t + n_p m(t)),$$

где n_p – индекс фазовой модуляции.

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

Широко распространенным случаем аналоговой модуляции является метод квадратичной амплитудной модуляции - QAM (Quadrature Amplitude Modulation). 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.

Билет № 10.

Физические среды передачи данных.

Раздел 2.3. Среды передачи

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

- полоса пропускания

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

Кроме вышеперечисленных, есть и другие, например:

- достоверность передачи
- затухание
- помехоустойчивость
- и т.д.

2.3.1. Магнитные носители

Магнитная лента или магнитный диск в сочетании с обычным транспортным средством (автомашина, железная дорога и т.п.) 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.

2.3.2. Витая пара

Хотя вагон с магнитной лентой - это очень дешевый способ передачи, но задержка при передаче очень большая: в лучшем случае часы, обычно сутки. Для многих приложений нужен оперативный обмен информацией. Самой старой и все еще используемой средой передачи со времен появления телефона является витая пара. Витая пара состоит из двух медных изолированных проводов, один из которых обвит вокруг другого. Этот второй, вьющийся провод предназначен для устранения взаимного влияния между соседними витыми парами.

Витая пара широко используется в телефонии. Между абонентами и АТС линии из витой пары могут иметь протяженность до нескольких километров без промежуточного усиления. Например, в России в городских условиях средняя длина абонентской линии равна 1,5 км. Витые пары объединяются в многопарные кабели.

Витая пара может быть использована для передачи как цифровых, так и аналоговых сигналов. Ее пропускная способность зависит от толщины используемых проводов и расстояния. Скорость в несколько мегабит в секунду вполне достижима с помощью соответствующих методов передачи. На коротких расстояниях (до сотни метров) может быть достигнута скорость до 1 Гбит/сек., на больших расстояниях (несколько километров) - не превышает 4 Мбит/сек. Учитывая это, а также низкую стоимость витой пары, она широко используется при создании ЛВС и, скорее всего, будет продолжать использоваться.

Наиболее часто используемыми являются кабели категории 3 и категории 5. Кабель категории 3 содержит по четыре витые пары с невысокой плотностью навивки и имеет полосу пропускания до 16 МГц. Кабель категории 5 имеет тоже четыре пары, но с более плотной навивкой, что позволяет достичь более высоких скоростей, и имеет полосу пропускания 100 МГц. В таблице 2-15 в первых 4-х столбцах приведены характеристики затухания сигнала для витой пары категорий 3 и 5, а также для экранированной витой пары 150 ом.

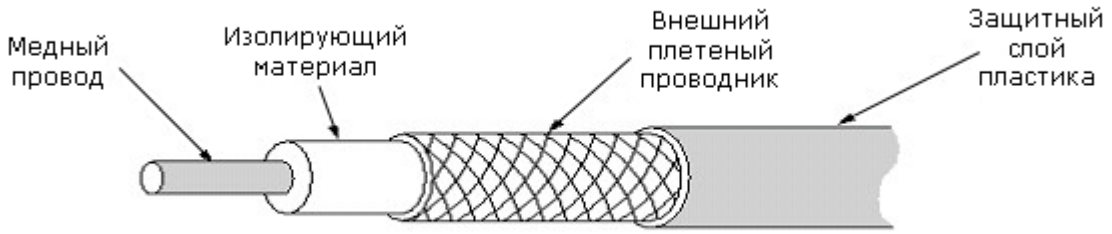
Таблица 2-15. Характеристики затухания для разных видов витой пары

	Затухание (дБ на каждые 100 м)			Помехи вследствие интерференции проводов (дБ)		
Частота (МГц)	Категория 3	Категория 5	Экранированная, 150 ом	Категория 3	Категория 5	Экранированная, 150 ом
1	2,6	2,0	1,1	41	62	58
4	5,6	4,1	2,2	32	53	58
16	13,1	8,2	4,4	23	44	50,4
25	-	10,4	6,2	-	41	47,5
100	-	22,0	12,3	-	32	38,5
300	-	-	21,4	-	-	31,3

2.3.3. Коаксиальные кабели

Как и у витой пары, у коаксиального кабеля есть два проводника. Однако устроены они иначе, что позволяет существенно увеличить полосу пропускания. На рисунке 2-16 показано устройство коаксиала. 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. Обычно толщина коаксиала от 1 до 2,5 см, поэтому монтировать и прокладывать его сложнее, чем витую пару. Однако у коаксиала полоса пропускания шире и характеристики по затуханию сигнала (см. рисунок 2-17) лучше, чем у витой пары. Из этого рисунка видно, что коаксиальные кабели работают на частотах от 1 МГц до 500 МГц. Поэтому эти кабели применяют на больших расстояниях и по ним могут передаваться одновременно несколько потоков данных от разных компьютеров.

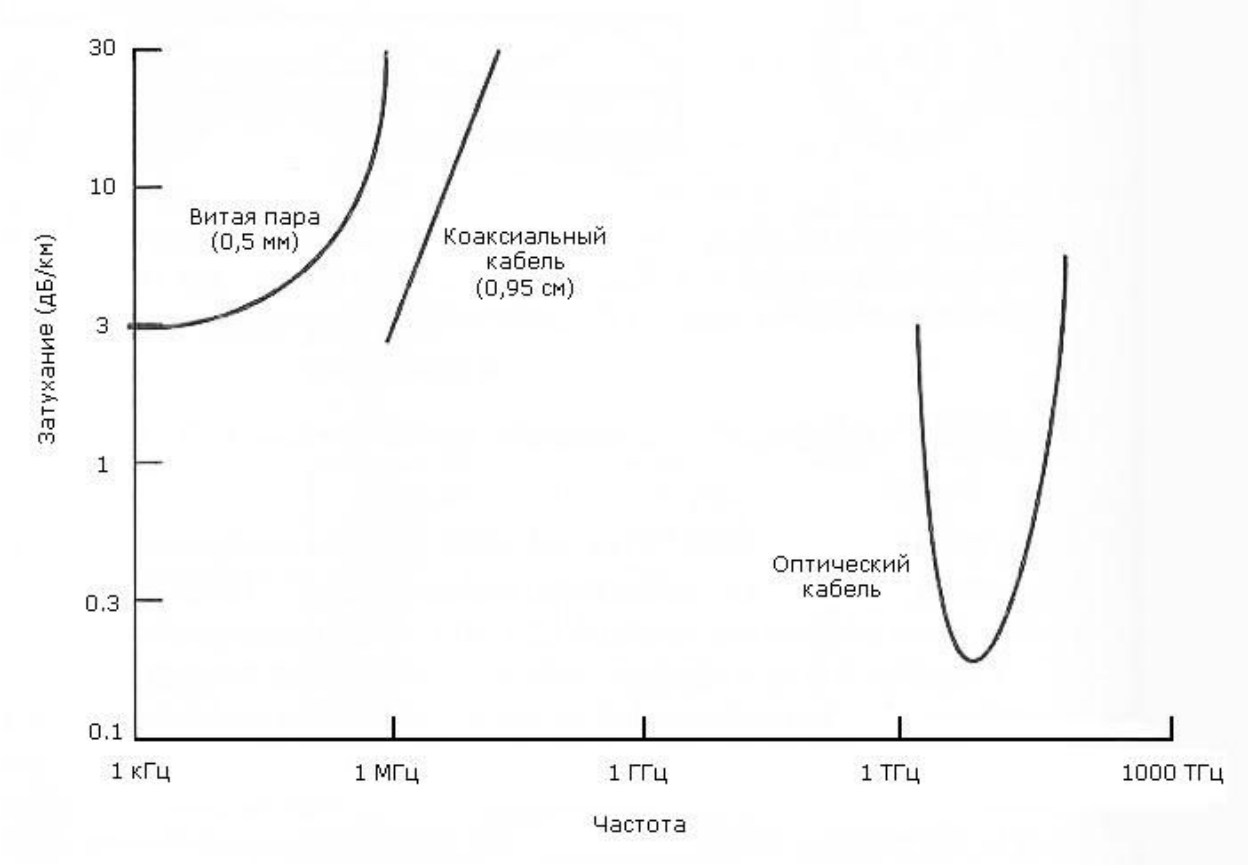
Рисунок 2-16. Устройство коаксиального кабеля



Такие кабели находят самое широкое применение. Наиболее популярные из них:

- передача телевизионных сигналов, включая системы кабельного телевидения
- передача нескольких телефонных разговоров одновременно на большие расстояния между телефонными станциями, построение ЛВС
- подключение компьютерных периферийных устройств на небольших расстояниях

Рисунок 2-17. Характеристики затухания сигнала для разных видов кабелей



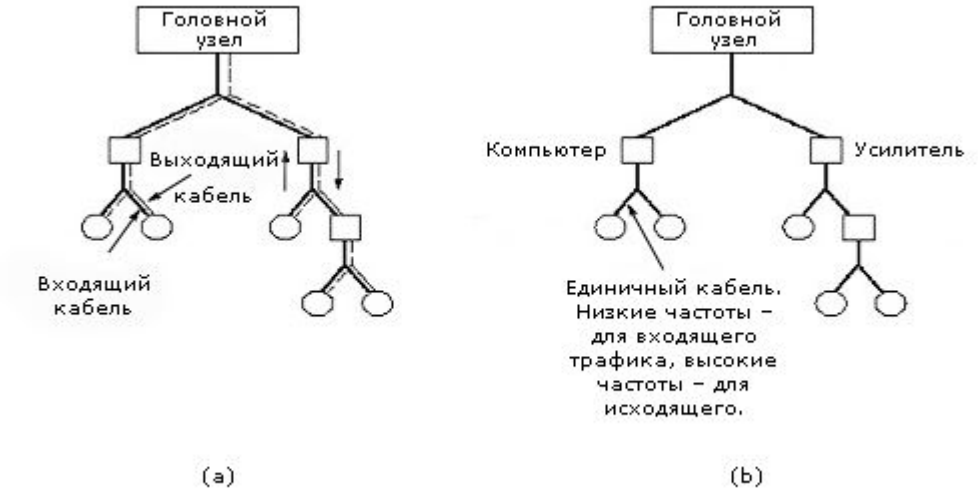
Коаксиальные кабели используют для передачи как аналоговых, так и цифровых сигналов. Как видно из рисунка 2-15, коаксиальные кабели превосходят по своим характеристикам витую пару. У них шире полоса пропускания, а следовательно, выше скорость передачи данных. Основными ограничителями скорости и расстояния при передаче без усиления являются в этих кабелях затухание сигнала, тепловой шум и интермодуляционный шум. Последний вид шума возникает, когда всю полосу пропускания кабеля разбивают на более узкие полосы и каждую такую полосу используют как отдельный канал. Интермодуляционный шум возникает на границах таких каналов.

Есть два основных вида коаксиальных кабелей: узкополосный с волновым сопротивлением 50 Ом и широкополосный с волновым сопротивлением 75 Ом.

Узкополосный кабель позволяет достигать скорости в несколько Гбит/сек при длине в 1-2 км и высокой помехозащищенности. При большем расстоянии нужны промежуточные усилители. Эти кабели широко использовались между АТС. Они позволяют передавать более 10000 разговоров одновременно. В настоящее время они заменяются оптоволоконными линиями.

Существенное различие между узкополосным кабелем и широкополосным в том, что широкополосный кабель применяется для передачи аналоговых сигналов на больших расстояниях и, следовательно, требует промежуточных аналоговых усилителей. Эти промежуточные усилители пропускают сигналы только в одном направлении. Поэтому машина, получившая поток битов, не может использовать для ответа тот же путь, по которому поток битов к ней пришел. Для решения этой проблемы есть два вида систем: двухкабельные и однокабельные системы. (См. рисунок 2-18).

Рисунок 2-18. Двухкабельные и однокабельные системы



В двухкабельных системах (рисунок 2-18 (a)) прокладывается сразу два кабеля: один кабель используется для входящего потока, а второй для исходящего. Компьютер соединен этими кабелями со специальной головной станцией, которая перебрасывает трафик с одного кабеля на другой, идущий в нужном направлении. В однокабельных системах полоса частот разделяется между входящим и исходящим трафиками. Например, полоса от 5 до 30 МГц служит для входного трафика, а полоса от 40 до 300 МГц – для выходного. Эта граница в каждой стране устанавливается своя. Низкая полоса частот используется для передачи данных от компьютера к головной станции, которая сдвигает их в сторону высоких частот и передает на другие компьютеры.

Коаксиальные кабели активно используют в системах кабельного телевидения. Кабельное телевидение, которое охватывает во многих странах до 90% всех домов (США, Голландия) становится претендентом на роль городской вычислительной сети (MAN). Системы кабельного телевидения используют также для телефонных разговоров и передачи данных. В настоящее время в этой области идет жесткая конкурентная борьба между телефонными компаниями и компаниями кабельного телевидения.

2.3.4. Оптоволокно

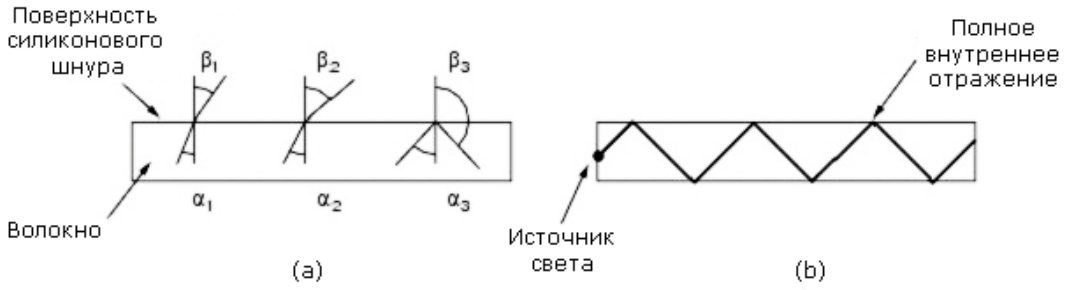
Волоконнооптические линии - одно из наиболее интенсивно развиваемых направлений средств связи. Если сравнить темпы развития трех основных движущих сил средств передачи и обработки данных: микропроцессорную технику, средства телекоммуникаций и инженерию программного обеспечения, то мы увидим, что микропроцессоры удваивают свою производительность каждые 18 месяцев, пропускная способность каналов связи вырастает на 75% в год. По прогнозам специалистов, к 2011 году кремниевая технология исчерпает свои потенциальные возможности по дальнейшему увеличению производительности. На горизонте развития оптоволоконных линий связи, которые уже сейчас имеют пропускную способность в 50000 Гбит/сек., пока подобных проблем не видно. Поэтому можно сказать, что эту гонку скоростей пока выигрывают линии связи. И главную роль здесь, конечно, играют волоконнооптические кабели.

Для использования оптической связи нужен источник света, светопроводящая среда и детектор, преобразующий световой поток в электрический. На одном конце волоконнооптической линии находится передатчик - источник света, световой импульс от

этого источника проходит по светопроводящему волокну и попадает на детектор, который расположен на другом конце этой линии и преобразует этот импульс в электрический.

Одна из основных проблем создания оптоволоконных систем состояла в том, чтобы не дать световому пучку рассеяться через боковую поверхность силиконового шнура. Количество рассеиваемой энергии зависело от угла падения светового луча на стенки шнура. На рисунке 2-19 показана эта зависимость. При углах больше некоторого критического угла, называемого углом полного внутреннего отражения, вся энергия луча отражается обратно внутрь.

Рисунок 2-19. Угол падения луча в оптоволоконном кабеле



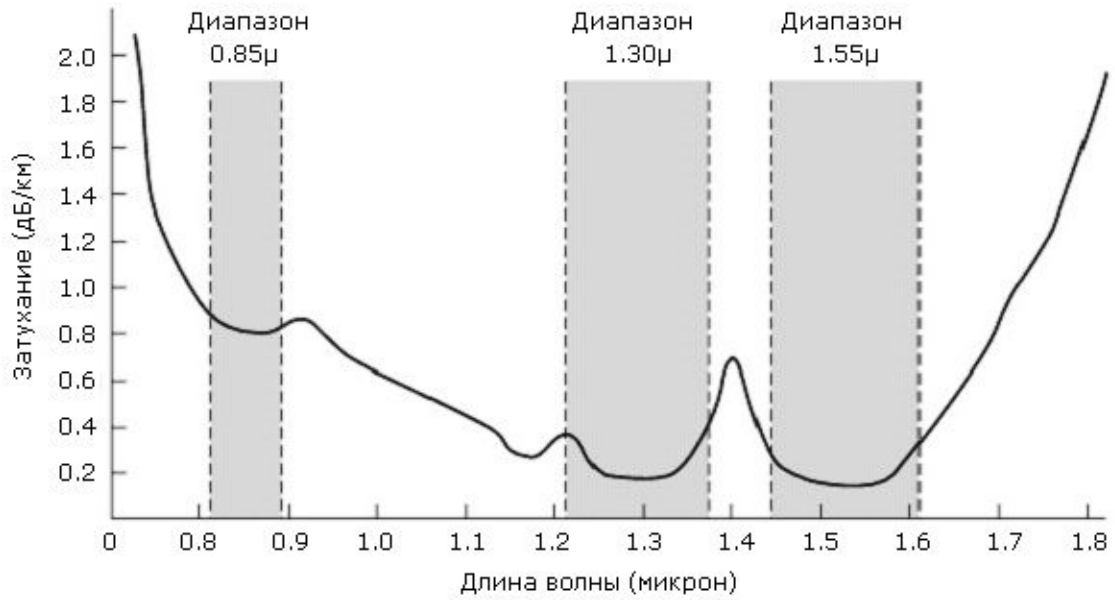
Если сделать силиконовый шнур толщиной, близкой к длине волны источника света, то этот шнур будет работать как провод для тока, без потерь на внутреннее отражение. По такому одномодовому шнуру можно передавать данные со скоростью несколько Гбит/сек. на сотню километров без промежуточного усиления.

Поскольку можно испускать несколько лучей разной длины волны так, чтобы они попадали на границы шнура под углом, большим угла полного внутреннего отражения, то по одному шнуру можно пускать несколько лучей. Каждый луч, как говорят, имеет свою моду. Так получается многомодовый шнур.

2.3.4.1. Прохождение света через оптоволокно

Оптоволокно делают из стеклоподобного материала, который, в свою очередь, производят из песка и других широко распространенных материалов. Стекло известно со времен египтян. Однако прозрачное оконное стекло научились делать только в эпоху Ренессанса. Современное стекло, используемое для оптоволокна, настолько прозрачно, что если им заполнить океан, то в любой его точке мы смогли бы видеть дно, как мы видим землю с борта самолета.

Рисунок 2-20. Зависимость затухания оптического сигнала от длины волны



Затухание оптического сигнала в стекле зависит от длины волны источника света. На рисунке 2-20 показана зависимость затухания от длины волны. Затухание измеряется в dB по следующей формуле:

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

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

Из этой формулы следует, что при падении мощности сигнала в два раза затухание будет равно примерно 3 дБ. На рисунке 2-20 видно, что затухание меньше всего в инфракрасной части спектра, которую и используют на практике. Видимая часть спектра располагается в области более коротких волн 0,4 – 0,7 микрон (1 мкм = 10^{-6} м).

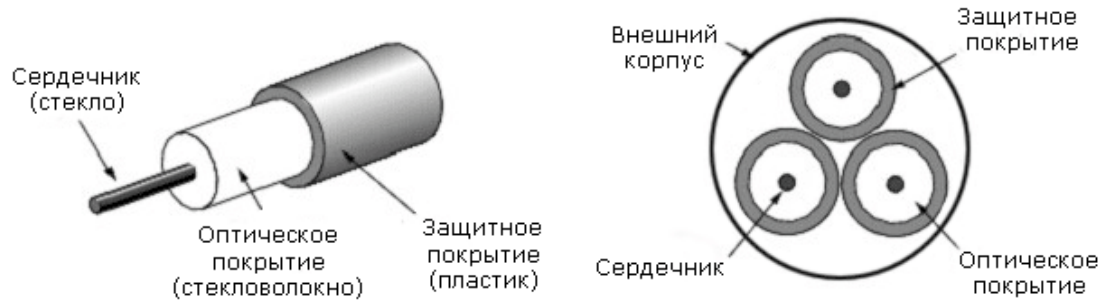
Для передачи используются три полосы с длинами волн 0,85, 1,30 и 1,55 мкм. Две последние обладают тем замечательным свойством, что их затухание составляет менее 5% на километр. Длина волны в 0,85 мкм имеет большее затухание, но хороша тем, что лучше соответствует возможностям лазерных источников света. 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. Это явление называется дисперсией. Величина этих искажений также зависит от длины волны. Одно из возможных решений - увеличить расстояние между соседними сигналами. Однако это сократит скорость передачи. К счастью, исследования показали, что если придать сигналу некоторую специальную форму, то дисперсионные эффекты почти исчезают и сигнал можно передавать на тысячи километров. Сигналы в этой специальной форме называются силитонами.

2.3.4.2. Оптоволоконный кабель.

Устройство оптоволоконного кабеля показано на рисунке 2-21. Кабель состоит из сердечника, состоящего из сверхпрозрачного оптоволокна. В одномодовом кабеле сердечник имеет толщину 8-10 микрон, в многомодовом - около 50 микрон (это примерно толщина человеческого волоса). Сердечник окружен оптическим покрытием: стекловолокном с низким коэффициентом рефракции, сокращающим потери света через границу сердечника. Сверху все покрыто защитным пластиком.

Рисунок 2-21. Устройство оптоволоконного кабеля



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

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

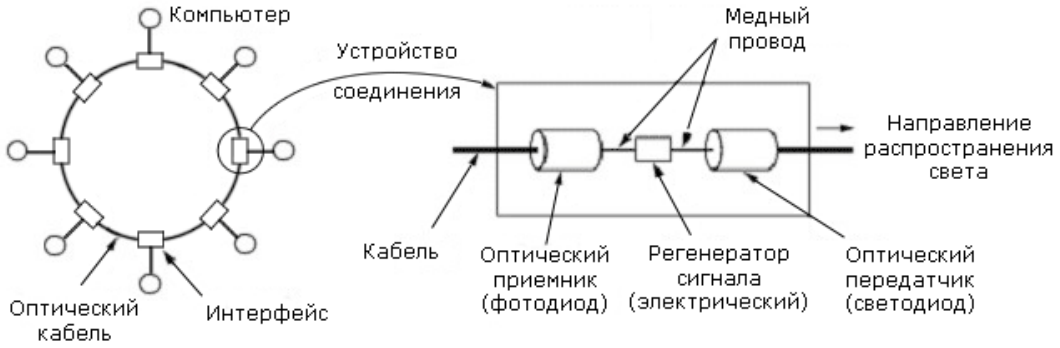
Таблица 2-22. Сравнение свойств светодиода и полупроводникового лазера

Свойство	Светодиод	Полупроводниковый лазер
Скорость передачи	Низкая	Высокая
Дальность передачи	Низкая	Высокая
Модовость	Мультимодовый	Мульти- или одномодовый
Срок службы	Короткий	Долгий
Чувствительность к температурным контрастам	Низкая	Значительная
Стоимость	Низкая	Высокая

2.3.4.3. Оптоволоконные сети

С помощью оптоволокна можно строить как LAN, так и сети большего масштаба. Подключение к оптоволоконной сети более сложное, чем к Ethernet-сети. Чтобы понять, как решается проблема построения сети из оптоволокна, надо осознать, что сеть типа «кольцо» представляет из себя цепочку соединений типа «точка-точка», как показано на рисунке 2-23.

Рисунок 2-23. Оптоволоконное кольцо

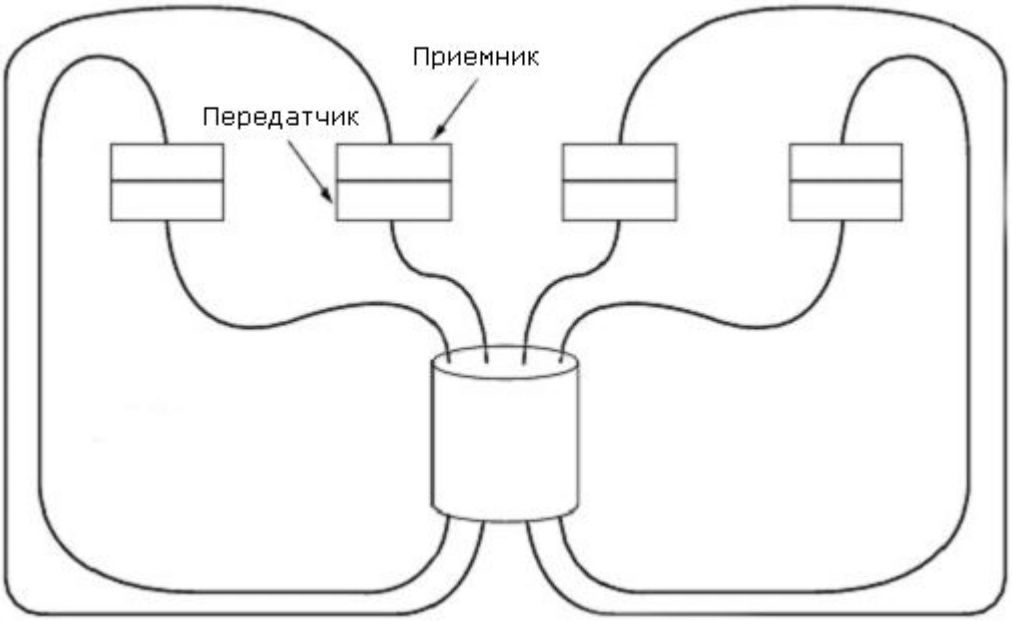


Такие соединения могут быть двух видов: пассивное и активное. У пассивного есть светодиод либо лазер, и фотодиод. Принимая сигнал через фотодиод, это соединение передает электрический сигнал компьютеру или транслирует его дальше с помощью светодиода или лазера. Это абсолютно надежное соединение. Выход из строя любого из компонентов не нарушает связь по кольцу, а лишь блокирует работу отдельного компьютера.

Активное подключение (рисунок 2-23, правая часть) содержит промежуточный усилитель электрического сигнала. Фотодиод преобразует оптический сигнал в электрический. Этот сигнал усиливается, передается компьютеру либо транслируется дальше с помощью лазера или светодиода.

Кроме кольца, возможны соединения типа пассивной звезды (рисунок 2-24). Все линии, по которым оптический сигнал передается от компьютера, заходят в специальное устройство пассивной звезды, сигналы от них воспринимаются по всем линиям, исходящим из этого устройства и передают к надлежащим приемникам.

Рисунок 2-24. Соединение типа «пассивная звезда»



2.3.4.4. Сравнение возможностей медного кабеля и оптоволокна

В заключение сравним возможности медного кабеля и оптоволокна:

6. Ширина полосы пропускания у оптоволокна несравненно больше, чем у медного кабеля, что позволяет достичь скорости в сотни Гбит/сек на расстояниях в десятки километров. Напомним, что коаксиал дает скорость максимум в несколько сотен Мбит/сек. примерно на 1 километре. Витая пара дает несколько Мбит/сек. на 1 километр и из нее можно выжать до 1 Гбит/сек. на расстоянии до 100 м.
7. 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.
8. Затухание сигнала в оптоволокне существенно меньше, чем в коаксиале и витой паре, и остается постоянным для широкого диапазона частот.
9. 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.
10. 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.

В таблице 2-25 приведены основные характеристики витой пары, коаксиала и оптоволокна.

Таблица 2-25. Сравнение характеристик витой пары, коаксиала и оптоволокна

	Диапазон частот	Стандартное затухание	Стандартная задержка	Расстояние между репитерами

Витая пара	0-3,5 кГц	0,2 дБ при 1 кГц	50 мсек./км	2 км
Многопарный кабель	0-1 МГц	3 дБ/км при 1 кГц	5 мсек./км	2 км
Коаксиал	0-500 МГц	7 дБ/км	5 мсек./км	1-9 км
Оптический кабель	180-370 ТГц	0,2-0,5 дБ/км	5 мсек./км	40 км

Билет № 11.

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

Беспроводная связь востребована не только при мобильных вычислительных средствах, но и там, где прокладка любого кабеля затруднительна, либо невозможна (горы, старые здания), либо где требуется быстрое создание коммуникации. Это особенно актуально для нашей страны, где почти 2/3 территории приходится на зону вечной мерзлоты и горы.

Электромагнитный спектр.

Как известно, электроны при движении образуют электромагнитные колебания. Если к источнику электромагнитных волн подключить антенну соответствующего размера, то волны будут распространяться и регистрироваться приемниками. Длина антенны, как у приемника, так и у передатчика, и длина излучаемой/принимаемой ею волны связаны определенными соотношениями. Например, длина антенны приемника не может быть короче половины длины принимаемой ею волны.

При определенных условиях волны будут распространяться в строго определенном направлении. В этом случае антенна приемника должна быть должным образом ориентирована в пространстве по отношению к антенне передатчика, чтобы принимать сигналы. При других условиях антенна передатчика распространяет электромагнитные волны во всех направлениях.

В вакууме электромагнитная волна распространяется со скоростью света (C=3x10^8 м/сек.). В медном проводнике эта скорость составляет 2/3 от скорости в вакууме. Будем обозначать f - частоту, а λ - длину волны. Фундаментальное соотношение, соединяющее f, C и λ, таково: $f \cdot \lambda = C$ (2-1).

Поскольку C - константа, зная λ, мы знаем f, и наоборот. Напомним, что длина волны определяет размер и геометрию антенны.

Для передачи информации из всего этого спектра используется только следующие диапазоны: радио, микроволновый, инфракрасный, видимый и, частично, ультрафиолетовый. Диапазоны рентгеновского излучения, гамма-излучения и большая часть ультрафиолетового, хотя и имеют большие частоты, а потому и более предпочтительны для передачи, однако требуют сложной аппаратуры для генерации и модуляции, плохо преодолевают препятствия и, что самое главное, опасны для живой материи.

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

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

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

Переписав уравнение 2-2 в разностной форме, получим:

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

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

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

Радиопередача.

Радиоволны распространяются на большие расстояния, легко преодолевают преграды, техника их генерации и приема хорошо изучена, есть много специалистов по ее применению. Поэтому они широко используются для связи как вне, так внутри помещений. Поскольку радиоволны распространяются во всех направлениях, то принимающая и передающая антенны не требуют дополнительной настройки и взаимного расположения. Однако свойство радиоволн распространяться во всех направлениях не всегда оказывается полезным.

Свойства радиоволн зависят от их частоты. На низких частотах, т.е. длинных волнах, они прекрасно преодолевают препятствия, но мощность сигнала падает пропорционально 1/r3 , где r - расстояние до источника.

На высоких частотах радиоволны распространяются по прямой, но хуже преодолевают препятствия. Для некоторых частот помехой становится даже дождь. На всех частотах радиоволны чувствительны к помехам от электрических устройств. В силу перечисленных выше свойств лицензирование, т.е. право на использование частот в радиодиапазоне, находится под жестким контролем государства.

Короткие волны хотя и поглощаются земной поверхностью, но за счет отражения от ионосферы также могут распространяться на большие расстояния.

Микроволновая передача.

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

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

Стремление увеличить пропускную способность канала заставляет использовать все более и более высокие частоты. Сегодня частота 10 ГГц - обычное дело. Однако здесь возникает серьезная проблема: начиная с частоты 8 ГГц волны поглощаются водой и, в частности, дождем. Единственный выход из положения - изменить маршрут передачи и обойти область дождя.

На сегодня микроволновый диапазон широко используется в телефонии, сотовой телефонии, телевидении и других приложениях. Одно из главных достоинств микроволнового диапазона - не надо ничего прокладывать. Достаточно получить права на небольшую площадку земли (сотню квадратных метров) установить башню-ретранслятор, и так через каждые 50 км. Такая технология особенно оправдана в условиях гор, труднопроходимой местности, где прокладка кабеля затруднена. Это справедливо и в городе, где земля дорогая, а коммуникации прокладывать очень сложно.

Есть несколько частотных полос, в диапазоне 2400-2484 ГГц, например, инфракрасные волны, которые можно использовать свободно без специального разрешения. В этих диапазонах работают микроволновые печи, радиотелефоны, радиоуправляемые двери и т.п. Эти частоты также используются для сетевых целей на небольших расстояниях. Однако в разных странах могут быть и дополнительные диапазоны.

Инфракрасные и миллиметровые волны.

Инфракрасное излучение и излучение в миллиметровом диапазоне используется на небольших расстояниях в блоках дистанционного управления. Основной недостаток излучения в этом диапазоне - оно не проходит через преграду. Для инфракрасного излучения лист бумаги – непреодолимое препятствие.

Этот недостаток одновременно является преимуществом, когда излучение в одной комнате не интерферирует с излучением в другой. На эту частоту не надо получать разрешения. Это прекрасный канал для передачи данных внутри помещений на небольших расстояниях.

Видимое излучение.

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

Билет № 12.

Телефонные сети: структура, проблема локальной петли (последняя миля). Технологии xDSL

Структура телефонной сети.

Структура современной телефонной сети весьма избыточная и многоуровневая. На рисунке 2-31 показана структура телефонной сети России. На этом рисунке используются следующие обозначения: АКТС - автоматическая коммутируемая телефонная сеть; ТФОП - телефоны общего пользования.

Если мы посмотрим на структуру телефонного номера на рисунке 2-32, то увидим четыре компонента: код страны, код региона в стране, затем код района или города в регионе и только потом номер абонента. В отдельных случаях крупным городам, например таким, как Москва, Санкт-Петербург присваивается код, как отдельному региону. В этом случае k1k2k3 - это код района в таком крупном городе.

Рисунок 2-31. Телефонная сеть России

Рисунок 2-32. Структура телефонного номера

10 x1x2 код страны	8 m1m2m3 код региона	k1k2k3 код города/района	n1n2n3n4 номер абонента

Каждый абонент соединен двумя витыми парами с ближайшей местной телефонной станцией (ТС), это соединение называют локальным соединением, абонентской линией или последней милей. В России протяженность локального соединения колеблется от сотен метров до 6-8 км. В городе оно короче, в сельской местности длиннее.

Местная ТС соединена в крупных городах с районной ТС либо городской ТС. Районные и городские ТС соединены с региональными или междугородными ТС, и т.д. в соответствии со структурой телефонного номера, изображенной на рисунке 2-32.

Если абонент звонит другому абоненту, который подключен к той же местной ТС, что и звонящий, то коммутаторы этой ТС соединяют абонентов напрямую. Каждая местная ТС соединена с ТС следующего уровня: районными или городскими ТС и междугородними ТС. Если абонент звонит абоненту, телефон которого подключен к другой местной ТС, то местная ТС звонящего соединяется с надлежащей ТС вышележащего уровня, которая устанавливает соединение с местной ТС, того кому звонят. В результате создается прямое соединение между абонентами. ТС соединяются между собой магистральными линиями.

В дальнейшем телефонные станции любого уровня мы будем просто называть узлами коммутации. Соединения между узлами коммутации должны обладать большой пропускной

способностью, чтобы по ним можно было передавать одновременно несколько разговоров. Пропускная способность местной линии должна быть достаточной для одного телефонного разговора. Для абонентских линий чаще всего применяли и применяют витую пару. Для магистралей между узлами коммутации используют коаксиальные кабели, оптоволокно и радиорелейные линии на микроволнах.

В прошлом телефонная система на всех уровнях была аналоговая, т.е. по проводам передавали колебания по напряжению в соответствии с акустическими колебаниями, принимаемыми на мембране микрофона. С появлением цифровых методов передачи аналоговая техника стала вытесняться, и на сегодня аналоговыми остались только абонентские линии.

Итак, современная телефонная сеть состоит из:

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

Локальное соединение.

Локальное соединение, или абонентская линия связывает абонента с ближайшим узлом коммутации. Это соединение также называют последней милей. На рисунке 2-34 показана организация локального соединения. Как видно из рисунка, при передаче данных приходится преобразовывать данные четыре раза из цифровой формы в аналоговую и обратно. Несмотря на то, что между узлами коммутации передача осуществляется в цифровой форме, в локальном соединении она пока аналоговая.

Рисунок 2-34. Передача цифровых данных по телефонной сети

При передаче аналогового сигнала есть три источника искажений:

- затухание
- искажение формы
- шум

Затухание возникает в любой среде из-за потери энергии сигнала при его распространении. При передаче по медному проводу затухание достигает нескольких дБ на километр. Затухание также зависит от частоты передаваемого сигнала. Как мы уже отмечали, промежуточное усиление может помочь лишь частично. Усилитель не может полностью восстановить исходную форму сигнала.

Искажения формы происходят также из-за разницы в скорости распространения сигналов разной частоты. Поскольку каждый сигнал есть комбинация гармоник разной частоты, а гармоники

разной частоты распространяются с разной скоростью, то гармоники одного сигнала могут накладываться на гармоники предыдущего и вызывать искажения исходной формы передаваемого сигнала.

Шум возникает вследствие посторонних источников энергии. Одним из таких источников является тепловой шум. Он неизбежен. Другими источниками могут быть атмосферные явления, соседние линии и т.п.

Из-за вышерассмотренных искажений сигнала желательно использовать при передаче как можно меньше гармоник. Однако скачкообразная форма цифрового сигнала как раз требует большого числа гармоник при передаче, чтобы как можно точнее воспроизвести форму сигнала, что требует от канала в свою очередь широкой полосы пропускания.

Решение проблемы лежит в использовании несущей частоты в сочетании с разными способами модуляции сигнала.

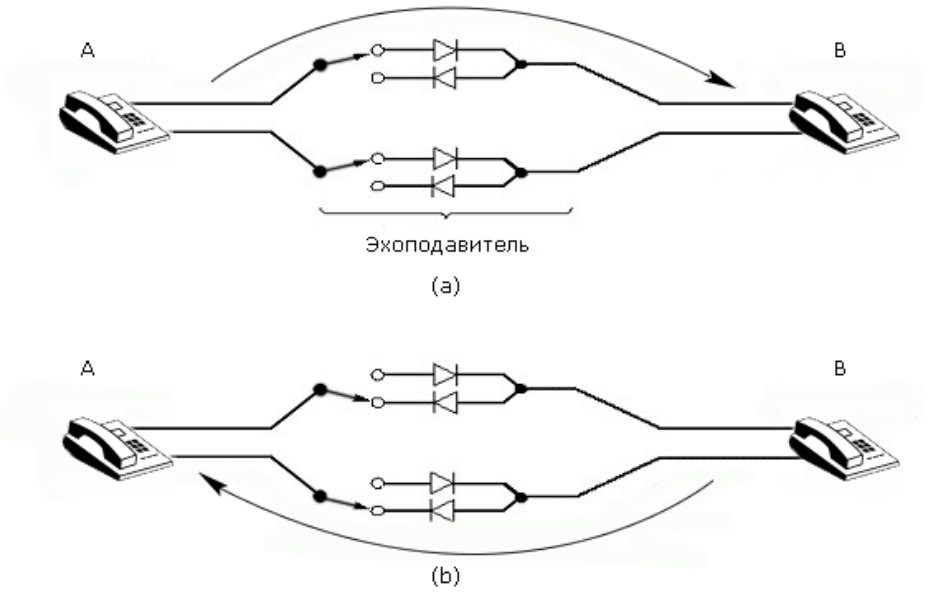
Устройство, которое преобразует поток битов в модулированный сигнал и обратно, называется модем. Чтобы увеличить скорость передачи, недостаточно увеличивать частоту несущей волны. Надо увеличивать число бит на осцилляцию, т.е. изменение уровня сигнала. Для соединения оба модема должны поддерживать одну и ту же схему модуляции.

В модем также встраивают средства для контроля и коррекции ошибок, которые используют специальные способы кодирования, мы их рассмотрим в главе 3. Самый простой из таких способов - добавление бита четности. В сочетании с кодами, исправляющими ошибку, в модемах используют алгоритмы сжатия. Например, широко распространенный алгоритм MNP 5 убирает из потока повторяющиеся байты.

Другой важной проблемой при использовании телефонной линии является эхо. Причина этого явления проста - когда сигнал достигает приемника, часть его энергии отражается и возвращается к передатчику. При небольших расстояниях между приемником и передатчиком это практически незаметно. Когда расстояние велико, задержка между сигналом и эхом становится значительной. При телефонном разговоре вы наверняка сталкивались с эффектом эхо.

На рисунке 2-36 показана схема подавления эха. Недостаток этой схемы состоит в том, что она не позволяет использовать полный дуплекс. Есть другое решение: устройство периодически измеряет величину эха и удаляет его из обратно получаемого сигнала. Здесь не требуется релейных устройств, а связь становиться полнодуплексной.

Рисунок 2-36. Схема подавления эха



Проблема «последней мили».

По мере развития сетевых услуг - информационное обслуживание, образование, видео по заказу и т.д. - росла потребность доступа в сеть среди индивидуальных абонентов. Пропускной способности 3 КГц обычной телефонной абонентской линии стало недостаточно. Возникла проблема, как обеспечить частные квартиры и дома линиями связи надлежащей пропускной способности, - так называемая «проблема последней мили».

Работы по решению этой проблемы велись в 4-х направлениях. Первое направление, достаточно «прямолинейное», было связано с подведением оптоволокна прямо в квартиру. Это направление называется FTTH (Fiber To The Home). Такое решение обеспечивало огромную пропускную способность, какую вряд ли индивидуальный пользователь сможет в полной мере использовать в ближайшее время. Стоимость его была под стать пропускной способности. Поэтому это решение имело смысл для крупных фирм, а не индивидуальных абонентов.

Второе направление было связано со стремлением сократить длину локального соединения до минимума. По имеющимся данным, в городских телефонных сетях России средняя длина абонентской линии составляет 1280 м (коэффициент вариации 0.59), ни одна абонентская линия ни в городе, ни в сельской местности не превышает 5 км. Было предложено подтянуть оптоволокно от местного узла коммутации до опорного шкафа развязки внутри микрорайона, а далее возможны были два варианта. От опорного шкафа использовать обычную витую пару с технологией HDSL из семейства xDSL, либо использовать коаксиальные кабели сети кабельного телевидения (это решение получило название Hybrid Fiber Coac – HFC).

Коаксиальный кабель в сочетании с оптоволокном обеспечивает одновременную передачу 40-50 аналоговых каналов, в том числе радиовещание, телевизионные передачи, телетекст. При использовании ADSL – асимметричной DSL-технологии (о которой речь пойдет чуть ниже), обеспечивающей интерактивность, добавляются видео по заказу, игры, доступ в Интернет.

Третий вариант решения – это использовать беспроводные технологии (WLL – Wireless Local Loop). Доступный для них диапазон частот сильно ограничен международными соглашениями. Скорости передачи данных уступают проводным технологиям.

Четвертый вариант решения – это использовать стандарты серии xDSL.

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

Таблица 2-39. Технологии организации последней мили

	WLL	ADSL	HFC	FTTH
Ширина полосы пропускания	Малая	Средняя	Большая	Очень большая
Круг пользователей	Индивидуальные абоненты, фирмы	Индивидуальные абоненты, небольшие фирмы	Индивидуальные абоненты	Крупные фирмы
Предоставляемые услуги	Телефон, радио, телевидение	То же, что у WLL, плюс видеотелефон, видео по заказу, компьютерные игры, дистанционные покупки	Радио, телевидение, телетекст	То же, что у ADSL, плюс видеоконференции, дистанционная медицина, газеты на экране и др.
Стоимость: первоначальные вложения развитие	 Низкая Низкая	 Низкая Средняя	 Высокая Низкая	 Высокая Средняя
Сроки ввода в эксплуатацию	Небольшой	Небольшой	Длительный	Длительный

Технологии xDSL.

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

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

Это весьма важное направление развития физических линий связи, поэтому мы хотя бы кратко опишем каждую из технологий этого семейства.

Родившееся как технология цифровых каналов в ISDN-сетях, семейство технологий xDSL получило развитие в новой сфере – абонентский доступ в Интернет.

По аналогии с модемами для работы по физической линии, модемы xDSL не ограничиваются для передачи информации спектром канала телефонных частот. Они используют всю полосу пропускания витой пары. Широкая полоса сигнала, используемого в этом семействе технологий, не позволяет работать по коммутируемым телефонным линиям (телефонные коммутаторы не рассчитаны на такой спектр частот). Поэтому xDSL-модемы могут работать только на участке телефонных кабельных систем между абонентом и сетью поставщика услуг или между двумя абонентами при непосредственном соединении их абонентских линий (без участия станции коммутации). Это так называемые выделенные линии.

Отличительной чертой семейства xDSL, по сравнению с модемами для физических линий, является использование спектра частот, не пересекающегося со спектром канала телефонных частот, благодаря чему по абонентской линии можно вести телефонные переговоры одновременно с передачей цифровой информации.

Технология DSL – «цифровая абонентская линия» – позволяет использовать существующие линии связи для передачи цифровой информации по одной витой паре со скоростью до 160 кбит/сек. (при этом в прямом и обратном направлении поддерживается одинаковая скорость). Реализация в оборудовании DSL-интерфейса ISDN BRI получила название IDSL. В оборудовании IDSL не предусматривается поддержка аналоговой телефонной линии, так как телефонная связь может осуществляться по цифровым каналам ISDN. Сейчас существуют модификации оборудования DSL – Fast DSL, передающие информацию со скоростью до 256 кбит/сек.

Технология DSL поддерживает аналоговую телефонную линию. Стандартный метод линейного кодирования – 2B1Q применяется практически во всех типах оборудования xDSL, за исключением оборудования подсемейств ADSL и VDSL. Максимальное расстояние (то есть максимальная длина двухпроводной линии, на которой может работать аппаратура) для этой технологии составляет 7,5 км при диаметре жилы кабеля 0,5 мм, что вполне покрывает длину абонентских линий в России.

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

Оборудование HDSL в основном предназначено для применения в корпоративных сетях. Отсутствие поддержки аналоговой телефонной линии компенсируется возможностью передачи речи в цифровом виде через интерфейсы E1.

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

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

Существующие образцы аппаратуры VDSL обеспечивают организацию канала связи при максимальных скоростях передачи на расстоянии не более 1,5 км. Применение оптического кабеля позволит значительно увеличить дальность связи, но потребует замены существующих медных абонентских кабелей. Предполагаемое разделение полосы частот таково:

- Голосовой телефонный сервис: 0-4 КГц
- ISDN: 4-80 КГц

- Исходящий поток: 300-700 КГц
- Входящий поток: 1 МГц

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

Указанные выше предельные скорости передачи в прямом и обратном направлении могут быть снижены в зависимости от конкретного типа оборудования, кабеля и протяженности абонентской линии. Оборудование ADSL способно автоматически или принудительно настраиваться так, чтобы на конкретной абонентской линии достичь максимальной скорости передачи с минимальным коэффициентом ошибок.

В ADSL используют усовершенствованный вариант частотной модуляции, позволяющей максимально использовать полосу в 1 МГц, обеспечиваемую витой парой. Максимальное расстояние передачи без повторителей 5.5 км.

Разновидностью ADSL-технологии является технология ***RADSL*** (Rate-adaptive DSL), которая может функционировать в асимметричном режиме как ADSL и в симметричном – как HDSL. Технология RADSL позволяет отслеживать текущее состояние кабеля (электрические параметры и уровень шума (помех)) и динамически регулировать пропускную способность каналов связи, а также поддерживать максимально возможную степень передачи при требуемом минимальном уровне ошибок в канале связи.

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

Билет № 13.

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

Магистрали и мультиплексирование

Наряду с абонентской линией, следующим важным компонентом телефонных систем являются магистрали, соединяющие узлы коммутации разного уровня. Здесь мы рассмотрим их организацию и функционирование.

Одним из существенных факторов при организации магистрали был и остается экономический. Дело в том, что затраты на прокладку кабеля в значительной степени определяют внешние условия (город, сельская местность, глубина залегания, наличие инженерных коммуникаций и т.д.), а не технические характеристики, например, пропускная способность. Поэтому чем больше абонентов смогут использовать один и тот же кабель, тем быстрее окупятся затраты на его прокладку, тем дешевле будет стоить каждому из них его эксплуатация. За 100 лет существования телефона были инвестированы огромные средства в создание методов и оборудования, позволяющих использовать одну и ту же магистраль одновременно для передачи нескольких разговоров. Такой технический прием называют мультиплексированием, или уплотнением.

Созданные в телефонии схемы мультиплексирования можно разделить на два больших класса: мультиплексирование с разделением частот и мультиплексирование с разделением по времени. Кроме этого, были разработаны методы мультиплексирования на основе разделения длин волн и на основе разделения кодов. Метод разделения длин волн применяют в оптоволоконных системах. Методы разделения кодов используют в системах беспроводной связи.

Мультиплексирование с разделением частот.

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

12 голосовых каналов с пропускной способностью по 4000 Гц мультиплексируют в полосе от 60 до 108 кГц. Такое соединение называют группой. Пять групп по 12 каналов мультиплексируют в супергруппу, затем пять супергрупп - в мастер-группу. Современные стандарты МКТТ позволяют объединять до 230 000 голосовых каналов.

Мультиплексирование с разделением длины волны.

Этот способ мультиплексирования используется для волоконноптических каналов. Два волоконнооптических кабеля с импульсами разной длины волны подводят к одной призме. Свет, пройдя через призму (или дифракционную решетку), смешивается в единый луч, который на другом конце разделяется с помощью другой призмы. Поскольку каждый канал занимает лишь несколько ГГц, а пропускная способность одного оптоволоконного канала около 25 000 ГГц (быстрее преобразовывать световой сигнал в электрический пока не могут), то возможности оптоволокна для мультиплексирования огромны. Метод мультиплексирования с разделением длин волн применяется в технологии FTTC.

Мультиплексирование с разделением по времени.

Частотное мультиплексирование требует применения аналоговых схем и малопригодно для управления компьютером. Мультиплексирование с разделением времени или TDM-мультиплексирование (Time Division Multiplexing), наоборот, предполагает использование цифрового оборудования и хорошо соответствует возможностям компьютера. Следует отметить, что оно подходит только для работы с данными в цифровой форме. Поскольку по абонентской линии телефонный сигнал передают в аналоговой форме, то его надо сначала оцифровать.

Оцифровка сигнала происходит на местном узле коммутации, куда сходятся абонентские линии с аналоговыми сигналами.

На местном узле коммутации аналоговые сигналы с абонентских линий оцифровываются, объединяются и передаются на узлы коммутации следующего уровня по магистральным шинам. Здесь мы рассмотрим, как это все происходит.

Есть два основных метода преобразования аналогового сигнала в цифровую форму и обратно. Это метод импульсно-кодовой модуляции (ИКМ-метод) и разностный метод Дельта-модуляции.

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

Стандарт E1 предполагает мультиплексирование 30 каналов. Каждая из 30 линий сканируется с частотой 8 000 Гц. Результаты каждого измерения представляют 8-битовое число. Это означает, что в методе ИКМ используются 256 уровней. В случае стандарта T1 используются 7 бит, т.е. 128 уровней.

Полученные 240 бит упаковывают в кадр. Кадр в стандарте E1 содержит 32 канала по 8 разрядов и занимает 125 мксек. 30 каналов используют для передачи данных, а два - для целей управления. Таким образом, стандарт E1 обеспечивает скорость 2,048 Мбит/сек и мультиплексирует 30 линий одновременно.

Стандарт T1 позволяет мультиплексировать 24 линии, но в каждом канале под данные используются лишь 7 разрядов и один разряд для целей управления. Кадр в T1 содержит 193 бита и занимает 125 мксек, что обеспечивает скорость в 1,544 Мбит/сек. Отметим, что в E1 из 256 битов кадра 16 используются для служебных целей, в T1 из 193 битов для служебных целей используются 24, т.е. E1 экономнее.

Так как аналоговый сигнал оцифрован, возникает искушение сжать передаваемые данные. Примером такого метода может служить метод разностной импульсно-кодовой модуляции. Идея сжатия в этом методе состоит в том, что если разность между последовательными замерами сигнала не превосходит, например, 8 уровней, в то время как собственно значения колеблются в диапазоне ±64, то вместо 6 разрядов цифрового кода нам потребуется всего 3 уровня. Частный случай такого подхода – это дельта-модуляция. В этом методе предполагается, что соседние значения отличаются не более чем на ±1. Для голоса этот метод работает неплохо.

Другой метод основан на экстраполяции очередного значения на основе предыдущих. Это так называемый метод статистической импульсно-кодовой модуляции. В этом методе передается разница между предсказанием и фактическим значением. Очевидно, что на обоих концах канала должен быть использован один и тот же алгоритм предсказания.

TDM-мультиплексирование позволяет мультиплексировать уже мультиплексированные каналы. Так, согласно стандарту T1, 4 канала T1 могут быть объединены в один T2, затем 6 в один T3 и 7 в один T4. Согласно E1, могут группироваться только 4 канала, но зато есть 4 уровня вложенности, а не три, как в T1. Поэтому скорость передачи в этом случае E1 = 2,048; E2 = 8,848; E3 = 34,304; E4 = 139,264; E5 = 565,148 Мбит/сек.

Стандарт SONET/SDH.

SONET (Synchronous Optical NETwork) – это интерфейс передачи по оптическим линиям связи, предложенный американской компанией Bell Core и стандартизированный ANSI. Позднее МККТ выпустил стандарт, совместимый с SONET и названный SDH (Synchronous Digital Hierarchy), который был опубликован в рекомендациях G.707, G.708, G.709. Этот стандарт был разработан для того, чтобы устранять разнобой в передаче сигналов по оптоволоконным линиям в области телефонии.

На первых порах каждая телефонная компания устанавливала свои стандарты TDM-мультиплексирования по оптическим линиям. В настоящее время многие телефонные компании, в том числе и в России, на своих магистральных линиях используют стандарт SDH.

Ниже кратко перечислены цели и конструктивные особенности стандарта SONET. Создание этого стандарта преследовало четыре основные цели:

- позволить использовать разные физические среды в сети, что требует проработки стандартов кодировки на физическом уровне, выбора длины волны, частоты, временных характеристик, структуры кадра
- унифицировать американские, европейские и японские цифровые системы, которые используют каналы 64 Кбит/сек. с импульсно-кодовой модуляцией, но по-разному
- обеспечить иерархическое мультиплексирование нескольких цифровых каналов (на сегодня его используют до уровня T3, хотя стандарт определяет и T4)
- определить правила функционирования, администрирования и поддержки оптических каналов связи

С самого начала было принято решение использовать в SONET традиционное TDM-мультиплексирование, где вся ширина оптоволоконной линии используется под один канал, который содержит временные слоты подканалов. Поэтому SONET создавали как синхронную систему. У нее есть главные часы, которые тактируют ее работу с частотой 10-9 сек. с высокой точностью.

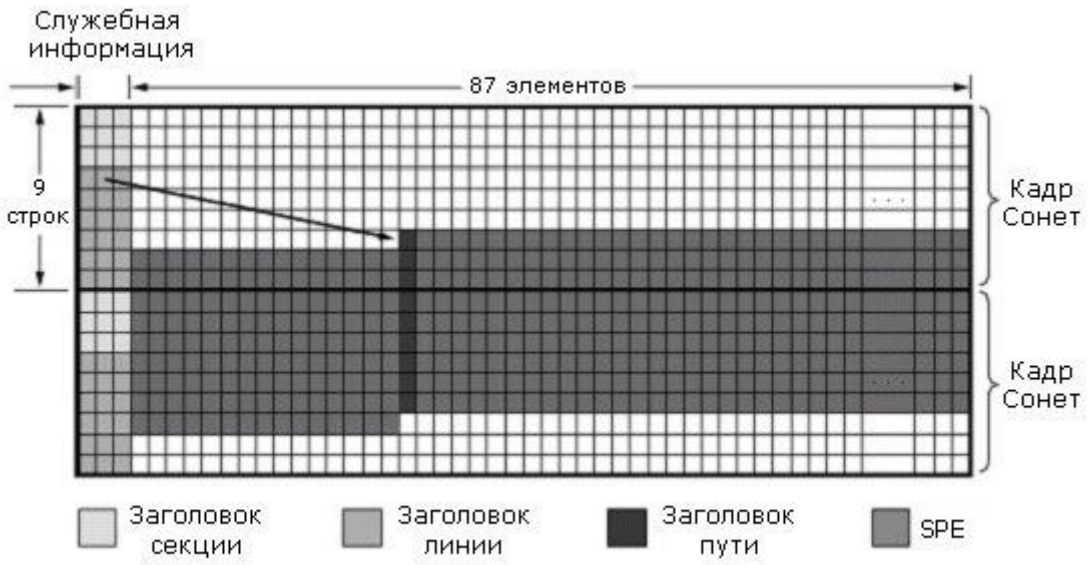
Биты на линии SONET имеют строго выверенную длительность, контролируемую едиными главными часами. Когда позднее для высокоскоростного ISDN был предложен метод передачи, где кадры могли поступать через нерегулярные интервалы времени, то этот метод, в противоположность SONET, был назван асинхронным и известен ныне как ATM.

Система SONET состоит из переключателей, мультиплексоров и повторителей, соединенных оптическими линиями. В терминологии SONET сплошной фрагмент оптоволоконного кабеля между двумя устройствами называется секцией. Канал между двумя мультиплексорами, возможно, с несколькими повторителями между ними, называется линией. Канал между двумя оконечными абонентами называется путем.

Кадр SONET содержит 810 байт и занимает 125 мксек. SONET допускает топологию каналов связи «решетка», но чаще это двунаправленное кольцо. Так как система SONET синхронная, то кадры генерируются строго один за другим без перерывов вне зависимости от того, есть данные на передачу или нет. Скорость в 8000 кадров/сек. как раз соответствует каналам с ИКМ-модуляцией, используемым в цифровой телефонии. Исходя из этого, нетрудно подсчитать, что пропускная способность канала SONET равна 51,84 Мбит/сек.

Для описания кадра SONET представим его 810 байт в виде матрицы 9 строк на 90 столбцов, как показано на рисунке 2-46. Каждый элемент матрицы – один байт. Первые три элемента в каждой строке – это служебная информация, используемая для администрирования и управления передачей. Первые три элемента первых трех строк образуют заголовок секции, в следующих 6 строках – заголовок линии. Заголовки секции генерируются и проверяются в начале и в конце каждой секции. Аналогичным образом поступают на каждой линии с заголовком линии. 8000 кадров в секунду образуют основной канал, называемый Synchronous Transport Signal-1 (STS-1).

Рисунок 2-46. Устройство кадра SONET



Оставшиеся в 87 столбцах и 9 строках 783 байта приходятся на данные пользователей, которые образуют так называемый SPE-конверт (Synchronous Payload Envelope). Учитывая, что в SONET

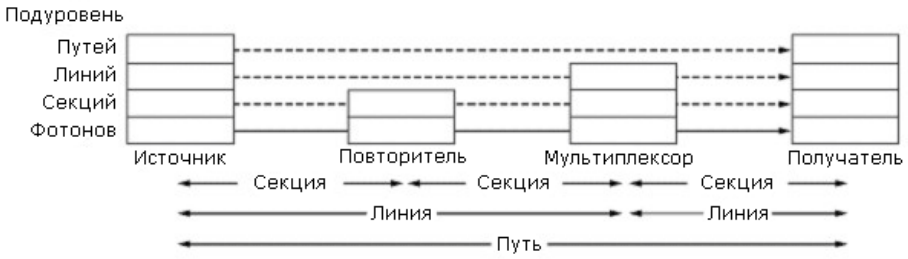
генерируется 8 000 кадров в секунду, получаем, что полезная пропускная способность составит 8000 x 783 x 8 = 50,112 Мбит/сек.

Мультиплексирование множественных потоков данных, называемых в SONET притоками, показано на рисунке 2-47. Мультиплексирование происходит побайтно. Например, когда три STS-1 притока, каждый со скоростью 51,84 Мбит/сек., объединяют в один STS-3 приток со скоростью 155,52 Мбит/сек., мультиплексор сначала берет 1-й байт 1-го притока, затем 1-й байт 2-го притока, затем 1-й байт 3-го. Только после этого он переходит ко вторым байтам этих притоков. Кадр STS-3 состоит из 270x9=2430 байтов и занимает 125 мксек. Таким образом, на этом уровне битовая скорость равна 155,52 Мбит/сек. На рисунке 2-48 приведены основные данные об иерархии мультиплексирования в SONET и SDH.

Скорость в ATM равна 155 Мбит/сек. для того, чтобы сделать SONET и SDH совместимыми с ATM на ранних этапах мультиплексирования.

Наличие заголовков секций, линий и путей говорит о наличии в SONET нескольких уровней протоколов передачи. Их четыре (они показаны на рисунке 2-49). Это уровень фотонов, или физический, уровень секций, линий и путей. Уровень фотонов определяет физические характеристики света и оптики. Уровень секции управляет передачей внутри секции, генерацией заголовка в начале секции и проверкой этого заголовка в конце секции.

Рисунок 2-49. Уровни протоколов передачи SONET



Уровень линии отвечает за мультиплексирование нескольких притоков разных линий в поток на одной линии на одном конце и демультиплексирование на другом. Уровень пути управляет передачей между оконечными пользователями.

Билет № 14.

Телефонные сети: структура, методы коммутации.

Коммутация

Третий важный компонент телефонной сети – работа телефонных станций, или, как мы их еще называем, узлов коммутации, а точнее, их основу – коммутаторы. В телефонных сетях используются два разных способа коммутации: коммутация каналов и коммутация пакетов.

Коммутация каналов и коммутация пакетов.

На рисунке 2-50 показаны схемы работы коммутатора при коммутации каналов и при коммутации пакетов. Каждый из шести прямоугольников на рисунке 2-50 (а) представляет узел коммутации определенного уровня. В данном случае у каждого узла по три входящие и по три исходящие линии. Когда по одной из входящих линий поступает сигнал вызова, то он направляется по одной из исходящих линий. В результате входящая и исходящая линии замыкаются напрямую, образуя как бы единую линию. На рисунке это показано пунктирной линией.

Рисунок 2-50. Коммутация каналов (а) и коммутация пакетов (b)

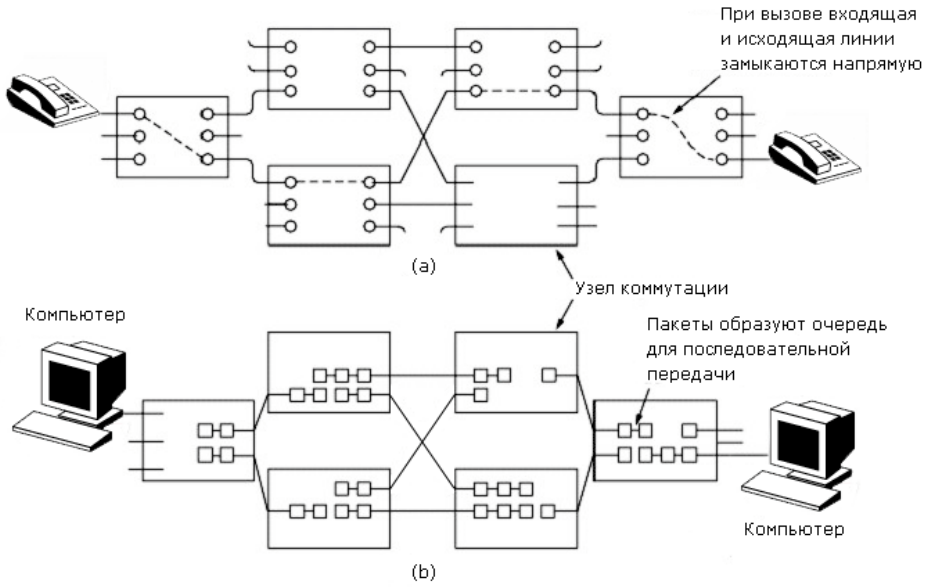


Рисунок 2-50 (а) существенно упрощает реальную ситуацию. Мы уже отмечали, что между узлами коммутации используют магистрали с мультиплексированием сотен и тысяч вызовов одновременно. Эти магистрали не обязательно используют кабели. Они могут быть реализованы с помощью радиорелейной связи. Однако в целом основная идея коммутации каналов на этом рисунке отображена верно: при наличии вызова создается физическое соединение за счет коммутации нескольких каналов, которое сохраняется до тех пор, пока не будут переданы данные и не поступит команда разрыва соединения. Для создания соединения сигнал вызова должен пройти от точки возникновения до места назначения и быть подтвержден сигналом, что соединение успешно создано.

Основной особенностью коммутации каналов является то, что создается канал точка-точка, до того как данные начнут передаваться. Время соединения исчисляется секундами, а при удаленных звонках - до минуты. Прежде чем соединение возникнет, сигнал вызова должен проложить маршрут. Это требует времени. Для многих компьютерных приложений такая большая задержка неприемлема или нежелательна.

Если соединение установлено, то задержка при передаче составит 5 мксек. на 1000 км. При установленном соединении нет опасности, что во время разговора вы услышите сигнал «занято» из-за нехватки свободных линий у какого-либо коммутатора или малой пропускной способности одного из каналов, через который проходит соединение.

Альтернативой коммутации каналов является коммутация сообщений. 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. Здесь не нужно было создавать соединение заранее. Однако для такого способа передачи необходимо обеспечить на каждом узле коммутации нужное количество памяти для буферизации любого сообщения, сколь угодно большого. Для преодоления этого недостатка был предложен метод коммутации пакетов: сообщение разбивают на фрагменты фиксированной длины. Эти фрагменты называются пакеты. Пакеты одного сообщения передают от одного узла коммутации к другому, пока они не достигнут места назначения. Каждый пакет можно передавать независимо от других. Поскольку пакет имеет фиксированную длину, то абонент не может монополизировать линию, а поэтому возможен интерактивный режим работы. Одну и ту же линию могут разделять пакеты разных пользователей. Другое достоинство коммутации пакетов – конвейерность: второй пакет можно отправить, не дожидаясь когда первый достигнет места назначения. Послав второй, можно начать передачу третьего и т.д.

Основные различия между коммутацией каналов и коммутацией пакетов приведены ниже:

• При коммутации каналов создается соединение, пропускная способность которого полностью резервируется за двумя абонентами, вне зависимости от того, какая пропускная способность реально им потребуется. При коммутации пакетов физическая линия может быть использована пакетами разных абонентов. Следует иметь в виду, что так как при коммутации пакетов не происходит жесткого закрепления канала, то резкое увеличение потока пакетов в узле коммутации (в случае коммутации пакетов эти узлы называют маршрутизаторами), может привести к их перегрузке и потере части пакетов.

• При коммутации каналов гарантировано, что все данные поступят абоненту и в том порядке, в каком их послали. При коммутации пакетов из-за ошибок маршрутизации пакеты могут быть направлены не по назначению, сохранение их исходного порядка получателю не гарантируется.

• Коммутация каналов абсолютно прозрачна для абонентов. Они могут пересылать данные в любой кодировке и формате. При коммутации пакетов формат и способ кодировки пакетов задан заранее и определяется оператором связи.

• При коммутации пакетов плата взимается за время соединения и число переданных пакетов. При коммутации каналов плата берется исключительно за время и длину соединения.

Иерархия узлов коммутации.

Совокупность узлов коммутации, оконечных абонентских устройств и соединяющих их каналов и линий связи называют сетью телефонной связи.

Сети связи создаются для передачи информации между абонентами и бывают коммутируемыми и некоммутируемыми. Сеть называется коммутируемой, когда тракт передачи информации создается по запросу абонента на время сообщения, и некоммутируемой, когда тракт передачи информации обеспечивается постоянным соединением между определенными абонентами и нет необходимости в коммутации. Телефонные сети являются коммутируемыми. Общегосударственная телефонная сеть (ОАКТС) состоит из междугородной телефонной сети и зоновых телефонных сетей. Междугородная телефонная сеть обеспечивает соединение автоматических междугородных телефонных станций (АМТС) различных зон.

Зоновая телефонная сеть состоит из местных телефонных сетей, расположенных на территории зоны, и внутризоновой телефонной сети, которая соединяет между собой эти сети. Местные телефонные сети разделяются на городские, обслуживающие город и ближайшие пригороды (ГТС), и сельские (СТС), обеспечивающие связь в пределах сельского административного района.

Учрежденческо-производственная телефонная сеть (УПТС) служит для внутренней связи предприятий, учреждений, организаций и может быть соединена с сетью общего пользования либо быть автономной.

Зоновая телефонная сеть включает всех абонентов определенной территории, охватываемой единой семизначной нумерацией, и является частью ОАКТС. Территории зоновых сетей совпадают с территориями административных областей (республик). В зависимости от конфигурации области и телефонной плотности территории нескольких областей могут быть объединены в одну зону и, наоборот, одна область может быть разделена на две зоны и более. Зоновая сеть включает в себя ГТС и СТС, причем на территории одной зоны могут быть несколько ГТС и СТС. Крупные города с семизначной нумерацией выделяются в отдельные зоны.

Сельские телефонные сети охватывают более обширные территории, чем городские, но плотность телефонных аппаратов значительно меньше. Поэтому емкость автоматических телефонных станций АТС в сельских местностях значительно меньше, чем в городах.

Городская телефонная сеть состоит из комплекса сооружений (станционное оборудование, здание, линейные сооружения, абонентские устройства и др.), обеспечивающих телефонной связью

абонентов города и прилегающих к нему пригородов. Стоимость линейных сооружений в значительной степени зависит от принципа построения ГТС и ее емкости.

При емкости ГТС от 10 000 до 50 000 номеров территория города делится на районы, обслуживаемые районными АТС (РАТС). Протяженность абонентских линий районированной ГТС сокращается, так как АТС приближается к местам установки телефонных аппаратов. Районные АТС соединяются соединительными линиями (СЛ) по принципу «каждая с каждой», при этом достигается более высокое использование пучков СЛ. Так как телефонные сообщения, возникающее на каждой РАТС, распределяются по небольшому числу направлений, пучки СЛ между РАТС получаются крупными.

Нумерация абонентов.

Нумерация абонентов может быть закрытой и открытой. Нумерация называется закрытой (единой), если абонент вызывается набором одного и того же номера независимо от местонахождения вызывающего пункта. При закрытой системе нумерации номер вызывающего абонента не зависит от вида связи – местной, зоновой или междугородной. Нумерация называется открытой, если зависит от вида связи: местной, зоновой или междугородной.

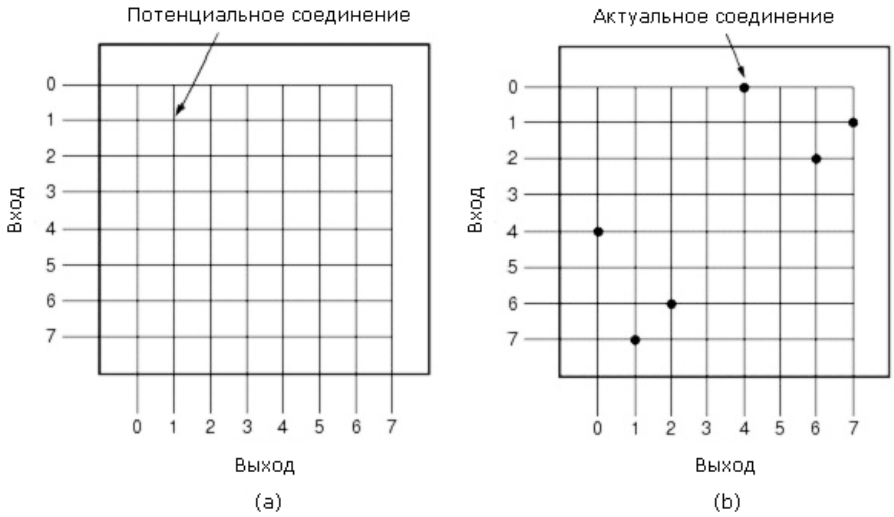
В ОАКТС принята открытая система нумерации с постоянными кодами. Междугородный номер абонента сети страны содержит десять цифр и имеет структуру АВСабхххxx, где АВС – постоянный трехзначный код зоны, аб – код местной сети или стотысячной группы абонентов, последние пять цифр хххxx – пятизначный номер абонента. В соответствии с принятым еще в СССР зоновым принципом нумерации вся территория разделена на 166 телефонных зон с единой семизначной нумерацией абонентов.

В ГТС нашей страны, как правило, применяют закрытую систему нумерации. Число знаков в номере абонента зависит только от емкости ГТС. Если на ГТС принята семизначная нумерация, то местный и зоновый номера совпадают (например, ГТС Москвы, Санкт-Петербурга, Киева). В автоматической международной телефонной связи абонент должен набрать: цифры 8, 10, международный номер (где 10 – индекс выхода на автоматическую международную телефонную сеть). Полный международный номер вызываемого абонента может иметь 11-12 знаков.

Коммутаторы каскадные.

Теперь, познакомившись с иерархией телефонных станций (узлов коммутации), давайте рассмотрим, как устроен сам коммутатор. Самый простой вид коммутаторов - это прямой коммутатор n x n, у которого есть n входных и n выходных линий. Он показан на рисунке 2-52. В каждой точке пересечения стоит полупроводниковый переключатель, который замыкает соответствующие линии.

Рисунок 2-52. Прямой коммутатор



Основной недостаток этого типа коммутаторов - квадратичный рост сложности при увеличении n. Сложность коммутатора измеряется в количестве точек пересечения. Даже если учесть, что в случае дуплексных линий и отсутствии самосоединений нам требуется только половина пересечений (выше или ниже диагонали), то все равно нам надо порядка n(n-1)/2 переключателей. При n=1000 на кристалле можно поместить такое количество переключателей, но приделать к нему 2000 ножек невозможно. Поэтому такие прямолинейные решения возможны лишь для небольших организаций.

На рисунке 2-53 показан каскадный коммутатор. Идея построения этого типа коммутаторов такова: разделить простой коммутатор на части, соединить эти части между собой промежуточными дополнительными коммутаторами. Рассмотрим пример трехслойного каскадного коммутатора. В первом слое N входных линий разбиваются на группы по n линий в каждой. На втором слое N/n прямых коммутаторов n x k линий каждый соединяются с k коммутаторами N/n x N/n линий. Третий каскад повторяет первый в обратном порядке: не n x k, а k x n.

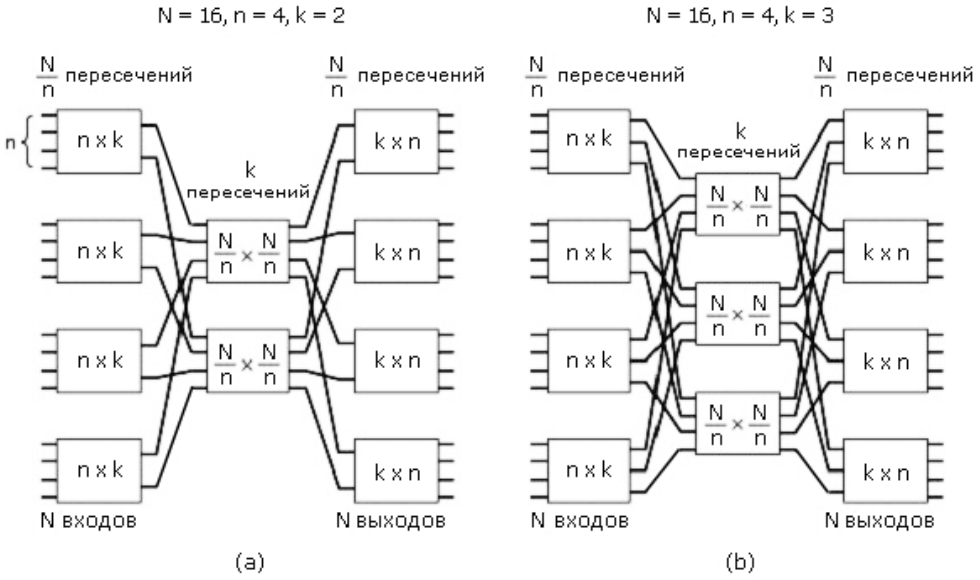
Подсчитаем сложность такого каскадного коммутатора. Первый каскад содержит

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

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

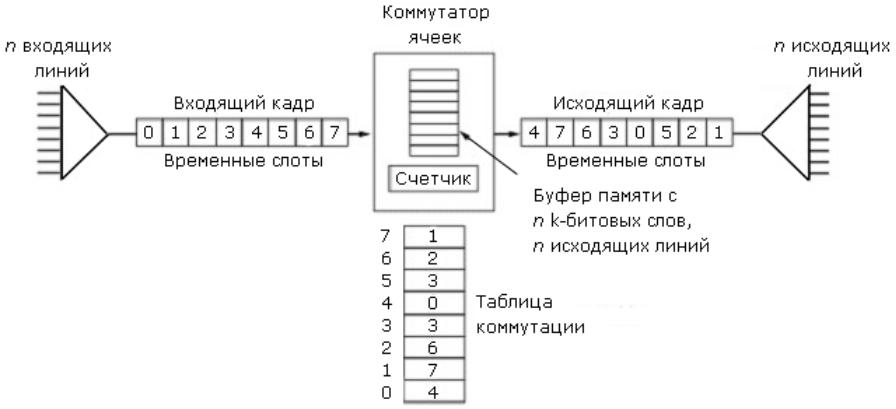
Каскадные коммутаторы имеют недостаток - блокировка коммутаторов второго слоя. На рисунке 2-53 (a) второй слой может коммутировать одновременно только 8 звонков. Девятый звонок буден заблокирован. Коммутатор на 2-53 (b) лучше. В нем 12 входов на втором каскаде, но он и дороже. Клос (Clos) в 1953 году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

Рисунок 2-53. Устройство каскадных коммутаторов



Коммутаторы с разделением времени.

Рисунок 2-54. Коммутатор с разделением времени



На рисунке 2-54 показан совершенно другой способ коммутации – коммутация с разделением времени. Пусть у нас есть n линий, которые нам надо коммутировать. Эти линии сканируют последовательно одна за другой в течение определенного временного слота. Образуется кадр из n ячеек по k битов в каждой. Например, в стандарте E1 каждая ячейка содержит по 8 бит, кадр – 32 ячейки, а всего за секунду проходит 8000 кадров.

Затем кадр попадает в коммутатор ячеек. Коммутатор ячеек переставляет ячейки в соответствии с таблицей коммутации. Обработка кадра происходит следующим образом. Входной кадр записывается в память в том порядке, как ячейки считывались с линий. Затем ячейка считываются из памяти в порядке, задаваемом таблицей коммутации.

Ясно, что таблица коммутации - это вектор перестановок, а скорость коммутации ограничена скоростью считывания из памяти. Например, если временной слот - 125 мксек. и нам надо обработать кадр из n ячеек, а время считывания из памяти T мксек., то 2nT=125 мксек. или n=125/(2T). Если скорость памяти 100 нсек., то мы сможем обработать не более 625 линий.

Билет № 15.

Цифровые сети с интегрированным сервисом (ISDN сети).

МКТТ в 1984 году собрал конгресс, где было принято решение о создании новой полностью цифровой телекоммуникационной сети, которой дали название «Цифровая сеть с интегрированным сервисом» (ISDN - Integrated Service Digital Network).

ISDN задумывалась как всемирная телекоммуникационная сеть, которая должна была заменить телефонные сети. С точки зрения приложений, ISDN должна была поддерживать передачу голоса, звука, изображения и данных.

ISDN-телефон по замыслу проекта должен был обеспечивать самый разнообразный сервис: программируемые функции, показ номера телефона, от которого поступил звонок, имя звонящего, умение работать с компьютером - выдать запрос к базе данных и высветить на экране ответ, переадресовать звонки, удаленный доступ к своему телефону, автоматические звонки в скорую помощь, полицию, пожарную службу в случае опасности и т.д. Эта технология должна обеспечивать подключение прямо в сеть, без использования модемов, цифровые приборы и оборудование.

Проект ISDN постоянно находится в развитии. Он оказывает огромное влияние как на операторов связи, так и на производителей оборудования. В рамках проекта ISDN значительные усилия сосредоточены на стандартизации интерфейсов разных уровней.

Несмотря на то что ISDN еще не достиг того же уровня распространения, как обычный телефон, уже появилось второе поколение этого проекта. Первое поколение называют narrowband ISDN – узкополосный, или низкоскоростной ISDN (N-ISDN). Он поддерживает аналоговые и цифровые каналы с пропускной способностью 64 Кбит/сек. и основан на коммутации каналов. Одним из важных технических новшеств N-ISDN стал метод передачи Frame Relay.

Второе поколение ISDN, называемое broadband ISDN, – широкополосный, или высокоскоростной ISDN, поддерживает высокую скорость передачи данных (сотни Мбит/сек.) и функционирует на основе коммутации пакетов. Одним из основных технических новшеств B-ISDN стал асинхронный метод передачи (ATM).

Принципы ISDN.

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

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

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

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

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

• Для OSI-приложений уже создано много стандартов. Пример - HDLC, уровень 3 в стандарте X.25 для доступа к сервису с коммутацией пакетов в ISDN.

• Новые ISDN-стандарты могут быть основаны на уже существующих стандартах, тем самым сокращается стоимость их реализации.

• Стандарты разных уровней можно независимо развивать и реализовывать.

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

Архитектура сетей N-ISDN.

Основой ISDN-архитектуры является концепция битового потока в цифровом тракте или просто цифрового тракта между пользователем и транспортной средой, через которую поток битов передается. При этом не важно, как был сформирован этот поток битов - телефоном, факс-машиной, компьютером и т.п. Важно, что биты можно передавать по тракту в обоих направлениях.

Цифровые тракты могут мультиплексировать с разделением по времени несколько независимых каналов. Концепция цифрового тракта строго специфицирована. В этой спецификации определены интерфейсы, формат цифрового потока и правила мультиплексирования потоков. Было разработано два стандарта: один для низкоскоростной передачи (для домашнего использования) и высокоскоростной (для бизнес приложений).

Кофигурация для дома. Поставщик сервиса, или, как его еще называют, оператор, устанавливает оконечное сетевое устройство - NT1. NT1 соединено, с одной стороны, с ISDN-оборудованием пользователя, а с другой - с ISDN-устройством обмена в помещении поставщика сервиса. NT1 может быть удалено от ISDN-устройства обмена на несколько километров и соединено с ним витой парой, оставшейся от обычного телефонного соединения. К одному NT1 может быть подключено до 8 ISDN-устройств пользователя. С точки зрения пользователя, граница сети передачи данных – NT1-устройство.

Для производственных такая конфигурация не подходит, так как может потребоваться существенно больше оконечных ISDN-устройств, функционирующих одновременно, например, телефонов. Поэтому в промышленности используется другая конфигурация. В этой конфигурации используется устройство NT2 - PBX (Private Branch eXchange), которое мы будем называть устройством обмена второго уровня. PBX соединен с NT1 и обеспечивает связь с телефонами, терминалами в офисе и их мультиплексирование. Таким образом, PBX - это по существу небольшой ISDN-коммутатор.

МКТТ определило четыре вида точек подключения для ISDN-сетей: R, S, T, U. U-соединение определяет соединение между ISDN-устройством обмена и NT1. На сегодня это либо медная витая пара, либо оптоволоконная линия. T - определяет подключение NT1 к оборудованию в офисе пользователя. S - подключение PBX- и ISDN-терминалов. R - адаптер между ISDN-терминалом и не-ISDN оборудованием.

Подключение типа T позволяет подключить 23 канала по 64 Кбит/сек., что хорошо укладывается в стандарт T1 в США и Японии, и 30 каналов по 64 Кбит/сек. для Европы. Однако надо подчеркнуть, что для одного N-ISDN терминала доступна скорость не более 64 Кбит/сек.

Битовый тракт в ISDN подразумевает мультиплексирование нескольких стандартных каналов. Стандарты ISDN определяют следующие типы каналов:

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

Канал типа B подразумевает четыре вида соединений:

- С коммутацией каналов. Абонент инициирует вызов, под воздействием которого устанавливается соединение с коммутацией каналов, которое соединяет абонента с другим абонентом сети.
- С коммутацией пакетов. Абонент подключен к узлу сети с коммутацией пакетов и обменивается данными с другими абонентами посредством протоколов X.25.
- Соединение Frame Relay. Абонент подсоединяется к узлу сети Frame Relay, через которую происходит обмен данными.
- Постоянное соединение. Это соединение с другим абонентом, которое было установлено заранее и динамически изменено быть не может. Это соединение подобно выделенной линии.

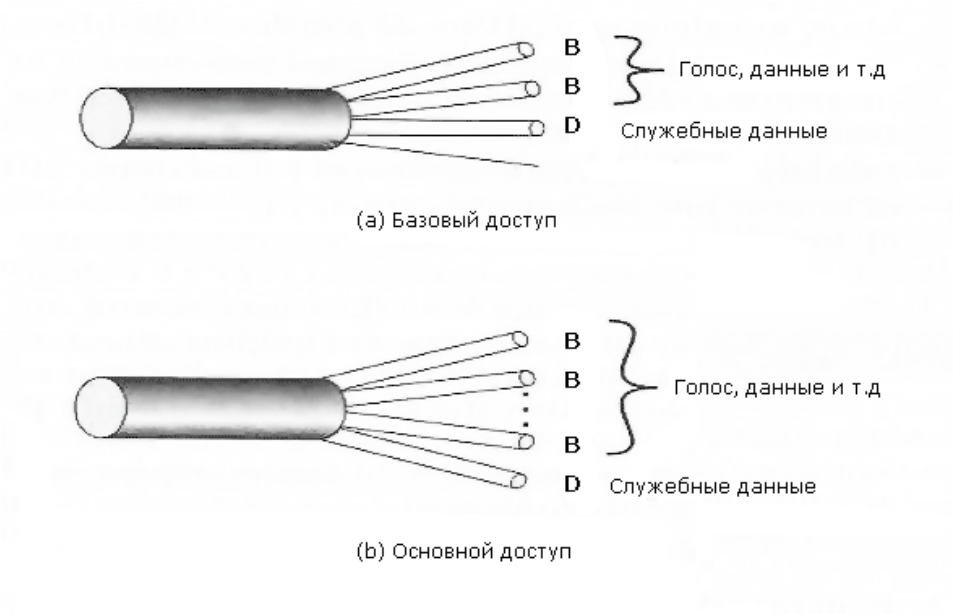
Канал типа D служит двум целям. Во-первых, он служит для управления коммутацией каналов, инициированной вызовом по интерфейсу, с абонентом через канал B. Кроме этого, канал D можно использовать, когда он свободен, для коммутации пакетов или получения данных от оборудования на низкой скорости (до 100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

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

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

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

Рисунок 2-60. Структура ISDN-каналов



Основной доступ предназначен для пользователей, которым нужна высокая скорость передачи. Как видно на рисунке, есть несколько вариантов основного доступа: для поддержки стандарта T1 и для поддержки стандарта E1.

ISDN-сети предоставляют четыре вида соединений конечных пользователей:

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

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

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

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

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

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

Теперь любой абонент ISDN-сети может обмениваться данными через X.25 с любым абонентом PSPDN-сети. Если между абонентом ISDN-сети и PH-узлом PSPDN-сети есть постоянное соединение, то абонент с помощью X.25 может сразу установить внутреннее соединение с другим пользователем. Если между ними можно установить соединение с коммутацией каналов, то, кроме B-канала, нужен D-канал.

Когда коммутацию пакетов обеспечивает ISDN-сеть, то управление пакетами обеспечивает либо специальное устройство, либо устройство обмена. Это устройство названо PH-устройством. Пользователь может быть соединен с PH-устройством либо B-каналом, либо D-каналом. В случае B-канала соединение может быть либо постоянным, либо коммутируемым. Этот случай мы уже обсуждали выше.

В случае D-канала ISDN-сеть обеспечивает постоянное соединение с PH-устройством ISDN-сети. Специальные меры на канальном уровне X.25 позволяют разделить в D-канале поток пакетов X.25 от управляющих пакетов ISDN.

Высокоскоростные ISDN-сети и ATM-сети.

МКТТ быстро осознало отставание N-ISDN и предложило новое поколение ISDN-сетей - B-ISDN (Broadband ISDN), высокоскоростной ISDN. Фактически, B-ISDN - это цифровые виртуальные каналы, по которым движутся пакеты фиксированной длины (ячейки) со скоростью 155 Мбит/сек.

Этой скорости вполне достаточно, чтобы обслуживать даже такие приложения, как высококачественное телевидение и, похоже, что эта скорость будет в ближайшие годы увеличена.

Основу B-ISDN составляет ATM-метод. ATM - технология с коммутацией пакетов.

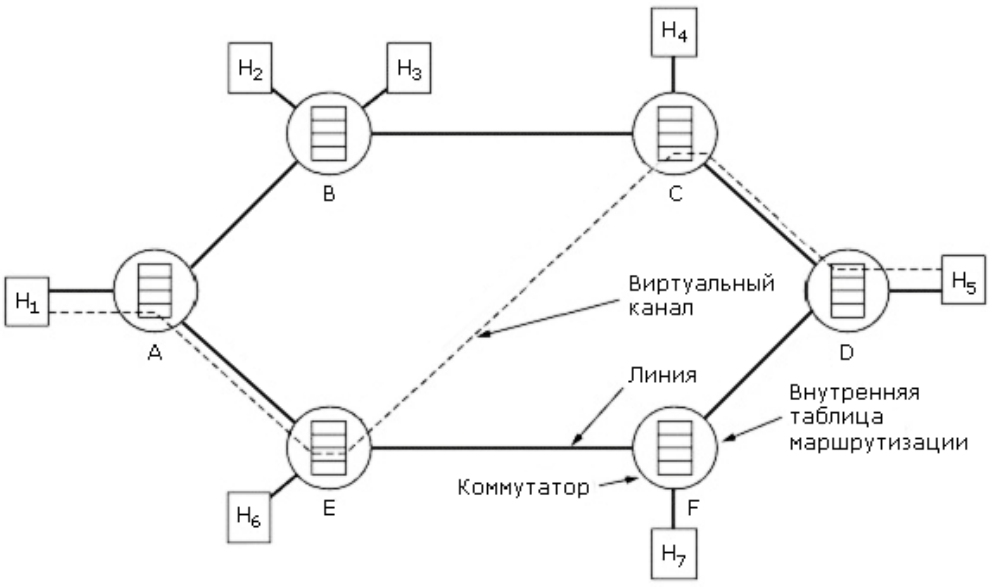
Ясно, что для B-ISDN витая пара – основной вид абонентской линии, скорее всего, не подойдет. Существующие телефонные коммутаторы не годятся и должны быть заменены коммутаторами нового поколения, работающими на иных принципах. Единственное, что, похоже, удастся сохранить - оптоволоконные магистрали.

Виртуальные каналы и коммутация каналов.

B-ISDN построен на своего рода компромиссе между коммутацией каналов и коммутацией пакетов. Сервис в этих сетях ориентирован на соединения, но эти соединения не являются коммутированными физическими каналами. Это - коммутируемые виртуальные каналы. Есть два вида виртуальных каналов - постоянные и коммутируемые. Постоянные каналы устанавливает оператор по запросу пользователя. Установка такого виртуального канала занимает обычно несколько дней, а период его действия, как правило, – несколько месяцев или лет. Коммутируемые каналы, подобно телефонным каналам, устанавливаются динамически по требованию и перестают действовать сразу после их использования.

В сети с коммутацией каналов установить соединение означает создать физическое соединение между источником и получателем. Это хорошо видно на системах с каскадными коммутаторами, в транспортных средах с коммутаторами с разделением времени это не столь очевидно. В сетях с виртуальными каналами, таких как ATM, установление соединения означает, что маршрут между источником и получателем выбран. Таким образом, в таблицах коммутаторов заранее известно, по какому маршруту направлять тот или иной пакет. На рисунке 2-63 показан пример коммутации виртуальных каналов между H1 и H5. Когда пакет поступает на коммутатор, его заголовок просматривается, что позволяет определить, к какому виртуальному соединению этот пакет принадлежит, и направляется по надлежащей физической линии.

Рисунок 2-63. Пример виртуального соединения в среде с коммутацией пакетов



Установление постоянного соединения означает, что в таблицах коммутаторов заранее прописаны соответствующие значения, независимо от того, есть трафик или нет.

Билет № 16.

Передача данных в ATM сетях.

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

В ATM нет строго порядка поступления ячеек от различных источников. Ячейки могут поступать от разных источников и в разном порядке. Не важно даже, чтобы поток ячеек от одного компьютера был непрерывен. Если возникают разрывы, то они заполняются ячейками ожидания.

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

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

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

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

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

ATM-переключатели.

Общая схема организации ATM-переключателя: Есть набор входных линий, по которым ячейки поступают в переключатель, и, как правило, такое же число выходных линий, по которым ячейки двигаются после коммутации. Обычно переключатель работает синхронно: длительность цикла строго фиксирована. В течение каждого цикла просматриваются все входные линии и, если на линию к этому моменту целиком поступила ячейка, то она считывается и передается в центр коммутации, а затем на выходную линию.

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

Ячейки поступают со скоростью 155 Мбит/сек. Учитывая размер ячейки в 53 байта, получаем около 360 000 ячеек/сек. Таким образом, на обработку одной ячейки приходится около 2,7 мксек. Выпускаемые на сегодня переключатели имеют от 16 до 1024 входных линий, т.е. переключатель должен быть в состоянии обрабатывать за 2,7 мксек от 16 до 1024 ячеек. При скорости 622 Мбит/сек. переключающий центр должен обрабатывать очередную порцию ячеек за 700 нсек. Благодаря тому, что ячейки фиксированной длины и небольшого размера (53 байта), коммутация на таких скоростях становится возможной. При переменной длине и большем размере ячейки задача создания ATM-переключателя была бы намного сложнее.

Все ATM-переключатели должны удовлетворять следующим требованиям:

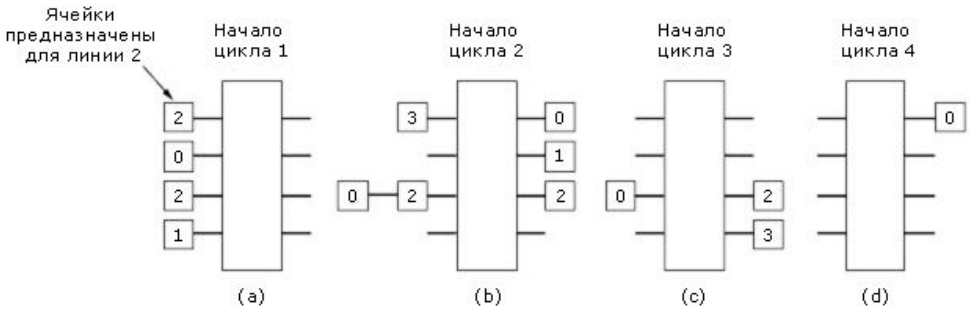
- терять как можно меньше ячеек
- никогда не менять порядок поступления ячеек по каждому виртуальному соединению

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

Одна из ключевых проблем конструкции ATM-переключателей состоит в следующем: что делать, когда сразу по нескольким линиям пришли ячейки, которые должны быть отправлены по одной и той же выходной линии? Напрашивается решение: взять одну ячейку, обработать ее, а другую сбросить. Но в силу требования 1 оно не годится.

Возможно другое решение: буферизовать ячейки на входе. Идея этого решения показана на рисунке 2-66. Пусть в начале цикла 1 (рисунок 2-66(a)) поступило четыре ячейки, две из которых должны быть отправлены по линии 2. Поскольку из-за линии 2 возник конфликт, то только три ячейки передаются на выходные линии. Поэтому к началу цикла 2 (рисунок 2-66(b)) на выходе переключателя появятся три ячейки, но на вход поступят новые. К началу цикла 3 (рисунок 2-66(c)) на входе останется только одна ячейка, и очередь рассосется только на четвертом цикле. В случае буферизации на входе надо следить за тем, чтобы дисциплина обслуживания возникающих очередей была бы справедливой и равномерно обслуживала очереди на всех линиях.

Рисунок 2-66. Буферизация ячеек на входе

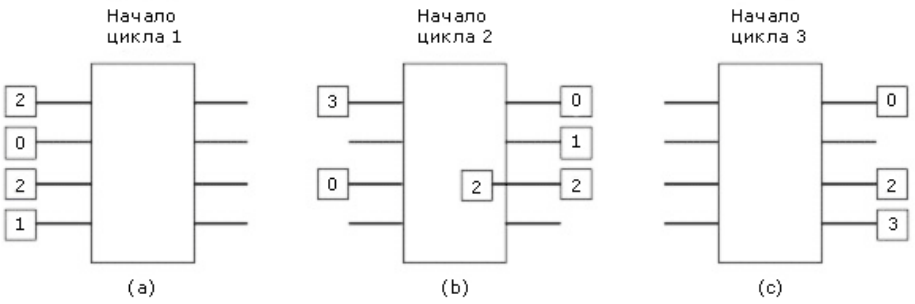


Недостаток этого решения в том, что очередь на входе может блокировать даже те ячейки, которые должны быть перекоммутированы на линии, на которых нет конфликтов. Поэтому по

соответствующему виртуальному соединению скорость упадет. Этот эффект называется блокировкой на входе. Кроме этого, буферизация ячейки на входе требует дополнительной логики в схемах, что усложняет конструкцию ATM-переключателя.

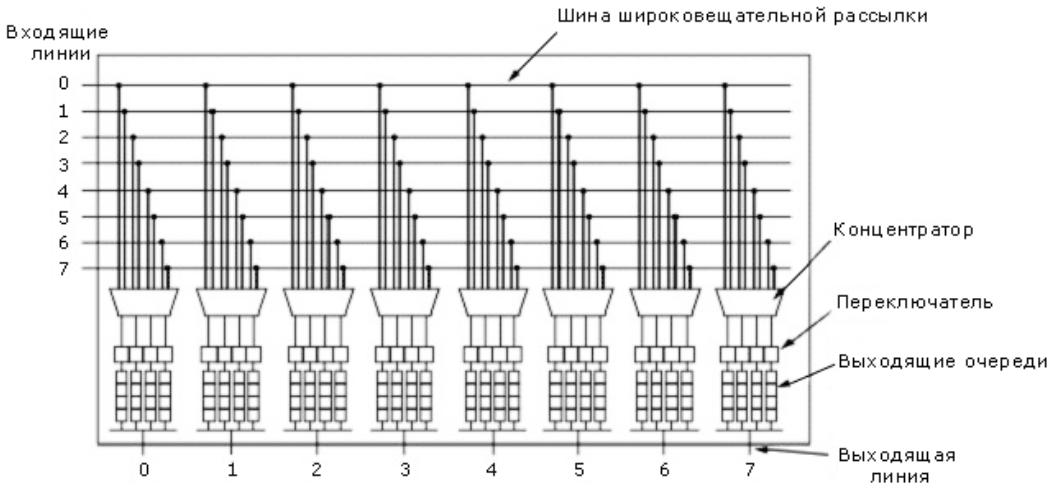
Альтернативным решением может быть буферизация на выходе. Это решение показано на рисунке 2-67. Если несколько ячеек должны уйти по одной и той же линии, то они передаются на выход и буферизуются там. Это требует меньше циклов, в нашем примере только 3. В общем случае Karol 1987 показал, что буферизация на выходе эффективнее, чем буферизация на входе.

Рисунок 2-67. Буферизация на выходе



Рассмотрим конструкцию ATM-переключателя, использующего буферизацию на выходе. Этот тип переключателей называется переключатель выталкивающего типа. Он показан на рисунке 2-68 для конфигурации 8x8 линий. Здесь каждая входная линия соединена с шиной, к которой подключены все выходные линии. Каждая входная шина имеет свой механизм управления, не зависящий от других, что существенно упрощает конструкцию.

Рисунок 2-68. Переключатель выталкивающего типа



У каждой поступающей ячейки аппаратно анализируется заголовок, чтобы определить, какому виртуальному соединению она принадлежит. Затем, с помощью таблицы коммутации, определяется выходная линия, через которую эта ячейка должна покинуть переключатель. Пересечение с соответствующей выходной линией активизируется, и, когда ячейка доходит до этого пересечения, она попадает в буфер. Ресурсов переключателя достаточно, чтобы буферизовать на одном выходе

ячейки со всех входов, если это потребуется, или размножить ячейки, если их надо разослать по нескольким виртуальным соединениям.

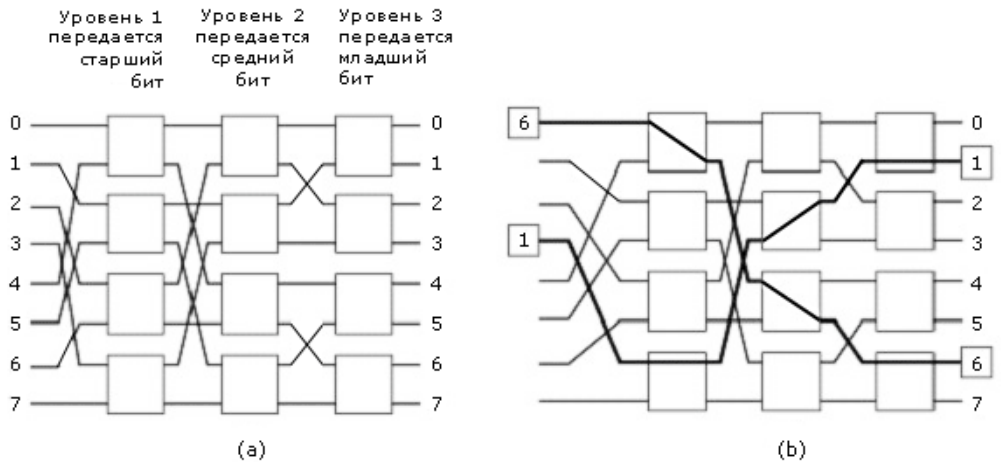
Естественно было бы буферизовать все конфликтующие ячейки в выходном буфере. Однако для переключателей, например, на 1024 линий, нам потребовалось бы 1024 буферов по 1024x53 байтов. Слишком много! Выход из этой ситуации - выделение лишь n байтов на буфер, где n – параметр настройки. Если конфликтующих ячеек больше, то ячейки, не попавшие в буфер, сбрасываются. Здесь опять-таки надо быть осторожным, определяя на каких входных линиях сбрасывать ячейки, из каких выходных буферов выталкивать ячейки на очередном цикле так, чтобы не было дискриминации. Регулируя параметр n, можно варьировать стоимость и число сбрасываемых ячеек, что влияет на цену переключателя.

Переключатели Батчера-Баньяна.

Основным недостатком переключателей выталкивающего типа является то, что центр коммутации - простой коммутатор, а это означает, что его сложность растет квадратично от числа коммутируемых линий. Из рассмотрения принципов построения коммутаторов для коммутации каналов мы уже знаем, что одно из решений - каскадные коммутаторы. Аналогичное решение возможно и для коммутации пакетов.

Это решение называют переключателем Батчера-Баньяна. Как и переключатели выталкивающего типа, переключатель Батчера-Баньяна синхронный, т.е. за один цикл он может обрабатывать несколько входных линий. На рисунке 2-69(а) показан трехступенчатый 8x8 переключатель Баньяна. Он называется так, поскольку похож на корни баньянового дерева. В баньяновых переключателях для каждого входа существует ровно один путь к любому из выходов. Маршрутизация пакета происходит в каждом узле на основе адреса выходной линии, которой должен достичь пакет. Адрес выходной линии определяют на входе по номеру виртуального соединения. В данном случае трехбитовый номер впереди ячейки используется в каждом узле для маршрутизации.

Рисунок 2-69. Трехступенчатый 8x8 переключатель Баньяна



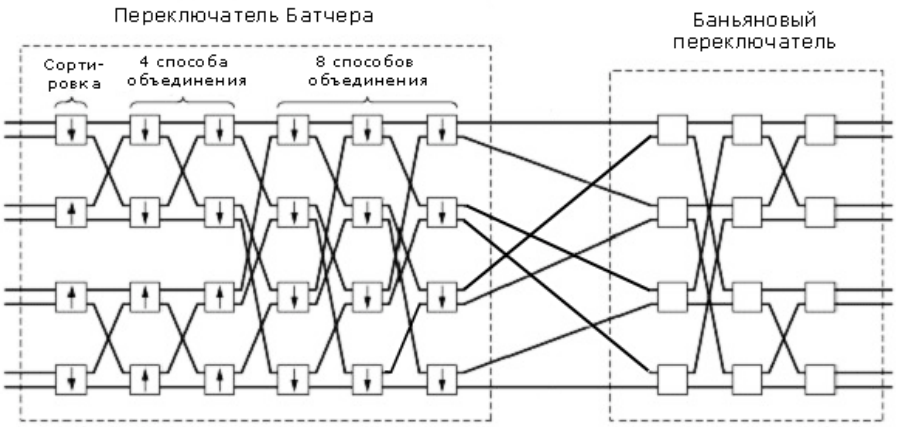
В каждом из 12 переключающих элементов есть два входа и два выхода. В зависимости от значения соответствующего разряда ячейка направляется либо в порт 0, либо в порт 1. Если обе ячейки, поступившие на вход одного и того же коммутирующего элемента, должны быть направлены на один и тот же порт, то направляется одна, а вторая сбрасывается.

Итак, адрес выходной линии анализируется в каждом элементе слева направо. Например, код 001 означает, что соответствующая ячейка будет направлена на верхний, потом еще раз на верхний, а затем на нижний порты. Коллизии в баньяновской сети возникают, когда в одном и том же элементе

в одно и тоже время надо использовать один и тот же порт. В зависимости от распределения ячеек на входе баньяновская сеть либо будет терять ячейки, либо нет.

Идея Батчера состояла в том, чтобы переставить ячейки на входах так, чтобы в баньяновской сети конфликтов не возникало. Для сортировки входов Батчер в 1968 году предложил специальный переключатель. Подобно баньяновскому переключателю, переключатель Батчера строится из элементов 2x2, работает синхронно и дискретно. В каждом элементе выходные адреса ячеек сравниваются. Больший направляется по стрелке, а меньший - в противоположном направлении. Если ячейка одна, то против стрелки. Подчеркнем, что сравниваются не отдельные биты, а весь адрес как число.

Рисунок 2-71. Действие переключателя Батчера-Баньяна



На рисунке 2-71 показан переключатель Батчера-Баньяна 8x8. Сложность операции перестановки для устройства Батчера - $n \log_2 n$. Ячейки, отсортированные Батчеровской сетью, подаются на вход сети баньяна, где они пересылаются без конфликтов.

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

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

Было предложено несколько промышленных переключателей этого типа, которые по-разному преодолевают эти недостатки. Основная их идея заключается во встраивании между переключателем Батчера и сетью баньяна специальной схемы, которая позволяет преодолеть эти две трудности.

Билет № 17.

Спутниковые системы связи: геостационарные спутниковые системы (примеры).

Спутник связи имеет несколько приемопередатчиков - транспондеров, или стволов. Каждый транспондер слушает свою часть спектра, усиливает полученный сигнал и передает его обратно на землю в нужном направлении, на нужной частоте, отличной от частоты приема, чтобы избежать интерференции с принимаемым сигналом. Возвращаемый луч может быть по желанию либо широким, покрывая большую территорию, либо наоборот узконаправленным.

Согласно третьему закону Кеплера, период вращения спутника пропорционален радиусу орбиты в степени 3/2. На высоте примерно 36000 км над экватором период вращения спутника будет равен 24 часам. Такой спутник наблюдателю на экваторе будет казаться неподвижным. Благодаря этой неподвижности можно существенно упростить устройство наземной приемно-передающей антенной системы.

Из-за интерференции волн неразумно было бы размещать такие спутники ближе, чем 2 градуса экваториальной плоскости друг от друга, если они работают на одинаковых частотах. Таким образом, в одно и тоже время на экваториальной орбите может находиться не более 180 спутников, работающих на одной и той же частоте. Так как часть из этих орбит зарезервирована не только для целей связи, то спутников связи на самом деле меньше.

Обычно спутник связи имеет 12-20 транспондеров с полосой пропускания 36-50 МГц каждый. Транспондер с пропускной способностью в 50 Мбит/сек. может быть использован для передачи одного потока данных на скорости 50 Мбит/сек., либо для передачи 800 телефонных разговоров на скорости 64 Кбит/сек. каждый, либо иначе комбинируя скорости и количество передаваемых потоков данных. За счет поляризации сигнала можно сделать так, что два транспондера смогут использовать одну и ту же частоту.

Первые спутники связи имели один широкий луч. Современные имеют несколько более узких лучей, пятно которых охватывает несколько сот километров поверхности Земли.

Спутниковые системы связи имеют существенные отличия от наземных систем точка-точка. Несмотря на то что сигнал распространяется со скоростью света, из-за больших расстояний задержка при передаче велика - 250-300 мсек., против 3-5 мксек./км на коаксиале, оптоволокне и т.д.

Спутниковые системы принципиально вещательного типа. Для некоторых приложений это очень важно. Стоимость передачи не зависит, скольким получателям сообщение предназначено. Однако проблема безопасности передаваемой информации здесь требует особого внимания - все слышат все, что передается. Решение этой проблемы - только шифрование.

Стоимость передачи не зависит от расстояния.

Такой способ передачи имеет очень низкий коэффициент ошибок при передаче.

Система спутниковой связи и передачи данных ASTROLINK.

В системе предполагается использовать девять геостационарных ИСЗ, расположенных в точках стояния 97°, 21,5° з.д., 38°, 130°, 170,25° в.д., которые формируют практически глобальную зону обслуживания. Система ASTROLINK зарегистрирована в октябре 1995 г. В мае 1997 г. федеральная комиссия США гарантировала компании Lockheed Martin Telecommunications действие лицензии.

Система предусматривает цифровую телефонную связь, передачу данных и трансляцию широкополосной видеоинформации для медицинских учреждений, правительственных организаций, транспортных и туристических компаний. Предусматривается организация распространения электронных версий различных изданий, дистанционное обучение, передача медицинских томографических данных и решение многих других задач, требующих передачи больших объемов информации.

Наземный сегмент будет включать стационарные и передвижные абонентские станции с антеннами диаметром 65, 85 и 120 см. Станции предусматривают систему автоматического поддержания уровня излучаемой мощности и рассчитаны на работу в ISDN-сетях и сетях, использующих технологию ATM. Станции для крупных пользователей имеют те же возможности, но диаметр антенны увеличен до 1,2-2,4 м, и обеспечивают возможность коллективного доступа к ресурсам системы. Центральные региональные станции имеют выход в наземные телефонные сети

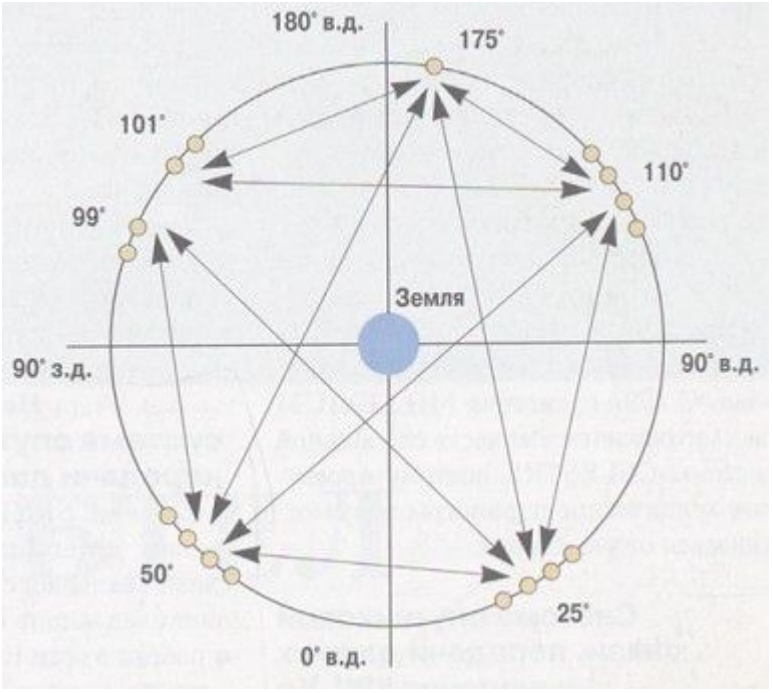
общего пользования (диаметр антенны 2,4 м). В каждой рабочей зоне может быть расположено до 24 центральных региональных станций.

Межрегиональная система спутниковой связи и передачи данных SPACEWAY.

Система SPASEWAY предназначена для организации международных видеотелефонных сетей и высокоинформативных сетей передачи данных в глобальном масштабе. Космический сегмент по проекту предусматривает 17 геостационарных ИСЗ, расположенных в точках стояния 50°, 101° з.д., 25°, 110, 175° в.д. Все ИСЗ связаны межспутниковыми линиями (рисунок 2-82). Многолучевые антенны спутников создают несколько региональных рабочих зон:

- Регион 1 - Северная Америка
- Регион 2 - Центральная Америка, Южная Америка
- Регион 3 - Африка, Средний Восток, Европа
- Регион 4 - Австралия, Океания, Дальний Восток

Рисунок 2-82. Схема размещения спутников в системе SPACEWAY



Каждый регион обслуживают четыре ИСЗ, образующие кластер в соответствующей точке геостационарной орбиты. Один ИСЗ (175° в.д.) выделен для обеспечения трафика между США и Азиатско-Тихоокеанским регионом. Рабочие зоны формируются многолучевыми антеннами. Каждый из 48 лучей поддерживает передачу цифрового потока со скоростью 92 Мбит/с. На Земле используются различные типы абонентских станций USAT с диаметром антенны 66 см и передатчиками примерно 2 Вт. Проектный срок запуска системы в эксплуатацию - 2004 год. Параллельно рассматривается концепция создания дополнительной космической группировки на основе среднеорбитальных ИСЗ при сокращении числа геостационарных ИСЗ.

Спутниковая система для видеотелефонной связи в США CYBERSTAR.

Космический сегмент системы CYBERSTAR предусматривает создание в точке стояния 110° з.д. кластера из трех геостационарных ИСЗ CYBERSTAR (дополнительный резервный ИСЗ будет находиться на Земле). Спутники должны обеспечить трансляцию сверхширокополосной информации для обеспечения многоканальной цифровой видеотелефонной связи на территории США (включая Аляску и Гавайи). Система CYBERSTAR имеет лицензию на работу в Ка-диапазоне частот. Ретрансляционная аппаратура ИСЗ обеспечивает регенерацию сигналов. Для абонентов сетей VSAT

используются скорости передачи информации от 384 Кбит/сек. до 3,088 Мбит/сек. Станции VSAT имеют диаметр антенны 0,7-1,5 м.

Билет № 18.

Спутниковые системы связи: низко орбитальные спутники (примеры).

Изначально для целей передачи данных низколетящие спутники серьезно не рассматривались. Слишком быстро они проносились над определенным местом на поверхности Земли. В 1990 компания Моторола выдвинула проект системы низколетящих спутников. Идея была очень проста: когда пятно луча одного спутника уходило из определенного места, к этому месту подлетал другой спутник, пятно которого охватывало это место. Подлетевший спутник подхватывал передачу/прием, которую вел улетающий спутник, и связь сохранялась. Компания подсчитала, что для реализации этой идеи потребуется 77 спутников на высоте 750 км. Позднее, после уточнения параметров проекта, это число сократилось до 66. Этот проект получил название Иридиум (по названию 77-го элемента в таблице Менделеева).

Основной целью этого проекта являлось обеспечение связи с наземными средствами, даже портативными, всей поверхности Земли. Этот проект вызвал ожесточенную конкуренцию со стороны других компаний. Все захотели строить низколетящие спутниковые системы. Было предложено множество других проектов, но все они похожи на Иридиум.

Iridium.

Iridium – первая в мире система глобальной персональной спутниковой телефонной связи и пейджинга.

Схема проекта. Вдоль меридиана на расстоянии 32 градуса располагаются 11 спутников, летящих на высоте 750 км. Таких ожерелий 6, они охватывают всю Землю. Каждый спутник имеет 48 пятен, так что 1628 пятен (сот) покрывают Землю (рисунок 2-74 (b)). Каждая сота имеет 174 дуплексных канала на частоте обычного сотового радиотелефона. Таким образом, во всем мире поддерживаются 283 272 канала. Некоторые из них используются для пейджинга и для навигации и не требуют большой пропускной способности.

Прием и передача идут на частоте 1,6 ГГц, что позволяет использовать устройства, работающие от батарей. Если сообщение, принятое одним спутников, адресовано в область, покрываемую другим, то оно будет передано от одного спутника другому. На время оставим рассмотрение этого проекта. Мы еще к нему вернемся.

В России Iridium использовался около двух лет, после чего было объявлено о банкротстве консорциума. Коммерческую эксплуатацию системы в нашей стране осуществляла операторская компания ОАО «Иридиум-Евразия». По данным этой компании, в 1999 г. сеть Iridium насчитывала около 30 тыс. абонентов в мире, из них 1% - в России. В Iridium (как и системах сотовой связи стандарта GSM) формат TDMA-кадра состоит из восьми временных слотов. Но, в отличие от GSM, кадры для радиолиний «вверх» и «вниз», хотя и идентичные по структуре, различаются по скорости передачи (180 и 400 кбит/сек. соответственно). Суммарная длительность кадра равна 90 мсек. Ширина полосы частот каждого канала составляет 126 кГц (линия «вверх») и 280 кГц (линия «вниз»). На одной частотной несущей в каждый момент может передаваться 29 (4 служебных) и приниматься 64 (9 служебных) каналов.

Компания Motorola изготовила портативный спутниковый терминал со сменными картриджами, которые обеспечивают его использование в качестве сотового аппарата (для каждого стандарта: GSM, AMPS, TDMA, CDMA – свой картридж). Средняя мощность абонентского передатчика 0,57 Вт, чувствительность приемника – 118,4 дБ. Штатная батарея рассчитана на непрерывную работу до 2 ч в режиме разговора и до 16 ч в режиме ожидания.

Каждый спутниковый телефон имеет свой модуль идентификации абонента (SIM-карту), содержащий единый в системе номер телефона, данные об абоненте, блокирующие коды и т.д.

Кроме телефонных трубок, Motorola выпустила автомобильный и офисный терминалы, которые обеспечивали весь спектр услуг спутниковой телефонной связи. Автомобильный имел выдвижную антенну, а офисный – выносную. Последний представлял собой многофункциональный телефонный адаптер массой до 1,5 кг и габаритами 240x200x64 мм.

Inmarsat.

Международная организация спутниковой связи, которая сегодня насчитывает 86 стран-участниц, обеспечивает работу (по состоянию на 1999 г.) более чем 143 тыс. земных пользовательских станций спутниковой связи Inmarsat, размещенных прежде всего на морских судах и других транспортных средствах. Система Inmarsat базируется на среднеорбитальной группировке спутников. За 20 лет своего существования она, пожалуй, единственная из всех прошла все этапы развития и внедрения подвижной спутниковой связи, «опробовала» абонентское оборудование практически любого типа (начиная от первых судовых станций, весивших до 200 кг, до современных портативных терминалов не тяжелее 15 кг). Следует отметить, что за прошедшие годы тарифы и цены на оборудование снижались неоднократно. Сейчас портативный мобильный терминал стоит примерно 2 тыс. долл., а одна минута разговора через него – не более 3 долл.

ГУП «Морсвязьспутник», представляющее в Inmarsat интересы России, является не только административным органом системы Inmarsat, но и эксклюзивным провайдером ее услуг в нашей стране. Услуги (в том числе телефонная и факсимильная связь со скоростью 2,4-9,6 кбит/сек. и высокоскоростная передача данных в зональном луче со скоростью 56/64 кбит/сек.) предоставляются через береговые станции Inmarsat с единым для всех зон действия (океанских регионов) российским кодом доступа 015.

В настоящее время ГП «Морсвязьспутник» готово приступить к реализации в рамках проекта Inmarsat-M4 новой услуги, которая позволит интегрировать международные и корпоративные информационные сети с глобальной спутниковой связью, впервые обеспечив высокоскоростную передачу данных (64 кбит/сек.) практически в любой точке земного шара.

Globalstar.

Другая низкоорбитальная система глобальной персональной спутниковой связи, которая начала действовать на территории России – это Globalstar. Территорию России она охватывает почти полностью – до 70° с.ш. включительно (это около 700 км севернее полярного круга). Официальной датой старта работы сети Globalstar было объявлено 11 октября 1999 г., когда в Женеве на всемирной выставке «Телеком-99» было сделано более 30 тыс. пробных звонков с использованием средств системы через станции сопряжения в Италии (оператор Elsacom) и Франции (оператор TE.SA.M). Опытная эксплуатация российского сегмента началась в марте 2000 г.

Эксплуатацией и предоставлением услуг Globalstar в нашей стране занимается компания «ГлобалТел». В настоящее время процесс формирования орбитальной группировки практически завершен: 48 спутников выведено на круговые орбиты высотой 1414 км. Последний старт ракеты-носителя Delta в начале февраля 2000 г., когда на орбиту были запущены четыре резервных КА, закончил этап формирования этой группировки.

Спутники Globalstar имеют простые ретрансляторы без обработки сигналов, что обуславливает их малый вес (450 кг), высокую надежность, длительный срок жизни (7,5 года), а также более низкую стоимость по сравнению с другими проектами аналогичного назначения. Межспутниковые линии связи не предусмотрены. Принятый бортовым ретранслятором поток транслируется на Землю в следующем диапазоне частот: 6875,95 – 7052,9 МГц (передача) и 5091 – 5250 МГц (прием).

Проект Globalstar будет иметь свыше 50 станций сопряжения по всему миру, из которых уже построена половина. Центр управления связью и полетами располагается на территории США, а Центры управления национальных поставщиков услуг – на территории страны сервис-провайдера. В России есть три станции сопряжения: в Москве (Павловский Посад), Новосибирске и Хабаровске. Каждая станция сопряжения связана с сетью общего пользования РФ и интегрирована с действующими стационарными и сотовыми сетями. На территории РФ сеть Globalstar имеет выделенный код негеографической зоны DEF («город ГлобалТел» – код 954). Российским станциям сопряжения присвоен статус междугородных станций национальной сети, и они подключены к телефонной сети через узлы автоматической коммутации и МКЦ.

Технологической основой Globalstar является стандарт CDMA, что обеспечивает (как показала практика) высокое качество речи и лучшую защищенность от прослушивания по сравнению с другими системами. Речевой кодек с линейным предсказанием и переменной скоростью от 1,2 до 9,6 кбит/с обеспечивает среднюю скорость передачи (с учетом шумоподавления) 2,4 кбит/с. На станциях сопряжения Globalstar применяются декодеры с эхоподавителями.

Использование технологии CDMA в сочетании с непрерывным охватом каждого региона несколькими спутниками позволяет осуществить плавную эстафетную передачу сигнала со спутника на спутник и сводит к минимуму потери из-за экранирования сигналов городскими строениями и рельефом местности.

Абонентское оборудование системы Globalstar представлено многорежимными «трубками» и стационарным телефонным аппаратом.

ICO.

Международная система спутниковой связи ICO построена на основе средневысотных спутников, а ее название происходит от английского сокращения ICO – Intermediate Circular Orbit. Разработку системы осуществляет компания ICO Global Communications – международная организация, которая выделилась из Inmarsat в январе 1995 г. В нее входит более 50 компаний-инвесторов из 46 стран, в том числе ряд национальных операторов сотовой связи. Штаб-квартира организации расположена в Лондоне. Сеть ICO (см. «Сети», 1998, № 2, с. 66) должна стать одним из первых реальных поставщиков услуг персональной связи в диапазонах частот 1980–2100 и 2170–2200 МГц. Глобальный охват обеспечит орбитальная группировка из 10 спутников на высоте 10390 км. Максимальное время пребывания КА в зоне радиовидимости - 6 ч. Предусмотрены также два резервных КА.

К настоящему времени несколько КА уже запущены с помощью комплекса «Морской старт» в течение 2000-2002 гг. Наземная структура строится на базе сети ICONET (ICO network), которая объединяет 12 спутниковых узлов доступа SAM (Satellite Access Mode), размещенных в разных странах мира. Большинство из них уже смонтировано и готово к вводу в эксплуатацию. Сегмент управления системой состоит из двух центров управления полетами и двух центров управления сетью, размещенных в Лондоне и Токио. Российские узлы SAN создаются в центральной части страны и в районе Новосибирска. В качестве базового терминала в системе ICO используют мобильный двухрежимный терминал, совмещенный с сотовым телефоном со встроенным ЗУ для хранения данных и внешним портом.

Низкоорбитальная система спутниковой связи и передачи данных SKYBRIDGE.

Система SKYBRIDGE предусматривает интеграцию наземных сетей связи (включая сотовые) и предоставляет следующие виды услуг:

- Работа в сети Интернет в режимах обмена и доступа к электронным базам данных
- Оплата покупок, рекламируемых в телевизионных программах
- Дистанционное обучение, проведение видеоконференций, пересылка файлов, электронная почта

- Видеотелефонная связь
- Развлекательные телевизионные программы по заказу абонента, интерактивные компьютерные игры
- Передача информации медицинской диагностики, дистанционные медицинские консультации

Для регионов, в которых средства наземных сетей связи развиты слабо, система SKYBRIDGE может стать основой их создания и наращивания на современном уровне.

Запуск ИСЗ начался в 2001 г. Предусматривается практически глобальная зона обслуживания, ограниченная 68° ю.ш. и 68° с.ш. Космический сегмент состоит из 64 спутников, расположенных на круговых орбитах с высотой 1475 км и наклоном 55°, которые разбиты на две подгруппы. В каждой подгруппе по 32 спутника, расположенных по четыре ИСЗ на каждой из восьми орбитальных плоскостей. Столь сложное построение орбитальной группировки связано с оптимизацией системы SKYBRIDGE по критерию совместимости работы с геостационарными системами в диапазоне частот 10–18 ГГц и условием наилучшего обслуживания малонаселенных регионов Земли. В системе применяется многостанционный доступ CDMA/TDMA. Канальная скорость передачи данных на линии ИСЗ - Земля составляет 41,5 Мбит/с, а Земля - ИСЗ – 5,2 Мбит/с. Рабочая полоса частот передающего канала 22,6 МГц, (на линии ИСЗ - Земля) и 2,93 МГц (на линии Земля - ИСЗ). Энергетические параметры ИСЗ и наземных станции обеспечивают достижение вероятности ошибочного приема не более 10-6 на бит информации.

Абонентские станции имеют возможность вызова через центральную фидерную станцию, в зоне которой они находятся. Центральные фидерные станции обеспечивают маршрутизацию сигналов и стыковку с наземными сетями общего пользования. Каждая такая станция имеет зону действия диаметром около 350 км. Дополнительно между центральными станциями образованы высокоинформативные каналы связи, которые не имеют выхода в наземные сети общего пользования. При организации наземной сети используется технология ATM, которая обеспечивает быстрое и независимое расширение абонентской сети.

Система спутниковой связи и передачи данных TELEDESIC.

Система спутниковой связи TELEDESIC принципиально отличается от других систем спутниковой связи как по назначению, так и по предполагаемым техническим решениям. В первую очередь, она обеспечивает не только фиксированных, но и подвижных абонентов высококачественной телефонной связью, а также другие службы, нуждающиеся в обмене широкополосной информацией в глобальном масштабе. По своей потенциальной пропускной способности система сравнима с волоконнооптической и рассчитана на их совместную эксплуатацию в магистральных линиях связи. Основные идейные создатели системы - фирма McCaw Cellular Communications, специализирующаяся на проводных сетях связи, и фирма Microsoft. Проектные работы начаты в 1990 г. Первоначально предполагалось, что с 2001 г. начнется коммерческая эксплуатация системы, однако она была перенесена на 2003 г. (запуск первого экспериментального ИСЗ произведен 27 февраля 1998 г.).

Космический сегмент системы образован группировкой из 288 ИСЗ, расположенных на круговых орбитах высотой 1375 км в 12 плоскостях с наклонением 98,142° - 98,182° (солнечно-синхронные орбиты). На каждой орбите расположено 24 действующих ИСЗ. Пространственное разнесение орбит в плоскости экватора - примерно 18°.

Для обеспечения глобальной зоны обслуживания в системе TELEDESIC предусматриваются межспутниковые радиолинии. Общая зона обслуживания разбита на 20 000 локальных зон размером 100x100 км. Каждая локальная зона включает девять парциальных зон (ячеек) размером 53,3x53,3 км. Локальные зоны образуют «ленты», параллельные экватору (250 локальных зон вдоль экватора с уменьшением их числа к полюсам). Каждый из действующих ИСЗ создает рабочую область, включающую 64 локальные зоны (диаметр 1400 км, 576 ячеек).

При движении ИСЗ вдоль орбиты изменяется пространственная ориентация 64 лучей антенной системы, обеспечивая стационарность расположения ячеек. За каждой ячейкой закреплен определенный заранее ресурс пропускной способности ИСЗ. В результате можно достаточно точно описать границы обслуживаемых территорий, достоверно учесть плотность распределения потребителей и, соответственно, рационально использовать пропускную способность каждого ИСЗ. Кроме того, проще избежать интерференции сигналов. Межспутниковые радиолинии работают в диапазоне частот 60 ГГц, что обеспечивает помехозащищенность системы. Максимальная дальность межспутниковой радиолинии - 2586 км. Пропускная способность - 1,531 Гбит/сек.

Наземный сегмент состоит из различных терминалов. Для фиксированных абонентов предусматриваются терминалы с антеннами диаметром 0,16-1,8 м, для мобильных - 0,08 м. Мощность передающих устройств 0,01-4,7 Вт. Скорость передачи в зависимости от типа терминала и его комплектации составляет 16 кбит/сек. - 2,048 Мбит/сек. Для организации высокоскоростных линий связи используются терминалы с антеннами диаметром от 0,28 до 1,6 м при мощности передатчиков от 1 до 49 Вт в зависимости от требуемой скорости потока от 155,5 Мбит/сек. до 1,24416 Гбит/сек.

Каждый ИСЗ может поддерживать работу 16 высокоскоростных терминалов, находящихся в его рабочей зоне. Внутри отдельной ячейки предусмотрена возможность организации сетей с пропускной способностью 1400 каналов по 16 кбит/сек. или 15 каналов по 1,544 Мбит/сек. (возможна эквивалентная комбинация каналов). Между центральными станциями системы и специальными государственными пользователями предусматривается возможность организации сверхскоростных каналов от 155 Мбит/сек. до 2 Гбит/сек.

Билет № 19.

Спутниковые системы связи: VSAT спутниковые системы (примеры).

Сегодня VSAT-сети - наиболее динамично развивающаяся категория СЗ. Если в конце 1999 г. в мире было установлено более 300 тыс. приемо-передающих терминалов VSAT, то к концу 2000 г. их уже стало около 500 тыс. Аналитики продолжают утверждать, что рынок VSAT еще далек от насыщения даже в развитых странах, таких как США, Великобритания и Япония.

Современные глобальные корпоративные сети чаще всего базируются на технологии VSAT, т. е. на использовании малогабаритных спутниковых терминалов и антенн диаметром от 1,0 до 2,5 м.

Этот вид сетей широко распространен во многих странах, но особенно актуальны они в России, где наземная инфраструктура связи на значительной части территории не развита. Оптимальным решением для труднодоступных районов считается сочетание магистральных каналов наземной связи и выделенных систем С2. При этом наиболее рентабельными системы С2 становятся там, где развертывание наземных сетей экономически нецелесообразно или просто невозможно.

Выделенные сети на базе VSAT-терминалов способны предоставить своим удаленным пользователям широкий спектр услуг, включая высококачественную телефонную и факсимильную связь, передачу данных с различной скоростью, организацию видеоконференций и распределение телепрограмм.

VSAT-сети телефонной и факсимильной связи могут иметь любую топологию — от простейшей двухточечной до полнодоступной схемы «каждый-с-каждым». Выделение спутникового канала может быть организовано по-разному: для постоянного использования или по требованию.

При создании сетей корпоративной связи (то есть СПД предприятия) в сельской местности или при подключении удаленных станций к существующим сетям, в том числе к коммутируемой сети общего пользования (например, телефонной сети), данный вид услуги является приоритетным.

Современное VSAT-оборудование обеспечивает возможность подключения к наземным сетям ISDN. Типовая скорость передачи данных при таком соединении (один интерфейс BRI) колеблется от 128 кбит/с до 160 кбит/с. Использование современных алгоритмов сжатия данных позволяет «упаковать» речевой канал в полосу пропускания 6,4 или 4,8 кбит/с, благодаря чему пропускная способность спутникового канала при передаче речи повышается в 10-12 раз.

VSAT-терминалы поддерживают практически все типовые сетевые интерфейсы: RS232, RS449/422, Ethernet (IEEE 802.3), Token Ring (IEEE 802.5) (о стандартах IEEE 802.3, 802.5 подробно см. главу 3), а потому могут использоваться для объединения локальных сетей на базе наиболее популярных протоколов IP, IPX, Net-BIOS, которые мы уже упоминали во введении. Кроме того, применение многопротокольной среды и технологии Frame Relay (об этой технологии мы упоминали во Введении, подробнее она рассмотрена в главе 3) позволяет создавать сети с гибкой сменой скорости и качества услуг передачи. Например, скорость передачи в таких сетях может меняться от 64 кбит/с до 8,448 Мбит/с. Основными потребителями таких услуг высокоскоростной передачи данных и мультимедиа являются банки и страховые компании, средства массовой информации, государственные учреждения.

Технология VSAT допускает также создание корпоративных многоцелевых сетей с коммутацией пакетов с большим числом удаленных станций. Скорость передачи в таких сетях обычно не превышает 64 кбит/сек., а передача данных осуществляется с использованием стандартных протоколов X.25, X.3/X.28, LAP-B, HDLC, SNA/SDLC. Такие сети с множеством узлов характеризуются асимметричным трафиком с лавинообразной или непредсказуемой нагрузкой. Однако VSAT-технология позволяет организовать постоянный или дополнительный канал «по требованию» и обеспечить приоритезацию трафика. В качестве примера можно привести сети бензозаправочных станций с проверкой кредитных карточек в режиме реального времени, сети контроля за банкоматами, сети сбора и обработки телеметрической и метеорологической информации и т.п.

Стремительный рост популярности сети Интернет и бурное развитие сетей VSAT дает основание говорить о слиянии в перспективе этих технологий в одну. Сегодня через спутник можно напрямую подключить сервер корпоративной сети к шлюзам Интернета в США, Европе, Австралии и получить полный пакет услуг Сети по выбранному каналу - от 19,2 кбит/сек. до 8,448 Мбит/сек. Доступ в Интернет может быть организован как по асимметричной, так и по симметричной схеме. Интерфейс передачи данных - RS232, Ethernet (IEEE 802.3) или Token Ring (IEEE 802.5).

В отличие от сетей С2, использующих глобальный луч КА, в VSAT-сетях вся зона обслуживания делится на узкие парциальные зоны, каждая из которых образована одним узким лучом. Как уже отмечалось, сеть С2 обслуживает территории, где инфраструктура систем общего пользования развита довольно слабо (или полностью отсутствует) и поэтому нагрузка на сеть С2 достаточно высока. Для снижения общего уровня нагрузки в сети VSAT, наряду с абонентскими каналами с низким уровнем трафика, организуют несколько направлений связи с большим количеством групповых трактов, реализуемых на закрепленных спутниковых каналах PAMA (Permanently Assignment Multiple Access) различной пропускной способности.

В сетях VSAT разных технологий используются разные базовые технологии доступа: для схемы «точка-точка» - один канал на несущую - SCPC (Single Channel Per Carrier), для схемы «каждый-с-каждым» - множественный доступ по требованию - DAMA (Demand Assignment Multiple Access) и постоянный множественный доступ PAMA, для «звезды» - множественный доступ с временным разделением каналов (TDMA).

SCPC-технология позволяет обеспечить прямую дуплексную связь между двумя удаленными пунктами и лучше всего подходит для создания небольших корпоративных сетей с малым числом наземных станций ЗС (15-20), обычно расположенных в труднодоступных регионах. Сеть отличается сравнительно недорогим оборудованием, однако через нее невозможно организовать взаимодействие локальных сетей из-за большого времени задержки. Мы подробно рассмотрим вопрос

взаимодействия локальных сетей при рассмотрении канального уровня. Еще один существенный недостаток технологии SCPC - неэффективное использование спутникового ресурса.

Технология доступа с предоставлением каналов по требованию (DAMA) обеспечивает прямые соединения между любыми точками сети. Такая полнодоступная структура позволяет устанавливать связь с минимальной задержкой без повторного приема информации на центральной станции - так называемую связь за один скачок. Данная технология доступа оптимальна при создании телефонных сетей в удаленных и труднодоступных районах, где доля трафика на направлениях между абонентами выше, чем в направлении центральной станции. Используя DAMA, можно организовать передачу данных и взаимодействие с локальными сетями, но эффективность такого взаимодействия не очень высока. Сети на основе технологии DAMA обладают повышенной «живучестью» и гибкостью, однако стоимость абонентских VSAT-терминалов для них значительно выше, чем для сетей на базе SCPC.

Сети с топологией «звезда», основанные на технологии TDMA, применяются наиболее часто. Их сфера - многоточечные сети передачи данных с большим числом удаленных терминалов (не имеющих взаимного трафика) и центральной станцией (телепортом). Типичный пример - сеть по продаже авиабилетов. Данное техническое решение для VSAT-сети позволяет использовать на центральной станции (hub) антенны большого диаметра и мощные передатчики, а для абонентских периферийных терминалов - относительно дешевые VSAT-станции с малыми антеннами без потерь скорости передачи (32-2048 кбит/с).

В сетях VSAT с централизованным управлением, создаваемых крупными операторами связи, часто применяются так называемые комбинированные сети на основе топологии «звезда», в которых существуют собственные сети типа «звезда» или «каждый-с-каждым», организованные на базе крупных периферийных станций. Рассмотрим теперь несколько примеров СПД-систем на VSAT-сетях.

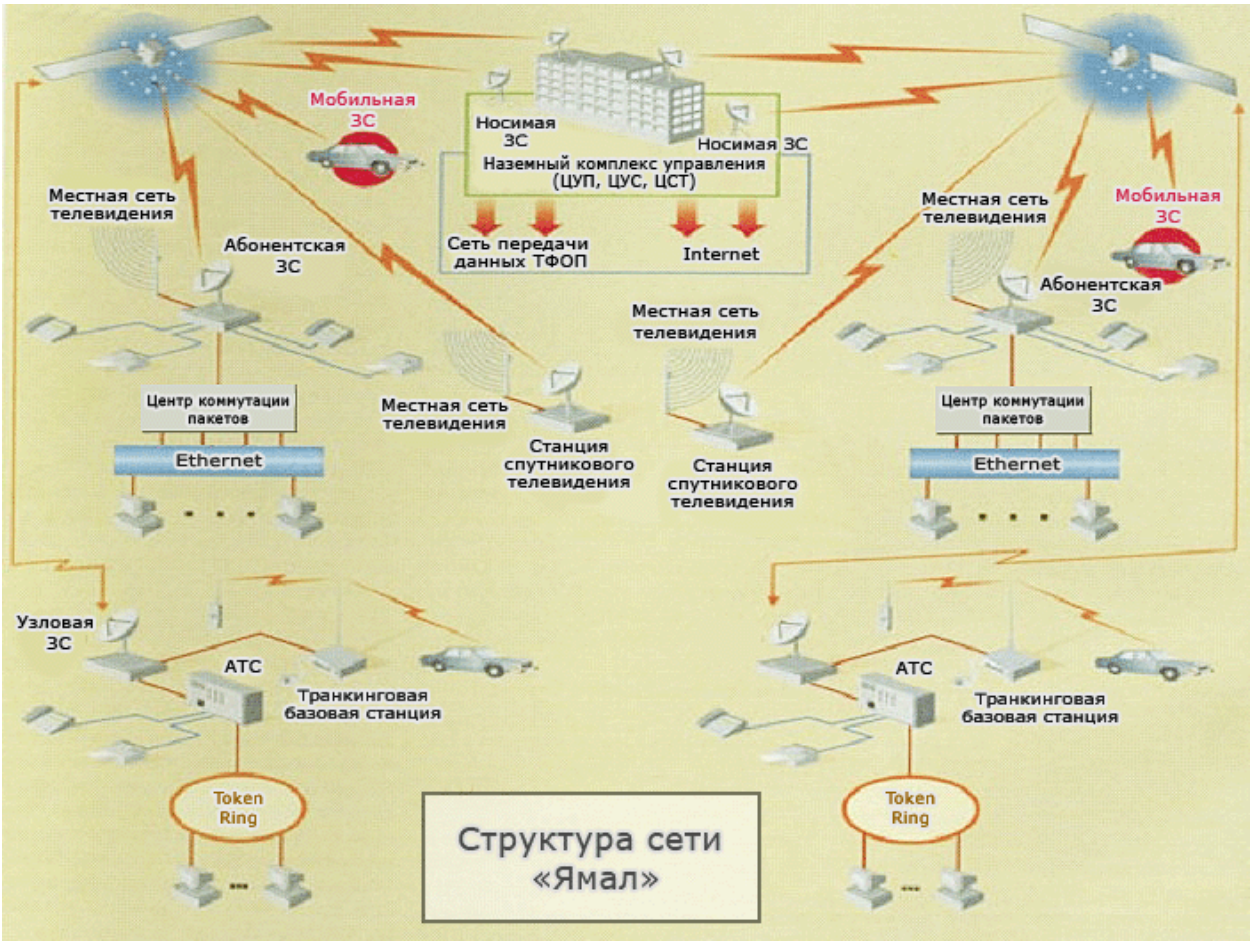
Система С2 «Ямал» РАО «Газпром».

Сегодня РАО «Газпром» владеет сетью газопроводов протяженностью более 140 тыс. км, а значительная часть этих километров расположена в местах полного отсутствия проводной наземной связи. Для обеспечения российских предприятий газовой промышленности современными видами связи была создана спутниковая система связи "Ямал". С этой задачей системы С2 РАО «Газпром» сопряжена еще одна - контроль за состоянием потенциально опасных объектов.

С самого начала основные усилия разработчиков были направлены на создание собственного космического сегмента и развертывание на его базе корпоративных сетей связи для отделений «Газпрома». Архитектура сети из-за большой рассредоточенности объектов ориентирована главным образом на технологию DAMA (для всех объектов сети) и PAMA (на отдельных направлениях, характеризующихся более высоким трафиком). Земной сегмент «Ямала» включает три типа ЗС (рисунок 2-79), которые имеют возможность наращивания числа каналов (без отключения рабочих) и оснащены автоматическим управлением:

- узловые станции сопряжения с пропускной способностью до 8448 кбит/сек., мощностью передатчиков 125-700 Вт и диаметром антенн 4,5-7 м
- абонентские станции VSAT, обеспечивающие скорость передачи до 2048 кбит/сек. и имеющие передатчики мощностью 2-40 Вт и антенны диаметром 2,5-3,7 м
- малогабаритные возимые, стационарные и носимые станции с передатчиками мощностью до 5 Вт и антеннами диаметром 0,6-1,5 м, позволяющие передавать данные со скоростью до 64 кбит/сек.

Рисунок 2-79. Взаимодействие спутникового и наземного сегментов в сети «Ямал»



Общее число наземных станций — около 60. Ретранслятор КА «Ямал» обеспечивает обмен данными с ЗС, расположенными в девяти зонах, с помощью девяти лучей. Переключение стволов с одного луча на другой производится по командам с Земли. Формирование лучей выполняется на этапе создания КА. Адаптация к возможным изменениям трафика производится в процессе эксплуатации путем перекоммутации части стволов по лучам.

В системе С2 «Ямал» реализована транспортная среда, объединяющая в единое информационное пространство существующие выделенные сети передачи данных предприятий газовой промышленности и сеть аварийной радиосвязи. Сеть передачи данных АСУ связывает вычислительные комплексы РАО «Газпром» в единую сеть («центр-регион» и «регион-регион»), а также обеспечивает связь регионов с сетями передач и данных (СПД) России и международными сетями (см. рисунок 2-79). Используемое в системе С2 «Ямал» каналообразующее оборудование позволяет организовывать как синхронные (скорость передачи 9,6 – 2048 кбит/сек.), так и асинхронные (0,075–19,2 кбит/сек.) цифровые каналы.

Развертывание СПД в полном масштабе позволит создать транспортную среду для разрабатываемых в АО «Газпром» систем экологического мониторинга, управления буровыми работами, контроля и управления электроснабжением, сбора геофизической информации.

Система С2 «Банкир».

Несоответствие инфраструктуры первичных наземных сетей районного уровня высоким требованиям к надежности и достоверности передачи информации по каналам Frame Relay не позволяет строить банковскую сеть России с использованием только магистральных волоконнооптических каналов и оборудования существующих наземных сетей. Поэтому было принято решение о создании банковской сети ЦБ РФ – системы С2 «Банкир» с цифровыми каналами, обеспечивающими скорость передачи от 64 до 512 кбит/сек. Многоуровневая архитектура сети

объединяет три выделенные системы С2, которые строятся по единому принципу, но имеют различную техническую реализацию:

- «Банкир-1» развертывается в Северо-Западном, Волго-Вятском, Поволжском, Сибирском регионах.
- «Банкир-2» охватит Центральный, Центрально-Черноземный, Северо-Кавказский и Уральский регионы.
- «Банкир-3» будет работать на Дальнем Востоке.

Основными пользователями ССС ЦБ РФ являются Главный вычислительный центр (ГВЦ) Банка РФ, 78 Главных управлений ЦБ РФ и Национальных Банков республик и 1096 расчетно-кассовых центров. Общая емкость сети — около 1200 ЗС.

Первоначально предполагалось организовать связь через два КА отечественного производства «Купон», первый из которых был выведен на орбиту в конце 1997 г. Однако отказ первого КА после четырех месяцев работы (сбой в системе частотообразования) радикально изменил ситуацию. ЦБ РФ решил приостановить работы по разработке собственных КА и арендовать частотный ресурс на других спутниках (таблица 2-81). В настоящее время заключено соглашение с международной организацией Intelsat на предоставление частотной емкости КА lntelsat-704 (три ствола) для сетей «Банкир-1» и «Банкир-2», а частотный ресурс для сети «Банкир-3» арендуется на отечественных КА «Горизонт-33» и «Ямал-100».

Система С2 «Банкир» предоставляет следующие услуги: передача данных по протоколам Frame Relay и X.25; телефонная связь через учрежденческие АТС; проведение видеоконференций. Передача данных (электронные платежи) осуществляется через спутник: на магистральном уровне (между ЦЗС и УЗС) – каналы FDMA/SCPC/PAMA (скорость 256 кбит/сек.), на региональном уровне (между УЗС и АЗС) – каналы FDMA/SCPC/PAMA (скорость 64 кбит/сек.) и FDMA/TDM, FDMA/TDMA (скорость 128 и 64 кбит/сек.). Для передачи речи используется спутниковый канал FDMA/SCPC/DAMA (скорость 16 кбит/с). Сегодня состояние сети ЦБ таково:

- Развернуто 47 региональных подсистем, установлено около 600 ЗС для ССС «Банкир-1», «Банкир-2» и «Банкир-3» (50% от общего числа).
- Находится в эксплуатации 27 региональных подсистем, около 300 ЗС подсетей «Банкир-1» и «Банкир-3» (50% от числа развернутых и 25% от общего числа). Полное развертывание сетей «Банкир-1» и «Банкир-3» завершилось в 2001 г., а сети «Банкир-2» — в конце 2001 г.

Билет № 20.

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

Раздел 3.1. Проблемы, решаемые на уровне канала данных

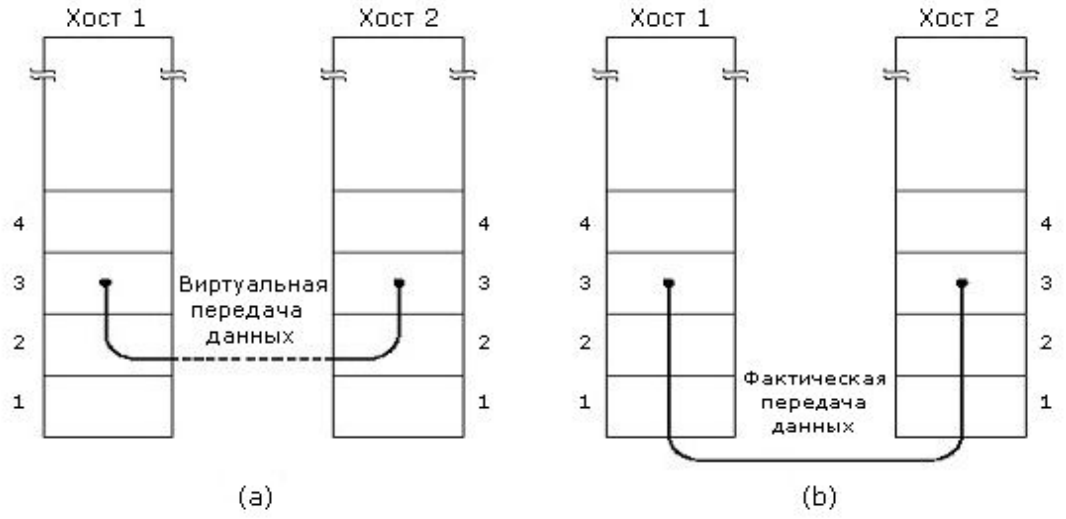
На уровне канала данных решается ряд проблем, присущих только этому уровню: реализация сервиса для сетевого уровня, объединение битов, поступающих с физического уровня в кадры, обработка ошибок передачи, управление потоком кадров и другие.

3.1.1. Сервис, предоставляемый сетевому уровню

Основная задача канального уровня - обеспечить сервис сетевому уровню по передаче и приему данных. Назначение этого сервиса - передать данные от процесса на сетевом уровне одной машины процессу на сетевой уровень другой машины. Как это происходит, показано

на рисунке 3-1 (a). Фактически процесс передачи сложнее, что проиллюстрировано на рисунке 3-1 (b). Однако для простоты изложения мы будем придерживаться первой схемы.

Рисунок 3-1. Передача данных - виртуальная (a), фактическая (b)



Канальный уровень может обеспечивать различный сервис. Хотя этот сервис может варьироваться от системы к системе, есть три общих видов сервиса:

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

Сервис без уведомления и без соединения не предполагает, что до начала передачи должно быть установлено соединение, которое после передачи должно быть разорвано, что факт приема переданного кадра должен подтверждаться специальным сообщением. Если в результате помех на физическом уровне кадр будет потерян, то никаких попыток его восстановить на канальном уровне произведено не будет. Этот класс сервиса используется там, где физический уровень обеспечивает настолько высокую надежность при передаче, что потери кадров происходят редко и восстановление при потере кадров можно переложить на верхние уровни. Этот вид сервиса также применяют при передаче данных в реальном времени там, где лучше потерять часть данных, чем увеличить задержку при их доставке. Например, передача речи, видео изображения. Большинство ЛВС использует этот вид сервиса на канальном уровне.

Следующий вид сервиса – сервис с уведомлением без соединения. В этом виде сервиса получение каждого посланного кадра должно быть подтверждено. Если подтверждения не пришло в течение определенного промежутка времени, то считают, что кадр не принят и должен быть послан опять. Этот вид сервиса используется в ненадежной физической среде передачи, например, беспроводной.

Можно было бы, конечно, подтверждать не кадры, а все сообщение целиком на сетевом уровне. Однако это было бы невыгодно для больших сообщений. Например, если при передаче этих данных разрушалось бы 10-20% кадров, то повторная передача таких

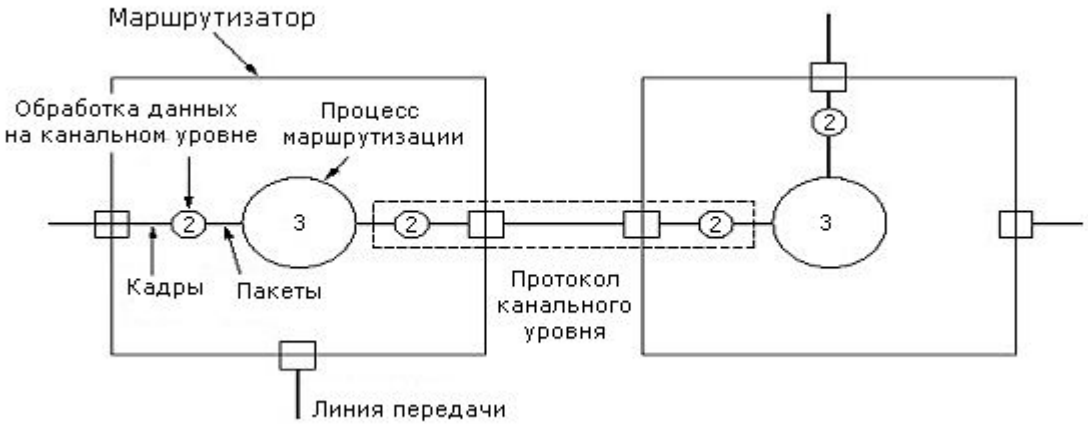
сообщений заняла бы больше времени, чем передача их отдельных фрагментов. Поэтому подтверждение на канальном уровне следует рассматривать как минимизацию затрат на передачу данных, а не необходимость.

Наиболее сложный класс сервиса на канальном уровне - сервис с соединением и уведомлением. Этот класс сервиса предполагает, что до начала передачи между машинами устанавливают соединение и данные передают по этому соединению. Каждый передаваемый кадр нумеруется, и канальный уровень гарантирует, что он будет обязательно получен, причем только один раз, а также что все кадры будут получены в надлежащей последовательности. При сервисе без соединения этого гарантировать нельзя, потому что потеря подтверждения получения кадра приведет к его пересылке, что, в свою очередь, приведет к появлению нескольких идентичных кадров. Использование сервиса с соединением особенно полезно в том случае, когда канал образует СПД. Как мы уже видели, в СПД может быть достаточно сложная организация канала, при которой может происходить коммутация потоков данных самыми разными способами. Здесь полезно вспомнить структуру кадра в X.25, рассмотренную в разделе 2.3. С точки зрения структуры, в X.25 есть три вида кадров: с 3-битовым полем номера кадра, 7- и 12-битовым. Таким образом, в X.25 предусмотрен сервис с соединением, причем, в зависимости от длины передаваемых данных, можно оптимизировать затраты на поддержку соединения.

При сервисе с соединением и уведомлением передача данных разбивается на три этапа. На первом этапе устанавливают соединение: на обеих машинах инициируют переменные и счетчики, отслеживающие, кадры с какими номерами были приняты, а с какими нет. На втором этапе передают нужные кадры. На третьем - соединение разрывают: переменные, счетчики, буферы и другие ресурсы, использованные для поддержки соединения, освобождают.

На рисунке 3-2 показан типичный фрагмент WAN, где два маршрутизатора соединены через телефонную линию. Когда кадр поступает на маршрутизатор, аппаратура проверяет контрольную сумму и передает кадр канальному уровню. Канальный уровень проверяет, является ли поступивший кадр ожидаемым, и если да, то передает пакет, расположенный в кадре, сетевому уровню на другой машине.

Рисунок 3-2. Размещение протокола канального уровня



3.1.2. Разбиение на кадры

Сервис, создаваемый канальным уровнем для сетевого уровня, опирается на сервис, создаваемый физическим уровнем. На физическом уровне принимают и передают потоки битов. Отправленное количество битов не обязательно должно быть равно принятому количеству битов, значение посланного бита также не обязательно должно быть равно принятому значению бита. Поэтому на канальном уровне нужны специальные действия по обнаружению и исправлению таких ошибок.

Типовой подход к решению подобных проблем - разбиение потока битов на кадры, подсчет контрольной суммы для каждого кадра и передача этой суммы вместе с кадром данных. При приеме контрольная сумма вычисляется для каждого кадра заново и сравнивается с той, что хранится в кадре. Если они различаются, то это признак ошибки передачи. На канальном уровне должны быть приняты меры к исправлению ошибки, например, сбросить плохой кадр и послать сообщение об ошибке тому, кто прислал этот кадр.

Разбиение потока битов на кадры - задача не простая. Один из способов - делать паузу между битами разных кадров. Однако в сети, где нет единого таймера, нет гарантии, что эта пауза всегда будет одинаковой, или, наоборот, не появится новая пауза там, где ее не должно было быть.

Так как методы, использующие временные задержки, не надежны, применяются другие методы. Здесь мы рассмотрим четыре основных метода:

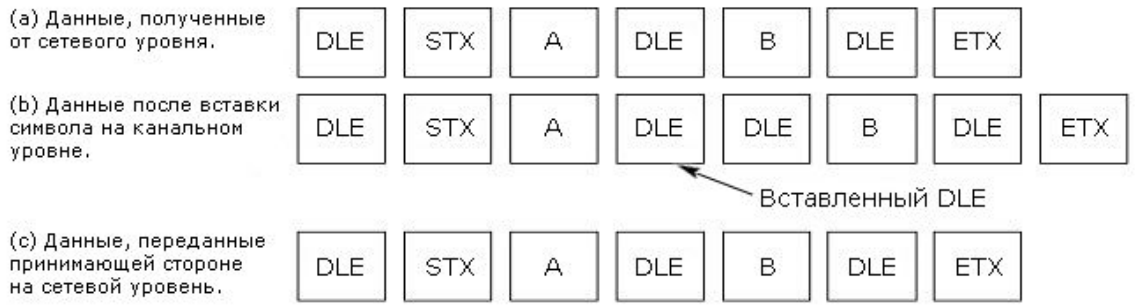
1. счетчик символов
2. вставка специальных стартовых и конечных символов
3. вставка стартовых и концевых битов
4. нарушение кодировки на физическом уровне

Первый метод (счетчик символов)показан на рисунке 3-3. В начале каждого кадра указывают, сколько символов в кадре. При приеме кадра вновь подсчитывают число принятых символов. Если число полученных символов отлично от ожидаемого числа, то этот факт воспринимают как ошибку. Однако этот метод имеет существенный недостаток: счетчик символов может быть искажен при передаче. Тогда принимающая сторона не сможет обнаружить границы кадра. Даже обнаружив несовпадение контрольных сумм, принимающая сторона не сможет сообщить передающей, какой кадр надо переслать и сколько символов пропало. Этот метод сейчас используется редко.

Рисунок 3-3. Поток символов: (a) без ошибок; (b) с одной ошибкой

Второй метод (вставка специальных стартовых и конечных символов)построен на вставке специальных символов. Обычно для этого используют последовательность символов DLE STX для начала кадра и DLE ETX для конца кадра. DLE (Data Link Escape), STX (Start TeXt), ETX (End TeXt) – это специальные символы, имеющие специальную кодировку. При этом методе, если даже была потеряна граница текущего кадра, нужно просто найти ближайшую последовательность DLE STX или DLE ETX. Однако здесь есть одна опасность: при передаче чисел или программы в объектном коде такие последовательности могут уже содержаться в передаваемых данных. Для решения этой проблемы используют прием экранирования: каждая последовательность DLE или STX просто дублируется в передаваемых данных. Поэтому, если при приеме есть два последовательных DLE, то один удаляется. Этот метод проиллюстрирован на рисунке 3-4.

Рисунок 3-4. Метод экранирования

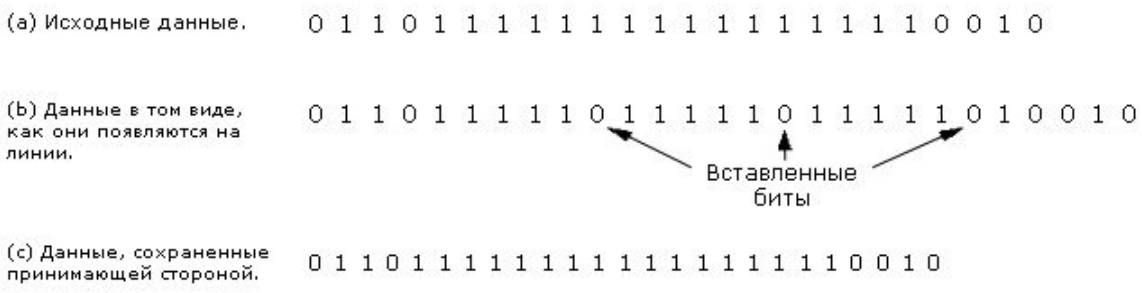


Основным недостатком только что рассмотренного метода является то, что он жестко связан с размером байта и конкретным методом кодировки символов - ASCII. По мере развития сетей эта связь становилась все более и более обременительной. Кроме этого, на стороне отправителя надо было просматривать кадр, чтобы обнаружить недопустимые последовательности.

Был предложен иной прием, позволяющий использовать любое число битов на символ и любую кодировку. (вставка стартовых и концевых битов)Его идея состоит в том, что каждый кадр начинается и заканчивается специальным флаг-байтом: 01111110. Чтобы избежать аналогичной последовательности, внутри кадра поступают следующим образом. Посылающая сторона, встретив последовательно 5 единиц, обязательно вставит 0. Принимающая сторона, приняв 5 последовательных единиц, обязательно удалит следующий за ними 0. Таким образом, если в передаваемых данных встретится конфигурация флаг-

байта, то она будет преобразована в конфигурацию 011111010. Этот метод иллюстрирует рисунок 3-5. Этот метод прозрачен для сетевого уровня так же, как и метод вставки байтов.

Рисунок 3-5. Замена шестой единицы нулем



Таким образом, кадр легко может быть распознан по флаг-байту. Если граница очередного кадра по какой-то причине была потеряна, то все что надо делать – «ловить» ближайший флаг-байт.

Последний метод - нарушение кодировки используется там, где применяется специальная кодировка битов на физическом уровне. Например, пусть для передачи одного бита используется два импульса. «1» кодируется как переход «высокое-низкое», «0» - как переход «низкое-высокое». Сочетания «низкое-низкое» или «высокое-высокое» не используются для передачи данных. Их и используют для границ кадра. Так делают в стандарте IEEE 802 для ЛВС, который мы рассмотрим позже.

На практике используют, как правило, комбинацию этих методов. Например, счетчик символов с одним из выше перечисленных. Тогда, если число символов в кадре совпадает с кодировкой границы кадра, кадр считается переданным правильно.

3.1.3. Обнаружение ошибок

Решив проблему разбиения на кадры, мы приходим к следующей проблеме: как обеспечить, чтобы кадры попадали на сетевой уровень в надлежащей последовательности? Если для отправляющей стороны все равно, в какой последовательности поступают кадры, то этой проблемы нет. Например, если нам нужен сервис без уведомления и без соединения. Однако как быть, если нам нужен сервис с уведомлением 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 же в нем сообщение об ошибке, то переданные кадры надо передать заново.

Однако возможны ситуации, когда из-за ошибок в канале кадр исчезнет целиком. В этом случае получатель не будет никак реагировать, а отправитель будет сколь угодно долго ждать подтверждения. Для решения этой проблемы на канальном уровне вводят таймеры. Когда передается очередной кадр, то одновременно устанавливается таймер на определенное время. Этого времени должно хватать на то, чтобы получатель получил кадр и отправил уведомление, а отправитель получил его.

Если отправитель не получит уведомление раньше, чем истечет время, установленное на таймере, то он будет считать, что кадр потерян и повторит его еще раз.

Однако если кадр-подтверждение был утерян, то вполне возможно, что один и тот же кадр получатель получит дважды. Как быть? Для решения этой проблемы каждому кадру присваивают порядковый номер. С помощью этого номера получатель может обнаружить дубли.

Итак, таймеры и нумерация кадров - основные средства на канальном уровне, обеспечивающие доставку каждого кадра до сетевого уровня в единственном экземпляре и в нужном порядке.

3.1.4. Управление потоком

Другая важная проблема, которую надо решать на канальном уровне - управление потоком. Вполне может случиться, что отправитель будет посылать кадры столь часто, что получатель не будет успевать их обрабатывать. Это может произойти, если, например, машина-отправитель более мощная или загружена слабее, чем машина-получатель.

Для борьбы с такими ситуациями вводят специальный механизм управления потоком. Этот механизм предполагает обратную связь между отправителем и получателем, которая позволяет им урегулировать темп передачи.

Существует много схем управления потоком, но все они в основе своей используют следующий сценарий. Прежде чем отправитель начнет передачу, он спрашивает у получателя, сколько кадров тот может принять. Получатель сообщает ему определенное число. Отправитель, после того как передаст это число кадров, должен приостановить передачу и спросить у получателя еще раз, сколько кадров тот может принять, и т.д. Позднее на примерах мы познакомимся с конкретными механизмами управления потоком.

Раздел 3.3. Простейшие протоколы канала данных

Рассмотрение протоколов уровня канала данных мы начнем с нескольких предположений. Будем предполагать, что физический уровень, уровень канала данных, сетевой уровень – реализованы в виде независимых процессов, взаимодействующих с помощью передачи сообщений. В некоторых случаях физический уровень и уровень канала данных могут выполняться на некотором вспомогательном процессоре ввода-вывода, внешнем по отношению к основному процессору; в некоторых случаях все процессы могут выполняться на основном процессоре. Возможны также разные реализации: физический и канальный уровни могут быть реализованы в виде процедур, вызываемых сетевым уровнем, и т.д. Однако мы будем предполагать, что все три уровня представлены как независимые процессы.

Также мы предположим, что есть две машины: A и B. У машины A есть бесконечно длинный набор данных, который надо передать машине B с помощью надежного сервиса, ориентированного на соединение. Передача всегда происходит от A к B, хотя позднее мы допустим одновременную передачу от B к A. Также будем предполагать, что если канальный уровень на машине A запрашивает данные для передачи от сетевого уровня, то они всегда есть и нет задержки на их подготовку.

Канальный уровень рассматривает данные, которые он получает от сетевого, как неструктурированные, несмотря на то, что там есть хотя бы заголовок сетевого уровня. Все эти данные должны быть переданы равнозначному сетевому уровню. Когда канальный уровень получает пакет, он погружает его в кадр, добавляя заголовок и концевик. Этот кадр затем передается по физическому уровню. Будем предполагать, что есть две библиотечные процедуры:

from_physical_layer - для получения кадра с физического уровня, и to_physical_layer - для передачи кадра на физический уровень. Предполагаем, что вычисление и добавление контрольных сумм происходит аппаратно.

Изначально получатель просто ожидает, ничего не предпринимая, наступления какого-либо события. В наших примерах это будет выражаться в вызове процедуры wait_for_event(&event), где параметр event возвращает информацию о произошедшем событии. Ясно, что в действительности никто не будет ожидать в цикле (будут использованы прерывания), но мы для простоты будем считать так.

Когда кадр поступает к получателю, контрольная сумма вычисляется аппаратно. Если она неверна, то канальному уровню сообщается: event=cksum_err. Если кадр поступил без повреждений, то канальный уровень информируется так: event=frame_arrivel.

На рисунке 3-8 приведены многие структуры, используемые позднее. Там же указаны процедуры, которые используются при построении протоколов.

Рисунок 3-8. Перечень функций, используемых в описании протокола канального уровня

```
#define MAX_PKT 1024          /* determines packet size in bytes */

typedef enum {false, true} boolean;  /* boolean type */
typedef unsigned int seq_nr;         /* sequence or ack numbers */
typedef struct {unsigned char data[MAX_PKT];} packet;/* packet definition */
typedef enum {data, ack, nak} frame_kind;  /* frame_kind definition */

typedef struct {                     /* frames are transported in this layer */
  frame_kind kind;                   /* what kind of a frame is it? */
  seq_nr seq;                        /* sequence number */
  seq_nr ack;                        /* acknowledgement number */
  packet info;                       /* the network layer packet */
} frame;

/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);

/* Stop the clock and disable the timeout event. */
void stop_timer(seq_nr k);

/* Start an auxiliary timer and enable the ack_timeout event. */
void start_ack_timer(void);

/* Stop the auxiliary timer and disable the ack_timeout event. */
void stop_ack_timer(void);

/* Allow the network layer to cause a network_layer_ready event. */
void enable_network_layer(void);

/* Forbid the network layer from causing a network_layer_ready event. */
void disable_network_layer(void);

/* Macro inc is expanded in-line: Increment k circularly. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

Как мы уже отмечали, для того чтобы обнаруживать случаи потери кадров, уровень канала, отправляя кадр, должен устанавливать таймер. Если подтверждение не придет раньше, чем истечет время таймера, то считается, что кадр не дошел. В этом случае event=timeout. Процедуры start_timer и stop_timer используют для пуска и остановки таймера. Процедуру запуска таймера можно вызывать, не ожидая окончания предыдущего запуска. Подобное обращение будет означать перезапуск таймера на новый интервал.

Процедуры start_act_timer и stop_act_timer используются для управления дополнительным таймером, используемым в определенных случаях для уведомления.

Процедуры enable_network_layer и disable_network_layer используются в сложных протоколах, когда предполагается, что на сетевом уровне нет пакетов для передачи. Когда канальный уровень разрешит сетевому уровню снова передавать пакеты, сетевой информирует об этом событием event=network_layer_ready. Если канальный уровень выполнил процедуру disable_network_layer, то event=network_layer_ready может и не последовать. Таким способом канальный уровень может управлять потоком от сетевого уровня.

Симплекс-протокол без ограничений.

На рисунке 3-9 представлен простейший протокол канального уровня. Данные передаются только в одном направлении. Получатель и отправитель всегда готовы к отправке и получению данных. Время обработки данных игнорируется. Предполагается, что буфер неограниченного размера. Данные в канале не теряются и не искажаются.

Рисунок 3-9. Симплексный протокол канального уровня

```
/* Protocol 1 (utopia) provides for data transmission in one direction only, from
   sender to receiver.  The communication channel is assumed to be error free,
   and the receiver is assumed to be able to process all the input infinitely fast.
   Consequently, the sender just sits in a loop pumping data out onto the line as
   fast as it can. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
  frame s;                      /* buffer for an outbound frame */
  packet buffer;                /* buffer for an outbound packet */

  while (true) {
      from_network_layer(&buffer);        /* go get something to send */
      s.info = buffer;          /* copy it into s for transmission */
      to_physical_layer(&s);    /* send it on its way */
  }                             /* Tomorrow, and tomorrow, and tomorrow,
                                   Creeps in this petty pace from day to day
                                   To the last syllable of recorded time
                                      - Macbeth, V, v */
}

void receiver1(void)
{
  frame r;
  event_type event;             /* filled in by wait, but not used here */

  while (true) {
      wait_for_event(&event); /* only possibility is frame_arrival */
      from_physical_layer(&r);/* go get the inbound frame */
      to_network_layer(&r.info);          /* pass the data to the network layer */
  }
}
```

Симплексный старт-стопный протокол.

Теперь снимем одно из ограничений предыдущего протокола - способность сетевого уровня обрабатывать поступающие данные сколь угодно быстро. Все остальные предположения остаются в силе: канал абсолютно надежный, трафик однонаправленный.

Основная проблема - как предотвратить ситуацию, когда отправитель «заваливает» данными получателя. Если получателю требуется время Δt, чтобы исполнить from_physical_layer плюс to_network_layer, то отправитель должен передавать данные со средней скоростью один кадр в Δt.

Решением такой проблемы может быть введение коротких специальных служебных сообщений. Получатель, получив один или несколько кадров, отправляет отправителю короткий специальный кадр, означающий, что отправитель может передавать следующий. Это так называемый старт-стопный протокол, показанный на рисунке 3-10.

Рисунок 3-10. Однонаправленный старт-стопный протокол канального уровня

```
/* Protocol 2 (stop-and-wait) also provides for a one-directional flow of data from
   sender to receiver. The communication channel is once again assumed to be error
   free, as in protocol 1. However, this time, the receiver has only a finite buffer
   capacity and a finite processing speed, so the protocol must explicitly prevent
   the sender from flooding the receiver with data faster than it can be handled. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;                    /* buffer for an outbound frame */
  packet buffer;              /* buffer for an outbound packet */
  event_type event;           /* frame_arrival is the only possibility */

  while (true) {
      from_network_layer(&buffer);                    /* go get something to send */
      s.info = buffer;        /* copy it into s for transmission */
      to_physical_layer(&s);  /* bye bye little frame */
      wait_for_event(&event); /* do not proceed until given the go ahead */
  }
}

void receiver2(void)
{
  frame r, s;                 /* buffers for frames */
  event_type event;           /* frame_arrival is the only possibility */
  while (true) {
      wait_for_event(&event); /* only possibility is frame_arrival */
      from_physical_layer(&r); /* go get the inbound frame */
      to_network_layer(&r.info);/* pass the data to the network layer */
      to_physical_layer(&s);  /* send a dummy frame to awaken sender */
  }
}
```

Симплексный протокол для канала с шумом.

Основная проблема при передаче состоит в том, что кадр с подтверждением о получении может потеряться целиком. Как отличить кадр, переданный первый раз, от кадра, переданного повторно?

Одно из очевидных решений - нумерация передаваемых кадров. Однако сколько места отводить под эту нумерацию? Поскольку проблема различения стоит для кадров m и m+1, то достаточно одного разряда. 0 - для только что посланного кадра и 1 - для следующего ожидаемого. Все кадры, не содержащие корректной нумерации, просто сбрасываются при приеме.

На рисунке 3-11 показана программа для этого варианта протокола.

Рисунок 3-11. Протокол с подтверждением и восстановлением

```
/* Protocol 2 (stop-and-wait) also provides for a one-directional flow of data from
   sender to receiver. The communication channel is once again assumed to be error
   free, as in protocol 1. However, this time, the receiver has only a finite buffer
   capacity and a finite processing speed, so the protocol must explicitly prevent
   the sender from flooding the receiver with data faster than it can be handled. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;                      /* buffer for an outbound frame */
  packet buffer;                /* buffer for an outbound packet */
  event_type event;             /* frame_arrival is the only possibility */

  while (true) {
      from_network_layer(&buffer);                    /* go get something to send */
      s.info = buffer;          /* copy it into s for transmission */
      to_physical_layer(&s);    /* bye bye little frame */
      wait_for_event(&event);   /* do not proceed until given the go ahead */
  }
}

void receiver2(void)
{
  frame r, s;                   /* buffers for frames */
  event_type event;             /* frame_arrival is the only possibility */
  while (true) {
      wait_for_event(&event);   /* only possibility is frame_arrival */
      from_physical_layer(&r);  /* go get the inbound frame */
      to_network_layer(&r.info);/* pass the data to the network layer */
      to_physical_layer(&s);    /* send a dummy frame to awaken sender */
  }
}
```

Билет № 21.

Проблемы передачи данных на канальном уровне (Сервис, предоставляемый сетевому уровню, Разбиение на кадры, Контроль ошибок, Управление потоком). Обнаружение и исправление ошибок (Коды исправляющие ошибки, Коды обнаруживающие ошибки).

При рассмотрении физической среды мы отмечали, что у беспроводных систем связи и аналоговых каналов, например, абонентской линии в телефонных сетях, достаточно высокий уровень ошибок в канале. Поэтому ошибки при передаче на физическом уровне - это реальность, которую надо обязательно учитывать.

В разных средах характер ошибок разный. Ошибки могут быть одиночные, а могут возникать группами, сразу по несколько штук. У групповых ошибок есть свои достоинства и недостатки. Достоинство заключается в следующем. Пусть данные передаются блоками по 1000 бит, а частота ошибки - 10^{-3} на бит, т.е. одна на каждые 1000 бит.

Если ошибки изолированные и независимые, то практически каждый блок в среднем будет содержать ошибку. Если же они возникают группами по 100 сразу, то ошибки будут содержать в среднем 1-2 блока из каждых 100. 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 возникли искажения. Это так называемые коды с исправлением ошибок. Второй метод - 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 избыточных, контрольных разрядов. Нам необходимо передать слово длины n = m+r, которое называют n-битовым кодословом. Пусть у нас есть два кодослова - 10001001 и 10110001. С помощью операции EXCLUSIVE OR легко определить число различных разрядов в двух кодословах. В данном случае таких разрядов 3. Количество разных битов в двух кодословах называется расстоянием Хемминга между этими словами. Поэтому, если два кодослова находятся на расстоянии d по Хеммингу, это значит, что надо преобразовать ровно d разрядов, чтобы преобразовать одно кодослово в другое.

В силу того, что избыточные контрольные разряды могут принимать только вполне определенные значения, то не все 2^n кодовых слов возможны. Зная алгоритм установки контрольных разрядов, мы можем вычислить минимальное расстояние по Хеммингу между двумя правильными кодословами.

Способен код исправлять ошибки или только обнаруживать их - зависит от расстояния между кодословами по Хеммингу. Если мы хотим обнаруживать d ошибок, то необходимо, чтобы два кодослова отстояли друг от друга на расстоянии d+1. Тогда, если принятый код отстоит на расстоянии k<d, то принятое кодослово содержит k ошибок. Если мы хотим исправлять d ошибок, то нужно, чтобы кодослова отстояли друг от друга на 2d+1. Поэтому, даже если переданное кодослово содержит d ошибок, оно все равно ближе к правильному кодослову, чем к какому-либо еще, и, таким образом, можно определить исходное слово.

Простым примером кода с обнаружением одной ошибки является код с битом четности. Конструкция его такова: к исходному кодослову добавляется бит четности. Если число единиц в исходном кодослове четно, то значение этого бита - 0. Если нечетно, то - 1. Кодослова с битом четности имеют расстояние Хемминга 2, так как любая ошибка в одном бите породит ошибку четности. Однако, если возможны двойные ошибки, то бит четности проблему не решит.

Для примера кода с исправлением ошибки рассмотрим код, у которого есть только четыре правильных кодослова: 0000000000, 0000011111, 1111100000, 1111111111. Расстояние по Хеммингу у этого кода 5, следовательно, он может исправлять двойные ошибки. Если получатель получит слово 0000000111, то ясно, что исходное слово имело вид 0000011111. Однако, если допустимы тройные ошибки, то 0000000111 может означать 0000000000.

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^n 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 $(n+1)2^m \le 2$, учитывая, что n=m+r, получаем: $(m+r+1) \le 2^r$

Для заданного m эта формула задает минимальное число контрольных разрядов, необходимых для исправления единичных ошибок. Этот теоретический предел достижим при использовании метода, предложенного Хеммингом. Идея его в следующем:

- Разряды кодослова нумеруются слева направо, начиная с 1.
- Все биты, номера которых являются степенью 2 (1, 2, 4, 8, 16 и т.д.) - контрольные, остальные - биты сообщения.

- Каждый контрольный бит отвечает за четность группы битов, включая себя. Чтобы определить группу битов, за четность которой отвечает определенный контрольный бит, нужно представить номер позиции каждого бита по степеням двойки. Те биты, в номера которых входит степень двойки, равная номеру контрольного бита, и есть искомая группа. Например, 11=1+2+8, 39=1+2+4+32. Таким образом, бит в позиции 11 входит в группу, контролируемую битом в позиции 2.

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 заносится в счетчик. Если после этой проверки счетчик на нуле, то все в порядке. Если нет, то он содержит номер неправильного разряда. Например, если 1, 2, 8 - ошибочные контрольные разряды, то ошибка содержится в 11-м разряде, так как только он связан одновременно с этими контрольными разрядами.

Код Хемминга может исправлять только одиночные ошибки. Однако есть прием, который позволяет распространить идеи Хемминга на случай групповых ошибок. Пусть нам надо передать k кодослов. Расположим их в виде матрицы: одно слово - строка. Обычно передают слово за словом. Но мы поступим иначе, передадим слово длины k из первых разрядов всех слов, затем - вторых, и т.д. После приема всех слов матрица восстанавливается. Если мы хотим обнаруживать групповые ошибки размера k, то в каждой строке восстановленной матрицы будет не более одной ошибки. А с одиночными ошибками код Хемминга справится.

Коды с обнаружением ошибок.

Рассмотрение кодов, обнаруживающих ошибки, начнем с небольшого примера. Пусть у нас есть канал с одиночными ошибками с частотой 10^{-6} на бит. Если мы хотим исправлять единичные ошибки при передаче блока в 1000 бит, то нам потребуется 10 контрольных бит ($(m+r+1) \le 2^r r$, где m =1000; $(1001+r) \le 2^r r$, следовательно, r = 10).

При передаче 1 Мбит данных потребуется 10 000 контрольных бит. В то же время для обнаружения единичной ошибки достаточно одного бита четности. Поэтому, если мы применим технику повторной передачи, то на передачу 1000 блоков надо будет потратить 1001 бит дополнительно или с повторной передачей 2002 бит, вместо 10000 бит в случае кода с исправлением ошибки.

Применение техники четности «в лоб» в случае групповых ошибок не даст нужного результата. Однако ее можно скорректировать. Пусть нам требуется передать n слов по k бит. Расположим их в виде матрицы n x k. Для каждого столбца вычислим бит четности и разместим его в дополнительной строке. Матрица затем передается по строкам. При получении матрица восстанавливается, и если хоть один бит нарушен, то весь блок передается повторно.

Этот метод позволяет обнаружить групповые ошибки длины n. Против групповых ошибок длины n+1 он бессилен. В общем случае вероятность правильной передачи при длине групповой ошибки n равна 2^{-n}. Поэтому на практике применяют другую технику, которая называется циклическим избыточным кодом (Cyclic Redundancy Code), или CRC-кодом.

CRC-коды построены на рассмотрении битовой строки как строки коэффициентов полинома. Битовую строку длины k рассматривают как коэффициенты полинома степени k-1. Самый левый бит строки - коэффициент при старшей степени. Например, строка 110001 представляет полином $x^5+x^4+x^0$.

Использование полиномиальных кодов при передаче заключается в следующем. Отправитель и получатель заранее договариваются о конкретном генераторе полиномов G(x), у которых 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 должно быть r<m. 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. Полиномиальная арифметика выполняется по модулю 2. Сложение и вычитание происходит без переноса разрядов. Таким образом, обе эти операции эквивалентны EXCLUSIVE OR. Деление выполняется, как обычно в двоичной системе, с той лишь разницей, что вычитание выполняется по модулю два.

Алгоритм вычисления контрольной суммы:

Здесь r - степень G(x).

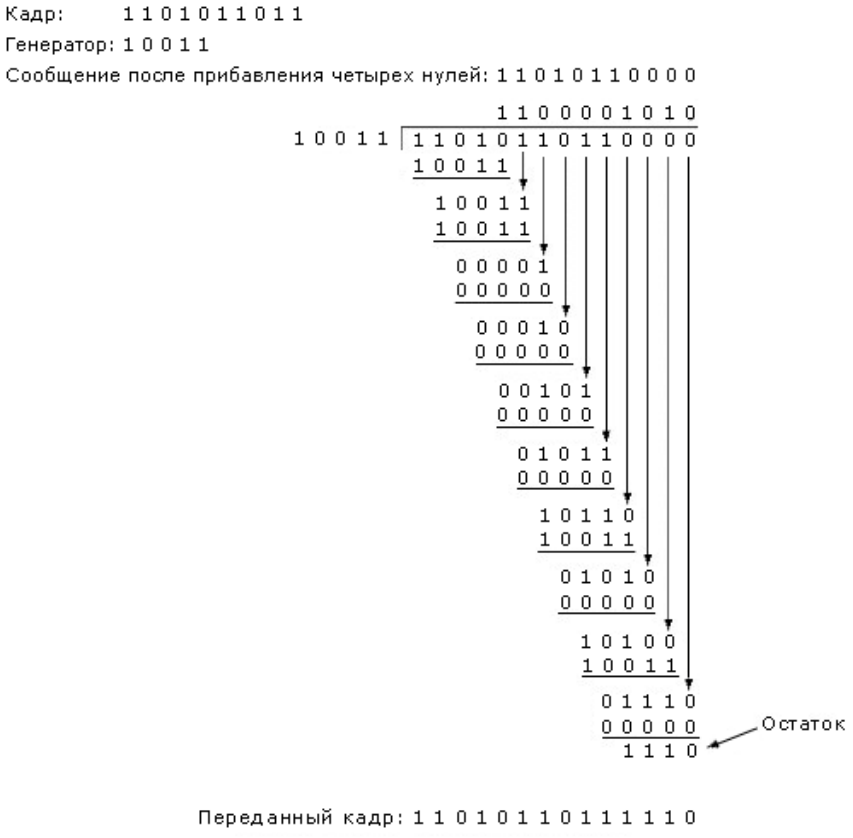
1. Добавить r нулей в конец блока так, что он содержал m+r разрядов и соответствовал полиному $x^r M(x)$.

2. Разделить по модулю 2 полином $x^r M(x)$ на G(x).

3. 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 (назовем его T(x)).

Рисунок 3-7 показывает этот алгоритм для блока 1101011011 и $G(x) = x^4+x+1$.

Рисунок 3-7. Расчет контрольной суммы для полиномиального кода



Данный метод позволяет обнаруживать одиночные ошибки. Групповые ошибки длины не более r. Нечетное число отдельных ошибок. 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, групповые ошибки длины не более 16 и нечетное число изолированных ошибок с вероятностью 99,997%.

Билет № 22.

Проблемы передачи данных на канальном уровне (Сервис, предоставляемый сетевому уровню, Разбиение на кадры, Контроль ошибок, Управление потоком). Протоколы скользящего окна.

Для передачи в обоих направлениях можно потребовать на физическом уровне двух симплексных каналов. Один для передачи кадров, другой - для передачи подтверждений. Однако использование канала только для подтверждений - довольно дорогое удовольствие. Можно смешивать кадры с данными и кадры с подтверждениями на одном канале. Это, конечно. решение проблемы, но по-прежнему на подтверждения будет тратиться полезная пропускная способность канала.

А что, если для подтверждения использовать полезные кадры с данными? Получатель не сразу отправляет подтверждение, а ожидает от сетевого уровня очередного пакета. Как только такой пакет возникает, то канальный уровень помещает в кадр с пакетом также уведомление о получении в специальное поле ack. Такой прием позволяет полнее использовать имеющуюся пропускную способность канала. Меньше кадров - меньше прерываний на канальном уровне на их обработку, меньше затрат на буферизацию.

Однако применение этой идеи усложняет протокол. Что делать, если тайм-аут у отправителя на получения подтверждения заканчивается, а с сетевого уровня получателя не поступает запроса на передачу пакета? Поэтому на канальном уровне должен быть фиксированный интервал времени, в течение которого канальный уровень ждет от сетевого попутного кадра. Если до истечения этого срока пакет с сетевого уровня не поступил, то канальный уровень отправляет подтверждение отдельным кадром.

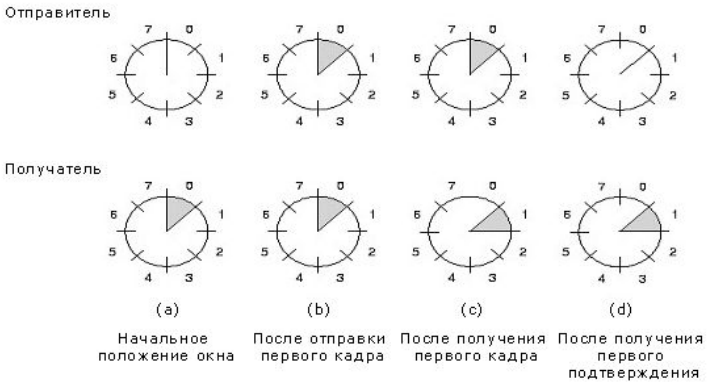
Рассмотренный здесь протокол является представителем класса протоколов скользящего окна. Кроме вышесказанного, протоколы этого класса делают следующее: у отправителя и получателя есть определенная константа n - число кадров, которое отправитель может послать, не ожидая подтверждения для каждого кадра. По мере получения подтверждений отправленные кадры будут сбрасываться из буфера отправителя, и буфер будет пополняться новыми кадрами.

Мы уже сталкивались с подобными протоколами (старт-стопный протокол). В них n было равно 1. Обычно $n=2^k-1$. У получателя и отправителя есть набор последовательных чисел - номеров кадров, которые отправитель может отправить, не ожидая подтверждения каждого. Эти кадры образуют окно отправки. Аналогично, у получателя есть буфер для получения и временного хранения получаемых кадров - окно получения.

Хотя в этих условиях у отправителя есть определенная свобода в порядке отправления кадров, мы по-прежнему будем считать, что кадры отправляют в соответствии с порядковыми номерами. У окон отправки и получения есть верхняя и нижняя границы. Порядковые номера кадров в окне отправки - кадры отправленные, но не подтвержденные. Как только от сетевого уровня поступил еще один пакет, ему присваивают первый свободный наибольший номер, и верхняя граница окна отправителя поднимается. Как только приходит подтверждение, нижняя граница окна поднимается. Таким образом, в окне все время находятся неподтвержденные кадры.

Рисунок 3-12 показывает работу такого протокола для n=1 в форме диаграммы.

Рисунок 3-12. Протокол скользящего окна



Протокол скользящего окна в 1 бит.

Прежде чем переходить к общему случаю, рассмотрим протокол скользящего окна с максимальным размером окна в 1 бит. Такой протокол использует старт-стопный режим и, послав кадр, не шлет другой, пока не придет подтверждение на первый.

На рисунке 3-13 показан текст протокола для этого простейшего случая. Как и все, он начинается с определения переменных. Next_frame_to_send указывает, какой кадр посылается. Переменная frame_expected определяет, какой кадр получатель ожидает. Есть только два значения - 0 или 1.

Рисунок 3-13. Протокол скользящего окна в 1 бит

```
/* Protocol 4 (sliding window) is bidirectional and is more robust than protocol 3. */
#define MAX_SEQ 1                       /* must be 1 for protocol 4 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void protocol4 (void)
{
  seq_nr next_frame_to_send;            /* 0 or 1 only */
  seq_nr frame_expected;                /* 0 or 1 only */
  frame r, s;                           /* scratch variables */
  packet buffer;                        /* current packet being sent */
  event_type event;
  next_frame_to_send = 0;               /* next frame on the outbound stream */
  frame_expected = 0;                   /* number of frame arriving frame expected */
  from_network_layer(&buffer);          /* fetch a packet from the network layer */
  s.info = buffer;                      /* prepare to send the initial frame */
  s.seq = next_frame_to_send;           /* insert sequence number into frame */
  s.ack = 1 - frame_expected;           /* piggybacked ack */
  to_physical_layer(&s);                /* transmit the frame */
  start_timer(s.seq);                   /* start the timer running */
  while (true) {
      wait_for_event(&event);           /* frame_arrival, cksum_err, or timeout */
      if (event == frame_arrival) { /* a frame has arrived undamaged. */
          from_physical_layer(&r);      /* go get it */
          if (r.seq == frame_expected) {
              /* Handle inbound frame stream. */
              to_network_layer(&r.info);    /* pass packet to network layer */
              inc(frame_expected);  /* invert sequence number expected next */
          }
          if (r.ack == next_frame_to_send) { /* handle outbound frame stream. */
              from_network_layer(&buffer);          /* fetch new pkt from network layer */
              inc(next_frame_to_send);      /* invert sender's sequence number */
          }
      }
      s.info = buffer;                  /* construct outbound frame */
      s.seq = next_frame_to_send;       /* insert sequence number into it */
      s.ack = 1 - frame_expected;       /* seq number of last received frame */
      to_physical_layer(&s);            /* transmit a frame */
      start_timer(s.seq);               /* start the timer running */
  }
}
```

Есть два случая: первый - простой и наиболее удобный, когда только один из канальных уровней первым начинает передачу. В этом случае вне тела основного цикла одной из программ канального уровня есть обращения к процедурам to_phisical_layer и start_timer. Случай, когда оба

уровня одновременно могут начинать передачу, описывается позже, поскольку он требует более детального рассмотрения.

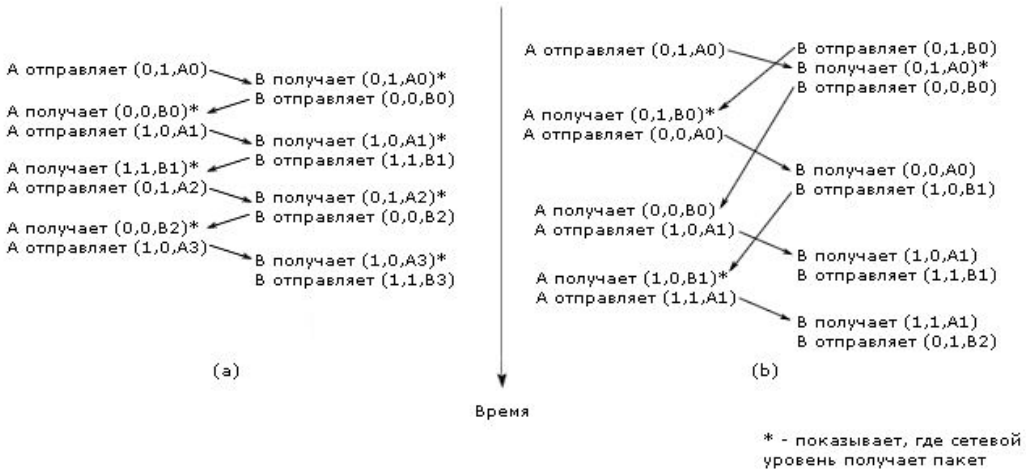
Машина, инициирующая обмен, берет пакет от сетевого уровня, формирует кадр и посылает его. Когда он (или любой другой кадр) поступает, канальный уровень-получатель проверяет: не является ли этот кадр дубликатом. Если поступивший кадр тот, что ожидался, то он передается на сетевой уровень и окно получателя сдвигается вверх.

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

На рисунке 3-14 показан протокол 4. Если у A очень короткий тайм-аут, то все дубликаты кадра пойдут с одним и тем же значением полей seq и ask. Поэтому, получив исправный кадр, B установит значение переменной frame_expected равным 1 и пошлет подтверждение. Все последующие дубликаты будут им отвергнуты, так как он будет ожидать кадра с 1, а не 0.

Случай, когда оба канальных уровня начинаю передачу одновременно, показан на рисунке 3-14 (b). В нем возникает много повторных передач одного и того же кадра даже при отсутствии ошибок в передаче.

Рисунок 3-14. Два сценария для протокола 4



Протокол с возвратом на n кадров и протокол с выборочным повтором.

До сих пор мы предполагали, что время доставки кадра и время доставки подтверждения пренебрежимо малы. В некоторых случаях это предположение очевидно не работает. Оно может приводить к серьезным бесполезным тратам пропускной способности канала.

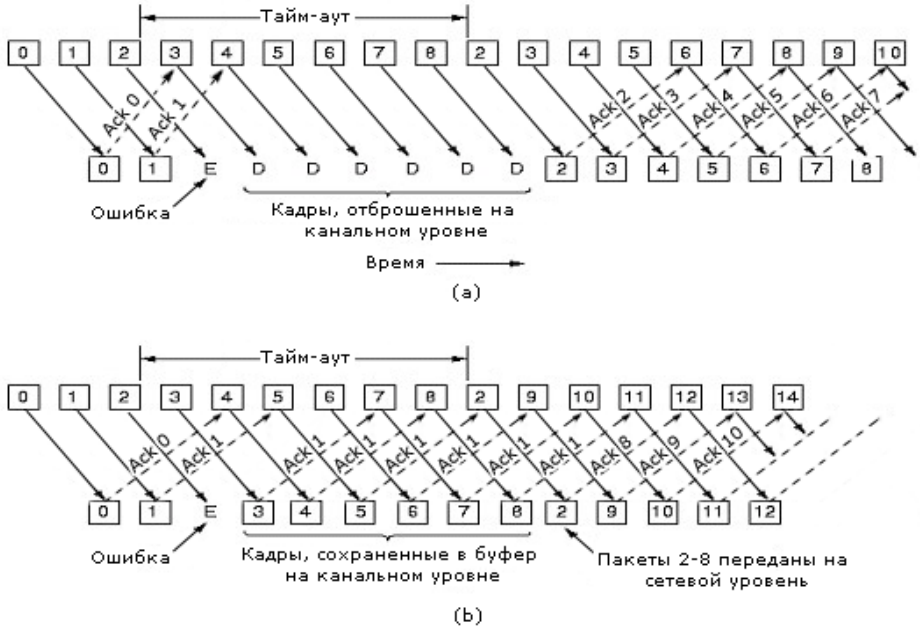
Эта проблема есть следствие правила, по которому отправитель ждет подтверждения прежде, чем пошлет следующий кадр. Это требование можно ослабить - разрешить отправителю отправлять до w кадров, не дожидаясь их подтверждения. Надлежащим выбором значения w отправитель может заполнить все время, необходимое на отправку кадра и получение его подтверждения. В вышеприведенном примере w должно быть равным, по крайней мере, 26. Это то количество кадров, какое отправитель успеет отправить за 520 мсек., прежде чем придет подтверждение на кадр 0. Таким

образом, неподтвержденными будут 25 из 26 кадров, размер окна отправителя будет равным 26 кадров.

Эта техника известна как конвейер. Ее применение в случае ненадежного канала наталкивается на ряд проблем. Первая - что делать, если в середине потока пропадет или попадется поврежденный кадр? Получатель уже получит большое количество кадров к тому моменту, когда отправитель обнаружит, что что-то произошло. Когда получатель получил поврежденный кадр, он его должен сбросить, что делать с последующими кадрами? Помните, что канальный уровень обязан передавать пакеты на сетевой уровень в том порядке, в каком их отправлял отправитель.

Есть два приема для решения этих вопросов: откат и выборочный повтор. При откате все кадры, поступившие после поврежденного кадра, сбрасываются и не подтверждаются. Отправитель по тайм-ауту повторно отправляет все кадры, начиная с первого неподтвержденного кадра. Этот подход показан на рисунке 3-15 (а), где размер окна у получателя - 1.

Рисунок 3-15. Влияние ошибки при окне размером 1 (а) и окне большого размера (b)



При выборочном повторе у получателя длина окна такая же, как и у отправителя. Отправитель отмечает неподтвержденный кадр и посылает его еще раз. Получатель не передает на сетевой уровень последовательность пакетов, если в ней есть разрывы (рисунок 3-15 (b)).

Билет № 23.

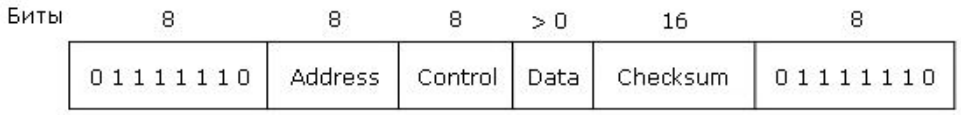
Проблемы передачи данных на канальном уровне (Сервис, предоставляемый сетевому уровню, Разбиение на кадры, Контроль ошибок, Управление потоком). Примеры протоколов канального уровня (HDLC, Frame Relay).

Протокол HDLC (High Level Data Link Control).

Познакомимся с группой давно известных, но по-прежнему широко используемых на практике протоколов. Все они имеют одного предшественника - SDLC (Synchronous Data Link Control) - протокола управления синхронным каналом, предложенного фирмой IBM в рамках архитектуры SNA. ISO модифицировало этот протокол и выпустило под название HDLC - High level Data Link Control. МКТТ модифицировало HDLC для X.25 и выпустило под именем LAP - Link Access Procedure. Позднее он был модифицирован в LAPB.

Все эти протоколы построены на одних и тех же принципах. Они используют технику вставки специальных последовательностей битов и являются бит–ориентированными протоколами. Различия между ними незначительные.

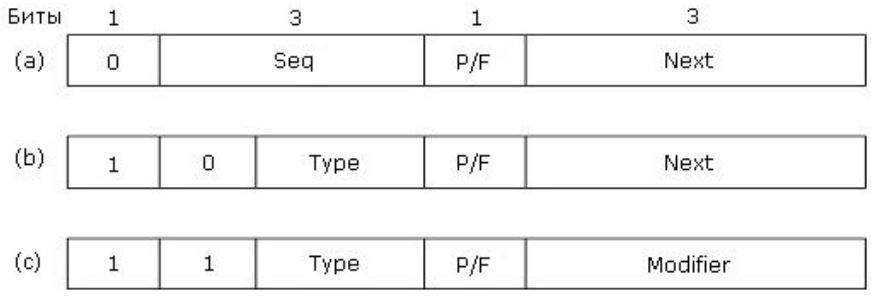
Рисунок 3-16. Типовая структура кадра протокола HDLC



- Поле Address используют для адресации терминала, если их несколько на линии. Для линий точка-точка это поле используется для того, чтобы отличать команду от ответа.
- Поле Control используется для последовательных номеров кадров, подтверждений и других нужд.
- Поле Data может быть сколь угодно большим и используется для передачи данных. Надо только иметь в виду, что чем длиннее это поле, тем больше вероятность повреждения кадра на линии.
- Поле Checksum - это поле используется для передачи CRC-кода.

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

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



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

Кадры Supervisory бывают четырех типов.

- Тип 0 - уведомление в ожидании следующего кадра (RECEIVE READY). Используется, когда нет встречного трафика, чтобы передать уведомление в кадре с данными.
- Тип 1 - негативное уведомление (REJECT) - указывает на ошибку при передаче. Поле Next указывает номер кадра, начиная с которого надо перепослать кадры.

- Тип 2 - RECEIVE NOT READY. Подтверждает все кадры, кроме указанного в Next. Используется, чтобы сообщить источнику кадров о необходимости приостановить передачу в силу каких-то проблем у получателя. После устранения этих проблем получатель шлет RECEIVE REDAY, REJECT или другой надлежащий управляющий кадр.
- Тип 3 - SELECTIVE REJECT - указывает на необходимость перепослать только кадр, указанный в Next. LAPB и SDLC не используют кадры этого типа.

Третий класс кадров - Unnumbered. Кадры этого класса иногда используются для целей управления, но чаще для передачи данных при ненадежной передаче без соединения.

Все протоколы имеют команду DISConnect - для сообщения о разрыве соединения. Команды SNRM и SABM используются для установки счетчиков кадров в ноль, сброса соединения в начальное состояние, установки соподчиненности на линии. Команда FRMR указывает на повреждение управляющего кадра (например, когда контрольная сумма верна, а значения полей противоречивы).

Frame Relay.

Ретрансляция кадров (Frame Relay, FR) - это метод доставки сообщений в сетях передачи данных (СПД) с коммутацией пакетов. Первоначально разработка стандарта FR ориентировалась на цифровые сети интегрированного обслуживания (ISDN - Integrated Services Digital Networks), однако позже стало ясно, что FR применим и в других СПД (здесь под данными понимается любое сообщение, представленное в цифровой форме). К числу достоинств рассматриваемого метода прежде всего необходимо отнести малое время задержки, простой формат кадров, содержащих минимум управляющей информации, и независимость от протоколов верхних уровней модели OSI.

В настоящее время разработкой и исследованием стандартов FR занимаются три организации: Frame Relay Forum (FRF) - международный консорциум, включающий в себя свыше 300 поставщиков оборудования и услуг, American National Standards Institute (ANSI, Американский национальный институт по стандартизации), Международный союз электросвязи (ITU-T).

В январе 1992 г. этот проект стандарта, включающего в себя спецификации ANSI, которые обязательны для выполнения членами FRF, был доработан Техническим комитетом FRF и утвержден собранием членов FRF.

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

Рисунок 3-18. Структура и формат кадра Frame Relay

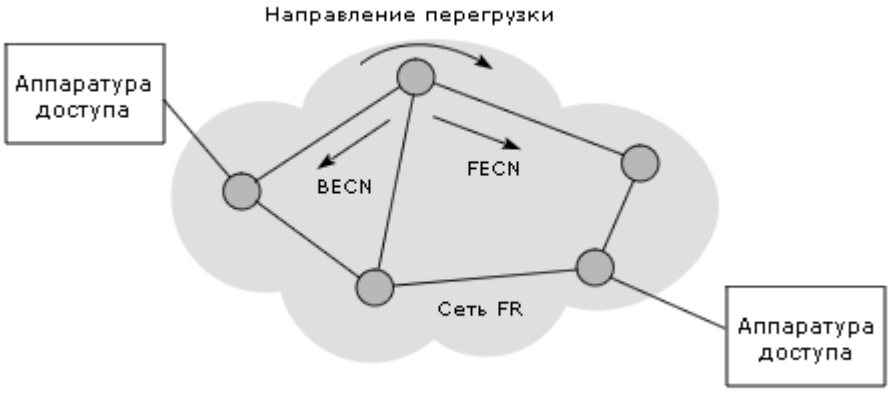
Структура кадра FR (рисунок 3-18) включает в себя следующие элементы:

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

2. Заголовок:

- Адрес в пределах кадра FR (стандарт FRF), состоит из шести бит первого байта и четырех бит второго байта заголовка кадра (стандарты ANSI и ITU-T допускают размер заголовка до 4 байтов). Эти 10 бит представляют собой идентификатор канала передачи данных (Data Link Connection Identifier, DLCI) и определяют абонентский адрес в сети FR.
- Бит «опрос/финал» (Command/ Response - CR) зарезервирован для возможного применения в различных протоколах более высоких уровней управления OSI. Этот бит не используется протоколом FR и «прозрачно» пропускается аппаратно-программными средствами сети FR.
- Бит расширения адреса (Extended Address - EA). DLCI содержится в 10 битах, входящих в два байта заголовка. Однако возможно расширение заголовка на целое число дополнительных байтов с целью указания адреса, состоящего более чем из 10 бит. Бит EA устанавливают в конце каждого байта заголовка; если он имеет значение «1», то это означает, что данный байт в заголовке последний. Стандарт FRF рекомендует использовать заголовки, состоящие из двух байтов. В этом случае значение бита EA первого байта будет соответствовать «0», а второго - «1».
- Бит уведомления (сигнализации) приемника о явной перегрузке (Forward Explicit Congestion Notification - FECN) устанавливается в «1», если надо информировать получателя о том, что произошла перегрузка в направлении передачи данного кадра (рисунок 3-19).
- Бит уведомления (сигнализации) отправителя о явной перегрузке (Backward Explicit Congestion Notification - BECN). Этот бит устанавливают в «1» для уведомления отправителя сообщения о том, что произошла перегрузка в направлении, обратном направлению передачи содержащего этот бит кадра. Бит BECN может не использоваться терминалами абонентов (см. рисунок 3-19), т.е. в этом направлении возник слишком большой поток кадров.
- Бит разрешения сброса (Discard Eligibility - DE) устанавливают в «1» в случае явной перегрузки. Он указывает на то, что данный кадр может быть уничтожен в первую очередь, т.е. пользователю предоставлено право выбирать, какими кадрами он может «пожертвовать». Однако при перегрузках узлы коммутации сети FR уничтожают не только кадры с битом DE.

Рисунок 3-19. Установка бит перегрузки



3. Информационное поле содержит данные пользователя и состоит из целого числа байтов. Его максимальный размер определен стандартом FRF и составляет 1600 байтов (минимальный размер - 1 байт), но возможны и другие максимальные размеры (вплоть до 4096 байтов). Содержание информационного поля пользователя передается неизменным.

4. Проверочная последовательность кадра (Frame Check Sequence - FCS) используется для обнаружения возможных ошибок при его передаче и состоит из двух байтов. Данная последовательность формируется аналогично циклическому коду HDLC.

Все указанные поля должны присутствовать в каждом кадре FR, который передается между двумя оконечными пользовательскими системами.

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

Протокол FR является весьма простым по сравнению с HDLC и включает в себя небольшой свод правил и процедур организации информационного обмена. Основная процедура состоит в том, что если кадр получен без искажений, он должен быть направлен далее по соответствующему маршруту. При возникновении проблем, связанных с перегрузкой сети FR, ее узлы могут сбрасывать любой кадр.

Узлам сети FR разрешено уничтожать искаженные кадры, не уведомляя об этом пользователя. Искаженным считается кадр, которому присущ какой-либо из следующих признаков:

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

Для FR характерно:

- заполнение канала связи комбинацией «флаг» при отсутствии данных для передачи
- резервирование одного DLCI для интерфейса локального управления и сигнализации
- содержание поля данных пользователя в любом кадре не должно подвергаться какой-либо обработке со стороны аппаратуры канала данных (могут обрабатываться лишь данные в локальном канале управления)

Управление доступом к сети FR возлагается на интерфейс локального управления (Local Management Interface - LMI). Именно LMI реализует интерфейс UNI (Unified Network Interface). Доступ в сеть FR обеспечивают интерфейсы FR («порты FR») и FR-адаптеры - сборщики/разборщики кадров (FR assembler/disassembler, FRAD).

Добиться высокой эффективности использования пропускной способности физических линий и каналов связи, а также исключения перегрузок узлов связи и всей сети FR позволяет метод статистического мультиплексирования кадров, который подразумевает:

- постоянное наблюдение за потоком заявок от пользователей на передачу сообщений и за текущей загрузкой сети (линий, каналов и узлов связи)
- перераспределение свободного (и высвобождающегося) ресурса пропускной способности в соответствии с реальными потребностями абонентов
- предоставление пользователям каналов информационного обмена, удовлетворяющих их требованиям

Данный метод обеспечивает синхронный ввод сообщений пользователей в высокоскоростной канал связи на основе соглашений, заключенных между пользователем и поставщиком услуг сети FR. Услуги различаются по следующим параметрам:

- максимальный размер поля информации в кадре FR (в байтах)
- пропускная способность порта, посредством которого абонент подключается к сети FR

• гарантированная скорость передачи данных (Committed Information Rate, CIR) - при обеспечении требуемого качества доставки
• гарантированный объем передачи информации (Committed Burst Size, BC) - при обеспечении требуемого качества доставки
• дополнительный объем передачи информации (Excess Burst Size, BE) - при возможном снижении качества передачи данных

Предварительные соглашения реализуются следующим образом.

1. Абонент выбирает (и оплачивает) пропускную способность порта и гарантированную скорость передачи данных для фиксированного виртуального соединения (PVC).

2. Узел доступа к сети FR измеряет «реальную потребность абонента» в ресурсе пропускной способности канала связи.

3. Если этот ресурс (выраженный реальной скоростью передачи информации) не превышает CIR, то кадры передаются без изменений. Если требуемая скорость превышает CIR, но соответствует пропускной способности порта, то бит DE устанавливается в «1», что дает возможность удалять соответствующие кадры при возникновении перегрузок (абонент также имеет право решать, какие кадры для него менее важны). Наконец, если превышена пропускная способность порта, кадры уничтожаются вне зависимости от каких-либо условий.

Абонент способен воспользоваться предварительным соглашением и для того, чтобы уменьшить свои затраты следующим оригинальным способом. Некоторые операторы сетей (поставщики услуг) предлагают значительные скидки при передаче кадров с битом DE, установленным в «1». При наличии в сети значительного запаса пропускной способности абонент может установить гарантированную скорость передачи, равную нулю. В этом случае во всех передаваемых кадрах бит DE будет установлен в «1».

Билет № 24.

Проблемы передачи данных на канальном уровне (Сервис, предоставляемый сетевому уровню, Разбиение на кадры, Контроль ошибок, Управление потоком). Примеры протоколов канального уровня в Internet (протоколы SLIP, PPP, Уровень канала данных в ATM).

Рассмотрим протоколы, которые используются для каналов «точка-точка» в Интернете. На уровне канала данных соединения «точка-точка» возникают между маршрутизаторами либо коммутирующими элементами в СПД. Другой часто встречающийся случай для таких соединений - соединение из дома через модем с интернет-провайдером.

Для упомянутых выше соединений: «маршрутизатор-маршрутизатор» и «хост-маршрутизатор» через телефонную линию было предложено два протокола: SLIP и PPP.

SLIP - Serial Line IP.

SLIP - наиболее старый из этих двух протоколов. Он был создан в 1984 году для соединения рабочих станций SUN через модем. Этот протокол был описан в RFC 1055. Его работа очень проста: он вставляет специальные флаг-байты в начало и конец IP-пакета.

Последние версии этого протокола осуществляют также сжатие заголовков TCP и IP у последовательных пакетов, так как они несут очень много одинаковой информации. Одна из последних версий этого протокола описана в RFC 1144.

SLIP имеет ряд серьезных недостатков - он не занимается контролем и исправлением ошибок, оставляя это протоколам верхних уровней. Во-вторых, он работает только с IP-пакетами. В современных условиях, когда Интернет объединяет самые разнообразные сети, это серьезный недостаток.

В-третьих, IP-адреса взаимодействующих сторон должны быть известны заранее. В условиях нехватки IP-адресов это недостаток, так как было бы удобнее задавать IP-адрес динамически, лишь на период действия соединения.

В-четвертых, этот протокол не обеспечивает какой-либо проверки аутентичности взаимодействующих сторон. Так что вы не можете быть уверены, с кем вы общаетесь.

В-пятых, для этого протокола нет стандарта, и существует множество его версий, не все из которых совместимы.

PPP - протокол «точка-точка».

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

Протокол PPP обеспечивает три основных функции:

1. Распознавание кадров. Однозначно определяется конец кадра и начало нового. Здесь же происходит обнаружение ошибок.

2. Управление линией, т.е. активизация линии, ее проверка, определение основных параметров передачи в диалоге, корректное завершение передачи со сбросом параметров. Этот протокол называет LCP (Link Control Protocol).

3. Определение основных параметров соединения между сетевыми уровнями, чтобы обеспечить независимость от реализации сетевого уровня. Выбранный метод предполагает наличие разных NCP (Network Control Protocol) на каждом поддерживаемом сетевом уровне.

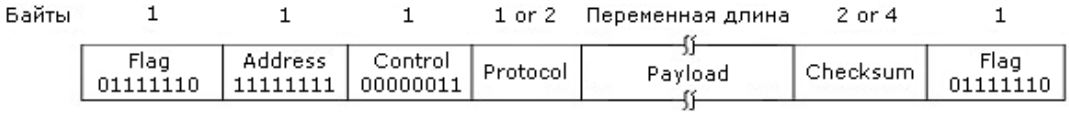
Чтобы лучше понять, как это все работает вместе, рассмотрим типичный сценарий, когда пользователь из дома по телефонной линии хочет подключить свой PC к Интернету. PC звонит на маршрутизатор сервис-провайдера. После того как маршрутизатор принял звонок и установил физическое соединение, PC посылает несколько LCP-пакетов в PPP-кадрах. Маршрутизатор отвечает LCP-пакетами в PPP-кадрах. В результате такого обмена определяются параметры соединения.

После этого следует обмен NCP-пакетами для настройки сетевого уровня. В частности, здесь происходит временное присваивание PC IP-адреса, который действует только на период соединения. Это происходит, если обе стороны хотят использовать TCP/IP-стек.

Теперь, когда PC стала полноправной машиной в Интернете, она может обмениваться IP-пакетами с другими машинами. Когда пользователь закончит работу, NCP разрывает соединение с сетевым уровнем и освобождает ранее занятый IP-адрес. После этого LCP-протокол разрывает соединение на канальном уровне. А затем компьютер говорит модему: «Положи трубку».

PPP-кадры имеют формат, очень близкий к HDLC-кадрам. Основное различие состоит в том, что PPP - байт-ориентированный, а HDLC - бит-ориентированный. Для HDLC возможен кадр размером в 30,25 байт, а для PPP - нет.

Рисунок 3-21. Формат PPP-кадра



Все PPP-кадры начинаются со стандартного байта: 01111110. Поле «Address» по умолчанию равно 11111111. Поле «Control» по умолчанию равно 00000011, что означает «Unnumbered-кадр», т.е. нумерация передаваемых кадров и подтверждений в их получении не предполагается. В случае ненадежной среды передачи данных есть вариант надежной передачи, описанный в RFC 1663.

Так как значения полей «Address» и «Control» - константы, то LCP-протокол опускает их, экономя два байта на передаче. В поле «Protocol» указывается, какой тип пакетов будет в поле «Payload». Там допускаются пакеты протоколов LCP, NCP, IP, IPX, Apple Talk и других. Поле «Payload» имеет переменную длину, по умолчанию она равна 1600 байт.

Рисунок 3-22. Основные фазы установления соединения и его разрыва

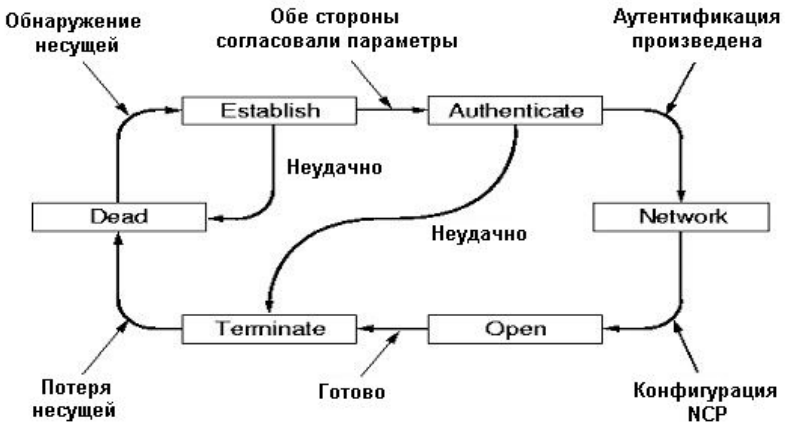


Таблица 3-23. Типы LCP-пакетов, допустимых по протоколу RFC 1661

Название	Направление	Значение
Configure-request	→	Список предлагаемых параметров и их значений
Configure-ack	←	Все параметры приняты
Configure-nak	←	Некоторые параметры не приняты
Configure-reject	←	Некоторые параметры недоступны
Terminate-request	→	Требуется закрыть соединение
Terminate-ack	←	ОК, соединение закрыто
Code-reject	←	Получен неизвестный запрос
Protocol-reject	←	Запрошен неизвестный протокол

Echo-request	→	Пришлите кадр обратно
Echo-reply	←	Вот ваш кадр
Discard-request	→	Сбросьте этот кадр (для проверки)

Уровень канала данных в ATM.

Физический уровень в ATM покрывает физический уровень и уровень канала данных в OSI. Поскольку физический уровень ATM на подуровне физической зависимости не предъявляет каких-то специальных требований к физической среде, то мы сосредоточим наше внимание на TC-подуровне - подуровне преобразования при передаче.

Когда прикладная программа посылает сообщение, оно движется вниз по ATM-стеку, получая заголовки, концевики, разбивается на ячейки и т.д. Проследим, что с ним происходит, когда ячейки достигают TC-подуровня и далее.

Первый шаг - вычисление контрольной суммы заголовка. Заголовок состоит из 5 байт - 4 байта идентифицируют виртуальное соединение и несут контрольную информацию, за ними следует 1 байт с контрольной суммой. Контрольная сумма защищает только первые четыре байта и не затрагивает данные в ячейке. Контрольная сумма вычисляется как остаток от деления содержимого 4 байтов на полином x8+x2+x+1. К этому остатку добавляется константа 01010101 для повышения надежности, в случае если заголовок содержит много нулей.

Решение защищать контрольной суммой только управляющую информацию было принято с целью сократить затраты на обработку на нижних уровнях. Защита собственно данных возложена на верхние уровни, если это необходимо. Как мы уже отмечали, многие приложения реального времени - передача видео-, аудиоданных - более критичны к времени передачи, чем к степени искажения отдельных ячеек. Поскольку контрольная сумма покрывает только заголовок, то этот байт так и называется - HEC (Header Error Control - контроль ошибки в заголовке).

Другим важным фактором, повлиявшим на выбор этой схемы контрольной суммы, было то, что основной средой для ATM является оптоволокно. Исследования, выполненные компанией AT&T, показали, что оптоволокно - высоконадежная среда и единичные ошибки происходят в ней с вероятностью менее 1%. Схема HEC прекрасно справляется как с однобитными ошибками, так и множественными.

Для надежной передачи ячеек была предложена схема, когда две последовательные ячейки объединяются через EXCLUSIVE OR, после чего получается новая ячейка, которая добавляется в последовательность после первых двух. В результате если хоть одна ячейка была принята с ошибкой или потеряна, то она легко может быть восстановлена.

После того как HEC вычислен и добавлен в заголовок, ячейка готова к передаче. Среда передачи может быть двух категорий - синхронной и асинхронной. В асинхронной среде ячейка посылается сразу, как только она готова к передаче. В синхронной среде ячейка передается в соответствии с временными соглашениями. Если нет ячейки для передачи, то TC-подуровень должен сгенерировать специальную ячейку ожидания.

Другой вид служебных ячеек - OAM (Operation And Maintenance). Эти ячейки используются ATM-переключателями для проверки работоспособности системы.

Ячейки ожидания обрабатываются соответствующим TC-подуровнем, а OAM-ячейки передаются на ATM-уровень.

Другой важной функцией TC подуровня является генерирование ячеек в формате физической среды передачи. Это значит, что TC-подуровень генерирует обычную ATC-ячейку и упаковывает ее в кадр надлежащей среды передачи.

Итак, на выходе ТС-подуровень формирует HEC-заголовок, преобразует ячейку в кадр, формирует ATM-ячейки и передает поток битов на физический уровень. На противоположном конце ТС-подуровень производит те же самые действия, но в обратном порядке: разбивает поток бит на кадры, выделяет ячейки, проверяет HEC-заголовки и передает ячейки на ATM-уровень.

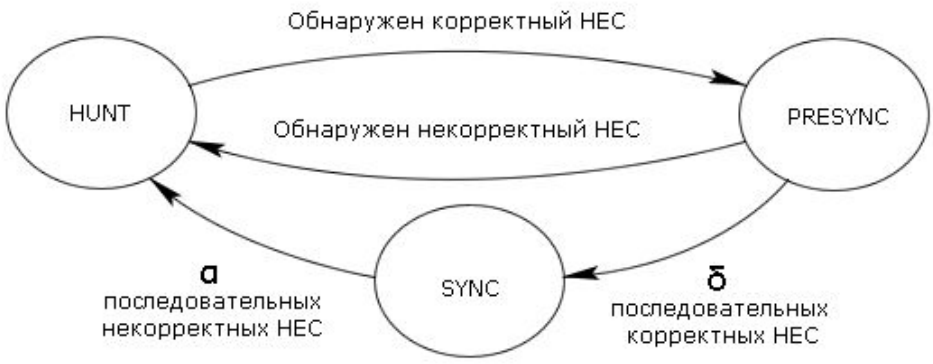
Самое трудное - выделить кадр из потока битов. На уровне битов ячейка - это 53x8 = 424 бита. Нет маркеров ни начала, ни конца кадра. Как определить границы кадра?

На ТС-подуровне есть сдвиговый регистр на 40 бит. Если в этих 40 бит правые 8 представляют собой HEC, то последующие 32 левых бита - заголовок ячейки. Если условие не выполнено, то все сдвигается на один бит и проверка повторяется. Этот процесс продолжается до тех пор, пока не будет обнаружен HEC.

Схема распознавания в том виде, как она описана не надежна. Вероятность того, что случайный байт будет выглядеть как HEC, равна 1/256. Чтобы исправить эту схему, используют автомат, схема состояний которого изображена на рисунке 3-24. Есть три состояния: HUNT, PRESYNCH, SYNCH. В состоянии HUNT ищется HEC. Как только найден похожий байт, автомат переходит в состояние PRESYNCH и отчитывает следующие 53 байта. Если предположение о том, что найденный HEC - начало ячейки, то сдвиг на 53 байта приведет к следующему HEC. Происходит проверка последовательно δ ячеек, после этого происходит переход в состояние SYNCH.

Если в состоянии SYNCH α последовательных ячеек оказались плохими, происходит переход в состояние HUNT.

Рисунок 3-24. Процедура поиска ячеек



Билет № 25.

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).

ALOHA.

В 70-х годах Норман Абрамсон (Norman Abramson) со своими коллегами из университета Гавайи предложил простой способ распределения доступа к каналу. Абрамсон назвал систему, реализующую этот способ распределения канала, ALOHA, что по-гавайски означает что-то вроде «привет». Система состояла из наземных радиостанций, связывающих острова между собой. Идея была позволить в вещательной среде любому количеству пользователей неконтролируемо использовать один и тот же канал.

Мы здесь рассмотрим два варианта системы: чистая ALOHA и слотированная ALOHA, т.е. разбитая на слоты. Основное различие - в первом случае никакой синхронизации пользователей не требуется, во втором она нужна.

Чистая ALOHA.

Идея чистой ALOHA проста - любой пользователь, желающий передать сообщение, сразу пытается это сделать. Благодаря тому, что в вещательной среде он всегда имеет обратную связь, т.е. может определить, пытался ли кто-то еще передавать на его частоте, то он может установить возникновение конфликта при передаче. Такая обратная связь в среде LAN происходит практически мгновенно, в системах спутниковой связи задержка составляет около 270 мсек. Обнаружив конфликт, пользователь ожидает некоторый случайный отрезок времени, после чего повторяет попытку. Интервал времени на ожидание должен быть случайным, иначе конкуренты будут повторять попытки в одно и то же время, что приведет к их блокировке. Системы подобного типа, где пользователи конкурируют за получение доступа к общему каналу, называются системами с состязаниями.

Не важно, когда произошел конфликт: когда первый бит одного кадра «наехал» на последний бит другого кадра или как-то иначе, оба кадра считаются испорченными и должны быть переданы повторно. Контрольная сумма, защищающая данные в кадре, не позволяет различать разные случаи наложения кадров.

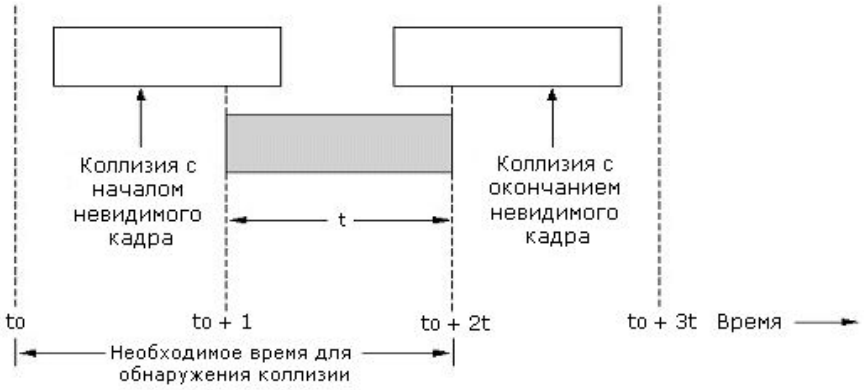
Какова эффективность системы ALOHA, измеренная в количестве кадров, которые избежали коллизий? Для ответа на этот вопрос рассмотрим следующую модель. Есть неограниченное число пользователей, работающих на компьютерах. Все что они могут делать, - это либо набирать текст, либо ждать, пока набранный текст будет передан. Когда пользователь заканчивает набирать очередную строку, он останавливается и ждет ответа от системы. Система пытается передать эту строку. Когда она сделает это, пользователь видит отклик и может продолжать работу.

Назовем временем кадра время, необходимое на передачу кадра стандартной фиксированной длины. Предполагаем, что число пользователей неограниченно, и они порождают кадры по закону Пуассона со средним S кадров за время кадра. Поскольку при $S>1$ очередь на передачу будет только расти и все кадры будут страдать от коллизий, то мы будем предполагать $0<S<1$.

Также будем предполагать, что вероятность за время кадра сделать k попыток передачи распределена по закону Пуассона со средним G. Понятно, что должно быть GiS, иначе очередь будет расти бесконечно. При слабой загрузке ($S»0$) будет мало передач, а следовательно и коллизий, поэтому допустимо $G»S$. При высокой загрузке должно быть $G>S$. При любой нагрузке пропускная способность это - число кадров, которые надо передать, умноженное на вероятность успешной передачи. Если обозначить P0 вероятность отсутствия коллизий при передаче кадра, то $S=GP0$.

Рассмотрим внимательно, сколько времени нужно отправителю, чтобы обнаружить коллизию. Пусть он начал передачу в момент времени t_0 и пусть требуется время t, чтобы кадр достиг самой отдаленной станции. Тогда, если в тот момент, когда кадр почти достиг этой отдаленной станции, она начнет передачу (ведь в системе ALOHA станция сначала передает, а потом слушает), то отправитель узнает об этом только через t_0+2t (рисунок 4-1).

Рисунок 4-1. Время, требуемое на обнаружение коллизии



Вероятность появления k кадров при передаче кадра при распределении Пуассона равна

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

поэтому вероятность, что появится 0 кадров, равна e^{-G}.

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

$P_0=e^{-2G}$

а так как $S=GP_0$, то

$S=Ge^{-2G}$

Зависимость между нагрузкой и пропускной способностью показана на рисунке 4-3. Максимальная пропускная способность достигается при G=0,5 при S=1/2e, что составляет примерно 18% от номинальной пропускной способности. Это означает, что если мы будем генерировать кадры с большей скоростью, чем 18% от скорости канала, то очереди переполнятся и система «захлебнется». Результат не очень вдохновляющий, но это плата за удобство: каждый передает, когда захочет.

Слотированная ALOHA.

В 1972 году Робертс (Roberts) предложил модификацию чистой ALOHA. Все время работы канала разделяют на слоты. Размер слота определяют так, чтобы он был равен максимальному времени кадра. Ясно, что такая организация работы канала требует синхронизации. Кто-то, например, одна из станций испускает сигнал начала очередного слота. Поскольку передачу теперь можно начинать не в любой момент, а только по специальному сигналу, то время на обнаружение коллизии сокращается вдвое. Отсюда

$S=Ge^{-G}$

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

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

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

Среднее ожидаемое число повторных передач будет равно

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

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

Протоколы множественного доступа с обнаружением несущей.

В локальных сетях есть возможность определить, что делают другие станции, и только после этого решать, что делать самому. Протоколы, которые реализуют именно эту идею – сначала определить, занят канал или нет и только после этого действовать - называются протоколами с обнаружением несущей CSMA (Carrier Sensitive Multiple Access).

Настойчивые и ненастойчивые CSMA-протоколы.

Согласно протоколу, который мы сейчас рассмотрим, станция, прежде чем что-либо передавать, определяет состояние канала. Если канал занят, то она ждет. Как только канал освободился, она пытается начать передачу. Если при этом произошла коллизия, она ожидает случайный интервал времени и все начинает с начала. Этот протокол называется настойчивым CSMA-протоколом первого уровня или 1-настойчивым CSMA-протоколом, потому что станция, следуя этому протоколу, начинает передачу с вероятностью 1, как только обнаруживает, что канал свободен.

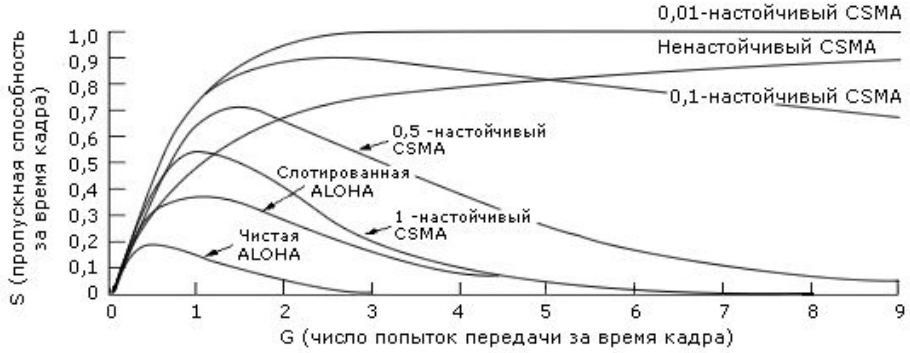
Здесь важную роль играет задержка распространения сигнала в канале. Всегда есть шанс, что, как только одна станция начала передачу, другая также стала готовой передавать. Если вторая станция проверит состояние канала прежде, чем до нее дойдет сигнал от первой о том, что она заняла канал, то вторая станция сочтет канал свободным и начнет передачу. В результате - коллизия. Чем больше время задержки, тем больше вероятность такого случая, тем хуже производительность канала.

Однако даже если время задержки будет равно 0, коллизии все равно могут возникать. Например, если готовыми передавать оказались две станции, пока одна станция продолжает передавать. Они вежливо подождут, пока первая закончит передачу, а потом будут состязаться между собой. Тем не менее, этот протокол более эффективен, чем любая из систем ALOHA, так как станция учитывает состояние канала, прежде чем начать действовать.

Другой вариант CSMA-протокола - ненастойчивый CSMA-протокол. Основное отличие его от предыдущего в том, что готовая к передаче станция опрашивает канал. Если он свободен, то она начинает передачу. Если он занят, то она не будет настойчиво его опрашивать в ожидании, когда он освободится, а будет делать это через случайные отрезки времени. Это несколько увеличивает задержку при передаче, но общая эффективность протокола возрастает.

И, наконец, настойчивый CSMA-протокол уровня p. Он применяется к слотированным каналам. Когда станция готова к передаче, она опрашивает канал. Если он свободен, то она с вероятностью p передает свой кадр и с вероятностью q=1-p ждет следующего слота. Так она действует, пока не передаст кадр. Если произошла коллизия во время передачи, она ожидает случайный интервал времени и опрашивает канал опять. Если при опросе канала он оказался занят, станция ждет начала следующего слота, и весь алгоритм повторяется. На рисунке 4-3 показана пропускная способность этого протокола в зависимости от нагрузки.

Рисунок 4-3. Пропускная способность настойчивого CSMA-протокола уровня p по сравнению с другими

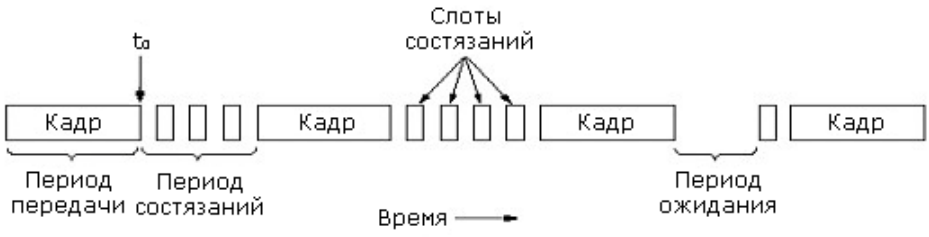


CSMA-протокол с обнаружением коллизий.

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

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

Рисунок 4-4. Стадии работы протокола CSMA/CD



Рассмотрим подробнее алгоритм состязаний. Сколько времени станции, начавшей передачу, нужно, чтобы определить коллизию? Обозначим через τ время распространения сигнала до самой удаленной станции на линии. Для коаксиала в 1 км τ=5 мксек., в таком случае минимальное время для определения коллизии будет равно 2τ. Поэтому станция не может быть уверена, что она захватила канал до тех пор, пока не убедится, что в течение 2τ секунд не было коллизий. Поэтому мы будем рассматривать период состязаний как слотированную систему ALOHA со слотом 2τ секунд на один бит. Захватив канал, станция может далее передавать кадр с любой скоростью.

Стоит подчеркнуть, что обнаружение коллизий – это аналоговый процесс. Поэтому, чтобы обнаруживать их, нужно использовать специальные кодировки на физическом уровне. Надо также отметить, что MAC-подуровень обеспечивает надежную передачу, используя специальные приемы кодирования данных. Примеры таких кодировок мы рассматривали в гл. 2 (см. Манчестерские коды).

Билет № 26.

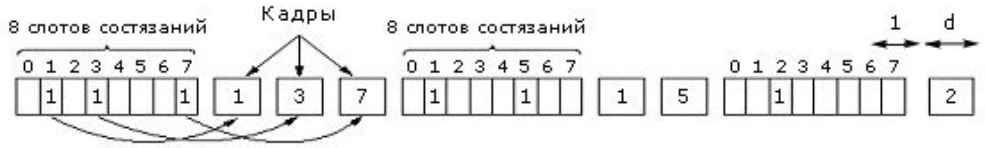
Протоколы множественного доступа к каналу: Бесконфликтные протоколы (Bit-Map протокол, Адресный счетчик). Протоколы с ограниченным числом конфликтов. Протоколы с множественным доступом и разделением частот

Будем предполагать, что у нас есть N станций с адресами от 0 до N-1. Все адреса уникальны. Основным является вопрос: как определить, кто будет владеть каналом, когда закончится текущая передача?

Bit-Map-протокол.

Идея этого метода показана на рисунке 4-5. Выделяют специальный период состязаний, где количество слотов равно числу станций. Каждая станция, имеющая кадр для передачи, проставляет 1 в свой слот. Поскольку мы рассматриваем канал с множественным доступом (т.е. все видят, что проходит в канале), то в конце состязаний все станции знают, кто будет передавать кадры и в каком порядке. Передача происходит в том же порядке, в каком пронумерованы слоты. Раз станции знают, кто будет передавать и в каком порядке, то конфликтов не будет. Если станция опоздала с заявкой на передачу, то она должна ждать следующего периода состязаний, который начнется по окончании передач, заявленных на предыдущем периоде состязаний. Такие протоколы, когда заявки на передачу откладываются и могут быть сделаны лишь в определенные периоды времени, называются протоколами с резервированием.

Рисунок 4-5. Bit-Map-протокол



Теперь рассмотрим производительность этого метода. Будем для удобства измерять время в количестве слотов состязаний. Будем также предполагать, что передача одного кадра будет занимать ровно d таких слотов. Для станции с небольшим номером, например, 0 или 1, время ожидания на передачу в среднем будет равняться 1,5 N, потому что она, пропустив начало состязаний, будет ждать 0,5 N в первом периоде состязаний и N единиц времени во втором. Другая участь у станций со старшими номерами. Эти станции будут ожидать в среднем N/2 слотов до начала передачи. Таким образом, в среднем любая станция должна будет ждать N слотов до передачи.

При небольшой нагрузке накладные расходы на передачу одного кадра будут N бит, а эффективность передачи одного кадра - d/(d+N), где N - накладные расходы на передачу кадра. При плотной загрузке, когда практически каждая станция каждый раз что-то посылает, накладные расходы будут 1 бит на кадр, т.е. d/(d+1). Средняя задержка кадра будет равна средней задержке кадра внутри очереди в станции плюс N(d+1)/2 слотов ожидания, когда кадр достигнет заголовка очереди. Отсюда видно, что с ростом N, хотя накладные расходы на передачу одного кадра и падают, задержка кадра в канале существенно возрастает, и эффективность падает. Следует также отметить, что если d и N - сопоставимые величины, то значительную часть пропускной способности канала мы будем тратить на состязания.

Один из недостатков bit-map протокола - затраты в 1 бит на кадр. При коротких кадрах это накладно. Есть другая возможность, позволяющая повысить эффективность использования канала. Она основана на двоичном представлении адреса станции.

В этом методе каждая станция, готовая к передаче, выставляет свой адрес бит за битом, начиная со старшего разряда. Эти разряды подвергаются логическому сложению. Если станция выставила на очередном шаге 0, а результат логического сложения - 1, то она должна ждать и в текущих состязаниях участия не принимает. Эффективность использования канала в этом методе - d/(d+lnN). Если структура заголовка кадра была выбрана так, чтобы его можно было использовать для выбора очередной станции для передачи, то lnN битов также будет использовано, тем самым эффективность использования канала достигнет 100%.

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

Протоколы с ограниченным числом конфликтов.

Рассмотренные нами только что протоколы показывают, что при небольшой загрузке конфликты не опасны ввиду небольшой задержки на передачу. По мере роста нагрузки они снижают эффективность использования канала. Поэтому при высокой загруженности канала арбитраж желателен и протоколы без коллизий предпочтительнее. А вот при низкой загрузке они лишь вызывают дополнительные накладные расходы.

Естественно попытаться создать протокол, объединяющий достоинства этих двух групп методов, т.е. использовать состязания при небольших нагрузках и бесконфликтные методы - при высоких. Такие протоколы существуют и называются протоколами с ограниченным числом конфликтов. Их изучением мы и закончим рассмотрение класса протоколов с контролем несущей.

До сих пор мы рассматривали протоколы с состязаниями только в так называемой симметричной конфигурации: все станции, пытающиеся передать кадр, получали канал с одной и той же для всех вероятностью p. Оказывается, общая производительность системы может быть улучшена, если разным станциям будет сопоставлена разная вероятность.

Рассмотрим производительность в случае симметричного случая. Пусть у нас есть k станций, каждая из которых с вероятностью p готова передать кадр. Тогда вероятность, что какая-то станция успешно передаст свой кадр, равна $kp(1-p)^{k-1}$. Эта вероятность достигает максимума при p=1/k. Тогда вероятность передать сообщение какой-либо станцией равна

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

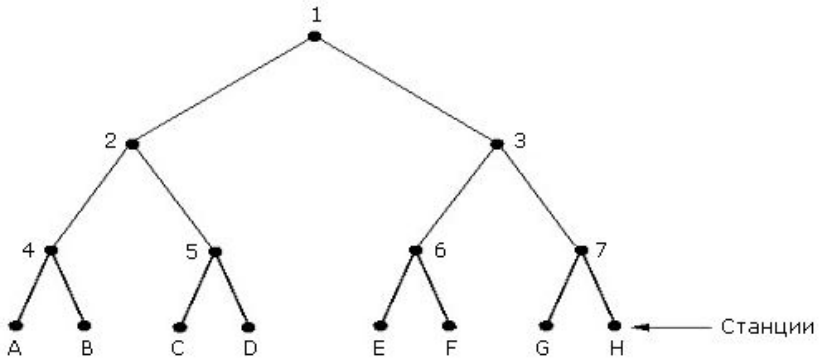
При небольшом числе станций шансы передать кадр достаточно велики, но с ростом числа станций эти шансы резко падают. Единственным способом увеличить шансы на передачу является сократить конфликты. Для этого в протоколах с ограниченным числом конфликтов все станции разбивают на непересекающиеся группы.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

Адаптивный древовидный протокол устроен по принципу тестирования солдат американской армии на сифилис во второй мировой войне. У n солдат брали кровь на анализ. В первой пробе в общей пробирке смешивали часть крови каждого солдата. Если тест давал отрицательный результат,

то все n считались здоровыми. Если тест давал положительную реакцию, то в пробирке смешивали только кровь первой половины солдат и опять тестировали. Если был положительный результат, то эту половину делили опять пополам и т.д., пока не обнаруживали носителя.

На рисунке 4-8 показано, как эта процедура применяется к станциям. Станции - листья. За слот 0 борются все станции. Если какая-то победила - хорошо. Если нет, то за слот 1 борются только станции поддерева с корнем в вершине 2. Если какая-то победила, то следующий слот резервируется для станций поддерева 3. Если был конфликт, то за следующий слот борются станции поддерева 4, и т.д.

Рисунок 4-8. Дерево для восьми станций



Когда число станций велико и все они готовы передавать, то вряд ли целесообразно начинать поиск с уровня 0 в дереве. Возникает вопрос: с какого уровня надо начинать эту процедуру при заданном числе станций? Пусть число станций, готовых к передаче, нормально распределено. Обозначим это число через q. Тогда число станций, готовых к передаче и расположенных ниже уровня i, будет ровно $2^{-i}q$. Заметим, что их доля от общего числа станций, расположенных в дереве ниже уровня i, равна 2^{-i}. Естественно, надо подобрать такое соотношение между i и q, когда количество конкурирующих станций будет 1, т.е. $2^{-i}q=1$, или $\log_2 q=i$.

У этого алгоритма есть много вариантов. Мы здесь описали лишь основную идею.

Протоколы с множественным доступом и разделением частот.

Иной подход к распределению доступа к каналу основан на разделении канала на подканалы, используя FDM-, TDM-метод или сразу оба этих метода.

Здесь мы рассмотрим работу протоколов множественного доступа для оптоволоконных систем. Они построены на идее разделения частот. Вся полоса разделяется на каналы по два на станцию. Один, узкий, - управляющий канал, второй, широкий, - для передачи данных. Каждый канал разбит на слоты. Все слоты синхронизируются от единых часов. Отмечается только нулевой слот, чтобы можно было определить начало каждого цикла.

Обозначим через m число слотов в управляющем канале и через n+1 - в канале данных. Из них n слотов - для данных, а последний - для сообщения о статусе канала. Протокол поддерживает три класса трафика:

1. Постоянный с соединением (видео)

2. Переменный с соединением (передача файлов)

3. Дейтаграммный (типа UDP)

У каждой станции есть два ресивера и два трансивера:

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

Другими словами, каждая станция постоянно слушает свой управляющий канал, но должна настраиваться при приеме на волну передающей станции. Рассмотрим, как станция A устанавливает соединение класса 2 со станцией B для передачи файла. A настраивается на управляющий канал станции B, чтобы определить, какие слоты уже заняты, а какие свободны. На рисунке 4-9 видно, что у B из 8 управляющих слотов свободны 0, 4 и 5.

Рисунок 4-9. Множественный доступ с разделением частот

A выбирает, например, 4-й слот и помещает туда свой CONNECTION REQUEST. Станция B видит этот запрос и закрепляет слот 4 за станцией A, о чем сообщает ей через статусный слот. Для станции A это означает, что установлено однонаправленное соединение от A к B. Если нужно двунаправленное соединение, то B должна повторить все, что сделала A. Если в момент попытки A захватить слот у B другая станция, например, C, также попытается это сделать, возникнет конфликт, о котором и A, и C узнают через статусный слот управляющего канала.

После того как соединение установлено, A посылает B управляющее сообщение типа: «Жди. В слоте 3 канала данных есть кадр». Получив такое сообщение, B настраивается на волну канала A и считывает кадр. Таким образом, мы имеем бесконфликтный канал. Хотя может случиться, что если A и C имеют соединение с B и оба скажут «смотри на слот 3, там кадр от меня», то от какого из двух получит сообщение B, сказать заранее нельзя.

В случае дейтаграмм A шлет не запрос на соединение, а сообщение типа: «В слоте 3 для тебя есть кадр». Существует несколько вариантов этого WDMA-протокола.

Билет № 27.

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

Развитая мобильная телефонная система – AMPS.

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

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

При перемещении телефона ближайшие базовые станции сравнивают уровень сигнала от него и та база, у которой этот уровень выше, чем у других, берет его под свой контроль. Телефон получает сообщение об изменении базы.

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

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

В системе AMPS у каждого телефона есть встроенный 32-битовый серийный номер и телефонный номер, состоящий из 10 цифр: 3 цифры – код зоны (10 бит) и 7 цифр (24 бита) – номер абонента. Когда телефон включают, он начинает сканировать запрограммированный в нем список из 21 каналов управления, чтобы обнаружить наиболее мощный сигнал. По информации из управляющего канала он узнает распределение каналов для сообщений, установки соединений и доступа, передачи данных.

Затем телефон сообщает свой 32-битовый серийный номер и 34-битовый телефонный номер. Эта информация в AMPS-системе передается пакетом в цифровом виде несколько раз, кодируется специальным кодом с коррекцией ошибок, хотя голос передают по аналоговому каналу.

Когда базовая станция получает такой пакет от телефона, она запрашивает у своего MSC-центра информацию о новом клиенте и сообщает домашней MSC, т.е. MSC, к которой приписан этот телефон, о его текущем местоположении. Обычно такая перерегистрация телефона происходит каждые 15 минут.

Чтобы позвонить, абонент включает телефон, набирает номер нужного абонента и нажимает кнопку «Послать» (Send). Телефон по каналу установки доступа посылает в цифровом виде пакет, содержащий информацию о нем и о телефоне вызываемого абонента. Если происходит коллизия или ошибка, то попытка повторяется несколько раз. Получив запрос, базовая станция информирует о нем MSC. Если нужный абонент – это абонент компании, которой принадлежит MSC, то MSC ищет свободный канал для данных. Если такой найден, то MSC информирует о нем вызывающий телефон по каналу управления. Вызывающий телефон переключается на прием по указанному каналу и ждет, когда на вызываемом телефоне поднимут трубку (нажмут кнопку «Прием»).

Входящий звонок обрабатывается несколько иначе. В режиме ожидания телефон постоянно следит за каналом сообщений: не появится ли там сообщение для него. Когда вызывающий телефон сгенерировал запрос, то от MSC поступает запрос на домашнюю MSC вызываемого телефона, чтобы определить, в какой соте находится вызываемый телефон. Пакет с вызовом направляется последней базовой станции, зарегистрировавшей телефон с искомым номером, например, 46. Базовая станция распространяет по каналу сообщений специальное сообщение типа: «46-й, ты здесь?» Вызываемый телефон отвечает по каналу управления специальным пакетом типа «Да». Тогда базовая станция шлет по каналу управления пакет «46-ой, для вас вызов на канале 8». После этого вызываемый телефон переключается на канал 8 и начинает звонить.

К сожалению, аналоговые сотовые телефоны абсолютно не защищены. Любой, у кого есть радиоприемник нужного диапазона, может, настроив его на один из голосовых каналов, просто прослушать разговор. Злоумышленник может перехватывать информацию из каналов управления, содержащую 32-битовые номера телефонных трубок и 34-битовые номера, а затем разговаривать за чужой счет. И многое, многое другое. Это один из главных недостатков аналоговых сотовых телефонов.

GSM – Глобальная система для мобильной связи.

Первые сотовые телефонные системы были аналоговыми. Им на смену пришли цифровые системы, которые составили второе поколение сотовых систем. В настоящее время происходит переход на сотовые системы 3G – системы третьего поколения.

В 80-е годы в Европе существовало пять разных сотовых аналоговых телефонных систем. Поэтому, переезжая из страны в страну, пользователи были вынуждены менять и телефонные аппараты. Ясно, что это было чрезвычайно неудобно. Как результат, европейцы создали единую цифровую систему, известную как GSM (Global System for Mobile communications), которая была введена в действие ранее американских и японских аналогов.

Итак, GSM - это полностью цифровая система. Ее успех был во многом связан с тем, что она проектировалась без оглядки на уже существующие аналоговые системы, ее авторы не пытались сделать ее совместимой с ними.

Основная цель стандарта GSM была обеспечить людям возможность, свободно передвигаясь, как внутри страны, так и между странами, поддерживать связь с любыми абонентами сети. При этом в каждой стране может быть одна или несколько функционирующих сетей. Каждая такая сеть называется Региональной мобильной сетью оператора (PLMN). Зона действия каждой PLMN-сети ограничена национальными границами, в одной стране, впрочем, может быть несколько PLMN-сетей.

GSM-пользователь заключает контракт с одной из PLMN-сетей, называемой домашней. В этом контракте указаны услуги, доступные этому пользователю. При желании во время работы пользователь может выбрать другую PLMN-сеть, если ему доступны ее услуги. Терминал пользователя (в GSM его называют мобильной станцией – MS) обеспечивает пользователю такой выбор и показывает список доступных PLMN-сетей. Выбор из этого списка пользователь может сделать сам явно, или MS-терминал сделает это автоматически с помощью заложенного в нее программного обеспечения.

Как и в AMPS-системе, в GSM территория разбивается на области, обслуживаемые Центром Мобильной Коммутации (MSC). Оператор PLMN-сети абсолютно свободен в разбиении области действия MSC-станции на соты. У каждой PLMN-сети есть логически единая база данных, называемая Home Location Registers (HLR), где хранится информация обо всех пользователях, для которых эта PLMN-сеть домашняя. Физически HLR-база может быть распределенной. У каждой MSC-станции есть база данных визитеров – Visitor Location Registers (VLR). Одна VLR-база обычно обслуживает одну MSC-станцию, но может обслуживать и несколько. HLR- и VLR-базы данных обеспечивают отслеживание текущего местонахождения каждого MS-терминала, находящегося в зоне действия MSC-станции, запрашиваемых услуг и т.д.

Мобильная станция GSM, в просторечии «трубка», разделяется на две части. Одна обеспечивает радиоинтерфейс, другая - интерфейс с базами HLR и VLR и содержит информацию, идентифицирующую пользователя (Subscriber Identify Module - SIM). SIM-карта идентифицирует пользователя, а не MS-терминал. Поэтому она может быть вынута из одного MS-терминала и вставлена в другой. Каждая SIM-карта уникальна в системе GSM и связана с идентификатором IMSI (International Mobil System Identify). На этой карте хранится идентификационная информация, список услуг, список выбираемых PLMN-сетей и т.п. Она защищена паролем (PIN – Personal Identification Number). Вставив свою SIM-карту в трубку, пользователь тем самым персонифицирует ее. Благодаря SIM-карте поддерживается роуминг, т.е. доступ к услугам связи в чужую PLMN-сеть.

Теперь рассмотрим, как в GSM отслеживаются перемещения пользователей. Когда MS-терминал входит в новую область регистрации, информация о нем заносится в VLR-базу, и он получает TMSI-идентификатор – Temporary Mobil Subscriber Identify. TMSI идентификатор короче IMSI-идентификатора, и именно он передается при взаимодействии MS-терминала и VLR-базы. TMSI-идентификатор действует только в зоне MSC-станции, ассоциированной с VLR-базой, выдавшей его. Идентификаторы IMSI и TMSI – это внутренние идентификаторы системы, связанные с SIM-картой. Для соединения с абонентом используется телефонный номер, который в GSM называется Mobil Subscriber Integrated Service Digital Network Number (MSISDNM).

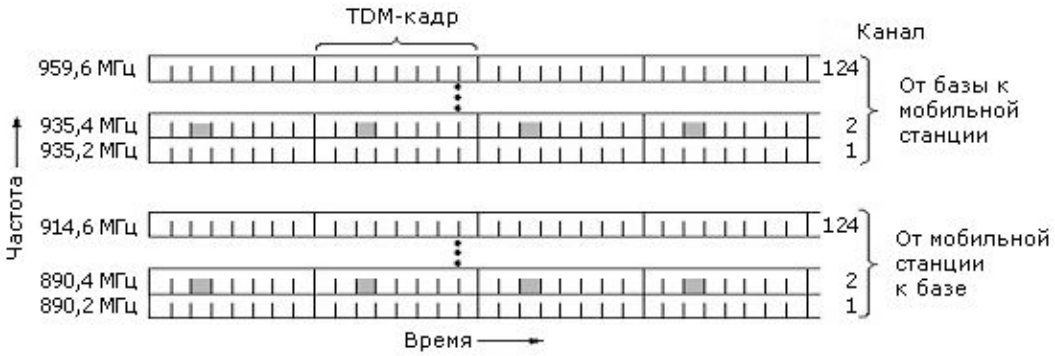
MS-терминал всегда может определить, находится ли он в старой или новой области регистрации. Это происходит благодаря периодически рассылаемой BS-станцией информации внутри обслуживаемой ею соты. Если MS-терминал обнаруживает, что он оказался в новой области, то он инициирует запрос на обновление регистрации, в котором он сообщает идентификатор предыдущей области и TMSI-идентификатор, который терминал там получил. Этот запрос BS-станция передает в MSC-центр, который, в свою очередь, передает его в VLR-базу. Эта база, назовем ее новая, инициирует запрос к старой VLR-базе с просьбой предоставить IMSI-идентификатор терминала, соответствующий указанному TMSI-идентификатору. Получив от старой VLR-базы необходимую информацию, новая VLR-база начинает процедуру идентификации MS-терминала по информации, полученной от старой. Если процедура идентификации прошла успешно, то новая VLR-база, используя IMSI-идентификатор терминала, определяет адрес его HLR-базы.

Эта процедура весьма близка к аналогичной процедуре в AMPS-системе (стандарт IS-41). Основное ее отличие от ее AMPS-аналога состоит в усилении информационной безопасности. Так, например, идентификация пользователя и доступных ему услуг происходит на основе информации, получаемой новой VLR-базой, как от старой VLR-базы, так от HLR-базы идентифицируемого MS-терминала, а не только от HLR-базы, как в AMPS-системе. Процедура установления соединения в GSM-системе аналогична процедуре установления соединения в AMPS-системе. Стандарт GSM занимает более 5000 страниц, и здесь мы приводим лишь самое общее его описание.

В большинстве стран GSM использует частоты 900 МГц и 1800 МГц. В США из-за особенностей национального распределения частот используется другой диапазон. В каждой GSM-соте может быть максимально до 200 полнодуплексных каналов, из которых 124 в работе, остальные в резерве и для служебных целей. Каждый канал поддерживает связь как от MS-терминала к BS-станции (MS-BS), так и от BS-станции к MS-терминалу (BS-MS). Ширина полосы в каждом направлении - 200 КГц.

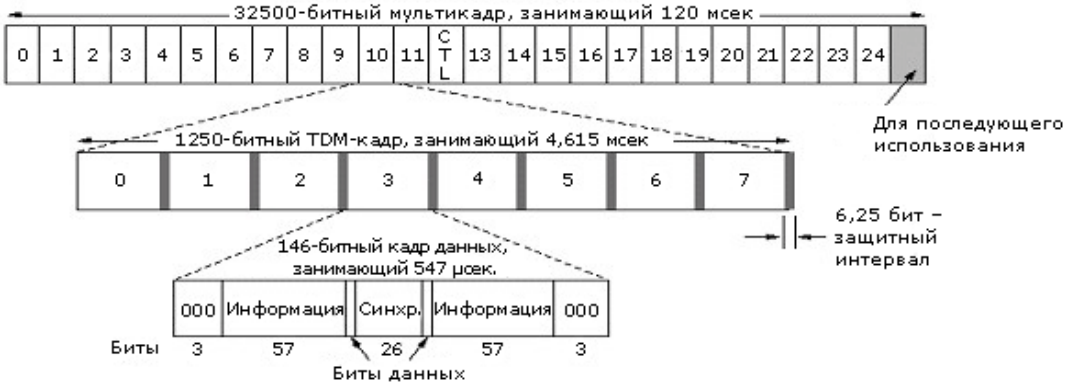
Каждый канал поддерживает 8 разных соединений с помощью мультиплексирования с разделением по времени (TDM-метод). Каждому MS-терминалу выделяется один временной слот на одном из каналов.

Рисунок 4-12. Устройство GSM-каналов



На рисунке 4-13 показана иерархия кадров в GSM, которая имеет достаточно сложную структуру. Каждый TDM-слот состоит из 8 кадров данных по 148 бит каждый. Один 1250-битовый TDM-кадр занимает 4,615 мсек. TDM-кадры объединяются в 26-слотовый мультикадр, который занимает 120 мсек. Кроме этого, есть 51-слотовый мультикадр, который не показан на рисунке и который используется для нескольких каналов управления на системном уровне. Например, таких, как канал управления сотой – по этому каналу передается информация для поддержки базы данных соты, канал общего управления, который отвечает за регистрацию новых мобильных терминалов, поступивших в зону соты, выделение слотов для таких терминалов и многое другое.

Рисунок 4-13. Иерархия кадров в GSM



GPRS-служба.

Вполне естественно возникновение идеи по применению GSM-сетей для организации связи между компьютерами. Одним из существенных недостатков сетей сотовой связи стандарта GSM на сегодняшний день является низкая скорость передачи данных (максимум 9,6 кбит/сек.) по одному каналу. Для передачи данных абоненту выделяется всего один голосовой канал, а оплата осуществляется, исходя из времени соединения (причем по тарифам, мало отличающимся от голосовых).

Для высокоскоростной передачи данных посредством существующих GSM-сетей была разработана GPRS (General Packet Radio Service) - служба пакетной передачи данных по радиоканалу. Необходимо отметить, что, кроме повышения скорости (максимум составляет 171,2 кбит/сек.), новая система предполагает иную схему оплаты услуги передачи данных - при использовании GPRS-службы расчеты производятся пропорционально объему переданной информации, а не времени использования канала. К тому же GPRS-служба более рационально использует выделяемую полосу частот: особо не вдаваясь в технические тонкости, можно сказать, что пакеты данных передают одновременно по многим каналам (именно в одновременном использовании нескольких каналов и

заключается выигрыш в скорости) в паузах между передачей речи. И только в паузах голосовой трафик имеет безусловный приоритет перед данными, поэтому скорость передачи информации определяется не только возможностями сетевого и абонентского оборудования, но и загрузкой сети. Ни один канал GPRS-службы не занимают под передачу данных целиком - и это основное качественное отличие новой технологии от описанных выше.

Доработку GSM-сети для предоставления GPRS-услуг можно условно разделить на два аспекта - программный и аппаратный. Если говорить о программном обеспечении, то оно нуждается в замене или обновлении практически всюду - начиная с баз HLR-VLR и заканчивая базовыми станциями BS. В частности, вводится режим многопользовательского доступа к временным кадрам каналов GSM, а в HLR-базе, например, появляется новый параметр - Mobile Station Multislot Capability (количество каналов, с которыми одновременно может работать мобильный телефон абонента, подробнее об этом ниже).

Ядро системы GPRS (GPRS Core Network) состоит (рисунок 2-73) из двух основных блоков - SGSN-узел (Serving GPRS Support Node - узел поддержки GPRS-сервиса) и GGPRS-узел (Gateway GPRS Support Node - шлюзовой узел GPRS). Остановимся на их функциях более подробно.

SGSN, в некотором смысле, можно назвать аналогом MSC – центра мобильной коммутации сети GSM. SGSN контролирует доставку пакетов данных пользователям, взаимодействует с HLR-базой собственных абонентов сети, проверяя, разрешены ли запрашиваемые пользователями услуги, ведет мониторинг находящихся онлайн пользователей, организует регистрацию абонентов вновь появившихся в зоне действия сети, и т.п. Так же как и MSC-центр, SGSN-узел в системе может быть не один, в этом случае каждый узел отвечает за свой участок сети. Например, SGSN-узел производства компании Motorola имеет следующие характеристики: каждый узел поддерживает передачу до 2000 пакетов в секунду, одновременно контролирует до 10000 находящихся онлайн пользователей. Всего же в системе может быть до 18 SGSN-узлов производства Motorola.

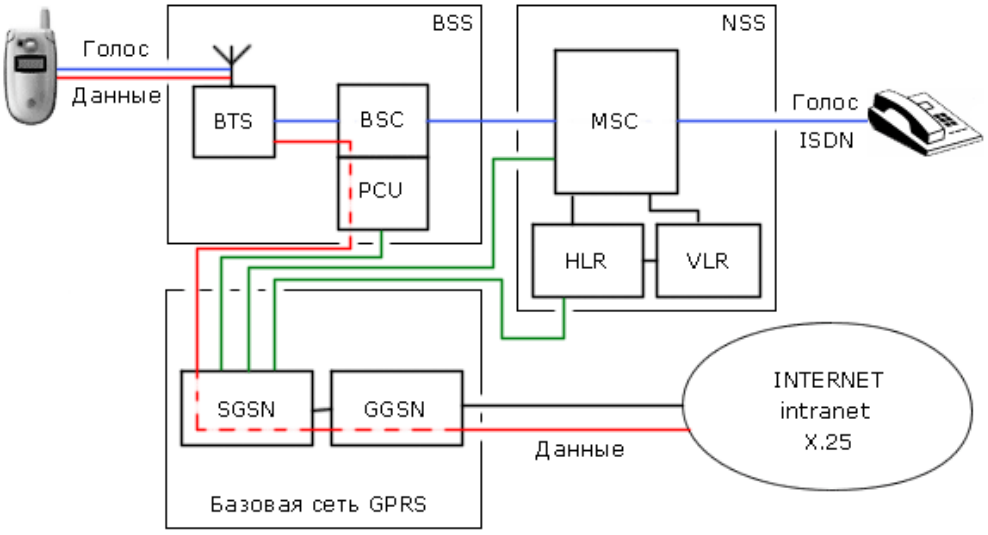
Назначение GGSN-узла видно из его названия - это шлюз между сотовой сетью (вернее, ее частью для передачи данных через GPRS-службу) и внешними информационными магистралями (Интернетом, корпоративными интранет-сетями, другими GPRS-системами и так далее). Основной задачей GGSN-узла является маршрутизация данных, идущих к абоненту через SGSN-узел и от него. Вторичными функциями GGSN-узла является адресация данных, динамическая выдача адресов в Интернет (IP-адресов), а также отслеживание информации о внешних сетях и собственных абонентах (в том числе тарификация услуг). Эти функции относятся к сетевому уровню модели OSI ISO.

В GPRS-службу заложена хорошая масштабируемость: при появлении новых абонентов оператор может увеличивать число SGSN-узлов, а при эскалации суммарного трафика - добавлять в систему новые GGSN-узлы. Внутри ядра GPRS-службы (между SGSN- и GGSN-узлами) данные передаются с помощью специального туннельного протокола GTP (GPRS Tunneling Protocol).

Еще одной составной частью системы GPRS является PCU-блок (Packet Control Unit - устройство контроля пакетной передачи). PCU-блок стыкуется с контроллером базовых станций BSC и отвечает за направление трафика данных непосредственно от BSC к SGSN.

В перспективе (при ориентации системы на мобильный Интернет) возможно добавление специального узла - IGSN (Internet GPRS Support Node - узел поддержки Интернета).

Рисунок 2-73. Внутренняя организация GPRS-службы



Следует отметить такой важный параметр функционирования GPRS-службы, как QoS (Quality of Service - качество сервиса). Очевидно, что видеоконференция в режиме реального времени и отправка сообщения электронной почты предъявляют разные требования, например, к задержкам на пути пакетов данных. Поэтому в GPRS существует несколько классов QoS, подразделяющихся по следующим признакам:

- необходимому приоритету (существует высокий, средний и низкий приоритет данных)
- надежности (разделение на три класса по количеству возможных ошибок разного рода, потерянных пакетов и т.п.)
- задержкам (задержки информации вне GPRS-сети в расчет не принимаются)
- количественным характеристикам (пиковое и среднее значение скорости)

Класс QoS выбирается индивидуально для каждой новой сессии передачи данных.

Стандарт услуги GPRS предусматривает два режима соединений:

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

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

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

Развитие стандарта GPRS-службы предполагает вскоре поддержку режима «многоточечной» передачи информации PTM.

Новый стандарт для 3G-сетей.

Следующим шагом от GSM к сетям третьего поколения (3G-сети) или UMTS-системам (Universal Mobile Telephone System) является EDGE-служба (Enhanced Data Rates for GSM Evolution, в вольном переводе - «ускоренная передача данных»), позволяющая осуществлять передачу информации на скоростях до 384 кбит/сек. в восьми GSM-каналах (48 Кбит/сек. на канал).

С EDGE-службой мобильный Интернет становится реальностью. Добавление EDGE-службы к существующим сетям второго поколения делает их совместимыми со стандартами ITU для 3G-сетей.

EDGE-служба – это решение для 3G-сетей, которое позволит существующей сетевой инфраструктуре предоставлять мощные современные мультимедийные услуги для мобильных терминалов. Реализация EDGE позволяет усилить и основные преимущества технологии GPRS-службы: быстрое установление соединений пакетной передачи и более высокая скорость в радиоинтерфейсе.

Для внедрения EDGE-службы «поверх GPRS» операторам необходимо заменить аппаратуру базовых станций BS, а пользователям - приобрести поддерживающие EDGE телефонные аппараты. Хотя на настоящий момент сложно представить, какие приложения должен использовать абонент сотовой сети GSM, чтобы ему не хватало скорости в 170 кбит/сек., предлагаемой GPRS. Но в наше время бурно развивающихся цифровых технологий прогнозы - дело неблагодарное…

UMTS (Universal Mobile Telecommunications System) - Универсальная система мобильных телекоммуникаций – это один из стандартов, разрабатываемый Европейским институтом стандартов телекоммуникаций (ETSI) для внедрения 3G-сетей в Европе. Сегодня основным фактором, определяющим развитие мобильной связи, является голосовая телефония. Появление GPRS и EDGE, а затем переход к UMTS-системе открывают дорогу ко многим дополнительным возможностям, помимо голосовой связи. UMTS - это высокоскоростная передача данных, мобильный Интернет, различные приложения на основе Интернета, интранета и мультимедиа (подробно об этих приложениях речь пойдет в главе 7).

Ключевой технологией для UMTS является широкополосный многостанционный доступ с разделением кодов (WCDMA). Эта революционная технология радиодоступа, выбранная в сентябре 1998 года Европейским институтом стандартов телекоммуникаций, поддерживает все мультимедийные услуги 3G-сетей. Системы WCDMA/UMTS включают усовершенствованную базовую сеть GSM и радиоинтерфейс по технологии WCDMA. Скорость передачи в радиоканале для мобильного абонента достигает 2 Мбит/сек. WCDMA предназначена для использования в системах, работающих в частотном диапазоне 2 ГГц, который позволит в полной мере использовать все преимущества этой технологии. Например, всего одна несущая WCDMA шириной 5 МГц должна обеспечить предоставление смешанных услуг, требующих скоростей передачи от 8 кбит/сек. до 2 Мбит/сек. А мобильные терминалы, совместимые с WCDMA, смогут в соответствии с рекомендациями ITU работать сразу с несколькими услугами.

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

GSM – пример системы, где использована довольно сложная комбинация техник FDM, TDM, ALOHA для беспроводной сотовой связи. В ней ни один из пользователей системы не может использовать всю полосу пропускания, предоставленную системе. Если при этом принять в расчет сужение полосы пропускания из-за проблем на границе сот, падение мощности сигналов от мобильных терминалов в пограничных сотовых зонах, накладных расходов на шифрование в целях безопасности, то становится ясно, что высокую скорость передачи в этой системе получить не просто.

Метод CDMA основан на принципиально иной идеи – каждый участник связи может использовать всю полосу пропускания канала. У каждого свой уникальный «язык», поэтому все могут говорить сразу. Понимать друг друга будут только те, кто говорит на одном языке.

В CDMA-системе каждый бит сообщения кодируется последовательностью из m частиц. Бит со значением 0 передается инвертированной последовательностью частиц, бит 1 – прямой. Каждой мобильной станции присваивается уникальный код – последовательность частиц.

Ясно, что такая техника возможна, только если при увеличении объема передаваемой информации будет пропорционально увеличиваться ширина полосы пропускания. При использовании техники FDM канал 1 МГц может быть разделен на 100 подканалов по 10 кГц каждый. Таким образом, мы сможем осуществлять передачу по таким подканалам со скоростью 10 кбит/сек. (1 бит на 1 Гц). В случае CDMA каждый может использовать всю полосу, т.е. 1 МГц. Если

мы будем использовать 10-разрядные последовательности частиц (что предполагает 2^{10} разных последовательностей), то сможем передавать данные со скоростью 100 кбит/сек.

Кроме этого, поскольку каждая станция имеет уникальную последовательность частиц, то не требуется дополнительного шифрования. Отсюда ясно преимущество CDMA по отношению к TDM- и FDM-техникам.

Идея уникальности последовательности частиц для каждой станции основана на ортогональных кодах. Суть этих кодов состоит в следующем: если обозначить последовательности частиц для станции S как S, а для станции T - T, то

$$(\mathbf{S},\mathbf{T}) = \frac{1}{m}\sum_{i=1}^{m} S_i T_i = 0$$

Как получатель узнает последовательность частиц отправителя? Например, за счет соответствующего быстродействия он может слышать всех, обрабатывая алгоритмом декодирования для каждой последовательности в параллель. На практике поступают несколько иначе. Однако мы не будем здесь заниматься этим вопросом.

Билет № 28.

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

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

Стандарт IEEE 802.3 относится к 1-настойчивым протоколам CSMA/CD для локальных сетей. Напомним, что прежде чем начать передачу, станция, использующая такой протокол, опрашивает канал. Если он занят, то она ждет и как только он освободится, она начинает передачу. Если несколько станций одновременно начали передачу, то возникает коллизия. Тут же передача прекращается. Станции ожидают некоторый случайный отрезок времени, и все начинается сначала.

Стандарт IEEE 802.3 имеет очень интересную историю. Начало положила ALOHA. Потом компания XEROX построила CSMA/CD канал на 2,94 Мбит/сек., объединивший 100 персональных компьютеров на 1 километре кабеля. Эта система была названа Ethernet. Ethernet Xerox'а получил такой большой успех, что Xerox, DEC и Intel решили объединиться и создали Ethernet 10 Мбит/сек. Эта разработка и составила основу стандарта IEEE 802.3. Отличие стандарта от оригинальной разработки состояло в том, что стандарт охватывал все семейство 1-настойчивых алгоритмов, работающих со скоростью от 1-10 Мбит/сек. Есть отличия в заголовке кадров. Стандарт определяет также параметры физической среды для 50-омного коаксиального кабеля.

IEEE 802. Кабели.

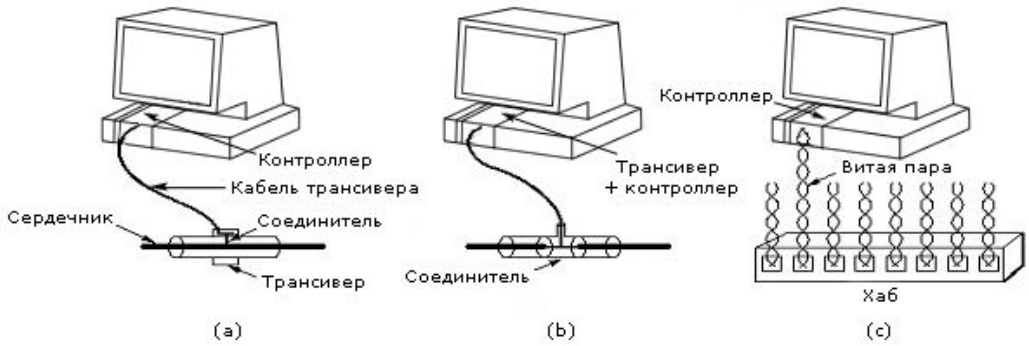
Всего по стандарту допускается четыре категории кабелей. Исторически первым был так называемый «толстый» Ethernet - 10Base5. Это желтого цвета кабель с отметками через каждые 2,5 метра, которые указывают, где можно делать подключения. Подключение делается через специальные розетки с трансивером, которые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

Вторым исторически появился кабель 10Base2 - «тонкий» Ethernet. Это более простой в употреблении кабель с простым подключением через BNC-коннектор. Этот коннектор представляет собой Т-образное соединение коаксиальных кабелей. Кабель для тонкого Ethernet дешевле. Однако его сегмент не должен превосходить 200 метров и содержать более 30 машин.

Проблемы поиска обрыва, частичного повреждения кабеля или плохого контакта в коннекторе привели к созданию совершенно иной кабельной конфигурации на витой паре. Здесь каждая машина соединена со специальным устройством - хабом (hub) витой парой. Этот способ подключения называется 10Base-T.

Данные три способа подключения показаны на рисунке 4-16. В 10Base5 (рисунок 4-16 (a)) трансивер размещается прямо на кабеле. Он отвечает за обнаружение несущей частоты и коллизий. Когда трансивер обнаруживает коллизию, он посылает специальный сигнал по кабелю, чтобы гарантировать, что другие трансиверы услышат коллизию. Трансивер на кабеле соединяется с компьютером трансиверным кабелем. Его длина не должна превосходить 50 метров. Он состоит из 5 витых пар. Две - для передачи данных к компьютеру и от него, две - для передачи управляющей информации в обе стороны, и пятая пара - для подачи питания на трансивер. Некоторые трансиверы позволяют подключать к себе до восьми машин.

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



Трансиверный кабель подключается к контроллеру в компьютере. На этом контроллере есть специальная микросхема, которая отвечает за прием кадров и их отправку, проверку и формирование контрольной суммы. В некоторых случаях она отвечает за управление буферами на канальном уровне, очередью буферов на отправку, прямой доступ к памяти машины и другие вопросы доступа к сети.

У 10Base2 трансивер расположен на контроллере. Каждая машина должна иметь свой индивидуальный трансивер (рисунок 4-16 (b)).

У 10Base-T трансивера нет вовсе (рисунок 4-16 (c)). Машина соединяется с хабом витой парой, длина которой не должна превосходить 100 метров. Вся электроника сосредоточена в хабе.

Наконец, последний вид кабеля 10Base-F - оптоволоконный вариант. Он относительно дорог, но низкий уровень шума и длина одного сегмента - важные достоинства этого кабеля.

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.

Манчестерский код.

Ни одна версия IEEE 802.3 не использует прямого кодирования, т.к. оно очень неоднозначно. Так, например, оно не позволяет однозначно отличить 00100011 от 10001100 или 01000110 без дополнительных усилий на синхронизацию. Нужен был метод, который бы позволял определять начало, середину и конец передачи каждого бита без особой побитной синхронизации. Было предложено два метода для этого: манчестерский код и дифференциальный манчестерский код.

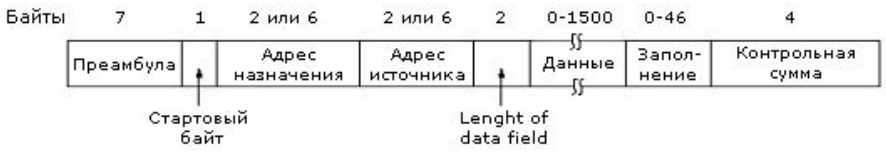
При использовании Манчестерского кода весь период передачи бита разбивается на два равных интервала. При передаче 1 передается высокий сигнал в первом интервале и низкий - во втором. При передаче 0 - наоборот. Такой подход имеет переход в середине передачи каждого бита, что позволяет синхронизироваться 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: протокол MAC-подуровня.

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

Рисунок 4-19. Структура кадра IEEE 802.3



Хотя стандарт допускает двух- и шестибайтные адреса, для 10Base используются только 6-байтные. 0 в старшем бите адреса получателя указывает на обычный адрес. Если там 1, это признак группового адреса. Групповой адрес позволяет обращаться сразу к нескольким станциям одновременно. Если адрес получателя состоит из одних единиц - это вещательный адрес, т.е. этот кадр должны получить все станции в сети.

Другой интересной возможностью адресации является различение локального адреса и глобального. На то, какой адрес используется, указывают 46 бит. Если этот бит 1 - это локальный адрес, который устанавливает сетевой администратор и вне данной сети этот адрес смысла не имеет. Глобальный адрес устанавливает IEEE и гарантирует, что нигде в мире нет такого второго. С помощью 46 битов можно получить $7x10^{13}$ глобальных адресов.

Поле длины указывает на длину поля данных. Она может быть от 0 до 1500 байт. То, что поле данных может иметь длину 0, вызывает проблему для обнаружения коллизий. Поэтому IEEE 802.3 предписывает, что кадр не может быть короче 64 байт. Если длина поля данных недостаточна, то поле Pad компенсирует нехватку длины.

Ограничение на длину кадра связано со следующей проблемой. Если кадр короткий, то станция может закончить передачу прежде, чем начало кадра достигнет самого отдаленного получателя. В этом случае она может пропустить коллизию и ошибочно считать, что кадр доставлен благополучно.

Пусть τ - минимальное время распространения сигнала до самой удаленной станции. Тогда минимальная длина кадра должна быть такой, чтобы время передачи кадра такой длины занимало не менее 2τ секунд. Для IEEE 802.3 (2,5 км и четырех репитерах) это время равно 51,2 мксек., что соответствует 64 байтам. При больших скоростях длина кадра должна быть еще больше. И это становится проблемой при переходе на высокие скорости передачи.

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

Двоичный экспоненциальный алгоритм задержки.

Теперь рассмотрим, как определяется случайная величина задержки при возникновении коллизий. При возникновении коллизии время разбивается на слоты длиной, соответствующей наибольшему времени распространения сигнала в оба конца (2τ). Для 802.3, как уже было указано, это время при длине линии 2,5 км и четырех репитерах равно 51,2 мксек.

При первой коллизии станции, участвовавшие в ней, случайно выбирают 0 или 1 слот для ожидания. Если они выберут одно и то же число, то коллизия возникнет опять. Тогда выбор будет происходить среди чисел 0, 2^i, 1, где i - порядковый номер очередной коллизии.

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

Этот алгоритм называется алгоритм двоичной экспоненциальной задержки. Он позволяет динамически подстраиваться под число конкурирующих станций. Если для каждой коллизии случайный интервал был бы равен 1023, то вероятность повторной коллизии для двух станций была бы пренебрежимо мала. Однако среднее время ожидания разрешения коллизии было бы сотни слотов. Если бы случайный интервал был бы постоянно 0 или 1, то при 100 станциях разрешение коллизии потребовало бы годы, так как 99 станций должны были бы случайно выбрать, скажем, 0 и лишь одна - 1.

Производительность IEEE 802.3.

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

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

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

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

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

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

Из этой формулы видно, что чем длиннее кабель, тем хуже эффективность, т.к. растет длительность периода состязаний. При длительности 51,2 мксек, что соответствует 2,5 км при четырех репитерах и скорости передачи 10 Мбит/сек., минимальный размер кадра - 512 бит, или 64 байта. Хотя с ростом длины кадра эффективность канала растет, время задержки кадра в системе также увеличивается.

Билет № 29.

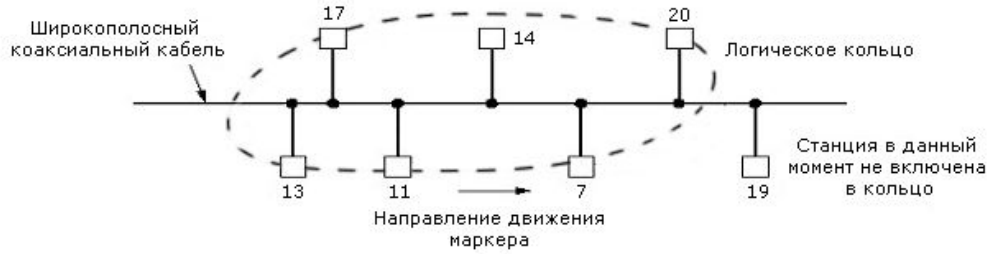
Стандарты IEEE 802.x для локальных и муниципальных сетей: Стандарт IEEE 802.4 – шина с маркером (область применения, протокол MAC подуровня, логическая поддержка логического кольца). Стандарт IEEE 802.2: управление логическим каналом.

Стандарт 802.3 получил очень широкое распространение. Однако там, где возникала потребность в режиме реального времени, он вызывал нарекания. Во-первых, потому что с ненулевой вероятностью станция может ожидать сколь угодно долго отправки кадра. В стандарте нет понятия приоритета кадра, что очень важно для приложений реального времени.

Простейшая система с заранее известным наихудшим случаем ожидания - кольцо. Если есть n станций, соединенных в кольцо, и передача кадра занимает T сек., то максимальное время ожидания передачи кадра будет не более nT. Специалистам по системам реального времени нравилась идея кольца, но не нравилась ее физическая реализация. Во-первых, кольцо не надежно - обрыв в одном месте разрушает всю систему. Во-вторых, оно плохо соответствовало топологии многих сборочных линий на заводах. В результате был разработан стандарт, который объединял достоинства 802.3 с гарантированным наихудшим временем передачи и приоритетностью кадров.

Этот стандарт был назван 802.4 и описывал шину с маркером. Физически шина с маркером имеет линейную или древовидную топологию. Логически станции объединены в кольцо (рисунок 4-22), где каждая станция знает своего соседа справа и слева. Когда кольцо инициализировано, станция с наибольшим номером может послать первый кадр. После этого она передает разрешение на передачу кадра своему непосредственному соседу, посылая ему специальный управляющий кадр - маркер. Передача кадра разрешена только той станции, которая владеет маркером. Так как маркер один, то всегда только одна станция может осуществлять передачу, и коллизий не возникает.

Рисунок 4-22. Маркерная шина



Важно отметить, что на порядок передач влияет только логические номера станций, а не их физическое размещение. Маркер передается только логическому соседу. Естественно, протокол должен учитывать случай, когда станция подключается к кольцу в ходе функционирования.

802.4 MAC - очень сложный протокол, который поддерживает 10 таймеров и более 24 внутренних переменных. Его описание занимает более 200 страниц.

На физическом уровне 802.4 использует коаксиальный 75-омный кабель, три разные схемы аналоговой модуляции, скорость передачи - 1,5 и 10 Мбит/сек. Он полностью несовместим с физическим уровнем 802.3.

MAC-протокол для шины с маркером.

При инициализации станции образуют кольцо в соответствии с их адресами от старших к младшим. Маркер передают от станций с большими адресами к станциям с меньшими адресами. Каждый раз, когда станция получает маркер, она может передавать кадры в течение определенного промежутка времени. После этого она должна передать маркер следующей станции. Если кадры достаточно короткие, то может быть послано несколько последовательных кадров. Если у станции нет данных для передачи, то она передает маркер дальше, немедленно по его получении.

Шина с маркером определяет четыре приоритета для кадров: 0, 2, 4 и 6. Для простоты можно представить, что станция разделена внутри на четыре подстанции, по одной на уровень приоритета. Как только кадр поступает сверху, он распределяется на одну из подстанций в соответствии с приоритетом. Таким образом, каждая подстанция имеет свою очередь кадров на передачу.

Когда маркер поступил по кабелю, он попадает на подстанцию с приоритетом 6. Если у нее есть кадр на передачу, она его передает, если нет, то маркер передается подстанции с приоритетом 4. Эта подстанция передает свои кадры в течение своего интервала времени, либо по истечении определенного временного промежутка передает маркер подстанции с приоритетом 2. Так продолжается до тех пор, пока либо подстанция с приоритетом 0 перешлет свои кадры, либо ее таймер исчерпается и она отдаст маркер следующей станции.

Из приведенной схемы ясно, что подстанция с номером 6 имеет наивысший приоритет и в любом случае ее кадрам обеспечена некая гарантированная пропускная способность. Эта подстанция и используется для передачи трафика реального времени.

На рисунке 4-23 показан формат кадра для шины с маркером. Поле Preamble предназначено для синхронизации таймера получателя. Его длина не короче одного байта. Поля Start delimiter и End delimiter предназначены для распознавания начала и конца кадра. Они имеют специальную кодировку, которая не может встретиться у пользователя. Поэтому поля длины кадра не требуется. Поле Frame control отделяет управляющие поля от полей данных. Для кадров данных здесь указывается приоритет кадра. Это поле также используется станцией-получателем для подтверждения корректного или некорректного получения кадра. Для этого отправитель устанавливает в этом поле специальный индикатор подтверждения. При наличии такой установки станция-получатель, даже не имея маркера, может послать подтверждение. Без этого поля получатель был бы лишен возможности давать подтверждения - у него было бы маркера.

Рисунок 4-23. Формат кадра для шины с маркером

В управляющих кадрах это поле используется для указания типа кадра. Среди них передача маркера, всевозможные кадры для поддержки кольца, например, включение станции в кольцо и исключение станции из кольца.

Поле адреса получателя и адреса отправителя такие же, как и в стандарте 802.3. В нем адреса могут быть 2-байтные или 6-байтные. Поле данных может иметь длину не более 8182 байта при 2-байтном адресе и 8174 - при 6-байтном адресе. Это в пять раз длиннее, чем в 802.3, т.к. в нем необходимо предотвратить захват одной станцией канала надолго. Здесь это не опасно, т.к. есть таймер, а для реального времени бывает полезно иметь длинные кадры. Контрольная сумма, как и в 802.3, используется для обнаружения ошибок.

Поддержка логического кольца.

Поддержка логического кольца в основном связана с проблемами включения и выключения станций. MAC-подуровень 802.4 детально описывает алгоритм, позволяющий сохранять известным наихудший случай при передаче маркера. Ниже мы рассмотрим кадры, которые используются в этом случае (таблица 4-24).

Таблица 4-24. Управляющие кадры шины с маркером

Контрольное поле	Название	Значение поля
00000000	Claim_token	Запуск маркера при инициализации
00000001	Solicit_successor_1	Разрешение присоединиться к кольцу
00000010	Solicit_successor_2	Разрешение присоединиться к кольцу
00000011	Who_follows	Восстановление при потере маркера
00000100	Resolve_contention	Запуск разрешения коллизии
00001000	Token	Передача маркера
00001100	Set_successor	Разрешение покинуть кольцо

Когда кольцо установлено, интерфейс каждой станции хранит адреса предшествующей и последующей станции. Периодически держатель маркера рассылает один из кадров SOLICIT_SUCCESSOR, предлагая новым станциям присоединиться к кольцу. В этом кадре указаны адрес отправителя и адрес следующей за ним станции в кольце. Станции с адресами в этом диапазоне адресов могут присоединиться к кольцу. Таким образом, сохраняется упорядоченность (по возрастанию) адресов в кольце.

Если ни одна станция не откликнулась на SOLICIT_SUCCESSOR, то станция-обладатель маркера закрывает окно ответа и продолжает функционировать, как обычно. Если есть ровно один отклик, то откликнувшаяся станция включается в кольцо и становится следующей в кольце. Если две или более станции откликнулись, то фиксируется коллизия. Станция-обладатель маркера запускает алгоритм разрешения коллизий, посылая кадр RESOLVE_CONTENTION. Этот алгоритм - модификация алгоритма обратного двоичного счетчика на два разряда.

У каждой станции в интерфейсе есть два бита, устанавливаемых случайно. Их значения 0, 1, 2 и 3. Значение этих битов определяют величину задержки при отклике станции на приглашение подключиться к кольцу. Значения этих бит переустанавливаются каждые 50 мсек.

Процедура подключения новой станции к кольцу не нарушает наихудшее гарантированное время для передачи маркера по кольцу. У каждой станции есть таймер, который сбрасывается, когда станция получает маркер. Прежде чем он будет сброшен, его значение сравнивается с некоторой величиной. Если оно больше, то процедура подключения станции к кольцу не запускается. В любом случае за один раз подключается не более одной станции. Теоретически станция может ждать подключения к кольцу сколь угодно долго, на практике, не более нескольких секунд. Однако с точки зрения приложений реального времени это одно из наиболее слабых мест 802.4.

Отключение станции от кольца очень просто. Станция X с предшественником S и последователем P шлет кадр SET_SUCCESSOR, который указывает P, что отныне его предшественником является S. После этого X прекращает передачу.

Инициализация кольца - это специальный случай подключения станции к кольцу. В начальный момент станция включается и слушает канал. Если она не обнаруживает признаков передачи, то она генерирует маркер CLAIM_TOKEN. Если конкурентов не обнаружилось, то она генерирует маркер сама и устанавливает кольцо из одной станции. Периодически она генерирует кадры SOLICIT_SUCCESSOR, приглашая другие станции включиться в кольцо. Если в начальный момент сразу две станции были включены, то запускается алгоритм обратного двоичного счетчика с двумя разрядами.

Из-за ошибок передач и сбоев оборудования могут возникать проблем с передачей маркера. Например, станция передала маркер соседней, а та неожиданно «грохнулась» - что делать? Стандарт дает прямолинейное решение - передав маркер, станция слушает. Если не последует передач кадра или маркера, то маркер посылается вторично.

Если и при повторной передаче маркера ничего не последовало, то станция посылает кадр WHO_FOLLOWS, где указан не отвечающий сосед. Увидев этот кадр, станция, для которой не отвечающая станция - предшественник, шлет кадр SET_SUCCESSOR и становится новым соседом. При этом не отвечающая станция за плохое поведение исключается из кольца.

Теперь предположим, что остановилась не только следующая станция, но и следующая за ней. В этом случае запускается новая процедура посылкой кадра SOLICIT_SUCCESSOR_2. В ней участвует процедура разрешения конфликтов. При этом все, кто хочет подключиться к кольцу, могут это сделать. Фактически кольцо переустанавливается.

Другой вид проблем возникает, когда останавливается держатель маркера и маркер исчезает из кольца. Эта проблема решается запуском процедуры инициализации кольца. У каждой станции есть таймер, который сбрасывается каждый раз, когда маркер появляется. Если значение этого таймера превысит некоторой заранее установленное значение, то станция генерирует кадр CLAIM_TOKEN. При этом запускается алгоритм обратного двоичного счетчика.

Если оказалось два и более маркера на шине, станция, владеющая маркером, увидев передачу маркера на шине, сбрасывает свой маркер. Так повторяется до тех пор, пока не останется ровно один маркер в системе.

Билет № 30.

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

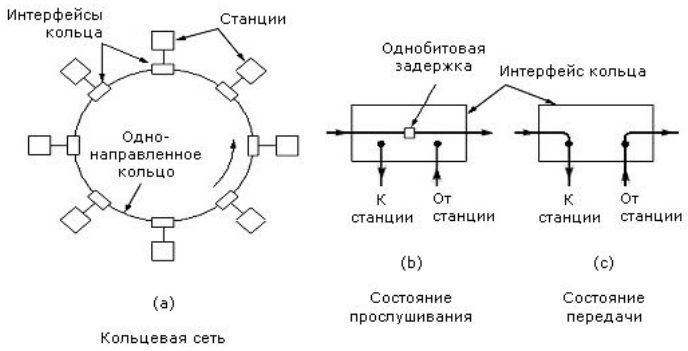
Сети с кольцевой топологией известны давно и используются широко. Среди их многочисленных достоинств есть одно особенно важное - это не среда с множественным доступом, а последовательность соединений точка-точка, образующих кольцо. Соединения точка-точка хорошо изучены, могут работать на разных физических средах: витая пара, коаксиал или оптоволокно. Способ передачи в основном цифровой, в то время как у 802.3 есть значительный аналоговый компонент. Кольцо также представляет справедливую среду с известной верхней границей доступа к каналу. В силу этих причин IBM выбрало кольцо как основу своего стандарта, а IEEE включило его как стандарт 802.5 - кольцо с маркером.

Важной проблемой при создании кольцевой сети является «физическая длина» бита. Пусть данные передаются со скоростью R Мбит/сек. Это значит, что через каждые 1/R мксек. на линии появляется бит. Учитывая, что сигнал распространяется со скоростью 200 м/мксек., то один бит

занимает 200/R метров кольца. Отсюда, при скорости 1 Мбит/сек. и длине окружности 1 км кольцо вмещает не более 5 бит одновременно.

Как уже отмечалось, кольцо - это последовательность соединений точка - точка. Бит, поступая на интерфейс, копируется во внутренний буфер интерфейса и передается по кольцу дальше (см. рисунок 4-25). В буфере бит может быть проанализирован и, возможно, изменен. Эти операции вносят задержку на один бит в каждом интерфейсе.

Рисунок 4-25. Устройство кольца



Пока станциям нечего передавать, в кольце циркулирует маркер - особая последовательность бит. Если станции нужно передать данные, она должна захватить маркер и удалить его из кольца. Это достигается изменением одного бита в 3-х байтном маркере, в результате чего маркер тут же превращается в заголовок обычного кадра. Поскольку в кольце может быть только один маркер, то только одна станция может передавать данные. Так в сети «кольцо с маркером» решается вопрос доступа.

Как следствие конструкции кольца с маркером, сеть должна иметь достаточную протяженность, чтобы маркер мог уместиться в ней целиком, даже когда все станции находятся в ожидании. Задержки складываются из двух компонентов: 1 бит - задержка на интерфейсе станции и задержка на распространение сигнала. Учитывая, что станции могут выключаться, например, на ночь, следует, что на кольце должна быть искусственная задержка, если оно недостаточно длинное.

Интерфейс станций может работать в двух режимах: прослушивания и передачи. В режиме прослушивания он лишь копирует бит в свой буфер и передает этот бит дальше по кольцу. В режиме передачи, предварительно захватив маркер, интерфейс разрывает связь между входом и выходом и начинает передачу. Чтобы быть способным за однобитовую задержку переключиться из одного режима в другой, интерфейс должен предварительно забуферизовать данные для передачи.

По мере распространения передаваемых битов по кольцу они будут возвращаться к передающей станции. Она должна убирать их либо с линии, либо в буфер для последующего анализа с целью повышения надежности передачи, либо просто сбрасывая. Поскольку кадр целиком никогда не появляется на линии, то архитектура сети кольца с маркером не накладывает никаких ограничений на длину кадра. Как только станция закончила передачу и удалила последний переданный бит с линии, она должна сгенерировать маркер и переключиться в режим прослушивания.

В такой сети просто уведомлять о получении кадра. В каждом кадре есть бит уведомления. Станция-получатель устанавливает этот бит при получении кадра. Станция-отправитель при возвращении кадра анализирует этот бит и может определить, был ли этот кадр получен. Так же

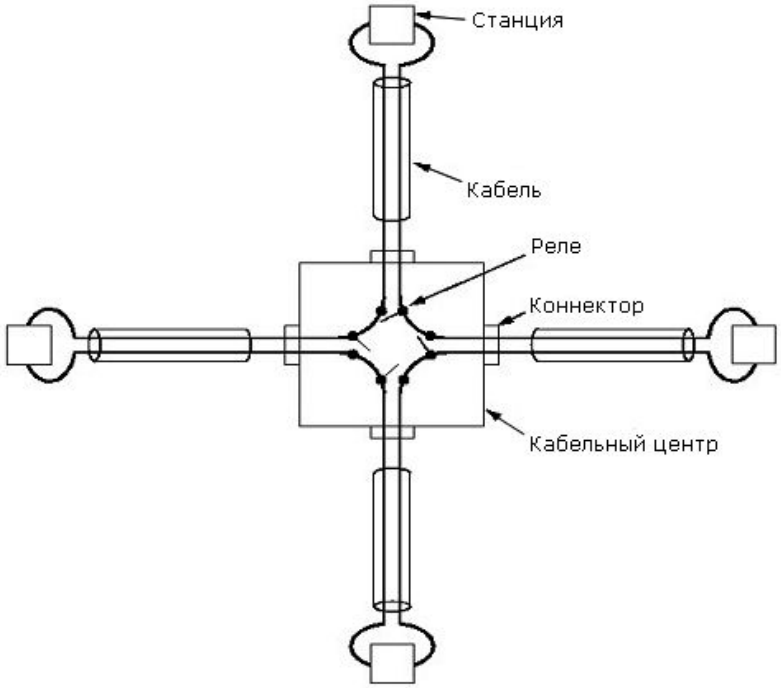
можно поступать и с проверкой контрольной суммы, главное, чтобы эта проверка могла быть выполнена за однобитовую задержку.

При малой загрузке станции в кольце сразу могут передавать свои сообщения. По мере роста загрузки у станций будут расти очереди на передачу и они в соответствии с кольцевым алгоритмом будут захватывать маркер и вести передачу. Постепенно загрузка кольца будет расти, пока не достигнет 100%.

Теперь обратимся непосредственно к стандарту IEEE 802.5. На физическом уровне он использует витую пару со скоростью 1 или 4 Мбит/сек., хотя IBM позднее ввела 16 Мбит/сек. Сигнал на линии кодируется с помощью дифференциального манчестерского кода, используя запрещенные комбинации low-low и high-high для управляющих байтов.

С кольцом связана одна серьезная проблема - если связь в кольце где-то нарушается, то вся конфигурация становится неработоспособной. Проблема решается с помощью так называемого кабельного центра. Это решение показано на рисунке 4-26. В случае, если какая-то станция выходит из строя, реле замыкается и станция исключается из кольца. Реле может управляться и программно, выводя временно станцию из кольца, например, для тестирования. Хотя стандарт 802.5 непосредственно не предписывает использование кабельного центра, на практике он часто используется с целью повышения надежности и удобства обслуживания сети.

Рисунок 4-26. 4 станции, соединенные через кабельный центр

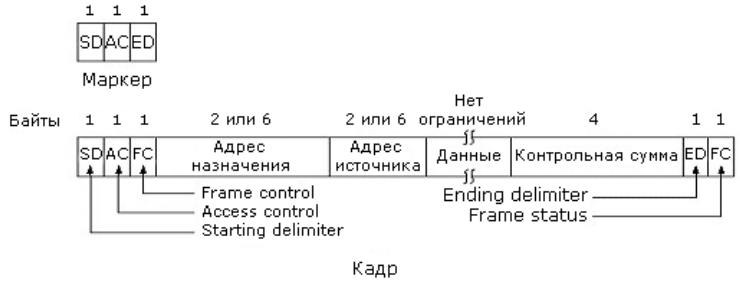


Вместо станции к кабельному центру может присоединяться другой кабельный центр. Таким образом, кабельные центры могут объединяться в структуры, подобно тому как хабы соединяются в 802.3. Однако форматы и протоколы у них разные.

Кольцо с маркером: протокол MAC-подуровня.

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

Рисунок 4-27. Устройство маркера и кадра передачи данных



В нормальных условиях станция-отправитель должна постоянно забирать с линии биты, которые возвращаются к ней, обойдя кольцо. Даже очень длинное кольцо вряд ли будет способно вместить короткий кадр. Поэтому ранее посланные биты начнут возвращаться прежде, чем станция закончит передавать кадр.

Станция может держать маркер не более 10 мсек., если при инсталляции не было установлено иного значения. Если после отправки кадра остается достаточно времени, то посылаются следующие. После того как посланы все кадры или истекло время владения маркером, станция обязана сгенерировать маркер и вернуть его на линию.

Байты Starting delimiter и Ending delimiter отмечают начало и конец кадра соответственно. Они содержат запрещенные в дифференциальных манчестерских кодах последовательности. Байт Access control содержит маркерный бит, Monitor bit, Priority bits, Reservations bits (они будут описаны позднее).

Поля Destination address и Source address такие же, как и в стандартах 802.3 и 802.4. За ними следует поле данных, которое может быть сколь угодно длинное, лишь бы его передача уместилась во время владения маркером. Поле контрольной суммы такое же, как и в 802.3 и 802.4.

Байт, которого нет ни в 802.3 ни в 802.4 - Frame status. В нем есть биты A и C. Когда кадр поступает к станции-получателю, ее интерфейс инвертирует бит A. Если кадр успешно скопирован, то инвертируется и бит C. Кадр может быть не скопирован в силу разных причин: задержки, отсутствия места в буфере и т.п.

Когда станция-получатель снимает ранее посланные биты с линии, она анализирует биты A и C. По их комбинации она может определить, успешно ли прошла передача. Возможны три комбинации значений этих битов:

1. A=0, C=0 - получатель отсутствует

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

3. A=1, C=1 - получатель есть и кадр принят

Биты A и C обеспечивают автоматическое уведомление о получении кадра. Если кадр почему-то не был принят, то у станции есть несколько попыток передать его. Биты A и C дублируются в байте Frame status с целью повысить надежность, так как этот байт не подпадает под контрольную сумму.

Ending delimiter содержит специальный бит, который устанавливает интерфейс любой станции, если он обнаруживает ошибку. Там также есть бит, которым можно помечать последний кадр в логической последовательности кадров.

В 802.5 есть тщательно проработанная схема работы с приоритетами. В среднем байте 3-байтного маркера есть поле, отведенное для приоритета. Если станции надо передать кадр с приоритетом n, то ей придется ждать, пока появится маркер с приоритетом, меньшим или равным n. Кроме того, когда кадр с данными проходит по кольцу, станция может указать значение приоритета, который ей нужен. Для этого она записывает нужное значение в поле Reservation bits. Однако если в нем уже записан более высокий приоритет, станция не может этого делать. После завершения передачи кадра генерируется маркер с приоритетом, зарезервированном в этом кадре.

Описанный механизм приоритетов имеет один недостаток: приоритет все время растет. Поэтому 802.5 предусматривает довольно сложные правила понижения приоритета. Суть этих правил сводится к тому, что станция, установившая наивысший приоритет, обязана его понизить после передачи кадра.

Заметим, что работа с приоритетами в кольце с маркером и шине с маркером организована по-разному. В шине с маркером каждая станция получает свою долю пропускной способности канала. В кольце с маркером станция с низким приоритетом может ждать сколь угодно долго маркера с надлежащим приоритетом. Это различие отражает различие подходов двух стандартов: что важнее - высокоприоритетный трафик или равномерное обслуживание всех станций.

Поддержка кольца.

В стандарте 802.5 поддержка кольца организована иначе, чем это сделано в 802.4. Там был создан довольно длинный протокол для полностью децентрализованной поддержки. В 802.5 предусмотрено, что в кольце всегда есть станция-монитор, контролирующая кольцо. Если станция-монитор по какой-либо причине потеряет работоспособность, есть протокол выбора и объявления другой станции-монитора на кольце. Любая станция способна быть монитором.

При включении или если какая-то станция заметит отсутствие монитора, она посылает кадр CLAIM_TOKEN. Если она первая, кто послал такой кадр, то она и становится монитором. В таблице 4-28 показаны кадры для поддержки кольца.

Таблица 4-28. Кадры поддержки кольца

Контрольное поле	Название	Значение кадра
00000000	Duplicate address test	Проверка, имеют ли 2 станции одинаковый адрес
00000010	Beacon	Локализация разрыва кольца
00000011	Claim token	Попытка стать монитором
00000100	Purge	Реинициализация кольца
00000101	Active monitor present	Периодически рассылается монитором
00000101	Standby monitor present	Заявление о наличии потенциальных мониторов

Среди задач, которые должен решать монитор, есть следующие: слежение за наличием маркера, выполнение определенных действий, если нарушено, устранение грязи или беспризорных кадров. Такие кадры могут появиться, если станция начала передачу и не закончила по какой-либо причине. Монитор обнаруживает отсутствие маркера с помощью специального таймера, отмечающего время отсутствия маркера на кольце. Если значение этого таймера превысит некоторое значение, то считается, что маркер потерян, и монитор обязан принять надлежащие меры.

При появлении грязи, т.е. кадра с неверным форматом или контрольной суммой, монитор снимает его с линии и генерирует маркер. Беспризорные кадры монитор обнаруживает с помощью Monitor bit в байте Access control. Когда кадр проходит через монитор первый раз, монитор устанавливает этот бит в единицу. Поэтому, если очередной кадр пришел с единицей в этом бите, то этот кадр не был принят и монитор должен его удалить из кольца.

Важной функцией монитора является установка задержки на кольце, достаточной, чтобы в кольце уместился 24-битный маркер.

Монитор не может определить, где произошел разрыв кольца. Поэтому, если какая-то станция заметит отсутствие соседа, она генерирует BEACON-кадр. По мере распространения этого кадра по кольцу определяется, какие станции живы, а какие нет, и последние удаляют из кольца с помощью кабельного центра.

Нет механизма, позволяющего сменить существующий монитор.

Билет № 31.

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

Сравнение 802.3, 802.4 и 802.5.

Начать следует с того, что все три стандарта используют примерно одинаковую технологию и имеют примерно одинаковую производительность. Инженеры могут бесконечно спорить о том, что лучше - коаксиал или витая пара, но рядовому пользователю, компьютерному обывателю это, как правило, не важно.

Достоинства 802.3

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

Недостатки 802.3

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

Достоинства 802.4

1. Используется обычный телевизионный кабель, высоко надежный.

2. Более детерминирован чем 802.3.

3. Минимальная длина кадра короче.

4. Поддерживает приоритеты.

5. Есть гарантированная доля трафика.

6. Прекрасная эффективность и пропускная способность при высокой загрузке.

7. Коаксиальный кабель может поддерживать несколько каналов.

Недостатки 802.4

1. Используется много аналоговой аппаратуры.

2. Очень сложный протокол.

3. Большие задержки при низкой загрузке.

4. Используется относительно немногими пользователями.

5. Нельзя использовать оптоволокно.

Достоинства 802.5

1. Использует соединения точка-точка: полностью цифровое и простое в обращении.

2. Можно использовать любую физическую среду - от почтовых голубей до оптоволокна.

3. Стандартная витая пара дешева и проста в обращении.

4. Наличие кабельного центра делает кольцо с маркером единственным стандартом, где нарушения физической среды могут восстанавливаться автоматически.

5. Есть приоритеты, однако схема их установки не так справедлива, как в шине с маркером.

6. Небольшой размер минимального кадра и неограниченный размер передаваемого кадра.

7. Прекрасная пропускная способность при высокой загрузке, как и у 802.4.

Недостатки 802.5

1. Основной недостаток - наличие централизованного монитора, который является критическим компонентом.

2. Из-за схемы передачи маркера относительно большие задержки при небольшой загрузке.

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

Стандарт IEEE 802.2: управление логическим каналом.

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

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

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

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

Билет № 32.

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

Довольно часто в организации возникает потребность соединить между собой несколько ЛВС. Этой цели служат специальные устройства, называемые мосты, которые функционируют на уровне канала данных. Это означает, что это устройство не анализирует заголовки пакетов сетевого уровня и выше и, таким образом, может просто копировать пакеты IP, IPX, OSI, в то время как маршрутизаторы могут работать только с определенными пакетами.

Далее мы рассмотрим устройство мостов, а частности, для соединения сетей в стандарте 802.3, 802.4, 802.5. Но прежде рассмотрим типичные ситуации, где применяются мосты. Их как минимум шесть:

1. Многие подразделения в организации имеют свои собственные локальные сети. Например, сети факультетов в университетах, отделы в институтах и т.п. В силу различий стоящих перед ним задач, они используют разные приложения и, что естественно, сети, отличные от сетей других подразделений. Рано или поздно наступает момент, когда необходимо интегрировать информационные потоки всей организации и, соответственно, объединять сети между собой.

2. Организация может занимать несколько зданий и, возможно, будет целесообразно в каждом здании иметь свою сеть, объединив их через мосты.

Иногда, при высоких рабочих нагрузках, приходится разбивать сеть на несколько, с целью локализации трафика в каждой подсети. Например, ясно, что класс рабочих станций для студентов лучше оформлять как самостоятельную сеть, локализовав трафик в этой сети: к чему распространять по всему факультету трафик с фотографиями и музыкой?

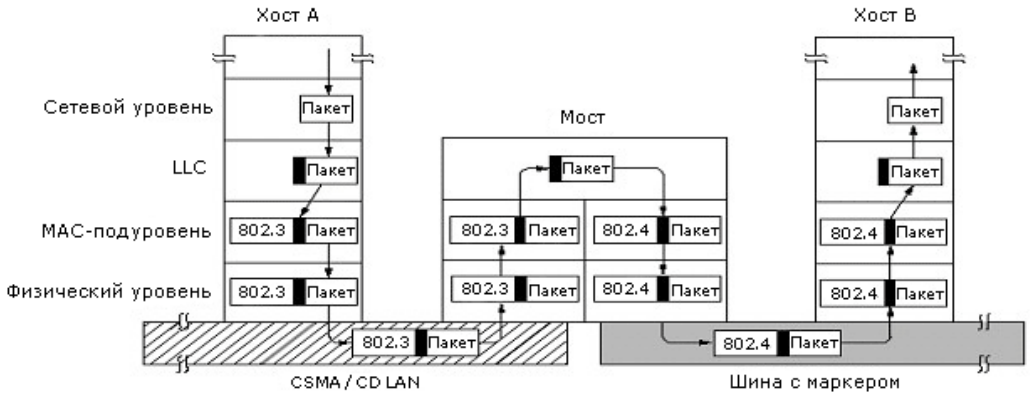
4. В некоторых случаях причиной для использования моста может служить большое расстояние между объединяемыми сетями. Дело в том, что, используя мост, можно увеличить длину сегмента.

5. Мост может увеличить надежность сети. В локальной сети один узел может нарушить работоспособность сети в целом. Мосты, размещенные в критических точках сети, подобно запасным пожарным выходам, могут заблокировать такой узел и предотвратить нарушение работы системы в целом.

6. Мост может повысить безопасность сети. Большинство интерфейсов в ЛВС имеют специальный режим, при котором компьютер получает все пакеты, проходящие в сети, а не только адресованные ему. Всякого рода злоумышленники и просто любопытствующая публика просто обожают эту возможность. С помощью правильно расставленных мостов можно добиться, чтобы определенный трафик проходил лишь по определенным маршрутам, где он бы не мог попасть в чужие руки.

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

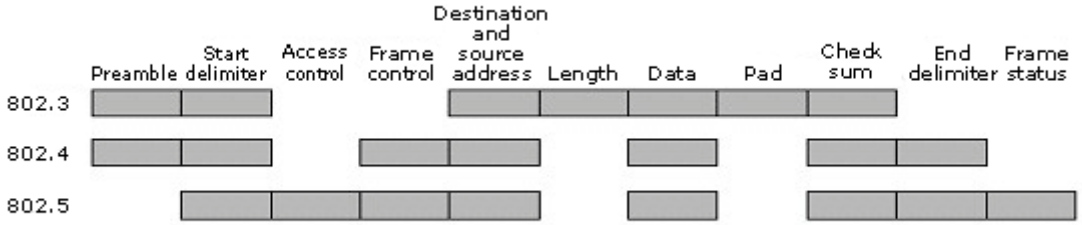
Рисунок 4-31. Двухпортовый мост между 802.3 и 802.4



Мосты из 802 в 802.

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

Рисунок 4-32. Форматы кадров IEEE 802



Следующая проблема - разные сети могут работать с разной скоростью. Если передача идет из скоростной сети в медленную, то мост должен обладать достаточным буфером. Эта проблема может усугубляться непостоянством скорости передачи из-за коллизий, как, например, в 802.3. К тому же несколько сетей могут посылать трафик одной и той же сети, что опять приведет к перекосу скоростей.

Другой тонкой, но важной проблемой является проблема моста как источника временной задержки, которая может влиять на тайм-аут на верхних уровнях. Предположим, что сетевой уровень над 802.4 пытается послать длинное сообщение в виде последовательности кадров. После отправки последнего кадра таймер устанавливается на ожидание уведомления о получении. Если сообщение проходит через мост с медленной 802.5, то есть опасность, что тайм-аут наступит прежде, чем последний кадр будет передан в медленную сеть. Сетевой уровень решит, что все сообщение утеряно, и начнет все сначала. После нескольких попыток сетевой уровень сообщит транспортному, что получатель отсутствует.

Третьей, и наиболее серьезной проблемой является то, что все три стандарта имеют разную максимальную длину кадра. Для 802.3 на 10 Мбит/сек. это 1500 байт, для 802.4 - 8191 байт, для 802.5 максимальная длина ограничена временем удержания маркера. Последняя величина по умолчанию имеет значение 10 мсек., что соответствует 5000 байтам.

Очевидная проблема возникает, когда длинный кадр надо доставить в сеть с короткими кадрами. Разбиение кадра на части не является задачей данного уровня. Это не значит, что таких протоколов не было создано, просто стандарты 802 не предусматривают надлежащих средств. Слишком длинные кадры просто сбрасываются.

Теперь рассмотрим все девять пар.

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

ЛВС-адресат

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

Действия:

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

Установленные параметры:

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

802.3 - 802.3

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

802.4 - 802.3

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

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

802.5 - 802.3

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

802.3 - 802.4

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

802.4 - 802.4

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

802.5 - 802.4

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

802.3 - 802.5

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

802.4 - 802.5

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

802.5 - 802.5

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

Билет № 33.

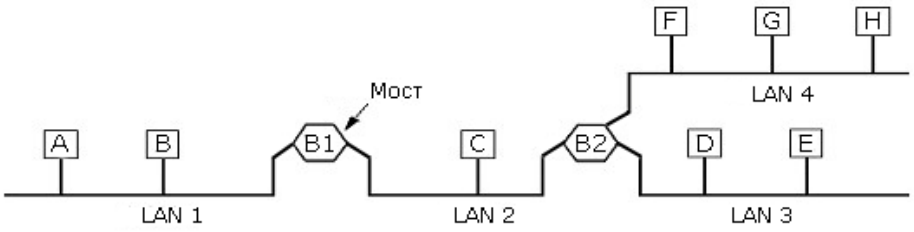
Прозрачные мосты (Мосты с соединяющими деревьями). Мосты с маршрутизацией от источника. Удаленные мосты.

Прозрачные мосты.

Первым мостом 802 является прозрачный мост, или мост с деревом соединений. Основной заботой разработчиков этого моста было обеспечение его полной прозрачности. Они хотели создать устройство по стандарту IEEE, которое пользователь мог бы купить в магазине, подключить кабели своих многочисленных сетей и работать. Подключение в сеть этого устройства не должно было бы требовать каких-либо изменений в оборудовании, программном обеспечении, переинсталляции сетей, загрузки каких-либо таблиц и т.п. Просто купил, принес, включил, и все работает. Как это ни удивительно, но они почти достигли своей цели.

Прозрачный мост функционирует в режиме общедоступности, т.е. ему доступны все пакеты от всех сетей, подключенных к нему. Рассмотрим пример на рисунке 4-34. Кадр для A, поступивший из сети LAN 1, должен быть сброшен, т.к. он уже в нужной сети, а вот кадр из LAN 1 для C или F надо передать в нужную сеть.

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



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

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

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

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

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

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

1. Если адрес отправителя и адрес получателя один и тот же, кадр сбрасывается.

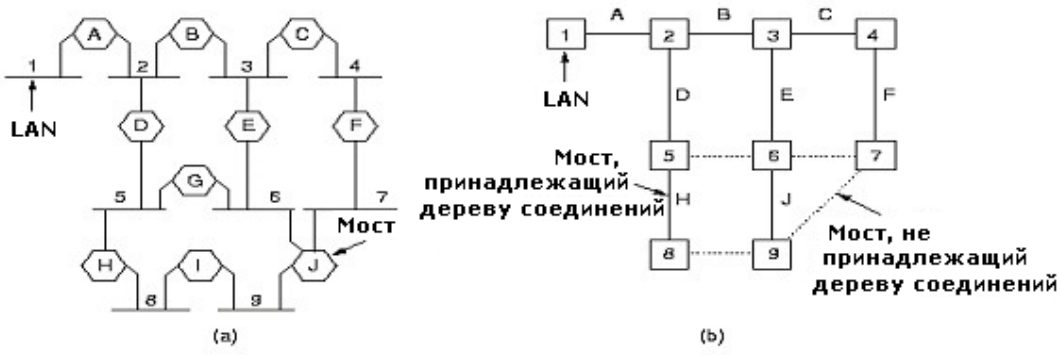
2. Если адрес отправителя и адрес получателя разные, то кадр направляется в надлежащую сеть.

3. Если нет информации в таблице, куда направлять кадр, его посылают во все доступные сети.

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

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

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



Как строится дерево соединений? Прежде всего, надо выбрать один мост из всех в качестве корня дерева. Это делается с помощью уникальных адресов, которые присваиваются мостам при изготовлении, подобно Ethernet-адресам. Из всех мостов корневым выбирается мост с наименьшим номером. Затем для полученного корня строится дерево кратчайших путей, соединяющих корень с каждым мостом и сетью. Полученное дерево и есть дерево соединений.

В результате алгоритм строит единственный маршрут от корня в любую сеть. Хотя дерево соединений охватывает все сети, в нем представлены не все мосты.

Мосты с маршрутизацией от источника.

Прозрачные мосты хороши тем, что их достаточно только подключить, и все работает. Однако они никак не учитывают оптимальное распределение пропускной способности и не могут этого делать. Это привело к появлению иной схемы работы мостов - маршрутизации от источника.

Будем предполагать, что отправитель знает, находится получатель в его локальной сети или нет. Если получатель не в его локальной сети, то отправитель устанавливает старший разряд в адресе получателя в 1. Кроме этого, в заголовке кадра указывается точный маршрут, по которому будут следовать кадр. Этот маршрут представляет собой чередование 12-разрядного адреса сети и 4-разрядного адреса моста. (См. рисунок 4-34.)

Мост с маршрутизацией от источника ловит только те кадры, у которых старший разряд в адресе получателя равен 1. Для каждого такого кадра просматривается описание маршрута и определяется, в какую сеть отправлять этот кадр. Номер сети указан после номера моста в описании маршрута.

Этот алгоритм предполагает, что каждый отправитель знает или может определить наилучший маршрут. Основная идея алгоритма поиска такого маршрута состоит в следующем. Если маршрут к получателю не известен, то отправитель посылает так называемый поисковый кадр. Этот поисковый кадр рассылается всеми мостами по всем сетям. Когда поисковый кадр возвращается обратно, каждый мост оставляет в нем информацию о себе. Таким образом, отправитель, получив ответы на свой поисковый кадр, может выбрать наилучший маршрут среди всех имеющихся.

Этот алгоритм действительно позволяет найти наилучший маршрут, но он имеет один серьезный недостаток - экспоненциальный рост числа поисковых кадров. К тому моменту, как поисковый кадр достигнет уровня N, число поисковых кадров в сети будет порядка 3N-1. Нечто подобное происходит и в сетях с прозрачными мостами, но там этот рост происходит только вдоль дерева связей.

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

Сравнение мостов для 802.

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

Таблица 4-38. Сравнение мостов для 802

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

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

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

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

Одно из основных преимуществ мостов с маршрутизацией от источника - они хотя бы теоретически могут строить оптимальные маршруты, в то время как прозрачные мосты ограничены в выборе маршрута деревом соединений. Мосты с маршрутизацией от источника также могут работать с мостами, включенными в параллель.

Механизмы определения места доставки кадра также имеют свои достоинства и недостатки. В случае с прозрачными мостами недостаток тот, что надо ждать, когда придет кадр от машины получателя. В другом случае - имеем экспоненциальный рост числа поисковых кадров.

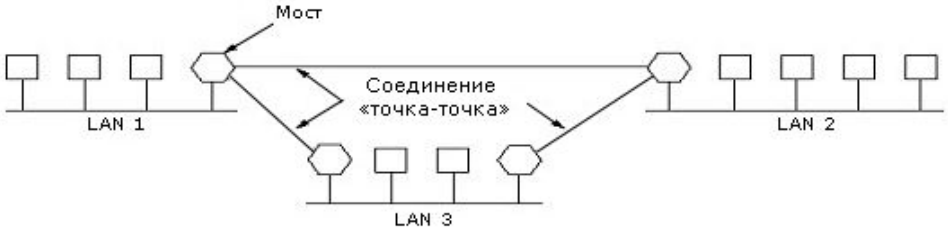
Реакция на ошибки в сетях в обоих случаях разная. В первом случае все происходит автоматически, мосты сами слушают, что происходит в сетях. и реагируют должным образом. Самим хостам делать ничего не приходится. В случае мостов с маршрутизацией от источника ситуация противоположная: вся сложность исправления ошибок и отказов в сети ложится на хосты.

Удаленные мосты.

Как мы уже говорили, основная цель использования мостов – соединение отдельных ЛВС между собой. Это можно сделать, сопоставив каждой ЛВС мост и соединив мосты между собой

каналом «точка-точка». Пример такого соединения показан на рисунке 4-39. Здесь каждый канал точка-точка можно рассматривать как ЛВС без абонентских машин (кто сказал, что сеть обязана всегда их иметь). Тогда у нас есть шесть ЛВС, соединенных через четыре моста.

Рисунок 4-39. Удаленные мосты, соединяющие ЛВС



Для соединений точка-точка можно использовать разные протоколы. Например, можно использовать любой протокол точка-точка и в его кадрах целиком размещать кадры MAC-подуровня. Этот прием хорошо работает в случае идентичных сетей. При этом возникает только одна трудность – маршрутизация кадров в нужную сеть.

Другая возможность – отрезать заголовки MAC-подуровня и на их место вставить заголовки соответствующего канального уровня. Новый MAC-заголовок будет сгенерирован на стороне моста-получателя. Здесь трудности возникают при работе с полем контрольной суммы. Либо надо ее каждый раз перевычислять, либо мы потеряем возможность контролировать ошибки при передаче.

Билет № 34.

Протоколы для высокоскоростных локальных сетей.

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

Основы технологии FDDI.

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

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

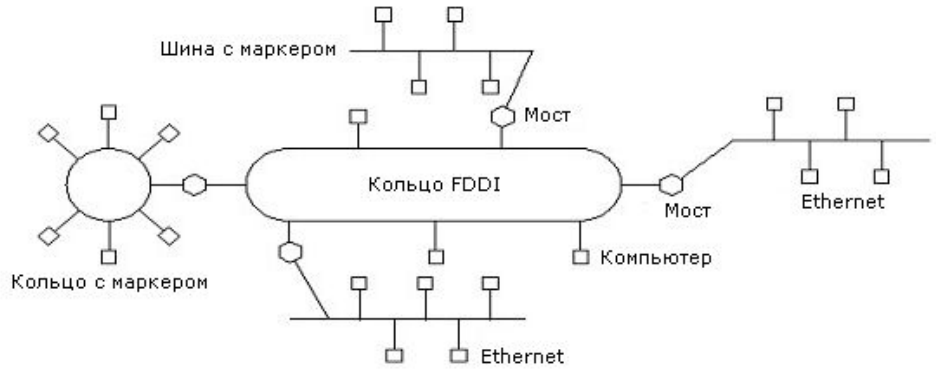
Сеть FDDI строится на основе двух оптоволоконных колец. По одному трафик направлен по часовой стрелке, по другому – против. В случае выхода из строя одного из колец его трафик может быть запущен через второе кольцо. Если оба кольца окажутся поврежденными в одном и том же месте, то они могут быть объединены в одно кольцо, как показано на рисунке 4-41. Такая реконфигурация сети происходит силами концентраторов и/или сетевых адаптеров FDDI.

Использование двух колец - это основной способ повышения отказоустойчивости в сети FDDI, и узлы, которые хотят им воспользоваться, должны быть подключены к обоим кольцам. В

нормальном режиме работы сети данные проходят через все узлы и все участки кабеля первичного (Primary) кольца, поэтому этот режим назван режимом Thru - «сквозным» или «транзитным». Вторичное кольцо (Secondary) в этом режиме не используется. При образовании общего кольца из двух колец передатчики станций по-прежнему остаются подключенными к приемникам соседних станций, что позволяет правильно передавать и принимать информацию соседними станциями.

В стандартах FDDI отводится много внимания различным процедурам, которые позволяют определить наличие отказа в сети, а затем произвести необходимую реконфигурацию. Сеть FDDI может полностью восстанавливать свою работоспособность в случае единичных отказов ее элементов. При множественных отказах сеть распадается на несколько не связанных сетей.

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



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

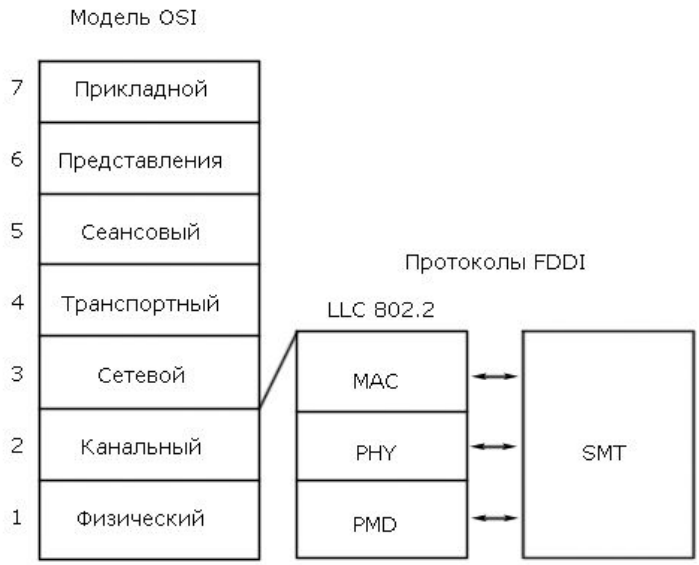
Каждая станция в сети постоянно принимает передаваемые ей предшествующим соседом кадры и анализирует их адрес назначения. Если адрес назначения не совпадает с ее собственным, то она транслирует кадр своему последующему соседу.

В передаваемом в сеть кадре станция назначения отмечает три признака: распознавания адреса, копирования кадра и отсутствия или наличия в нем ошибок. Станция, являющаяся источником кадра для сети, проверяет признаки кадра, дошел ли он до станции назначения, и не был ли при этом поврежден. Процесс восстановления информационных кадров не входит в обязанности протокола FDDI, этим должны заниматься протоколы более высоких уровней.

На рисунке 4-42 приведена структура протоколов технологии FDDI в сравнении с семиуровневой моделью OSI. FDDI определяет протокол физического уровня и протокол подуровня доступа к среде (MAC) канального уровня. Как и многие другие технологии локальных сетей, технология FDDI использует протокол 802.2 подуровня управления каналом данных (LLC), определенный в стандартах IEEE 802.2 и ISO 8802.2. В FDDI используется первый тип процедур

LLC, при котором узлы работают в дейтаграммном режиме - без установления соединений и без восстановления потерянных или поврежденных кадров.

Рисунок 4-42. Структура протоколов технологии FDDI



Физический уровень разделен на два подуровня: независимый от среды подуровень PHY (Physical) и зависящий от среды подуровень PMD (Physical Media Dependent). Работу всех уровней контролирует протокол управления станцией SMT (Station Management). Здесь видна аналогия с организацией физического уровня в СПД АТМ.

Уровень PMD обеспечивает необходимые средства для передачи данных от одной станции к другой по оптоволокну. В его спецификации определяются:

Уровень PHY выполняет кодирование и декодирование данных, циркулирующих между MAC-уровнем и уровнем PMD, а также обеспечивает тактирование информационных сигналов. В его спецификации определяются:

Уровень MAC ответственен за управление доступом к сети, а также за прием и обработку кадров данных. В нем определены следующие параметры:

Уровень SMT выполняет все функции по управлению и мониторингу всех остальных уровней стека протоколов FDDI. В управлении кольцом принимает участие каждый узел сети FDDI. Поэтому все узлы обмениваются специальными кадрами SMT для управления сетью. В спецификации SMT определено следующее:

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

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

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

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

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

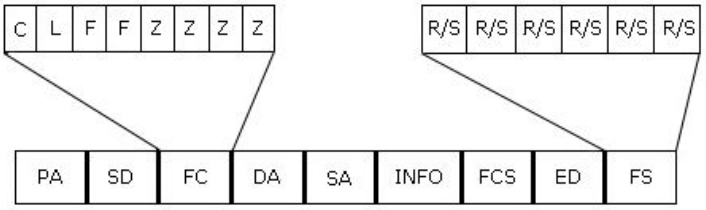
Стандарт FDDI не вводит собственное определение подуровня LLC, а использует его сервисы, описанные в документе IEEE 802.2 LLC. Подуровень MAC выполняет в технологии FDDI следующие функции:

- Поддерживает сервисы для подуровня LLC.
- Формирует кадр определенного формата.
- Управляет процедурой передачи маркера.
- Управляет доступом станции к среде.
- Адресует станции в сети.
- Копирует кадры, предназначенные для данной станции в буфер и уведомляет подуровень LLC и блок управления станцией SMT о прибытии кадра.
- Генерирует контрольную сумму кадра с помощью CRC-кода и проверяет ее у всех кадров, циркулирующих по кольцу.
- Удаляет из кольца все кадры, которые сгенерировала данная станция.
- Управляет таймерами, которые контролируют логическую работу кольца - таймером удержания маркера, таймером оборота маркера и т.д.
- Ведет ряд счетчиков событий, что помогает обнаружить и локализовать неисправности.
- Определяет механизмы, используемые кольцом для реакции на ошибочные ситуации - повреждение кадра, потерю кадра, потерю маркера и т.д.

В каждом блоке MAC параллельно работают два процесса: процесс передачи символов - MAC Transmit - и процесс приема символов - MAC Receive. За счет этого MAC может одновременно передавать символы одного кадра и принимать символы другого кадра.

По сети FDDI информация передается в форме двух блоков данных: кадра и маркера. Формат кадра FDDI представлен на рисунке 4-44.

Рисунок 4-44. Формат кадра FDDI



Рассмотрим назначение полей кадра:

- Преамбула (PA). Любой кадр должен предваряться преамбулой, состоящей как минимум из 16 символов Idle (I). Эта последовательность предназначена для синхронизации приемника и передатчика кадра.
- Начальный ограничитель (Starting Delimiter, SD). Состоит из пары символов JK, которые позволяют однозначно определить границы для остальных символов кадра.

- Поле управления (Frame Control, FC). Идентифицирует тип кадра и детали работы с ним. Имеет 8-битовый формат и передается с помощью двух символов. Состоит из подполей, обозначаемых как CLFFZZZZ, которые имеют следующее назначение:
  - C говорит о том, какой тип трафика переносит кадр - синхронный (значение 1) или асинхронный (значение 0).
  - L определяет длину адреса кадра, который может состоять из 2-х байт или из 6-ти байт.
  - FF - тип кадра, может иметь значение 01 для обозначения кадра LLC (пользовательские данные) или 00 для обозначения служебного кадра MAC-уровня. Служебными кадрами MAC-уровня являются кадры трех типов - кадры процедуры инициализации кольца Claim Frame, кадры процедуры сигнализации о логической неисправности Beacon Frame и кадры процедуры управления кольцом SMT Frame.
  - ZZZZ детализирует тип кадра.
- Адрес назначения (Destination Address, DA). Идентифицирует станцию (уникальный адрес) или группу станций (групповой адрес), которым предназначен кадр. Может состоять из двух или шести байт.
- Адрес источника (Source Address, SA). Идентифицирует станцию, сгенерировавшую данный кадр. Поле должно быть той же длины, что и поле адреса назначения.
- Информация (INFO). Содержит информацию, относящуюся к операции, указанной в поле управления. Поле может иметь длину от 0 до 4478 байт (от 0 до 8956 символов). Стандарт FDDI допускает размещение в этом поле маршрутной информации алгоритма Source Routing, определенной в стандарте 802.5. При этом в два старших бита поля адреса источника SA помещается комбинация 102 - групповой адрес, комбинация, не имеющая отношения к адресу источника, а обозначающая присутствие маршрутной информации в поле данных.
- Контрольная последовательность (Frame Check Sequence, FCS). Содержит 32-битную последовательность, вычисленную по стандартному методу CRC-32, принятому и для других протоколов IEEE 802. Контрольная последовательность охватывает поля FC, DA, SA, INFO и FCS.
- Конечный ограничитель (Ending Delimiter, ED). Содержит единственный символ Terminate (T), обозначающий границу кадра. За ним располагаются признаки статуса кадра.
- Статус кадра (Frame Status, FS). Первые три признака в поле статуса являются индикаторами ошибки (Error, E), распознавания адреса (Address recognized, A) и копирования кадра (Frame Copied, C). Каждый из этих индикаторов кодируется одним символом, причем нулевое состояние индикатора обозначается символом Reset (R), а единичное - Set (S). Стандарт позволяет производителям оборудования добавлять свои индикаторы после трех обязательных.

С помощью операций MAC-уровня станции получают доступ к кольцу и передают свои кадры данных. Цикл передачи кадра от одной станции к другой состоит из нескольких этапов: захвата маркера станцией, которой необходимо передать кадр, передачей одного или нескольких кадров данных, освобождением маркера передающей станцией, ретрансляцией кадра промежуточными станциями, распознаванием и копированием кадра станцией-получателем и удалением кадра из сети станцией-отправителем. Рассмотрим эти операции.

Захват маркера. Если станция имеет право захватить маркер, то после ретрансляции на выходной порт символов PA и SD маркера она удаляет из кольца символ FC, по которому она распознала маркер, а также конечный ограничитель ED. Затем она передает вслед за уже переданным символом SD символы своего кадра. Таким образом, как и прежде, она формирует новый кадр из маркера, который она захватила.

Передача кадра. После удаления полей FC и ED маркера станция начинает передавать символы кадров, которые ей предоставил для передачи уровень LLC. Станция может передавать кадры до тех пор, пока не истечет время удержания маркера.

Для сетей FDDI предусмотрена передача кадров двух типов трафика - синхронного и асинхронного. Синхронный трафик предназначен для приложений, которые требуют предоставления им гарантированной пропускной способности для передачи голоса, видеоизображений, управления процессами и других случаев работы в реальном времени. Для такого трафика каждой станции предоставляется фиксированная часть пропускной способности кольца FDDI, поэтому станция имеет право передавать кадры синхронного трафика всегда, когда она получает маркер от предыдущей

станции. Асинхронный трафик - это обычный трафик локальных сетей, не предъявляющий высоких требований к задержкам обслуживания. Станция может передавать асинхронные кадры только в том случае, если осталось неизрасходованным время удержания маркера. Каждая станция самостоятельно вычисляет текущее значение этого параметра по специальному алгоритму.

Станция прекращает передачу кадров в двух случаях: либо по истечении времени удержания маркера THT, либо при передаче всех имеющихся у нее кадров до истечения этого срока. После передачи последнего своего кадра станция формирует маркер и передает его следующей станции.

Обработка кадра станцией назначения. Станция назначения, распознав свой адрес в поле DA, начинает копировать символы кадра во внутренний буфер одновременно с повторением их на выходном порту. При этом станция назначения устанавливает признак распознавания адреса. Если же кадр скопирован во внутренний буфер, то устанавливается и признак копирования (невыполнение копирования может произойти, например, из-за переполнения внутреннего буфера). Устанавливается также и признак ошибки, если ее обнаружила проверка по контрольной последовательности.

Удаление кадра из кольца. Каждый MAC-узел ответственен за удаление из кольца кадров, которые он ранее в него поместил. Если MAC-узел при получении своего кадра занят передачей следующих кадров, то он удаляет все символы вернувшегося по кольцу кадра. Если же он уже освободил маркер, то он повторяет на выходе несколько полей этого кадра, прежде чем распознает свой адрес в поле SA. В этом случае в кольце возникает усеченный кадр, у которого после поля SA следуют символы Idle и отсутствует конечный ограничитель. Этот усеченный кадр будет удален из кольца какой-нибудь станцией, принявшей его в состоянии собственной передачи.

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

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

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

Если какая-либо станция решает начать процесс инициализации кольца по своей инициативе, то она формирует кадр Claim Token со своим значением требуемого времени оборота маркера. Захвата маркера для этого не требуется. Любая другая станция, получив кадр Claim Token, начинает выполнять процедуру Claim Token.

Для выполнения процедуры инициализации каждая станция поддерживает таймер текущего времени оборота маркера TRT (Token Rotation Timer), который используется также и в дальнейшем при работе кольца в нормальном режиме. Таймер TRT запускается каждой станцией при

обнаружении начала процедуры Claim Token. В качестве предельного значения таймера выбирается максимально допустимое время оборота маркера, то есть 165 мсек. Истечение таймера TRT до завершения процедуры означает ее неудачное окончание - кольцо не удалось инициализировать. В случае неудачи процесса Claim Token запускается процедуры, с помощью которых станции кольца пытаются выявить некорректно работающую часть кольца и отключить ее от сети.

Схематично работа процедуры Clime Token выглядит следующим образом. Каждая станция генерирует кадр Clime со своим значением T_Req, равным значению ее параметра TTRT. При этом она устанавливает значение T_Opr, равное значению TTRT. Станция, приняв кадр Claim от предыдущей станции, обязана сравнить значение T_Req, указанное в кадре со своим предложенным значением TTRT. Если другая станция просит установить время оборота маркера меньше, чем данная (то есть, T_Req < TTRT), то данная станция перестает генерировать собственные кадры Claim и начинает повторять чужие кадры Claim, так как видит, что в кольце есть более требовательные станции. Одновременно станция фиксирует в своей переменной T_Opr минимальное значение T_Req, которое ей встретилось в чужих кадрах Claim. Если же пришедший кадр имеет значение T_Req больше, чем собственное значение TTRT, то он удаляется из кольца.

Процесс Claim завершается для станции в том случае, если она получает кадр Claim со своим адресом назначения. Это означает, что данная станция является победителем состязательного процесса и ее значение TTRT оказалось минимальным. При равных значениях параметра TTRT преимущество отдается станции с большим значением MAC-адреса.

После того как станция обнаруживает, что она оказалась победителем процесса Claim Token, она должна сформировать маркер и отправить его по кольцу. Первый оборот маркера - служебный, так как за время этого оборота станции кольца узнают, что процесс Claim Token успешно завершился. При этом они устанавливают признак Ring_Operational в состояние True, означающее начало нормальной работы кольца. При следующем проходе маркера его можно будет использовать для захвата и передачи кадров данных.

Если же у какой-либо станции во время выполнения процедур инициализации таймер TRT истек, а маркер так и не появился на входе станции, то станция начинает процесс Beacon. После нормального завершения процесса инициализации у всех станций кольца устанавливается одинаковое значение переменной T_Opr.

Управление доступом к кольцу FDDI распределено между его станциями. Каждая станция, получив маркер, самостоятельно решает, может она его захватить или нет, а если да, то на какое время. Если у станции есть для передачи синхронные кадры, то она всегда может захватить маркер на фиксированное время, выделенное ей администратором. Если же у станции для передачи есть лишь асинхронные кадры, то условия захвата маркера определяются следующим образом.

Станция ведет уже упомянутый таймер текущего времени оборота маркера TRT, а также счетчик количества опозданий маркера Late_Ct. Напомним, что время истечения таймера TRT равно значению максимального времени оборота маркера T_Opr, выбранному станциями при инициализации кольца.

Счетчик Late_Ct всегда обнуляется, когда маркер проходит через станцию. Если же маркер опаздывает, то TRT-таймер достигает значения T_Opr раньше очередного прибытия маркера. При этом таймер обнуляется и начинает отсчет времени заново, а счетчик Late_Ct увеличивается на единицу, фиксируя факт опоздания маркера. При прибытии опоздавшего маркера (при этом Late_Ct = 1) TRT-таймер не сбрасывается, а продолжает считать, накапливая время опоздания маркера. Если же маркер прибыл раньше, чем истек интервал T_Opr у таймера TRT, то таймер сбрасывается в момент прибытия маркера.

Станция может захватывать маркер только в том случае, когда он прибывает вовремя - то есть если в момент его прибытия счетчик Late_Ct равен нулю.

Время удержания маркера управляется таймером удержания маркера THT (Token Holding Timer). Значение этого таймера полагается равным (T_Opr - TRT), где TRT - значение таймера TRT в момент прихода маркера. Если у станции есть в буфере кадры для передачи в момент прибытия маркера и маркер прибыл вовремя, то станция захватывает его и удерживает в течение этого периода. Для отслеживания разрешенного времени удержания маркера в момент захвата маркера значение TRT присваивается таймеру THT, а затем таймер TRT обнуляется и перезапускается. Таймер THT считает до границы T_Opr, после чего считается, что время удержания маркера исчерпано. Станция перестает передавать кадры данных и передает маркер следующей станции.

Описанный алгоритм позволяет адаптивно распределять пропускную способность кольца между станциями, а точнее - ту ее часть, которая осталась после распределения между синхронным трафиком станций.

В стандарте FDDI определено еще два механизма управления доступом к кольцу. Во-первых, в маркере можно задавать уровень приоритета маркера, а для каждого уровня приоритета задается свое время порога, до которого считает таймер удержания маркера THT. Во-вторых, определена особая форма маркера - сдерживающий маркер (restricted token), с помощью которого две станции могут монопольно некоторое время обмениваться данными по кольцу.

Если таймер TRT истечет при значении Late_Ct, равном 1, то такое событие считается потерей маркера и порождает выполнение процесса реинициализации кольца Claim Token.

Fast Ethernet.

Термином Fast Ethernet называют набор спецификаций, разработанных комитетом IEEE 802.3, чтобы обеспечить недорогой, Ethernet-совместимый стандарт, способный обеспечить работу ЛВС на скорости 100 Мбит/сек.

Из-за чего возникла необходимость в таких скоростях? К этому времени существенно увеличилась диспропорция между скоростью работы процессоров рабочих станций, скоростью работы их устройств памяти и каналов ввода/вывода, в том числе и сетевых. Эта диспропорция не позволяла эффективно использовать возможности рабочих станций в сети.

В мае 1995 года комитет IEEE принял спецификацию Fast Ethernet в качестве стандарта 802.3u, который не является самостоятельным стандартом, а представляет собой дополнение к существующему стандарту 802.3 в виде глав с 21 по 30. Отличия Fast Ethernet от Ethernet сосредоточены на физическом уровне.

Отметим главные особенности эволюционного развития от сетей Ethernet к сетям Fast Ethernet стандарта IEEE 802.3u:

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

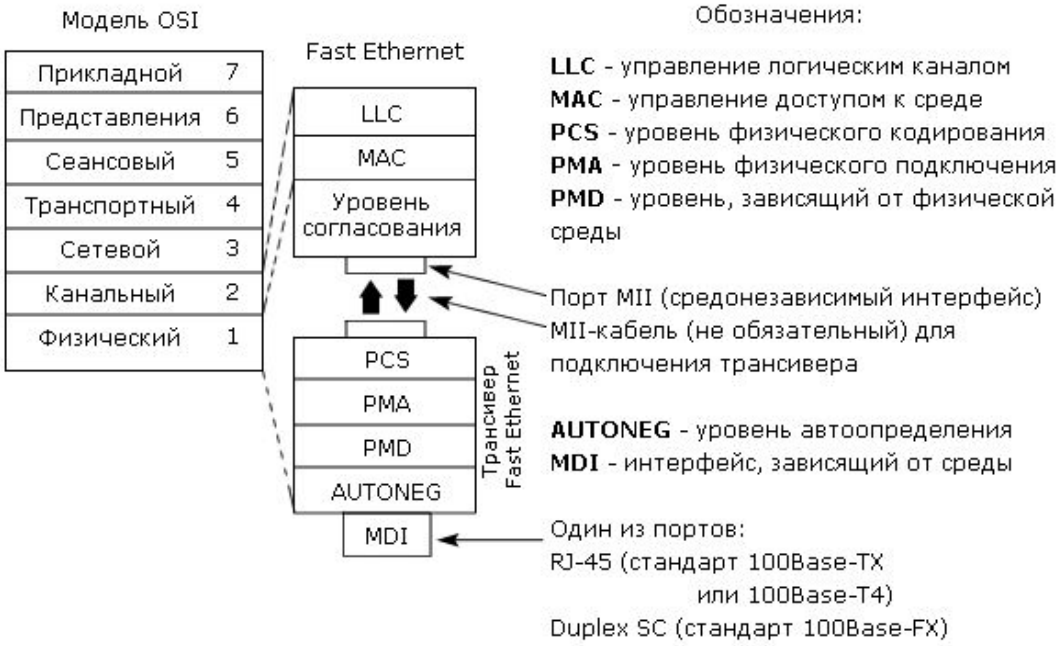
Кроме указанных свойств, важной функцией этого стандарта является поддержка двух скоростей передачи 10/100 Мбит/сек. и автоматический выбор одной из них, встраиваемая в сетевые карты и коммутаторы Fast Ethernet. Все это позволяет осуществлять плавный переход от сетей Ethernet к более скоростным сетям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

На рисунке 4-46 показана структура уровней Fast Ethernet. Более сложная структура физического уровня технологии Fast Ethernet вызвана тем, что в ней используются три варианта

кабельных систем - оптоволокно, двухпарная витая пара категории 5 и четырехпарная витая пара категории 3. Причем, по сравнению с вариантами физической реализации Ethernet (а их насчитывается шесть), здесь отличия каждого варианта от других глубже - меняется и количество проводников, и методы кодирования. А так как физические варианты Fast Ethernet создавались одновременно, а не эволюционно, то появилась возможность детально определить те подуровни физического уровня, которые не изменяются от варианта к варианту, и те, что специфичны для каждого варианта.

В стандарте Fast Ethernet функции кодирования выполняет подуровень кодирования PCS, размещенный ниже средонезависимого интерфейса MII. В результате этого каждый трансивер должен использовать свой собственный набор схем кодирования, наилучшим образом подходящий для соответствующего физического интерфейса, например, набор 4B/5B и NRZI для интерфейса 100Base-FX.

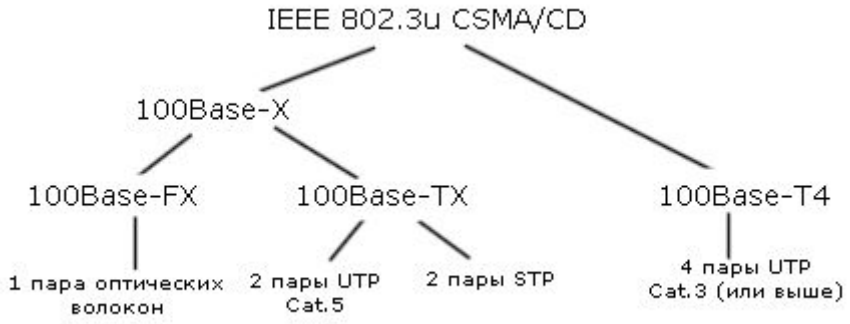
Рисунок 4-46. Структура уровней стандарта Fast Ethernet, MII-интерфейс и трансивер Fast Ethernet



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

Стандартом Fast Ethernet IEEE 802.3u установлены три типа физического интерфейса (рисунок 47, таблица 48): 100Base-FX, 100Base-TX и 100Base-T4.

Рисунок 4-47. Физические интерфейсы стандарта Fast Ethernet



100Base-FX

Стандарт этого волоконно-оптического интерфейса полностью идентичен стандарту FDDI PMD. Интерфейс Duplex SC допускает дуплексный канал связи.

100Base-TX

Стандарт этого физического интерфейса предполагает использование неэкранированной витой пары категории не ниже 5. Он полностью идентичен стандарту FDDI UTP PMD. Порт RJ-45 на сетевой карте и на коммутаторе может поддерживать наряду с режимом 100Base-TX режим 10Base-T, или функцию автоопределения скорости. Большинство современных сетевых карт и коммутаторов поддерживают эту функцию по портам RJ-45 и, кроме этого, могут работать в дуплексном режиме.

Метод кодирования 4B/5B. 10 Мбит/сек. версии Ethernet используют манчестерское кодирование для представления данных при передаче по кабелю. Метод кодирования 4B/5B определен в стандарте FDDI и без изменений перенесен в спецификацию PHY FX/TX. При этом методе каждые 4 бита данных MAC-подуровня (называемых символами) представляются 5 битами. Использование избыточного бита позволяет применить потенциальные коды при представлении каждого из пяти бит в виде электрических или оптических импульсов. Потенциальные коды обладают, по сравнению с манчестерскими кодами, более узкой полосой спектра сигнала, а, следовательно, предъявляют меньшие требования к полосе пропускания кабеля. Однако прямое использование потенциальных кодов для передачи исходных данных без избыточного бита невозможно из-за плохой самосинхронизации приемника и источника данных: при передаче длинной последовательности единиц или нулей в течение долгого времени сигнал не изменяется, и приемник не может определить момент чтения очередного бита.

При использовании пяти бит для кодирования шестнадцати исходных 4-битовых комбинаций можно построить такую таблицу кодирования, в которой любой исходный 4-битовый код представляется 5-битовым кодом с чередующимися нулями и единицами. Тем самым обеспечивается синхронизация приемника с передатчиком. Так как исходные биты MAC-подуровня должны передаваться со скоростью 100Мбит/сек., то наличие одного избыточного бита вынуждает передавать биты результирующего кода 4B/5B со скоростью 125 Мбит/сек., таким образом, межбитовое расстояние в устройстве PHY составляет 8 наносекунд.

Так как из 32 возможных комбинаций 5-битовых порций для кодирования порций исходных данных нужно только 16, то остальные 16 комбинаций в коде 4B/5B используются в служебных целях.

100Base-T4

Этот тип интерфейса позволяет обеспечить полудуплексный канал связи по витой паре UTP Cat.3 и выше. Именно возможность перехода предприятия со стандарта Ethernet на стандарт Fast Ethernet без радикальной замены существующей кабельной системы на основе UTP Cat.3 следует считать главным преимуществом этого стандарта.

Символьное кодирование 8B/6T. Если бы использовалось манчестерское кодирование, то битовая скорость в расчете на одну витую пару была бы 33,33 Мбит/с, что превышало бы установленный предел 30 МГц для таких кабелей. Эффективное уменьшение частоты модуляции достигается, если вместо прямого (2-уровневого) бинарного кода использовать 3-уровневый троичный код. Этот код, известный как 8B/6T, предполагает, что прежде, чем происходит передача, каждый набор из 8 бинарных битов (символ) сначала преобразуется в соответствии с определенными правилами в 6 троичных (3-уровневых) символов. На примере, показанном на рисунке 4-49 (b), можно определить скорость 3-уровневого символьного сигнала (100x6/8)/3=25МГц, значение которой не превышает установленный предел.

Интерфейс 100Base-T4 имеет один существенный недостаток - принципиальную невозможность поддержки дуплексного режима передачи. И если при строительстве небольших сетей Fast Ethernet с использованием повторителей 100Base-TX не имеет преимуществ перед 100Base-T4 (существует коллизионный домен, полоса пропускания которого не больше 100 Мбит/сек.), то при строительстве сетей с использованием коммутаторов недостаток интерфейса 100Base-T4 становится очевидным и очень серьезным. Поэтому данный интерфейс не получил столь большого распространения, как 100Base-TX и 100Base-FX.

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

Трансивер - это (по аналогии с трансивером Ethernet) двухпортовое устройство, охватывающее подуровни PCS, PMA, PMD и AUTONEG, и имеющее с одной стороны MII-интерфейс, с другой - один из средозависимых физических интерфейсов (100Base-FX, 100Base-TX или 100Base-T4). Трансиверы используются сравнительно редко, как и редко используются сетевые карты, повторители и коммутаторы с интерфейсом MII.

Сетевая карта. Наиболее широкое распространение сегодня получили сетевые карты с интерфейсом 100Base-TX на шину PCI. Необязательными, но крайне желательными функциями порта RJ-45 является автоконфигурирование 100/10 Мбит/сек. и поддержка дуплексного режима. Большинство современных выпускаемых карт поддерживают эти функции.

Конвертер (media converter) - это двухпортовое устройство, оба порта которого представляют средозависимые интерфейсы. Конвертеры, в отличие от повторителей, могут работать в дуплексном режиме, за исключением случая, когда имеется порт 100Base-T4. Распространены конвертеры 100Base-TX/100Base-FX.

Коммутатор - одно из наиболее важных устройств при построении корпоративных сетей. Большинство современных коммутаторов Fast Ethernet поддерживает автоконфигурирование 100/10 Мбит/с по портам RJ-45 и могут обеспечивать дуплексный канал связи по всем портам (за исключением 100Base-T4). Коммутаторы могут иметь специальные дополнительные слоты для установления uplink-модуля. В качестве интерфейсов у таких модулей могут выступать оптические порты типа Fast Ethernet 100Base-FX, FDDI , ATM (155 Мбит/сек.), Gigabit Ethernet и др.

Gigabit Ethernet.

Интерес к технологиям для локальных сетей с гигабитными скоростями повысился в связи с двумя обстоятельствами - во-первых, успехом сравнительно недорогих (по сравнению с FDDI) технологий Fast Ethernet, во-вторых, со слишком большими трудностями, испытываемыми технологией ATM на пути к конечному пользователю.

В марте 1996 года комитет IEEE 802.3 одобрил проект стандартизации Gigabit Ethernet 802.3z. В мае 1996 года 11 компаний организовали Gigabit Ethernet Alliance.

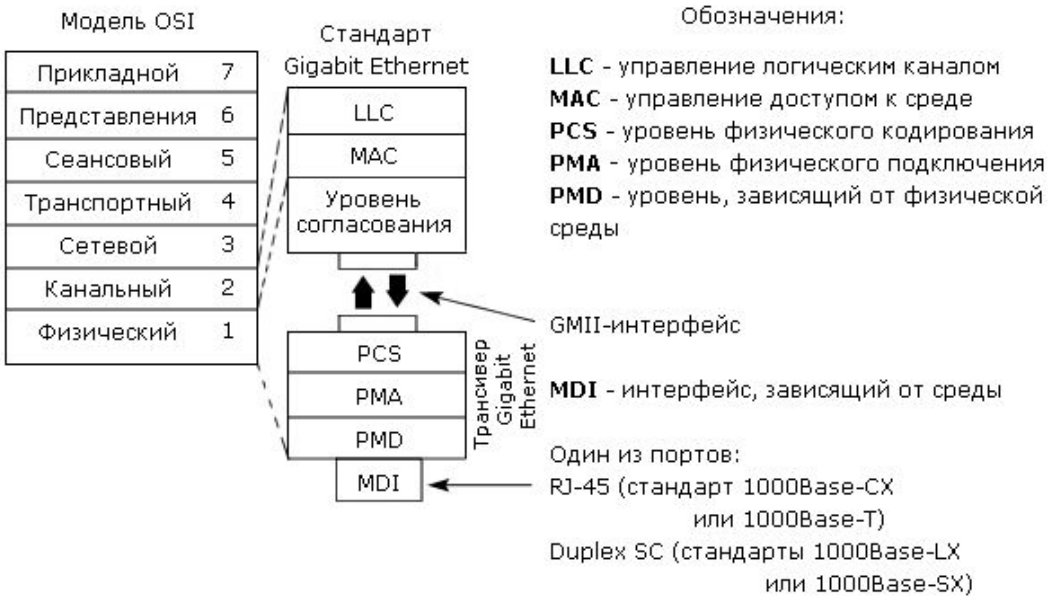
Альянс объединил усилия большого числа ведущих производителей сетевого оборудования на пути выработки единого стандарта и выпуска совместимых продуктов Gigabit Ethernet и преследовал следующие цели:

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

29 июня 1998 г., с задержкой примерно на полгода от первоначально запланированного графика, был принят стандарт IEEE 802.3z. Соответствующие спецификации регламентируют использование одномодового, многомодового волокна, а также витой пары UTP cat.5 на коротких расстояниях (до 25 м). Стандартизация системы передачи Gigabit Ethernet по неэкранированной витой паре на расстояния до 100 м требовала разработки специального помехоустойчивого кода для чего был создан отдельный подкомитет P802.3ab. 28 июня 1999 г. был принят соответствующий стандарт.

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

Рисунок 4-50. Структура уровней стандарта Gigabit Ethernet, GII-интерфейс и трансивер Gigabit Ethernet



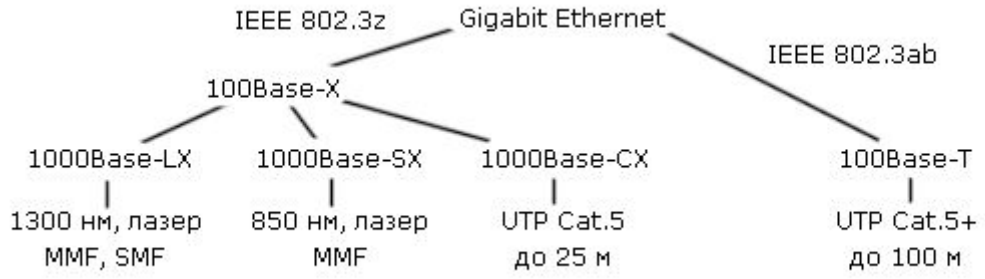
GMII-интерфейс. Средонезависимый интерфейс GMII (gigabit media independent interface) обеспечивает взаимодействие между уровнем MAC и физическим уровнем. GMII-интерфейс является расширением интерфейса MII и может поддерживать скорости 10, 100 и 1000 Мбит/сек. Он имеет отдельные 8-битные приемник и передатчик и может поддерживать как полудуплексный, так и дуплексный режимы. Кроме этого, GMII-интерфейс несет один сигнал, обеспечивающий синхронизацию (clock signal), и два сигнала состояния линии - первый (в состоянии ON) указывает наличие несущей, а второй (в состоянии ON) говорит об отсутствии коллизий. Также GMII-интерфейс обеспечивает еще несколько сигнальных каналов и питание. Трансиверный модуль, охватывающий физический уровень и обеспечивающий один из физических средозависимых

интерфейсов, может подключаться, например, к коммутатору Gigabit Ethernet посредством GMII-интерфейса.

Подуровень физического кодирования PCS. При подключении интерфейсов группы 1000Base-X подуровень PCS использует блочное избыточное кодирование 8B10B, заимствованное из стандарта ANSI X3T11 Fibre Channel. На основе сложной кодовой таблицы каждые 8 входных битов, предназначенные для передачи на удаленный узел, преобразовываются в 10-битные символы (code groups). Кроме этого, в выходном последовательном потоке присутствуют специальные контрольные 10-битные символы. Примером контрольных символов могут служить символы, используемые для расширения носителя (дополняют кадр Gigabit Ethernet до его минимального размера - 512 байт). При подключении интерфейса 1000Base-T подуровень PCS осуществляет специальное помехоустойчивое кодирование для обеспечения передачи по витой паре UTP Cat.5 на расстояние до 100 метров. Два сигнала состояния линии - сигнал наличия несущей и сигнал отсутствия коллизий - генерируются этим подуровнем.

Подуровни PMA и PMD. Физический уровень Gigabit Ethernet использует несколько интерфейсов, включая традиционную витую пару категории 5, а также многомодовое и одномодовое волокно. Подуровень PMA преобразует параллельный поток символов от PCS в последовательный поток, а также выполняет обратное преобразование (распараллеливание) входящего последовательного потока от PMD. Подуровень PMD определяет оптические/электрические характеристики физических сигналов для разных сред. Всего определено 4 различных типов физических интерфейсов среды, которые отражены в спецификация стандарта 802.3z (1000Base-X) и 802.3ab (1000Base-T) (рисунок 4-51).

Рисунок 4-51. Физические интерфейсы стандарта Gigabit Ethernet



Интерфейс 1000Base-X основан на стандарте физического уровня Fibre Channel. Эта технология будет подробнее рассмотрена ниже. Fibre Channel - это технология взаимодействия рабочих станций, суперкомпьютеров, устройств хранения и периферийных узлов. Fibre Channel имеет 4-уровневую архитектуру. Два нижних уровня FC-0 (интерфейсы и среда) и FC-1 (кодирование/декодирование) перенесены в Gigabit Ethernet. Поскольку Fibre Channel является одобренной технологией, то такое перенесение сильно сократило время на разработку оригинального стандарта Gigabit Ethernet.

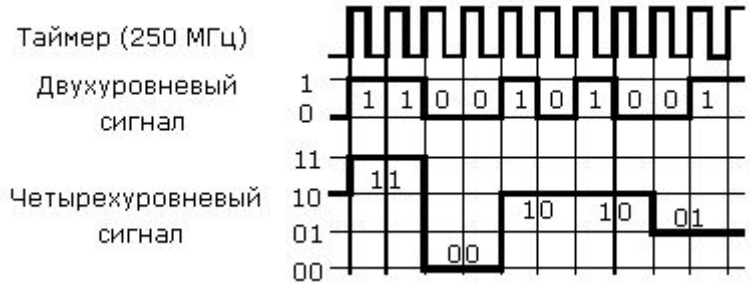
1000Base-X подразделяется на три физических интерфейса, различающихся характеристиками источника и приемника излучения: интерфейс 1000Base-SX и 1000Base-LX для многомодового оптоволокна и 1000Base-CX для экранированной витой пары (STP «twinax») на коротких расстояниях.

1000Base-T - это стандартный интерфейс Gigabit Ethernet для передачи по неэкранированной витой паре категории 5 и выше на расстояния до 100 метров. Для такой передачи используются все четыре пары медного кабеля, скорость передачи по одной паре - 250 Мбит/сек. Предполагается, что стандарт будет обеспечивать дуплексную передачу, причем данные по каждой паре будут передаваться одновременно сразу в двух направлениях (двойной дуплекс). 1000Base-T. Технически реализовать дуплексную передачу 1 Гбит/сек. по витой паре UTP cat.5 оказалось довольно сложно, значительно сложней, чем в стандарте 100Base-TX. В качестве кандидатов на утверждение в

стандарте 1000Base-T рассматривались первоначально несколько методов кодирования, среди которых 5-уровневое импульсно-амплитудное кодирование PAM-5, квадратурная амплитудная модуляция QAM-25 и др. Ниже кратко приведены идеи PAM-5, окончательно утвержденного в качестве стандарта.

Почему выбрано 5-уровневое кодирование? Распространенное четырехуровневое кодирование обрабатывает входящие биты парами. Всего существует 4 различных комбинации - 00, 01, 10, 11. Передатчик может для каждой пары бит установить свой уровень напряжения передаваемого сигнала, что уменьшает в 2 раза частоту модуляции четырехуровневого сигнала, 125 МГц вместо 250 МГц (рисунок 4-52), и, следовательно, частоту излучения. Пятый уровень добавлен для создания избыточности кода, в результате чего становится возможной коррекция ошибок на приеме.

Рисунок 4-52. Схема 4-уровневого кодирования PAM-4



Уровень MAC-стандарта Gigabit Ethernet использует тот же самый протокол передачи CSMA/CD, что и его предки Ethernet и Fast Ethernet. Основные ограничения на максимальную длину сегмента (или коллизионного домена) определяются этим протоколом. В стандарте Ethernet IEEE 802.3 принят минимальный размер кадра, равный 64 байтам. Как уже неоднократно отмечалось, именно значение минимального размера кадра определяет максимально допустимое расстояние между станциями. Время, за которое станция передает такой кадр (время канала), равно, как мы уже отмечали, 51,2 мксек. Максимальная длина сети Ethernet определяется из условия разрешения коллизий, а именно, время, за которое сигнал доходит до удаленного узла и возвращается обратно, не должно превышать 51,2 мксек. (без учета преамбулы).

При переходе от Ethernet к Fast Ethernet скорость передачи возрастает, а время трансляции кадра длины 64 байта соответственно сокращается - оно равно 5,12 мксек. Чтобы можно было обнаруживать все коллизии до конца передачи кадра, как и раньше, необходимо выполнить одно из условий:

1. Сохранить прежнюю максимальную длину сегмента, но увеличить время канала (и, следовательно, увеличить минимальную длину кадра)

2. Сохранить время канала (сохранить прежний размер кадра), но уменьшить максимальную длину сегмента

Опять же в силу преемственности, стандарт Gigabit Ethernet должен поддерживать те же минимальный и максимальный размеры кадра, которые приняты в Ethernet и Fast Ethernet. Но поскольку скорость передачи возрастает, то, соответственно, уменьшается и время передачи пакета аналогичной длины. При сохранении прежней минимальной длины кадра это привело бы к уменьшению диаметра сети, который не превышал бы 20 метров, что могло быть мало полезным. Поэтому при разработке стандарта Gigabit Ethernet было принято решение увеличить время передачи. В Gigabit Ethernet оно в 8 раз превосходит время Ethernet и Fast Ethernet. Но, чтобы поддержать совместимость со стандартами Ethernet и Fast Ethernet, минимальный размер кадра не был увеличен, зато к кадру было добавлено дополнительное поле, получившее название «расширение носителя».

Символы в дополнительном поле обычно не несут служебной информации, но они заполняют канал. В результате коллизия будет регистрироваться всеми станциями при большем диаметре коллизионного домена.

Если станции нужно передать короткий (меньше 512 байт) кадр, то при передаче добавляется поле «расширение носителя», дополняющее кадр до 512 байт. Поле контрольной суммы вычисляется только для оригинального кадра и не распространяется на поле расширения. При приеме кадра поле расширения отбрасывается. Поэтому уровень LLC даже и не знает о наличии такого поля. Если размер кадра равен или превосходит 512 байт, то поле расширения носителя отсутствует. На рисунке 4-53 показан формат кадра Gigabit Ethernet при использовании расширения носителя.

Рисунок 4-53. Кадр Gigabit Ethernet с полем расширения носителя

SDF: Start of frame Delimiter - ограничитель начала кадров
DA: Destination Address - адрес назначения
SA: Source Address - адрес источника
L: длина поля данных (для кадра 802.3)
T: тип поля данных (для кадра Ethernet II)
FCS: Frame Check Sequence - контрольная последовательность кадра

В настоящее время поставляется полный перечень сетевых продуктов Gigabit Ethernet: сетевые карты, повторители, коммутаторы, а также маршрутизаторы. Предпочтение отдается устройствам с оптическими интерфейсами.

Fibre Channel.

Fibre Channel сочетает в себе преимущества канальных и сетевых технологий. Он призван объединить в себе простоту и скорость I/O-канала с гибкостью и возможностями установления сетевых соединений на основе протоколов. Такое сочетание позволит разработчикам систем объединять традиционные подключения периферии, сетевые методы передачи данных и управления соединениями, методы соединения процессоров, используемые при создании кластеров, в единые мультипротокольные интерфейсы.

Работы по разработке стандарта FC были начаты группой ANSI в 1988 году. В настоящее время Fibre Channel конкурирует как с Ethernet, так и с SCSI (Small Computer System Interface). Последний, используемый как интерфейс к высокоскоростным устройствам рабочих станций, например, дискам, уже сейчас превосходит по быстродействию существующие сети в 10-100 раз. Fibre Channel имеет уникальную систему физического интерфейса и форматы кадров, которые позволяют этому стандарту обеспечить простую стыковку с канальными протоколами IPI (Intelligent Peripheral Interface), SCSI, HIPPI (High Performance Parallel Interface), ATM, IP и 802.2. Это позволяет, например, организовать скоростной канал между компьютером и дисковой накопительной системой RAID.

Быстродействие сетей Fibre Channel составляет nx100 Мбайт/сек. при длинах канала 10 км и более, где n – число каналов. Предусмотрена работа и на меньших скоростях (например, 12,5 Мбайт/сек.). Предельная скорость передачи составляет 4,25 Гбод. В качестве физической среды

может использоваться одномодовое или мультимодовое оптическое волокно. Допускается применение медного коаксиального кабеля и витых пар (при скоростях до 200 Мбайт/сек.).

Компоненты FC-сети делятся на простые и коммутирующие. Простой компонент имеет от одного до нескольких портов типа n_port. Эти порты используются для связи компонентов между собой. Коммутирующие компоненты представляют собой коммутаторы, которые могут быть объединены в структуры (fabric). Структуры имеют множественные порты, именуемые f_port.

Fibre Channel обеспечивает шесть независимых классов услуг (каждый класс представляет определенную стратегию обмена информацией), которые облегчают решение широкого диапазона прикладных задач:

Класс 1 Соединение с коммутацией каналов по схеме точка-точка (end-to-end) между портами типа n_port. Класс удобен для аудио- и видеоприложений, например, видеоконференций. После установления соединения используется вся доступная полоса пропускания канала. При этом гарантируется, что кадры будут получены в том же порядке, в каком они были отправлены.

Класс 2 Обмен без установления соединения с коммутацией пакетов, гарантирующий доставку данных. Так как соединение не устанавливается, порт может взаимодействовать одновременно с любым числом портов типа n_port, получая и передавая кадры. Здесь не гарантируется, что кадры будут доставлены в том же порядке, в каком были переданы (за исключением случаев соединения «точка-точка» или «кольцо с арбитражем»). В этом классе допустимы схемы управления потоком «буфер-буфер» и «точка-точка». Класс характерен для локальных сетей, где время доставки данных не является критическим.

Класс 3 Обмен дейтаграммами без установления соединения и без гарантии доставки. Схема управления потоком - «буфер-буфер». Применяется для каналов SCSI.

Класс 4 Обеспечивает выделение определенной доли пропускной способности канала с заданным значением качества обслуживания (QoS). Работает только с топологией структура (fabric), где соединяются два порта типа n_port. При этом формируются два виртуальных соединения, обслуживающих встречные потоки данных. Пропускная способность этих соединения может быть разной. Как и в классе 1, здесь гарантируется порядок доставки кадров. Допускается одновременное соединение более чем с одним портом типа n_port. Используется схема управления потоком «буфер-буфер».

Класс 5 Регламентирующие документы находятся в процессе подготовки.

Класс 6 Предусматривает групповое обслуживание в рамках топологии типа структура (fabric).

Fibre Channel использует пакеты переменной длины (до 2148 байт), содержащие до 2112 байт данных. Такая длина пакета заметно снижает издержки, связанные с пересылкой заголовков (эффективность 98%). С этой точки зрения в наихудшем положении оказывается ATM (83%-ная эффективность: 48 байт данных при 53-байтном пакете). Только FDDI превосходит FC по этому параметру (99%).

В отличие от других локальных сетей, использующих 6-октетные адреса, FC работает с 3-байтовыми адресами, распределяемыми динамически в процессе выполнения операции login. Адрес 0xffffff зарезервирован для широковещательной адресации. Адреса же в диапазоне 0xfffff0-0xfffffe выделены для обращения к «структуре» (fabric), мультикастинг-серверу и серверу псевдонимов (alias-server). n_port передает кадры от своего source_id (s_id) к destination_id (d_id). До выполнения операции fabric login s_id порта не определено. В случае арбитражного кольца применяются 3-октетные адреса al_pa, задаваемые при инициализации кольца. Для однозначной идентификации узлов используются 64-битовые имена-идентификаторы.

Формат пакетов в сетях FC показан на рисунке 4-54. Здесь используются 24-битовые адреса, что позволяет адресовать до 16 миллионов объектов. Сеть может строить соединения по схеме «точка-точка», допускается и кольцевая архитектура с возможностью арбитража. Есть и другие

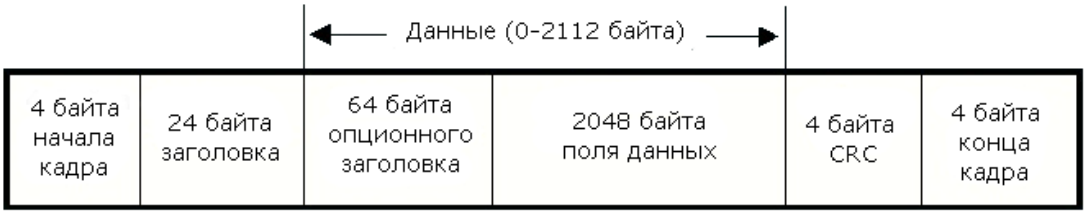
схемы, допускающие большое число независимых обменов одновременно. Схема кольцевого соединения показана на рисунке 4-55. К кольцу может быть подключено до 128 узлов.

Протокол Fibre Channel предусматривает 5 уровней, которые определяют физическую среду, скорость передачи, схему кодирования, форматы пакетов, управление потоком и различные виды услуг. Напомним, что первые два мы подробно рассмотрели в предыдущем разделе.

- FC-0 определяет физические характеристики интерфейса и среды, включая кабели, разъемы, драйверы (ECL, LED, лазеры), передатчики и приемники. Вместе с FC-1 этот уровень образует физический слой.
- FC-1 определяет метод кодирования/декодирования (8B/10B) и протокол передачи, где объединяется пересылка данных и синхронизирующей информации.
- FC-2 определяет правила сигнального протокола, классы услуг, топологию, методику сегментации, задает формат кадра и описывает передачу информационных кадров.
- FC-3 определяет работу нескольких портов на одном узле и обеспечивает общие виды сервиса.
- FC-4 обеспечивает реализацию набора прикладных команд и протоколов вышележащего уровня (например, для SCSI, IPI, IEEE 802, SBCCS, HIPPI, IP, ATM и т.д.)

FC-0 и FC-1 образуют физический уровень, соответствующей стандартной модели ISO.

Рисунок 4-54. Формат кадра Fibre Channel



Стандарт FC допускает соединение типа «точка-точка», кольцо с арбитражем и структура. Кольцевая архитектура обеспечивает самое дешевое подключение. Система арбитража допускает обмен только между двумя узлами одновременно. Следует учесть, что кольцевая структура не предполагает использования маркерной схемы доступа.

Перед передачей байты кадра преобразуются в 10-битовые кодовые последовательности, называемые символами передачи (кодировка 8B/10B).

В FC предусмотрено два режима обмена: «буфер-буфер» и «точка-точка». Передача данных осуществляется, только когда принимающая сторона готова к этому. Прежде чем что-либо посылать, стороны должны выполнить операцию login. В ходе ее выполнения определяется верхний предел числа передаваемых кадров (credit). Значение параметра credit задает число кадров, которые могут быть приняты. После передачи очередного кадра значение credit уменьшается на единицу. Когда значение этой переменной достигает нуля, дальнейшая передача блокируется до тех пор, пока получатель не обработает один или более кадров и будет готов продолжить прием. Здесь имеет место довольно тесная аналогия с протоколом скользящего окна.

Режим обмена «буфер-буфер» предполагает установление связи между портами N_Port и F_Port или между двумя N_Port. При установлении соединения каждая из сторон сообщает партнеру, сколько кадров она готова принять (значение переменной BB_Credit). Режим «точка-точка» реализуется между портами типа N_Port. Предельное число кадров, которое сторона может принять, задается переменной EE_Credit. Эта переменная устанавливается равной нулю при инициализации, увеличивается на единицу при передаче кадра и уменьшается при получении кадра ACK Link Control.

Кадр ACK может указывать на то, что порт получил и обработал один кадр, N кадров или всю последовательность кадров.

Билет № 35.

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

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

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

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

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

Справедливость значит, что все пакеты, вне зависимости от того, из какого канала они поступили, будут обслуживаться равномерно, никакому направлению не будет отдаваться предпочтение, для всех абонентов будет всегда выбираться оптимальный маршрут. Надо отметить, что справедливость и оптимальность часто могут вступать в противоречие друг с другом.

Прежде чем искать компромисс между оптимальностью и справедливостью, мы должны решить, что является критерием оптимизации. Один из возможных критериев - минимизация средней задержки пакета. Другой - максимизация пропускной способности сети. Однако эти критерии конфликтуют. Согласно теории массового обслуживания, если система с очередями функционирует близко к своему насыщению, то задержка в очереди увеличивается. Как компромисс, во многих сетях минимизируется число переходов между маршрутизаторами - один такой переход мы будем называть скачком (hop). Уменьшение числа скачков сокращает маршрут, а следовательно, сокращает задержку, а также минимизирует потребляемую пропускную способность при передаче пакета.

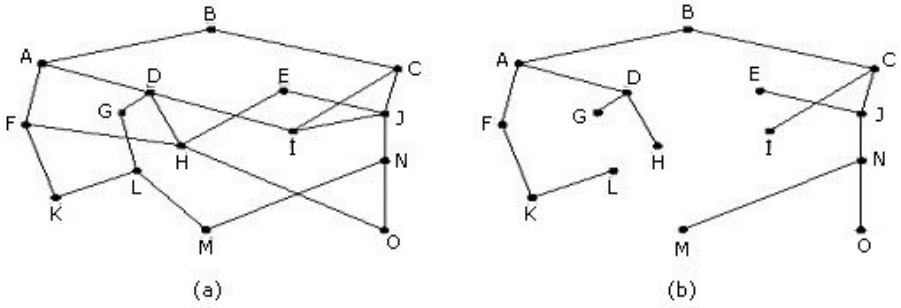
Алгоритмы маршрутизации можно разбить на два больших класса: адаптивные и неадаптивные. Неадаптивные алгоритмы не принимают в расчет текущую загрузку сети и состояние топологии. Все возможные маршруты вычисляются заранее и загружаются в маршрутизаторы при загрузке сети. Такая маршрутизация называется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

Адаптивные алгоритмы, наоборот, определяют маршрут, исходя из текущей загрузки сети и топологии. Адаптивные алгоритмы различаются тем, где и как они получают информацию (локально от соседних маршрутизаторов или глобально от всех), когда они меняют маршрут (каждые T секунд, когда меняется нагрузка, когда меняется топология), какая метрика используется при оптимизации (расстояние, число скачков, ожидаемое время передачи).

Прежде чем мы приступим к рассмотрению конкретных алгоритмов маршрутизации, сформулируем принцип оптимальности. Этот принцип утверждает, что если маршрутизатор J находится на оптимальном пути между маршрутизаторами I и K, то оптимальный маршрут между J и K принадлежит этому оптимальному пути. Это так, поскольку существование между J и K оптимального маршрута, отличного от части маршрута между I и K, противоречил бы утверждению об оптимальности маршрута между I и K. Дело в том, что если рассмотреть маршрут от I до K, как от I до J (назовем его S1) и от J до K (назовем его S2), то если между J и K есть маршрут лучше, чем S2, например S3, то маршрут S1S2 не может быть лучшим. Взяв конкатенацию маршрутов S1S3, мы получим лучший маршрут, чем маршрут S1S2.

Следствием из принципа оптимальности является утверждение, что все маршруты к заданной точке сети образуют дерево с корнем в этой точке. Это дерево называется деревом захода, оно проиллюстрировано на рисунке 5-5.

Рисунок 5-5. Дерево захода



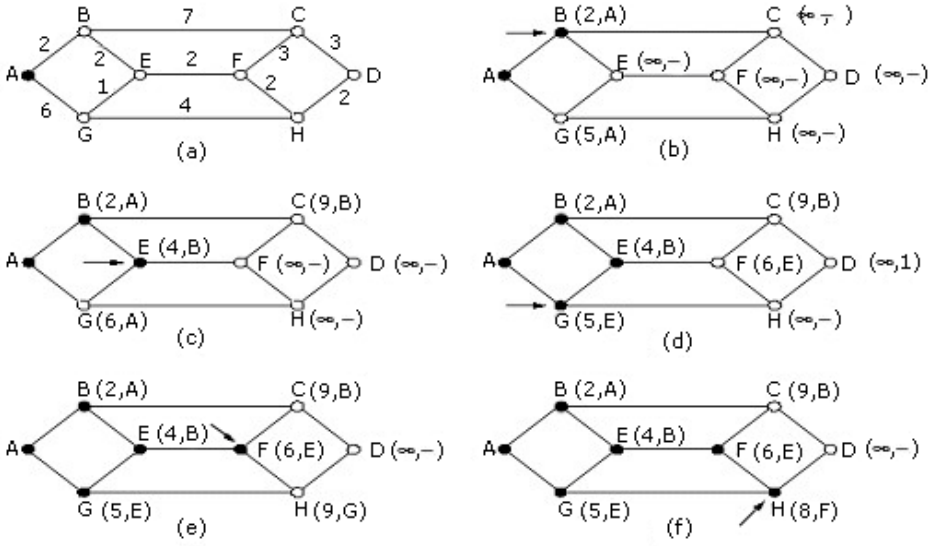
Поскольку дерево захода - это дерево, то там нет циклов, поэтому каждый пакет будет доставлен за конечное число шагов. На практике все может оказаться сложнее. Маршрутизаторы могут выходить из строя, и, наоборот, появляться новые, каналы могут выходить из строя, разные маршрутизаторы могут узнавать об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

Маршрутизация по наикратчайшему пути.

Наше изучение алгоритмов маршрутизации мы начнем со статического алгоритма, широко используемого на практике в силу его простоты. Идея этого алгоритма состоит в построении графа транспортной среды, где вершины - маршрутизаторы, а дуги - линии связи. Алгоритм находит для любой пары маршрутизаторов, а точнее абонентов, подключенных к этим маршрутизаторам, наикратчайший маршрут в этом графе.

Проиллюстрируем идею алгоритма нахождения наикратчайшего пути на рисунке 5-6 (стрелками обозначены задействованные узлы). На дугах этого графа указаны веса, которые представляют расстояние между дугами. Расстояние можно измерять в переходах, а можно в километрах. Возможны и другие меры. Например, дуги графа могут быть размечены весами, величина которых равна средней задержке пакетов в соответствующем канале. В графе с такой разметкой наикратчайший путь - наибыстрейший путь, хотя он не обязательно имеет минимальное число переходов или километров.

Рисунок 5-6. Расчет кратчайшего пути от A к B



В общем случае веса на дугах могут быть функциями от расстояния, пропускной способности канала, среднего трафика, стоимости передачи, средней длины очереди в буфере маршрутизатора к данному каналу и других факторов. Изменяя весовую функцию, алгоритм будет вычислять наикратчайший путь в смысле заданной метрики.

Известно несколько алгоритмов вычисления наикратчайшего пути в графе. Один из них предложил голландский математик Эдсгер Дейкстра. Идею этого алгоритма можно описать так. Все вершины в графе, смежные исходной вершине, помечают расстоянием (оно указано в скобках) до исходной вершины. Изначально никаких путей не известно и все вершины помечены бесконечностью. По мере работы алгоритма и нахождения путей, метки могут меняться. Метки могут быть двух видов: либо пробными, либо постоянными. Изначально все метки пробные. Когда обнаруживается, что метка представляет наикратчайший путь до исходной вершины, она превращается в постоянную метку и никогда более не меняется.

На рисунке 5-6 показан процесс построения маршрута из A в D. Помечаем вершину A как постоянную (вершина, закрашенная черным цветом). Все вершины, смежные A, помечаем как временные (эти вершины не закрашены), а также указываем в метке их вершину, из которой мы апробировали данную вершину. Это позволит нам впоследствии изменить маршрут, если надо. Кроме этого, все вершины, смежные A, помечаем расстоянием от A до этой вершины. Из всех смежных вершин мы выберем ту, расстояние до которой самое короткое, и ее объявляем рабочей. Таким образом, мы выберем на первом шаге вершину B, а затем E.

Самое интересное возникнет на шаге (d). В соответствии с принципом наикратчайшего пути мы в качестве рабочей выберем вершину G. Теперь, на шаге (e), когда мы начнем искать вершины, смежные H, то увидим, что путь F до H короче, чем от G до H. Поэтому на шаге (e) мы в качестве рабочей возьмем вершину F, а затем H. На рисунке 5-7 дано описание алгоритма Дейкстры. Надо сделать оговорку, что этот алгоритм строит наикратчайший путь, начиная от точки доставки, а не от точки отправления. Поскольку граф не ориентированный, то это никакого влияния на построение пути не оказывает.

Маршрутизация лавиной.

Другим примером статического алгоритма может служить следующий алгоритм: каждый поступающий пакет отправляют по всем имеющимся линиям, за исключением той, по которой он поступил. Ясно, что если ничем не ограничить число повторно генерируемых пакетов, то их число может расти неограниченно. Время жизни пакета ограничивают областью его распространения. Для этого в заголовке каждого изначально генерируемого пакета устанавливается счетчик переходов. При каждой пересылке этот счетчик уменьшается на единицу. Когда он достигает нуля, пакет сбрасывается и далее не посылается. В качестве начального значения счетчика выбирают наихудший случай, например, диаметр транспортной подсети.

Другим приемом, ограничивающим рост числа дублируемых пакетов, является отслеживание на каждом маршрутизаторе тех пакетов, которые через него однажды уже проходили. Такие пакеты сбрасываются и больше не пересылаются. Для этого каждый маршрутизатор, получая пакет непосредственно от абонентской машины, помечает его надлежащим числом. В свою очередь, каждый маршрутизатор ведет список номеров, сгенерированных другим маршрутизатором. Если поступивший пакет уже есть в списке, то этот пакет сбрасывается. Для предотвращения безграничного роста списка вводят ограничительную константу k. Считается, что все номера, начиная с k и далее, уже встречались.

Несмотря на кажущуюся неуклюжесть, этот алгоритм применяется, например, в распределенных базах данных, когда надо параллельно обновить данные во всех базах одновременно. Этот алгоритм всегда находит наикратчайший маршрут за самое короткое время, поскольку все возможные пути просматриваются параллельно.

Маршрутизация на основе потока.

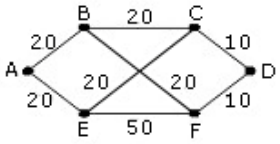
Рассмотрим статический алгоритм маршрутизации на основе потока, который учитывает как топологию, так и загрузку транспортной подсети.

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

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

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

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



(a)

Назначение

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

(b)

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

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

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

Билет № 36.

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

Маршрутизация по вектору расстояния.

Все современные транспортные подсети используют динамическую маршрутизацию, а не статическую. Один из наиболее популярных алгоритмов - маршрутизация по вектору расстояния. Этот алгоритм построен на идеях алгоритмов нахождения кратчайшего пути Беллмана-Форда и алгоритма Форда-Фолкерсона, определяющего максимальный поток в графе. Он изначально использовался в сети ARPA и используется по сей день в протоколе RIP (Routing IP).

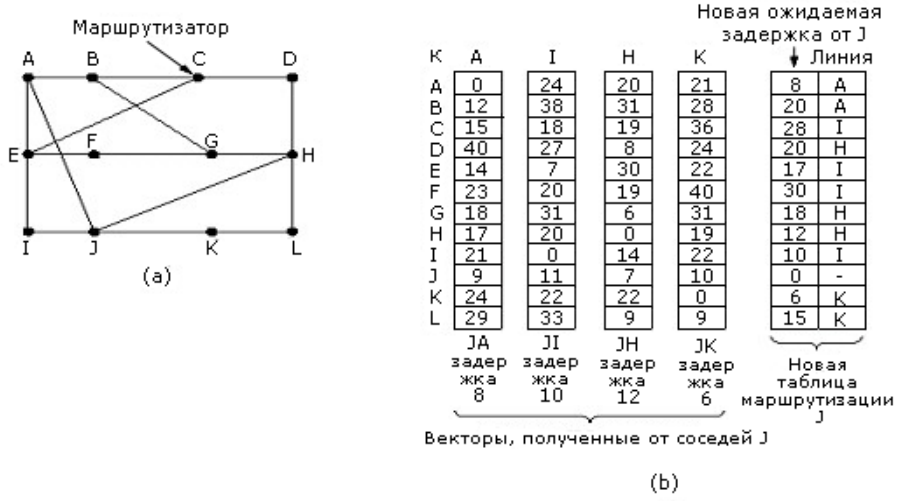
Алгоритм маршрутизации по вектору расстояния устроен следующим образом: у каждого маршрутизатора в транспортной подсети есть таблица расстояний до каждого маршрутизатора, принадлежащего подсети. Периодически маршрутизатор обменивается такой информацией со своими соседями и обновляет информацию в своей таблице. Каждый элемент таблицы состоит из двух полей: первое - номер канала, по которому надо отправлять пакеты, чтобы достичь нужного места, второе - величина задержки до места назначения. Величина задержки может быть измерена в разных единицах: числе переходов, миллисекундах, длине очереди на канале и т.д. Фактически в протоколе использовалась версия алгоритма, где эту задержку определяли не на основе пропускной способности канала, а на основе длины очереди к каналу.

Каждые T секунд маршрутизатор шлет своим соседям свой вектор задержек до всех маршрутизаторов в подсети. В свою очередь, он получает такие же вектора от своих соседей. Кроме этого, он постоянно замеряет задержки до своих соседей. Поэтому, имея вектора расстояний от соседей и зная расстояние до них, маршрутизатор всегда может вычислить кратчайший маршрут до определенного места в транспортной среде.

Рассмотрим пример на рисунке 5-10. На рисунке 5-10 (a) показана подсеть. На рисунке 5-10 (b) показаны вектора, которые маршрутизатор J получил от своих соседей, и его замеры задержек до них. Там же показана итоговая таблица маршрутизации, которую J вычислит на основании этой информации.

Рассмотрим, как маршрутизатор J с помощью этой таблицы вычислит маршрут до G. J знает, что он может достичь A за 8 мсек., A объявляет, что от него до G 18 мсек. Таким образом, J может достичь G за 26 мсек. через A. Аналогично можно подсчитать, что достичь G через I, H и K можно за 41 (31+10), 18 (6+12) и 37 (31+6) мсек. соответственно. Наилучшее значение – 18, поэтому это и есть наилучший маршрут.

Рисунок 5-10. Маршрутизация по вектору расстояния

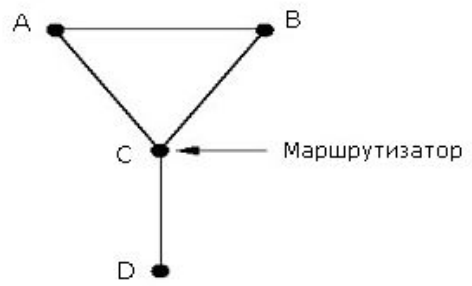


Алгоритм маршрутизации по вектору расстояния теоретически работает хорошо, но у него есть один недостаток: он очень медленно реагирует на разрушения каналов в транспортной среде. Информация о появлении хорошего маршрута в подсети распространяется более или менее быстро, а вот данные о потере, разрушении какого-то маршрута распространяются не столь быстро.

Одним из решений этой проблемы является следующий прием. Алгоритм работает так, как было описано, но при передаче вектора по линии, по которой направляются пакеты для маршрутизатора X, т.е. по которой достижим маршрутизатор X, расстояние до X указывается как бесконечность.

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

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



Маршрутизация по состоянию канала.

Алгоритм маршрутизации по вектору расстояний использовался в сети ARPANET до 1979 года, после чего он был заменен. Тому было две основных причины. Первая - пропускная способность канала никак не учитывалась, поскольку основной мерой задержки была длина очереди. Вторая проблема – медленная сходимость алгоритма при изменениях. По этим причинам был создан новый алгоритм маршрутизации по состоянию канала.

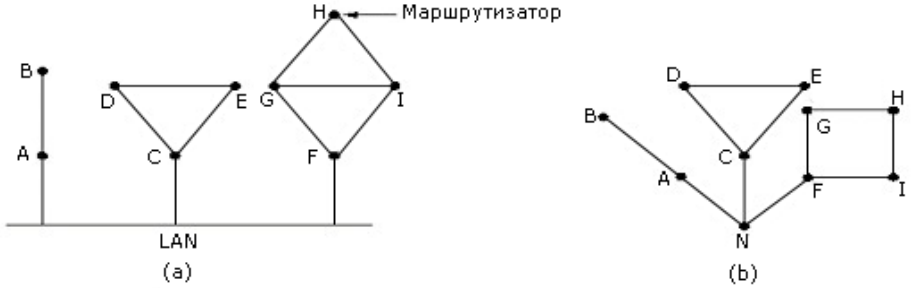
Основная идея построения этого алгоритма проста и состоит из пяти основных шагов:

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

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

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

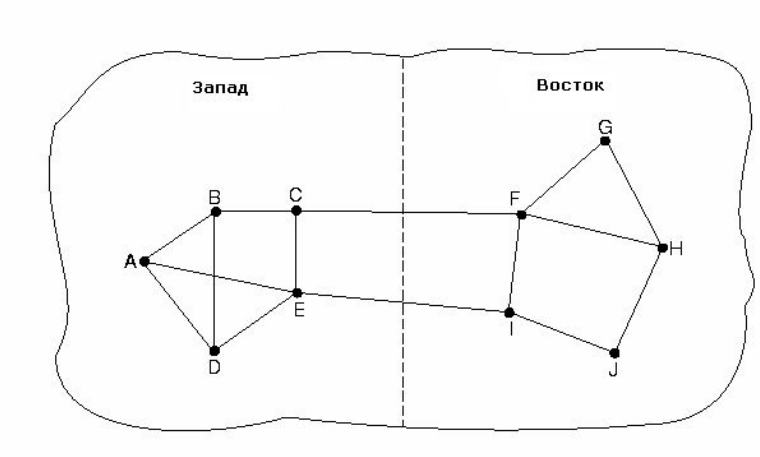
Рисунок 5-13. Определение соседей



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

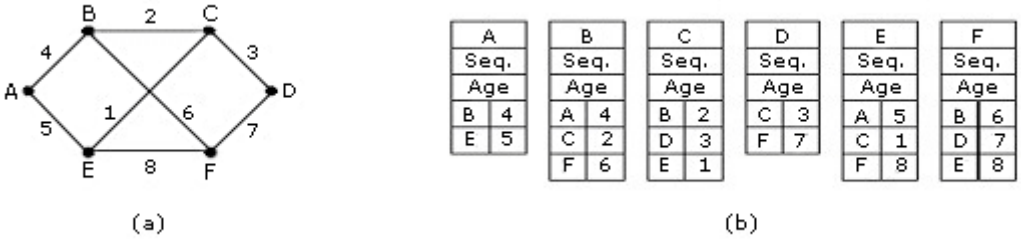
Здесь есть одна тонкость: учитывать загрузку в канале или нет? Если учитывать, то задержку надо замерять от момента поступления пакета в очередь к каналу. Если не учитывать, то от момента, когда пакет достиг головы очереди. Есть доводы, как в пользу учета нагрузки, так и против такого учета. Учитывая нагрузку, мы можем выбирать между двумя и более каналами с одинаковой пропускной способностью, получая лучшую производительность. Однако можно привести примеры, когда ее учет вызывает проблемы. Например, рассмотрим рисунок 5-13 (c). Здесь два одинаковых по производительности канала CF и EI соединяют две сети. Если на одном из них нагрузка меньше, то предпочтительнее будет другой, что через некоторое время приведет к его перегрузке и предпочтительнее должен стать первый. Если нагрузка не учитывается, то этой проблемы не возникает.

Рисунок 5-13 (c). Определение затрат



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

Рисунок 5-14. Формирование пакетов состояния канала



Наиболее хитрая часть этого алгоритма – как надежно распространить пакеты о состоянии каналов (СК-пакеты)? Как только СК-пакет получен и включен в работу, маршрутизатор будет его использовать при определении маршрута. При неудачном распространении СК-пакетов разные маршрутизаторы могут получить разное представление о топологии транспортной среды, что может приводить к возникновению циклов, недостижимых машин и другим проблемам.

Сначала рассмотрим базовый алгоритм, потом некоторые его усовершенствования. СК-пакеты распространяются методом лавины, т.е. СК-пакет рассылается всем соседям, те, в свою очередь, своим соседям, и т.д. Однако, чтобы не потерять контроль и не вызвать неограниченное дублирование СК-пакетов, каждый маршрутизатор ведет счетчик последовательных номеров СК-пакетов, которые он сгенерировал. Все маршрутизаторы запоминают пары «маршрутизатор, последовательное число», которые они уже встречали среди полученных СК-пакетов. Если поступивший СК-пакет содержит пару, которая еще не встречалась маршрутизатору, то он отправляет этот СК-пакет всем своим соседям, за исключением того, от которого он его получил. Если он уже встречал такой пакет, то пакет сбрасывается и никуда не дублируется.

У этого алгоритма есть несколько проблем, но все они разрешимые. Первая: размер поля последовательных номеров пакетов. Если оно будет недостаточно длинное, то его переполнение приведет к повтору номеров, что, в свою очередь, приведет к некорректной работе всего алгоритма. Решением является достаточно большое поле, например, 32-разрядное. При таком поле, если обмен СК-пакетами происходит раз в секунду, то потребуется 137 лет, чтобы возникло переполнение.

Вторая проблема: если маршрутизатор «упал» по какой-либо причине и потерял последовательность использованных последовательных номеров, то неясно, как ее восстановить.

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

Для решения этих проблем используется поле «возраст» СК-пакета. Там устанавливается некоторая величина, которая уменьшается периодически на единицу каждую секунду. Когда она достигнет нуля, пакет сбрасывается.

В целях сокращения числа рассылаемых СК-пакетов, когда такой пакет поступает, его не сразу дублируют и отправляют. Сначала его помещают в специальную область задержки. Там он находится некоторое время. Если за это время придет другой пакет от того же источника, то пакеты сравниваются. Если нет различий между ними, то вновь пришедший сбрасывается, если есть, то последний пришедший дублируется и отправляется другим, а первый сбрасывается. Все СК-пакеты передаются с уведомлением.

Структура данных, используемая маршрутизатором B из примера на рисунке 5-14 (а), показана на рисунке 5-15. Каждая строка в этой таблице соответствует последнему поступившему, но не до конца обработанному СК-пакету. В этой таблице для каждой линии маршрутизатора B указано в виде флага, был ли соответствующий СК-пакет отправлен и подтвержден. Флаг отправления означает, что соответствующий СК-пакет должен быть послан по указанной линии. Флаг уведомления указывает на то, что по соответствующей линии должно прийти подтверждение.

Рисунок 5-15. Буфер пакетов для маршрутизатора B из рисунка 5-14

Источник	Номер	Возраст	Флаги отправления A	C	F	Флаги уведомления 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	

На рисунке 5-15 СК-пакет от A прибыл и должен быть переслан в C и F и подтвержден по A. Аналогичная ситуация с СК-пакетом от F. Существенно отличная ситуация с СК-пакетом от E. От E поступило два пакета. Один - по маршруту EAB, другой – по маршруту EFB. Следовательно, СК-пакет должен быт послан только к C, а вот уведомления должны быть посланы и A, и F. Наличие дубликатов СК-пакетов легко распознать по состоянию флагов отправки.

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

В системе, где есть n маршрутизаторов с k линий у каждого, каждый маршрутизатор должен иметь достаточно памяти, чтобы хранить необходимую информацию о сети. При больших n эта величина может стать существенной. Кроме этого, в сетях, где число маршрутизаторов достигает десятков или сотен тысяч, проблемы, вызванные сбоем одного из них, могут оказаться весьма серьезными. Например, если из-за сбоя в маршрутизаторе послан неправильный СК-пакет или неправильно оценено состояние канала, то в дальнейшем это может привести к проблемам.

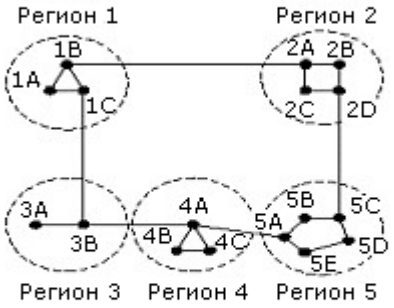
Маршрутизация по состоянию каналов широко используется в реальных сетях. Например, в протоколе OSPF, который мы будем подробно рассматривать позже. Другим важным примером может служить протокол IS-IS, который был изначально предложен фирмой DEC, а позже адаптирован ISO.

Иерархическая маршрутизация.

По мере роста транспортной среды размер таблиц, т.е. затраты памяти, время процессора на обработку этих таблиц, пропускная способность каналов, затрачиваемая на передачу служебной информации, может превысить разумные пределы. Таким образом, дальнейший рост сети, когда каждый маршрутизатор знает все о каждом другом маршрутизаторе, будет невозможен. Решение этой проблемы – иерархия сетей, подобно иерархии коммутаторов в телефонной сети.

На рисунке 5-16 (a) приведен пример сети с двухуровневой иерархией из пяти регионов. Здесь каждый из регионов соединен с двумя другими. На рисунке 5-16 (b) показана полная таблица маршрутизации для маршрутизатора 1A без иерархии. На рисунке 5-16 (c) – та же таблица при двухуровневой иерархии. Нетрудно видеть, что таблица маршрутизации во втором случае резко сократилась.

Рисунок 5-16. Иерархическая маршрутизация



(a)

Полная таблица

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	1C	5
5C	1B	5
5D	1C	5
5E	1C	5

(b)

Иерархическая таблица

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

(c)

Однако за эту экономию приходится платить эффективностью маршрутизации. Например, наилучший маршрут от 1A к 5C проходит через регион 2, однако в данном случае он пройдет через регион 3, поскольку большинство машин в регионе 5 действительно эффективнее достигнуть через регион 3.

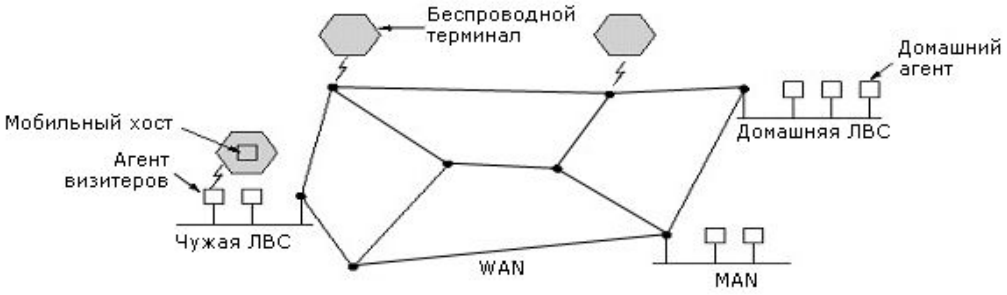
При построении иерархии возникает сразу несколько вопросов. Один из них: сколько уровней должно быть в иерархии при заданном размере сети? Например, если наша транспортная среда содержит 920 маршрутизаторов, то без иерархии таблица каждого будет иметь 920 строк. Если транспортную среду разбить на 40 регионов, то размер таблиц будет равен 23 строки для маршрутизации внутри региона плюс 40 для маршрутизации между регионами. Если использовать трехуровневую иерархию и объединить регионы в кластеры, то при 5 кластерах, по 8 регионов с 23 маршрутизаторами в каждом у таблицы будет 23 входа для внутрикластерной маршрутизации, плюс 7 входов для межкластерной, плюс 5 входов для межрегиональной маршрутизации. Итого 35 входов.

Клейнрок и Камоун показали, что оптимальное число уровней иерархии в транспортной среде при N узлах будет равняться lnN, при e*lnN строках в таблице маршрутизатора.

Маршрутизация для мобильного узла.

На рисунке 5-17 показана модель WAN с мобильным узлом. Всех пользователей мы можем разделить на две большие группы. Стационарные - это большая группа, их компьютеры подключены к сети стационарными средствами (проводами, кабелями) и редко меняют свое место положение. Другая группа постоянно меняет свое местоположение и стремится поддерживать связь с сетью. Этих пользователей мы будем называть мобильными.

Рисунок 5-17. Модель WAN с мобильным узлом



Предполагается, что в сети каждый пользователь имеет постоянное домашнее местоположение, которое никогда не меняется. Проблема маршрутизации в этих условиях заключается в том, чтобы посылать пакеты мобильному пользователю через его домашнее местоположение. При этом то, где находится сам пользователь, не имеет значения. Вся WAN на рисунке 5-17 разбивается на области. В каждой области есть агент визитеров, который знает о всех мобильных пользователях в своей области. В свою очередь, в каждой области есть домашний агент, который знает обо всех стационарных пользователях в своей области, которые в настоящий момент путешествуют.

Как только мобильный узел подключается к местной, локальной сети, он регистрируется у агента визитеров. Эта процедура примерно выглядит так:

1. Периодически агент визитеров рассылает по своей области пакет, где указано местоположение этого агента и его адрес. Если мобильный узел, подключившись к сети, долго не видит такого пакета, он рассылает свой пакет с просьбой агенту визитеров объявить свои координаты.

2. Мобильный узел регистрируется у агента визитеров, указывая свое текущее местоположение, домашнее местоположение и определенную информацию, связанную с безопасностью передаваемых данных.

3. Агент визитеров обращается через сеть к домашнему агенту домашнего местоположения визитера, указывая, что один из его пользователей сейчас находится в его области, передавая конфиденциальную информацию, которая должна убедить домашнего агента, что это действительно его пользователь пытается соединиться с ним.

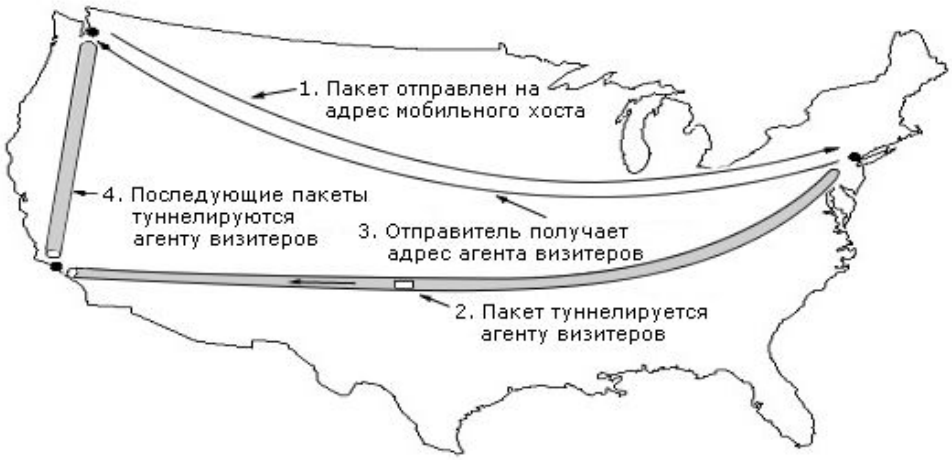
4. Домашний агент изучает конфиденциальные данные, время связи. Если эти данные соответствуют той информации, что есть у домашнего агента об этом пользователе, он дает добро на связь.

5. Агент визитеров, получив подтверждение от домашнего агента, заносит данные о мобильном узле в свои таблицы и регистрирует его.

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

Рассмотрим теперь, что происходит, когда кто-то посылает сообщения мобильному узлу (рисунок 5-18). Пакет поступает на адрес домашнего местоположения пользователя, где его перехватывает домашний агент. Домашний агент инкапсулирует этот пакет в свой пакет, который он отправляет по адресу агента визитеров той области, откуда последний раз был сеанс связи с пользователем. Одновременно с этим домашний агент посылает сообщение отправителю пакета, чтобы он все последующие пакеты мобильному узлу инкапсулировал в сообщениях, направляемых по адресу агента визитеров. Такой механизм инкапсулирования одних пакетов в другие называется туннелированием, и мы его подробно рассмотрим позднее.

Рисунок 5-18. Маршрутизация пакетов для мобильных пользователей



Здесь мы обрисовали лишь в общих чертах основную схему работы. Конкретных схем существует множество, которые различаются разными аспектами. Прежде всего тем, как распределяется работа между маршрутизаторами и хостами, какой уровень в стеке протоколов хоста отвечает за реализацию соответствующих протоколов. Во-вторых, есть схемы, где маршрутизаторы запоминают информацию о местонахождения мобильных узлов и могут вмешиваться в диалог между агентом визитеров и домашним агентом, по-разному маршрутизируя трафик. В некоторых схемах мобильный узел получает некоторый уникальный адрес, в других это адрес агента, который отвечает за маршрутизацию всего трафика мобильных узлов. Кроме этого, схемы различаются разным уровнем безопасности передаваемой информации.

Маршрутизация при вещании.

В некоторых приложениях возникает потребность переслать одно и то же сообщение всем машинам. Например, прогноз погоды, биржевые сводки, новости и т.д. Такой режим передачи называется вещанием. Есть несколько способов реализации такого режима.

Первый способ: источник знает, кому надо послать, и генерирует столько сообщений, сколько получателей. Это одно из самых плохих решений. Оно не требует никаких специальных средств, однако весьма накладно. Тратится не только пропускная способность каналов, но и память – ведь где-то кому-то надо хранить весь лист рассылки.

Метод лавины – другое решение. Однако, как мы уже видели, он затратен для каналов «точка-точка». Он слишком сильно расходует пропускную способность.

Третий подход – маршрутизация множественной доставки. Здесь каждый пакет должен иметь либо лист рассылки, либо карту рассылки. Каждый маршрутизатор, получив такой пакет, отправляет и дублирует его в соответствии с картой рассылки.

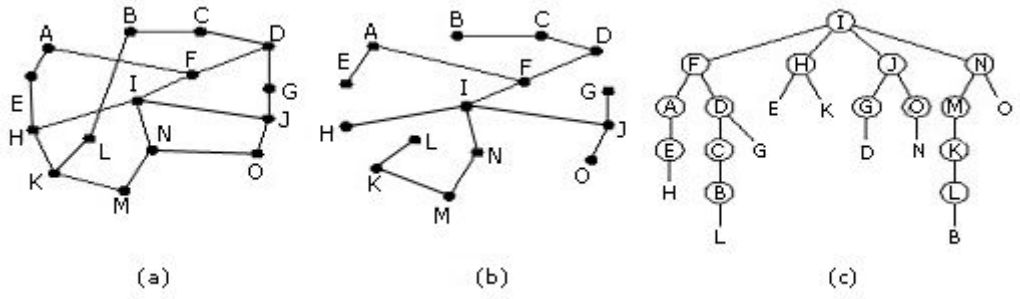
Четвертый подход основан на использовании дерева захода, либо любого другого подходящего дерева связей. Дерево захода позволяет избежать циклов и ненужного дублирования пакетов. Каждый маршрутизатор дублирует пакет вдоль линий, соответствующих дереву захода, кроме той, по которой пакет пришел. В этом подходе очень рационально используется пропускная способность каналов, генерируется абсолютный минимум пакетов при рассылке. Однако каждый маршрутизатор где-то должен иметь дерево захода, соответствующее случаю.

Пятый подход основан на неявном использовании дерева связей. Когда пакет поступает, маршрутизатор проверяет: если он поступил по линии, которая используется для отправления

пакетов источнику вещательного пакета, то вещательный пакет дублируется и рассылается по всем линиям, кроме той, по которой пакет пришел. Если нет, то он сбрасывается.

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

Рисунок 5-19. Пересылка вдоль обратного пути (a) Подсеть; (b) Связующее дерево; (c) Дерево, построенное методом пересылки вдоль обратного пути



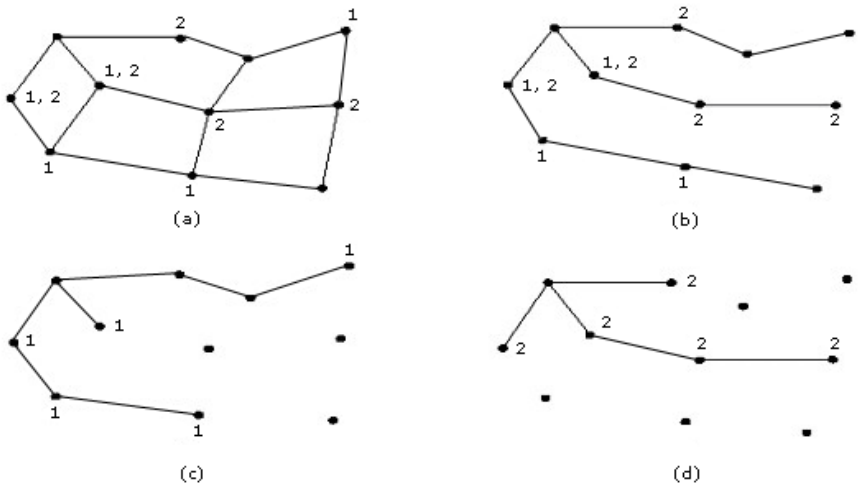
Маршрутизация при групповой передаче.

Этот вид передачи используют, когда надо обеспечить взаимодействие группы взаимосвязанных процессов, разбросанных по сети. Такие ситуации часто встречаются в распределенных базах данных. Если группа велика по сравнению с размерами сети, то можно воспользоваться методами вещания. Однако если ее размер мал относительно размера всей сети, то такой подход будет неэкономичен. Кроме того, если рассылаемая в группе информация конфиденциальная, то алгоритмы вещания не подходят. Этот вид маршрутизации называют групповой маршрутизацией.

В случае обмена информации в группе, кроме алгоритма групповой маршрутизации, нужны алгоритмы управления группой, которые должны обеспечивать средства для реконфигурации группы: включение новых членов, удаление старых и т.п. Однако эти проблемы не затрагивают алгоритм групповой маршрутизации, поэтому мы их здесь рассматривать не будем.

Алгоритм групповой маршрутизации, как правило, основан на дереве связей. Каждый маршрутизатор в транспортной среде вычисляет дерево связей, охватывающее все другие маршрутизаторы. На рисунке 5-20 (a) приведен пример транспортной среды, состоящей из двух групп (их номера указаны у вершин). Некоторые вершины принадлежат как группе 1, так и группе 2. На рисунке 5-20 (b) показано дерево связей для самой левой вершины. Когда процесс посылает групповой пакет, первый же маршрутизатор обрезает свое дерево связей, убирая из него все связи, которые не ведут в вершины, не являющиеся членами группы. На рисунке 5-20 (c) показано сокращенное дерево связей для группы 1. На рисунке 5-20 (d) дано сокращенное дерево связей для группы 2.

Рисунок 5-20. Групповая маршрутизация



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

Билет № 37.

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

Когда в транспортной среде находится в одно и тоже время слишком много пакетов, ее производительность начинает падать. Когда число пакетов, отправляемых абонентскими машинами в сеть, пропорционально возможностям сети, то число посланных пакетов пропорционально числу доставленных пакетов. Однако если число пакетов, поступающих в сеть, становится слишком большим, они начинают пропадать. При перегрузке сети может случиться, что доставка пакетов практически прекратится.

Перегрузка может возникнуть в силу нескольких причин. Например, если сразу несколько потоков, поступающих по нескольким входным линиям, устремятся на одну и ту же выходную линию. Очередь на этой линии может расти бесконечно, и пакеты начнут посылаться повторно, так как они слишком долго будут находиться в очереди. Если буфер маршрутизатора переполнится, то пакеты начнут теряться. Увеличение памяти в этом случае вряд ли исправит положение. Пакеты долго будут находиться в памяти, и отправители начнут их дублировать.

Перегрузки могут случаться и из-за недостаточной скорости процессора. Если процессор будет не в состоянии справиться своевременно с рутинными задачами (размещения пакета в буфере, корректировка таблиц и т.п.), то даже при наличии линий с достаточной пропускной способностью очередь будет расти. Аналогичная картина может случиться при быстром процессоре, но медленном канале и наоборот. Таким образом, источник проблемы - несбалансированность производительности компонентов системы.

Перегрузки имеют тенденцию к самостоятельному росту и ухудшению ситуации. Если у маршрутизатора не хватает памяти буфера, то он начинает сбрасывать пакеты. Отправитель, не получая пакеты, начинает их повторять снова и снова, усугубляя положения получателя.

Надо различать управление перегрузками сети и управление потоком. Перегрузка - это глобальная проблема в сети. Управление перегрузками - это такая организация потоков в транспортной среде, при которой потоки соответствуют пропускной способности подсети и не превышают ее. Это глобальная проблема в сети, затрагивающая поведение всех хостов и всех маршрутизаторов.

Управление потоком возникает между парой взаимодействующих машин. Это локальная проблема, касающаяся двух взаимодействующих машин. Ее решение гарантирует, что быстрый отправитель сообщений не «завалит» нерасторопного получателя. Здесь яркими примерами могут быть: один быстрый компьютер передает файл в 1 Гб более медленному компьютеру через сеть с пропускной способностью 1 Тбит/сек. со скоростью 1 Гбит/сек. Ясно, что здесь не будет перегрузки, хотя быстрый компьютер может создать такой поток пакетов, что он захлестнет медленный. В тоже время, если в сети с линиями на 1 Мбит/сек. и 1000 компьютеров хотя бы половина машин начнет передавать файлы со скоростью 100 Кбит/сек. другой половине, то ясно, что перегрузки не избежать.

Часто управление перегрузкой и управление потоком путают из-за того, что и там и там применяют одинаковые приемы, например, направляют источникам специальные пакеты, тормозящие нарастание потоков.

Основные принципы управления перегрузками.

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

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

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

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

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

Следующий шаг при использовании обратной связи - передать информацию о перегрузке туда, где что-то может быть сделано, чтобы исправить положение. Например, маршрутизатор, обнаруживший перегрузку, может направить сообщение о перегрузке всем источникам сообщений. Ясно, что это увеличит нагрузку в сети, причем именно в тот момент, когда это менее всего желательно. Однако есть и другие возможности. Например, в каждом пакете зарезервировать специальный бит перегрузки, и если какой-то маршрутизатор обнаружил перегрузку, то он устанавливает этот бит, тем самым сообщая другим о ней (вспомним структуру кадра во Frame Relay).

Другое решение напоминает прием, используемый некоторыми радиостанциями: направлять несколько автомашин по дорогам, чтобы обнаруживать пробки, а затем сообщать о них по

радиоканалам, предупреждая другие машины, призывая их пользоваться объездными путями. По аналогии с этим решением в сети рассылаются специальные пробные пакеты, которые проверяют нагрузку, и если где-то обнаружена перегрузка, то о ней сообщается всем и происходит перенаправление пакетов так, чтобы обогнуть перегруженные участки.

Алгоритмы управления перегрузками подразделяются, как мы уже сказали, на решения с открытым и решения с закрытым контуром. Решения с открытым контуром, в свою очередь, делятся на две группы: воздействующие на источники и воздействующие на получателей. Решения с закрытым контуром - с явной обратной связью и неявной обратной связью. Явная обратная связь предполагает, что источнику посылается специальный пакет, который информирует его о перегрузке. Неявная обратная связь основана на том, что источник сам определяет факт перегрузки на основе своих локальных наблюдений за трафиком, например, по величине задержки на поступление уведомления о доставке пакета.

Появление перегрузки означает, что нагрузка превысила, возможно временно, ресурсы системы или некоторой ее части. Есть два выхода из этого положения: увеличить ресурсы и сократить нагрузку. Увеличить ресурсы чаще всего невозможно. Тогда остается только сокращение нагрузки. Для этого есть несколько способов: отказать некоторым пользователям в сервисе, ухудшить сервис всем или некоторым пользователям, заставить пользователей планировать свои потоки определенным образом.

Методы, предотвращающие перегрузки.

Рассмотрение методов, предотвращающих перегрузки, начнем с методов для систем с открытым контуром. Эти методы ориентированы на минимизацию перегрузок при первых признаках их проявлений, а не на борьбу с перегрузками, когда они уже случились.

Таблица 5-22. Факторы, влияющие на перегрузки

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

	Управление потоком

Начнем с канального уровня. Вызвать перегрузку может повторная пересылка кадров. Если у источника сообщений часто возникает time_out и он начинает повторно передавать пакет, то тем самым он лишь усугубляет положение. Близко к этому фактору стоит нарушение порядка следования пакетов при передаче. Если получатель часто сбрасывает пакеты, поступившие не в надлежащем порядке от источника, то их повторная передача будет также усугублять перегрузку.

Организация рассылки уведомлений также влияет на перегрузку. Если уведомление происходит немедленно и специальными пакетами, то это увеличивает трафик и следовательно может привести к перегрузкам. Если для уведомления используются пакеты с сообщениями (прием piggybacking), то возможны time_out из-за отсутствия уведомлений вовремя и, как следствие, повторные пересылки пакетов, что может привести к перегрузкам. В то же время жесткая схема управления потоком (небольшое окно) сдерживает нарастание трафика и предотвращает появление перегрузок.

На сетевом уровне выбор схемы работы - с виртуальными соединениями или дейтаграммами - влияет на появление перегрузок. На этом уровне большинство методов борьбы с перегрузками ориентированы на виртуальные соединения. Методы управления очередями, организация очередей: одна общая на входе или одна общая на выходе; по одной на каждую входную линию или на каждую выходную; по одной очереди на каждую входную и выходную - все это влияет на появление перегрузок. Выбор метода сброса пакетов также влияет на перегрузки.

Правильная маршрутизация, равномерно использующая каналы в транспортной среде, позволяет избежать перегрузки. Методы, регулирующие время жизни пакета в сети, также влияют на образование перегрузок. Если пакет долго блуждает в сети, прежде чем будет принято решение о его сбросе, то это плохо, так как увеличивает трафик и может привести к перегрузке. Если поторопиться, то преждевременный сброс пакета может привести к повторным передачам, что опять-таки увеличит нагрузку.

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

Формирование трафика.

Одной из основных причин перегрузки является нерегулярный, взрывообразный трафик в сети. Если бы он был равномерным, то перегрузок можно было бы избежать. Один из методов с открытым контуром, часто используемым особенно в АТМ-сетях, - метод формирования трафика (shaping - т.е. придание формы), когда скорость передачи пакетов контролируется и регулируется.

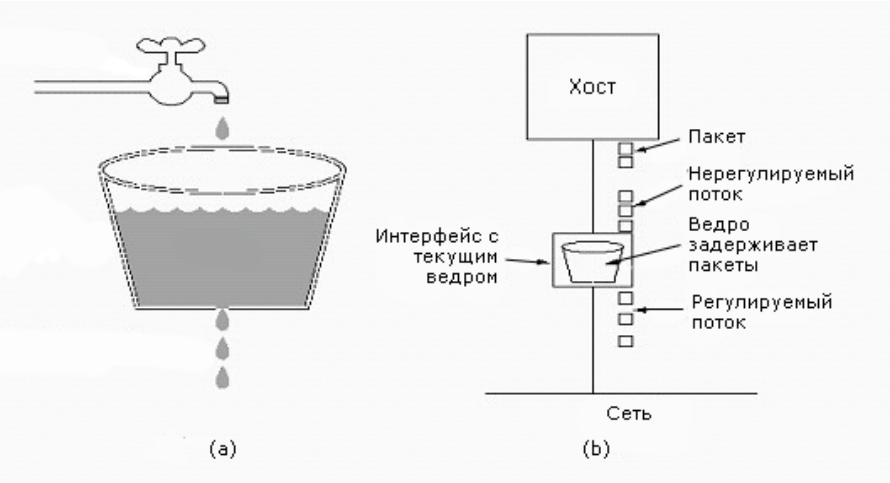
Формирование трафика регулирует среднюю скорость передачи данных так, чтобы сделать его по возможности гладким. Следует обратить внимание, что протокол скользящего окна, который мы рассматривали при изучении канального уровня, лишь регулирует объем данных, передаваемых за один раз, но не скорость передачи. Здесь же речь идет именно о скорости передачи. Когда виртуальное соединение устанавливается, то пользователь договаривается с транспортной средой о форме трафика. Если пользователь обеспечивает договоренную форму трафика, то транспортная среда обеспечивает ему доставку трафика с определенной скоростью. Для таких приложений, как передача видео- и аудиоданных в реальном времени, это очень важно. Здесь уместно вспомнить организацию работы СПД Frame Relay.

Когда пользователь и транспортная среда договариваются о форме трафика, то они приходят к соглашению не только о форме трафика, но также и о том, что произойдет, если эта форма будет нарушена пользователем. Это соглашение называется соглашением о трафике. Использование

техники формирования трафика и соглашения о трафике легче всего реализовать при использовании виртуальных соединений, чем в случае дейтаграмм. В случае дейтаграмм эти идеи могут быть применены к соединениям на транспортном уровне.

Алгоритм текущего ведра. Идея этого алгоритма показана на рисунке 5-23 (a). Ведро может наполняться с любой скоростью, но вытекать из него вода будет со строго определенной скоростью. Если вода будет поступать слишком быстро, то ее часть будет переливаться через края и пропадать. Скорость истечения воды из ведра зависит только от размера отверстия в днище.

Рисунок 5-23. Алгоритм текущего ведра

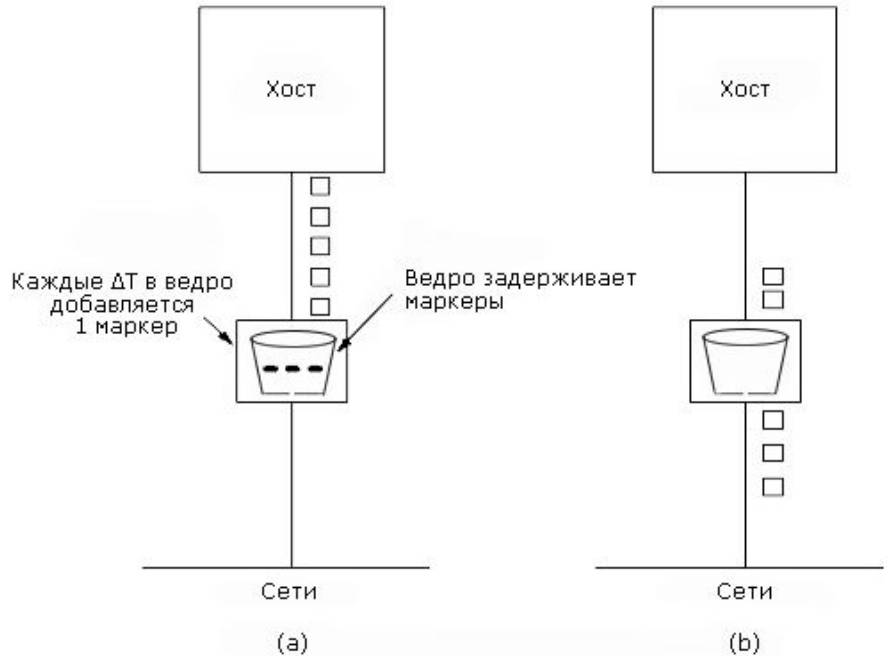


Этот прием можно применить и к пакетам, как показано на рисунке 5-23 (b). Каждая станция, подключенная к сети, имеет подобие текущего ведра в своем интерфейсе. Не важно, сколько процессов посылает пакеты в сеть. Если буфер переполнен, то пакеты будут сбрасываться в соответствии с соглашением о трафике. Это не что иное, как сервер с постоянной скоростью обслуживания.

В качестве регулятора скорости поступления пакетов можно использовать системные часы. В этом случае устанавливается предел числа пакетов, которые процесс может направить в сеть за один промежуток времени. Этот прием дает хорошие результаты, когда пакеты имеют фиксированную длину, как в ATM. В случае пакетов переменной длины это соглашение ограничивает количество байтов, направляемых в сеть.

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

Рисунок 5-25. Алгоритм ведра с маркерами



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

$$C + \rho S = MS$$

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

Таким образом, $S = C/(M - \rho)$.

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

Спецификация потока.

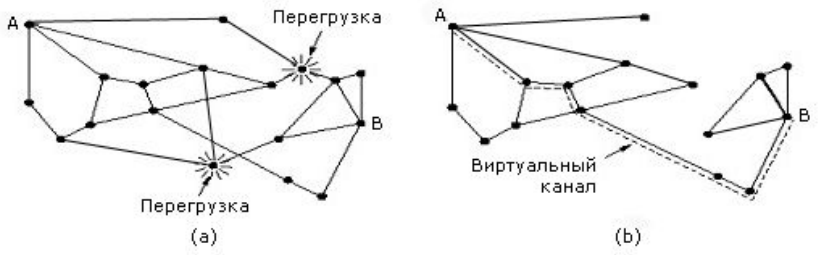
Формирование трафика эффективно тогда, когда отправитель, получатель и среда передачи заранее договорились о форме трафика. Это соглашение называется спецификацией потока. Она представляет собой структуру данных, которая определяет как форму выходного трафика, так и качество сервиса, необходимого приложению.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

Управление перегрузками в сетях с виртуальными каналами.

Прием, который широко используется, чтобы сдержать уже возникшую перегрузку и не дать положению ухудшиться - это контроль на входе. Идея очень проста - если обнаружена перегрузка, то все, что способствует увеличению трафика, запрещено. Прежде всего, запрещается создание новых виртуальных соединений. Таким образом, запрещается создание новых соединений на транспортном уровне. Этот метод хотя и грубоват, но прост в реализации и хорошо апробирован в телефонных сетях.

Другой подход разрешает установку новых виртуальных соединений, но только при наличии не перегруженных маршрутов, т.е. таких маршрутов, которые не пересекаются с перегруженными участками, даже если такие маршруты далеко не оптимальны. Рисунок 5-27 иллюстрирует этот подход.

Рисунок 5-27. (a) Участок сети с перегрузкой; (b) Измененный участок сети без перегрузки с виртуальным каналом AB



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

Подавляющие пакеты.

Теперь рассмотрим приемы, используемые как в средах с виртуальными каналами, так и в средах с дейтаграммами. Каждый маршрутизатор может контролировать степень загрузки своих выходных линий и другие ресурсы. Например, он может периодически вычислять степень загруженности своих выходных линий. Всякий раз, когда степень загруженности при очередном вычислении оказывается выше некоторого порога, эта линия переводится в состояние предупреждения. Каждый пакет, маршрутизируемый через такую линию, вызывает генерацию подавляющего пакета, направляемого отправителю маршрутизируемого пакета. При этом в пакете отправителя проставляется определенный разряд, предотвращающий генерацию подавляющих пакетов другими маршрутизаторами в дальнейшем.

Когда отправитель получает подавляющий пакет, он сокращает интенсивность своего трафика на определенную величину. Поскольку пакеты, направляемые одному и тому же получателю, могут маршрутизироваться по-разному, то отправитель вправе ожидать несколько подавляющих пакетов. В течение определенного времени отправитель будет игнорировать подавляющие пакеты, поступающие с направления получателя. По истечении этого периода времени отправитель ожидает появления подавляющих пакетов в течение следующего интервала. Если появился хоть один подавляющий пакет, то линия перегружена и отправитель ждет. Если в течение очередного интервала не поступило ни одного подавляющего пакета, то отправитель может увеличить интенсивность трафика.

Существует много вариантов этого алгоритма. Так, например, можно использовать не только загруженность линии, но и длину очереди, заполненность буфера и параметры спецификации трафика.

У методов на основе подавляющих пакетов есть один недостаток. Если есть несколько отправителей, работающих через одну и ту же выходную линию, то при определенных условиях маршрутизатор всем им пошлет подавляющие пакеты. Однако, так как отправители независимы, то один, например, сократит трафик значительно, тогда как другие лишь незначительно. Это приведет к несправедливому использованию пропускной способности канала между ними.

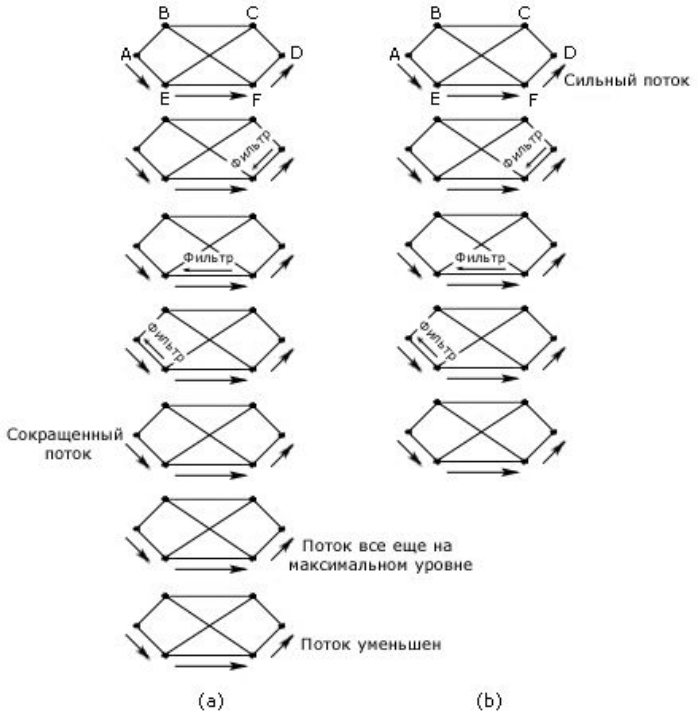
Для предотвращения такой ситуации был предложен алгоритм справедливого чередования. Суть его состоит в том, что для каждого отправителя у выходной линии строится своя очередь. Отправка пакетов из этих очередей происходит по кругу. Поэтому, если кто-то из отправителей незначительно сократит трафик, то это лишь увеличит скорость роста его очереди.

И у этого алгоритма есть недостаток: если один отправитель использует длинные пакеты, а другой - короткие, то последний получит меньшую долю пропускной способности линии. Для борьбы с этой несправедливостью в алгоритм обслуживания очередей вносят модификацию: пакеты из очередей передаются побайтно, а не весь пакет сразу.

Другие модификации алгоритма чередования связаны с установкой приоритетов между очередями, что дает большую гибкость в обслуживании отправителей. Так, если есть очереди для сервера и для клиента, то естественно обслуживать очередь сервера быстрее.

Представленные здесь алгоритмы с подавляющими пакетами плохо работают в высокоскоростных сетях и на больших расстояниях. Дело в том, что пока подавляющий пакет дойдет до отправителя, пройдет много времени и отправитель успеет «напихать» в сеть много пакетов. Для исправления этой ситуации была предложена его модификация – алгоритм с подавлением по скачкам. Суть его в том, что как только обнаружится перегрузка на выходной линии и маршрутизатор отправит подавляющий пакет, то ближайший маршрутизатор, получивший этот подавляющий пакет, сократит трафик к маршрутизатору, пославшему подавляющий пакет. И так скачок за скачком трафик будет быстро падать. Естественно, этот прием увеличит нагрузку на буфера маршрутизаторов, сокращающих трафик, но обеспечит быструю реакцию на возникающую перегрузку. На рисунке 5-29 дан пример алгоритма с подавлением скачками. На этом рисунке хорошо видно, что, применяя технику подавляющих пакетов «в лоб», мы достигнем сокращения нагрузки за 7 скачков (a). Техника подавляющих пакетов по скачкам позволяет добиться того же самого эффекта за пять скачков (b).

Рисунок 5-29. Алгоритм с подавлением скачками



Сброс нагрузки.

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

Маршрутизатор может сбрасывать пакеты, исходя из информации о приложении, пославшем эти пакеты. Если, например, передается файл, то старые пакеты, т.е. расположенные ближе к концу файла, сбрасывать лучше, поскольку приложение может потребовать перепослать все пакеты, начиная с пропущенного пакета. В этом случае, чем старее пакет, тем меньше придется перепосылать. Есть приложения, для которых все наоборот. Лучше сбрасывать новые пакеты, т.е. ближе к началу передачи, чем старые.

В общем случае подход на основе сброса нагрузки предполагает определенное взаимодействие между приложением и маршрутизатором. Например, при передаче изображений в целях компрессии сначала посылают всю картинку, а потом лишь ее изменения. Ясно, что потеря одного из изменений лишь ухудшит изображение на некоторое время, в то время, как потеря картинки будет означать потерю изображения вовсе. Для обеспечения такого взаимодействия с приложением вводят приоритеты среди пакетов. Это позволяет маршрутизатору минимизировать потери для приложения, когда маршрутизатор вынужден сбрасывать пакеты.

Для того, чтобы приложение не злоупотребляло приоритетными пакетами, можно увязать приоритет пакетов с величиной оплаты за трафик. Чем больше приоритетных пакетов, тем выше стоимость передачи. Приоритеты можно использовать также в целях формирования трафика. Например, при использовании алгоритма ведра с маркерами, если пакет пришел, а маркеров нет, можно применить прием, когда пакет все же будет передан, но низким приоритетом.

Приоритеты используются, например, в протоколах Frame Relay. Пока величина трафика находится в заранее оговоренных пределах, все пакеты идут с определенным приоритетом. При пиковых нагрузках, превосходящих номинальную величину, приоритеты пакетов начинают падать в зависимости от времени. Если увеличение размера трафика продолжается недопустимо долго, то вскоре приоритеты пакетов достигнут минимума, и при первых же признаках перегрузки их начнут сбрасывать.

Управление перегрузками при групповой передаче.

Все алгоритмы управления перегрузками, которые мы до сих пор рассматривали, относились к случаю, когда был один источник и один получатель. В этом разделе мы рассмотрим случай управления нагрузкой при групповой передаче, т.е. когда есть несколько источников и несколько получателей. Примером могут служить системы кабельного телевидения. Например, в системе может быть несколько источников телепередач (станций вещания), и зрители могут по желанию переключаться с одной станции на другую. Аналогичная ситуация может иметь место в системе видеоконференций, когда слушатели могут переключаться с одной конференции на другую.

Во многих подобных приложениях группы могут возникать и изменяться по составу динамически. В этих условиях прием, когда источник сообщений резервирует заранее необходимые для передачи ресурсы, не работает эффективно. Источнику сообщений придется при каждом изменении группы генерировать дерево связей. В случае кабельного телевидения, когда группы содержат миллионы зрителей, такой подход не годится.

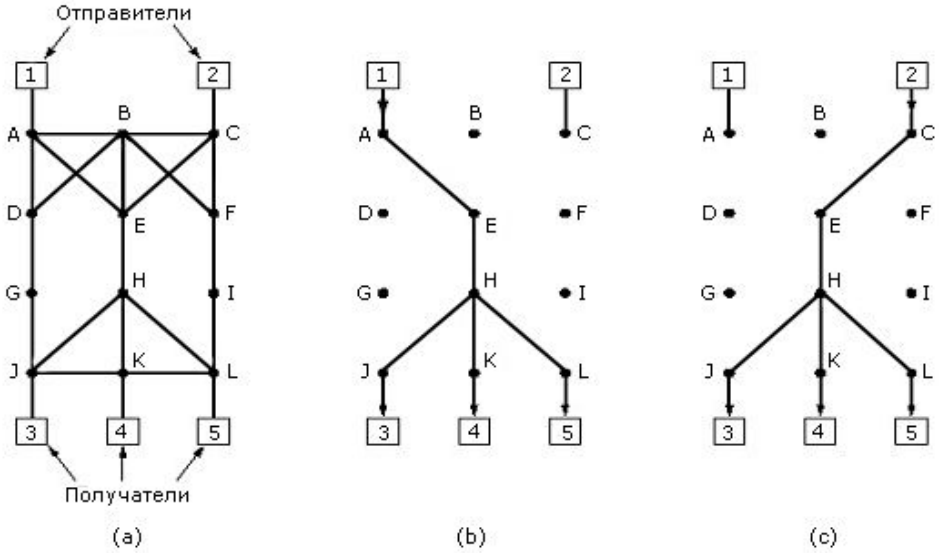
Одно из возможных решений для подобных случаев было предложено в 1993 году – это RSVP-протокол (Resource reSerVation Protocol – протокол резервирования ресурсов). Он позволяет нескольким отправителям передавать сообщения группам получателей, отдельным получателям

переходить из группы в группу, оптимизировать использование пропускной способности каналов, избегая перегрузок.

В простейшей форме этот протокол для групповой маршрутизации использует дерево связей так, как мы уже рассматривали ранее. Каждой группе приписана группа адресов. При отправке пакета отправитель помещает в него весь список адресов группы. После этого стандартный алгоритм групповой маршрутизации строит дерево связей, покрывающее все адреса группы. Собственно маршрутизация не является частью RSVP.

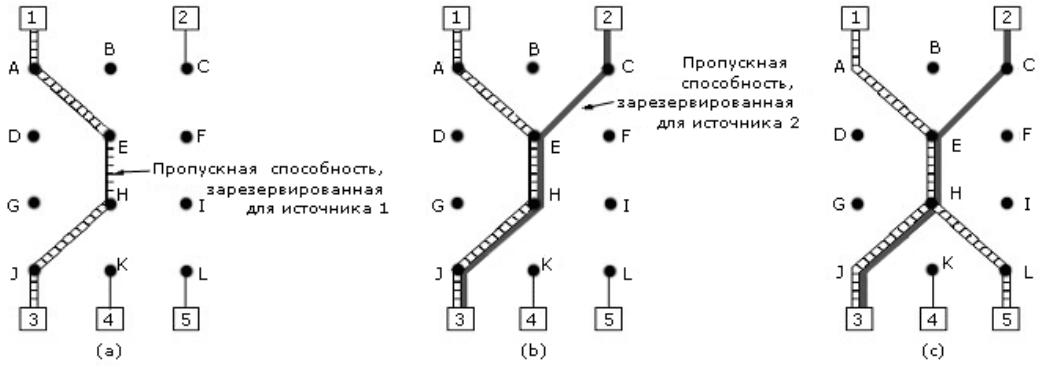
Чтобы понять действие алгоритма RSVP, рассмотрим пример. На рисунке 5-30 (a) показана сеть. Пусть 1 и 2 – групповые отправители, а 3, 4 и 5 – групповые получатели. То, что множество отправителей и получателей не пересекаются, сделано для простоты. Дерево связей для групповой рассылке от 1 дано на рисунке 5-30 (b), а от 2 – на рисунке 5-30 (c).

Рисунок 5-30. Групповая передача



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

Рисунок 5-31. Управление перегрузками при групповой передаче



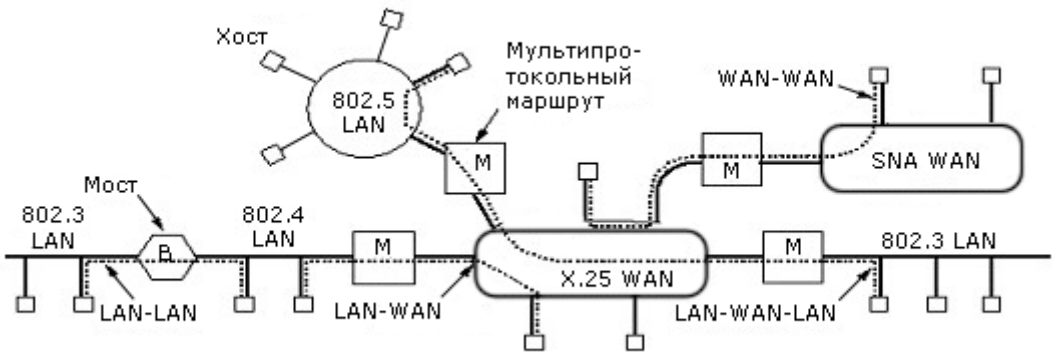
На рисунке 5-31 (a) показан путь резервирования между получателем 3 и отправителем 1. Как только канал между ними установлен, получатель 3 может получать поток пакетов от 1. Рассмотрим теперь, что произойдет, если получатель 3 захочет также получать данные от отправителя 2. Для этого резервирующее сообщение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

Билет № 38.

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

До сих пор мы предполагали, что соединения возникают в рамках однородной среды передачи. Теперь мы перейдем к рассмотрению случаев, когда соединение возникает между СПД-средами с разной архитектурой.

Рисунок 5-32. Межсетевое взаимодействие



На рисунке 5-32 показаны различные сценарии соединений между сетями.

1. LAN-LAN: научный сотрудник передает файл в рамках локальной сети кампуса.

2. LAN-WAN: научный сотрудник посылает письмо коллеге.

3. WAN-WAN: два гуманитария обмениваются мнениями.

4. LAN-WAN-LAN: научные сотрудники разных университетов общаются между собой.

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

1. Уровень 1: репитер копирует биты из одного кабельного сегмента в другой.

2. Уровень 2: мост передает пакеты канального уровня из одной ЛВС в другую.

3. Уровень 3: мультипротокольный маршрутизатор передает пакеты между сетями с разной архитектурой.

4. Уровень 4: транспортный шлюз соединяет байтовые потоки на транспортном уровне.

5. Над уровнем 4: прикладной шлюз соединяет приложения в разных сетях.

Напомним, что термин «шлюз» мы используем для обозначения устройства, соединяющего сети с разной архитектурой.

Репитер - устройство, обеспечивающее усиление и очистку сигнала. На MAC-уровне трансивер обеспечивает передачу сигнала в пределах 500 метров. Репитер обеспечивает передачу на 2,5 км.

Мост способен хранить и маршрутизировать пакеты на канальном уровне. Он получает канальный пакет целиком и решает, по какой линии его передать дальше.

Мультипротокольные маршрутизаторы функционально примерно то же самое, что и мосты, но работают на сетевом уровне. Они получают пакеты сетевого уровня и определяют, куда их передать. Однако при этом разные каналы могут принадлежать разным сетям, использующим разные протокольные стеки. Поэтому мультипротокольному маршрутизатору, кроме задачи маршрутизации, приходится решать и задачу сопряжения форматов пакетов на сетевом уровне в сетях с разной архитектурой.

Таблица 5-34. Различия сетей на сетевом уровне

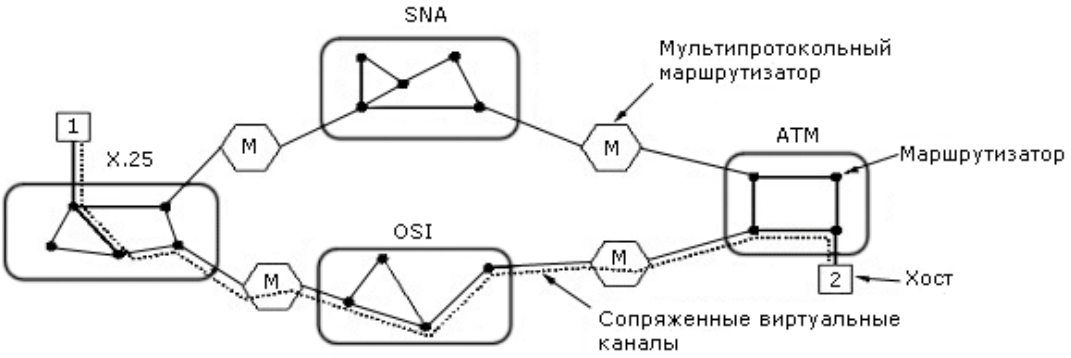
Признак	Возможные значения
Предоставляемый сервис	Ориентированные vs. неориентированные на соединение
Протоколы	IP, IPX, CLNP, AppleTalk, DECnet и т.д.
Адресация	Прямая (802) vs. иерархическая
Групповое вещание	Есть/нет (также транслирование)
Размер пакетов	У каждой сети свой максимальный размер
Качество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

Сопряжение виртуальных каналов.

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

мультипротокольного маршрутизатора, ближайшего к сети получателя. Затем от этого маршрутизатора до получателя - по правилам сети получателя. (Рассмотреть прохождение пакетов вдоль соединения.)

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



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

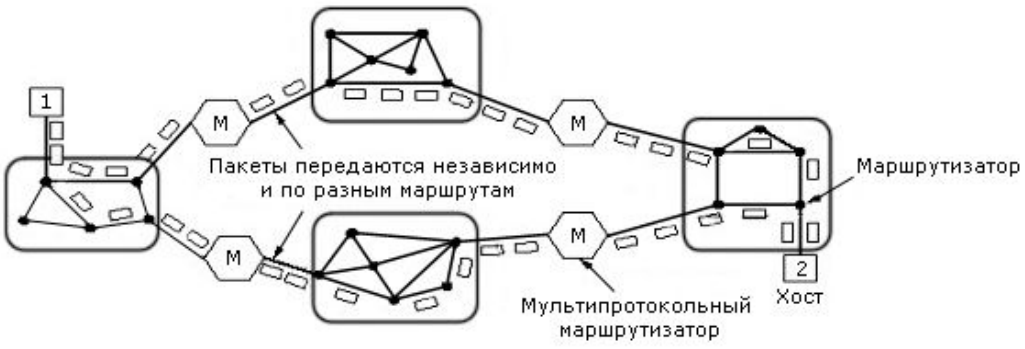
Достоинства сопряжения виртуальных каналов: буферы можно резервировать заранее, порядок пакетов сохраняется, проще управлять повторной передачей из-за задержки, короткие заголовки пакетов.

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

Межсетевое взаимодействие без соединений.

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

Рисунок 5-36. Межсетевое взаимодействие без соединений



Проблемы возникают с адресацией. Различия в адресации могут быть столь велики, что сопряжение станет невозможным. Например, в TCP/IP используется 32-разрядный адрес, а в OSI -

десятичный номер, подобный телефонному. Выход - распространять каждую адресацию на все машины в мире. Однако, очевидно, что такой подход не работает.

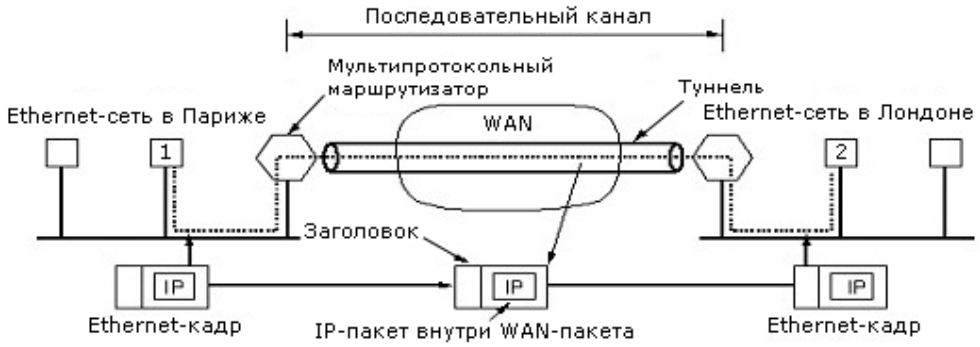
Другой выход - создать универсальный пакет, который понимали бы разные сети, - тоже не работает. Всех уговорить признать один формат как универсальный невозможно.

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

Туннелирование.

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

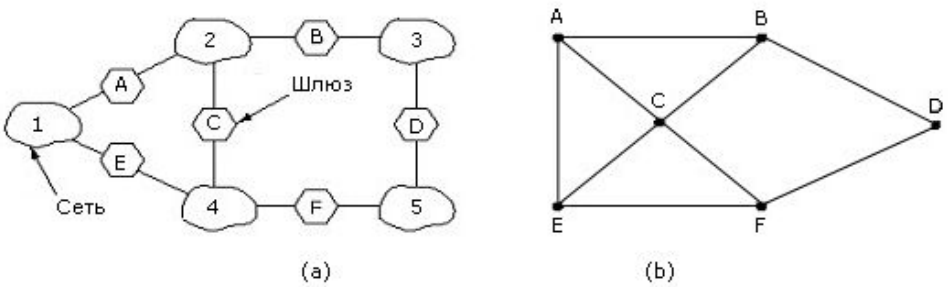
Рисунок 5-37. Транспортировка пакета методом туннелирования



Межсетевая маршрутизация.

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

Рисунок 5-39. Пример межсетевой маршрутизации



Имея граф соединений этих маршрутизаторов между собой, можно применять уже известные нам алгоритмы маршрутизации: по вектору расстояния или по состоянию канала. Так мы приходим к двум уровням маршрутизации: внутреннему межшлюзовому протоколу и внешнему межшлюзовому

протоколу. Поскольку каждая сеть в определенном смысле автономна, то для нее часто используют термин - автономная система.

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

Фрагментация.

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

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

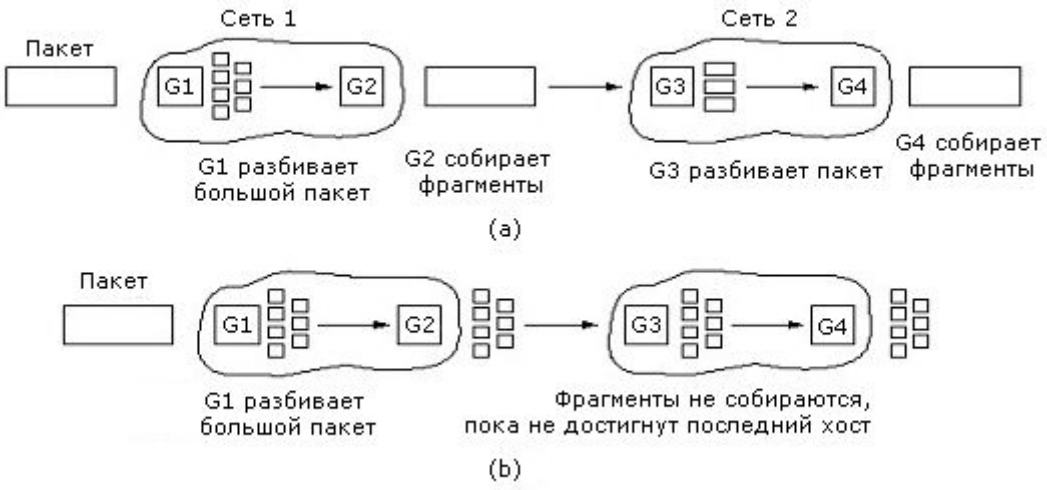
Максимальный размер пакета колеблется от 48 байтов в ATM-сети до 65515 байтов в IP-сети (у протоколов более высоких уровней он еще больше).

Очевидно, первая же проблема возникает при попытке передать большой пакет через сеть, у которой максимальный размер пакета меньше. Одно из решений - проложить маршрут для таких пакетов так, чтобы избежать подобной ситуации. Однако что делать, в такой сети расположен получатель?

Единственное решение - разрешить шлюзу разбивать пакет на фрагменты и отправлять каждый фрагмент независимо. В этом случае возникает проблема сборки фрагментов.

Есть два подхода к тому, как это осуществлять. Первый - делать фрагменты столь малыми, что любая сеть на их пути будет прозрачна для них. Это решение показано на рисунке 5-40 (а). Когда поступает большой пакет, его разбивают на малые и все пакеты отправляют на один и тот же выходной шлюз, где они собираются снова в большой пакет.

Рисунок 5-40. Фрагментация пакетов

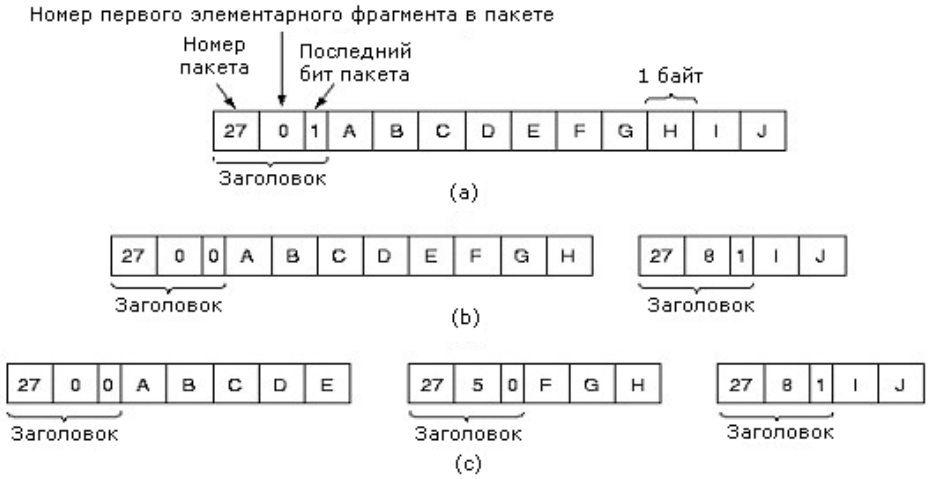


В такой фрагментации есть трудности: как узнать, что все фрагменты достигли выходного шлюза, как выбирать маршрут для фрагментов, накладные расходы на разбиение и сборку пакета из фрагментов.

Другой подход - разбив пакет на фрагменты, рассматривать каждый из них как обычный пакет. Это решение показано на рисунке 5-40 (b). Сборка фрагментов происходит только в узле назначения. Однако при таком подходе каждый хост должен уметь собирать пакеты из фрагментов.

На рисунке 5-41 показана фрагментация в случае, если размер элементарного фрагмента данных равен 1 байту.

Рисунок 5-41. Фрагментация: (a) Исходный пакет, содержащий 10 байтов данных; (b) Фрагменты после прохождения сети с максимальным размером пакета, равным 8 байтам; (c) Фрагменты после прохождения через шлюз с размером 5



Firewall.

Способность любого компьютера соединяться, где бы он ни был, с любым другим компьютером - благо для пользователя, но сущее наказание для службы безопасности любой организации. Здесь, кроме угрозы потери информации, есть угроза притока всякой гадости типа вирусов, червей и прочих цифровых паразитов.

Итак, нужен механизм, который бы различал «чистые» биты от «нечистых». Один способ - шифровать данные. Так поступают при передаче данных. Со способами шифрования мы познакомимся позднее. Но шифрование бессильно против вирусов, хакеров и прочей нечисти. Одним из средств борьбы с ними служат брандмауеры (firewall).

Барьер - современная форма крепостного рва. Компания может иметь сколь угодно сложную сеть, объединяющую много локальных сетей. Однако весь трафик в сеть и из этой сети направляется только через один шлюз, где происходит проверка пакета на соответствие определенным требованиям. Если пакет не удовлетворяет этим требованиям, то он не допускается в сеть или из нее.

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

Билет № 39.

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

Интернет представляет собой объединение подсетей, которые называются автономными системами. Автономные системы – это подсеть, охватывающая единую территорию, находящаяся под единым административным управлением и имеющая единую политику маршрутизации по отношению ко всем остальным сетям. В Интернете нет какой-либо регулярной, специально предусмотренной структуры подсетей. Он образован из соединения большого числа подсетей, среди которых можно выделить несколько остовых (backbone). К этим остовым сетям подключены региональные сети, к которым подключены локальные сети организаций. На рисунке 5-43 показана схема соединения таких остовых сетей.

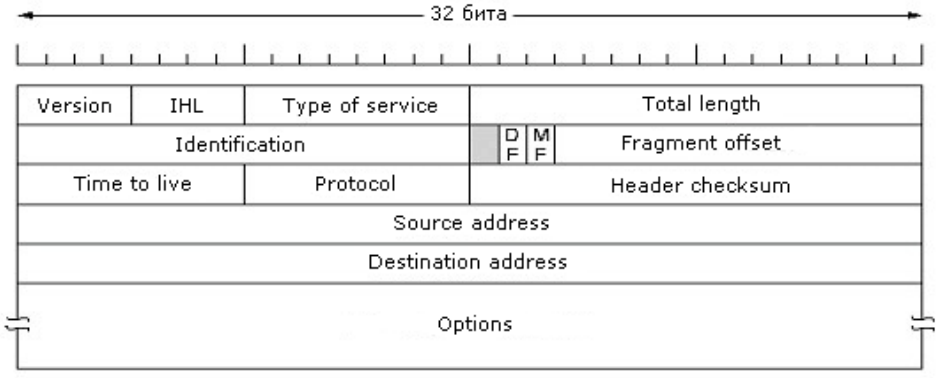
Соединяет все автономные системы вместе IP-протокол. В отличие от других протоколов сетевого уровня, этот протокол с самого начала создавался для объединения сетей. Его целью было наилучшим образом передавать дейтаграммы от одной машины к другой, где бы эти машины ни находились.

Как мы уже отмечали, подсеть в Интернете реализует сервис без соединений и работает следующим образом. Транспортный уровень получает поток данных и делит их на дейтаграммы. Дейтаграммы могут быть от 64К до 1500 байт. Они передаются через подсети в Интернет и, если необходимо, делятся на более короткие. Когда все дейтаграммы достигают места назначения, они собираются в исходные дейтаграммы на сетевом уровне и передаются на транспортный уровень, где и восстанавливается исходный поток данных.

IP-протокол.

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

Рисунок 5-44. Заголовок IP



- Version - указывает версию протокола.
- IHL - длина заголовка в 32-разрядных словах (минимум 5, максимум 15, что соответствует 60 байтам).
- Type of service - вид необходимого сервиса. Здесь возможны различные комбинации скорости и надежности: например, передача голоса, аккуратная доставка строки битов, файла и т.п.
- Identification - позволяет отличать фрагменты одной и той же дейтаграммы.
- DF – признак управления фрагментацией. Если он равен 1, то фрагментация невозможна.
- Total length - указывает общую длину дейтаграммы, включая заголовок и поле данных. Максимальная длина 65535 байт.
- Identification – указывает, какой дейтаграмме принадлежит очередной поступивший фрагмент. Все фрагменты одной дейтаграммы имеют в этом поле одно и то же значение.
- MF – содержит единицу только у последнего фрагмента дейтаграммы. Это поле позволяет отличить последний фрагмент от всех остальных.
- Fragment offset – указывает, где в дейтаграмме располагается данный фрагмент. Длина всех фрагментов, кроме последнего, должна быть кратна 8 байтам. Поскольку поле имеет 13 разрядов, то на одну дейтаграмму максимально может быть 8192 фрагментов.
- Time to live – время жизни пакета. Максимальное значение этого поля - 255, измеряется в секундах. Очень часто здесь используется счетчик скачков.
- Protocol - показывает, какому процессу на транспортном уровне передать собранную дейтаграмму (TCP, UDP и т.д.).
- Header checksum - контрольная сумма, которая охватывает только заголовок.
- Source address, Destination address – идентифицируют машину отправителя и получателя в сети.
- Options - предусмотрено для расширения возможностей протокола. Оно делится, в свою очередь, на следующие поля:
    - Security – указывает уровень секретности передаваемой информации. Маршрутизатор может на основании значения этого поля запретить определенные маршруты, например, если они пролегают через ненадежные регионы.
    - Strict source routing – здесь указан полный маршрут в виде списка IP-адресов. Это поле используется в алгоритме маршрутизации от источника, а также в критических ситуациях, например, когда таблица маршрутизации по какой-то причине оказалась испорченной.
    - Loose source routing – список маршрутизаторов, через которые фрагмент обязан пройти. Он может пройти и через другие маршрутизаторы, но данные обязательно должны принадлежать его маршруту.
    - Record route – указывает маршрутизаторам на необходимость заносить в поле свои адреса. Это позволяет проследить, как шел фрагмент.
    - Time stamp – вместе с предыдущим полем указывает маршрутизаторам на необходимость записывать не только свои адреса, но и время, когда фрагмент проходил через них. Это поле очень полезно при отладке алгоритмов маршрутизации.

IP-адресация.

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

Рисунок 5-45. Форматы IP-адресов

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

Рисунок 5-46. Специальные IP-адреса

Подсети.

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

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

Чтобы понять, как используется адрес подсети, надо проследить, как маршрутизатор использует IP-адрес. Есть две таблицы: некоторое количество IP-адресов вида «сеть, 0» и некоторое количество IP-адресов вида «эта_сеть, адрес машины». Первая показывает, как достичь интересующей сети. Вторая - как достичь узла внутри сети. Когда поступает IP-пакет, маршрутизатор ищет его адрес доставки в таблице маршрутизации. Если этот адрес – адрес другой сети, то пакет передают дальше тому маршрутизатору, который отвечает за связь с этой сетью. Если это адрес в локальной сети, то маршрутизатор направляет пакет прямо по месту назначения. Если адреса нет в таблице, то маршрутизатор направляет пакет специально выделенному по умолчанию маршрутизатору, который должен разобраться с этим случаем с помощью более подробной таблицы. Из этого описания видно, что алгоритм маршрутизации имеет дело только с сетями или локальными машинами, а не с парами «сеть, узел». Такая организация алгоритма позволяет существенно сократить размер таблиц в маршрутизаторах.

С появлением подсети структура адресов меняется. Теперь записи в таблице имеют форму «эта_сеть, подсеть, 0» и «эта_сеть, эта_подсеть, машина». Таким образом, маршрутизатор подсети в данной локальной сети знает, как достичь любой подсети в данной локальной сети и как найти конкретную машину в своей подсети. Все что ему нужно – это знать маску подсети. С помощью логической операции «&» маршрутизатор выделяет адрес подсети с помощью маски, показанной на рисунке 5-47. По своим таблицам он определяет, как достичь нужной подсети или (если это локальная подсеть данного маршрутизатора) как достичь конкретной машины.

Протоколы управления межсетевым взаимодействием.

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

Internet Control Message Protocol. Управление функционированием Интернета происходит через маршрутизаторы с помощью протокола ICMP (RFC 792). Этот протокол обеспечивает доставку сообщений любой машине, имеющей IP-адрес (хосту), от маршрутизаторов и других хостов в сети. Этот протокол обеспечивает обратную связь при возникновении проблем при передаче. Он выявляет и рассылает сообщения о десятках событий, наиболее важные из них показаны в таблице 5-48.

Таблица 5-48. Основные типы сообщений ICMP

Тип сообщений	Описание
Destination unreachable (Назначение недостижимо)	Пакет не может быть доставлен
Time exceeded (Время истекло)	Время жизни достигло "0"
Parameter problem (Проблемы с параметрами)	Недопустимое поле заголовка
Source quench (Источник отключен)	Подавляющий пакет
Redirect (Перенаправление)	Объясните маршрутизатору, где он находится
Echo request (Запрос отклика)	Спросите машину, работает ли она
Echo reply (Ответ на запрос)	Да, машина работает.
Timestamp request (Запрос с временной меткой)	То же, что Echo request, только с временной меткой
Timestamp reply (Ответ с временной меткой)	То же, что Echo reply, только с временной меткой

Протокол ICMP использует протокол IP и доставка его дейтаграмм не более надежна, чем любой IP-дейтаграммы в сети. Сообщение destination unreachable покрывает множество случаев: от случая, когда маршрутизатор не знает, как достигнуть нужной подсети или хоста, до случая, когда дейтаграмма при доставке должна быть фрагментирована, но установлен флаг, который запрещает это делать.

Сообщение time exceeded посылает маршрутизатор, если он обнаружил дейтаграмму с истекшим времени жизни. Хост генерирует такое сообщение, если он не успел завершить сборку дейтаграммы до истечения времени ее жизни.

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

Сообщение source quench обеспечивает средство управления потоком. Маршрутизатор или хост-получатель высылает этот пакет хосту-отправителю, если необходимо понизить скорость передачи. Сообщения этого типа будут генерироваться до тех пор, пока скорость поступления дейтаграмм от отправителя не достигнет нужной хосту-получателю величины. Это сообщение система может использовать для предотвращения перегрузки. Оно возникает всякий раз, когда маршрутизатор вынужден сбросить дейтаграмму из-за переполнения своего буфера.

Сообщение redirect позволяет маршрутизатору отправить рекомендацию о лучшем маршруте и впредь посылать дейтаграммы с определенным адресом через другой маршрутизатор.

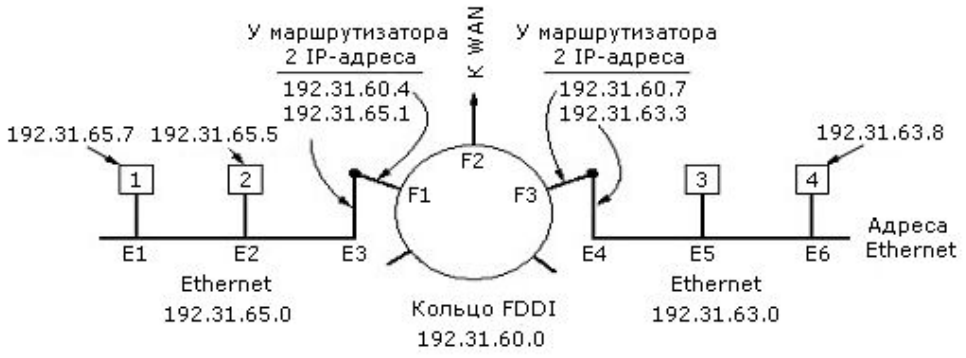
Сообщения echo request и echo reply обеспечивают механизм проверки работоспособности объектов в сети. Получатель сообщения echo request обязан ответить сообщением echo reply, причем с теми же параметрами, что и в echo request.

Сообщения timestamp request и timestamp reply обеспечивают механизм для измерения и изменения параметров временной задержки в Интернете. Этот механизм необходим, например, для работы алгоритма маршрутизации по состоянию канала.

Address Resolution Protocol – протокол определения адреса. Хотя каждая машина в Интернете имеет уникальный IP адрес, и даже не один, но при передаче пакета через сеть от этого мало пользы, так как канальный уровень не понимает IP адресов. Как правило, машина подключена к ЛВС через сетевую карту, которая понимает только ЛВС адреса канального уровня, например, Ethernet-адрес. Этот адрес имеет 48 разрядов. Сетевая карта знает только такие адреса и ничего об 32-разрядных IP.

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

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



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

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

Reverse Address Resolution Protocol (RARP) – обратный протокол определения адреса. Иногда возникает обратная проблема - известен Ethernet-адрес, но какой IP-адрес ему соответствует? Эта проблема возникает, например, при удаленной загрузке бездисковой станции. Как эта станция определит свой и соседние IP-адреса?

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

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

IPv6.

Появление новой версии протокола IP (IPv6, в настоящее время используется IPv4) обусловлено целым рядом причин. Одна из основных - стремительный рост всемирной сети Интернет. Фундаментальным принципом построения сетей на основе протокола IP, необходимым для правильной маршрутизации и доставки пакетов, является уникальность сетевых адресов, т.е. каждый IP-адрес может принадлежать только одному устройству. На сегодняшний день остались невыделенными около 1 400 000 000 адресов из возможных 4 294 967 296, то есть примерно 30%, чего должно хватить на несколько лет, а может быть и более. Дефицит адресов пока выражается в основном в том, что, по выражению одного из сетевых гуру, адрес класса A не смог бы получить и сам Господь Бог. Таких адресов может существовать всего 128 (формат: 0, адрес сети - 7 бит, адрес хоста - 24 бита), но каждый из них содержит 16 777 216 адресов. Однако появившиеся в последнее время новые устройства для доступа в Интернет и развитие цифрового телевидения, которое собирается превратить каждый телевизор в интернет-устройство, могут быстро исчерпать имеющиеся запасы неиспользованных адресов.

Если в компьютерных сетях для выхода в Интернет могут применяться технологии типа NAT (Network Address Translation, — преобразование сетевого адреса), при которой для взаимодействия с окружающей средой используется всего несколько уникальных адресов, предоставляемых, возможно, провайдером, а внутри локальной сети адресация может быть достаточно произвольной, то для сетевого телевизора этот способ не подходит, так как каждому устройству требуется свой уникальный адрес.

Кроме всего прочего, новые возможности предъявляют к протоколам сетевого уровня, каковым является IP, совершенно новые требования в части легкости получения и смены адресов, полностью автоматического конфигурирования (представьте себе домохозяйку, настраивающую DNS своего телевизора). Если новый протокол не появится своевременно, то фирмы-провайдеры начнут внедрять свои собственные, что может привести к невозможности гарантированного соединения «всех со всеми». Открытый протокол, удовлетворяющий требованиям необходимого адресного пространства, легкости конфигурирования и маршрутизации, способный работать совместно с имеющимся IPv4, поможет сохранить способность к соединению между собой любых устройств, поддерживающих IP, при наличии новых возможностей, которые основаны на анализе использования IPv4.

Кроме того, остается еще одна проблема: уникальность адреса вовсе не означает, что устройство будет правильно функционировать. Адреса нужны в первую очередь не для того, чтобы

«всех пересчитать», а для правильной маршрутизации при доставке пакетов. Таким образом, для беспрепятственного роста Интернета необходимо не только наличие свободных адресов, но и определенная методика их выделения, позволяющая решить проблему масштабируемости. Сведение к минимуму накладных расходов на маршрутизацию является сегодня одной из основных проблем, и ее важность будет возрастать в дальнейшем по мере роста Сети. Просто присвоить устройству адрес недостаточно, необходимо еще обеспечить условия для правильной маршрутизации с минимальными накладными расходами.

В настоящее время только одна известная технология, а именно, иерархическая маршрутизация, позволяет за счет приемлемых технических издержек обеспечить доставку пакетов в сети размерами с Интернет. Технология иерархической маршрутизации заключается в разбиении всей сети на более мелкие подсети, маршрутизация в которых производится самостоятельно. Подсети, в свою очередь, могут разбиваться на еще более мелкие, и т.д. В результате образуется древовидная структура, причем в качестве узлов выступают маршрутизаторы, а в качестве листьев - оконечные устройства-хосты. Путь, который проделывает пакет, передаваемый от одного листа до другого, может быть длиннее, чем при иной топологии, но зато он всегда может быть рассчитан с наименьшими издержками. Некоторую аналогию можно провести с телефонными номерами — первым идет код страны, за ним код города, а затем собственно номер, состоящий, в свою очередь, из кода АТС и собственно номера абонента.

История нового протокола восходит к концу 1992 года. Именно тогда IETF (Internet Engineering Task Force — рабочая группа по технической поддержке Интернет) приступила к анализу данных, необходимых для разработки нового протокола IP. К концу 1994 года был утвержден рекомендательный стандарт и разработаны все необходимые для реализации протокола вспомогательные стандарты и документы.

IPv6 является новой версией старого протокола, разработанной таким образом, чтобы обеспечить совместимость и «мягкий» переход, не приуроченный к конкретной дате и не требующий одновременных действий всех участников. По некоторым прогнозам, совместное существование двух протоколов будет продолжаться до десяти и более лет. Учитывая то обстоятельство, что среди выделенных типов адресов IPv6 имеется специальный тип адреса, эмулирующий адрес IPv4, можно ожидать относительно спокойного перехода, не сопровождающегося крупными неудобствами и неприятностями. Фактически на одном компьютере могут работать оба протокола, каждый из которых подключается по мере необходимости.

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

Однако использование старых адресов не является выходом из положения, поэтому протокол IPv6 предусматривает специальные возможности по присвоению новых адресов и их замене без вмешательства (или при минимальном вмешательстве) персонала. Для этого предусмотрена привязка к компьютеру не IP-адреса, а интерфейса. Сам же интерфейс может иметь несколько адресов, принадлежащих к трем категориям: действительный, прошлый, недействительный. При замене адреса «на лету» новый адрес становится действительным, а старый — прошлым. Все вновь осуществляемые соединения производятся при помощи действительного адреса, но уже имеющиеся продолжаются по прошлому адресу. Через некоторое время, которое может быть выбрано достаточно большим, чтобы гарантировать полный разрыв всех соединений по прошлому адресу, он переходит в категорию недействительных. Таким образом, практически гарантируется автоматическая замена адреса без участия персонала. Для полностью гарантированной автоматической замены адреса потребовалось бы внесение изменений в протоколы TCP и UDP, которые не входят в состав IP.

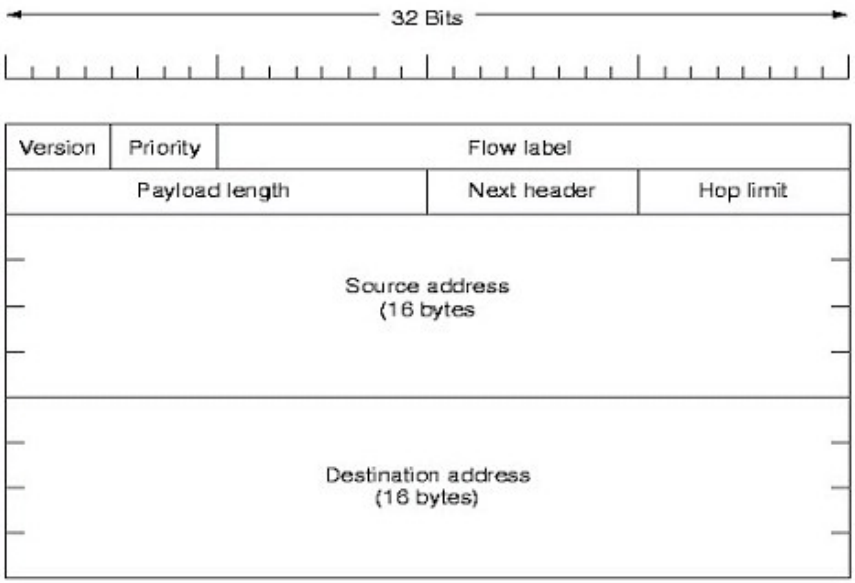
Замена адресов осуществляется двумя способами — явным и неявным. Явный способ использует соответствующим образом доработанный протокол DHCP. Неявный способ не требует

наличия сервера DHCP, а использует адрес подсети, получаемый от соседей и мостов. В качестве адреса хоста используется просто MAC-адрес хоста, т.е. адрес, используемый на канальном уровне. Этот способ, при всем своем изяществе, по понятным причинам не может присваивать адреса, совместимые с IPv4, и поэтому в переходный период его применение будет ограничено. К сожалению, механизм выделения новых адресов не затрагивает таких аспектов, как обновление базы данных DNS, адресов серверов DNS, конфигурации маршрутизаторов и фильтров, а также тех приложений «клиент-сервер», которые используют привязку к адресу, что делает полную замену адресов локальной сети не менее трудоемким мероприятием, чем при применении IPv4.

Протокол IPv6 предполагает также значительные улучшения при работе в локальной сети. Единый протокол NDP (Neighbor Discovery Protocol - протокол распознавания соседей) заменяет используемые в IPv4 протоколы ARP, ICMP и значительно расширяет их функциональные возможности. Вместо широковещательных пакетов канального уровня протокола ARP используются групповые сообщения (multicast), то есть адресованные всем членам подсети, причем не на канальном, а на сетевом уровне, что должно значительно снизить широковещательный трафик, являющийся бичом локальных сетей Ethernet. Усовершенствованы функции протокола ICMP, что облегчает работу разных подсетей в одном физическом сегменте. Включен механизм распознавания неисправных маршрутизаторов, что позволяет повысить устойчивость к сбоям оборудования. В дополнение к имевшимся ранее двум типам адресации - Unicast и Multicast (доставке уникальному получателю или группе получателей) - добавлен третий, Anycast, при котором осуществляется доставка любому получателю из группы.

Существенное отличие нового протокола от старого заключается в том, что длина адресной части составляет 128 бит — в четыре раза больше, чем 32 бита у IPv4. Чтобы представить эту величину, достаточно сказать, что на каждом квадратном метре поверхности суши и моря можно разместить примерно 6,7x1023 адресов. Из заголовка пакета IP изъяты как некоторые неиспользуемые поля, что позволило сократить издержки, связанные с их обработкой, и уменьшить размер заголовка (он длиннее, чем у IPv4, всего в два раза, несмотря на учетверенный размер адресной части).

Рисунок 5-59. Заголовок пакета IPv6



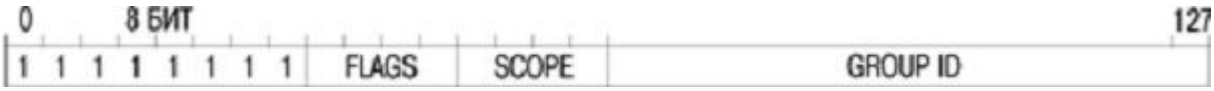
Первым идет четырехбитное поле Version (Версия), его значение равно 6. Следующее поле - Priority (Приоритет) - длиной 8 бит используется для установки приоритета пакета. Приоритет увеличивается с ростом значения этого поля. Значения 0...7 используются для пакетов, время доставки которых не лимитировано, например, значение 1 рекомендуется использовать для новостей,

2 — для почты, 7 — для служебного трафика (SNMP, маршрутизирующие протоколы). Значения 8...15 используются для пакетов, задержка доставки которых нежелательна, например аудио и видео в реальном времени. Далее следует поле Traffic Class, первоначально называвшееся «Flow Label», длиной 20 бит. Оно служит для идентификации последовательности пакетов. Его значение присваивается при помощи генератора случайных чисел и имеет одинаковую величину у всех пакетов данной последовательности. Следующее поле - Payload Length - содержит размер данных, следующих за заголовком, в байтах и имеет длину 16 бит. Следом расположено поле Next Header, идентичное по назначению полю Protocol протокола IPv4 и использующее те же значения. Восьмибитное поле Hop Limit аналогично по назначению полю Time to Live. Оно устанавливается источником согласно разумным предположениям о длине маршрута, а затем уменьшается на 1 при каждом прохождении через маршрутизатор. При снижении значения поля до нуля пакет снимается, как «заблудившийся». Последними идут поля адресов источника и приемника длиной 128 бит (16 байт) каждое. Адреса в стандарте IPv6 имеют более сложную структуру, чем в предыдущем, при этом используются префиксы разной длины.

В настоящее время для непосредственного использования предназначено около 15% адресов, остальные 85% зарезервированы для распределения в будущем.

Специальные типы адресов предназначаются для более гибкого использования. Provider-Based Unicast Address (Выделяемый провайдером уникальный адрес) служит для глобальной связи. Он состоит из префикса 010, Registry Id, идентифицирующего организацию, зарегистрировавшую провайдера; Provider Id, идентифицирующего провайдера; Subscriber Id, идентифицирующего организацию-клиента, и собственно адреса. Адреса для локального использования (Link Local Use и Site Local Use) предназначены для применения внутри одного сегмента или одной организации, т.е. пакеты с такими адресами не маршрутизируются за границы текущего сегмента или локальной сети соответственно. Они могут быть использованы, например, при автоматическом присвоении адресов. Для выхода в глобальную сеть может быть использована подстановка адресов по типу NAT. Если под заполнители-нули выделено достаточно места, то организация, ранее не имевшая соединения с Интернетом, может легко провести замену адресов на глобальные путем конкатенации REGISTRY.

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



ID + PROVIDER ID + SUBSCRIBER ID и локального адреса. К специальным типам адресов также относятся адреса, совместимые с IPv4. Первый тип относится к «совместимым» адресам, которые предназначены для туннелирования пакетов IPv6 через существующую инфраструктуру IPv4. Второй тип адресов отображает на IPv6 подмножество адресов IPv4 для тех устройств, которые не поддерживают новый протокол.

Широковещательный адрес благодаря использованию полей Flags и Scope может также использоваться более гибко. В четырехбитном поле Flags пока используется только младший бит для указания, является ли данный адрес постоянным и выделенным соответствующими организациями, ответственными за выдачу адресов, или используется единовременно. Поле Scope используется для ограничения области распространения широковещательных пакетов. Значения этого поля приведены в таблице 2.

Таблица 2.

0	Зарезервировано
1	Внутри узла
2	Внутри сегмента

5	Внутри локальной сети
8	Внутри организации
0Eh	Глобальная
0Fh	Зарезервировано

При рассмотрении возможностей, предоставляемых новым протоколом, может возникнуть вопрос, а зачем он все-таки нужен? Большинство функций либо уже имеются в IPv4, либо могут быть реализованы путем доработки соответствующих протоколов. Так, автоматическое выделение адресов производится при помощи протокола DHCP, адресный барьер преодолевается при помощи протокола NAT и т.д. и т.п. Однако разработка всех необходимых заплаток для протокола IPv4 потребовала бы не меньших (а то и больших) усилий, чем создание нового протокола «с чистого листа». Разумеется, лист был не совсем чистым, поскольку вопросы совместимости и совместной работы обоих протоколов имелись в виду с самого начала проектирования. В конце концов, Интернет все равно пришел бы к кризису дефицита адресов, так что заблаговременная разработка и постепенное внедрение протокола IPv6 были более чем уместны.

Для реализации перехода на новый протокол образовалась неформальная некоммерческая организация «6bone», включающая в себя более 100 организаций, в основном, сетевых провайдеров и университетов. Главная задача организации - создание инфраструктуры, позволяющей транспортировать пакеты стандарта IPv6 по всей сети Интернет. Как и существующая сегодня инфраструктура IPv4, она будет состоять из большого количества провайдеров и локальных сетей, объединенных в единую Сеть. В настоящее время в состав 6bone входят представители 41 страны, от США, Англии и Японии до Камеруна и Казахстана.

Необходимость создания такой инфраструктуры объясняется прежде всего тем, что без широкомасштабного тестирования и готовой инфраструктуры (или ее подобия) коммерческие провайдеры (и потребители, занимающиеся, в отличие от университетов, не исследованиями, а бизнесом) вряд ли будут охотно внедрять новый протокол. Таким образом, задачей сети 6bone не является организация параллельной инфраструктуры, а, скорее, тестирование и отработка методик взаимодействия «клиент-провайдер».

Сама сеть 6bone состоит из островков-сетей, которые полностью поддерживают IPv6, соединенных виртуальными туннелями. Эти сети работают на установленном у провайдеров оборудовании, которое используется и в коммерческих целях. Согласно существующему в настоящее время мнению, организация 6bone будет существовать до тех пор, пока будет актуальна ее основная цель - популяризация протокола IPv6.

Билет № 40.

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

OSPF - внутренний протокол маршрутизации шлюзов.

Интернет состоит из сетей, управляемых разными организациями. Каждая такая сеть использует внутри свои алгоритмы маршрутизации и управления и называется автономной системой. Наличие стандартов позволяет преодолеть различия во внутренней организации автономных систем и обеспечить их совместное функционирование. Алгоритмы маршрутизации, применяемые внутри АС, называются внутренними протоколами шлюзов. Алгоритмы маршрутизации, применяемые для маршрутизации между АС, называются внешними протоколами шлюзов.

Изначально в качестве внутреннего протокола шлюзов использовался протокол по вектору расстояния (RIP). Этот протокол работал хорошо, пока автономная система была небольшой. Однако по мере роста АС он начинал работать все хуже и хуже. Проблемы «счетчика до бесконечности» и

медленная сходимость не получили удовлетворительного решения. В 1979 году он был замещен протоколом маршрутизации по состоянию каналов. В 1988 году инженерный комитет Internet принял решение о разработке нового алгоритма маршрутизации. Этот алгоритм, названный OSPF - Open Shortest Path First, стал стандартом в 1990 году (RFC 1247).

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

В-четвертых, он должен поддерживать разные виды сервиса, поддерживать маршрутизацию для трафика в реальном времени одним способом, а для других типов трафика - другим. В IP-пакете есть поле Type of service, которое не использовалось существующими в то время протоколами.

В-пятых, он должен обеспечивать балансировку нагрузки и при необходимости разделять потоки по разным каналам. Все предыдущие протоколы использовали только один канал - наилучший.

В-шестых, он должен поддерживать иерархию. К 1988 году Интернет стал столь большим, что ни один маршрутизатор был уже не в состоянии хранить всю топологию. Поэтому новый протокол должен быть сконструирован так, чтобы по нему мог бы работать не один маршрутизатор.

В-седьмых, должна быть усилена безопасность маршрутизаторов для защиты от студентов, которые развлекались тем, что подсовывали маршрутизаторам неверную информацию о маршрутах. Наконец, надо было позаботиться о том, чтобы позволить маршрутизаторам общаться с помощью туннелирования.

OSPF поддерживает три вида соединений и сетей:

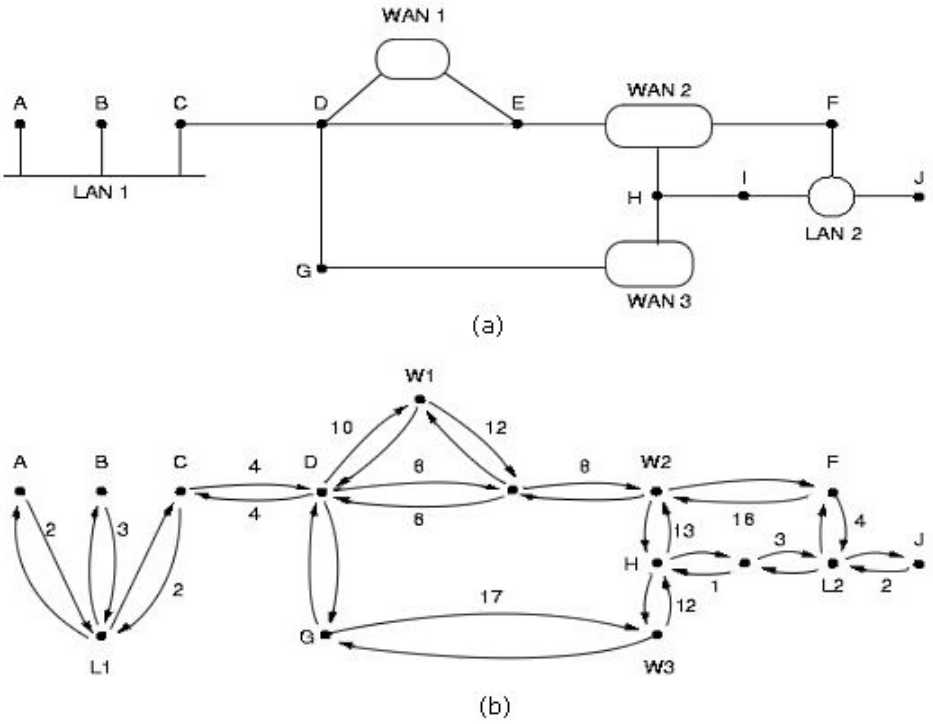
1. Точка-точка между двумя маршрутизаторами

2. Сети с множественным доступом и вещанием (большинство ЛВС)

3. Сети с множественным доступом без вещания (например, региональные сети с коммутацией пакетов)

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

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



Многие АС сами по себе представляют большие сети. OSPF позволяет разбивать их на области, где каждая область - это либо сеть, либо последовательность сетей. Области не пересекаются. Есть маршрутизаторы, которые не принадлежат никакой области. Область - обобщение понятия подсети. Вне области ее топология не видна.

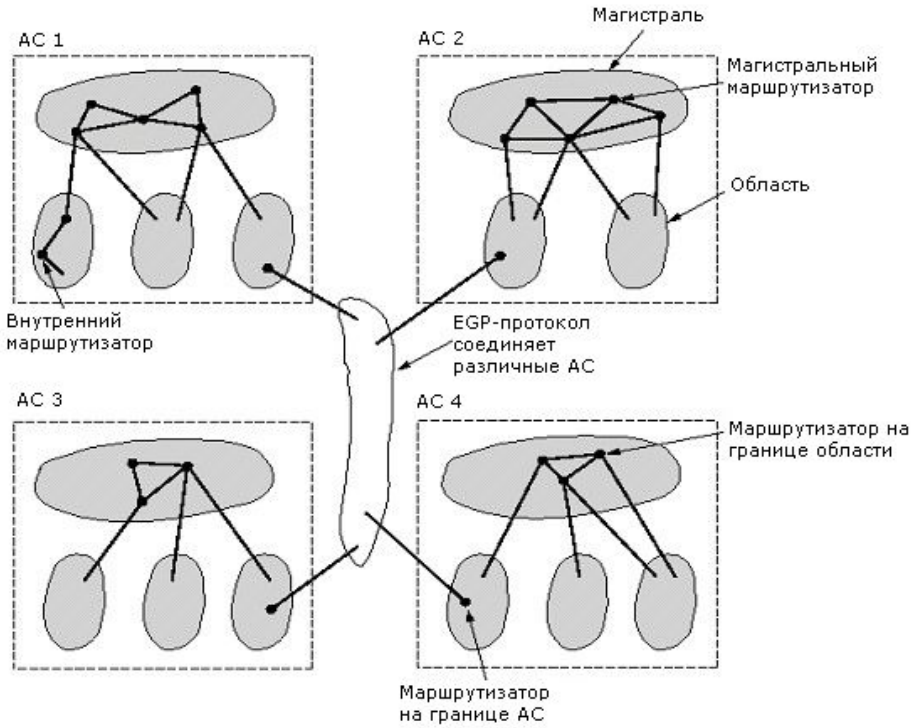
Каждая АС имеет остовую область, называемую «областью 0». Все области АС соединяются с остовой, возможно через туннелирование. Поэтому можно из одной области попасть в другую через остовую область. Туннель представлен в графе дугой с весом. Любой маршрутизатор, соединенный с двумя или более областями, - часть остовой области. Как и в других областях, топология остовой области не видна извне.

Внутри области у каждого маршрутизатора одинаковая база данных состояний каналов и одинаковый алгоритм наикратчайшего пути. Задача маршрутизатора - вычислить наикратчайший путь до другого маршрутизатора этой области, включая маршрутизатор, соединенный с остовой областью. Маршрутизатор, соединенный с двумя областями, должен иметь две базы данных и выполнять два алгоритма наикратчайшего пути независимо.

Чтобы поддерживать разные типы сервисов, OSPF использует несколько графов, один с разметкой относительно задержки, другой - относительно пропускной способности, третий - относительно надежности. Хотя все три требуют соответствующих вычислений, но зато мы получаем три маршрута, оптимизированных по задержке, пропускной способности и надежности.

Во время функционирования возникают три вида маршрутов: внутри области, между областями и между АС. Внутри области вычислить маршрут просто - им будет наикратчайший до маршрутизатора получателя. Маршрутизация между областями всегда выполняется в три этапа: от источника до остовой области, от остовой до области назначения, внутри области назначения. Этот алгоритм навязывает звездообразную топологию OSPF: остовая область – центр, ось, остальные области – лучи, спицы. Пакеты маршрутизируются без изменений, как есть, за исключением случая, когда область получателя соединена с остовой областью туннелем. Рисунок 5-51 показывает эти три вида маршрутов.

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



OSPF различает четыре класса маршрутизаторов:

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

Эти классы могут пересекаться. Например, все пограничные маршрутизаторы – остовые; маршрутизатор из остовой области, но не на ее границе - внутренний. Примеры этих классов маршрутизаторов показаны на рисунке 5–51.

Когда маршрутизатор загружается, он рассылает сообщение «Hello» всем своим соседям - на линиях «точка-точка», группам маршрутизаторов в ЛВС с множественным доступом, чтобы получить информацию о своем окружении.

В OSPF маршрутизаторы обмениваются данными не со своими соседями, а со смежными маршрутизаторами. Это не одно и то же. Маршрутизатор не общается со всеми маршрутизаторами, например, внутри ЛВС, а лишь с тем, который объявлен выделенным маршрутизатором. Этот выделенный маршрутизатор смежен всем другим. У выделенного маршрутизатора есть дублер, который имеет ту же информацию, что и основной.

Периодически в ходе нормальной работы каждый маршрутизатор рассылает всем своим смежным маршрутизаторам сообщение LINK STATE UPDATE. В этом сообщении он передает информацию о состоянии своих линий и их стоимости в разных метриках для базы данных топологии соединений. Это сообщение в целях надежности идет с подтверждением. Каждое такое сообщение имеет номер, который позволяет определить, несет ли пришедшее сообщение новую информацию по сравнению с той, что есть в его базе, или старую. Маршрутизаторы рассылают эти сообщения, когда у них появляются новые линии, разрушаются старые или меняется стоимость линии.

DATABASE DESCRIPTION – сообщение, содержащее состояние всех каналов в базе данных отправителя. Сравнивая свои значения с теми, что у отправителя, получатель может определить, у кого наиболее свежая информация.

Используя сообщение LINK STATE REQUEST, маршрутизатор может в любой момент запросить информацию о любой линии у другого маршрутизатора. Наиболее свежая информация распространяется другим. Все сообщения передаются как IP-пакеты. Все типы сообщений показаны в таблице 5-52.

Таблица 5-52. Типы OSPF-сообщений

Тип сообщений	Описание
Hello (Приветствую)	Используется, чтобы получить информацию о соседях.
Link state update (Обновление состояния канала)	Предоставляет соседям стоимости отправителя.
Link state ack (Подтверждение состояния канала)	Подтверждает обновление состояния канала.
Database description (Описание базы данных)	Объявляет, какие обновления есть у отправителя.
Link state request (Запрос о состоянии канала)	Запрашивает информацию у партнера.

Маршрутизаторы в остовой области делают все, что было описано выше, а также обмениваются информацией с пограничными маршрутизаторами, чтобы уметь вычислять наилучший маршрут от любого маршрутизатора остовой области до любого другого маршрутизатора.

BGP - внешний протокол маршрутизации шлюзов.

Для маршрутизации между АС используется BGP - протокол пограничных шлюзов. Его предшественником был протокол EGP. Однако с ростом Интернета протокол EGP перестал удовлетворять требованиям к протоколу внешней маршрутизации. Основное отличие BGP от OSPF проистекает из различия в целях. При маршрутизации внутри АС основная цель - наикратчайший маршрут. При маршрутизации между АС надо учитывать также ряд условий, вызванных политикой конкретной АС.

Например, какая-то АС может не допускать маршрутизацию через себя ни для какой другой АС (у них нет другого кратчайшего пути - это ваши проблемы); может разрешать лишь определенным. Типичными примерами таких ограничений могут быть:

1. Трафик не должен проходить через определенные АС.

2. Маршрут, начинающийся в министерстве обороны России, никогда не должен проходить через Чечню.

3. Трафик через Украину может проходить, только если нет другого маршрута.

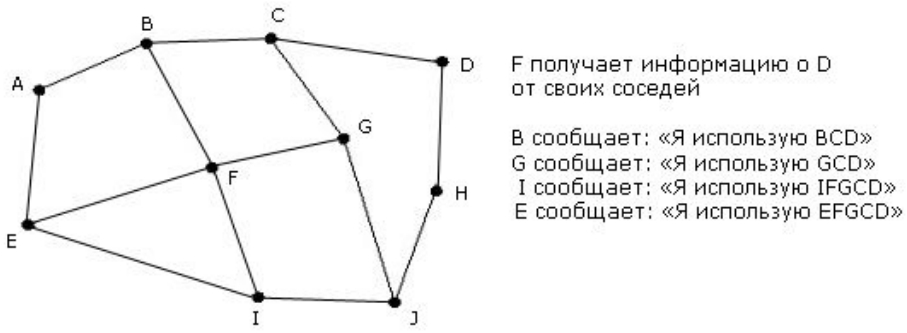
4. Трафик из IBM никогда не должен проходить через АС Microsoft.

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

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

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

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



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

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

Билет № 41.

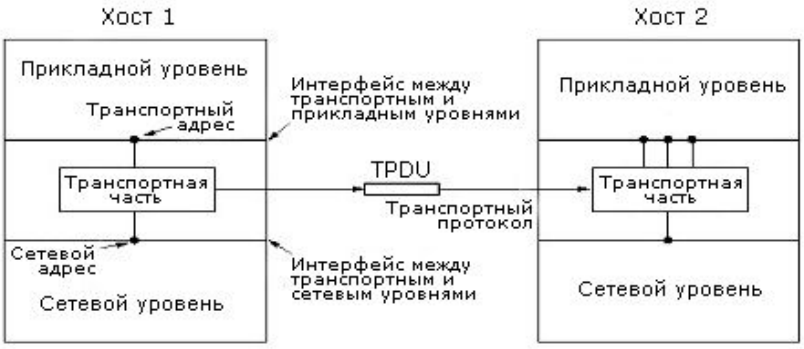
Транспортный уровень: сервис, примитивы, адресация, установление соединения, разрыв соединения, управление потоком и буферизация, мультиплексирование, восстановление разрывов.

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

Сервис.

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

Рисунок 6-1. Сетевой, транспортный и прикладной уровень



Подобно сетевому уровню, транспортный уровень также может поддерживать два вида сервиса - ориентированный на соединения и без соединений. Транспортный сервис, ориентированный на соединение, имеет много общего с аналогичным сетевым сервисом. Адресация и управление потоком также схожи на обоих уровнях.

Задача транспортного уровня в том, чтобы сделать сервис транспортного уровня для прикладного более надежным, чем сетевого для транспортного. Другое важное свойство транспортного уровня - прикладная программа, опираясь на транспортный сервис, становится независимой от сети и может работать в сети с любым сетевым сервисом. И, наконец, с транспортным сервисом работает прикладная программа, а с сетевым – транспортный уровень. Поэтому интерфейс с транспортным уровнем должен быть дружественным, удобным и эффективным.

В силу приведенных доводов первые четыре уровня называют поставщиками транспортного сервиса, а все, что выше четвертого, - пользователями транспортного сервиса.

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

- Connection establishment delay - задержка на установку соединения, определяет время между запросом на установку соединения и подтверждением его установки.
- 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, передаваемого 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. Этот параметр определяет отношение числа сообщений, при передаче которых были ошибки, включая потерянные сообщения, к общему числу переданных сообщений. 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 несанкционированного доступа третьей стороной. Косвенно он определяет, на какие затраты готов пойти пользователь для защиты своих данных от перехвата при передаче на транспортном уровне.
- Priority – приоритет, позволяет пользователю указать степень важности для него данного соединения среди остальных соединений.
- Resilience – устойчивость, определяет вероятность разрыва транспортным уровнем соединения в силу своих внутренних проблем или перегрузки.

Параметры качества сервиса определяет пользователь в момент установления транспортного соединения, указывая для каждого из них желаемое и минимальное значения. Если требуемое качество недостижимо, то транспортный уровень сразу сообщает об этом пользователю, даже не обращаясь к получателю сообщения. При этом пользователя информируют, что попытка установить соединение прошла неудачно, и о причинах неудачи. Процедура согласования параметров качества сервиса называется согласованием возможностей.

Примитивы транспортного уровня.

Примитивы транспортного уровня позволяют пользователю получить доступ к транспортному сервису. Транспортный сервис аналогичен сервису сетевого уровня. Однако между ними существует одно различие - сетевой сервис по природе своей ненадежен. Задача транспортного сервиса как раз обеспечить надежную доставку сообщений. Два процесса, соединенные между собой, ничего не должны знать о том, как физически они соединены. Один помещает данные на вход транспортного уровня, другой получает их. Задача транспортного уровня скрыть и от получателя и от отправителя все детали передачи, исправления ошибок и т.п.

Теоретически транспортный сервис может быть как ориентированным на соединения, так и нет. Однако дейтаграммный транспортный сервис - это редкость, поэтому мы будем рассматривать транспортный сервис, ориентированный на соединения.

Другое важное различие между сетевым и транспортным сервисами состоит в том, кто их использует. Сетевой сервис использует транспортный сервис, а вот транспортный использует пользователь, т.е. прикладные программы. Поэтому транспортный сервис должен быть ориентирован на пользователя, удобным и простым в использовании.

Таблица 6-3. Примитивы транспортного сервиса

Примитив	Отправляемый пакет TPDU	Значение
LISTEN	(нет)	Блокировка, пока какой-либо процесс не попытается установить соединение
CONNECT	CONNECTION REQ.	Активная попытка установить соединение
SEND	SEND	Информация об отправке
RECEIVE	(нет)	Блокировка до поступления DATA TPDU
DISCONNECT	DISCONNECTION REQ.	Данная сторона собирается прервать соединение

Использование этих примитивов может быть продемонстрировано следующим образом. Сервер приложения выполняет примитив LISTEN, в результате чего он блокируется до поступления запросов от клиентов. Клиент для установления соединения выполняет примитив CONNECT. Транспортный агент на стороне клиента блокирует клиента и посылает серверу пакет с запросом на установление соединения.

Напомним, что транспортные агенты обмениваются пакетами, которые имеют специальное название - Transport Protocol Data Unit (TPDU).

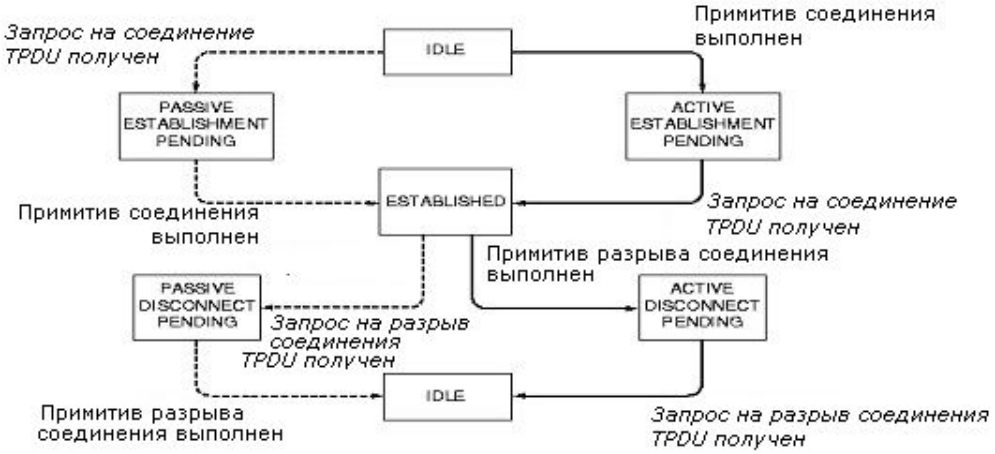
Рисунок 6-4. Взаимосвязь между кадрами, пакетами и TPDU

По примитиву CONNECT транспортный агент на стороне клиента шлет CONNECTION REQUEST TPDU. Транспортный агент сервера, видя, что сервер заблокирован по LISTEN, разблокирует сервер и посылает CONNECTION ACCEPTED TPDU. После этого транспортное соединение считается установленным и начинается обмен данными с помощью примитивов SENT и RECEIVE.

По окончании обмена соединение должно быть разорвано. Есть два варианта разрыва соединения: симметричный и асимметричный. Асимметричный разрыв предполагает, что для разрыва соединения одна из сторон посылает DISCONNECT TPDU. Получив этот TPDU, другая сторона считает соединение разорванным.

При симметричном разрыве каждое направление закрывается отдельно. Когда одна сторона посылает DISCONNECT TPDU, это значит, что с ее стороны больше данных не будет. На рисунке 6-5 показана диаграмма состояний при установлении и разрыве соединения. (Переходы, выделенные курсивом, вызываются прибытием пакета. Пунктирной линией обозначена последовательность состояний сервера, сплошной - клиента.)

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



В таблице 6-6 показан другой набор примитивов, так называемые сокеты Беркли. В этом наборе два основных отличия от того, что мы только что рассмотрели. Первые четыре примитива выполняются сервером в том порядке, как они указаны в таблице. Примитив SOCKET создает новую точку подключения к серверу, резервирует для нее место в таблице транспортного агента. Параметры обращения определяют формат адреса, тип желаемого сервиса, протокол и т.д. По примитиву BIND сервер выделяет сокету адрес. Причина, по которой адрес выделяется не сразу, в том, что некоторые процессы сами управляют своими адресами, которые жестко закреплены за ними. Второе – LISTEN - не блокирующий примитив. Он выделяет ресурсы и создает очередь, если несколько клиентов будут обращаться за соединением в одно и то же время. Примитив ACCEPT - блокирующий в ожидании запроса на соединение.

Таблица 6-6. Сокеты Беркли

Примитив	Значение
SOCKET	Создание новой точки подключения
BIND	Прикрепление локального адреса к сокету
LISTEN	Объявление готовности принимать соединения; сообщение о размере очереди
ACCEPT	Блокировка вызывающего в ожидании запроса на соединение
CONNECT	Активная попытка установить соединение
SEND	Отправка данных через данное соединение
RECEIVE	Получение данных через данное соединение
CLOSE	Прерывание соединения

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

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

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

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

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

Транспортный протокол должен решать следующие проблемы:

1. Как адресовать прикладной процесс, с которым надо установить соединение?

2. Как корректно установить соединение? Пакеты могут теряться. Как отличить пакеты нового соединения от повторных пакетов, оставшихся от старого?

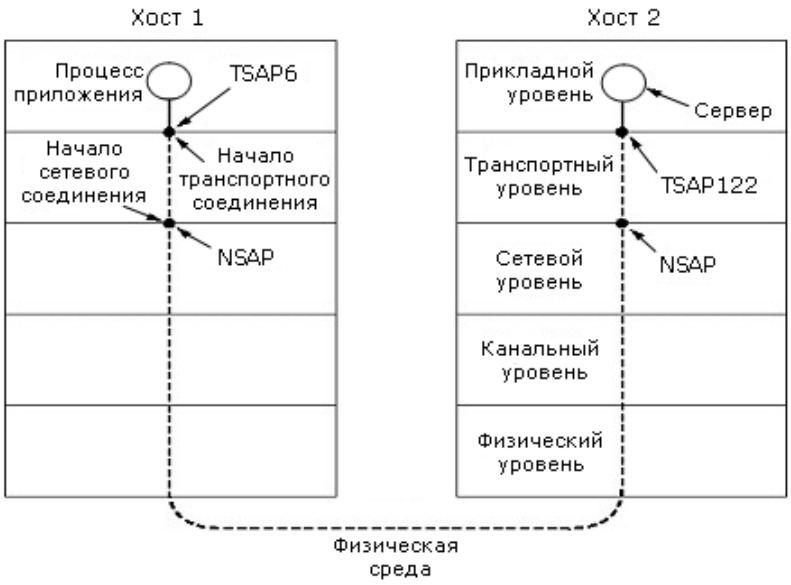
3. Как корректно разрывать соединение?

Адресация.

Проблема адресации состоит в том, как указать, с каким удаленным прикладным процессом надо установить соединение. Обычно для этого используется транспортный адрес, по которому прикладной процесс может слушать запросы на соединение. Вместо него мы будем здесь использовать термин TSAP - Transport Service Access Point. Аналогичное понятие существует и на сетевом уровне - IP-адрес - NSAP для сетевого уровня.

На рисунке 6-8 показана взаимосвязь TSAP и NSAP. Он же иллюстрирует сценарий использования TSAP для установления соединения между двумя удаленными процессами.

Рисунок 6-8. Взаимодействие TSAP и NSAP



Из этой иллюстрации не ясно лишь, как прикладной процесс на машине 1 узнает, что интересующий его сервер подключен к TSAP 122 на машине 2? Одно из возможных решений - если данный сервер всегда подключен к TSAP 122, и все процессы об этом знают.

Такое решение хорошо работает для часто используемого сервиса с длительным периодом активности, но как быть прикладным процессам пользователя, которые активизируются спорадически на короткое время? В операционной системе Unixиспользуется решение, которое называется протоколом установления начального соединения. На каждой машине есть специальный сервер процессов, который представляет все процессы, исполняемые на этой машине. Этот сервер слушает несколько TSAP, куда могут поступить запросы на TCP-соединение. Если нет свободного сервера, способного выполнить запрос, то соединение устанавливается с сервером процессов, который переключит соединение на нужный сервер, как только он освободится.

Однако есть случаи, когда этот подход с сервером процессов не работает. Не всегда можно запускать сервер сервиса по требованию пользователя. Например, файловый сервер. Он должен существовать всегда. Решение в этом случае - сервер имен. Пользователь устанавливает соединение с сервером имен, для которого TSAP известен, и передает ему имя сервиса. В ответ сервер имен шлет надлежащий TSAP. Клиент разрывает соединение с сервером имен и устанавливает его по полученному адресу.

Пусть пользователь узнал TSAP, но как он узнает, на какой машине этот TSAP расположен, какой сетевой адрес надо использовать? Ответ заключается в структуре TSAP-адреса, где указана вся необходимая информация. Этот адрес имеет иерархическую структуру.

Установление соединения.

Проблема установления транспортного соединения сложна потому, что пакеты могут теряться, храниться и дублироваться на сетевом уровне.

Типичный пример - установление соединения с банком для перевода денег с одного счета на другой. Из-за перегрузки в сети или по какой-либо другой причине может произойти большая задержка. Тогда по time_out активная сторона вышлет еще один запрос. Пакеты-дубли могут вызвать повторное соединение и вторичный перевод денег. Как быть?

Одно из возможных решений - временное TSAP. После того, как оно использовано, TSAP с таким адресом более не возникает. При этом решении не работает модель с сервером процессов, когда определенные TSAP имеют фиксированные адреса.

Другое решение - каждому транспортному соединению сопоставлять уникальный номер. Когда соединение разрывается, этот номер заносится в специальный список. К сожалению, этот список может расти бесконечно. Кроме этого, в случае сбоя машины он может быть потерян, и тогда...

Альтернативой может быть ограничение времени жизни пакетов. Достичь этого можно тремя путями:

1. ограничением конструкции подсети

2. установкой счетчиков скачков в каждом пакете

3. установкой временной метки на каждом пакете

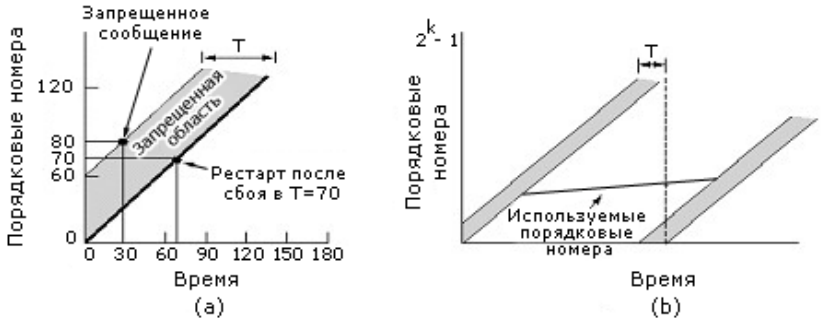
Заметим, что последний метод требует синхронизации маршрутизаторов в сети.

На практике нам надо обеспечить, чтобы стали недействительными не только сами пакеты, но и уведомления о них. Это значит, что надо ввести величину T – множитель для максимального, реального времени жизни пакета в сети. Его конкретное значение зависит от конкретного протокола и позволяет немного увеличить реальное время жизни так, чтобы по его истечении в сети не осталось ни самого пакета, ни уведомления о нем.

При ограничении времени жизни пакета можно построить безопасный способ установления соединения. Этот метод был предложен Томлинсоном (Tomlinson). Его идея состоит в следующем. Все машины в сети оснащены таймерами. Таймер работает даже в случае сбоя машины, т.е. он абсолютно надежен. Каждый таймер - двоичный счетчик достаточно большой разрядности, равной или превосходящей разрядность последовательных чисел, используемых для нумерации пакетов. При установлении соединения значения нескольких младших разрядов этого таймера берутся в качестве начального номера пакета. Главное, чтобы последовательности номеров пакетов одного соединения не приводили к переполнению счетчика и его обнулению. Эти номера можно также использовать для управления потоком в протоколе скользящего окна.

Проблема возникает, когда машина восстанавливается после сбоя. Транспортный агент не знает в этот момент, какое число можно использовать для очередного номера. Чтобы избежать повторного использования порядкового номера, который уже был сгенерирован перед сбоем машины, вводится специальная величина по времени, которая образует область запрещенных номеров (см. рисунок 6-10).

Рисунок 6-10. (a) TPDU не могут попасть в запрещенную область; (b) Проблема ресинхронизации



Машина, восстановленная после сбоя, не может выбирать номера из этой запретной зоны. Поэтому после восстановления следует подождать T сек., пока все ранее посланные пакеты не перестанут существовать. На практике поступают иначе, чтобы не тратить впустую эти T сек. Строят кривую скорости генерации номеров n=at, тогда после сбоя надо выбирать номера по формуле: n=at+T.

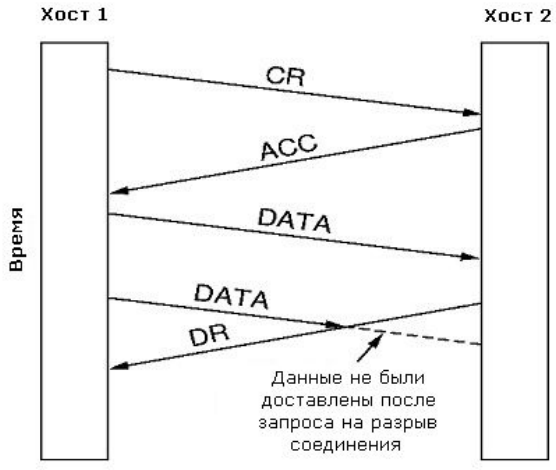
Проблема номеров может возникать по двум причинам. Либо потому, что машина генерирует слишком быстро пакеты и соединения, либо потому, что делает это слишком медленно. Чем больше разрядность счетчика последовательных номеров, тем дальше отодвигается момент попадания в запретную область.

Другая нетривиальная проблема - надежное установление соединения: пакеты ведь могут пропадать. Для ее решения Томлинсон предложил процедуру «троекратного рукопожатия» (three-way handshake). Эта процедура предполагает, что машина 1 шлет запрос на установление соединения под номером x. Машина 2 шлет подтверждение на запрос x, но со своим номером y. Машина 1 подтверждает получение подтверждения с номером y. На рисунке 6-11 (b) и (c) показано, что будет, если поступит запоздавший запрос на соединение и запрос и подтверждение на него.

Разрыв соединения.

Разрыв соединения, как уже было сказано, может быть асимметричным или симметричным. Асимметричный разрыв может привести к потере данных (см. рисунок 6-12).

Рисунок 6-12. Разрыв соединения с потерей данных



Симметричный разрыв каждая сторона проводит самостоятельно, когда она передала весь имеющийся объем данных. Однако определить этот факт не всегда просто. Здесь есть одна проблема, которая называется проблемой двух армий. Суть этой проблемы в следующем. Пусть есть две противоборствующие армии, скажем, А и В. Армия А представлена двумя группировками, между которыми расположена армия В. Суммарно ресурсы А превосходят ресурсы В, и, если обе группировки А ударят по В, то А победит. Дело лишь за тем как договорится, чтобы обе группировки ударили одновременно. Имеется сложность – гонец от А должен пройти через территорию, контролируемую В. Пусть группировка №1 шлет гонца с донесением, в котором указано время атаки. Вопрос, выслав гонца, может ли армия №1 выступать? Конечно, нет! Если гонец не доставил донесение, то атака будет отбита, ресурсы потрачены, и В победит. Выход из создавшегося положения – дождаться гонца от армии №2 с подтверждением. Пусть гонец от армии №2 прибыл. Можно ли наступать? Опять нельзя! Не получив подтверждения, что гонец доставил подтверждение, армия №2 не может выступить. Этот процесс ожидания подтверждений можно продолжать сколь угодно долго.

Внимательно изучив проблему разрыва соединения, мы придем к выводу, что ни одна армия не начнет атаки до тех пор, пока не получит подтверждения на подтверждение, и так до бесконечности. На самом деле, можно доказать, что нет протокола, который безопасно разрешает эту ситуацию. Обычно эту проблему решают, фиксируя число попыток разрыва.

Управление потоком и буферизация.

Теперь, рассмотрев, как устанавливают соединение, обратимся к тому, как им управляют. Прежде всего, рассмотрим управление потоком. Проблема управления потоком на транспортном уровне в чем-то аналогична проблеме управления потоком на канальном уровне. Различия в том, что у маршрутизатора число каналов невелико, в то время как на транспортном уровне соединений может быть очень много.

Канальный протокол сохранял пакеты как на стороне отправителя, так и на стороне получателя до тех пор, пока они не будут подтверждены. Если у нас есть 64 соединения и поле «время жизни» пакета занимает 4 разряда, то нам потребуется суммарная емкость буферов на 1024 TPDU-пакетов. Число буферов можно сократить, если есть информация о надежности сетевого уровня или о наличии буфера у получателя. На транспортном уровне отправитель сохраняет все пакеты на случай, если какой-то из них придется посылать вторично. Если получатель знает об этом, то он может иметь лишь один пул буферов для всех соединений, и, если пришел пакет и ему нет буфера в пуле, то он сбрасывается, в противном случае сохраняется и подтверждается.

Если сетевой уровень не надежный, то на транспортном уровне отправитель вынужден сохранять все отправленные пакеты до тех пор, пока они не будут подтверждены. При надежном сетевом сервисе, наоборот, отправителю нет нужды сохранять отправленные пакеты, если он уверен, что у получателя всегда есть буфер для сохранения полученного TPDU. Если такой уверенности нет, то ему придется сохранять пакеты.

Однако и в первом и во втором случае возникает проблема размера буфера. При фиксированной длине буфера естественно организовывать пул буферов одного размера. Однако при переменной длине пакетов проблема становится много сложнее. Если размер буфера устанавливать по максимальной длине пакета, то мы столкнемся с проблемой фрагментации, т.е. неэффективного использования пространства. Если по минимальной длине, то один пакет придется пересылать как несколько, с дополнительными накладными расходами. Можно установить схему динамического согласования размера буфера при установлении соединения.

Оптимальное соотношение между буферизацией на стороне отправителя или на стороне получателя зависит от типа трафика. Для низкоскоростного, нерегулярного трафика буферизацию лучше делать на обоих концах. В общем случае вопрос о количестве буферов лучше всего решать динамически. Здесь надо только позаботиться о решении проблемы потери управляющих пакетов.

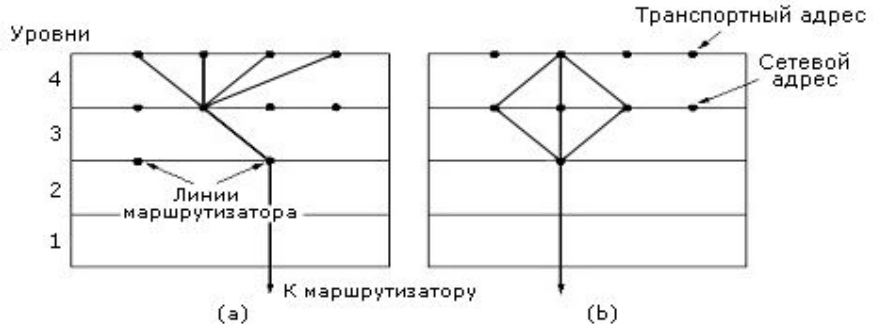
Другую проблему представляет согласование доступного числа буферов и пропускная способность сетевого уровня. Дело в том, что пропускная способность транспортной среды между двумя определенными хостами ограниченна, и если поток между ними превысит пропускную способность транспортной среды, то возникнет перегрузка. Эту проблему лучше всего решать динамически с помощью управляющих сообщений. Механизм управления потоком должен прежде всего учитывать пропускную способность подсети, а уже потом - возможности буферизации. Располагаться этот механизм будет на стороне отправителя, чтобы предотвращать накопление большого числа неподтвержденных сообщений.

Мультиплексирование.

Потребность в мультиплексировании нескольких потоков одного уровня на одном соединении, виртуальном канале, физической линии на других уровнях возникает постоянно. Эта проблема возникает и на транспортном уровне.

Например, если пользователь за терминалом установил транспортное соединение и отошел попить кофе, то транспортное соединение продолжает поддерживаться, под него резервируется буферное пространство, пространство в таблице маршрутизации и т.д. В целях удешевления стоимости транспортных соединений можно отобразить несколько транспортных соединений на одно сетевое. Такое отображение называется нисходящим мультиплексированием. Оно показано схематично на рисунке 6-15 (a).

Рисунок 6-15. (a) Восходящее мультиплексирование; (b) Нисходящее мультиплексирование



В некоторых случаях, наоборот, в целях увеличения пропускной способности по отдельным транспортным соединениям можно отобразить транспортное соединение на несколько сетевых и по каждому сетевому иметь свое скользящее окно. Тогда, быстро исчерпав возможности одного оконного буфера, можно переключиться на другое сетевое соединение и продолжить передачу по нему. В этом случае мы получим канал, пропускная способность которого равна сумме пропускных способностей отдельных каналов на сетевом уровне. Такое мультиплексирование называется восходящим (рисунок 6-15 (b)).

Восстановление после сбоев.

Восстановление после сбоев мы будем рассматривать в предположении, что транспортный агент целиком располагается на абонентской машине. Восстановление сетевого уровня достаточно просто. Если сетевой уровень предоставляет дейтаграммный сервис, то транспортный уровень знает, как исправлять подобные ситуации. При сервисе, ориентированном на соединение, транспортный уровень восстановит потерянное соединение и постарается в диалоге с транспортным агентом на другой стороне выяснить, что успели передать, а что нет.

Проблема становится сложнее, когда надо восстанавливать работоспособность машины, включая и транспортный уровень. Рассмотрим случай, когда транспортный сервер взаимодействует с

клиентами. Предположим, сервер упал и старается восстановить функционирование. Прежде всего, ему надо узнать у клиента, какое TPDU было последним неподтвержденным, и попросить повторить его. В свою очередь, клиент может находиться в одном из двух состояний: S1 – есть неподтвержденное TPDU, либо S0 – все TPDU подтверждены.

Казалось бы, все просто. Однако рассмотрим проблему внимательнее. Сервер, получив TPDU, либо сначала шлет подтверждение, а затем записывает полученное TPDU в буфер приложения, либо сначала записывает, а потом шлет подтверждение. Если сервер упал, послав подтверждение, но до того, как он осуществил запись, то клиент будет находиться после восстановления сервера в состоянии S0, хотя подтвержденное TPDU потеряно. Пусть, наоборот, сервер сначала записал TPDU, а потом упал. Тогда после сбоя сервер найдет клиента в состоянии S1 и решит, что надо повторить неподтвержденное TPDU. В результате получим повторное TPDU.

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

Билет № 42.

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

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

TCP (Transmission Control Protocol) - специально созданный протокол для надежной передачи потока байтов по соединению «точка-точка» через ненадежную сеть. TCP был сознательно разработан так, чтобы он мог адаптироваться к условиям и особенностям разных сетей, устойчиво и эффективно функционировать в условиях internet (нескольких сетей). На каждой машине, поддерживающей TCP, есть TCP-агент, который располагается либо в ядре ОС, либо в процессе пользователя, который управляет TCP-потоками и доступом к сервису IP-протокола.

TCP получает поток данных от прикладного процесса, дробит их на сегменты не более чем по 65 Кбайт (на практике не более 1,5 Кбайт) и отправляет их как отдельные IP-пакеты.

Поскольку IP-уровень не гарантирует доставку каждого пакета, то в задачу TCP входит определение потерь и организация повторной передачи потерянного. Поскольку на сетевом уровне в Internet соединения не поддерживаются, то сегменты могут поступать к получателю в неправильном порядке и задача TCP - восстановить этот порядок.

Модель сервиса TCP.

Доступ к TCP-сервису происходит через сокет. Сокет состоит из IP-адреса хоста и 16-разрядного локального номера на хосте, называемого порт. Сокеты создаются как отправителем, так и получателем. Порт - это TSAP для TCP. Каждое соединение идентифицируется парой сокетов, между которыми оно установлено. Один и тот же сокет может быть использован для разных соединений. Никаких дополнительных виртуальных соединений не создается.

Порты с номерами до 256 зарезервированы для стандартного сервиса. Например, если надо обеспечить FTP-передачу файла, то надо соединяться через 21-й порт, где находится FTP-демон. Для TELNET - через 23-й порт. Полный список таких портов можно найти в RFC 1700.

Все TCP-соединения - дуплексные, т.е. передача идет независимо в оба направления. TCP-соединение поддерживает только соединение «точка-точка». Не существует TCP-соединений от одного ко многим.

TCP обеспечивает поток байтов, а не поток сообщений. Напомним, это значит, что границы сообщений не поддерживаются автоматически в потоке.

После того, как приложение передало данные TCP агенту, эти данные могут быть отправлены сразу на сетевой уровень, а могут быть буферизованы, как поступить - решает TCP-агент. Однако в ряде случаев надо, чтобы они были отправлены сразу, например, если эти данные представляют собой команду для удаленной машины. Для этого в заголовке TCP-пакета есть флаг PUSH. Если он установлен, то это говорит о том, что данные должны быть переданы немедленно.

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

Каждый байт в TCP соединении имеет 32-разрядный номер. В сети с пропускной способностью 10 Мбит/сек. потребуется не менее часа, чтобы исчерпать все номера с 0 до 2^32. Эти номера используются как для уведомления, так и в механизме управления окнами.

TCP-агенты обмениваются сегментами данных. Каждый сегмент имеет заголовок от 20 байтов и более (по выбору) и тело переменной длины. Один сегмент может включать байты от разных отправителей, а может включать часть данных одного. Какой длины может быть тело, решает TCP-агент. Длину сегмента ограничивают два фактора. Во-первых, длина сегмента не должна превышать максимальную длину IP-пакета - 64 Кбайт. Во-вторых, каждая сеть имеет максимальную единицу передачи - MTU (maximum transfer unit), и каждый сегмент должен помещаться в MTU. В противном случае маршрутизаторам придется применять фрагментацию. При этом возрастают накладные расходы на передачу в сети, так как каждый фрагмент оформляется как самостоятельный пакет (с 20-байтным заголовком).

Основным протоколом, который используется TCP-агентом, является протокол скользящего окна. Это значит, что каждый посланный сегмент должен быть подтвержден. Одновременно с отправлением сегмента взводится таймер. Подтверждение придет либо с очередными данными в обратном направлении, если они есть, либо без данных, но с подтверждением. Подтверждение будет иметь порядковый номер очередного ожидаемого получателем сегмента. Если таймер исчерпается прежде, чем придет подтверждение, то сегмент посылается повторно.

Несмотря на кажущуюся простоту, TCP-протокол достаточно сложен и должен решать следующие основные проблемы:

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

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

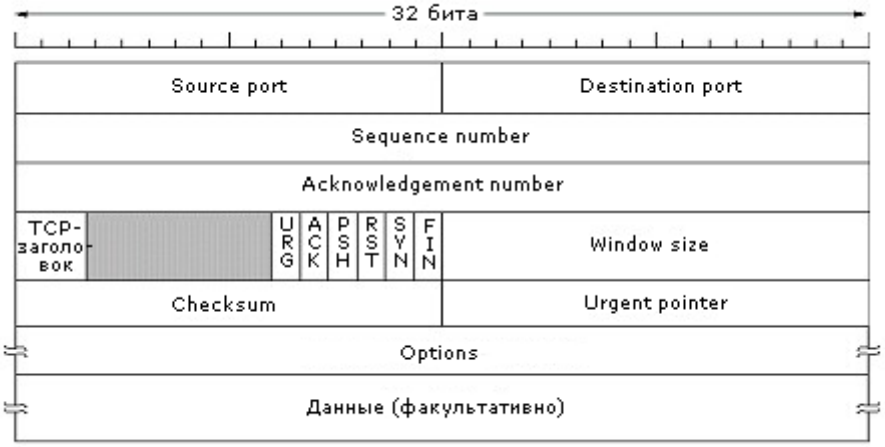
- Поля Source port и Destination port указывают сокеты на стороне отправителя и получателя соответственно.
- Sequence number и Acknowledgement number содержат порядковый номер ожидаемого байта и следующего ожидаемого, а не последнего полученного байта.

- 6-битное поле флагов.

  - Бит Urg используется вместе с полем Urgent pointer, которое указывает на начало области срочных данных.
  - ACK - 1, если поле Acknowledgement number используется, в противном случае – 0.
  - PSH - 1, если отправитель просит транспортного агента на стороне получателя сразу передать эти данные приложению и не буферизовать их.
  - RST – используется, чтобы переустановить соединение, которое по какой-либо причине стало некорректным. Получение пакета с таким флагом означает наличие проблемы, с которой надо разбираться.
  - SYN – 1, при запросе на соединение. Флаг ACK указывает на наличие или отсутствие подтверждения. SYN=1 ACK=0 – запрос на соединение, SYN=1 ACK=1 – подтверждение соединения.
  - FIN - запрос на разрыв соединения. У отправителя нет больше данных.

- Поле Window size используется алгоритмом управления окном.
- Поле Options используется для установления возможностей, не предусмотренных стандартным заголовком. Например, здесь часто указывается максимальный размер поля данных, допустимый по данному соединению.

Рисунок 6-24. Заголовок TCP



Как уже было сказано, установление TCP-соединения происходит по протоколу трехкратного рукопожатия. Флаги SYN и ASK в заголовке сегмента используются для реализации примитивов CONNECTION REQUEST и CONNECTION ACCEPTED. Флаг RST используется для реализации примитива REJECT. Это означает, что указанные выше примитивы вызывают посылку TCP-пакета с установленным соответствующим флагом.

Когда приходит запрос на соединение по определенному порту, транспортный агент проверяет, есть ли процесс, который выполнил примитив LISTEN на этом порту. Если такой процесс есть, то ему передается управление. Если такого процесса нет, то в ответ идет отказ от установления соединения.

Если два хоста одновременно пытаются установить соединение между двумя одинаковыми сокетами (коллизия), то поскольку каждое соединение идентифицируется парой сокетов, будет установлено только одно из соединений.

Таймер для последовательных номеров сегментов тактируется с частотой 4 мксек., максимальное время жизни пакета - 120 сек. Напомним, что начальный номер сегментов никогда не

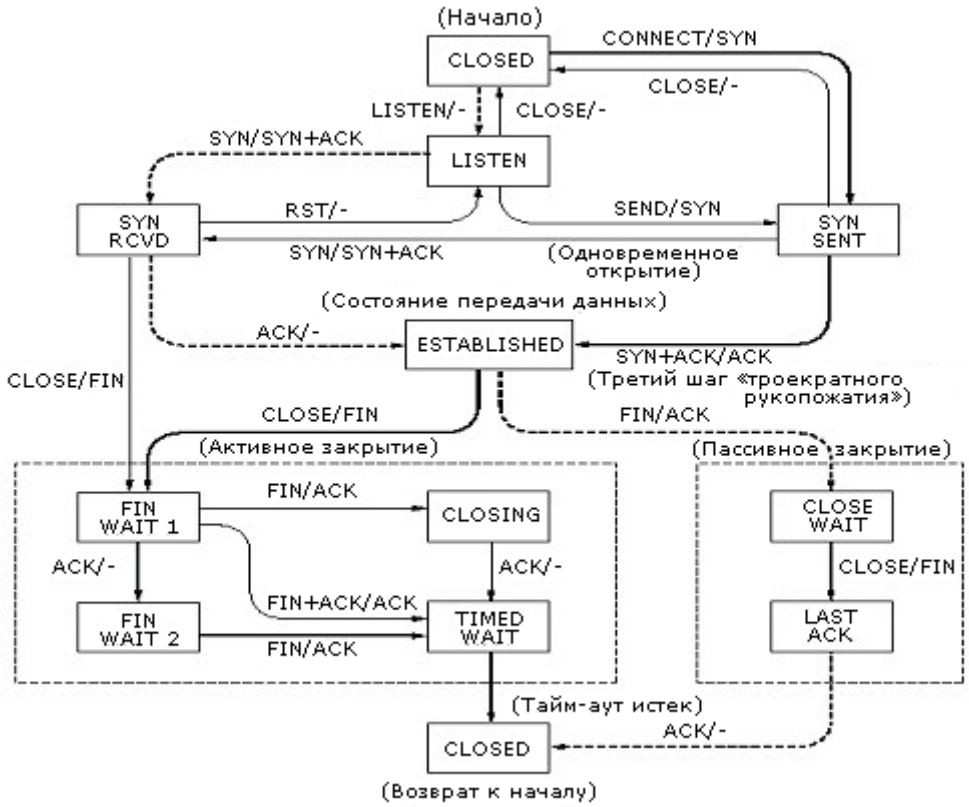
равен нулю, по соображениям, приведенным ранее. Для генерации последовательных номеров сегментов используют механизм логических часов.

TCP-соединение, как уже говорилось, - дуплексное, т.е. в каждом направлении данные передаются независимо и соединение разрывается независимо по каждому направлению. Поэтому лучше всего представлять его как два симплексных соединения. Если в очередном сегменте флаг FIN=1, то в этом направлении данных больше не будет. При получении подтверждения для этого сегмента соединение в этом направлении считается разорванным. В другом направлении передача может продолжаться сколь угодно долго. Если подтверждения на первый FIN нет в течение двух интервалов жизни пакетов, то по time-out соединение считается разорванным. Противоположная сторона также по истечении этого периода времени узнает, что никто от нее не ждет ответа. В таблице 6-18 и на рисунке 6-19 представлена процедура установления и разрыва соединения в виде диаграммы конечного автомата.

Таблица 6-18. Состояния, используемые в конечном автомате управления TCP-соединениями

Состояние	Описание
CLOSED	Нет активных или ожидающих соединений
LISTEN	Сервер ожидает входящего вызова
SYN RCVD	Запрос на соединение доставлен; ожидание подтверждения
SYN SENT	Приложение открывает соединение
ESTABLISHED	Состояние нормальной передачи данных
FIN WAIT 1	Приложение сообщило об окончании работы
FIN WAIT 2	Другая сторона согласилась разорвать соединение
TIMED WAIT	Ожидание, пока все пакеты прекратят свое существование
CLOSING	Попытка обоих сторон одновременно закрыть соединение
CLOSE WAIT	Противоположная сторона инициировала разрыв
LAST ACK	Ожидание, пока все пакеты прекратят свое существование

Рисунок 6-19. Конечный автомат управления TCP-соединениями



Управление окнами в протоколе TCP, как в управлении потоком на канальном уровне, не связано прямо с поступлением подтверждений. Предположим, что у получателя есть буферы в 4096 байт, как показано на рисунке 6-20. Если отправитель послал сегмент в 2048 байт, то получатель, получив и подтвердив этот сегмент, будет показывать окно в 2048 байт до тех пор, пока приложение не возьмет часть данных из полученного сегмента в 2048 байт. Отправитель посылает следующие 2048 байт. Теперь размер окна равен 0 байт.

Когда поле WIN=0, отправитель может послать сегмент в двух случаях. Первый - если это данные URGENT. Например, когда требуется убить процесс на удаленной машине. Второй - если это однобайтовый сегмент. Это может потребоваться, чтобы заставить получателя показать текущее состояние буфера, что очень важно, так как позволяет обойти тупик.

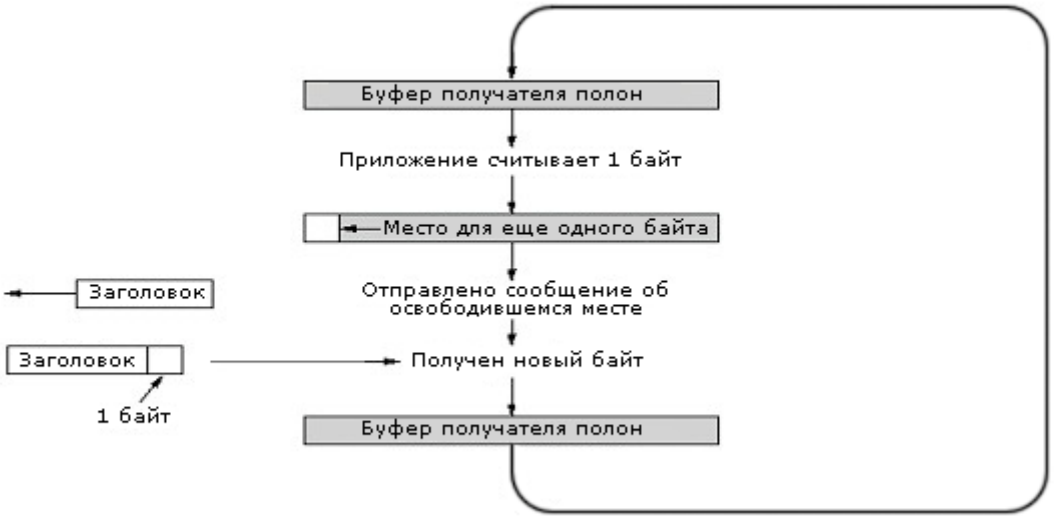
Заметим, что протокол TCP не требует от агента-отправителя сразу передавать сегмент, как только данные поступили от приложения. Эту свободу TCP использует, чтобы повысить свою производительность. Рассмотрим, к примеру, удаленный редактор TELNET. Если передавать по сети каждое движение пользователя мышкой или нажатие им клавиши на клавиатуре, то обмен будет очень не эффективным. На передачу одного символа будет приходиться передача сегмента в 160 байт. Поэтому часто при реализации протокола TCP вводят специальную задержку на посылку подтверждения и состояния окна, чтобы дать отправителю накопить буфер для отправки. Другую стратегию предложил Нагл (Nagle) – если работа идет с приложением, которое генерирует однобайтные сообщения, то надо первый байт послать, а все остальные буферизовать до тех пор, пока не придет подтверждение на посланный байт. Все буферизованные байты нужно послать одним сегментом, после чего буферизовать все байты, пока не придет подтверждение на посланный сегмент.

Алгоритм Нагла работает хорошо. Однако есть приложения, где его следует отключить, – X-Windows. Здесь перемещения мыши по экрану надо пересылать сразу без буферизации.

Другая проблема, которая может существенно понизить производительность протокола TCP – т.н. «синдром дурацкого окна» (рисунок 6-21). Он возникает, когда приложение на стороне

отправителя передает TCP-агенту данные большими блоками, а приложение на стороне получателя читает данные побайтно! В этой ситуации может произойти следующее. Буфер на стороне получателя полон и отправитель знает об этом. Приложение-получатель считывает один байт из буфера. TCP-агент получателя радостно шлет сообщение о доступном буфере в один байт. Отправитель обязан послать один байт. После чего буфер получателя опять полон, и т.д.

Рисунок 6-21. «Синдром дурацкого окна»



Кларк предложил запретить сообщать получателю в таких случаях об освободившемся месте на один байт. Получателя принуждают ждать, пока либо не освободится размер, равный максимальной длине сегмента, объявленной при установлении соединения, либо не освободится половина буфера. Отправитель также должен стараться избегать крохотных сегментов, а посылать большие.

У TCP-агента получателя также есть средства улучшить производительность соединения. Например, можно запретить приложению использовать примитив READ до тех пор, пока у агента есть данные в буфере. Конечно, такое решение увеличит время отклика, но для неинтерактивных приложений это не страшно.

Другая проблема для получателя, понижающая производительность соединения, - нарушение порядка поступления сегментов. Например, если поступили сегменты 0, 1, 2, 4, 5, 6, 7, получатель может подтвердить сегменты 0-2, забуферизовать сегменты 4-7. Тогда отправитель по time-out перешлет сегмент 3, после чего получатель подтвердит 4-7 сегменты.

Здесь мы рассмотрим, как протокол TCP борется с перегрузками. В основе всех методов лежит принцип сохранения количества пакетов: не посылать новый, пока старый не покинет сеть, т.е. не будет доставлен. Основная идея очень проста - при возникновении перегрузки не посылать новых пакетов. В протоколе TCP это реализуется динамически с помощью механизма окон.

Прежде всего, протокол TCP обнаруживает перегрузку по росту числа time_out. Если эта величина превышает некоторый предел, являющийся параметром протокола, то это фиксируется как перегрузка. Причин может быть две – шум в канале и сброс пакетов маршрутизатором. Различить их сложно. В наши дни каналы достаточно надежные, так что актуальной остается вторая причина.

Перегрузки возникают по двум причинам: нехватка буфера на стороне получателя – недостаточная емкость получателя; перегрузка внутри сети – недостаточная емкость сети.

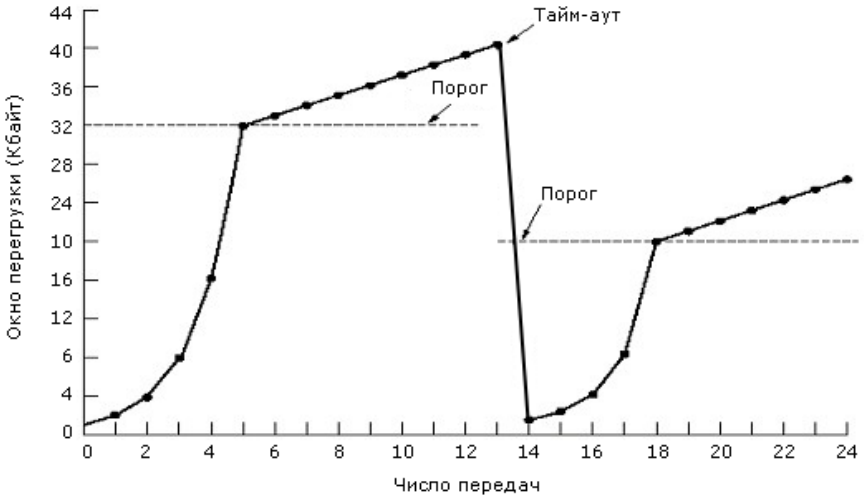
В Internet эти ситуации различаются как внутренняя емкость сети и емкость получателя. Поэтому каждый отправитель поддерживает два окна - обычное окно отправителя и окно перегрузки.

Каждое показывает количество байтов, которое отправитель может послать. Фактически отправляемое количество байтов - минимум из этих двух величин.

Сначала окно перегрузки полагают равным размеру максимального сегмента для данного соединения. Если сегмент успешно (без time_out) был передан, то окно перегрузки увеличивается вдвое. Это увеличение будет происходить до тех пор, пока либо не наступит time_out и произойдет возврат к предыдущему значению, либо размер окна перегрузки не достигнет размера окна получателя. Этот алгоритм называется slow start - медленный старт.

Другой параметр управления перегрузками в Internet – порог (threshold). Алгоритм медленного старта при возникновении перегрузки устанавливает этот параметр равным половине длины окна перегрузки, а окно перегрузки - равным размеру максимального сегмента. Окно перегрузки растет экспоненциально до тех пор, пока не сравняется с порогом, после чего оно растет линейно, пока не достигнет размера окна получателя. На этом рост прекращается до первой перегрузки. Работа этого алгоритма показана на рисунке 6-23.

Рисунок 6-23. Алгоритм управления перегрузками в internet



Протокол TCP использует несколько таймеров для управления передачей. Наиболее важный из них - таймер повторной передачи. Этот таймер устанавливают, когда отправляют сегмент. (Напомним, что так мы называем TPDU-пакет.) Если подтверждение пришло до исчерпания этого таймера, то его останавливают и сбрасывают. Если таймер исчерпан, то сегмент посылают повторно.

Здесь основная проблема - как удачно выбрать величину time_out: временной интервал, по истечении которого сегмент надо передать повторно. Как мы знаем, эта проблема встречается и на других уровнях. Однако на транспортном уровне она имеет особенность, которая заключается в следующем. На канальном уровне дисперсия величины задержки подтверждения имеет ярко

выраженный максимум. Другими словами, ее разброс невелик. Величину time_out на этом уровне устанавливают чуть больше ожидаемой величины прихода подтверждения. На транспортном уровне функция распределения величины задержки подтверждения носит более гладкий характер, чем на канальном уровне. Поэтому предсказать величину времени, которая нужна для передачи данных от источника до получателя и передачи подтверждения от получателя до источника, очень трудно. Заведомо эта величина не должна быть постоянной в силу гладкости функции распределения.

В основе используемого в протоколе TCP алгоритма, предложенного Якобсоном в 1988 году, лежит специальная переменная RTT для получения оптимального значения величины time_out (Round Trip Time), значение которой постоянно модифицируется. В этой переменной хранится наименьшее время подтверждения. При каждой передаче сегмента замеряется величина задержки подтверждения M. Если при очередной передаче подтверждение поступило прежде, чем наступил time_out, значение переменной RTT немного уменьшают, в противном случае - увеличивают по формуле:

$RTT = \lambda RTT + (1- \lambda)M$, где $\lambda=0,87$.

Однако, даже зная величину RTT, определить величину ожидания оказалось непросто. Якобсон предложил вычислять величину D - отклонения между ожидаемой величиной задержки и измеренной по формуле:

$D = \lambda D + (1-\lambda) |RTT - M|$.

Кроме этого, Якобсон показал как, зная D, вычислить величину time_out по формуле:

time_out = RTT + 4*D.

Позднее Филл Карн (Phill Karn) модифицировал эту формулу для случая повторно передаваемых сегментов. Действительно, когда поступает подтверждение для сегмента, то не ясно, то ли это подтверждение для первой передачи, толи для последней. Филла интересовал вопрос вычисления величины RTT для передачи данных по протоколам TCP/IP по КВ-радиоканалу. В результате экспериментов он показал, что для повторно передаваемых сегментов эффективно просто удваивать величину ожидания до тех пор, пока подтверждение не поступит с первого раза.

Другой важный таймер в протоколе TCP - таймер настойчивости. Он позволяет бороться со следующего типа тупиками. Когда получатель посылает сообщение с нулевым размером окна, отправитель останавливает передачу и ждет сообщения об изменении размера окна. Наконец, получатель послал это сообщение, а оно было потеряно. Все ждут. Чтобы избежать такой ситуации, используют таймер настойчивости. Если он исчерпан, то отправитель шлет сообщение получателю, напоминая ему о проблеме размера буфера.

Еще один важный таймер - таймер функционирования. Если по какой-либо причине по соединению долго не посылали сообщений, то надо проверить, функционирует ли оно. Когда этот таймер исчерпан, то соответствующая сторона шлет другой стороне запрос: «Жива ли ты?» Если ответа не поступает, то соединение считается разорванным.

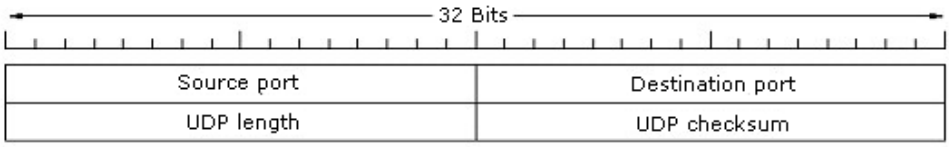
Протокол UDP.

Internet поддерживает также транспортный протокол без соединений - UDP (User Data Protocol). Протокол UDP (User Datagram Protocol) предназначен для обмена дейтаграммами между процессами компьютеров, входящих в единую сеть с коммутацией пакетов. В качестве протокола нижнего уровня UDP-протокол использует IP.

Протокол UDP предоставляет прикладным программам возможность отправлять сообщения другим приложениям, используя минимальное количество параметров протокола. Этот протокол не обеспечивает достоверность доставки пакетов, защиты от дублирования данных или от сбоев в

передаче. За исключением параметров приложения - номеров портов отправителя и получателя пакета, UDP практически ничего не добавляет к IP-дейтаграмме.

Рисунок 6-25. Заголовок UDP



Протокол UDP намного проще, чем TCP, и полезен в ситуациях, когда мощные механизмы обеспечения надежности протокола TCP не требуются или будут только помехой для решения определенного рода задач, например, аутентификации пользователей. Структура UDP-заголовка показана на рисунке 6-25.

- Source Port (16 бит). Порт отправителя. Это поле может содержать номер порта, с которого был отправлен пакет, когда это имеет значение (например, когда отправитель ожидает ответа). Если это поле не используется, оно заполняется нулями.
- Destination Port (16 бит). Порт назначения - это порт компьютера, на который пакет будет доставлен.
- Length (16 бит). Поле длины. Длина (в байтах) этой дейтаграммы, включая заголовок и данные. (Минимальное значение этого поля равно 8).
- Checksum (16 бит). Поле контрольной суммы. Контрольная сумма UDP-пакета представляет собой побитное дополнение 16-битной суммы 16-битных слов (аналогично TCP). В вычислении участвуют: данные пакета, заголовок UDP-пакета, псевдозаголовок (информация от IP-протокола), поля выравнивания по 16-битной границе (нулевые).

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

TCP и UDP в беспроводных коммуникациях.

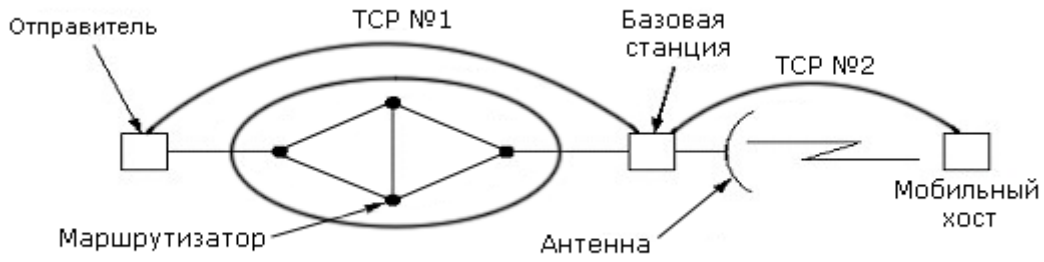
Теоретически TCP не должен зависеть от того, над какой средой он работает – оптической или беспроводной. Однако на практике дело обстоит иначе. TCP-протокол тщательно оптимизировали при разных предположениях, которые не выполняются в беспроводной среде. Так, например, предполагалось, что сетевая среда достаточно надежна и рост числа time_out – это результат перегрузки, а не потери пакетов.

В беспроводной среде потеря пакета - дело частое. Поэтому применение протокола TCP «в лоб» с протоколом Якобсона медленного старта лишь усугубит положение. Так, например, если потери составляют 20%, то при пропускной способности канала в 100 пакетов в секунду фактически будем иметь лишь 80 пакетов. Согласно алгоритму медленного старта, при увеличении числа time_out надо понизить скорость, скажем до 50 пакетов в секунду, что приведет к фактической скорости 40 пакетов.

Поэтому, если увеличилось число отказов в обычном канале, надо сбросить скорость, а если это число возросло в беспроводной среде, надо не понижать скорость, а слать пакеты повторно. Так что без знаний о среде принять решение трудно.

Основным источником проблем является то, что в беспроводной среде соединения часто неоднородные. На рисунке 6-26 показан пример. Была предложена модификация TCP – разбить такое соединение на два так, чтобы каждое стало однородным. Есть и другие решения, связанные с модификацией не самого TCP, а канального уровня для базовых станций.

Рисунок 6-26. Разбиение TCP-соединения на два



Билет № 43.

Безопасность и способы защиты данных в сетях ЭВМ: методы шифрования. Обычное шифрование. Рассеивание и перемешивание. Два основных принципа шифрования. Алгоритмы с секретными ключами (Алгоритм DES, Раскрытие DES). Алгоритмы с открытыми ключами.

Проблема безопасности сети очень многогранна и охватывает широкий спектр вопросов. Большую их часть можно разделить на следующие группы:

1. Секретность

• Конфиденциальность – только санкционированный доступ к информации (никто не может прочесть ваши письма без вашего ведома).
• Целостность - только санкционированное изменение информации (никто без вашего разрешения не может изменить данные о вашем банковском счете).

2. Идентификация подлинности пользователей

• Имея с кем-то дело через сеть, вы должны быть уверены, что это тот, за кого он себя выдает (если вы получили сообщение от налоговой инспекции уплатить определенную сумму денег, вы должны быть уверены, что это не шутка).

3. Идентификация подлинности документа

• Получив через сеть электронную версию документа, как определить, что он подлинный и не был фальсифицирован?

4. Надежность управления

• Несанкционированное использование ресурсов (если вы получите счет за телефонные переговоры, которые вы не делали, вам это вряд ли понравится).
• Обеспечение доступности ресурсов для авторизованных пользователей.

Разные люди по разным мотивам пытаются нарушить безопасность сети. В таблице 7-1 приведен список категорий людей и их возможная мотивация.

Таблица 7-1. Кто и с какими целями вызывает проблемы с безопасностью сети

Злоумышленник	Цель
Студент	Поразвлечься, читая чужую почту
Хакер	Проверить чью-либо систему безопасности; украсть данные
Торговый представитель	Заявить, что он представляет всю Европу, а не только Албанию
Бизнесмен	Узнать о маркетинговых планах конкурента
Бывший сотрудник	Отомстить за недавнее увольнение
Бухгалтер	Присвоить себе деньги компании
Брокер	Отказаться от обещания, сделанного по электронной почте
Мошенник	Украсть номера кредитных карточек для их продажи
Шпион	Изучить военный потенциал противника
Террорист	Узнать секрет бактериологического оружия

Прежде чем приступить к рассмотрению методов решений перечисленных проблем, подумаем о том, где, в каком месте стека протоколов должно располагаться обеспечение безопасности, защита сети. Одного такого места нет! Каждый уровень способен внести свой вклад. Например, на физическом уровне, чтобы контролировать доступ к физическому каналу, можно поместить кабель в опечатанную трубу, заполненную газом под давлением. Любая попытка просверлить трубу приведет к падению давления газа и срабатыванию датчика давления. Это, в свою очередь, включит сигнал тревоги.

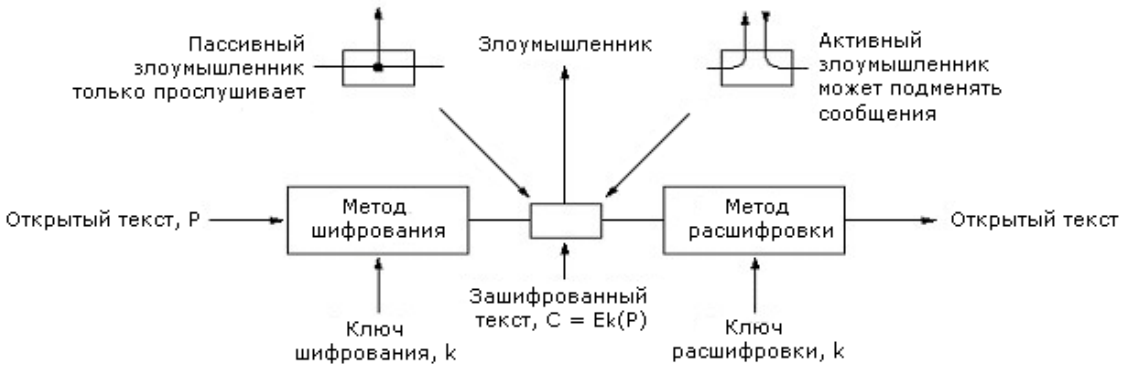
На канальном уровне данные могут быть зашифрованы на одной машине и расшифрованы на другой. Об этом шифре верхние уровни могут ничего не знать. Однако, поскольку пакет дешифруется на каждом маршрутизаторе, то в памяти маршрутизатора он может стать предметом атаки. Тем не менее, при передаче данных этот метод, называемый шифрованием канала, часто применяется в сетях.

На сетевом уровне распространенным решением является брандмауер (firewall). Напомним, что это средство, которое позволяет фильтровать как входящие, так и исходящие пакеты на сетевом уровне (см. главу 5). На транспортном уровне проблему секретности данных при передаче решают шифрованием всех сегментов транспортного соединения. Однако в сети до сих пор нет удовлетворительного решения проблемы идентификации пользователя и идентификации документа.

Обычное шифрование.

Стандартная схема шифрования такова (см. рисунок 7-2). Исходный текст, называемый также открытым текстом (plain text), обрабатывают специальной функцией со специальным параметром, называемым ключом. В результате этой обработки получают так называемый шифр-текст (ciphertext), или криптограмму. Злоумышленник аккуратно копирует все шифр-тексты. Однако, в отличие от получателя, у него нет ключа, и он не может быстро прочесть сообщение. Иногда злоумышленник может не только копировать сообщение, но позже отправлять свои, имитируя настоящего отправителя, чьи сообщения он копировал. Такого злоумышленника называют активным. Искусство создания шифра называют криптографией, а вскрытия – криптоанализом. Обе эти дисциплины образуют криптологию.

Рисунок 7-2. Схема шифрования



Основное правило шифрования - криптоаналитик знает основные приемы шифрования. Другими словами, создавать шифр, предполагая, что какие-то приемы шифрования криптоаналитик не знает, было бы самонадеянностью и непростительной ошибкой. Смена метода шифрования, его создание, тестирование, внедрение - всегда сопровождаются огромными затратами. Эти организационные моменты всегда являлись точками уязвимости любого шифра. Людей, отвечающих за применение и использование шифра всегда мучили вопросы, типа: как часто надо менять шифр? Как определить, что шифр уже вскрыт?

Один из способов повышения надежности шифра - шифрование на основе ключа. Ключ - относительно короткая строка текста, которая используется при шифровании и расшифровке сообщений. При этом, сама процедура, алгоритм шифрования могут быть известными. Тогда вся схема шифрования всем хорошо известна, менять ничего не надо, надо лишь время от времени изменять ключи.

Основа секретности – ключ. Его длина – один из основных вопросов разработки. Рассмотрим, как пример, комбинационный цифровой замок. Все знают, что его ключ – последовательность цифр. Для замка из двух цифр надо перебрать 100 комбинаций, для 3 – 1000, и т.д. Длина ключа определяет объем работы, который надо проделать криптоаналитику, чтобы вскрыть шифр. Этот объем растет экспоненциально в зависимости от длины ключа. Секретность достигается не за счет сложности алгоритма, а за счет длины ключа. Чтобы защититься от чтения почты, 64-разрядного ключа вполне достаточно. Для засекречивания государственных документов потребуется ключ в 128 или даже 256 разрядов.

С точки зрения криптоаналитика проблема дешифровки возникает в трех вариантах:

- есть только шифрограмма
- есть шифрограмма и само сообщение
- есть фрагменты исходного сообщения и их шифрограммы

Шифрование замещением.

Шифрование замещением состоит в том, что буква или группа букв замещается другой буквой или группой букв. Например, шифр Юлия Цезаря состоял в замене каждой буквы третьей следующей за ней в алфавите.

а б в г д е ё ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я

г д е ё ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я а б в

пришёл увидел победил

тулыио целжзо тсдзжло

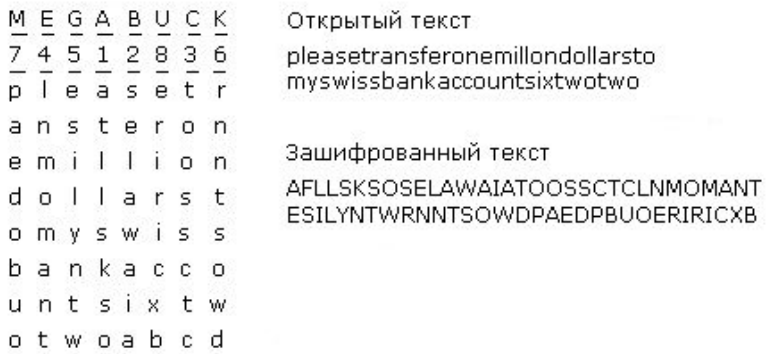
Это так называемое моноалфавитное замещение, где ключом является 33-буквенная строка, соответствующая алфавиту. Здесь возможно 33!=4x1033 ключей. Даже если на применение одного ключа компьютер будет тратить 1 мксек., то на расшифровку уйдет около 1013 лет.

Этот прием можно применять, когда алфавит исходного текста и алфавит шифрограммы разные. Например, алфавит исходного текста - кириллица, а алфавит шифрограммы - целые двухзначные числа. Однако если применить знания частотных характеристик языка, а именно, частоту встречаемости отдельных букв, двухбуквенных буквосочетаний, трехбуквенных сочетаний и т.д., то решение можно получить быстрее. Надо подсчитать частоту букв в шифр-тексте и попытаться сопоставить наиболее часто встречающимся буквам в шифре наиболее часто встречающиеся буквы в языке. Затем найти устойчивые буквосочетания, и т.д. Поэтому значение имеют дополнительные сведения, на каком языке написано исходное сообщение, его длина. Чем длиннее сообщение, тем представительнее выборка для его анализа по частоте встречаемости букв и буквосочетаний.

Шифрование перестановкой.

Шифрование перестановкой состоит в изменении порядка букв без изменения самих букв. Один из таких методов – шифрование по столбцам. Выбираем ключ – последовательность неповторяющихся символов. Символы в этой последовательности нумеруются в соответствии с их местом в алфавите. Номер один получает буква, расположенная ближе всего к началу алфавита, номер два, следующая за ней и т.д. Чем ближе к началу алфавита символ, тем меньше его номер. Шифруемый текст размещается по строкам. Длина строки – длина ключа. Получаем массив. Столбцы нумеруются в соответствии с номером символа в ключе. Каждому столбцу соответствует символ ключа, который имеет определенный номер. Упорядочим столбцы по возрастанию этих номеров. Все символы первого столбца выписываются первыми, затем символы второго и т.д. (рисунок 7-3). Этот метод можно усовершенствовать многими способами.

Рисунок 7-3. Шифрование перестановкой



Для раскрытия этого типа шифров криптоаналитик, прежде всего, должен убедиться, что он имеет дело с шифрованием перестановкой. Для этого он должен подсчитать частоту встречаемости букв в шифре. Если она соответствует частоте букв в языке, то это означает, что он имеет дело именно с перестановкой. Намек на порядок столбцов могут дать устойчивые буквосочетания в языке.

Построить нераскрываемый шифр достаточно просто. Выберем случайным образом битовую строку – это будет ключ. Текст представим как битовую строку. Разделим эту строку по длине ключа. Выполним над полученными фрагментами строки операцию «исключающее ИЛИ» (EXCLUSIVE OR). Полученные фрагменты объединим в строку. Новая строка и есть криптограмма.

Этот метод, называемый методом одноразовой подложки, имеет ряд недостатков. Трудно запомнить ключ. Его где-то надо записать или носить с собой. Это делает метод уязвимым. Например, набор одноразовых подложек можно записать на CD и закамуфлировать под запись последних хитов. Объем шифруемых данных по этому методу ограничен длиной ключа. Метод также очень чувствителен к потере символов при передаче. Правда, используя возможности компьютеров, этот метод вполне можно применять в ряде случаев.

Основная угроза раскрытия текста при криптоанализе состоит в высокой избыточности естественного языка. Например, частотные характеристики встречаемости букв, устойчивые буквосочетания, приветствия и т.д. В связи с этим Шеннон предложил два основных криптографических метода: рассеивание и перемешивание.

Цель рассеивания состоит в перераспределении избыточности исходного языка на весь исходный текст. Этот прием может быть реализован как перестановкой по некоторому правилу, так и замещением. Последнее, например, достигается тем, что замещающая комбинация зависит не только от замещаемой буквы, но от ей предшествующих букв.

Цель перемешивания состоит в том, чтобы сделать зависимость между ключом и шифр-текстом настолько сложной, насколько это возможно. Криптоаналитик на основе анализа шифр-текста не должен получать сколь-нибудь полезной информации о ключе. Этот метод как раз реализуется с помощью перестановок.

Следует учитывать, что применение порознь ни рассеивания, ни перемешивания не дает желаемого результата. Их совместное использование делает криптосистему намного более стойкой.

Два основных принципа шифрования.

Методов шифрования существует множество. Однако есть два важных, основополагающих принципа, которые должны соблюдаться в каждом методе. Первый: все шифруемые сообщения должны иметь избыточность, т.е. информацию, которая не нужна для понимания сообщения.

Эта избыточность позволит нам отличить нормально зашифрованное сообщение от подсунутого. Пример - уволенный сотрудник, который может напосылать ложных сообщений, если он знает структуру служебных сообщений в компании. Избыточность позволит обнаружить подделку. Например, если в заказах на поставку после имени заказчика стоит 3 байтовое поле заказа (2 байта – код продукта и 1 байт количество), зашифрованное с помощью ключа, то злоумышленник, прихватив с работы справочник заказчиков, может устроить компании «веселую жизнь». Для этого он сгенерирует от имени заказчиков из справочника заявки, где последние 3 байта - случайные числа. Если же длину заявки сделать не 3, а, например, 12 байтов, где первые 9 байтов - 0, и шифровать эту избыточную запись, то уже случайными числами здесь обойтись трудно, и подделку легко распознать.

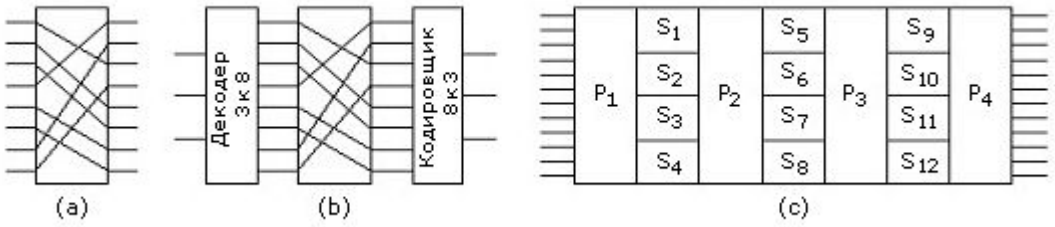
Однако такая избыточность имеет недостаток, эта избыточность может служить для криптоаналитика дополнительной информацией. Например, если в первом случае догадка о ключе и применение этого ключа к записи не дает криптоаналитику дополнительной информации о правильности ключа, то во втором случае, если в результате применение ключа-догадки, мы получим запись из 9 нулей с последующими данными, то это уже будет дополнительной информацией о правильности догадки.

Второй – надо позаботиться о специальных мерах от активного злоумышленника, который может копировать, а потом пересылать модифицированные копии. Например, временная метка позволит обнаружить сообщения, которые где-то были задержаны по непонятным причинам.

Алгоритмы с секретными ключами.

Современная криптология использует те же идеи, что и раньше. Однако акценты расставлены иначе. Если раньше алгоритм был прост, а вся сложность заключалась в ключе, то теперь, наоборот, стараются алгоритм делать как можно изощреннее. Его стараются делать таким, чтобы, если криптоаналитик получил как угодно много зашифрованного текста, он не смог ничего сделать.

Рисунок 7-4. Основные составляющие шифрования: (a) P-схема; (b) S-схема; (c) Результат

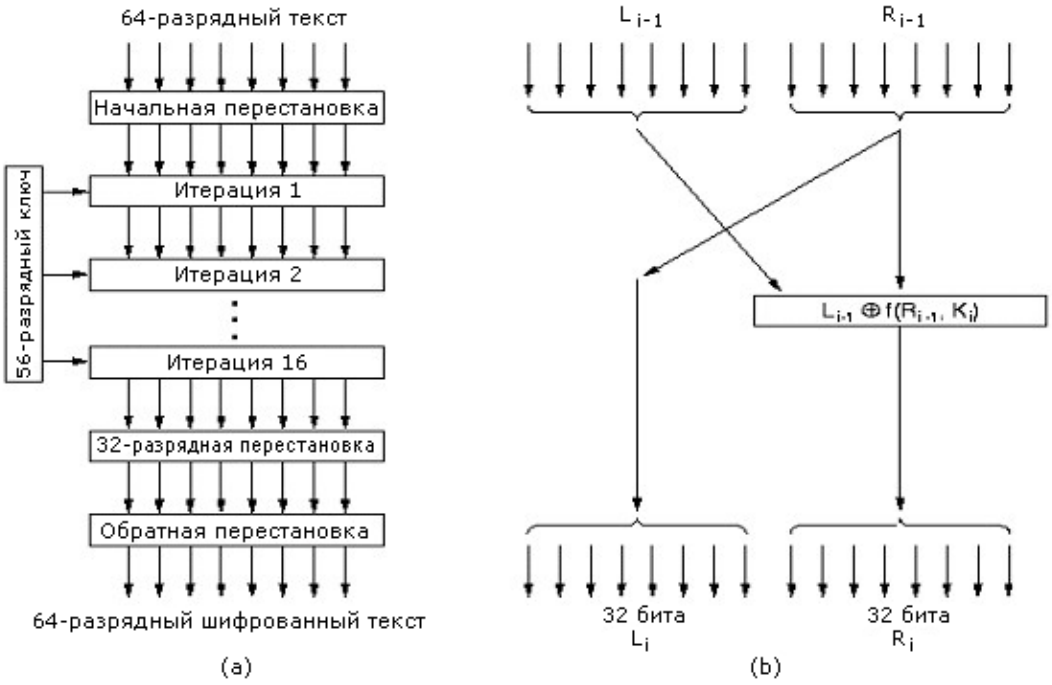


Перестановка и замещение реализуются простыми схемами, показанными на рисунке 7-4. P- и S-схемы могут объединяться в сложные каскады. В этом случае выход становится очень сложной функцией входа. На этом рисунке P-схема выполняет перестановку над словом из 8 разрядов. Схема S действует несколько сложнее, выполняя операцию замещения. Она кодирует трехразрядное слово одной из 8 линий на выходе, устанавливая ее в 1. Затем схема P переставляет эти 8 разрядов, после чего S-схема выполняет замещение 8 на 3.

Алгоритм DES.

В январе 1977 правительство США приняло стандарт в области шифрования (Data Encryption Standard), созданный на базе разработки фирмы IBM. Разница между этими разработками состояла в том, что DES предполагал 56-разрядный ключ, а у IBM он 128-разрядный. На рисунке 7-5 показана схема этого алгоритма.

Рисунок 7-5. Стандарт DES: a) Общая схема; b) Действие одной итерации



Алгоритм состоит из 19 этапов. На первом этапе исходный текст разбивается на блоки по 64 бит каждый. Над каждым блоком выполняется перестановка. Последний этап является инверсией первой перестановки. Предпоследний этап состоит в обмене местами 32 самых левых битов с 32 самыми правыми битами.

Из исходного ключа с помощью специального преобразования строят 16 частных ключей для каждого промежуточного этапа алгоритма со второго по семнадцатый.

Все промежуточные этапы схожи. У них два входа по 32 бита. Правые 32 бита входа копируют на выход, как левые 32 бита. Над каждым битом из правых 32 битов выполняется преобразование, которое состоит из EXCLUSIVE OR с соответствующим левым битом и значением специальной функции над правым битом и частным ключом данного этапа. Вся сложность алгоритма заключена в этой функции.

Функция состоит из четырех шагов. На первом строят 48-разрядный номер E как расширение 32-разрядного Ri-1 в соответствии с определенными правилами дублирования и перестановки. Над полученным номером и ключом данного шага выполняется операция EXCLUSIVE OR – это второй шаг. Результат разбивается на 8 групп по 6 бит в каждой. Каждая группа пропускается через свой S-box с четырьмя выходами каждый. На последнем шаге 8x4 бит пропускаются через P-box.

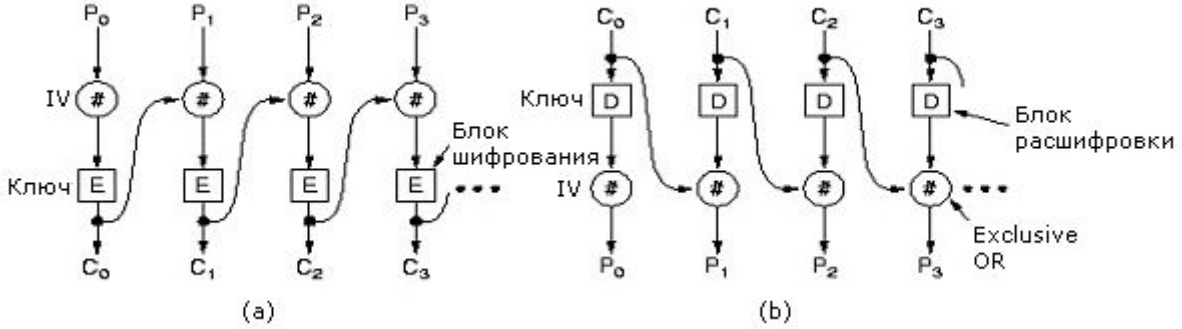
На каждой из 16 итераций используется свой ключ. До начала итераций к исходному ключу применяют 56-разрядную перестановку. Перед каждой итерацией ключ разбивают на две части по 28 разрядов каждая. Каждую часть циклически сдвигают влево на число разрядов, равное номеру итерации. Ki получают как 48-разрядную выборку из 56-разрядного ключа, полученного циклическими сдвигами с дополнительной перестановкой.

У предложенного алгоритма есть два недостатка. Он представляет собой моноалфавитное замещение с 64-разрядным символом. Всегда, когда одни и те же 64 разряда исходного текста подают на вход, одни и те же 64 разряда получают на выходе. Это свойство может использовать криптоаналитик. Одни и те же поля исходного текста попадут в одни и те же места шифра. Этим можно воспользоваться. Пример с премиями. Один сотрудник может приписать себе премию другого, если знает взаимное расположение соответствующих полей в исходной записи. Шифр ему знать ни к чему.

Второй недостаток – для начала шифрования надо иметь сразу весь 64-разрядный блок исходного текста. Это не совсем удобно, когда приходится иметь дело с интерактивными приложениями.

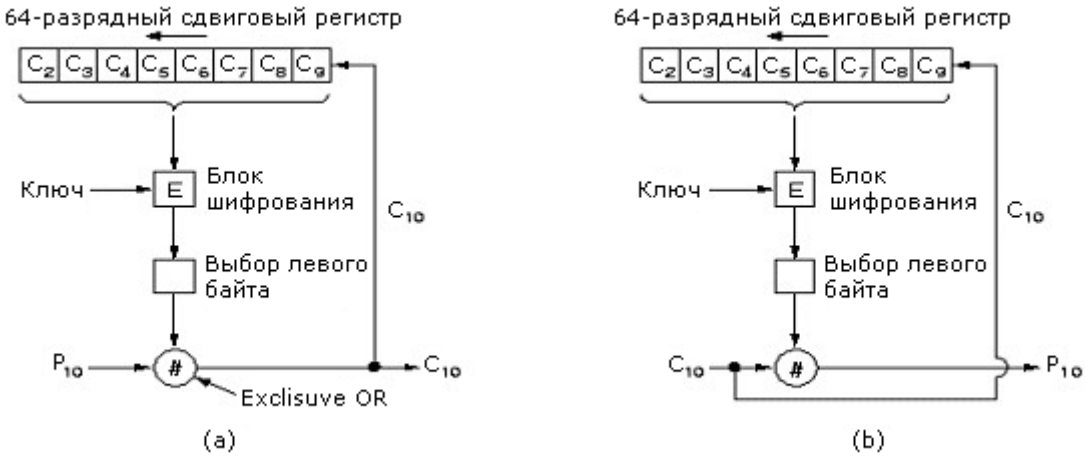
Первый недостаток устраняется модификацией алгоритма DES, в которой очередной блок исходного текста через операцию EXCLUSIVE OR смешивается с шифром предыдущего блока. Для первого блока используется специальный инициирующий блок, который передается вместе с шифром (рисунок 7-7).

Рисунок 7-7. Поблочная передача зашифрованного текста



Для устранения второго недостатка используется т.н. обратная связь по шифру (рисунок 7-8). На рисунке 7-8 (а) показана ситуация, когда первые 10 байт текста уже были закодированы, а последние 8 из них сохранены в сдвиговом регистре. Когда поступает очередной байт, выбирается самый левый байт 64-разрядного шифра, и над ними выполняется EXCLUSIVE OR, результат посылается по линии как шифр байта. Этот шифрованный байт посылается в сдвиговый регистр, а левый байт выталкивается из регистра. На стороне получателя над поступившим байтом выполняют ту же самую операцию. Этот алгоритм работает хорошо, пока оба сдвиговых 64-разрядных регистра одинаковы. Если при передаче шифрованного байта произойдет однобитовая ошибка, то пока испорченный байт не вытолкнут из регистра, расшифрованный текст пойдет с ошибкой. После этого ошибка исчезнет. Таким образом, одна битовая ошибка приведет к порче 64 бит текста. Этот алгоритм также предполагает наличие некоторого инициирующего блока, который обеспечивает работу алгоритма, пока все 8 правых байт не поступили.

Рисунок 7-8. Обратная связь по шифру



Раскрытие DES.

В 1977 году Диффи и Хеллман из Стэнфорда опубликовали проект машины стоимостью в 20 миллионов долларов, которая вскрывала DES. Достаточно было подать на вход этой машине небольшой фрагмент зашифрованного текста и соответствующий фрагмент исходного текста, как эта машина в течение дня находила ключ шифра. Существует много способов атаковать шифр. Все они основаны на распараллеливании перебора множества возможных ключей. Например, 140000 человек, вооруженных компьютерами, способными выполнять 250000 шифрований с секунду, смогут перебрать пространство 7x1016 ключей за месяц.

Основной вывод - DES не является надежным шифром и его нельзя использовать для ответственных документов. Удвоение длины ключа до 112 бит кардинально меняет ситуацию. Теперь, если использовать даже миллиард аппаратных дешифраторов, выполняющих по миллиард операций в секунду, потребуется 100 миллионов лет, чтобы перебрать пространство из 1016x1016 112-разрядных ключей. Эти рассуждения наталкивают на идею увеличения длины ключа за счет двукратного применения DES с разными ключами K1 и K2.

Однако в 1981 году Хеллман и Меркл обнаружили, что простое использование двух ключей не дает надежной схемы. Дело в том, что применение дешифрации к зашифрованному тексту дает шифр, который получается после применения первого ключа к исходному тексту. Эту мысль поясняют следующие формулы:

$Ci = EK2(EK1(Pi))$; $DK2(Ci) = EK1(Pi)$

В этом случае можно предложить следующую процедуру взлома:

1. Вычислить все возможные применения функции E к шифруемому тексту.

2. Вычислить все возможные дешифрации зашифрованного текста однократным применением дешифрирующей функции.

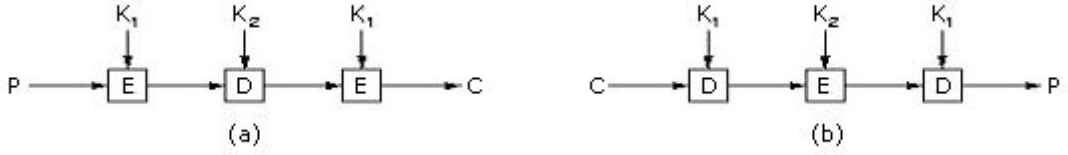
3. Отсортировать полученные таблицы и искать совпадающие строки.

4. Полученная пара номеров строк – пара ключей.

5. Проверить эту пару на совпадение шифрования; если неудачный результат, продолжить с шага 1.

Тройное шифрование совершенно меняет дело. На рисунке 7-9 показана модификация схемы шифрования с двумя ключами в три этапа - EDE-схема. Ее никому еще не удалось вскрыть. Она была положена в основу международного стандарта. Здесь может возникнуть два вопроса. Первый: почему в этой схеме используются 2, а не 3 ключа. Второй: почему используется схема EDE, а не EEE? Ответ на первый вопрос состоит в том, что двух ключей более чем достаточно для большинства применений. Использование схемы EDE вместо EEE связано с особенностями организации алгоритма DES.

Рисунок 7-9. Тройное шифрование с помощью алгоритма DES



Алгоритмы с открытыми ключами.

Идея алгоритмов шифрования с открытыми ключами была предложена в 1976 году Диффи и Хеллманом и состоит в следующем. Пусть у нас есть алгоритмы E и D, которые удовлетворяют следующим требованиям:

- D(E(P))=P
- Чрезвычайно трудно получить D из E.
- E нельзя вскрыть через анализ исходных текстов.

Алгоритм шифрования E и его ключ публикуют или помещают так, чтобы каждый мог их получить, алгоритм D также публикуют, чтобы подвергнуть его изучению, а вот ключи к последнему хранят в секрете. В этом случае взаимодействие двух абонентов A и B будет выглядеть следующим образом. Пусть A хочет послать B сообщение P. A шифрует EB(P), зная алгоритм и открытый ключ для шифрования. B, получив EB(P), использует DB с секретным ключом, т.е. вычисляет DB(EB(P))=P. Никто не прочтет P кроме A и B, т.к. по условию алгоритм EB не раскрываем по условию, а DB не выводим из EB.

Примером такого алгоритма является алгоритм RSA, предложенный Ривестом, Шамиром и Адлеманом в 1978 году. Общая схема этого алгоритма такова:

1. Выберем два больших (больше 10100) простых числа p и q.

2. Вычислим n=pxq и z=(p-1)x(q-1).

3. Выберем d, относительно простое к z.

4. Вычислим e такое, что exd=1 mod z.

Разбиваем исходный текст на блоки P так, чтобы каждый блок, как число не превосходил n. Для этого выбираем наибольшее k такое, чтобы P=2k<n. Вычисляем C=Pe(mod n), чтобы зашифровать сообщение P. Для расшифровки вычисляем P=Cd(mod n).

Для шифрования нам нужны (e, n) – это открытый ключ, для расшифровки (d, n) – это закрытый ключ. Можно доказать, что для любого P в указанном выше диапазоне функции шифрования и дешифрования взаимообратные. Безопасность этого метода основана на высокой вычислительной сложности операции разложения на множители больших чисел. Так, например, разложение на множители 200-разрядного числа потребует 4 миллиардов лет.

Рисунок 7-11. Пример использования алгоритма RSA

Открытый текст (P)		Шифрованный текст (C)			После расшифровки	
Символ	Число	P^3	P^3(mod33)	C^7	C^7(mod33)	Символ
S	19	6859	28	13492928512	19	S
U	21	9261	21	1601066541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	1	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	5	E
Вычисления отправителя				Вычисления получателя		

На рисунке 7-11 дан простой учебный пример применения RSA алгоритма для p=3, q=11, n=33, z=20, d=7, откуда e=3.

Один из основных недостатков алгоритма RSA – он работает медленно.

Билет № 44.

Безопасность и способы защиты данных в сетях ЭВМ: протоколы установления подлинности документов и пользователей (аутентификация на основе закрытого разделяемого ключа, установка разделяемого ключа, проверка подлинности через центр раздачи ключей, установление подлинности протоколом Цербер, установление подлинности, используя шифрование с открытым ключом). Электронная подпись (подпись с секретным ключом, подпись на основе открытого ключа). Сокращение сообщения.

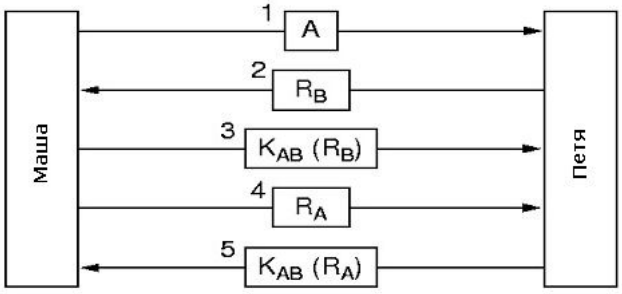
Протоколы установления подлинности (аутентификации) позволяют процессу убедиться, что он взаимодействует с тем, с кем должен, а не с тем, кто лишь представляется таковым.

Общая схема всех протоколов аутентификации такова: сторона А и сторона В начинают обмениваться сообщениями между собой или с Центром раздачи ключей (ЦРК). ЦРК - всегда надежный партнер. Протокол аутентификации должен быть устроен так, что даже если злоумышленник перехватит сообщения между А и В, то ни А, ни В не спутают друг друга со злоумышленником.

Аутентификация на основе закрытого разделяемого ключа.

Основная идея первого протокола аутентификации, так называемого «протокола ответ по вызову», состоит в том, что одна сторона посылает некоторое число (вызов), другая сторона, получив это число, преобразует его по определенному алгоритму и отправляет обратно. Посмотрев на результат преобразования и зная исходное число, инициатор может судить, правильно сделано преобразование или нет. Алгоритм преобразования является общим секретом взаимодействующих сторон. Будем предполагать, что стороны А и В имеют общий секретный ключ KAB. Этот секретный ключ взаимодействующие стороны как-то установили заранее, например, по телефону. Описанная выше процедура показана на рисунке 7-12.

Рисунок 7-12. Двусторонняя аутентификация с запросом и подтверждением



На этом рисунке:

A,B - идентификаторы взаимодействующих сторон

Ri - вызов, где индекс указывает, кто его послал

Kj - ключ, индекс которого указывает на его владельца

На рисунке 7-13 дана схема, где сокращено количество передач между сторонами по сравнению с рисунком 7-12.

Рисунок 7-13. Сокращенная двусторонняя аутентификация с запросом и подтверждением

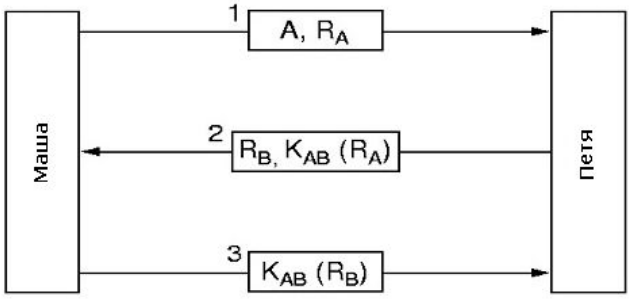
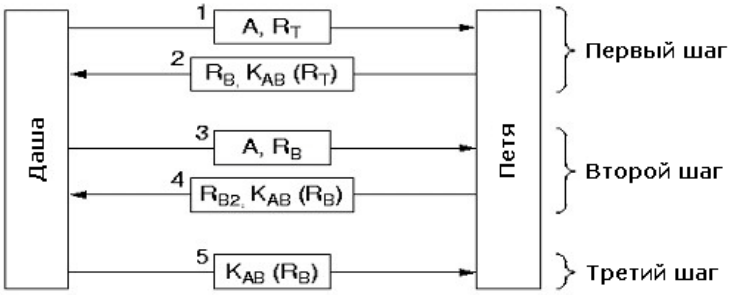


Рисунок 7-14 показывает «дыру» в схеме 7-13 и то, как злоумышленник может этой дырой воспользоваться. Это так называемая атака отражением. Идея этой атаки состоит в том, чтобы «заставить» сторону B дать некоторое число RB (см. шаг 2). А затем, на шаге 3, подсунуть стороне B это же число как свой вызов. На этот вызов сторона B, согласно протоколу, ответит преобразованным KAB(RB). В результате злоумышленник получит и RB, и KAB(RB).

Рисунок 7-14. Атака отражением



Есть несколько общих правил построения протоколов аутентификации (протокол проверки подлинности или просто подлинности):

1. Инициатор должен доказать, кто он есть, прежде чем вы пошлете ему какую-либо важную информацию.

2. Инициатор и отвечающий должны использовать разные ключи.

3. Инициатор и отвечающий должны использовать начальные вызовы из разных непересекающихся множеств.

В схеме на рисунке 7-13 все эти три правила нарушены.

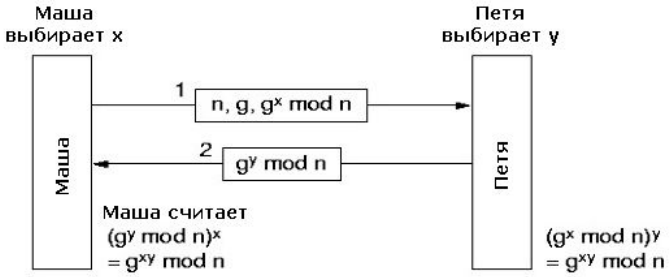
Установка разделяемого ключа.

До сих пор мы предполагали, что A и B имеют общий секретный ключ. Рассмотрим теперь, как они могут его установить. Например, они могут воспользоваться телефоном. Однако, как B убедится,

что ему звонит именно A, а не злоумышленник? Можно договориться о личной встрече, куда принести паспорт и прочее, удостоверяющее личность. Однако есть протокол, который позволяет двум незнакомым людям установить общий ключ даже при условии, что за ними следит злоумышленник.

Это протокол обмена ключом Диффи-Хеллмана. Его схема показана на рисунке 7-15. Прежде всего, A и B должны договориться об использовании двух больших простых чисел n и g, удовлетворяющих определенным условиям. Эти числа могут быть общеизвестны. Затем, A выбирает большое число, скажем x, и хранит его в секрете. То же самое делает B. Его число – y.

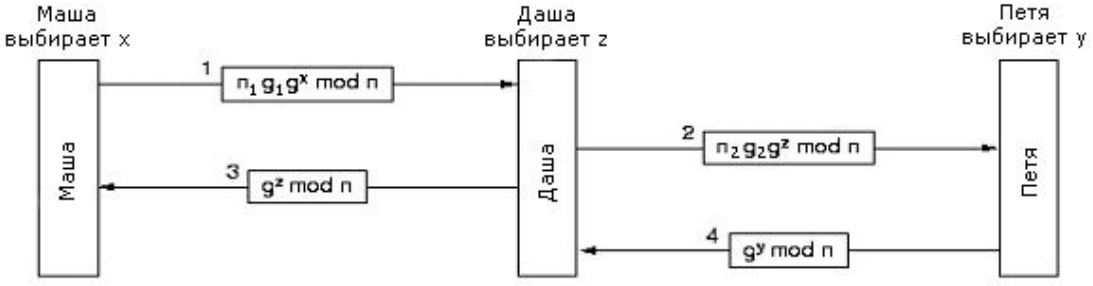
Рисунок 7-15. Обмен ключом Диффи-Хеллмана



A шлет B сообщение (n, g, gx mod n), B шлет в ответ (gy mod n). Теперь A выполняет операцию (gy mod n)x, B - (gx mod n)y. Удивительно, но теперь оба имеют общий ключ - gxy mod n! Например, если n=47, g=3, x=8, y=10, то A шлет B сообщение (47, 3, 28), поскольку 38 mod 47=28. B шлет A (17). A вычисляет 178 mod 47=4, B вычисляет 2810 mod 47=4. Ключ установлен - 4!

Злоумышленник следит за всем этим. Единственно, что мешает ему вычислить x и y, – это то, что не известно алгоритма с приемлемой сложностью для вычисления логарифма от модуля для простых чисел. Однако и у этого алгоритма есть слабое место. Рисунок 7-16 показывает его. Такой прием называется чужой в середине.

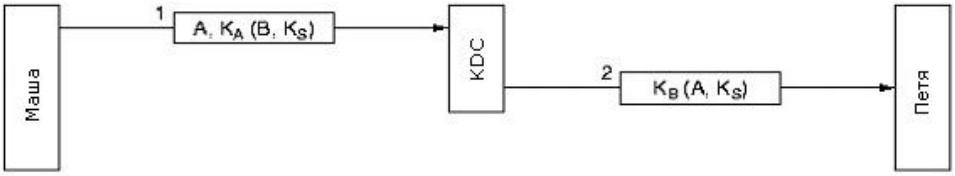
Рисунок 7-16. Взлом «чужой в середине»



Проверка подлинности через центр раздачи ключей.

Договориться с незнакомцем об общем секрете можно, но вряд ли это следует делать сразу. Кроме этого, общение с n людьми потребует хранения n ключей, что для общительных или популярных личностей может быть проблемой. Другое решение можно получить, введя надежный центр раздачи ключей (KDC).

Рисунок 7-17. Проверка подлинности через центр раздачи ключей



Идея этого протокола состоит в следующем. А выбирает ключ сессии KS. Используя свой ключ KA, А шлет в центр KDC запрос на соединение с В. Центр KDC знает В и его ключ KB. С помощью этого ключа KDC сообщает В ключ сессии KS и информацию о тех, кто хочет с ним соединиться.

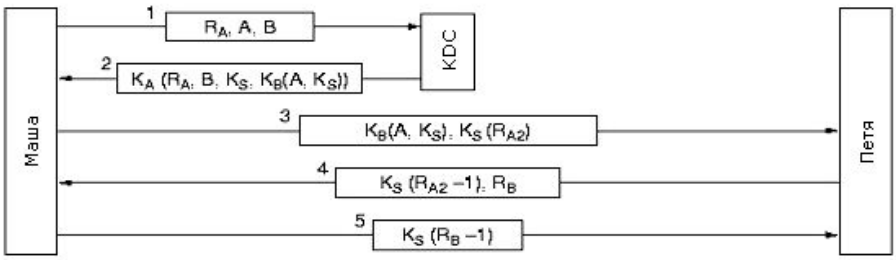
Однако решение с центром KDC имеет изъян. Пусть злоумышленник как-то убедил А связаться с В и скопировал весь обмен сообщениями. Позже он может воспроизвести этот обмен за А и заставить В действовать так, как если бы с В говорил А! Этот способ атаки называется атака подменой.

Против такой атаки есть несколько средств. Одно из них – временные метки. Это решение, однако, требует синхронизации часов. Поскольку в сети всегда есть расхождение в показаниях часов, то надо будет задать определенный интервал, в течение которого допустимо считать сообщение верным. Злоумышленник может использовать прием атаки подменой в течение этого интервала.

Другое решение - использование разовых меток. Однако при этом каждая из сторон должна помнить все разовые метки, использованные ранее. Это обременительно. Кроме этого, если список использованных разовых меток будет утерян по каким-либо причинам, то весь метод перестанет работать. Можно комбинировать решения разовых меток и временных меток.

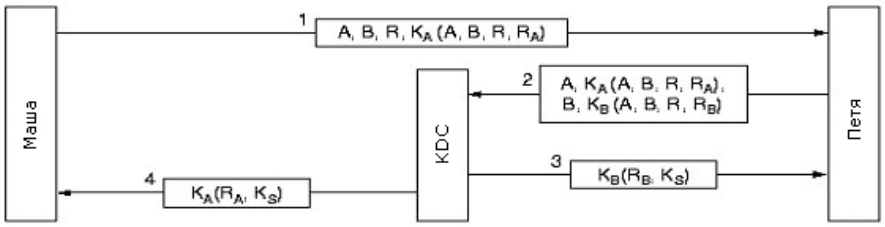
Более тонкое решение установления подлинности дает многосторонний протокол «вызов-ответ». Хорошо известным примером такого протокола является протокол Нидхема-Шредера, вариант которого показан на рисунке 7-18. Вначале А сообщает KDC, что он хочет взаимодействовать с В. KDC сообщает ключ сессии, разовую метку RA, шифруя сообщение ключом А. Разовая метка защищает А от подмены. Теперь, имея ключ сессии, А начинает обмен сообщениями с В. RA2 и RB – разовые метки, защищающие А и В от подмен.

Рисунок 7-18. Протокол аутентификации Нидхема-Шредера



Хотя этот протокол в целом надежен, но и при его использовании есть небольшая опасность. Если злоумышленник раздобудет все-таки старый ключ сессии, то он сможет подменить сообщение 3 старым и убедить В, что это А! На рисунке 7-19 приведена схема исправленного протокола, который предложили Отвей и Риис. В этой модификации KDC следит, чтобы R было одним и тем же в обеих частях сообщения 2.

Рисунок 7-19. Протокол аутентификации Отвея и Рииса



Установление подлинности протоколом «Цербер».

Протокол установления подлинности «Цербер» используется многими действующими системами. Он представляет собой вариант протокола Нидхема-Шредера и был разработан в MIT для безопасного доступа в сеть (предотвращения несанкционированного использования ресурсов сети). В нем используется предположение, что все часы в сети хорошо синхронизованы.

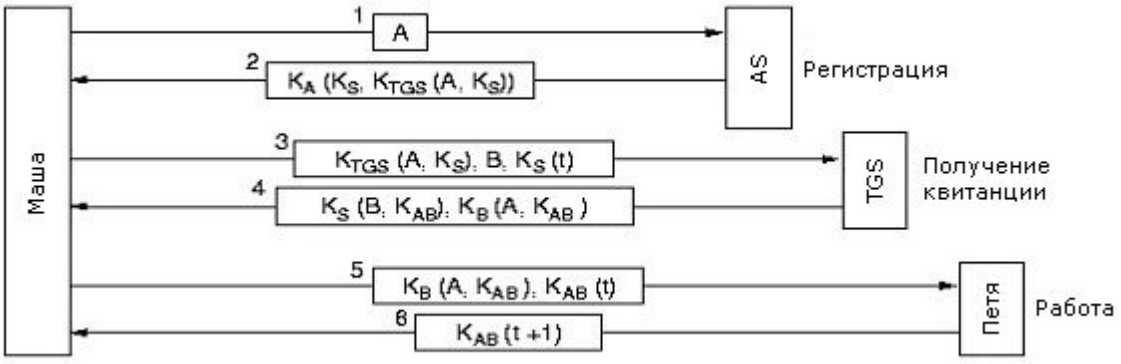
Протокол «Цербер» предполагает использование кроме рабочей станции A еще трех серверов:

- Сервер установления подлинности (СП) – проверяет пользователей на этапе login.
- Сервер выдачи квитанций (СВБ) – идентификация квитанции.
- Сервер B – тот кто должен выполнить работу, необходимую A.

СП аналогичен KDC и знает секретный пароль для каждого пользователя. СВБ выдает квитанции, которые подтверждают подлинность заказчиков работ.

На рисунке 7-20 показана работа протокола Цербер. Сначала пользователь садится за рабочую станцию и шлет открыто свое имя серверу СП. СП отвечает ключом сессии и квитанцией – <KS, KTGS(A, KS)>. Все это зашифровано секретным ключом A. Когда сообщение 2 пришло на рабочую станцию, у A запрашивают пароль, чтобы по нему установить KA для расшифровки сообщения 2. Пароль перезаписывается с временной меткой, чтобы предотвратить его захват злоумышленником. Выполнив login, пользователь может сообщить станции, что ему нужен сервер B. Рабочая станция обращается к СВБ за квитанцией для использования сервера B. Ключевым элементом этого запроса является KTGS(A, KS), зашифрованное секретным ключом СВБ. В ответ СВБ шлет ключ для работы A и B - KAB.

Рисунок 7-20. Работа протокола «Цербер»



Теперь A может обращаться непосредственно к B с этим ключом. Это взаимодействие сопровождается временными метками, чтобы защититься от подмены. Если позднее A понадобится работать с сервером C, то A должен будет повторить сообщение 3, но указать там сервер C.

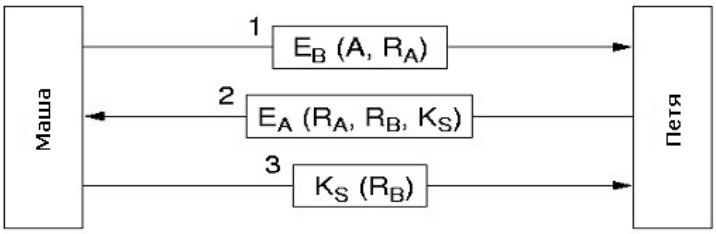
Поскольку сеть может быть очень большой, то нельзя требовать, чтобы все использовали один и тот же СП. Сеть разбивают на области, в каждой из которых - свои СП и СВБ, которые взаимодействую между собой.

Установление подлинности с шифрованием с открытым ключом.

Установить взаимную подлинность можно с помощью шифрования с открытым ключом. Пусть A и B уже знают открытые ключи друг друга. Они их используют, чтобы установить подлинность друг друга, а затем используют шифрование с секретным ключом, которое на несколько порядков быстрее.

На рисунке 7-21 показана схема установления подлинности с шифрованием открытыми ключами. Здесь RA и RB используются, чтобы убедить A и B в их подлинности. Единственным слабым местом этого протокола является предположение, что A и B уже знают открытые ключи друг друга. Обмен такими ключами уязвим для атаки типа «чужой в середине».

Рисунок 7-21. Взаимная аутентификация с использованием шифрования с открытым ключом



Ривст и Шамир предложили протокол, защищенный от атаки «чужой в средине». Это так называемый протокол с внутренним замком. Его идея заключается в том, чтобы передавать сообщения в два этапа: сначала только четные биты, затем нечетные.

Электронная подпись.

Подлинность многих юридических, финансовых и прочих документов устанавливается наличием подписи уполномоченного лица. Поскольку есть способы отличить фотокопии от подлинника, то фотокопии не рассматриваются. Такая же проблема возникает для документов в электронной форме.

Проблема электронного аналога для ручной подписи весьма сложна. Нужна система, которая позволяла бы одной стороне посылать «подписанный» документ другой стороне так, чтобы:

1. Получатель мог удостовериться в подлинности отправителя.

2. Отправитель позднее не мог отречься от документа.

3. Получатель не мог подделать документ.

Первое требование важно, например, при взаимодействии с банком, чтобы убедиться, что тот, кто проводит операцию, действительно является владельцем счета.

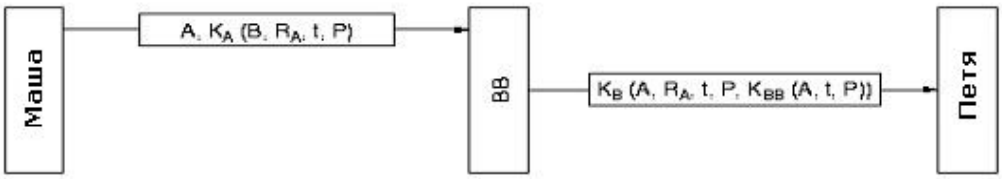
Второе требование нужно в случаях, когда, например, клиент запросил закупить тонну золота, цена которого после этого на бирже неожиданно упала. У клиента может возникнуть соблазн отказаться от своей заявки.

Третье требование предотвращает ситуации типа следующей: цена на золото в предыдущем примере неожиданно подскочила, тогда у банка может появиться соблазн изобразить, что клиент просил купить не тонну, а, скажем, килограмм золота.

Подпись с секретным ключом.

Одно из решений проблемы электронной подписи – наделить полномочиями третью сторону, которую знают все, которая знает всех и которой верят все. Назовем ее «Большой Брат» (Big Brother - BB). На рисунке 7-22 показана схема такого решения. Что произойдет, если А позже откажется от посланного сообщения? В суде В предъявит сообщение А, KBB (A, t, P), которое будет отправлено BB как непререкаемому авторитету. СД расшифрует своим ключом эту запись и все увидят А, t, P.

Рисунок 7-22. Электронная подпись посредством BB



Единственная слабость такого решения – злоумышленник может скопировать диалог между отправителем и получателем через BB и позже его повторить. Механизм временных меток позволяет уменьшить эту проблему. Кроме этого, сохранение последних RA позволяет В заметить их повторное использование.

Подпись на основе открытого ключа.

Недостаток вышеописанного решения в том, что все должны доверять СД, который может читать сообщения. Кандидатами на его роль может быть правительство, банк, нотариус. Однако далеко не все испытывают доверие к этим организациям.

Рисунок 7-23. Электронная подпись на основе открытого ключа

На рисунке 7-23 показана схема электронной подписи с использованием открытых ключей. Здесь:

- DX- закрытый ключ, EX – открытый ключ, где X – A или B.
- Предполагаем E(D(P))=P дополнительно к D(E(P))=P (этим свойством обладает алгоритм шифрования RSA).

Здесь есть два недостатка. Оба основаны на том, что схема работает до тех пор, пока сторона А либо умышленно не рассекретила свой ключ, либо не изменила его в одностороннем порядке. При наступлении судебного случая В предъявляет Р и DA(P), так как он не знает закрытый ключ А, то он не мог подделать DA(P). При этом должно быть EA(DA(P))=P, в чем суд легко может убедиться.

Если А обращается в суд, то есть Р и EA(P), что легко сопоставить с тем, что есть у В. Однако, если А заявит, что у него украли ключи, а сам тайно передаст их, либо сменит их, не сообщив об этом В, то в последнем случае текущий EA будет не применим к тому DA(P), который предъявит В. Здесь надо сопоставлять даты передачи сообщения и смены ключей.

Сокращение сообщения.

Рассмотренные методы электронной подписи критикуют за то, что они подменяют задачу установления подлинности задачей соблюдения секретности. Зачастую нужно только установление подлинности. Кроме этого, шифрование – операция медленная. Поэтому часто желательно просто поставить подпись, чтобы удостоверить подлинность сообщения. Ниже рассматривается метод, который не требует шифрования всего сообщения.

Эта схема основана на идее однозначной функции перемешивания, которая по сообщению вычисляет битовую строку фиксированной длины. Эта функция называется сокращение сообщения (MD). Она обладает тремя важными свойствами:

1. Для заданного Р вычислить MD(P) просто.

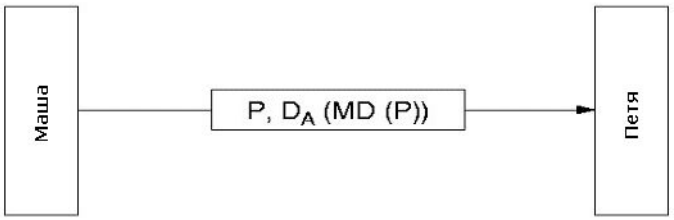
2. Имея MD(P), невозможно восстановить Р.

3. Никто не сможет подобрать таких два сообщения, что MD от них будут одинаковыми.

Этот метод можно применять как с закрытым ключом, так и с открытым. В случае закрытого ключа, как в схеме на рисунке 7-22, надо пятый элемент в сообщении от ВВ к В заменить на KBB(A, t, MD(P)). «Большой брат» подписывает не все сообщение, а лишь его сокращение. В случае возникновения спора В легко докажет в суде, что он получил именно сообщение Р, так как нет другого сообщения, которое даст такое же MD(P). Сторона А также легко докажет свою правоту, так как MD вычисляет ВВ.

В случае открытого ключа этот метод также работает. Его схема показана на рисунке 7-24. Здесь сторона А шифрует своим ключом не все сообщение, а лишь его сокращение. При этом сторона В защищена от злоумышленника, так как если он подменит сообщение, то сторона В, получив сообщение и вычислив его MD, сравнит полученное MD и обнаружит подмену.

Рисунок 7-24. Электронная подпись с использованием сокращения сообщения



Было предложено несколько алгоритмов для MD. Наиболее распространенные – MD5 и SHA.

Билет № 45.

Служба DNS. Организация, функционирование и основные протоколы почтовой службы в Internet. Протокол FTP.

Рассмотрим, как в Интернете пользователь на абонентской машине может узнать адреса других абонентских машин. Дело в том, что, несмотря на то, что любая машина в Интернете имеет уникальный IP-адрес, который представляет собой число, люди предпочитают работать с именами. Все интернет-приложения позволяют пользоваться системными именами вместо числовых адресов. Все эти имена зафиксированы в специальной распределенной базе данных DNS, которая поддерживает иерархическую систему имен для идентификации абонентских машин или узлов в сети Интернет.

При рассмотрении ее организации будем использовать почтовую аналогию. Сетевые численные адреса вполне аналогичны почтовой индексации. Машины, сортирующие корреспонденцию на почтовых узлах, ориентируются именно по индексам, и только если с индексами выходит какая-то несуразность, передают почту на рассмотрение людям, которые по адресу могут определить правильный индекс почтового отделения места назначения. Людям же приятнее и удобнее иметь дело с географическими названиями – это аналоги доменных имен.

Процесс поиска адреса в Интернете совершенно аналогичен процессу поиска индекса для письма без почтового индекса. Как определяется этот индекс? Все регионы пронумерованы – это первые цифры индекса. Письмо пересылается на центральный почтамт региона, где имеется справочник с нумерацией районов этого региона – это следующие цифры индекса. Теперь письмо идет на центральный почтамт соответствующего района, где уже знают все почтовые отделения в подопечном районе. Таким образом, по географическому адресу определяется почтовый индекс, ему соответствующий. Так же определяется и адрес компьютера в Интернете, но путешествует не послание, а запрос вашего компьютера об этом адресе. И, в отличие от случая с почтой, информация доходит до вас, как если бы районный почтамт места назначения отправлял вам письмо, любезно уведомляя вас на будущее об индексе, которого вы не изволили знать.

Конечно, такое именование имеет свои собственные проблемы. Прежде всего, следует убедиться, что никакие два компьютера, включенные в сеть, не имеют одинаковых имен. Необходимо также обеспечить преобразование имен в числовые адреса, чтобы машины (и программы) могли понимать нас, людей, пользующихся именами: техника по-прежнему общается на языке цифр.

Вначале Интернет был не велик, иметь дело с именами было довольно просто. NIC создал регистратуру. Можно было послать запрос, и в ответ высылали список имен и адресов. Этот файл называется “host file” (файл абонентских машин), регулярно распространялся по всей сети – рассылался всем машинам. Имена были простыми словами, все они были уникальны. Если вы использовали имя, то ваш компьютер просматривал этот файл и подставлял вместо имени реальный числовой адрес. Так же, как работает телефонный аппарат со встроенным списком абонентов. Всем хватало простых имен, был один Мокий, один Мокридий, один Пафнутий и одна Перепетуя.

По мере развития и расширения Интернета возрастало количество абонентских машин, а потому разрастался и упомянутый файл. Возникали значительные задержки при регистрации и получении имени для новых компьютеров, стало затруднительно изыскивать имена, которые еще никто не использовал, слишком много сетевого времени затрачивалось на рассылку этого огромного файла всем машинам, в нем упомянутым. Стало очевидно, что при таких темпах изменений и роста сети нужна распределенная оперативная система, опирающаяся на новый принцип. Таковая была создана, ее назвали доменной системой имен – DNS, а способ адресации – способом адресации по доменному принципу. DNS также иногда называют региональной системой наименований.

Структура региональной системы имен.

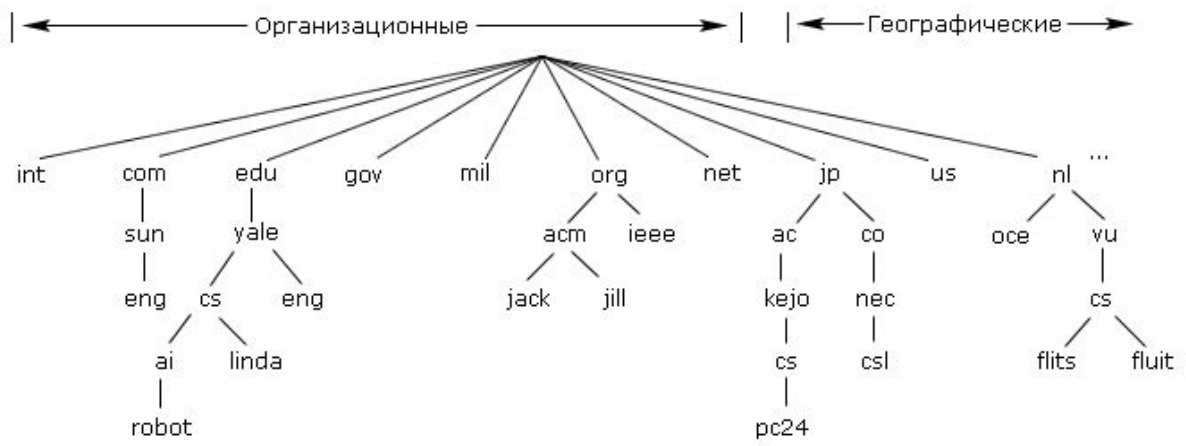
Доменная система имен - это метод назначения имен путем передачи сетевым группам ответственности за подмножество имен в своем домене. Каждый уровень этой системы называется

доменом. Домены в имени отделяют друг от друга точками: cs.msu.su, math.msu.su. В имени может быть различное количество доменов, но практически их не более пяти. Первым в имени стоит название абонентской машины - реального компьютера с IP-адресом. Это имя создано и поддерживается группой (например, компьютер redsun в группе cs (факультет вычислительной математики и кибернетики)), к которой он относится. Группа входит в более крупное подразделение msu (университетское объединение – сеть МГУ), которое, в свою очередь, является частью национальной сети (например, стран бывшего СССР, домен su).

Все пространство доменов распределено на зоны. Как это происходит, мы кратко рассмотрим ниже. Имена зон можно условно разделить на организационные и географические. На высшем уровне в этой иерархии зарегистрированы следующие организационные зоны:

- com – commercial (коммерческие)
- edu – educational (образовательные)
- gov – government (правительственные)
- mil – military (военные)
- net – network (организации, обеспечивающие работу сети)
- org – organization (некоммерческие организации)

Рисунок 7.25. Часть пространства доменных имен в Интернете



В данный момент, чтобы разгрузить домен «com», собираются создать несколько новых доменов.

В организационных зонах обычно размещаются непосредственно домены организаций.

Каждая страна (государство) имеет свой географический домен из двух букв, например:

- ae – United Arab Emirates (Объединенные Арабские Эмираты)
- au – Australia (Австралия)
- be – Belgium (Бельгия)
- br – Brazil (Бразилия)
- by – Belarus (Белоруссия)

С левого конца доменного имени находятся имена абонентских машин. Имена бывают собственные и функциональные. Имена собственные каждый придумывает в меру фантазии: машинам присваиваются имена членов семьи, животных, растений, музыкантов и артистов, литературных персонажей – кто во что горазд.

Имена функциональные вытекают из функций, выполняемых машиной:

- www – Сервер HTTP (WWW)
- ftp – FTP-сервер
- ns, nss, dns – Сервер DNS (Name)
- mail – Mail-сервер
- relay – Mail Exchanger
- *proxy – соответствующий proxy-сервер

Считается нежелательным присваивать какой-либо машине функциональное имя – в любой момент может потребоваться перенести соответствующую функцию на другую машину. Для этого лучше всего использовать псевдонимы, которые перенаправляют запросы к данному имени на записи, относящиеся к другому имени.

Группа может создавать или изменять любые принадлежащие ей имена. Если группа cs решит ввести в эксплуатацию новый компьютер и назвать его chronos, то для этого ни у кого не надо спрашивать разрешения, все, что от нее требуется, - это добавить новое имя в соответствующую часть соответствующей базы данных, и рано или поздно каждый, кому потребуется, узнает об этом имени. Аналогично, если в МГУ решат создать новую группу, например college, они (домен msu) могут также это сделать, ни у кого не спрашивая разрешения. Тогда, если каждая группа придерживается этих простых правил и поддерживается уникальность имен компьютеров в группе, то у любых разных систем в сети Интернет будут всегда разные имена.

Описанный выше механизм аналогичен механизму присвоения почтовых адресов. Названия всех стран различаются. Различаются названия всех областей, республик в Федерации, и эти названия утверждаются в государственном масштабе из центра (конечно, обычно сами регионы заботятся об уникальности своих названий, поэтому здесь царит полная демократия: как республика хочет, так она и называется). В республиках – субъектах федерации – решают вопросы о названиях районов и округов, в пределах одной республики они различаются. Аналогично далее с городами и улицами городов. В разных городах могут быть улицы с одинаковыми названиями, например, почти во всех городах СССР были улицы Ленина и Мира. Однако то были улицы в разных городах. В пределах же одного населенного пункта улицы всегда имеют разные названия, причем именование улиц контролирует соответствующий центральный орган местной администрации (мэрии, сельсовета, горсовета).

Поскольку Интернет – сеть всемирная, то нужен был механизм распределения имен на самом верхнем, межгосударственном уровне. Сейчас принята двухбуквенная кодировка государств. Это оговорено в RFC 822. Так, например, домен «Канада» называется «ca», бывший СССР – «su», США – «us» и т.д. США включили в эту систему структурирования для всеобщности и порядка. Всего же кодов стран почти 300. Единый каталог Интернета находится в SRI International (Менло-Парк, Калифорния, США) – в государственной организации.

Поиск адреса по доменному имени.

Теперь, после того как мы узнали, как соотносятся домены и создаются имена, познакомимся с тем, как использовать эту замечательную систему. Она работает автоматически. Нам не надо разыскивать адрес, соответствующий имени или подавать специальную команду для его поиска (в UNIX – команда nslookup). Все компьютеры в Интернете способны пользоваться доменной системой.

Когда используют имя, например, www.lvk.cs.msu.su, надо преобразовать его в адрес. Для этого приложение начинает запрашивать помощь у DNS-серверов. Эти приложения обладают соответствующей базой данных, в число обязанностей которых входит обслуживание такого рода запросов. DNS-сервер начинает обработку имени с его правого конца и двигается по нему влево, т.е. сначала производится поиск адреса в самой верхней группе иерархии, потом постепенно поиск опускается по иерархии, тем самым сужая область поиска. Однако с целью сокращения поиска, на первом шаге опрашивается локальный узел DNS. Здесь возможны три случая:

• Местный сервер знает адрес, потому что этот адрес содержится в его части всемирной базы данных. Например, если вы подсоединены к сети Института Физики Высоких Энергий (IHEP), то ваш местный сервер должен обладать информацией о всех компьютерах локальной сети этого института.

• Местный сервер знает адрес, потому что кто-то недавно уже запрашивал его. Когда запрашивается адрес, сервер DNS придерживает его у себя в памяти некоторое время, как раз на случай, если кому-нибудь потребуется тот же адрес – это повышает эффективность системы.

• Местный сервер адрес не знает.

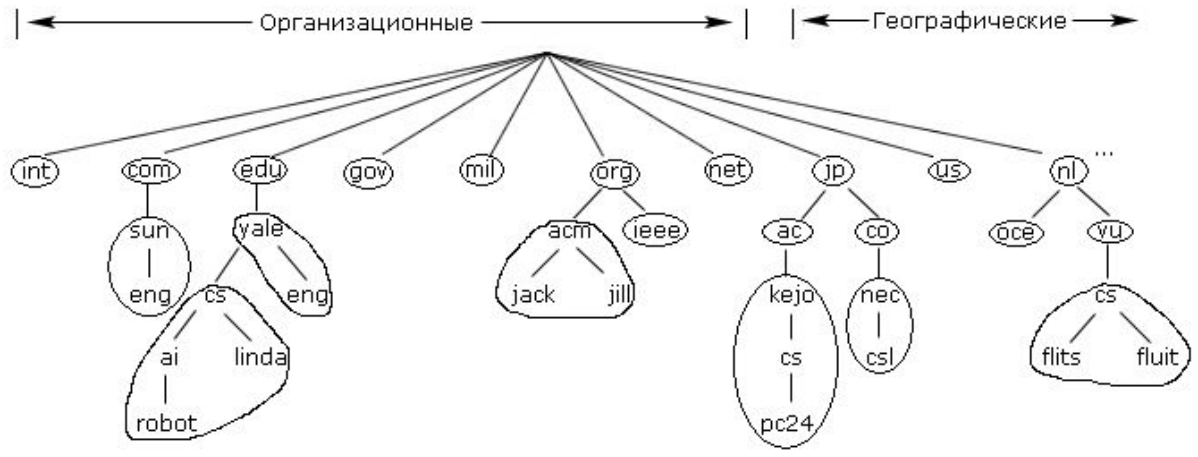
В последнем случае местный сервер обращается к корневому серверу. Это сервер, который знает адреса серверов имен высшего уровня (самых правых в имени), в нашем случае это уровень государств (ранга домена su). У него запрашивается адрес компьютера, ответственного за зону su. Местный DNS-сервер связывается с этим сервером, расположенным на вершине иерархии, и запрашивает у него адрес сервера, ответственного за домен msu.su. Теперь уже запрашивается сервер, отвечающий за домен msu, потом опрашивается сервер домена cs, затем – lvk, и у него запрашивается адрес рабочей машины www.

Как уже было сказано, для повышения эффективности поиск начинается не с самого верха, а с наименьшего домена, в который входите и вы, и компьютер, имя которого вы запросили.

Серверы имен.

Должно быть ясно, что нет и не может быть единого сервера, содержащего всю базу DNS. Его не может быть как в силу вопросов безопасности и надежности функционирования сети Интернет, так и в силу производительности. Чтобы сделать базу распределенной, все пространство имен доменов разбивают на непересекающиеся зоны. На рисунке 7-26 показан пример такого разбиения. Границы зоны определяет администратор зоны. Каждая зона покрывает часть дерева доменов, в нее входят сервера имен этих доменов. Обычно в каждой зоне есть основной сервер зоны и несколько вспомогательных серверов имен. Часто из соображений надежности сервер зоны располагают вне зоны.

Рисунок 7-26. Часть пространства доменных имен с делением на зоны



Весь процесс поиска IP-адреса по имени домена, описанный в разделе 7.2.2., реализуют сервера имен. Если запрос относится к юрисдикции того сервера имен, к которому обратились, т.е. запрашиваемый домен находится в ведении данного сервера имен, тогда этот сервер генерирует ответ, содержащий записи всех ресурсов, соответствующих запросу. Этот ответ считается авторитетным, т.е. содержащаяся в нем информация считается a priori верной. Если запрос относится

к удаленному домену, то сервер имен генерирует запрос к соответствующему удаленному серверу имен.

Однако прежде чем обратиться к удаленному серверу имен, обращающийся сервер посмотрит записи ресурсов в своей кэш-памяти. Записи в кэш-памяти не являются авторитетными. Время актуальности содержащейся в них информации, определяет поле времени жизни (см. назначение поля времени жизни в разделе 7.2.4.).

Записи ресурсов.

С каждым доменом связано множество ресурсов, отнесенных к этому домену. Эти записи хранятся в базе DNS. Когда происходит обращение к DNS с каким-либо именем, в ответ приходит не только IP-адрес, но и запись о ресурсах, соответствующих указанному имени.

Запись ресурса состоит из пяти полей: «Имя домена» (Domain name), «Время жизни» (Time to live), «Класс» (Class), «Тип» (Type), «Источник полномочий» (SOA - Start Of Authority).

В поле «Имя домена» указано имя домена, к которому относится эта запись. При обращении к базе DNS с таким ключом в ответ поступают все записи, у которых в этом поле указано заданное имя.

Поля «Время жизни» указывает интервал времени в секундах, в течение которого поля этой записи не меняются. Например, 86 400 - это число секунд в сутках. Если в этом поле указано такое число, то это значит, что запись меняется не чаще одного раза в сутки.

В третьем поле «Класс» указано IN, если ресурс, к которому относится эта запись, является ресурсом Интернета. Здесь могут быть и другие значения, но они встречаются редко.

Значения поля «Тип» указаны в таблице 7-28 и рассматриваются ниже.

Таблица 7-28. Основные типы записи ресурса DNS

Тип	Значение	Описание
SOA	Источник полномочий	Параметры данной зоны
A	IP-адрес хоста	32-разрядное число
MX	Обмен электронной почтой	Приоритет, домен, принимающий электронную почту
NS	Сервер имен	Имя сервера для данного домена
CNAME	Каноническое имя	Имя домена
PTR	Указатель	Псевдоним для IP-адреса
HINFO	Описание хоста	Центральный процессор и оперативная система в кодировке ASCII
TXT	Текст	Неинтерпретируемый ASCII-текст

В поле «Источник полномочий» указано имя источника информации о зоне сервера имен (об этом сервере будет сказано ниже). Указывается также адрес электронной почты администратора сервера имен и другая служебная информацию. Если в этом поле указано значение A, то это значит, что в следующем поле указан IP-адрес этого ресурса. Если там указано значение MX, то 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 нужна при пересылке почты в другие домены.

Записи типа CNAME и PTR позволяют создавать псевдонимы. Например, человек может иметь несколько адресов электронной почты, но все они будут относиться к одному почтовому ящику.

Запись CNAME отличается от записи PTR тем, что интерпретация последней зависит от контекста ее использования.

Запись типа HINFO позволяет определять тип машины и операционной системы, соответствующего ресурса.

Замечания по региональной системе имен.

Распространено несколько заблуждений, с которыми вы можете столкнуться, имея дело с именами. Приведем несколько верных утверждений в качестве опорных, чтобы вывести вас из таких заблуждений или предостеречь от них.

- Доменная система именования указывает на то, кто ответственен за поддержку имени, то есть в чьем ведении оно находится. Однако она ничего не сообщает о владельце компьютера, где эта машина находится (несмотря на коды стран). Вполне возможно существование в Антарктиде машины с именем inr.msk.su. Это, конечно, не нормально, но никаким законам не противоречит.
- Понятия доменного имени и сети, вообще говоря, не связаны. Часто доменные имена и сети перекрываются, и жестких связей между ними нет: две машины одного домена могут не принадлежать к одной сети. Например, системы io.cs.msu.su и fox.cs.msu.su могут находиться в совершенно разных сетях: доменные имена указывают на ответственного за домен.
- У машины может быть много имен. В частности, это верно для машин, предоставляющих какие-либо услуги, которые в будущем могут быть помещены под опеку другой машины. Когда эти службы будут перемещены, то имя, под которым эта машина выступала в качестве такого сервера, будет передано новой машине-серверу вместе с услугами, - 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 занимается обслуживанием. Имена, по смыслу относящиеся к службе, называются каноническими именами. В Интернете они встречаются довольно часто.

Для связи имена необязательны. Как-нибудь вам придет сообщение: «адресат неизвестен», что означает, что Интернет не может преобразовать использованное вами имя в адрес, – имя более недействительно в том виде, в котором его знает ваш компьютер. Однажды заполучив числовой эквивалент имени, ваша система перестает использовать для связи на машинном уровне доменную форму адреса. Запоминать лучше имена, а не числовые адреса. Некоторым кажется, что система имен это «еще одно звено в цепи, которое может выйти из строя». Но эти адреса привязаны к конкретным точкам сети. Если компьютер, предоставляющий некие услуги, переносится из одного здания в другое, его сетевое расположение, а значит и адрес, скорее всего изменятся. Имя же менять не надо и не следует. Когда администратор присваивает новый адрес, ему нужно только обновить запись имени в базе данных так, чтобы имя указывало на новый адрес. Так как имя работает по-прежнему, вас совершенно не должно заботить то, что компьютер расположен уже в другом месте.

Региональная система имен, возможно, и выглядит сложно, но это одна из тех составляющих, делающих общение с сетью более простым и удобным. Несомненное преимущество доменной системы состоит в том, что она разбивает громадье Интернета на набор вполне обозримых и управляемых частей. Хотя сеть включает миллионы компьютеров, все они поименованы, и именование это организовано в удобной и рациональной форме, что упрощает работу.

Электронная почта.

Архитектура почтовой системы состоит из двух основных компонентов - агента пользователя и агента передачи сообщений. Первый отвечает за интерфейс с пользователем, составление и отправку сообщений. Второй – за доставку сообщения от отправителя к получателю.

Обычно почтовая система поддерживает пять базовых функций:

• Композиция. Обеспечивает создание сообщений и ответов. Хотя для формирования тела сообщения может быть использован любой текстовый редактор, система обеспечивает заполнение многочисленных полей заголовка сообщения. Например, если формируется ответ, система автоматически выделит адрес из исходного сообщения и подставит его как адрес получателя.

• Передача. Эта функция обеспечивает передачу сообщения от отправителя к получателю без вмешательства пользователей.

• Отчет перед отправителем о доставке: было ли сообщение доставлено? Было ли отвергнуто? Было ли потеряно? Для многих приложений эти отчеты крайне важны.

• Показ сообщения является существенной функцией почтовой службы. Часто она должна выполнять перекодировку, изменять формат и т.д.

• Размещение - это последний этап, на котором определяют, что делать с сообщением: надо ли его уничтожить после прочтения (или до), если сохранить, то где. Поиск интересующего сообщения, перенаправление сообщения, повторное прочтение ранее полученного сообщения – все это относится к данной функции.

Кроме этих обязательных функций, большинство почтовых систем имеют ряд более сложных функций. Например, если пользователь уехал, он может перенаправить сообщения, поступающие в его отсутствие, куда-либо еще. Во многих системах пользователь может создавать так называемые 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.

Важной функцией является сообщение с уведомлением. В любом случае крайне полезно для пользователя получать сообщения о состоянии его сообщения. Другой пример – копия отправленных сообщений, приоритетность сообщений, секретность и т.д.

Ключевым моментом всех современных почтовых систем является разделение почтового отправления на конверт сообщения и собственно сообщение. Системой доставки используется конверт. Он содержит всю необходимую 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. Тело предназначено исключительно для пользователя.

Агент пользователя – обычно программа, имеющая множество команд для получения, составления сообщения и ответа на сообщение, а также для работы с почтовым ящиком. Некоторые агенты используют командную строку, некоторые - графический интерфейс.

Чтобы послать сообщение, пользователь должен предоставить адрес назначения, само сообщение и другие параметры, например, приоритетность, секретность и т.п. Для создания сообщения может быть использован любой текстовый процессор либо текстовый редактор, встроенный в агент пользователя. Все параметры должны быть заданы в формате, который понимает и с которым может работать агент пользователя. Большинство агентов пользователя ожидает адрес назначения в формате DNS: mailbox@location.

Следует отметить, что существуют и другие форматы, например, X.400. Этот формат предполагает довольно сложное описание адреса назначения. В частности, сообщение с адресом может быть доставлено, даже если адрес назначения выписан не полностью. Однако агенты пользователя, поддерживающие такой формат, стали доступны не скоро, как результат – такой формат адреса не прижился.

Агент пользователя также поддерживает лист рассылки. Этот лист позволяет рассылать одно и то же сообщение сразу нескольким пользователям. Причем сообщение размножается не обязательно самим агентом, а там, где поддерживается лист рассылки.

Прежде чем агент пользователя (далее АП) что-либо высветит на экране при загрузке, он просмотрит почтовый ящик на предмет того, что нового туда поступило. Затем он высветит на экране его содержимое с краткой аннотацией каждого сообщения. В простых почтовых АП высвечиваемые поля встроены в АП, в развитых – пользователь сам определяет, что показывать, а что нет. (Эта информация содержится в файле «user profile».)

После того как содержимое ящика показано пользователю, последний может выполнять любую из имеющихся команд. Типичный состав команд показан в таблице 7-50.

Таблица 7-50. Типичный состав команд почтового ящика

Команда	Параметр	Описание
h	№	Отобразить заголовки на экране
c		Отобразить только текущий заголовок
t	№	Вывести сообщение на экран
s	address	Отправить сообщение
f	№	Переслать сообщение
a	№	Ответить на сообщение
d	№	Удалить сообщение
u	№	Восстановить ранее удаленное сообщение
m	№	Переместить сообщение в другой почтовый ящик
k	№	Оставить сообщение после выхода
r	mailbox	Просмотреть другой почтовый ящик
n		Перейти к следующему сообщению
b		Вернуться к предыдущему сообщению
g	№	Перейти к конкретному сообщению, но не отображать его
e		Выйти и обновить почтовый ящик

Рассмотрим формат самого сообщения. 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 различал четко заголовок и конверт. В современных почтовых системах это различие проведено лучше, и агент пользователя имеет дело с заголовком, а агент передачи – с конвертом, который он формирует на основе заголовка.

Важные поля заголовка перечислены в таблице 7-51.

Таблица 7-51. Поля заголовка RFC 822

Заголовок	Значение
Поля, относящиеся к транспортировке сообщения	
To:	Адрес основного получателя
Cc:	Адрес дополнительного получателя
Bcc:	Адрес для «слепых» копий
From:	Создатель данного сообщения
Sender:	Адрес отправителя
Received:	Строка, добавляемая каждым агентом на пути сообщения

Return-Path:	Может использоваться для определения обратного пути к отправителю
Некоторые поля, используемые в заголовке сообщения	
Date:	Дата и время отправки сообщения
Reply-To:	Адрес, на который нужно отправлять ответ
Message-Id:	Уникальный номер сообщения для использования в дальнейшем
In-Reply-To:	Идентификатор сообщения, ответом на которое является текущее сообщение
References:	Другие релевантные идентификаторы сообщения
Keywords:	Ключевые слова, выбранные пользователем
Subject:	Краткий отрывок сообщения для отображения одной строкой

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 подходило вполне. В наши дни такой подход уже не годится. Необходимо, чтобы почтовая система умела работать:

1. с сообщениями на европейских языках (французском, немецком и т.д.)
2. с сообщениями не в латинском алфавите (русском, арабском и т.д.)
3. с сообщениями не в алфавите (японский, китайский)
4. с сообщениями, не содержащими текст (звук, видео, графика)

Такое решение предложено в RFC 1341 и RFC 1521. Это решение называется MIME – многоцелевое расширение почтовой службы в Internet. Основная идея MIME – расширение RFC 822 в целях введения структуры в тело сообщения и правил кодировки ASCII-сообщений. Естественно, что это повлияло на программы доставки и отправки сообщений.

MIME определяет пять новых заголовков, показанных в таблице 7-52. Первый сообщает агенту пользователя, что он имеет дело с MIME-сообщением, и то, какая версия MIME используется. Второй и третий характеризуют сообщение. Например, второе поле можно использовать для фильтрации сообщений.

Таблица 7-52. Пять заголовков RCF 822, определенных в MIME

Заголовок	Значение
MIME-Version:	Идентифицирует версию MIME
Content-Description:	Строка, описывающая содержимое сообщения
Content-Id:	Уникальный идентификатор
Content-Transfer-Encoding:	Способ подготовки тела письма к передаче
Content-Type:	Тип сообщения

Заголовок Content_Transfer_Encoding определяет, как сообщение должно быть подготовлено для передачи через сеть. Для этого используется пять основных схем. Простейшая схема – для передачи ASCII-текста – 7 бит на символ. Однако для учета национальных алфавитов используется 8 битная схема. Однако ни первая, ни вторая схемы не должны превышать 1000 символов в строке.

Сложная ситуация с передачей двоичных данных. Для корректной передачи такого типа данных (исполняемый код программ) используется схема base64 encoding. Эта схема разбивает сообщение на

блоки по 24 бита. Каждый блок разбивается на 4 группы по 6 бит каждая. Для сообщений, которые являются «почти» ASCII-сообщениями с небольшими исключениями, используется другая схема – quoted-printable encoding. Можно указать и какую-то особую схему в поле Content-Transfer-Encoding.

Последнее поле заголовка указывает тип сообщения. Возможные значения этого поля указаны в таблице 7-53.

Таблица 7-53. Типы и подтипы сообщений MIME

Тип	Подтип	Описание
Text	Plain	Неформатированный текст
	Richtext	Текст с простым форматированием
Image	Gif	Изображение в формате GIF
	Jpeg	Изображение в формате JPEG
Audio	Basic	Звук
Video	Mpeg	Фильм в формате MPEG
Application	Octet-stream	Неинтерпретируемая последовательность байтов
	Postscript	Готовый к печати документ в формате PostScript
Message	Rfc822	Сообщение MIME RFC 822
	Partial	Сообщение перед передачей было разбито на несколько частей
	External-body	Непосредственно сообщение следует передать через сеть
Multipart	Mixed	Независимые части в определенной последовательности
	Alternative	Вышеупомянутое сообщение в различных форматах
	Parallel	Части сообщения необходимо просматривать вместе
	Digest	Каждая часть является самостоятельным сообщением RFC 822

Основная задача системы передачи почтовых сообщений – надежно доставить сообщение от отправителя к получателю. Самый простой способ сделать это – установить транспортное соединение и по нему передавать почту.

SMTP (Simple Mail Transfer Protocol) – Простой протокол передачи почты. В Internet почта передается следующим образом. Машина отправитель устанавливает TCP-соединение с 25-м портом машины получателя. На 25 порту находится почтовый демон, который работает по протоколу SMTP. Он принимает соединение и распределяет поступающие сообщения по почтовым ящикам машины-получателя.

SMTP - это простой ASCII-протокол. После установления соединения машина-отправитель работает как клиент, а машина-получатель – как сервер. Сервер шлет текстовую строку, идентифицирующую его и готовность принимать почту. Если он не готов принимать почту, то клиент разрывает соединение и повторяет всю процедуру позже.

Если сервер подтвердил свою готовность, то клиент сообщает, от кого и кому предназначено очередное сообщение. Если сервер подтвердил наличие получателя, то он дает команду клиенту, и сообщение передается без контрольных сумм и подтверждений, так как TCP-соединение обеспечивает надежный поток байтов. Если сообщений несколько, то все они передаются. Обмен по соединению происходит в обоих направлениях.

Замечание. Клиент всегда посылает 4-символьные команды. Команды сервера в основном цифровые.

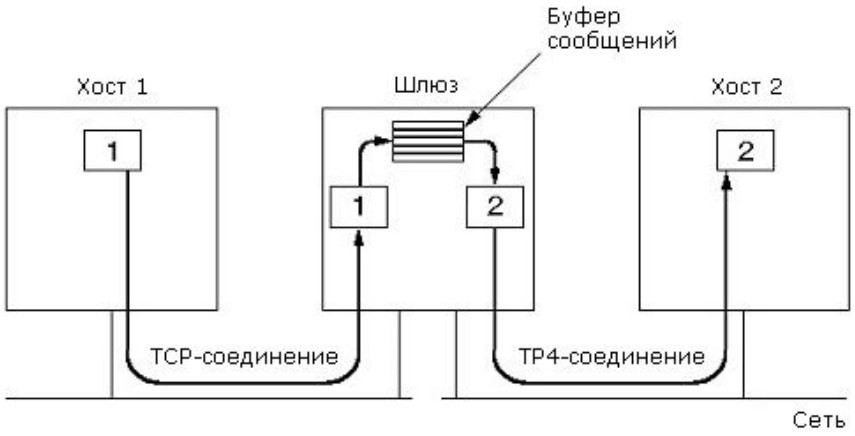
Получателей может быть несколько, тогда используют несколько RCTP-команд.

У SMTP протокола, несмотря на то что он хорошо описан в RFC 821, есть несколько проблем. Первая – длина сообщения не может превосходить 64 Кбайт. Другая – time out. Если время ожидания подтверждения у отправителя и получателя не согласовано, то один будет разрывать соединение, не дождавшись, тогда как другой просто очень загружен. Третья – почтовый ураган. Пусть машина-получатель имеет лист рассылки, где указана машина-отправитель, и наоборот. Тогда отправка сообщения по листу рассылки вызовет бесконечно долгие обмены сообщениями между этими машинами.

Для преодоления этих проблем в RFC 1425 был описан протокол ESMTP. Клиент вначале шлет команду EHLO, если она отвергается сервером, то это означает, что сервер работает по SMTP.

Протокол SMTP хорош, когда обе машины находятся в Internet. Однако это не всегда так. Многие компании в целях сетевой защиты соединяют свои сети через надлежащие средства, либо используют другие протоколы. Например, отправитель или получатель могут использовать протокол X.400. Такая ситуация показана на рисунке 7-55. Отправитель передает сообщение шлюзу, тот его буферизует и позднее передает получателю. Звучит просто, но на деле все сложнее.

Рисунок 7-55. Передача электронного письма с использованием почтового шлюза на прикладном уровне



Первая проблема – соответствие адресов. Вторая – соответствие конвертов и заголовков. Третья – соответствие тела сообщения. Например, в случае, если тело содержит аудиофайл, а на стороне получателя с ним работать не умеют. Или отправитель поставил следующее условие: если передача сообщения не пройдет по почтовому соединению, то нужно повторить его по факсу, а получатель не умеет работать с факсом. Однако для простых неструктурированных ASCII-сообщений SMTP-шлюз – решение проблемы.

До сих пор мы предполагали, что машина пользователя может и отправлять сообщения, и получать их. Однако часто машина пользователя – это персональный компьютер или ноутбук, которая время от времени связывается с почтовым сервером, чтобы отправить или получить почту.

Как это происходит? Простой протокол для изъятия почты из удаленного почтового ящика – POP3 (Post Office Protocol – RFC 1225). Он позволяет входить в удаленную систему и выходить из нее, передавать письма и принимать их. Главное - он позволяет забирать почту с сервера и хранить ее на машине пользователя.

Более сложный протокол IMAP – Interactive Mail Access Protocol (RFC 1064). Он позволяет одному и тому же пользователю заходить с разных машин на сервер, чтобы прочесть или отправить

почту. Это по существу удаленное хранилище писем. Он, например, позволяет получать доступ к письму не только по его номеру, но и по содержанию.

Третий часто используемый протокол – DMSP (Distributed Mail System Protocol RFC 1056). Этот протокол не предполагает, что пользователь работает все время с одной и той же почтовой службой. Пользователь может обратиться к серверу и забрать почту на свою локальную машину, после чего разорвать соединение. Обработав почту, он может ее отправить позже, когда будет установлено очередное соединение.

Важными почтовыми сервисами являются:

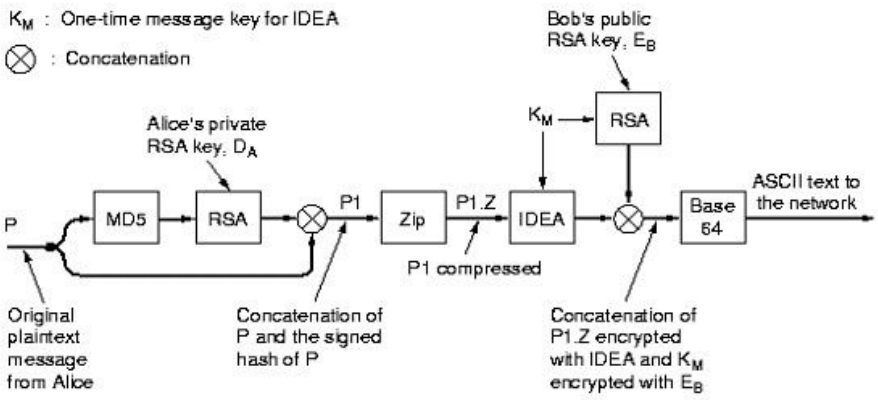
- Фильтры
- Пересылка поступающей почты на другие адреса
- Демон отсутствия
- Почтовый робот (программа, анализирующая входящие письма и отвечающая на них)

Пославший почту, естественно, предполагает, что ее никто не читает кроме адресата. Однако если об этом специально не позаботиться, то гарантировать этого нельзя. Далее мы рассмотрим две широко распространенных безопасных почтовых системы - PGP и PEM.

PGP – Pretty Good Privacy. PGP – дословно: «вполне хорошая конфиденциальность» – разработка одного человека - Фила Зиммермана (Phil Zimmermann, 1995). Это полный пакет безопасности, который включает средства конфиденциальности, установления подлинности, электронной подписи, сжатия, и все это в удобной для использования форме. Благодаря тому, что это разработка далекого от государственных структур человека, качественная, работает как на платформе Unix, так и MS-DOS/Windows, Macintosh и распространяется бесплатно, она получила очень широкое распространение.

PGP использует алгоритмы шифрования RSA, IDEA и MD5. PGP поддерживает компрессию передаваемых данных их секретность, электронную подпись и средства управления доступом к ключам. Схема работы PGP показана на рисунке 7-56. На этом рисунке DA, DB - личные (закрытые) ключи A и B соответственно, а EA, EB – их открытые ключи. Отметим, что секретный ключ для IDEA строится автоматически в ходе работы PGP на стороне A и называется ключом сессии - KM, который затем шифруется алгоритмом RSA с открытым ключом пользователя B. Также следует обратить внимание на то, что медленный алгоритм RSA используется для шифрования коротких фрагментов текста: 128-разрядного ключа для MD5 и 128-разрядного ключа для IDEA.

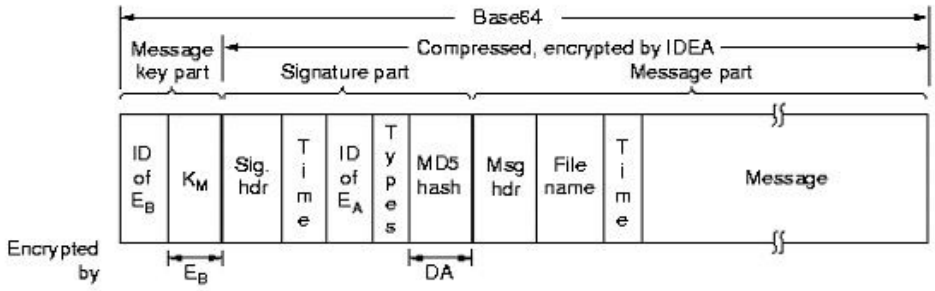
Рисунок 7-56. Действия PGP при отправке сообщения



PGP поддерживает три длины ключей:

- Обычный – 314 бит (может быть раскрыт за счет больших затрат).
- Коммерческий – 512 бит (может быть раскрыт специализированными организациями, названия которых, как правило, состоят из трех букв, например, ЦРУ, АНБ, ФСБ, МВД, ФБР).
- Военный – 1024 бита (не может быть раскрыт пока никем на Земле).

Рисунок 7-57. Формат PGP-сообщения



PEM – почтовая служба с повышенной конфиденциальностью. PEM имеет статус Internet-стандарта (RFC 1421, 1424). Сообщения, пересылаемые с помощью PEM, сначала преобразуются в каноническую форму. В этой форме соблюдены соглашения относительно спецсимволов типа табуляции, последовательных пробелов и т.п. Затем сообщение обрабатывается MD5 или MD2, шифруется с помощью DES (56-разрядный ключ) и передается с помощью кодировки base64. Передаваемый ключ защищается либо с помощью RSA, либо с помощью DES по схеме EDE.

Протокол передачи файлов – FTP.

FTP (File Transfer Protocol, Протокол передачи данных) - это один из первых и все еще широко используемых интернет-сервисов. Первые спецификации FTP относятся к 1971 году. С тех пор FTP претерпел множество модификаций и значительно расширил свои возможности.

Протокол FTP предназначен для решения следующих задач:

- разделение доступа к файлам на удаленных абонентских машинах
- прямое или косвенное использования ресурсов удаленных компьютеров
- обеспечение независимости клиента от файловых систем удаленных абонентских машин
- эффективная и надежная передача данных

FTP - это протокол прикладного уровня, который, как правило, использует в качестве транспортного протокола TCP. FTP не может использоваться для передачи конфиденциальных данных, поскольку не обеспечивает защиты передаваемой информации и передает между сервером и клиентом открытый текст. FTP-сервер может потребовать от FTP-клиента аутентификации (т.е. при подсоединении к серверу FTP-пользователь должен будет ввести свой идентификатор и пароль). Однако и пароль, и идентификатор пользователь будут переданы от клиента на сервер открытым текстом.

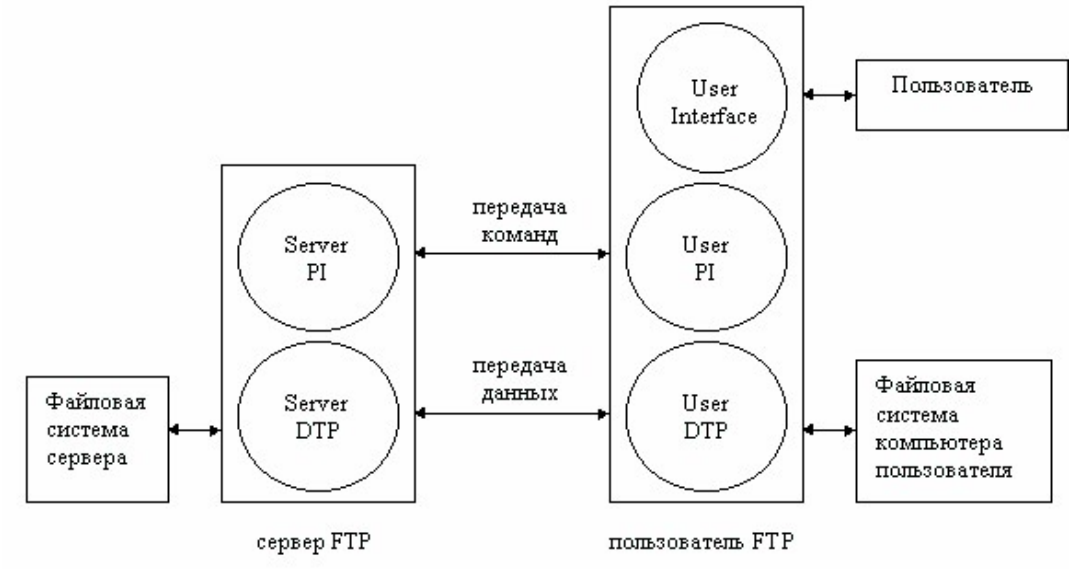
Простейшая модель работы протокола FTP представлена на рисунке 7-59, где введены следующие обозначения:

- «User Interface» - пользовательский интерфейс работы с FTP.
- «User-PI» - интерпретатор команд пользователя (User Protocol Interpretator). Этот объект взаимодействует с «Server-PI», чтобы обмениваться командами управления передачей данных по каналу передачи команд и с «User-DTP» - модулем, который осуществляет непосредственную передачу данных по каналу передачи данных.

- «User-DTP» - модуль, осуществляющий обмен данными (User Data Transfer Process) между клиентом и сервером FTP по каналу передачи данных на основании команд модуля «User-PI». Этот объект взаимодействует с файловой системой пользователя и объектом «Server-DTP».
- «Server-PI» - модуль управления обменом данных со стороны сервера (Server Protocol Interpretator) по каналу передачи команд.
- «Server-DTP» - модуль, осуществляющий обмен данными со стороны сервера (Server Data Transfer Process) по каналу передачи данных
- «Сервер FTP» - модуль, осуществляющий работу FTP-сервера. Он состоит из модуля управления передачей - «Server-PI» и модуля, осуществляющего передачу, - «Server-DTP».
- «Пользователь FTP» - модуль клиента FTP. Он состоит из модуля управления передачей - «User-PI» - и модуля, осуществляющего передачу - «User-DTP».

FTP поддерживает сразу два канала соединения - канал передачи команд (и статусов их обработки) и канал передачи данных. Канал передачи данных может использоваться для передачи как в одном, так и в другом направлении, кроме того, он может закрываться и открываться по командам управляющих модулей в процессе работы. Канал передачи команд открывается с установлением соединения и используется только для передачи команд и ответов их обработки.

Рисунок 7-59. Модель работы протокола FTP



Алгоритм работы протокола FTP состоит в следующем:

1. Сервер FTP использует в качестве управляющего соединение на TCP порт 21, который всегда находится в состоянии ожидания соединения со стороны FTP-клиента.

2. После того как устанавливается управляющее соединение модуля «User-PI» с модулем сервера - «Server-PI», клиент может отправлять на сервер команды. FTP-команды определяют параметры соединения передачи: роли участников соединения (активный или пассивный), порт соединения (как для «User-DTP», так и для «Server-DTP»), тип передачи, тип передаваемых данных, структуру данных и управляющие директивы, обозначающие действия, которые пользователь хочет совершить, например, сохранить, считать, добавить или удалить данные или файл.

3. После того как согласованы все параметры канала передачи данных, один из участников соединения, который является пассивным (например, клиентский модуль «User-DTP»), становится в режим ожидания открытия соединения на заданный для передачи данных порт. После этого активный модуль (например, «Server-DTP») открывает соединение и начинает передачу данных.

4. После окончания передачи данных соединение между «Server-DTP» и «User-DTP» закрывается, но управляющее соединение «Server-PI» - «User-PI» остается открытым. Пользователь, не закрывая сессии FTP, может еще раз открыть канал передачи данных, передать необходимую информацию и т.д.

FTP может использоваться не только при передаче файлов между клиентом и сервером, но и между двумя FTP-серверами, ни один из которых не расположен на локальном хосте пользователя.

Для этого пользователь сначала устанавливает управляющие соединения с двумя FTP-серверами, а затем устанавливает между ними канал передачи данных. В этом случае управляющая информация передается через модуль «User-PI», но данные транслируются через канал «Server1-DTP» - «Server2-DTP».

Алгоритм работы FTP-схемы, изображенной на рисунке 7-100, выглядит следующим образом:

1. Пользователь «User-PI» указал серверу «Server1-PI» работать в пассивном режиме, после чего сервер «Server1-PI» отправил пользователю «User-PI» адрес и номер порта (N), который он будет слушать.

2. «User-PI» назначил сервер «Server2-PI» активным участником соединения и указал ему передавать данные на хост «Server1-PI» на порт (N).

3. «User-PI» подал серверу «Server1-PI» команду «сохранить поступившие данные в таком-то файле», а серверу «Server2-PI» - «передать содержание такого-то файла».

4. Между серверами «Server1-DTP» и «Server2-DTP» образуется поток данных, который управляется клиентским хостом.

Основу передачи данных FTP составляет механизм установления соединения между соответствующими портами и механизм выбора параметров передачи. Каждый участник FTP-соединения должен поддерживать 21 порт передачи данных по умолчанию. По умолчанию «User-DTP» использует тот же порт, что и для передачи команд (обозначим его «U»), а «Server-DTP» использует порт номер (L-1), где L-управляющий порт. Однако, как правило, участниками соединения используются порты передачи данных, выбранные для них «User-PI», поскольку из управляющих процессов, участвующих в соединении, только он может изменить порты передачи данных как у «User-DTP», так и у «Server-DTP».

Пассивная сторона соединения должна до того, как будет подана команда начать передачу, слушать свой порт передачи данных. Активная сторона, подающая команду к началу передачи, определяет направление перемещения данных.

После того как соединение установлено, между «Server-DTP» и «User-DTP» начинается передача. Одновременно по каналу «Server-PI» - «User-PI» передаются уведомления о получении данных. Протокол FTP требует, чтобы управляющее соединение было открыто, пока по каналу обмена данными идет передача. Сессия FTP считается закрытой только после закрытия управляющего соединения.

Как правило, сервер FTP ответственен за открытие и закрытие канала передачи данных. Сервер FTP должен самостоятельно закрыть канал передачи данных в следующих случаях:

1. Сервер закончил передачу данных в формате, который требует закрытия соединения.

2. Сервер получил от пользователя команду прервать соединение.

3. Пользователь изменил параметры порта передачи данных.

4. Было закрыто управляющее соединение.

5. Возникли ошибки, при которых невозможно возобновить передачу данных.

FTP-протокол имеет двух «младших братьев»: TFTP - Trivial FTP и SFTP - Simple FTP.

TFTP-протокол - это простейший протокол передачи файлов. Он работает поверх транспортного протокола UDP и обеспечивает выполнение только самых элементарных операций передачи файлов, а именно, записи и чтения файлов. TFTP был разработан как простой и легкий в применении протокол. Он не позволяет вызывать список каталога и не имеет никаких средств аутентификации, но может передавать 8-битную информацию в соответствии со всеми интернет-стандартами.

Поскольку передача данных осуществляется поверх UDP, протокол TFTP реализует собственные методы надежной доставки данных - пакеты подтверждения, нумерация блоков данных и пакетов подтверждения и т.п. Все это очень похоже на упрощенный вариант эмуляции протокола TCP.

TFTP работает лишь с пятью командами:

1. Read request (RRQ) - запрос на чтение

2. Write request (WRQ) - запрос на запись

3. Data (DATA) - пакет данных

4. Acknowledgment (ACK) - подтверждение

5. Error (ERROR) - ошибка

Процесс передачи данных начинается с поступления от клиента TFTP на сервер запроса на чтение или запись файла. Соединение устанавливается после получения подтверждения готовности на один из запросов, либо на запись - WRQ, либо на чтение - RRQ.

При открытии соединения каждая из сторон выбирает (случайным образом) уникальный идентификатор - TID, который используется в UDP как порт соединения. Каждый пересылаемый пакет ассоциирован с двумя TID, соответствующими каждой стороне соединения. Первоначальный запрос отправляется инициатором TFTP-соединения на UDP-порт (порт инициализации), в котором указывается порт соединения. Дальнейший обмен уже происходит через порты, выбранные участниками передачи данных.

Если сервер разрешает запрос, соединение открывается и указанный файл передается (блоками по 512 байт). Каждый пакет передаваемых данных содержит один блок (512 байт) передаваемых данных и номер блока в передаваемом потоке. Поступление каждого блока на хост назначения должно быть подтверждено пакетом АСК (подтверждение) с номером поступившего блока. Только после получения пакета подтверждения будет отправлен следующий пакет данных.

Если длина пакета менее 512 байт, это служит сигналом для закрытия канала связи. В случае потери пакета при передаче через некоторый промежуток времени сервер отправит этот пакет данных повторно.

Три типа ситуаций порождают отправку ошибочных пакетов:

1. Не подтвержден запрос, например, не был найден файл, нет прав доступа, и др.

2. Неправильный формат пакета, например, не был найден файл, нет прав доступа, и др.

3. Потерян доступ к требуемому ресурсу.

При большом количестве сообщений об ошибках соединение может быть закрыто по инициативе одной из сторон.

SFTP-протокол передачи файлов пользуется популярностью в тех случаях, если пользователю необходим чуть более гибкий и надежный протокол, чем TFTP, и не такой сложный и громоздкий, как FTP.

SFTP поддерживает механизмы идентификации пользователя, передачу файлов, просмотр каталогов, изменение текущего каталога, переименование и удаление файлов. В большинстве операций, которые пользователь проводит с удаленным FTP-сервером, этого сервиса вполне достаточно. SFTP может передавать 8-битный поток данных и использует, подобно TFTP, только один канал соединения - как для команд, так и для данных. В отличие от TFTP, SFTP работает поверх TCP, порт 115.

Команды SFTP отправляются поочередно, после получения ответа обработки предшествующей команды. Все команды состоят из четырех ASCII-символов и символа пробела, который отделяет команду от аргументов. Ответ сервера состоит из кода ответа и текстового сообщения.

Команды SFTP имеют почти тот же синтаксис и предназначение, что и аналогичные команды FTP.

Билет № 46.

Управление сетями. Протокол SNMP. Вопросы производительности сетей.

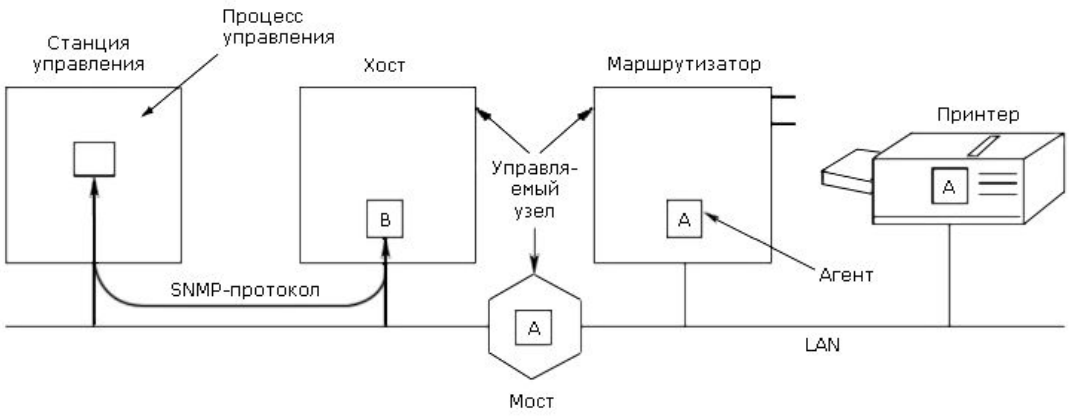
Ккогда Интернет стал охватывать большие территории, потребовались адекватные средства для управления сетью. Под словом «управление» в данном случае мы будем понимать возможность оперативно получать информацию о каждом устройстве в сети: о его работоспособности, состоянии, истории его функционирования. Позднее на понятии управления в сети мы остановимся подробнее. В 1990 году была опубликована первая версия протокола Simple Network Management Protocol (SNMP v.1). Эта версия описана в RFC 1155 и RFC 1157. В этих документах было описано систематическое наблюдение за сетью (какая информация могла накапливаться, как и где) и управление ею (как и какие параметры работы устройств сети можно было менять). Этот протокол получил широкое распространение и был реализован практически во всех устройствах, используемых в сетях.

Опыт использования протокола SNMP v.1 выявил ряд недостатков. Например, в первой версии недостаточно были проработаны вопросы безопасности. При опросе устройств явно указывался пароль, глядя на который, устройство аутентифицировало запрашивающее устройство. Во второй версии протокола SNMP v.2 (RFC 1441-1452) была введена криптографическая защита механизма аутентификации. Далее будет кратко описана именно вторая версия протокола SNMP.

Предполагается, что сеть состоит из четырех категорий компонентов:

- Станции управления
- Управляемые устройства
- Управляющая информация
- Протокол управления

Рисунок 7-29. Устройство модели управления SNMP



Модель управления, принятая в протоколе SNMP, показана на рисунке 7-29. Управление сетью осуществляют станции управления. Это компьютеры, на которых выполняются процессы, собирающие и накапливающие информацию об управляемых устройствах в сети. Сбор этой информации происходит по запросу от управляющей станции управляемому устройству. Запросы, передача и другие действия выполняются с помощью команд протокола SNMP.

На управляемых устройствах работают специальные SNMP-агенты (далее для краткости просто «агенты»), которые выполняют команды, передаваемые с помощью SNMP-протокола, фиксируют определенный набор параметров функционирования управляемого устройства. Управляемым устройством может быть маршрутизатор, мост, рабочая станция, устройство печати – любое устройство, где может работать SNMP-агент. Каждый агент поддерживает локальную базу данных MIB (Management Information Base), где хранится информация о состоянии агента, история его функционирования и переменные, характеризующие работу устройства, где функционирует агент.

SMI – структура управляющей информации.

В сети используется аппаратура сотен различных производителей. Естественно агент должен формировать данные о функционировании управляемого устройства в некотором унифицированном виде, например, по составу, способу представления независимо от того, кто изготовил это устройство.

В соответствии с терминологией, принятой в стандарте протокола SNMP, переменную, в которой агент накапливает информацию, будем называть объектом. Все объекты собраны в группы, определяемые стандартом, а группы – в модули. Чтобы все объекты имели единые правила идентификации, поступают следующим образом. Строят дерево стандарта, в котором отражают иерархию понятий, используемых в стандарте. Это дерево стандарта является поддеревом дерева стандартов.

На первом ярусе этого дерева расположены названия организаций, имеющих право выпускать международные стандарты – это ISO и МКТТ (теперь МСТ – международный союз телекоммуникаций). Есть в этом дереве и место для Интернета – ярус 4. На последнем ярусе указаны группы. В скобках рядом с именем каждой группы указано количество объектов в группе. Объекты имеют тип, определяемый через Object Identifier. Все объекты в этом дереве могут быть заданы указанием пути в дереве.

SMI в определенном смысле представляет собой язык для определения структур данных, представляющих объекты в базе данных MIB. В таблице 7-31 указаны типы данных, используемые для определения объектов, отслеживаемых протоколом SNMP.

Таблица 7-31. Типы данных, используемых при мониторинге SNMP

Имя	Тип	Байт	Значение
INTEGER	Числовой	4	Целое число (32 бита в текущих реализациях)
Counter32	Числовой	4	Беззнаковый 32-битный счетчик с переносом на новую строку
Gauge32	Числовой	4	Беззнаковое значение без перехода на новую строку
Integer32	Числовой	4	32 бита, даже на 64-битном процессоре
UInteger32	Числовой	4	Как у Integer32, но беззнаковый
Counter64	Числовой	8	64-битный счетчик
TimeTicks	Числовой	4	В сотых долях секунды с определенного начала отсчета
BIT STRING	Строка	4	Битовое отображение от 1 до 32 бит
OCTET STRING	Строка	≥0	Строка бит переменной длины
Opaque	Строка	≥0	Устарел; используется только для совместимости с ранними версиями
OBJECT ID	Строка	>0	Список целых чисел
IpAddress	Строка	4	Десятичный интернет-адрес с разделительными точками
NsapAddress	Строка	<22	NSAP-адрес OSI

MIB - База управляющей информации.

Коллекцию объектов, которыми можно управлять с помощью протокола SNMP, определяет база MIB. Все объекты этой базы сгруппированы в 10 групп, которые соответствуют 10 узлам, смежных узлу mib-2 в дереве на рисунке 7-30. Эти группы показаны в таблице 7-32.

Таблица 7-32. Группы объектов базы MIB-II

Группа	Кол-во объектов	Описание
System	7	Название, местоположение и описание оборудования
Interfaces	23	Сетевые интерфейсы и их измеряемый трафик
AT	3	Трансляция адреса
IP	42	Статистика IP-пакета
ICMP	26	Статистика полученных ICMP-сообщений
TCP	19	Алгоритмы, параметры и статистика ICMP
UDP	6	Статистика трафика UDP
EGP	20	Статистика трафика протокола EGP
Transmission	0	Зарезервировано для обусловленных средой MIB
SNMP	29	Статистика трафика SNMP

Группа System состоит из семи простых переменных (таблиц в этой группе нет). В таблице 7-33 приведены их имена, типы данных и описания.

Таблица 7-33. Простые переменные группы System

Имя	Тип данных	Описание

sysDescr	DisplayString	Текстовое описание пункта.
sysObjectID	ObjectID	Идентификатор поставщика.
sysUpTime	TimeTicks	Время (в сотых долях секунды), которое прошло после последней перезагрузки системы.
sysContact	DisplayString	Имя человека и адрес фирмы-производителя, отвечающей за работу данного устройства.
sysName	DisplayString	Полное имя домена, где расположен узел (FQDN).
sysLocation	DisplayString	Физическое расположение узла.
sysServices	[0..127]	Значение, указывающее на то, какие сервисы предоставляются узлом.

Для группы Interface определена только одна простая переменная: количество интерфейсов в системе.

Таблица 7-34. Простая переменная в группе if

Имя	Тип данных	R/W	Описание
ifNumber	INTEGER		Количество сетевых интерфейсов в системе.

В этой группе также определена таблица, состоящая из 22 строк. Каждая строка в таблице определяет характеристики каждого интерфейса из числа, указанного в переменной ifNumber, как показано в таблице 7-35.

Таблица 7-35. Переменные в таблице интерфейсов: ifTable

Имя	Тип данных	R/W	Описание
ifIndex	INTEGER		Индекс интерфейса.
ifDescr	DisplayString		Текстовое описание интерфейса.
ifType	INTEGER		Тип, например: 6 = Ethernet, 7 = 802.3 Ethernet, 9 = 802.5 Token ring, 23 = PPP, 28 = SLIP и многие другие переменные.
ifMtu	INTEGER		MTU интерфейса (максимальный размер блока).
ifSpeed	Gauge		Скорость в битах в секунду.
ifPhysAddress	PhysAddress		Физический адрес или строка нулевой длины для интерфейсов без физического адреса (например, последовательные каналы).
ifAdminStatus	[1..3]	.	Желательное состояние интерфейса: 1 = активен, 2 = выключен, 3 = тестируется.
ifOperStatus	[1..3]		Текущее состояние интерфейса: 1 = активен, 2 = выключен, 3 = тестируется.

ifLastChange	TimeTicks		Значение sysUpTime на момент, когда интерфейс вошел в текущее состояние функционирования.
ifInOctets	Counter		Полное количество принятых байтов, включая служебные байты заголовков.
ifInUcastPkts	Counter		Количество входящих пакетов, адресованных индивидуально этому интерфейсу, и доставленных.
ifInNUcastPkts	Counter		Количество не персональных (широковещательных или групповых) и доставленных пакетов.
ifInDiscards	Counter		Количество принятых и сброшенных пакетов, в том числе если в пакете не была обнаружена ошибка (переполнение буферов).
ifInErrors	Counter		Количество пакетов, принятых и сброшенных по причине ошибок.
ifInUnknownProtos	Counter		Количество принятых и сброшенных пакетов по причине того, что они принадлежали неизвестному протоколу.
ifOutOctets	Counter		Количество переданных байт, включая заголовки.
ifOutUcastPkts	Counter		Количество пакетов, переданных конкретным интерфейсам.
ifOutNUcastPkts	Counter		Количество не персональных (широковещательных или групповых) переданных пакетов.
ifOutDiscards	Counter		Количество исходящих пакетов, которые были сброшены, в том числе если в пакетах не была обнаружена ошибка (переполнение буферов).
ifOutErrors	Counter		Количество исходящих пакетов, сброшенных по причине ошибок.
ifOutQLen	Gauge		Количество пакетов, находящихся в выходной очереди.
ifSpecific	ObjectID		Ссылка на определение MIB конкретно для этого типа среды передачи.

Для SLIP-канала тип интерфейса указывается как последовательное соединение точка-точка, а не SLIP. Также не сообщается скорость SLIP-канала.

Группа трансляции адресов поддерживается во всех системах, однако ее значимость значительно уменьшилась, после того как стала использоваться MIB-II. Теперь каждая группа сетевых протоколов (например, IP) содержит свою собственную таблицу трансляции адресов. Для IP это ipNetToMediaTable.

В группе At определена только одна таблица из трех строк, как показано в таблице 7-36.

Таблица 7-36. Таблица трансляции адресов: atTable

Имя	Тип данных	R/W	Описание
atIfIndex	INTEGER	·	Номер интерфейса: ifIndex.

atPhysAddress	PhysAddress	·	Физический адрес. Установка этого параметра в строку нулевой длины приводит к тому, что пункт считается некорректным.
atNetAddress	NetworkAddress	·	IP-адрес

Группа IP определяет большое количество переменных, собранных в три таблицы, где накапливается подробная статистика о IP-трафике, входящем и исходящем из узла. В таблице 7-37 приведены простые переменные.

Таблица 7-37. Простые переменные группы IP

Имя	Тип данных	R/W	Описание
ipForwarding	[1..2]	·	1 означает, что система перенаправляет IP-дейтаграммы, а 2 - не перенаправляет.
ipDefaultTTL	INTEGER	·	Значение TTL по умолчанию, когда его не предоставляет транспортный уровень.
ipInReceives	Counter		Полное количество IP-дейтаграмм, полученных со всех интерфейсов.
ipInHdrErrors	Counter		Количество IP-дейтаграмм, отброшенных из-за ошибок в заголовке (например, ошибка контрольной суммы, несовпадение номера версии, истечение TTL).
ipInAddrErrors	Counter		Количество IP-дейтаграмм, отброшенных из-за неправильного адреса назначения.
IpForwDatagrams	Counter		Количество IP-дейтаграмм, для которых была сделана попытка перенаправления.
ipInUnknownProtos	Counter		Количество локально адресованных IP-дейтаграмм с неверным полем протокола.
ipInDiscards	Counter		Количество принятых IP-дейтаграмм, отброшенных из-за недостаточного размера буфера.
ipInDelivers	Counter		Количество IP-дейтаграмм, доставленных соответствующему модулю протокола.
ipOutRequests	Counter		Полное количество IP-дейтаграмм, переданных на IP-уровень для передачи. Сюда не включены те, которые были посчитаны в ipForwDatagrams.
ipOutDiscards	Counter		Количество исходящих IP-дейтаграмм, которые были отброшены из-за отсутствия места в буфере.
ipOutNoRoutes	Counter		Количество IP-дейтаграмм, которые были отброшены из-за того, что не был найден маршрут.
ipReasmTimeout	INTEGER		Максимальное количество секунд, в течение которого принятые фрагменты ждали повторной сборки.
ipReasmReqds	Counter		Количество принятых IP-фрагментов, которые должны быть собраны.
ipReasmOKs	Counter		Количество успешно собранных IP-дейтаграмм.

ipReasmFails	Counter		Количество сбоев в алгоритме повторной сборки IP.
ipFragOKs	Counter		Количество успешно фрагментированных IP-дейтаграмм.
ipFragFails	Counter		Количество IP-дейтаграмм, которые необходимо фрагментировать, чего не было сделано из-за установленного флага «не фрагментировать».
ipFragCreates	Counter		Количество IP-фрагментов, которые были получены при фрагментации.
ipRoutingDiscards	Counter		Количество пунктов маршрутизации, которые были проигнорированы, даже если они существовали и были верны.

Первая таблица в группе IP - это таблица IP-адресов. Она содержит по одной строке для каждого IP-адреса в системе. Каждая строка содержит пять переменных (таблица 7-38).

Таблица 7-38. Таблица IP-адресов: ipAddrTable

Имя	Тип данных	R/W	Описание
ipAdEntAddr	IpAddress		IP-адрес
ipAdEntIfIndex	INTEGER		Соответствующий номер интерфейса: ifIndex.
ipAdEntNetMask	IpAddress		Маска подсети для этого IP-адреса.
ipAdEntBcastAddr	[0..1]		Значение младших битов в широковещательном IP-адресе. Обычно равно 1.
ipAdEntReasmMaxSize	[0..65535]		Размер максимальной принятой IP-дейтаграммы для этого интерфейса, которая может быть повторно собрана.

Следующая таблица (таблица 7-39), это таблица IP-маршрутизации. В качестве индекса для получения доступа к каждой строке таблицы используется IP-адрес назначения.

Таблица 7-39. Таблица IP-маршрутизации: ipRouteTable

Имя	Тип данных	Описание
ipRouteDest	IpAddress	IP-адрес назначения. Значение 0.0.0.0 указывает на пункт по умолчанию.
ipRouteIfIndex	INTEGER	Номер интерфейса: ifIndex.
ipRouteMetric1	INTEGER	Первичный показатель маршрута. Значение показателя зависит от протокола маршрутизации (ipRouteProto). Значение -1 означает, что маршрут не используется.
ipRouteMetric2	INTEGER	Альтернативный показатель маршрута.
ipRouteMetric3	INTEGER	Альтернативный показатель маршрута.
ipRouteMetric4	INTEGER	Альтернативный показатель маршрута.

ipRouteNextHop	IpAddress	IP-адрес маршрутизатора следующей пересылки.
ipRouteType	INTEGER	Тип маршрута: 1 - другой, 2 - недействующий маршрут, 3 - прямой, 4 - непрямой.
ipRouteProto	INTEGER	Протокол маршрутизации: 1 - другой, 4 - ICMP-перенаправление, 8 - RIP, 13 - OSPF, 14 - BGP и другие.
ipRouteAge	INTEGER	Количество секунд, которое прошло с того момента, когда маршрут был последний раз обновлен или определен как корректный.
ipRouteMask	IpAddress	Маска, которая должна быть добавлена по логическому «и» к IP-адресу назначения, перед тем как она будет сравнена с ipRouteDest.
ipRouteMetric5	INTEGER	Альтернативный показатель маршрута.
ipRouteInfo	ObjectID	Ссылка на конкретное определение MIB для этого протокола маршрутизации.

Последняя таблица для группы IP - это таблица трансляции адресов, приведенная в таблице 7-40. Как мы говорили раньше, группа At в настоящее время практически не используется (как устаревшая), и эта таблица заменяет ее.

Таблица 7-40. Таблица трансляции IP-адресов: ipNetToMediaTable

Имя	Тип данных	Описание
ipNetToMediaIfIndex	INTEGER	Соответствующий интерфейс: ifIndex.
ipNetToMediaPhysAddress	PhysAddress	Физический адрес.
ipNetToMediaNetAddress	IpAddress	IP-адрес.
ipNetToMediaType	[1..4]	Тип сопоставления: 1 - другой, 2 - неиспользуемый, 3 - динамический, 4 - статический.

Группа ICMP накапливает информацию о каждом виде сообщений протокола ICMP. Эта группа состоит из четырех общих счетчиков (общее количество входящих и исходящих ICMP-сообщений и количество входящих и исходящих ICMP-сообщений с ошибками) и 22 счетчиков для различных типов ICMP-сообщений: 11 счетчиков на входящие сообщения и 11 счетчиков на исходящие сообщения. Это показано в таблице 7-41.

Таблица 7-41. Простые переменные группы ICMP

Имя	Тип данных	Описание
icmpInMsgs	Counter	Полное количество принятых ICMP-сообщений.
icmpInErrors	Counter	Количество принятых ICMP-сообщений с ошибками (например, ошибочная контрольная сумма ICMP).

icmpInDestUnreachs	Counter	Количество принятых ICMP-сообщений о недоступности источника.
icmpInTimeExcds	Counter	Количество принятых ICMP-сообщений об истечении времени.
icmpInParmProbs	Counter	Количество принятых ICMP-сообщений о проблемах с параметром.
icmpInSrcQuenchs	Counter	Количество принятых ICMP-сообщений о подавлении источника.
icmpInRedirects	Counter	Количество принятых ICMP-сообщений о перенаправлении.
icmpInEchos	Counter	Количество принятых ICMP-сообщений с эхо запросом.
icmpInEchoReps	Counter	Количество принятых ICMP-сообщений с эхо откликом.
icmpInTimestamps	Counter	Количество принятых ICMP-сообщений с запросом временной марки.
icmpInTimestampReps	Counter	Количество принятых ICMP-сообщений с откликом временной марки.
icmpInAddrMasks	Counter	Количество принятых ICMP-сообщений с запросом маски адреса.
icmpInAddrMaskReps	Counter	Количество принятых ICMP-сообщений с откликом маски адреса.
icmpOutMsgs	Counter	Полное количество исходящих ICMP-сообщений.
icmpOutErrors	Counter	Количество ICMP-сообщений, которые не были отправлены из-за проблем внутри ICMP (переполнение буферов).
icmpOutDestUnreachs	Counter	Количество посланных ICMP-сообщений о недоступности пункта назначения.
icmpOutTimeExcds	Counter	Количество посланных ICMP-сообщений об истечении времени.
icmpOutParmProbs	Counter	Количество посланных ICMP-сообщений о проблемах с параметром.
icmpOutSrcQuenchs	Counter	Количество посланных ICMP-сообщений о подавлении источника.
icmpOutRedirects	Counter	Количество посланных ICMP-сообщений о перенаправлении.
icmpOutEchos	Counter	Количество посланных ICMP-сообщений с эхо-запросом.
icmpOutEchoReps	Counter	Количество посланных ICMP-сообщений с эхо-откликом.
icmpOutTimestamps	Counter	Количество посланных ICMP-сообщений с запросом временной марки.
icmpOutTimestampReps	Counter	Количество посланных ICMP-сообщений с откликом временной марки.
icmpOutAddrMasks	Counter	Количество посланных ICMP-сообщений с запросом маски адреса.
icmpOutAddrMaskReps	Counter	Количество посланных ICMP-сообщений с откликом маски адреса.

Группа TCP накапливает сведения о соединениях протокола TCP, количестве посланных и полученных сегментов, ведет статистику ошибок.

В таблице 7-42 описаны простые переменные группы TCP. Группа TCP имеет одну таблицу - таблицу TCP-соединений, показанную в таблице 7-43. Она содержит по одной строке для каждого соединения. Каждая строка содержит пять переменных: состояние соединения, локальный IP-адрес, локальный номер порта, удаленный IP-адрес и удаленный номер порта.

Таблица 7-42. Простые переменные в группе TCP

Имя	Тип данных	R/W	Описание
tcpRtoAlgorithm	INTEGER		Алгоритм, используемый для расчета величин тайм-аутов и повторных передач: 1 - нет, 2 - постоянный RTO, 3 - MIL-STD-1778, 4 - алгоритм Van Jacobson.
tcpRtoMin	INTEGER		Минимальное значение тайм-аута повторной передачи в миллисекундах.
tcpRtoMax	INTEGER		Максимальное значение тайм-аута повторной передачи в миллисекундах.
tcpMaxConn	INTEGER		Максимальное количество TCP-соединений. Значение -1 означает, что эта величина определяется динамически.
tcpActiveOpens	Counter		Количество переходов от состояния CLOSED к состоянию SYN_SENT.
tcpPassiveOpens	Counter		Количество переходов от состояния LISTEN к состоянию SYN_RCVD.
tcpAttemptFails	Counter		Количество переходов от состояния SYN_SENT или SYN_RCVD к состоянию CLOSED плюс количество переходов от состояния SYN_RCVD к состоянию LISTEN.
tcpEstabResets	Counter		Количество переходов от состояния ESTABLISHED или CLOSE_WAIT к состоянию CLOSED.
tcpCurrEstab	Gauge		Количество соединений, находящихся в настоящее время в состоянии ESTABLISHED или CLOSE_WAIT.
tcpInSegs	Counter		Полное количество принятых сегментов.
tcpOutSegs	Counter		Полное количество отправленных сегментов за исключением тех, которые содержали только повторно передаваемые байты.
tcpRetransSegs	Counter		Полное количество повторно переданных сегментов.
tcpInErrs	Counter		Полное количество сегментов, принятых с ошибками (например, неверная контрольная сумма).
tcpOutRsts	Counter		Полное количество сегментов, посланных с установленным флагом RST.

Таблица 7-43. Таблица TCP-соединений: tcpConnTable

Имя	Тип данных	R/W	Описание

tcpConnState	[1..12]	·	Состояние соединения: 1 - CLOSED, 2 - LISTEN, 3 - SYN_SENT, 4 - SYN_RCVD, 5 - ESTABLISHED, 6 - FIN_WAIT_1, 7 - FIN_WAIT_2, 8 - CLOSE_WAIT, 9 - LAST_ACK, 10 - CLOSING, 11 - TIME_WAIT, 12 - удаление TCB. Единственное значение, которое может установить менеджер, это значение 12 (немедленное прекращение соединения).
tcpConnLocalAddress	IpAddress		Локальный IP адрес. 0.0.0.0 указывает на то, что слушающий процесс готов принять соединение с любого интерфейса.
tcpConnLocalPort	[0..65535]		Локальный номер порта.
tcpConnRemAddress	IpAddress		Удаленный IP-адрес.
tcpConnRemPort	[0..65535]		Удаленный номер порта.

Группа UDP накапливает информацию о количестве отправленных и полученных дейтаграмм, сколько полученных дейтаграмм невозможно доставить из-за неверно заданного адреса или по какой-либо другой причине.

Группа EGP используется маршрутизаторами, использующими EGP-протокол. Эта грамма учитывает, сколько пакетов вышло во внешнюю сеть, сколько вошло, сколько было доставлено, а сколько отвергнуто.

Группа Transmission зарезервирована для MIB-баз, которые предназначены для физических сред, имеющих свою специфику, например, Ethernet.

Последняя группа предназначена для сбора данных о функционировании самого протокола SNMP.

Протокол SNMP.

SNMP-протокол определяет пять типов сообщений, которыми обмениваются станция управления и устройство.

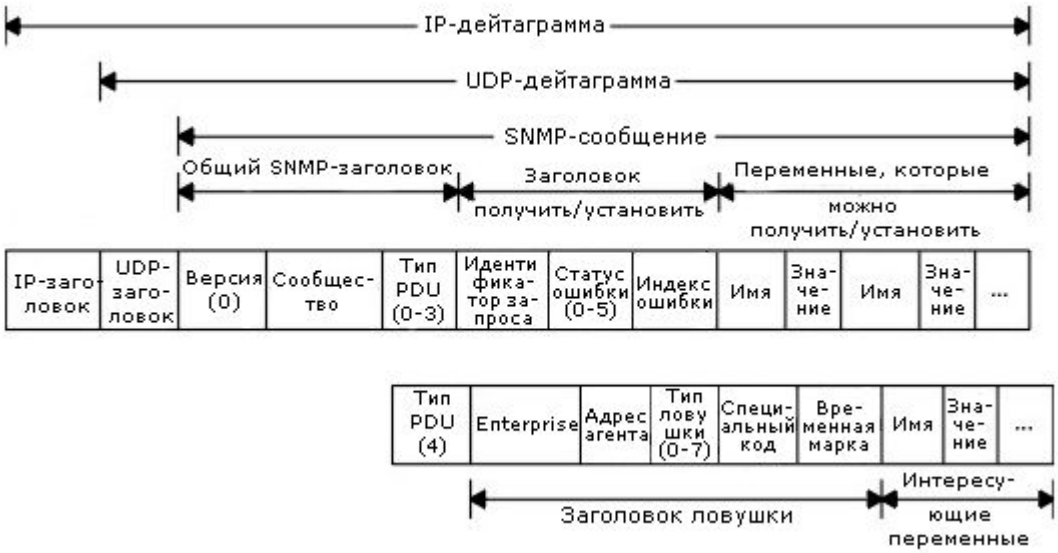
- get-request - получить значение одной или нескольких переменных.
- get-next-request - получить одну или несколько переменных, следующих после указанной переменной.
- set-request - установить значение одной или нескольких переменных.
- get-response - выдать значение одной или нескольких переменных. Это сообщение возвращается агентом станции управления в ответ на операторы get-request, get-next-request и set-request.
- оператор trap - уведомить станцию управления, когда что-либо произошло с агентом.

Первые три сообщения отправляются от станции управления к устройству, а последние два - от устройства к станции управления. Так как четыре из пяти SNMP-сообщений реализуются простой последовательностью «запрос-ответ», в SNMP-протоколе используют UDP-протокол. Это означает, что запрос от станции управления может не дойти до устройства, а отклик от устройства – до станции управления. В этом случае будет задействован механизм тайм-аута и повторная передача.

Станция управления отправляет все три запроса на UDP-порт 161. Устройство устанавливает ловушки (программные прерывания trap) на UDP-порт 162. Так как используются два разных порта,

одна и та же система может выступать и как станция управления, и как устройство. На рисунке 7-45 показан формат пяти SNMP-сообщений, инкапсулированных в UDP-дейтаграмму.

Рисунок 7-45. Формат пяти SNMP-сообщений



На этом рисунке мы указали в байтах только размер IP- и UDP-заголовков. Значение поля «версия» равно 0. Это значение в действительности равно номеру версии, уменьшенному на 1.

В таблице 7-46 показано значение поля «тип блока данных протокола» (PDU type).

Таблица 7-46. Типы PDU-сообщений SNMP

Тип PDU	Имя
0	get-request
1	get-next-request
2	set-request
3	get-response
4	trap

При взаимодействии между станцией управления и устройством используют пароль. Пароль - это 6-символьная строка, которую передавали в SNMP v.1 в открытом виде. В операторах get, get-next и set станция управления устанавливает идентификатор запроса (request ID), который возвращается устройством в сообщении get-response. Это повышает безопасность при взаимодействии. Это поле также позволяет станции управления выдать несколько запросов одному или нескольким устройствам, а затем отсортировать полученные отклики.

Статус ошибки (error status) - это целое число, которое возвращается агентам и указывает на ошибку. В таблице 7-47 показаны значения, имена и описания ошибок.

Таблица 7-47. Значения поля статуса ошибки SNMP

Статус ошибки	Имя	Описание

0	noError	Все в порядке.
1	tooBig	Устройство не может поместить отклик в одно SNMP-сообщение.
2	noSuchName	Запрос указывает на несуществующую переменную.
3	badValue	В запросе на установку использовано недопустимое значение или сделана ошибка в синтаксисе.
4	readOnly	Станция управления попыталась изменить переменную, которая помечена как «только для чтения».
5	genErr	Неопознанная ошибка.