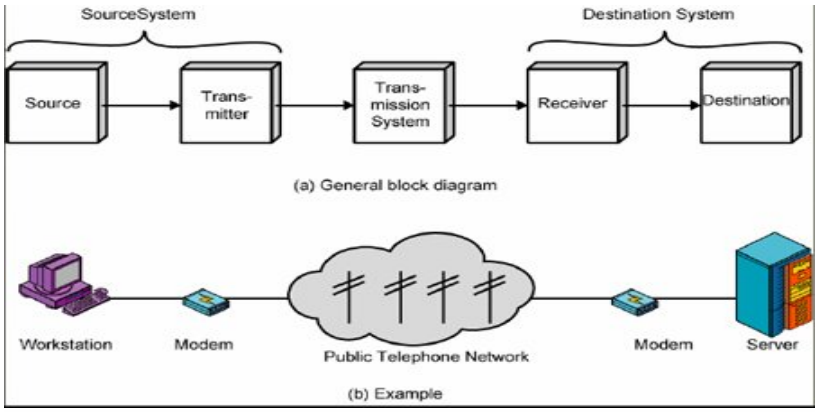
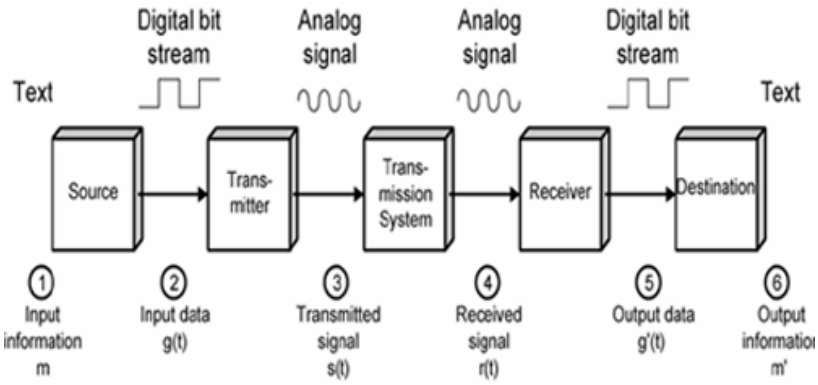
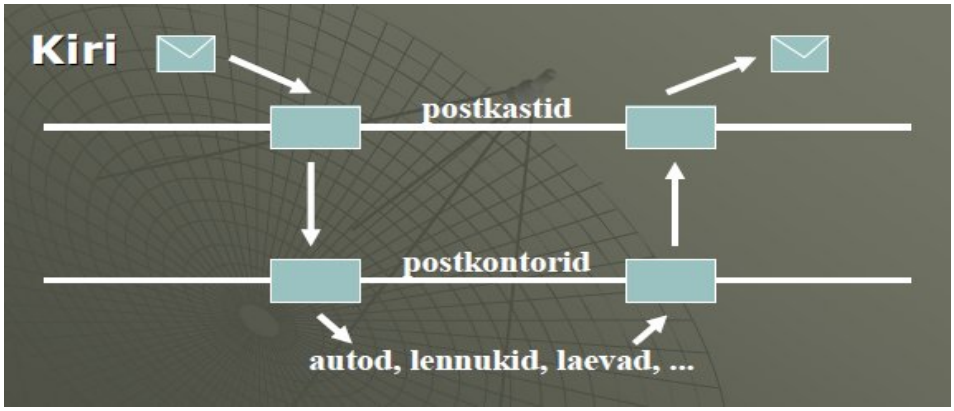
1. Üldine kommunikatsiooni mudel (Simplified communications model)




Главные элементы модели:
**Источник** -Объект, который генерирует данные -Человек, который говорит в телефон, или компьютер, посылая данные модему.
**Передатчик** - Устройство для преобразования/кодирования сигнала, произведенного источником - преобразованный сигнал посылается по системе передачи -Модем преобразовывает цифровые данные в аналоговый сигнал, который может быть обработан телефонной сетью Система передачи - Среда позволяющая транспорт сигнала от одного пункта до другого.
**Получатель** -Устройство, расшифровывающее полученный сигнал для того, чтобы обращаться устройством предназначения -Модемом преобразовывает полученные аналоговые данные назад в цифровой для использования компьютером
**Назначение** -Объект, который использует полученную информацию

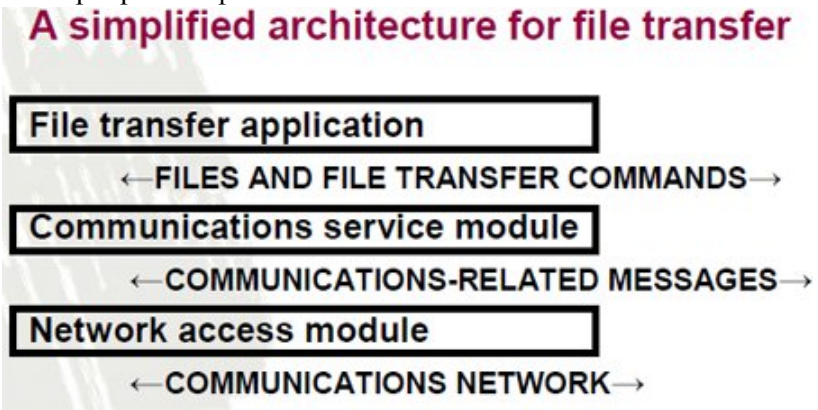
2. Kommunikatsioonisüsteemi ülesanded (Communications tasks).
Transmission system utilization – Транспортная система
Interfacing – Взаимодействие
Signal generation – Генерация сигнала
Synchronization – Синхронизация
Exchange management – Управление обменом
Error detection and correction – Обнаружение ошибок и коррекция (исправление)
Flow control – Управление потоком
Addressing – Адресация
Routing – Маршрутизация
Recovery – Восстановление
Message formatting – Форматирование сообщения
Security – Безопасность
Network management – Управление сетью
3. Mitmekiheline arhitektuur postisüsteemi näite basil




**Отправитель:** Письмо положили в конверт, запечатали, опустили в почтовый ящик.
Почта доставляется из почтового ящика в почтовое отделение. Письмо доставлено получателю почтовым отделением.
**Получатель:** Письмо достается из ящика, вытаскивается из конверта, читается.
Письмо доставлено перевозчиком в почтовое отделение. Письмо доставляется с почты в почтовый ящик.
**Мн**огослойная архитектура на примере почты заключается в том, что получатель \ отправитель видят только на уровне почтовый ящик. Почтовые конторы видят этот процесс, от почтового ящика с последующей сортировкой и отправкой до другого почтового отделения (где опять сортировка и т.д.) и уже доставка к получателю в почтовый ящик.

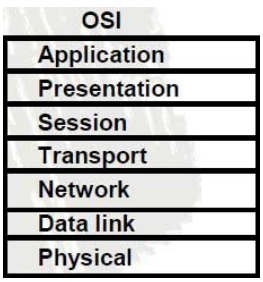
4. Kihid, teenused, protokollid ja andmete liikumine läbi kihtide.

**И**сточник должен либо активировать прямой путь передачи данных, либо сообщить сети идентификатор пункта назначения. Источник, перед отправкой должен убедиться, что получатель готов к приему данных. Приложение по отправки файлов, в системе отправителя, должно убедиться, что программа на машине получателя готова принять и сохранить файл для этого пользователя. Если формат файла, для двух машин является не совместимым, одна из сторон должна конвертировать файл.




**П**ротоколы:
Синтаксис – Заботиться о формате блоков данных.
Семантика – Включает в себя, управляющую информацию для координации и обработки данных.
Тайминг (timing) – Включает в себя, соответствие скорости и последовательности.

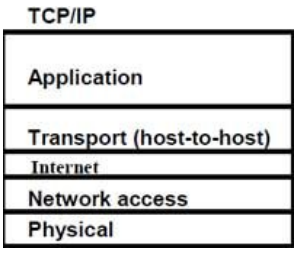
5. OSI mudel.




**A**pplication- Обеспечивает доступ для пользователя к среде OSI (синхронизация и кодирование информации)
**P**resentation – Обеспечивает независимость процессов программы (прочность передачи)
**S**ession – Структурный контроль, для общения между программами.
**T**ransport – Обеспечивает надежную, прозрачную, передачу данных между программами.
**N**etwork – Обеспечивает верхние слои независимостью от передачи данных и переключение технологий используемых для подключения систем. Отвечает за:

Создание, поддерживание и прекращение соединения.
**D**ata link – Обеспечивает надежную передачу информации через физическую связь (link). Посылает блоки с необходимой синхронизацией, контроль ошибок и управление потоком.
**P**hysical – Занимается передачей потока не структурированных битов через физическую среду. Речь идет о механических, электрических, функциональных и процедурных характеристик для доступа к физической среде.

6. TCP/IP mudel.



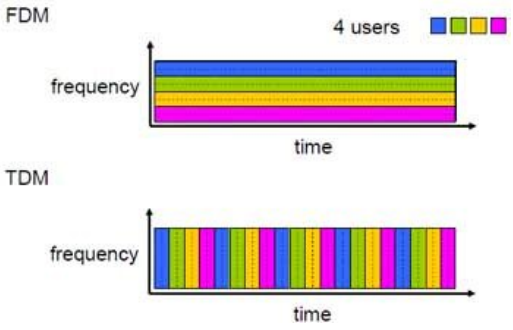

**A**pplication layer – Ответственен за предоставление услуг пользователям (Mail services, File transfer and access, remote log-in, access to the WWW)
**T**ransport layer – Ответственен за доставку сообщения от одного процесса к другому (адресация портов, разделение и слияние, контроль соединения, потока и ошибок)
**I**nternet layer –
**N**etwork access layer – Ответственен за доставку пакетов ( от отправителя до конечного получателя). Пример (маршрутизация).
**D**ata link – Ответственен за передачу кадров (frames) от одного узла к другому (физическая адресация, контроль потока, ошибок, доступа).
**P**hysical layer – Ответственен за передачу отдельных битов от одного узла к другому (синхронизация битов, скорость передачи данных)

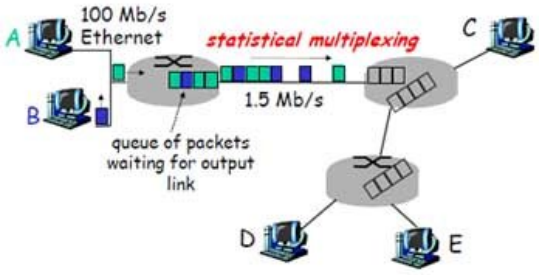
7. Ühendusele-orienteeritud ja ühenduseta andmeedastus. (Connection-oriented service and connectionless service).


В протоколах, **ориентированных на соединение**, станция А прежде чем осуществить обмен с В должна послать запрос на соединение (Connection REQUEST), в свободном домене канала управления станции В. Если станция В воспринимает запрос, обмен осуществляется через канал данных станции А. Отличительной особенностью службы с установлением логического соединения является то, что клиент и сервер перед передачей данных (например, сообщений электронной почты) сначала обмениваются специальными управляющими пакетами.
Эта процедура, иногда называемая рукопожатием, позволяет сторонам подготовиться к процессу основного обмена. Говорят, что по окончании процедуры рукопожатия соединение между оконечными системами является установленным. Основные цели: передача данных между конечными системами. 1. Handshaking (установить связь и приготовиться к передаче данных). 2.TCP, сервис, ориентированный на интернет соединение. TCP надежен для передачи, контроль потока, контроль перегруженности. Приложения: HTTP, FTP, SMTP.
Передача данных **без соединения**, не требует соединения между отправителем и получателем, передача между конечными системами. Как вы, вероятно, уже догадались, служба без установления логического соединения
не использует процедуру рукопожатия: вместо нее происходит простая передача пакетов. Это, с одной стороны, позволяет значительно сэкономить время при пересылке данных. С другой стороны, выигрыш во времени происходит за счет снижения надежности передачи: передающая сторона не имеет информации о том, была ли передача пакета успешной. Отправитель просто начинает отправлять пакеты получателю. Этот метод полезен для быстрых передач. Каждая система (получатель, отправитель) должны знать информацию друг о друге. UDP - Не надежная передача данных, отсутствует контроль потока(трафика), нет проверки на скопление. Приложения: Streaming media, DNS, Internet telephony.

8. Kanalikommutatsioon ja pakettkommutatsioon, paketti pikkus

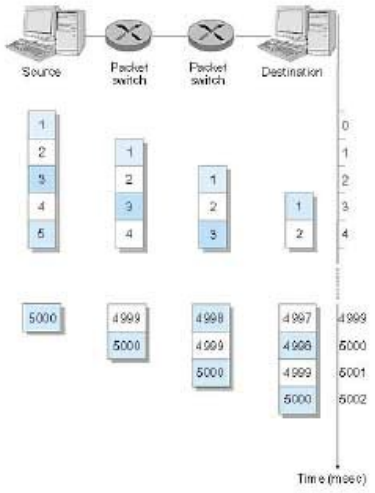
**Коммутация каналов:** При коммутации каналов происходит резервирование
на время сеанса связи необходимых ресурсов (буферов, диапазонов частот)
на всем сетевом пути. Разделяется на два вида FDM или TDM. TDM – Временное мультиплексирование, означает, что вся пропускная способность выходного канала предоставляется в течении фиксированных интервалов времени каждому входному каналу. Недостаток состоит в том, что даже если входной канал

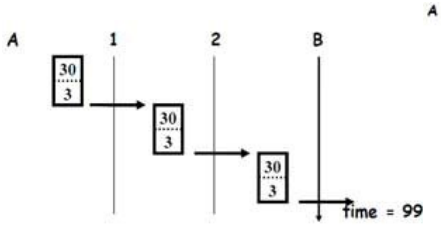


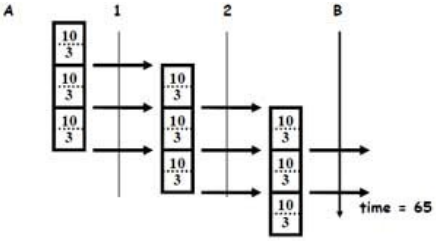



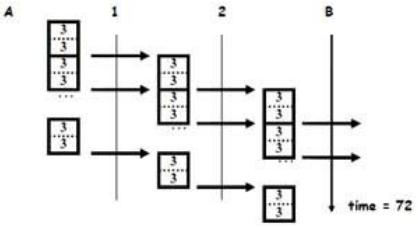

не используется для передачи, другие каналы не могут передавать данные в это время. FDM – Метод мультиплексирования, согласно которому отведенная полоса частот делится на несколько диапазонов, предоставляемых входным сигналом. Преимущество: Входные сигналы накладываются на разные частоты и поэтому в частотной области практически не пересекаются.

**Коммутация пакетов.** При коммутации пакетов ресурсы запрашиваются при необходимости и выделяются по требованию. Эта техника была специально разработана для эффективной передачи компьютерного трафика. Пакеты пользователей А и Б разделяют ресурсы сети. Каждый пакет использует полную ширину канала. Ресурсы используются по надобности. Общая сумма спроса на сеть может превышать имеющуюся. При перегруженности: пакеты ждут очереди, ожидая ссылки для использования. Пакеты проходят по одному прыжку за 1 раз. Последовательность пакетов А и Б не имеют ограниченной модели, ширина канала распределяется по требованию. При пакетной коммутации большее кол-во пользователей могут использовать сеть. **К примеру**; У нас есть 1 Mb/s сеть: 100kb/s когда активен, активен 10% от времени. При **пакетной** коммутации: до 35 пользователей (вероятность, что будет >10 активных в одно и тоже время, меньше чем 0.0004). При **канальной**: 10 пользователей. **Канальная коммутация:** Гарантирует ширину канала необходимую для audio/video приложений. **Пакетная коммутация** больше подходит для неравномерных данных, использование ресурсов. При высокой перегруженности задержка пакетов и потеря, при этом сама коммутация проще. **Пакетная:** Разобьем сообщение на 5000 пакетов, каждый пакет по 1500 битов, 1 мсек чтобы передать 1 пакет на один передатчик (звено, link). Каждое **звено?(link)** работает параллельно. Задержка, варьируется от 15 сек до 5.002 сек. **Длинна пакета:**




9. Multipleksimine sageduse, aja ja koodi järgi

Мультиплексирование – Процесс объединения отдельных потоков или каналов в один логический поток данных таким образом, что они позднее могут быть восстановлены в прежнем виде без ошибок. Поддерживает несколько соединений на одной машине. Отображение нескольких соединений на одном уровне от одного соединения к другому. Собирание определенного количества соединений в один оптический кабель. По способу уплотнения технологии мультиплексирования можно разделить на две основные категории: FDM и TDM. При FDM (Frequency division Multiplexing) частотный спектр делится на логические каналы, причем каждый пользователь получает этот канал в свое распоряжение на время разговора. При TDM (Time Division Multiplexing)

пользователям периодически выделяется вся полоса, но только на краткий период времени (см. вопрос 8).

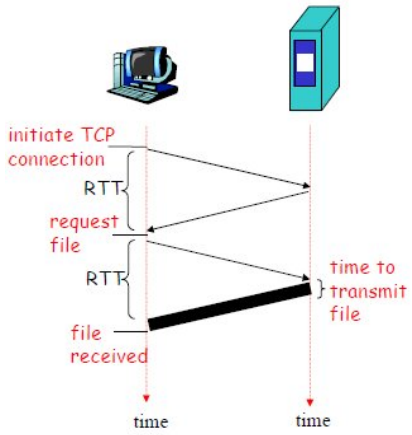
10. Ajalised viited võrkudes

RTT – это время путешествия пакета от клиента к серверу и назад, подтверждение получения.

Всего при передаче 2 времени + время передачи самого файла:

1) Инициализация TCP соединения
2) Запрос HTTP и первые несколько байт ответа возврата HTTP.

Компоненты: processing overhead, transition time – зависит от пропускной способности и длины сообщения, propagation delay – время, за которое один бит, проходит от начала соединения до конца, queuing delay – время ожидания для общего соединения.




11. **Arvutivõrgude ja Interneti ajalugu**

***1961-1972: Развитие коммутации пакетов.***

**1961** Первая научная работа по теории коммутации пакетов (Леонард Клейнрок). С помощью теории очеред эта работа наглядно продемонстрировала эффективность принципа коммутации пакетов в условиях неравномерной нагрузки.

**1964** Коммутация пакетов в защищенных военных сетях. (Пол Бэрен)

**1967** разработана схема компьютерной сети ARPAnet компанией ARPA, основанной на коммутации пакетов Является прообразом сети Интернет.

**1969** первый узел ARPAnet, связывающий между собой несколько научных организаций США.

**1970** коротковолновая сеть ALOHAnet на Гавайских островах.

**1972** первая публичная демонстрация ARPAnet на Международной конференции по компьютерным коммуникациям. В сети ARPAnet насчитывалось уже 15 узлов.

Первый протокол обмена информацией между оконечными системами – NCP(Network-Control Protocol – протокол управления сетью)

Первая программа для работы с e-mail

***1972-1980: Возникновение новых компьютерных сетей***.

**1974** создание сети, соединяющей несколько сетей (сеть сетей) – Уинтон Серф и Роберт Канн.

**1976** Ethernet(пакетная технология компьютерных сетей). Разработана корпорацией Xerox PARC

**Конец 1970х** патентованные архитектуры:DECnet(Digital Equipment Corporation network), SNA(Systems Net Architecture), XNA

Передача пакетов фиксированной длины (начальное развитие ATM)

**1979** В сети ARPAnet насчитывалось уже 200 узлов.

***1980-1990: Новые протоколы, распространение компьютерных сетей***.

Новые сети: CSNET, BITNET, NSFNET, Minitel

100 тысяч хостов подключено к конфедерации сетей.

**1982** создание SMTP протокола для работы с e-mail

**1983** замена стандартного протокола NCP стеком протоколов TCP/IP (с этого времени стек TCP/IP используется всеми хостами Интернета)

Разработана система доменных имен DNS (Domain Name System)(name-to-IP-address translation)

**1985** создание FTP протокола

**1988** Механизм управления нагрузкой в работе протокола TCP

***1990-е,2000-е: Распространение Интернета, Web***

**Начало 1990х** конец существования ARPAnet

Появление всемирной паутины Web (Тим Бернерс-Ли)

Развитие идеи гипертекста(Тим Бернерс-Ли), предложенные Бушем(1945) и Нельсоном(1960-е)

Первоначальные версии HTML, HTTP, web-сервера и браузера (Тим Бернерс-Ли).

**1991** ограничение на коммерческое использование NSFNET

**1994** разработан браузер Mosaic, позже переименованный Netscape. (Марк Андресен)

**1995** конец существования NSFNET

**Конец 1990х – 2000-е** Внедрение новых web-серверов

Небывалый прогресс в области Интернет-технологий: приложения электронной почты, для обемена мгновенными сообщениями, одноранговые (P2P) сети (доступ пользователя к ресурсам, обмен файлами)
Обеспечение сетевой безопасности
каналы связи, работающие на Gbps
около 50 млн. хостов, подключенных к Интернет, 100 млн. пользователей.
***2007-…*** Около 500 млн. хостов
Голосовые и видео сообщения
P2P приложения: BitTorrent(обмен файлами), Skype(VoiceIP), PPLive(видео)
Другие приложения: игры – онлайн, YouTube
Развитие беспроводных и переносных устройств

**12. Mida erinevad rakendused nõuavad võrkudelt**.

Для успешного обмена сообщениями между процессами, выполняющимися на двух различных хостах, необходимо, чтобы они могли идентифицировать друг друга. Идентификация требует следующей информац процессе: идентификатор процесса внутри хоста; имя или адрес хоста, которому принадлежит процесс. В Интернет-приложениях хосты идентифицируются с помощью IP-адресов(представляет собой 32-разрядное двоичное число). Идентификатор включает в себя IP-адрес и номер порта, уникальный для каждого процесс

***Службы, необходимые приложению***.

Сокет – программный интерфейс для обеспечения обмена данными между процессами, в частности, между прикладным процессом и протоколом транспортного уровня. На передающей стороне сообщения через соке оказываются на транспортном уровне, где получают возможность перемещаться внутри сети. Сетевые служ обеспечивают доставку сообщения на транпортный уровень адресата, где оно через сокет попадает в нужнее приложение и обрабатывается им.

Выделяются три основных требования, предъявляемых приложениями к транспортному уровню: надежная передача данных, скорость передачи и время передачи.

**Надежная передача данных**

Некоторые приложения(эл. почта, передача файлов, просмотр web-документов, финансовых операций) треб надежной 100% передачи данных, т.е. исключения вероятности потерь данных при передачи. Другие приложения – *толерантные к потерям данных*(мультимедиа, аудио и видео реального времени).Небольшие потери данных оборачиваются помехами, не приводящими к сбоям или серьезным потерям качества.

**Скорость передачи данных**

Приложения, эффективность которых зависит от скорости передачи данных, называют *чувствительными к скорости передачи данных* (многие мультимедиа-приложения). Другие приложения(приложения эл.почты, приложения) способны адаптироваться к используемому каналу связи и относятся к классу *эластичных* приложений.

**Время передачи данных**

Интерактивные приложения реального времени(Интернет-телефония, виртуальные миры, телеконференции,многопользовательские компьютерные игры) накладывают жесткие временные ограничен на время доставки данных в целях эффективности и во избежание, например в Интернет- телефонии, длительных пауз при разговоре.

Таблица 2.1. Требования приложений к качеству передачи данных

| Приложение доставки | Потеря данных | Скорость передачи | Ограничение на время |
|---|---|---|---|
| Передача файлов | Недопустима | Эластичность | Нет |
| Электронная почта | Недопустима | Эластичность | Нет |
| Работа с web-документами | Недопустима | Эластичность (несколько Кбит/с) | Нет |
| Аудио и видео реального времени | Допустима | Аудио: несколько Кбит/с-1 Мбит/с, видео: 10Кбит/с-5 Мбит/с | Есть, сотни миллисекунд |
| Записанное потоковое аудио и видео | Допустима | Аналогично предыдущему | Есть, несколько секунд |
| Интерактивные игры | Допустима | 1-10 Кбит/с | Есть, сотни миллисекунд |
| Обмен сообщениями в реальном времени | Недопустима | Эластичность | Есть и нет |

***Службы протоколов транспортного уровня***

**TCP**(Transmission Control Protocol – протокол управления передачей)
**UDP**(User Datagram Protocol – протокол пользовательских дейтаграмм)

Эти протоколы предлагают разные модели обслуживания.

**Протокол TCP**

- установление логического соединения между клиентом и сервером до начала передачи «полезных» данных. TCP-соединение является дуплексным, т.е. обе стороны могут осуществлять передачу одновременн TCP-соединение называется логическим, т.к. представляет собой совокупность информационных единиц переменных).
- Надежная передача данных: клиент – сервер получают гарантию, что все переданные данные будут доставлены без ошибок, потерь и в правильном порядке.
- Выходной поток байтов передающей стороны в точности соответствует входному потоку байтов принимающей.
- Включает механизм контроля перегрузки(если сеть перегружена, происходит автоматическое снижен скорости обмена данными)

Недостатки TCP: не обеспечивает гарантированную надежность, скорость и время передачи данных; принудительно снижает скорость обмена в случае перегрузки сети.

**Протокол UDP**

Предоставляет более простую модель обслуживания, т.к. не устанавливается логическое соединение между сокетами. Однако, UDP обеспечивает ненадежную передачу данных(отсутствие приложения, дающего гара что пакет будет принят адресатом); не гарантирует, что порядок передачи данных при получении совпадает порядком при отправлении данных. Обмен данными в приложении происходит с необходимой скоростью,т. нет контроля перегрузок.

Таблица 2.2. Протоколы, использующиеся в популярных Интернет-приложениях

| Приложение | Прикладной протокол | Транспортный протокол |
|---|---|---|
| Электронная почта | SMTP (RFC 2821) | TCP |
| Доступ с удаленного терминала | Telnet (RFC 854) | TCP |
| Web | HTTP (RFC 2616) | TCP |
| Передача файлов | FTP (RFC 959) | TCP |
| Удаленный файловый сервер | NFS [325] | UDP или TCP |
| Потоковое мультимедиа | Обычно фирменный (например, Real Networks) | UDP или TCP |
| Интернет-телефония | Обычно фирменный (например, Dialpad) | Как правило, UDP |

## 13. HTTP

HTTP(*HyperText Transfer Protocol* — «протокол передачи гипертекста») - протокол прикладного уровня передачи данных.

Терминология: *web-страница* состоит из *объектов*. Объект-это обычный файл в формате HTML, т.е. единица, обладающая собственным универсальным указателем ресурса (URL). *Браузером* называется агент пользователя web, он отображает web-страницы, а также выполняет служебные функции.

Реализуется с помощью двух программ: клиента и сервера, которые находясь на разных системах, обмениваются HTTP-сообщениями. Когда пользователь запрашивает web-страницу, браузер посылает серверу HTTP-запрос объектов, составляющих web-страницу. Сервер получает запрос и высылает ответные сообщения, содержащие требуемые объекты. Порядок обмена и содержания сообщений описаны в протоколе.

HTTP используют в качестве протокола транспортного уровня TCP. HTTP клиент сначала устанавливает TCP-соединение с сервером, а после соединения клиент и сервер начинают взаимодействовать и протоколом, происходит обмен HTTP-сообщениями и затем TCP ссоединение закрывается. Сервер не содержит информации о предыдущем клиенте (HTTP stateless).

Протокол HTTP поддерживает постоянные и непостоянные соединения. При непостоянном соединении протокол TCP получает лишь один объект, а при постоянном соединении все объекты.

Непостоянное соединение:

1. HTTP клиент инициирует TCP-соединение с сервером через порт 80.

1a. HTTP сервер ждёт TCP-соединение, принимает его и оповещает клиента.

2. HTTP клиент посылает запрос-сообщение(сод. URL) на сокет URL-соединения, которое показывает какое объект хочет клиент.

3. HTTP сервер принимает сообщение, оформляет ответное сообщение, содержащее запрашиваемый объект и посылает сообщение в его сокет.

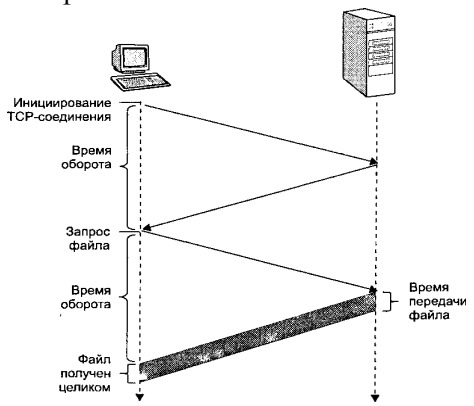
4. HTTP сервер закрывает TCP-ссоединение.

5. HTTP клиент принимает ответное сообщение содержащее HTML файл, извлекает файл, обрабатывает его и выделает ссылки на 10 объектов(JPEG-файлов)
6. Шаги 1-5 повторяются для каждого из 10 объектов.

Оценка времени ответа на запрос HTML-файла.

RTT-временной оборот, время за которое пакет малой длины передастся от клиента к серверу и обратно. Полное время на передачу=2*RTT+передача+время(см.рис.)

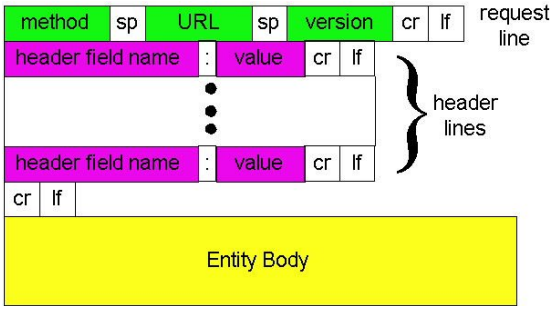
Недостатки непост.соед.: для каждого объекта новое соединение, которое требует от TCP-протокола выделения буфера, временные затраты.




Постоянное соединение:

- Сервер оставляет соединение открытым после передачи сообщений
- Сообщение посылаются через открытое соединение
- Клиент посылает запрос, как только завершится текущий приём
- Малое время на передачу

В HTTP существуют два типа сообщений: запросы и ответы.



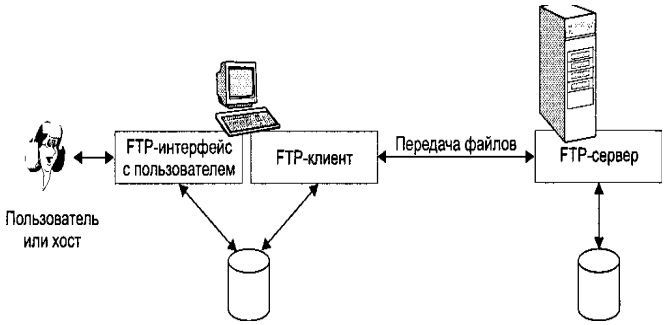

Запрос. ASCII-кодировка, состоит из 5 строк
Поле метода: GET, POST, HEAD
Сообщение ответ: состоит из трёх частей(строка состояния, 6 строк заголовка, тело сообщения)
Взаимодействие пользователя с сервером: **авторизация**-требуется ввести имя пользователя и пароль для доступа, клиент должен авторизироваться каждый раз. Клиент формирует запрос, сервер возвращает ответ с пустым телом и кодом состояния 401autorization req. После получения объекта клиент продолжает посылать имя и пароль, пока не закончит работу.
Cookie: 4 компонента:

1. заголовочная Cookie-строка в ответном сообщении сервера
2. заголовочная Cookie-строка в запросе клиента
3. Cookie-файл, нах. на стороне клиента и обрабатываемого браузером
4. удалённая база данных, расположенная на web-сайте

web-кэширование-сохранение уже загруженных объектов, сокращает время загрузки страницы, может осуществляться клиентом и промежуточным кэщ-сервером, содержимое кэщ-объекта постоянно устаревает, можно обновлять(GET-метод с условием)
Прокси. Во время использования прокси-сервера протягиваются все страницы через proxy сервер, если страниц нет, то сервер сам находит её новое место, чтобы в дальнейшем пользователи смогли попасть на нужную страницу.
Proxy u cache экономят время и нагрузку на линии.

## 14. FTP (file transfer protocol)

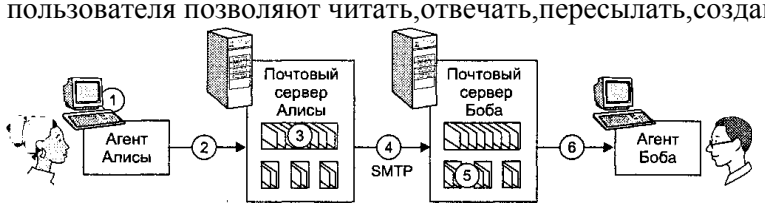
Используется для передачи файлов, находящихся на двух хостах-локальном и удалённом, модель клиент-сервер, ftp: RFC 959, порт 21. инфо пользователя и каталога сохраняются, не надо каждый раз водить пользователя и пароль в начале каждой передачи, пользователям посылается код ответа и его значение.




Локальная файловая система Удалённая файловая система

**15. Mail, SMTP, MIME, POP3**

Mail- средство связи, высокая скорость доставки, простота пользования, низкая стоимость. Структура системы почты(агенты пользователя, почтовые серверы, протокол SMTP). Агенты пользователя позволяют читать,отвечать,пересылать,создавать и сохранять письма




SMTP(Simple Mail Transfer Protocol)-клиент устанавливает TCP-соединение с 25 сервером, клиент определяет адреса почтовых ящиков получателя и отправителя сообщения, передача сообщения, после передачи клиент закрывает соединение с сервером.

MIME(Multipart Interner Mail Extensions)-многоцелевая почта Интернета. Заголовки: Content-type, Content-Transfer-Encoding. Разрешает посылает данные отличной от ASCII кодировки. Указывается от кого, кому, время, тема, и что именно.

POP3 используется для прочтения писем, один из самых простых протоколов, начинает действовать после того, как агент пользователя устанавливает TCP-соединение с портом 110 почтового сервера, обязательно выполнение трёх операций: авторизация, транзакция и обновление. Почтовый сервер сохраняет определённую информацию о состоянии, не надо сохранять инфо о сеансе.

**16. DNS**

Существуют два принципиально разных способа идентификации хостов: с помощью имен и с помощью IP-адресов. Имя хоста удобно для людей в силу своей мнемоничности, а IP-адрес, являющийся компактной числовой величиной фиксированного размера, проще обрабатывать маршрутизаторами. Для того чтобы установить связь между этими двумя идентификаторами, используется *система доменных имен* (Domain Name System, DNS). DNS представляет собой, с одной стороны, базу данных, распределенную между иерархически структурированными *серверами имен,* и, с другой стороны, протокол прикладного уровня, организующий взаимодействие между хостами и серверами имен для выполнения операций преобразования. Зачастую серверы имен являются UNIX-машинами, использующими программное обеспечение BIND (Berkeley Internet Name Domain — домен имен Интернета Беркли). Протоколу DNS назначен порт с номером 53, и работает DNS поверх протокола UDP транспортного уровня.

Обычно DNS используетсядругими протоколами прикладного уровня: HTTP, SMTP и FTP для получения IP-адресов вместо вводимых пользователями имен хостов. При работе протокола DNS пользовательский хост играет роль клиента. Браузер выделяет из URL-адреса страницы имя хоста и передает его клиентской стороне DNS-приложения, которая формирует и отправляет запрос DNS-серверу. DNS-сервер обрабатывает запрос и отсылает клиенту ответ, содержащий IP-адрес хоста. Помимо преобразования имен хостов в IP-адреса, DNS выполняет еще несколько важных функций, перечисленных ниже.

• *Поддержка псевдонимов серверов.* Обычно введение псевдонимов вызывается недостаточной мнемоничностью канонического имени. Получение канонического имени хоста по заданному псевдониму, как и IP-адреса, выполняется с помощью протокола DNS.

• *Поддержка псевдонимов почтовых серверов.* Очевидно, что мнемоничность особенно важна для адресов электронных почтовых ящиков.

• *Распределение загрузки.* В последнее время DNS все больше используется для распределения загрузки между дублирующими серверами. Популярные сайты,

например cnn.com, зачастую имеют несколько копий (или реплик, или зеркал), расположенных на различных серверах с различными IP-адресами.
Как устроена служба DNS? Ее можно представить себе в виде единственного центрального сервера имен, который содержит всю информацию об именах хостов и их IP-адресах. Этот сервер принимает все запросы и отсылает ответы напрямую каждому DNS-клиенту. К сожалению, огромное (и постоянно растущее) число Интернет-хостов не позволяет применить эту простую схему на практике. Централизованной системе присущи некоторые ≪врожденные≫ недостатки.
• *Единственная точка возможного отказа.* Сервер имен является ≪узким местом≫ с точки зрения надежности — его отказ парализует работу всего Интернета.
• *Объем трафика.* Сервер имен является ≪узким местом≫ с точки зрения загрузки, поскольку вынужден обрабатывать все запросы, поступающие с сотен миллионов хостов всего мира.
• *Удаленность централизованной базы данных.*
• *Обслуживание.*

**17. Töökindel andmeedastus**

Служба надежной передачи данных обслуживает канал, по которому осуществляется надежная передача сообщений верхних уровней коммуникационной модели. При надежной передаче не происходит искажений битов, то есть изменений их значений с 0 на 1 или наоборот; кроме того, данные доставляются в том порядке, в котором они были отправлены.

***Надежная передача данныхпо абсолютно надежному каналу***
Нет искажения битов;
Нет потери пакетов;


rdt_send(data)

packet=make_pkt(data)

udt_send(packet)

**a**


rdt_rcv(packet)

extract(packet.data)

deliver_data(data)

**b**

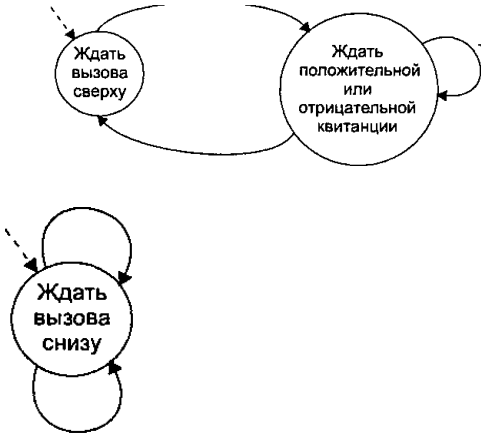
*Протокол rdt 1.0 — передача по абсолютно надежному каналу связи:*

*a — передающая сторона; b — принимающая сторона*

Передающая сторона посылает данные в низлежащий канал;
Принимающая сторона считывает данные из низлежащего канала;

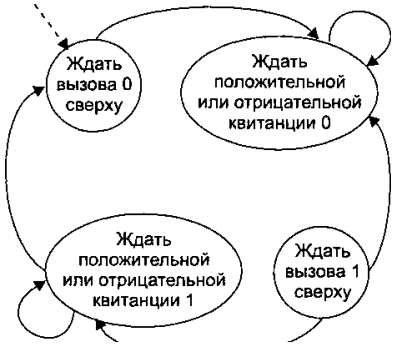
***Надежная передача данных по каналу, допускающему искажение битов***
Для разрешения проблем искажения битов используются три дополнительных механизма:
*Обнаружение ошибок.*
*Обратная связь с передающей стороной.*
ACKs (Acknowledgements)- получатель говорит отправителю, что пакет успешно принят.

NAKs (Negative acknowledgements) - получатель говорит отправителю, что пакет содержит ошибки
*Повторная передача.*




ACK/NAK могут содержать ошибки, быть поврежденными. В етом случае принимающая сторона не получает никакой информации о результатах передачи последних пакетов. Что делать?

- Отправитель повторно передает файл, который был искажен
- Отправитель добавляет порядковый номер к каждому пакету
- Получатель избавляется от продублированного пакета



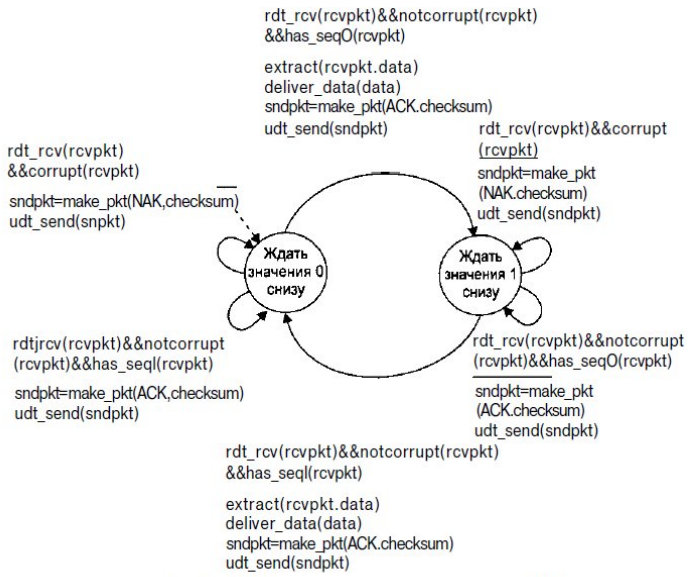



**Рис. 3.11. Принимающая сторона протокола rdt 2.1**

Отправитель:

- порядковый номер дается пакету
- 2-ух порядковых номеров (0,1) достаточно
- Проверяет ACK/NAK на искаженность
- есть состояние "запомнить", если данняи порядковый номер 0 или 1

Получатель:

- должен проверит, не является ли принятый файл дубликатом
- полу4ател не знает, без повреждений ли дошел его ACK/NAK

***Надежная передача данных по каналу, допускающему искажение битов и потерю пакетов.***

Канал передачи может также и терять пакеты (данные или ACK ). Что делать? Отправител ждет пока определенные данные или ACK не будут потеряны, затем посылает повторно. Как ето происходит? Отправител ждет определенное количество времени, пока не получит ACK . Если не пришло, то передает повторно. Если пакет/ACK просто задержался, то повторная передача будет продублирована и получатель должен опреелить порядковыи номер пакета, который получил ACK. Также отправителю нужен таимер обратного отсчета.

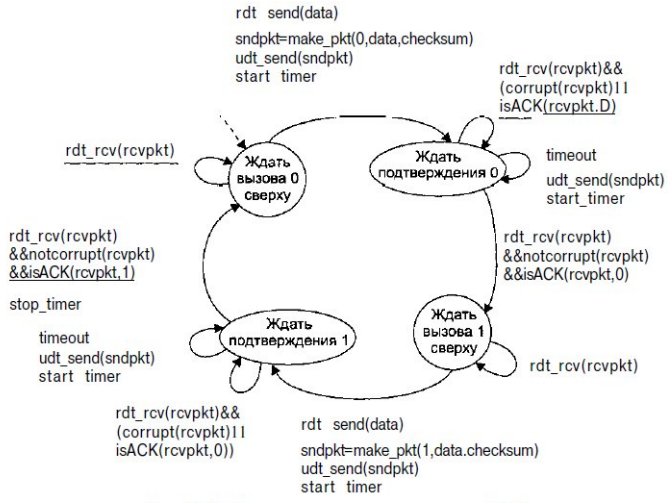



**Рис. 3.14. Передающая сторона протокола rdt 3.0**

**18. Go-back-n**

**Возвращение на N пакетов назад**

В протоколах, использующих метод *возвращения на N шагов назад* (Go-Back-N, GBN), передающая сторона может посылать последовательности пакетов, не ожидая квитанций, при этом максимальная длина последовательности ограничена некоторым числом *N.* На рис. 3.18 представлена схема определения диапазона порядковых номеров. Если base — порядковый номер самого ≪старого≫ неподтвержденного пакета, а nextseqnum — наименьший из ≪свободных≫ порядковых номеров (то есть номер, который будет присвоен следующему посылаемому пакету), то полный диапазон порядковых номеров можно разделить на четыре части. Номера в диапазоне от 0 до base-1 назначены переданным ранее пакетам, для которых уже получены квитанции. Номера от base до nextseqnum-1 также относятся к переданным пакетам, однако для этих пакетов квитанции еще не получены. Номера от nextseqnum до base+N-1 могут быть назначены пакетам, которые необходимо сформировать и отправить сразу при поступлении новых данных от прикладного уровня. Номера, превышающие значение base+N, нельзя использовать до тех пор, пока не будет получена квитанция для текущего пакета, то есть пакета с порядковым номером base.

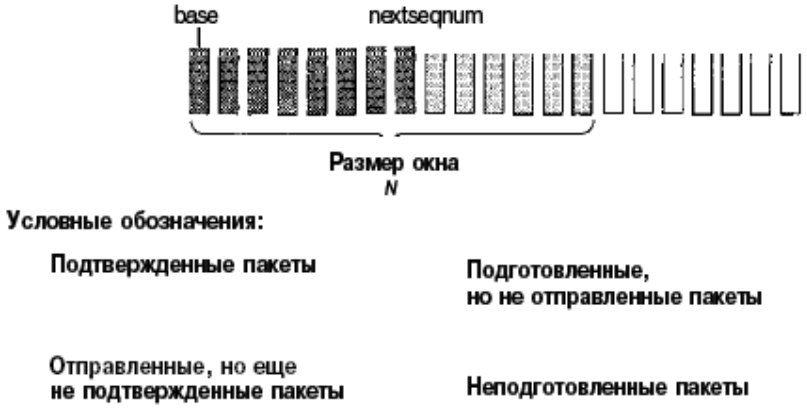



**Рис. 3.18. Диапазон порядковых номеров с точки зрения передающей стороны**

Как показано на рисунке, диапазон порядковых номеров для пакетов, которые могут быть переданы без ожидания квитанций, можно рассматривать в виде ≪окна≫ размером *N,* лежащего внутри множества порядковых номеров. В процессе выполнения протокола это окно ≪движется≫ в сторону увеличения порядковых номеров. Такое представление определило терминологию, в соответствии с которой значение *N* называют *размером окна,* а протокол GBN — *протоколом скользящего окна.* Вероятно, у вас возникает вопрос: для чего нужно ограничивать число пакетов, передаваемых без ожидания подтверждения, размером окна ЛГ? В следующем разделе этой главы мы увидим, что одной из причин является необходимость контролировать передаваемый поток данных.

На практике порядковый номер пакета хранится в одном из полей заголовка, имеющем фиксированную длину. Если *k* — число битов этого поля, то диапазоном порядковых номеров *является* [0,*2k* - 1]. Поскольку множество порядковых номеров конечно, все арифметические операции с ними производятся по модулю *2k* (другими словами, множество порядковых номеров можно представить в виде кольца, в котором числом, следующим за *2k* - 1, является 0). Возвращаясь к протоколу rdt 3.0, отметим, что разрядность его порядковых номеров равна 1, а диапазон — [0,1]. В конце этой главы перечислены несколько проблем, порождаемых ограниченностью множества порядковых номеров. Как мы увидим позже, в протоколе TCP порядковые номера имеют разрядность 32 и предназначены для подсчета байтов

(вместо пакетов) в байтовом потоке.

**19. Selective-repeat.**

Метод выборочного повторения (Selective Repeat, SR) направлен на снижение количества повторных передач путем отправки только тех пакетов, которые были зафиксированы принимающей стороной как потерянные или искаженные. Это требует введения отдельных квитанций для каждого успешно принятого пакета. Как и в GBN-протоколе, число пакетов, находящихся в конвейере, ограничивается размером окна N, однако передающая сторона может получать квитанции для некоторых пакетов окна.

Работа передающей стороны SR-протокола.

1. *Получены данные от верхнего уровня.*

Если следующий порядковый номер пренадлежит окну, посылается пакет с данными.

2. *Истек интервал ожидания.*

Для обнаружения потерь пакетов прибегают к использованию таймера, пакет посылается заново.

3. *Получено подтверждение.*

Передающая сторона помечает соответствующий пакет как принятый (при условии, что он принадлежит окну). Если оказывается, что порядковый номер пакета совпадает со значением send_base, база окна сдвигается вперед к неподтвержденному пакету с наименьшим порядковым номером. Если после сдвига окна обнаруживаются пакеты, попавшие внутрь окна и еще не переданные, осуществляется их передача.

Работа принимающей стороны SR-протокола.

1. *Успешно принят пакет с номером, лежащим в диапазоне [rcv_base,rcv_base + N - 1].*

В этом случае принятый пакет принадлежит окну, и принимающая сторона генерирует подтверждение для этого пакета. Если прием пакета осуществляется впервые, он заносится в буфер. В случае совпадения порядкового номера пакета с базой окна этот пакет вместе со всеми пакетами, хранящимися в буфере, образует последовательность с наименьшим номером rcv_base. Данные, извлеченные из последовательности, передаются верхнему уровню; затем происходит сдвиг окна передающей стороны на длину последовательности.

2. *Принят пакет с номером, лежащим в диапазоне [rev_base - N,rcv_base - 1].*

Несмотря на то что квитанция для этого пакета уже была послана передающей стороне ранее, в этом случае предусматривается повторное квитирование пакета.

3. *Иначе пакет игнорируется.*

**20. TCP ühenduse loomine ja sulgemine** - установление и закрытие TCP соединения

TCP – протокол транспортного уровня , обеспечивающий надежную передачу данных с установлением соединения .Процедура установления соединения выглядит так : 1)Клиентский процесс сообщает транспортному уровню о необходимости установить соединение с серверным процессом 2) Транспортный уровень клиента устанавливает соединение с транспортным уровнем серверного процесса : посылает спец. TCP сегмент , сервер отвечает TCP сегментом и клиент снова посылает TCP сегмент , первые 2 сегмента не содержат прикладные данные , а третий спец. Сегмент может уже содержать их . Это и есть ***процедура тройного рукопожатия*** . TCP обеспечивает дуплексную передачу данных , начать передачу может любая сторона

Подробнее : 1) Клиент посылает SYN сегмент в сторону сервера и добавляет туда порядковый номер (sequence nr) 2) Сервер посылает клиенту SYN сегмент добавляя туда порядковый номер и ответ ACK о принятии сегмента посланного клиентом. 3) Клиент посылает ответ ACK о принятом сегменте сервера … Создаются буферы и переменные на стороне клиента и сервера.

Закрытие : 1) Клиент посылает FIN сегмент серверу 2) Сервер посылает ответ ACK, закрывает соединение и посылает FIN сегмент 3) Клиент посылает ACK ответ и переходит в timed wait состояние + отвечает ACK на все FIN сегменты 4) Сервер получает ответ и закрывает соединение.

**21. TCP töökindel andmeedastus** , надежная передача данных - протокол сетевого уровня IP ненадежную службу передачи данных , он не даёт гарантий относительно целостности , доставки и сохранения порядка дейтаграмм. Дейтаграммы же являются средством передачи сегментов траспортного уровня .

К счастью , в TCP протоколе предусмотрена надежная передача данных . Он гарантирует , что данные которые TCP процесс считывает из своего буфера обладают целостностью ( не содержат пропусков и дублирований, без искажений и тд) Для этого применяются след. механизмы :

1)связывание каждого неподтвержденного сегмента с индивидуальным таймером и положительные квитанции-подтверждения – TCP протокол постоянно проверяет корректно принятые данные и в случае необходимости передаёт снова

2) Для обнаружения потерь и дублирования сегментов используются порядковые номера

3)Используется механизм конвейеризации - несколько сегментов передаются без ожидания проверки каждого из них . Конвейеризация значительно повышает коэфф. полезности.

**22. TCP taimerid** - Чтобы предотвратить зависание протокола в случае потери пакетов, каждое соединение поддерживает набор таймеров, использующихся для восстановления после потерь или сбоев другого узла.


1. Таймер повторных передач (retransmission; RTO) контролирует время прихода подтверждений (ACK). Таймер запускается в момент посылки сегмента. При получении отклика ACK до истечения времени таймера, он сбрасывается. Если же время таймера истекает до прихода ACK, сегмент посылается адресату повторно, а таймер перезапускается.
2. Таймер запросов (persist timer), контролирующий размер окна даже в случае, когда приемное окно закрыто. Если пакет будет потерян, возникнет тупик, тогда каждая из сторон ждет сигнала от партнера. Именно в этой ситуации и используется таймер запросов. По истечении времени этого таймера отправитель пошлет сегмент адресату. Отклик на этот сегмент будет содержать новое значение ширины окна. Таймер запускается каждый раз, когда получен сегмент с window=0.
3. Таймер контроля работоспособности (keepalive), который регистрирует факты выхода из строя или перезагрузки ЭВМ-партнеров. Время по умолчанию равно 2 часам.

При получении любого сегмента от клиента таймер сбрасывается и запускается вновь.

4. 2MSL-таймер (Maximum Segment Lifetime) контролирует время пребывания канала в состоянии TIME_WAIT. Таймер запускается при выполнении процедуры active close в момент посылки последнего ACK.

Важным параметром, определяющим рабочие параметры таймеров, является RTT (время путешествия пакета до адресата и обратно). TCP-агент самостоятельно измеряет RTT. Такие измерения производятся периодически и по их результатам корректируется среднее значение RTT:

$RTT_m = a \times RTT_m + (1-a) \times RTT_i$,

где $RTT_i$ - результат очередного измерения, $RTT_m$ - величина, полученная в результате усреднения предыдущих измерений, a - коэффициент сглаживания, обычно равный 0.9

**23. TCP voo juhtimine** , управление потоком –если между хостами установлено TCP соединение ,то у них также имеются приемные буферы , где хранится информация принятая без искажений и в правильном порядке. Приложение связанное с TCP соединением считывает данные из приемного буфера в произвольные моменты времени и в один момент может возникнуть угроза переполнения буфера. Таким образом , основная задача службы контроля потока – избежать переполнения буфера .

Для ускорения и оптимизации процесса передачи больших объемов данных протокол TCP определяет метод управления потоком, называемый методом скользящего окна, который позволяет

отправителю посылать очередной сегмент, не дожидаясь подтверждения о получении в пункте назначения предшествующего сегмента.
Протокол TCP формирует подтверждения не для каждого конкретного успешно полученного пакета, а для всех данных от начала посылки до некоторого порядкового номера ACK SN (Acknowledge Sequence Number) исключительно. В качестве подтверждения успешного приема, например, первых 2000 байт, высылается ACK SN = 2001: это означает, что все данные в байтовом потоке под номерами от ISN+1=1 до данного ACK SN -1 (2000) успешно получены (см. рис. 3.1.2).

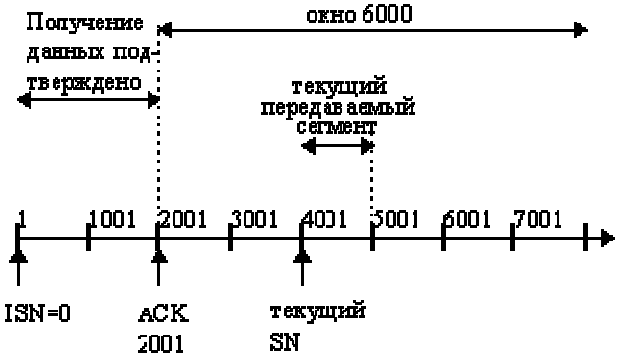



Рис. 3.1.2. Метод скользящего окна

Вместе с посылкой отправителю ACK SN получатель объявляет также “размер окна”, например - 6000. Это значит, что отправитель может посылать данные с порядковыми номерами от текущего ACK SN = 2001 до (ACK SN + размер окна -1) = 8000, не дожидаясь подтверждения со стороны получателя. Допустим, в данный момент отправитель посылает тысячеоктетный сегмент с порядковым номером данных SN=4001. Если не будет получено новое подтверждение (новый ACK SN), отправитель будет посылать данные, пока он остается в пределах объявленного окна, то есть до номера 8001. После этого посылка данных будет прекращена до получения очередного подтверждения и (возможно) нового размера окна. Однако размер окна выбирается таким образом, чтобы подтверждения успевали приходить вовремя и остановки передачи не происходило - для этого и предназначен метод скользящего окна. Размер окна может динамически изменяться получателем.
Для временной остановки посылки данных достаточно объявить нулевое окно. Но даже и в этом случае через определенные промежутки времени будут отправляться сегменты с одним октетом данных. Это делается для того, чтобы отправитель гарантированно узнал о том, что получатель вновь объявил ненулевое окно, поскольку получатель обязан подтвердить получение “пробных” сегментов, а в этих подтверждениях он укажет также и текущий размер своего окна.
Как уже было сказано выше, протокол TCP позволяет вести полнодуплексную передачу. Один и тот же сегмент, высылаемый, например, из B в A, может содержать в заголовке служебную информацию по подтверждению получения данных от A, а в поле данных - полезные данные для A.

**24. TCP koormuse juhtimine** - управление нагрузкой или контроль перегрузок

TCP протокол осуществляет контроль за перегрузками , потому как сетевой протокол IP не предоставляет механизмов для информации или контроля перегрузок в сети
Подход применяемый в протоколе TCP заключается в ограничении скорости передачи источников в зависимости от наблюдаемой нагрузки . Если перегрузки в сети не наблюдаются , источник увеличивает скорость передачи , иначе снижает её.
Механизм ограничения скорости : Каждая сторона TCP соединения состоит из буферов передачи и приема, а также набора переменных ( LastByteRead , RcvWindow , etc) . Механизм также использует доп. Переменную CongWin – *окно перегрузки*, она влияет на скорость передачи данных в сеть таким образом : объем неподтвержденных данных которые может передать источник не превышает минимального из значений CongWing , RcWindow , те
LastByteSent – LastByteAcked <= min ( Congwin, RcvWin)
Меняя значение окна перегрузки - протокол регулирует скорость.
Основные состовляющие алгоритма контроля за перегрузками : Аддитивное увеличение вместе мультипликаторным уменьшением , медленный старт и реакция на истечение интервала ожидания.
Общий алгоритм : Передачу начинают с использованием SlowStart алгоритма : CongWin = 1 , далее для каждого подтвержденного сегмента CongWin++ , пока не будет потерь или CongWin не дойдёт

до значения перехода. Происходит переход и значние перехода можно уменьшить в 2 раза CongWin / 2 , и снова запустить алгоритм Slowstart and CongWin=1
Пропускная способность = W * MSS / RTT(B/s)  где W – кол-во посланных сегментов за один RTT , MSS – max segment size, RTT – round trip time
Один из примеров перегрузки :

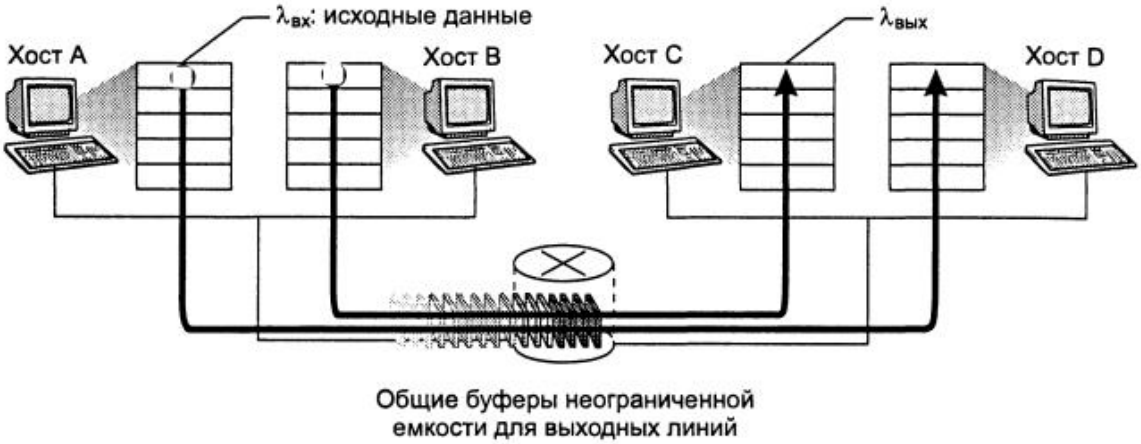



**Рис. 3.38.** Ситуация 1: два соединения совместно используют маршрутизатор с буферами неограниченной емкости

**25. UDP –user datagramm protocol** – простая модель обслуживания , протокол транспортного уровня – он выполняет минимальное количество действий необходимое для протокола транспортного уровня.В основном это операции мультиплексирования и демультиплексирования , плюс также простая проверка наличия ошибок в данных. Особенность и отличие протокола заключается в том , что логическое соединение между сокетами не устанавливается ( нет процедуры «рукопожатия»). Однако протокол обеспечивает ненадежную  передачу данных , тк нет никаких гарантий , что пакет  дойдёт до адресата и даже то , что порядок отправления информации может не соответствовать порядку получения информации
Также UDP не предполагает контроля перегрузок и фактически приложение может передавать данных с необходимой скоростью, однако протокол как и TCP не гарантирует время доставки данных
Использование этого протокола напрямую связано с протоколом сетевого уровня IP . UDP получает сообщения от протокола прикладного уровня,  добавляет к ним номера портов отправителя и получателя , и передает сегмент сетевому уровню . Сетевой уровень же передает дейтаграмму хосту получателю по «возможности» , поэтому протокол UDP и называют протоколом без рукопожатия .
Преимущества : *Отсутствие процедуры установления соединения* – протокол TCP требует тройного рукопожатия, UDP лишен этого и поэтому не имеет дополнительных задержек и используется как прикладной протокол DNS
*Отсутствие информации о состоянии соединения* – в отличие от TCP позволяет обслуживать большее кол-во клиентов
Размер заголовка – 8байт vs 20 байт у TCP
Механизм управления передачей данных приложением -  в UDP данные сразу упаковываются в сегмент и передаются сетевому уровню, тем самым UDP более актуален для приложения реального времени .


**Рис. 3.6.** Структура UDP-сегмента

**26. Datagramm and virtuaalahelatega vorgud** - дейтаграммные и виртуальные сети

**Дейтаграммные сети** – если провести аналогию, то дейтаграммные сети очень похожи на обычные почтовые службы . Каждый передаваемый пакет содержит информацию об адресе получателя , который имеет иерархическую структуру. Каждый раз при получении пакета коммутатор анализирует фрагмент адреса пакета и направляет пакет в соответствующую линию связи. Коммутатор снабжен таблицей маршрутизации, связывающей конечные адреса или их фрагменты с линиями связи. После считывания заголовка происходит выделение адреса, который используется в качестве индекса таблицы маршрутизации.

Преимущества дейтаграммной сети: 1) – при передаче пакетов в дейтаграммной сети отсутствует фаза установления логического виртуального канала. 2) – дейтаграммная служба более примитивна и допускает большую гибкость. Например, если один из узлов в сети с использованием виртуальных каналов становится перегруженным, то “открытые” виртуальные каналы, проходящие через этот узел, невозможно перестроить. В дейтаграммной сети при перегрузке одного из узлов другие узлы могут перенаправить приходящие пакеты в обход перегруженного узла. 3) – доставка самой дейтаграммы более надежна. При использовании виртуальных каналов, если узел повреждается, все проходящие через него виртуальные каналы также разрушаются.

**Сети с виртуальными каналами** - В сети с виртуальными каналами перед тем, как пакеты начинают идти, создается определенный маршрут следования. Этот маршрут служит для поддержки логического соединения между удаленными станциями.. Поскольку на время логического соединения маршрут строго фиксирован, то такое логическое соединение в некоторой степени аналогично образованию канала в сетях с коммутацией каналов и называется виртуальным каналом. Каждый пакет содержит идентификатор виртуального канала наряду с полем данных. Все узлы по маршруту знают, направлять такие пакеты – никакого решения по маршрутизации теперь эти узлы не принимают. В любое время каждая станция может установить один или несколько виртуальных каналов с другой станцией или станциями..

Главное различие с дейтаграммным подходом и классической маршрутизацией состоит в том, что в сетях с виртуальными каналами узел не принимает решение о отборе маршрута для каждого входящего пакета, а делает это только один раз – на этапе формирования виртуального канала.

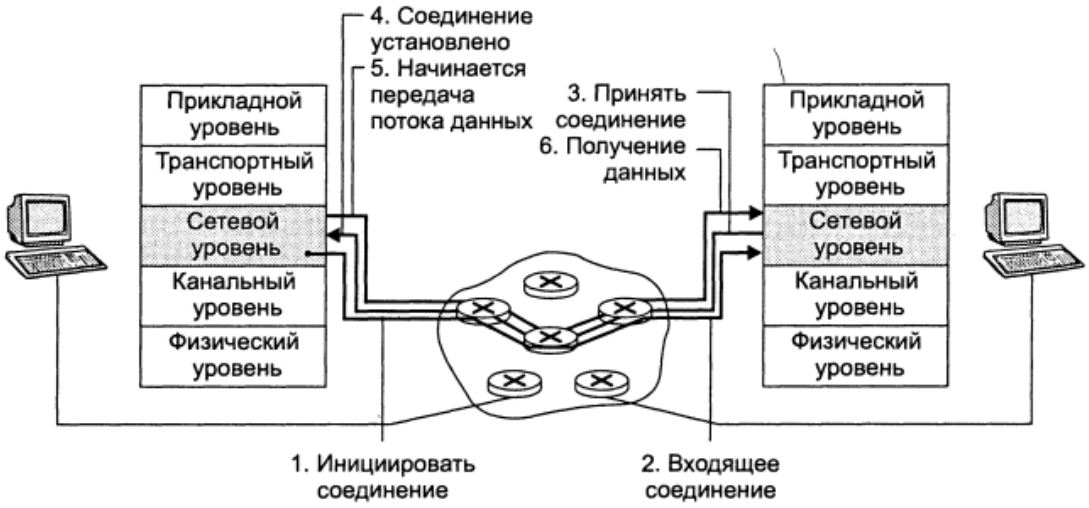



**Рис. 4.2.** Модель обслуживания на основе виртуальных каналов

**27. Marsuutimine - Маршрутизация** ,

В сети хосты соединены с маршрутизаторами , задача которых - определить путь пакетов от хоста к хосту .

Для передачи и обмена пакетов от отправителя к получателю необходимо определить маршрут следования пакетов между хостами сетевой уровень должен определить путь следования пакетов – этим занимается *протокол маршрутизации* сетевого уровня.

Основой любого протокола маршрутизации является *алгоритм маршрутизации* . Задача алгоритма маршрутизации в том , чтобы определить оптимальный путь(путь с минимальной стоимостью) пакетов от маршрутизатора источника к машрутизатору приемнику при заданном количестве маршрутизаторов и линий связи.

Различают глобальный(основан на состоянии линий) и децентрализованный(дистанционно-векторный) алгоритмы маршрутизации

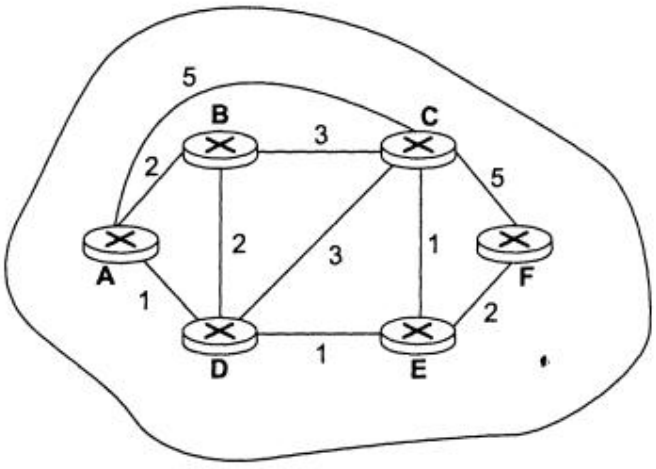



**Рис. 4.4.** Абстрактная модель сети

**28.Link state marsruutimisalgoritm** – алгоритм состояния линий,

Различают 2 основных алгоритма маршрутизации : алгоритм состояния линий и дистанционно-векторный.

Основной принцип алгоритма состояни линий в том , что каждый узел путем широковещательной рассылки отправляет свой идентификатор и оценку всех присоединенных к нему линий всем остальным узлам. Узел знает лишь идентификатор своих соседей и стоимость линий соединенных с ним

Особенность : это централизованный глобальный , не итерационно-распределенный алгоритм и он обладает полной информацией о топологии сети и стоимости линий ( отличие от дист.-векторного)

**29. Distance vector marsruutimisalgoritm**- алгоритм дистанционно-векторной маршрутизации.

Различают 2 основных алгоритма маршрутизации : алгоритм состояния линий и дистанционно-векторный.
Дист.Векторный алгоритм характеризуют следующие свойства : он *распределенный, итерационный и асинхронный* .
Распределенность – потому как каждый узел получает порцию информации от одного или нескольких напрямую соединненых с ним узлов, выполняет вычисления и распределенно отсылает результаты остальным узлам.
Итерационность – алгоритм работает до тех пор , пока соседние узлы не перестанут посылать/принимать информацию( при этом сигнал на завершение работы не требуется, останавливается сам)
Асинхронность – алгоритм не требует , чтоб узлы работали в постоянной взаимосвязи друг с другом
*Таблица расстояний* – это структура данных в алгоритме , содержащаяся на каждом узле. В таблице есть по строке для каждого адресата в сети и по столбцу для каждого ближайшего соседнего узла.
Особенность: форма общения – каждый узел должен знать наименьшую стоимость каждого пути от каждого из его соседей до каждого адресата. Процесс получения информации продолжается до тех пор, пока стоимости не перестанут меняться . Особенности 2 : простота, итерация медленая , некорректные данные могут распостраниться по всей сети
**30. Hierarhiline marsruutimine** – Иерархическая маршрутизация , необходима для решения проблем внутри сети таких как масштабирование и автономность сети.
В таком случае проблемы можно решить, если объединить все маршрутизаторы в отдельные области или автономные системы . Для них необходим алгоритм маршрутизации - *протокол внутренней маршрутизации(RIP ; OSPF) и протокол внешней маршрутизации(BGP) .,* а также промежуточные маршрутизаторы – *шлюзы*, для соединения всех устройств в одну систему .
В виде иерархии : на Сетевом Уровне : Протокол внутренней ⇔Таблица продвижения данных⇔Протокол внешней маршрутизации - дальше - Канальный уровень , Физический уровень

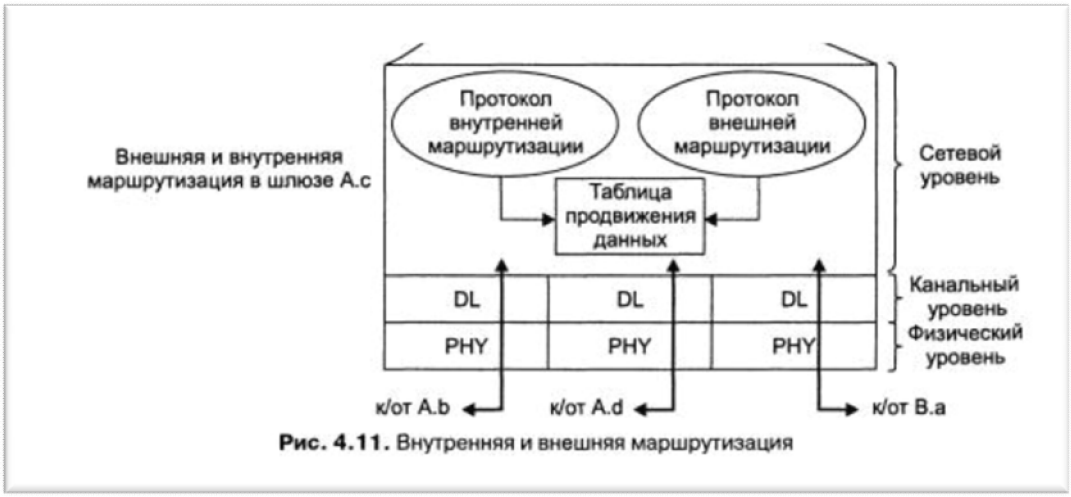
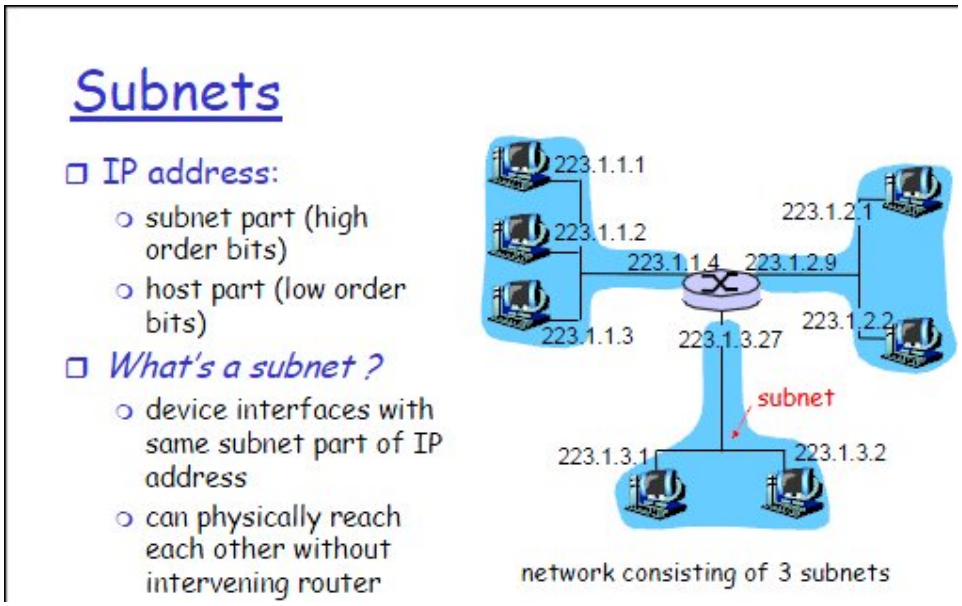


**Рис. 4.11.** Внутренняя и внешняя маршрутизация

**31.** Каждый **IP-адрес** представляет собой 32-разрядное число (четыре байта), поэтому всего может быть 232 IP-адресов. Как правило, эти адреса изображаются в виде четырех десятичных чисел, разделенных точками. Каждое десятичное число соответствует одному байту адреса, например 193.32.216.9. В двоичном виде этот же адрес будет выглядеть так: 11000001 00100000 11011000 00001001
У интерфейсов всех хостов и маршрутизаторов в Интернете должны быть уникальные IP-адреса. Поэтому эти адреса не могут выбираться произвольным образом.
Часть IP-адреса интерфейса определяется ?сетью?, с которой он соединен.




ICANN (Internet Corporation for Assigned Names and Numbers — организация по назначению адресов и имен в Интернете) управляет IP-адресами на основе набора правил, опубликованных в RFC 2050.

Хосту IP-адрес может быть выделен двумя способами.

• *Ручное конфигурирование.* Системный администратор вручную указывает IP-адрес хоста (как правило, в файле).

• *Использование протокола DHCP* (RFC 2131). Протокол DHCP (Dynamic Host Configuration Protocol — протокол динамической конфигурации хоста) позволяет хосту автоматически получить IP-адрес, а также дополнительную информацию, такую как адрес ближайшего маршрутизатора и адрес DNS-сервера.

Итак, теперь мы знаем, что у каждого устройства, поддерживающего протокол IP, должен быть IP-адрес.

**MAC-адрес.** В широковещательных сетях (таких, как сети на основе Ethernet) MAC-адрес позволяет уникально идентифицировать каждый узел сети и доставлять данные только этому узлу. Таким образом, MAC-адреса формируют основу сетей на канальном уровне, которую используют протоколы более высокого (сетевого) уровня. Для преобразования MAC-адресов в адреса сетевого уровня и обратно применяются специальные протоколы (например, ARP и RARP в сетях TCP/IP). **глобально администрируемый MAC-адрес** устройства **глобально уникален** и обычно «зашит» в аппаратуру.

**APR**

При передаче дейтаграмм одновременно используются адреса сетевого уровня (например, IP-адреса Интернета) и адреса канального уровня (то есть LAN-адреса), поэтому возникает необходимость в преобразовании одних адресов в другие. В Интернете эту работу выполняет *протокол ARP* (Address Resolution Protocol — протокол разрешения адресов). У каждого хоста, подключенного к Интернету, и маршрутизатора, соединенного с локальной сетью, есть *ARP-модуль* (RFC 826).

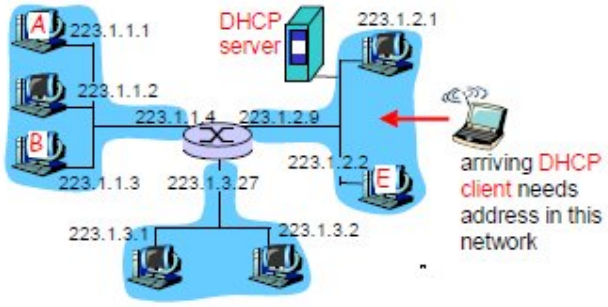
узел с IP-адресом 222.222.222.220 хочет послать IP-дейтаграмму узлу 222.222.222.222. Чтобы выполнить эту задачу, передающий узел должен передать адаптеру не только IP-дейтаграмму, но также LAN-адрес узла 222.222.222.222. Получив IP-дейтаграмму и LAN-адрес узла, адаптер передающего узла формирует кадр канального уровня, содержащий LAN-адрес принимающего узла, и передает кадр в локальную сеть. Но каким образом передающий узел определяет LAN-адрес узла с IP-адресом 222.222.222.222? Он делает это с помощью модуля ARP, передавая ему IP-адрес 222.222.222.222. На это ARP-модуль отвечает соответствующим LAN-адресом узла, то есть адресом 49-BD-D2-C7-56-2A.

Итак, ARP-модуль преобразует IP-адрес в LAN-адрес узла. Во многом это аналогично

системе DNS преобразующей имена хостов в IP-адреса. Однако важное различие между этими двумя схемами преобразования адресов заключается в том, что DNS преобразует имена хостов в IP-адреса во всем Интернете, тогда как протокол ARP занимается только IP-адресами в пределах одной локальной сети.

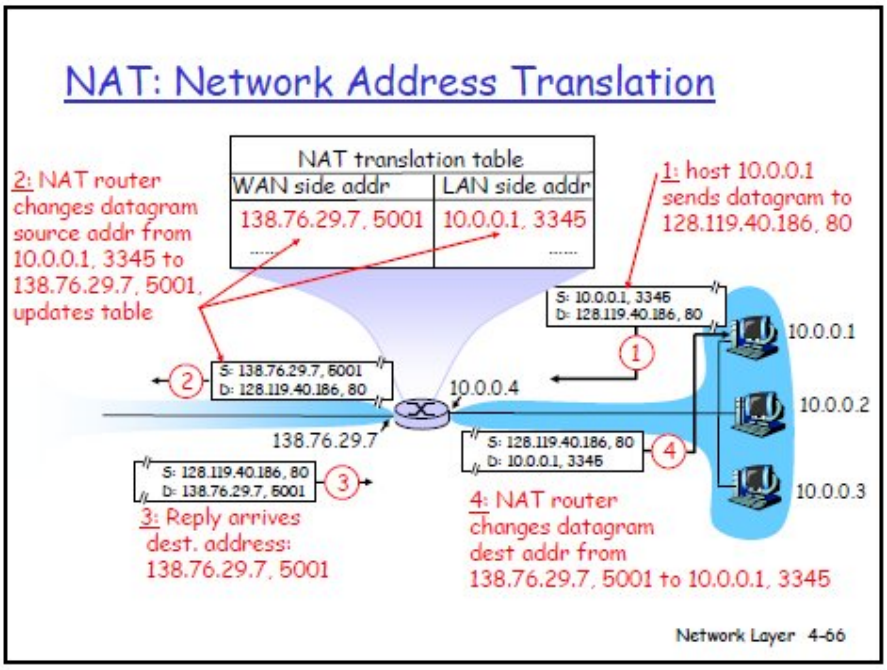
32. **Протокол DHCP** представляет собой протокол для связи клиента с сервером. Клиентом, как правило, выступает только что подключившийся к сети новый хост, желающий получить информацию о конфигурации сети и собственный IP-адрес. В простейшем случае в каждой сети (имеется в виду IP-сеть, см. рис. 4.15) есть свой DHCP-сервер. Если в сети нет сервера, необходим промежуточный DHCP-агент (как правило, маршрутизатор), которому известен адрес DHCP-сервера для этой сети. На рис. 4.25 показан DHCP-сервер, соединенный с сетью 233.1.2/24, и маршрутизатор, выступающий в роли промежуточного DHCP-агента для новых клиентов, присоединяющихся к сетям 223.1.1/24 и 223.1.3/24.
При появлении в сети нового клиента протокол DHCP запускает процесс из четырех этапов.
1. *Обнаружение DHCP-сервера.* Первым делом только что подключившийся хост должен найти DHCP-сервер, с которым можно взаимодействовать. Это делается при помощи сообщения о *поиске DHCP-сереерау* которое клиент посылает UDP-дейтаграмме через порт 67. Но кому он должен посылать эту дейтаграмму? Хосту не известен даже IP-адрес сети, к которой он подключился, а уж тем более адрес DHCP-сервера, обслуживающего эту сеть. Поэтому DHCP-клиент в поле адреса получателя дейтаграммы указывает широковещательный адрес 255.255.255.255, а себя (отправителя) обозначает как 0.0.0.0. Сообщение о поиске DHCP-сервера будет получено всеми машинами сети, включая все DHCP-серверы (и/или промежуточные агенты). В этом сообщении содержится идентификатор транзакции, позволяющий соотнести последующие ответы с запросом обнаружения DHCP-сервера.
2. *Предложение DHCP-сервера.* Получив сообщение о поиске DHCP-сервера DHCP-сервер отвечает клиенту сообщением с *DHCP-предложением.* Поскольку в сети могут находиться несколько DHCP-серверов, может случиться, что клиенту придется выбирать из нескольких предложений. Каждое DHCP-предложение содержит идентификатор транзакции полученного сообщения о поиске DHCP-сервера, предлагаемый IP-адрес клиента, маску сети и срок действия IP-адреса, называемый обычно *сроком аренды IP-адреса.* Как правило, сервер предоставляет новому хосту IP-адрес на срок от нескольких часов до нескольких дней [132]. Затем прибывшему клиенту посылается кадр канального уровня с IP-дейтаграммой, в которой содержится UDP-сегмент с DHCP-предложением (прекрасная иллюстрация многоуровневой иерархии протоколов). Как кадр канального уровня посылается клиенту, мы рассмотрим в главе 5, посвященной канальному уровню.
3. *DHCP-запрос.* Новый клиент выбирает одно из полученных им предложений от серверов и отвечает на выбранное им предложение DHCP-запросом, повторяя в нем конфигурационные параметры.
4. *DHCP-подтверждение.* Сервер отвечает на DHCP-запрос DHCP-подтверждением, подтверждая запрашиваемые параметры.

## DHCP client-server scenario



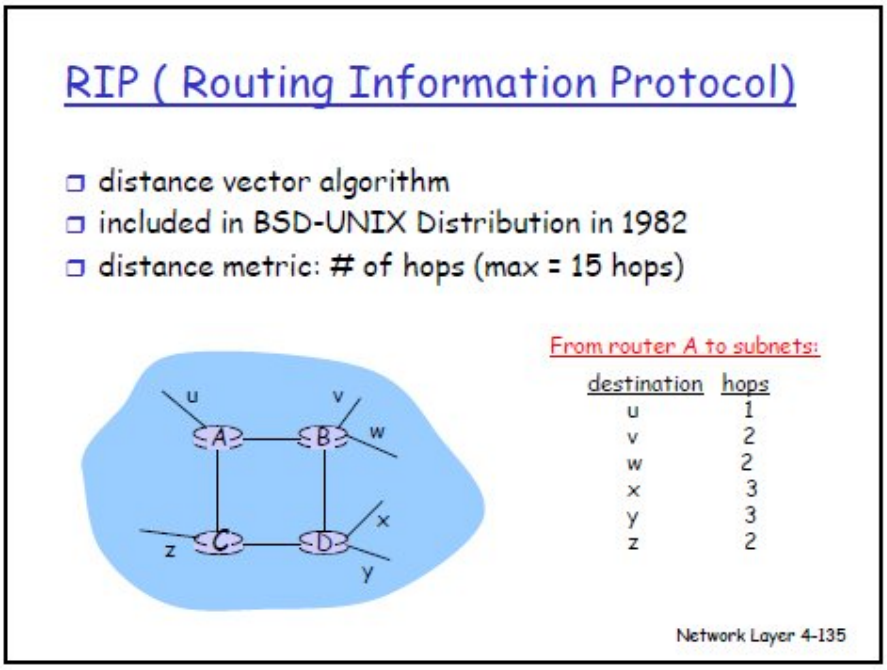

**33**. В случае роста сети (например, у детей в доме помимо настольных персональных компьютеров появились карманные компьютеры, телефоны с поддержкой протокола IP или сетевые игровые приставки) такому офису нужно было бы выделить блок адресов большего размера. А что, если Интернет-провайдер уже распределил все соседние участки адресного пространства? И что нужно знать об управлении IP-адресами типичному домовладельцу? К счастью, существует более простой метод выделения адресов, нашедший широкое применение в подобных ситуациях. Этим методом является (RFC 2663, RFC 3022) так называемая *трансляция сетевых адресов* (Network Address Translation, NAT).
Рисунок 4.27 иллюстрирует работу маршрутизатора, поддерживающего трансляцию сетевых адресов (NAT-маршрутизатора). У этого маршрутизатора есть интерфейс, представляющий собой часть домашней сети (на правой стороне рисунка). Адресация в пределах домашней сети полностью аналогична тому, что мы уже видели — у всех четырех интерфейсов домашней сети один и тот же сетевой адрес 10.0.0/24. От обсуждавшейся выше ситуации данный пример отличается взаимоотношением между домашним маршрутизатором и Интернет-провайдером.
На NAT-маршрутизаторе не работает протокол внутренней маршрутизации для связи с маршрутизатором Интернет-провайдера. В самом деле, для внешнего мира NAT-маршрутизатор выглядит как единое устройство с одним IP-адресом — весь трафик, исходящий из домашнего маршрутизатора в Интернет, помечается IP-адресом отправителя 138.76.29.7, а весь прибывающий из Интернета трафик должен иметь IP-адрес получателя 138.76.29.7. Таким образом, NAT-маршрутизатор скрывает от внешнего мира детали домашней сети. (Обычно маршрутизатор получает свой адрес у DHCP-сервера Интернет-провайдера и сам выступает в роли DHCP-сервера для компьютеров своей домашней сети.)
Теперь вам, вероятно, непонятно, как маршрутизатор определяет, кому предназначается прибывшая из Интернета дейтаграмма, если все дейтаграммы маркируются одним и тем же IP-адресом получателя. Дело в том, что маршрутизатор различает прибывающие дейтаграммы по номерам портов, которые с помощью таблицы трансляции адресов (NAT-таблицы) преобразуются во внутренние адреса и номера портов.
Вернемся к примеру на рисунке. Предположим, пользователь, сидящий за хостом 10.0.0.1, запрашивает web-страницу с некоторого web-сервера (порт 80) с IP-адресом 128.119.40.186. Хост выбирает себе произвольный номер порта отправителя 3345 и отправляет дейтаграмму в локальную сеть. NAT-маршрутизатор получает дейтаграмму, генерирует новый номер порта 5001, которым заменяет оригинальный номер порта, а IP-адрес отправителя заменяет своим IP-адресом 138.76.29.7.

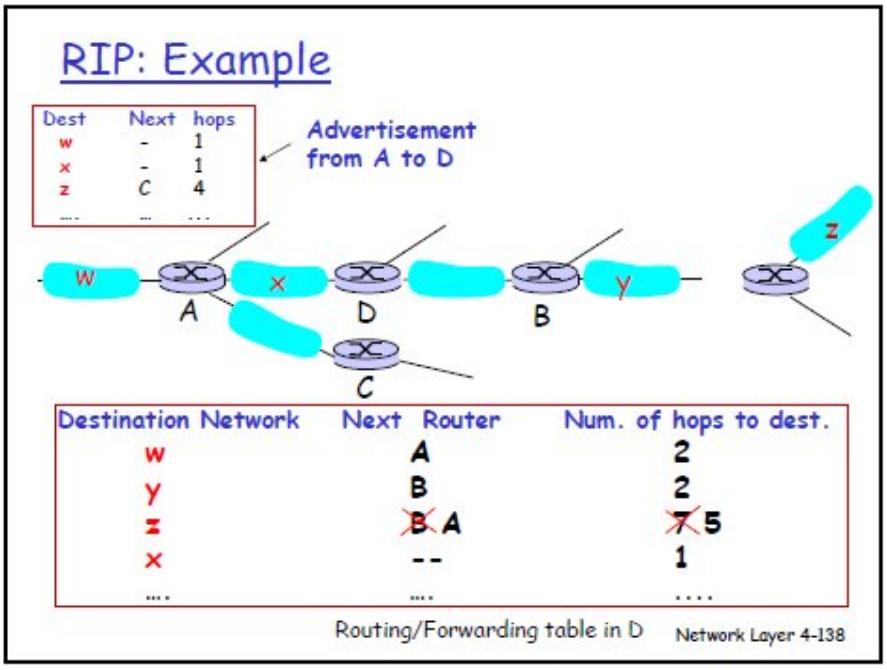
Генерируя новый номер порта, NAT-маршрутизатор может выбирать любое число, которого нет в NAT-таблице. (Обратите внимание, что поле номера порта состоит из 16 бит, поэтому протокол NAT может поддерживать более 60 000 одновременных соединений с единственным IP-адресом маршрутизатора!) Web-сервер, не имея представления о том, что полученная им дейтаграмма с HTTP-запросом была обработана NAT-маршрутизатором, отвечает дейтаграммой, в поле адреса получателя которой указан IP-адрес NAT-маршрутизатора, а в поле номера порта— порт 5001. Получив эту дейтаграмму, NAT-маршрутизатор по указанным в дейтаграмме IP-адресу получателя и номеру порта находит в NAT-таблице соответствующий IP-адрес (10.0.0.1) и номер порта (3345) браузера домашней сети.
Затем маршрутизатор заменяет оба этих поля найденными в таблице и переправляет дейтаграмму в домашнюю сеть.




34. **Протокол RIP** (Routing Information Protocol — протокол маршрутной информации) был одним из первых протоколов внутренней маршрутизации, применявшихся в Интернете; он и в наши дни по-прежнему популярен. Широкое распространение протокола RIP было во многом вызвано тем, что он был включен в версию 1982 года операционной системы Berkeley UNIX, поддерживающей стек протоколов TCP/IP.
Максимальная стоимость пути ограничена значением 15, таким образом, диаметр автономной системы, поддерживаемой протоколом RIP, не может превышать 15 ретрансляционных участков.
Вспомним также, что в дистанционно-векторных протоколах соседние маршрутизаторы обмениваются друг с другом информацией о маршрутах. В протоколе RIP обмен новыми сведениями между соседними маршрутизаторами происходит приблизительно через каждые 30 с, для чего используются так называемые *ответные RIP-сообщения* (RIP response messages). Ответное RIP-сообщение, посылаемое маршрутизатором или хостом, содержит список, в котором указаны до 25 сетей-адресатов в пределах автономной системы, а также расстояния до каждой из этих сетей от отправителя. Ответные RIP-сообщения также иногда называют RIP-объявлениями.
Рассмотрим теперь некоторые аспекты реализации протокола RIP. Если маршрутизатор не получает пакетов от своего соседа в течение 180 с, он решает, что данный сосед более недоступен. Это может означать, что сосед вышел из строя или выключен либо вышла из строя связывавшая их линия связи. Когда такое происходит, протокол RIP изменяет локальную таблицу продвижения данных, а затем распространяет эту информацию, рассылая объявления соседним маршрутизаторам (тем, которые

все еще доступны). Маршрутизатор может также запросить у соседа информацию о стоимости маршрута от
него до заданного адресата при помощи RIP-запроса. Маршрутизаторы пересылают друг другу RIP-запросы и RIP-ответы в UDP-пакетах через порт номер 520.
UDP-пакет переносится между маршрутизаторами в стандартном IP-пакете. Тот факт, что протокол RIP пользуется протоколом транспортного уровня (UDP) поверх протокола сетевого уровня (IP), может показаться излишне сложным (так оно и есть!). Чтобы прояснить данный вопрос, необходимо несколько углубиться в реализацию протокола RIP.
Поскольку протокол RIP реализован как процесс прикладного уровня (хотя и весьма специфический, например, он способен управлять таблицами продвижения данных, находящихся в ядре операционной системы UNIX), он может отправлять и получать сообщения через стандартный сокет и использовать стандартный транспортный протокол.



**Протокол OSPF**

Как и RIP, протокол OSPF (Open Shortest Path First —открытый протокол выбора кратчайшего маршрута) используется для маршрутизации внутри автономной системы.

Протокол OSPF считается преемником протокола RIP и обладает рядом дополнительных функций. Однако по своей сути протокол OSPF представляет собой протокол, основанный на учете состоянии линий и использующий метод лавинной рассылки для распространения информации о состоянии линий, а также алгоритм определения пути наименьшей стоимости Дейкстры. Маршрутизатор, работающий по протоколу OSPF, формирует полную топологическую карту (направленный граф) всей автономной системы. Затем маршрутизатор локально запускает алгоритм определения кратчайшего пути Дейкстры, чтобы найти дерево кратчайших путей ко всем сетям автономной системы. Далее из этого дерева кратчайших путей формируется таблица продвижения данных маршрутизатора. Стоимости линий настраиваются сетевым администратором. Администратор может установить стоимости всех линий равными 1, в результате путь наименьшей стоимости совпадет с кратчайшим путем, или установить весовой коэффициент каждой линии обратно пропорциональным пропускной способности линии, чтобы маршрутизаторы старались избегать линий с низкой пропускной способностью. Протокол OSPF не занимается определением стоимости линий (это работа сетевого администратора), а лишь предоставляет механизмы (протокол) определения пути наименьшей стоимости для заданного набора стоимостей линий.

Маршрутизатор, работающий по протоколу OSPF, путем широковещательной рассылки переправляет информацию о маршрутах всем маршрутизаторам автономной системы, а не только соседним.

Маршрутизатор рассылает всем информацию о состоянии линий при каждом изменении состояния какой-либо из линий (например, при изменении стоимости или включении/отключении). Он также рассылает информацию о состоянии линий периодически (по меньшей мере, раз в 30 мин), даже если состояние линии не изменилось. Таким образом, протокол OSPF должен сам заниматься такими вопросами, как надежность передачи сообщений и широковещательная рассылка информации о состоянии линий. Протокол OSPF также проверяет работоспособность линий (при помощи сообщения HELLO, посылаемого соседу) и позволяет маршрутизатору OSPF получать информацию из базы данных соседнего маршрутизатора о состоянии линий всей сети.

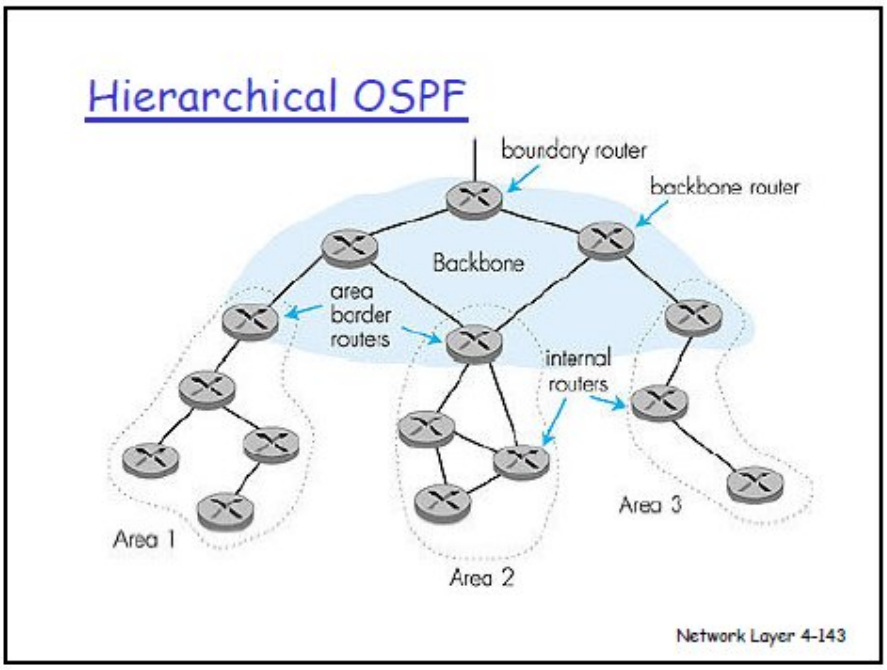
Ниже перечислены достоинства протокола OSPF.

•*Безопасность*. Весь обмен данными между OSPF-маршрутизаторами (например, новые сведения о состоянии линий) проходит процедуру аутентификации. Это означает, что только доверенные маршрутизаторы могут принимать участие в работе протокола OSPF в пределах домена, не позволяя, таким образом, злоумышленникам (или студентам, применяющим полученные знания для развлечений) вставлять в таблицы маршрутизации некорректную информацию.

•*Несколько путей с одинаковой стоимостью.* Когда у нескольких маршрутов к одному адресату оказывается одинаковая стоимость, протокол OSPF позволяет использовать все (нет проблемы выбора одного из этих маршрутов).

•*Интегрированная поддержка многоадресной и одноадресной маршрутизации.* Протокол MOSPF (Multicast OSPF —протокол OSPF для групповой рассыл-

напрямую присоединены.




**BGP**

Следует также отметить, что протокол BGP направляет пакеты сетям-адресатам (имеются в виду IP-сети), а не конкретным маршрутизаторам или хостам.

Как только дейтаграмма достигает сети-адресата, ее пересылкой к конечному получателю начинает заниматься протокол внутренней маршрутизации. Протокол BGP распознает автономные системы (RFC 1930) по глобально уникальным *номерам автономных систем* (Autonomous System Number, ASN). На практике не у каждой автономной системы есть номер. В частности, так называемая тупиковая автономная система (автономная система-заглушка), переносящая только трафик, для которого он является отправителем или получателем, как правило, не имеет номера.

В основе протокола BGP лежат объявления о маршрутах. Объявление о маршрутах посылается одним равноправным BGP-узлом другому равноправному BGP-узлу по соединению «точка-точка». Объявление состоит из адреса сети (например, 128.119.40/24) и набора атрибутов, ассоциированных с маршрутом к этой сети. Двумя наиболее важными атрибутами являются атрибут маршрута (явный список всех автономных систем на пути к указанной сети-адресату) и идентификатор следующего маршрутизатора на пути к указанной сети-адресату.

Работа протокола BGP в основном состоит из трех действий с объявлениями.

1. *Получение и фильтрация объявлений о маршрутах от напрямую присоединенных соседей.* BGP-маршрутизатор получает объявления о маршрутах от равноправного BGP-узла. Мы можем рассматривать полученное BGP-объявление о маршруте как обещание. Равноправный BGP-узел, посылающий объявление о маршруте к некоторой автономной системе, обещает, что, если соседняя автономная система пошлет ему дейтаграмму, адресованную указанной автономной системе, тогда он сможет переправить эту дейтаграмму далее по пути к адресату. BGP-маршрутизатор может также отфильтровывать (отбрасывать) полученные объявления о маршрутах. Например, BGP-маршрутизатор будет игнорировать объявления, содержащие номер его собственной автономной системы, указанный в качестве автономной системы-получателя, так как использование такого маршрута привело бы к появлению маршрутной петли. Поскольку маршрут указывается полностью, сетевой администратор может в значительной степени контролировать маршруты дейтаграмм его сети. Например, в автономной системе Hatfield.net можно реализовать политику, заключающуюся в том, чтобы трафик из автономной системы Hatfield.net не попадал в автономную

систему McCoy.net (семейства Hatfield и McCoy, живущие в США, представляют собой два знаменитых враждующих клана).

2. *Выбор маршрута.* BGP-маршрутизатор может получить несколько объявлений о маршрутах к одной и той же автономной системе-адресату. В этом случае он должен выбрать из объявленных маршрутов один. Данные об автономной системе-адресате и следующем ретрансляционном участке для выбранного маршрута должны быть помещены в таблицы продвижения данных (как, например, было показано на рис. 4.11). BGP-маршрутизатору могут быть известны несколько маршрутов к одному адресату, но в таблице продвижения данных, как правило, указывается лишь один маршрутизатор по направлению к данному адресату.

Но как BGP-маршрутизатор выбирает из нескольких маршрутов один? В протоколе BGP проводится четкое разграничение между *механизмом маршрутизации* и *политикой маршрутизации.* В частности, протокол BGP *не* определяет, как автономная система должна выбирать один маршрут из нескольких предложенных. Это *политическое* решение, оставляемое на усмотрение сетевого администратора автономной системы. Администратор сети может указать так называемые локальные предпочтения, например, что при наличии выбора маршрутизация через соседнюю автономную систему A всегда предпочтительнее маршрутизации через соседнюю автономную систему B. В отсутствие локальных предпочтений, как правило, выбирается кратчайший маршрут, то есть маршрут, состоящий из наименьшего количества автономных систем. Обсуждение алгоритмов выбора оптимального маршрута содержится в [77].

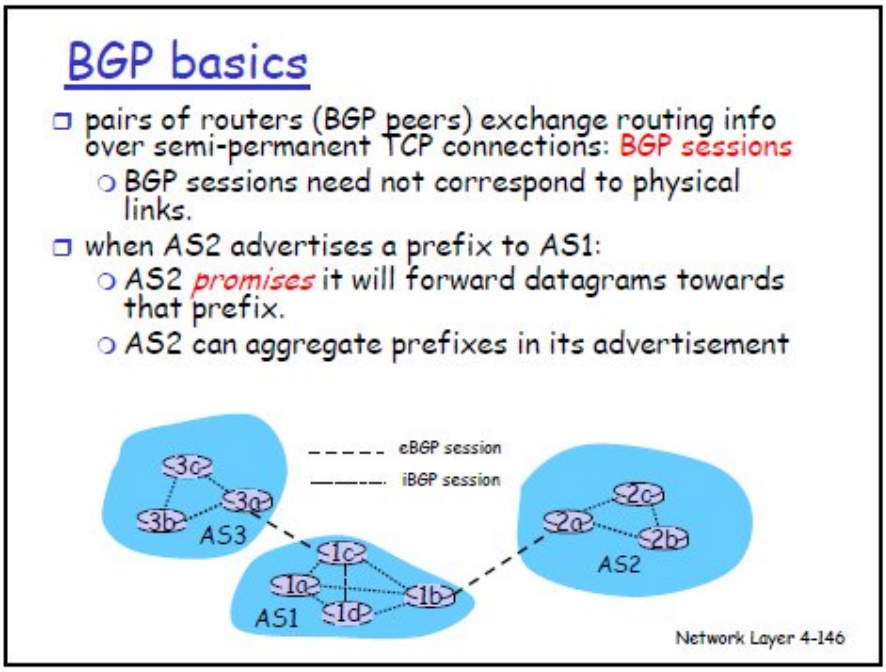
3. *Передача объявлений о маршрутах своим соседям.* BGP-маршрутизатор получает объявления о маршрутах от своих соседей, а также передает объявления о маршрутах своим соседям. Протокол BGP предоставляет механизм для таких объявлений, но не политику. В результате администратор сети получает значительный уровень контроля над трафиком, поступающим в его сеть. Применительно к нашему примеру с семействами Hatfield и McCoy, клан Hatfield легко может запретить трафику клана McCoy проходить через свою сеть. Например, если сеть клана McCoy расположена по соседству с сетью клана Hatfield, клан Hatfield может просто не информировать о маршрутах к сети клана McCoy, проходящих через сеть клана Hatfield. Однако эффективность ограничения трафика путем контролирования объявлений автономной системы о маршрутах может быть не полной. Например, если сеть Петровых располагается между сетями семейств Hatfield и McCoy, и клан Hatfield информируют о маршрутах к сети Петровых, проходящих через сеть клана Hatfield, тогда клан Hatfield не может запретить (с помощью протокола BGP) Петровым сообщить об этих маршрутах клану McCoy.

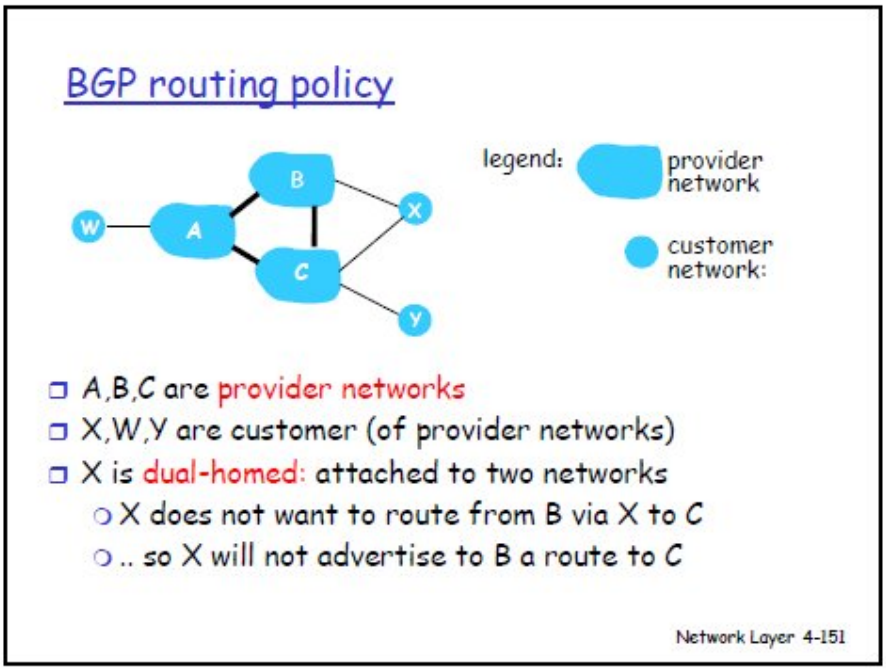
В протоколе BGP определены четыре типа сообщений: OPEN, UPDATE, KEEPALIVE и NOTIFICATION.

•*OPEN* (открытие). Когда BGP-маршрутизатор желает установить контакт с равноправным BGP-узлом (например, после того как сам маршрутизатор был загружен или была включена линия связи), сообщение OPEN пересылается равноправному BGP-узлу. Сообщение OPEN позволяет BGP-маршрутизатору идентифицировать и аутентифицировать себя, а также синхронизироваться. Если получивший сообщение OPEN равноправный BGP-узел считает это для себя приемлемым, он отвечает на него сообщением KEEPALIVE.

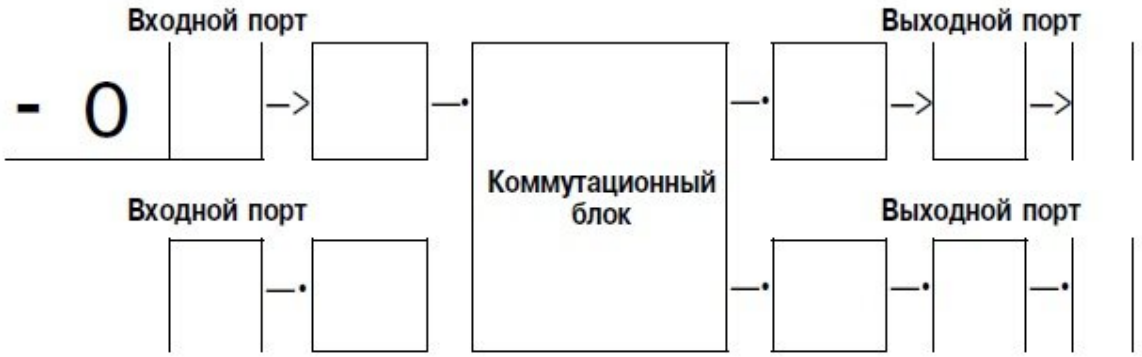
•*UPDATE* (обновление). BGP-маршрутизатор использует сообщение UPDATE, чтобы объявить равноправному BGP-узлу о маршруте к определенному адресату. Сообщение UPDATE может также использоваться для того, чтобы аннулировать объявленный ранее маршрут (то есть сообщить равноправному BGP-узлу, что объявленный ранее маршрут более не действителен). BGP-маршрут

считается действительным до тех пор, пока он не будет явно аннулирован.
•*KEEPALIVE* (дежурное сообщение). Это BGP-сообщение позволяет известить равноправный BGP-узел о том, что отправитель сообщения продолжает нормальную деятельность и у него нет другой информации для передачи. Это сообщение также служит подтверждением полученного сообщения OPEN.
•*NOTIFICATION*(уведомление). Это BGP-сообщение используется для информирования равноправного BGP-узла об обнаруженной ошибке (например, в переданном ранее BGP-сообщении) или о том, что отправитель собирается завершить BGP-сеанс.




35.




• *Входные порты.* Входной порт выполняет несколько функций. Он выполняет функции физического уровня (самый левый прямоугольник входного порта и самый правый прямоугольник выходного порта), завершая входную физическую линию маршрутизатора. Он также осуществляет функции канального уровня (средние прямоугольники входного и выходного портов), необходимые для взаимодействия с функциями канального уровня на другой стороне линии связи. Еще он выполняет функции поиска и продвижения данных (самый правый прямоугольник входного порта и самый левый прямоугольник выходного порта), так что пакет, переправленный в коммутационный блок маршрутизатора, на выходе из него появляется из того порта, из которого следует. Управляющие пакеты (например, пакеты, содержащие информацию протокола RIP, OSPF или BGP) продвигаются из входного порта в маршрутный процессор. На практике несколько портов часто объединяют на одной канальной карте маршрутизатора.

• *Коммутационный блок.* Коммутационный блок соединяет входные порты маршрутизатора с его выходными портами. Коммутационный блок целиком располагается внутри маршрутизатора — сеть внутри сетевого маршрутизатора!

• *Коммутация с использованием памяти.* Простейшие ранние маршрутизаторы часто представляли собой традиционные компьютеры, в которых коммутация между входными и выходными портами осуществлялась под непосредственным управлением центрального процессора (игавшего роль маршрутного процессора). Входные и выходные порты функционировали как традиционные устройства ввода-вывода в операционной системе. Получив пакет, входной порт сначала сигнализирует маршрутному процессору прерыванием. Затем пакет копируется из входного порта в память процессора. После этого маршрутный процессор извлекает из заголовка пакета адрес получателя, ищет соответствующий выходной порт в таблице продвижения данных и копирует пакет в буфер выходного порта. Обратите внимание, если пропускная способность памяти позволяет записать или прочитать $B$ пакетов в секунду, тогда суммарная пропускная способность коммутатора (полная скорость, с которой пакеты переносятся от входных портов к выходным портам) не может превышать $B/2$.

Многие современные маршрутизаторы также используют память для коммутации пакетов. Однако их главное отличие от ранних маршрутизаторов заключается в том, что поиск адреса получателя и хранение (коммутация) пакетов в соответствующей области памяти выполняется процессорами на карте входной линии.

• *Коммутация с использованием шины.* При таком подходе входные порты переносят пакеты напрямую в выходные порты по общей шине без вмешательства маршрутного процессора (обратите внимание, что в случае коммутации через

память пакет также должен пересечь системную шину, соединяющуюся с памятью). Хотя маршрутный процессор не участвует в переносе пакета по шине, поскольку шина используется коллективно, в каждый момент времени по шине может двигаться только один пакет. Пакет, поступивший на входной порт в тот момент, когда шина занята переносом другого пакета, блокируется и устанавливается в очередь входного порта. Поскольку каждый пакет должен пересечь единственную шину, пропускная способность подобного маршрутизатора ограничивается скоростью шины.

• *Коммутация с использованием соединительной сети.* Один из способов преодолеть ограничения пропускной способности одной общей шины заключается в использовании более сложной соединительной сети, подобной сети, связывающей процессоры в мультипроцессорных системах. Прибывающий на входной порт пакет перемещается по горизонтальной шине до пересечения с вертикальной шиной, ведущей к нужному выходному порту. Если ведущая к выходному порту вертикальная шина свободна, пакет переносится в выходной порт. Если в данный момент вертикальная шина уже используется для передачи пакета в тот же самый выходной порт из другого входного порта, тогда пакет блокируется и ставится в очередь входного порта.

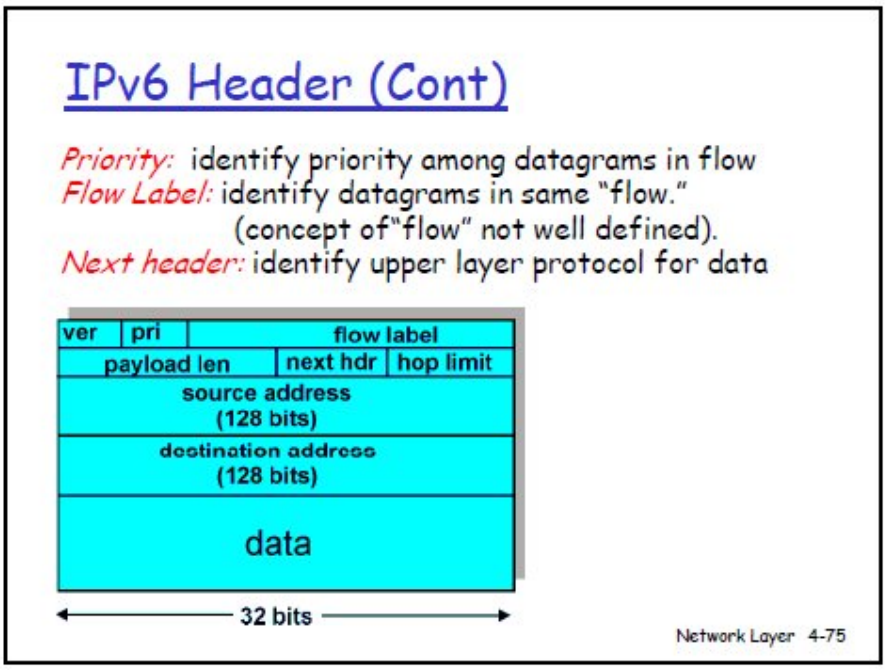
• *Выходные порты.* Выходной порт хранит пакеты, переправленные ему через коммутационный блок, а затем передает пакеты по выходной линии. Таким образом, выходной порт осуществляет функции физического и канального уровней, обратные функциям входного порта. В случае двунаправленной линии связи (то есть когда линия передает данные в оба направления) выходной порт линии связи, как правило, составляет пару с входным портом этой линии, располагаясь на той же самой карте канала.

• *Маршрутный процессор.* Маршрутный процессор выполняет функции протоколов маршрутизации, обрабатывает информацию о маршрутах, а также выполняет функции управления сетью в маршрутизаторе.

36. **Ipv4 ja Ipv6**

В начале 90-х годов группа IETF начала работы над очередной версией протокола IP. В первую очередь, необходимость подобных работ была вызвана осознанием того факта, что 32-разрядное адресное пространство протокола IP исчерпывается, а новые сети и IP-узлы присоединяются к Интернету с захватывающей дух скоростью. В ответ на потребность в большем адресном пространстве была разработана версия IPv6 протокола IP. При этом разработчики воспользовались удобным случаем, чтобы, используя накопленный опыт эксплуатации протокола IPv4, в чем-то его исправить и дополнить.

**Формат дейтаграммы протокола IPv6**




По новому формату можно судить о наиболее существенных изменениях в протоколе IP.

• *Расширенные возможности адресации.* В дейтаграмме протокола IPv6 размер IP-адреса увеличен с 32 до 128 бит. Это гарантирует, что адресного пространства будет хватать всем и всегда. Теперь можно дать IP-адрес каждой песчинке на планете. В дополнение к индивидуальным и групповым адресам в протоколе IPv6 появился новый тип адресов, называемых *адресами свободной рассылки* (anycast addresses), позволяющих пересылать дейтаграмму любому члену группы хостов. (Это может служить, например, для передачи сообщения HTTP GET ближайшему из нескольких зеркальных сайтов, содержащих данный документ.)

• *Упрощенный 40-разрядный заголовок.* Несколько полей протокола IPv4 были опущены или сделаны необязательными, о чем будет сказано далее. Получившийся в результате 40-разрядный заголовок фиксированной длины обеспечивает ускоренную обработку IP-дейтаграммы. Новый способ кодирования необязательных полей обеспечивает их более гибкую обработку.

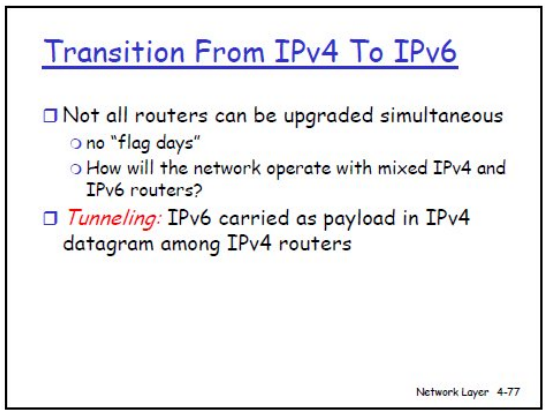
*Метка потока и приоритет.* Определение потока в протоколе IPv6 довольно расплывчато. В RFC 1752 и RFC 2460 утверждается, что поле метки потока позволяет маркировать пакеты, для которых отправителю требуется специальная обработка, например, обслуживание с отличным от предоставляемого по умолчанию уровнем качества или обслуживание в реальном времени. С одной стороны, обрабатываться в качестве потока могут, например, транслируемые аудио- или видеоданные, трафик высокоприоритетного пользователя (например, платящего за более качественное обслуживание своего трафика). С другой стороны, традиционные приложения, такие как приложения передачи файлов и электронной почты, в качестве потока могут не обрабатываться. Ясно лишь то, что разработчики протокола IPv6 предвидят необходимость в дифференцировании потоков, несмотря на то что точное понятие потока еще не определено.

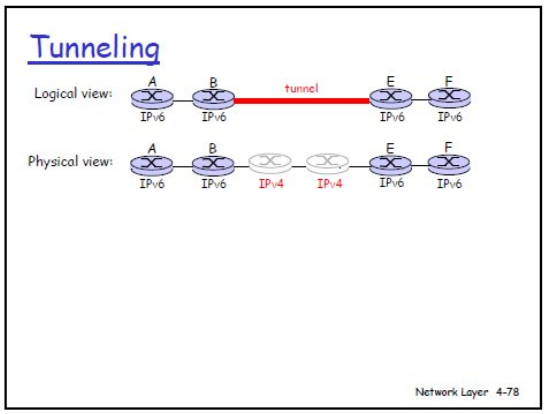
В заголовке IPv6 также есть восьмиразрядное поле класса трафика. Подобно полю TOS (Type Of Service — тип службы) протокола IPv4 это поле может использоваться для предоставления приоритета определенным пакетам потока, а также для предоставления приоритета дейтаграммам определенных приложений (например, ICMP-пакетам) по сравнению с дейтаграммами других приложений (например, пакетам с сетевыми новостями).

Ниже перечислены поля IPve-дейтаграммы.

• *Версия.* Это 4-разрядное поле идентифицирует номер версии протокола IP и для протокола IPv6 содержит значение 6. Обратите внимание, если поместить в это поле значение 4, то мы не получим корректную дейтаграмму протокола IPv4 (если бы это было так, жизнь была бы намного проще).
• *Класс трафика.* Это 8-разрядное поле отчасти напоминает поле TOS протокола IPv4.
• *Метка потока.* Как упоминалось выше, это 20-разрядное поле используется для идентификации "потока" дейтаграмм.
• *Длина полезной нагрузки.* Это 16-разрядное поле обрабатывается как целое число без знака и содержит количество байтов в 1Ру6-дейтаграмме, следующих за 40-разрядным заголовком дейтаграммы.
• *Следующий заголовок.* Это поле идентифицирует протокол, которому доставляется содержимое (поле данных) дейтаграммы (например, TCP или UDP). Для этого поля используются те же значения, что и для поля протокола в IPv4-**заго-**ловке.
• *Ограничение на число ретрансляционных участков.* Содержимое этого поля уменьшается на единицу на каждом маршрутизаторе, через который проходит дейтаграмма. Когда содержимое этого поля достигает нуля, дейтаграмма уничтожается.
• *Адреса отправителя и получателя.* Различные форматы 128-разрядных IPv6-адресов описаны в RFC 2373.
• *Данные.* Это полезная нагрузка 1Ру6-дейтаграммы. Когда дейтаграмма достигает пункта назначения, полезная нагрузка извлекается из нее и передается протоколу, указанному в поле следующего заголовка.

**Переход с IPv4 на IPv6**




Теперь, когда мы обсудили технические детали протокола IPv6, рассмотрим довольно практический вопрос: как перевести на IPv6 Интернет, функционирующий по протоколу IPv4? Проблема заключается в том, что, хотя новые IPv6-системы можно сделать обратно совместимыми, то есть реализовать в них поддержку дейтаграмм старого формата, уже работающие IPv4-системы не смогут обрабатывать IPv6-дейтаграммы.
Одним из решений могло бы стать объявление о некой дате перехода с IPv4 на IPv6. Последняя подобная глобальная смена технологий (переход с протокола NCP на TCP для предоставления надежной транспортной службы) произошла почти 20 лет назад (RFC 801). Однако уже довольно давно, еще когда Интернет был крошечный и управлялся небольшим количеством "волшебников", стало ясно, что подобный "день икс" невозможен. Сегодня такая операция затронула бы сотни миллионов машин и миллионы сетевых администраторов и пользователей. В RFC 2893 описываются два подхода (которые можно использовать вместе или по отдельности) постепенного ввода IP**V6-ХОСТОВ** в "мир" IPv4 (с долгосрочной целью, разумеется, полного перехода 1Ру4-узлов на протокол IPv6).
Вероятно, наиболее простой способ внедрить в Интернет узлы, поддерживающие

протокол IPv6, представляет собой метод *двойного стека,* при котором IPve-узлы обладают полной поддержкой протокола IPv4.
Альтернативный метод, называется *туннелированием.*
В основе туннелирования лежит следующая идея. Предположим, два IPve-узла хотят обменяться IPve-дейтаграммами, но их разделяют iPv4-маршрутизаторы. Метод туннелирования заключается в том, что передающий IPve-узел (например, узел В) помещает IPve-дейтаграмму целиком в поле данных дейтаграммы формата IPv4. Затем эта IPv4-дейтаграмма, адресованная получающему IPve-узлу на другой стороне туннеля (например, узлу Е), передается первому узлу туннеля (например, узлу С). Промежуточные 1Ру4-маршрутизаторы передают эту дейтаграмму друг другу, как обычную дейтаграмму, не зная о том, что она содержит полную дейтаграмму формата IPv6. Наконец, принимающий IPv6-узел на другой стороне туннеля получает IPv4-дейтаграмму, определяет, что она содержит дейтаграмму формата IPv6, извлекает IPv6-дейтаграмму и переправляет ее дальше, как если бы он получил ее по сети от непосредственного соседа.
Один важный урок, который мы должны усвоить из знакомства с протоколом IPv6, состоит в том, что сменить сетевой протокол крайне сложно. С начала 90-х годов множество новых сетевых протоколов провозглашались следующим революционным прорывом в Интернете, но большая часть их просуществовала очень недолго. В самом деле, введение нового протокола в сетевой уровень подобно замене фундамента у здания —это сложно сделать, не развалив весь дом или, по меньшей мере, не переселив временно его жильцов. С другой стороны, Интернет был свидетелем быстрого развертывания новых протоколов прикладного уровня. Классическими примерами являются web, передача сообщений в реальном времени, коллективное использование файлов одноранговыми (равноправными) узлами. К другим примерам относятся передача потокового аудио и видео, чат. Внедрение новых протоколов прикладного уровня подобно перекраске здания —это относительно легко сделать, и, если вы выберете привлекательный цвет, соседи быстро сделают то же самое. Итак, в будущем мы можем ожидать изменений на сетевом уровне стека протоколов Интернета, но эти изменения, скорее всего, будут происходить значительно медленнее изменений на прикладном уровне.

37. **Vigade avastamine ja parandamine, CRC**

На канальном уровне часто предоставляются услуги по обнаружению и исправлению ошибок в отдельных разрядах кадра, передаваемого между двумя физически соединенными узлами. Методы обнаружения и исправления ошибок иногда, *но не всегда,* позволяют получателю обнаруживать, что произошла ошибка в одном или нескольких разрядах кадра. Другими словами, даже при помехоустойчивом кодировании, то есть при добавлении к данным избыточной информации (контрольной суммы), остается вероятность, что *ошибка не будет обнаружена,* а получатель не получит информации об искажении по-лученных данных во время их прохождения по каналу связи. В результате канальный уровень получателя может передать сетевому уровню поврежденную дейтаграмму или не заметить повреждения какого-либо другого поля в заголовке кадра. Поэтому следует выбирать такую схему определения ошибок, при которой вероятность подобных событий мала. Как правило, чем изощреннее методы обнаружения и исправления ошибок (снижающие вероятность появления необнаруженных ошибок), тем больше накладные расходы — требуется больше операций для вычисления контрольной суммы и больше времени для передачи дополнительной информации.

**Контроль четности**

Возможно, простейшая форма обнаружения ошибок заключается в использовании одного *бита четности.* При проверке на четность отправитель просто добавляет к данным один бит, значение которого вычисляется как сумма всех *d* разрядов данных

по модулю 2. В этом случае количество единиц в получающемся в результате числе всегда будет четным. Применяются также схемы, в которых контрольный бит инвертируется, в результате чего количество единиц в получающемся в результате числе всегда будет нечетным. Действия, выполняемые получателем при использовании такой схемы, также очень просты. Получатель должен всего лишь сосчитать количество единиц в полученных $d + 1$ разрядах. Если при проверке на четность получатель обнаруживает, что в принятых им данных нечетное количество единичных разрядов, он понимает, что произошла ошибка, по меньшей мере, в одном разряде. В общем случае это
означает, что в полученных данных инвертировано нечетное количество разрядов (произошла ошибка нечетной кратности).

Что произойдет, если в полученном пакете данных произойдет четное количество однобитовых ошибок? В этом случае получатель не сможет обнаружить ошибку.

Если вероятность ошибки в одном разряде мала и можно предположить, что ошибки в отдельных разрядах возникают независимо друг от друга, тогда вероятность нескольких ошибок в одном пакете крайне мала. В таком случае единственного бита четности может быть достаточно. Однако практические наблюдения показали, что в действительности ошибки не являются независимыми, а часто группируются в пакеты ошибок. В случае пакетных ошибок вероятность того, что получатель не обнаружит ошибку в пакете, может приблизиться к величине 50 % .

**Вычисление контрольной суммы**

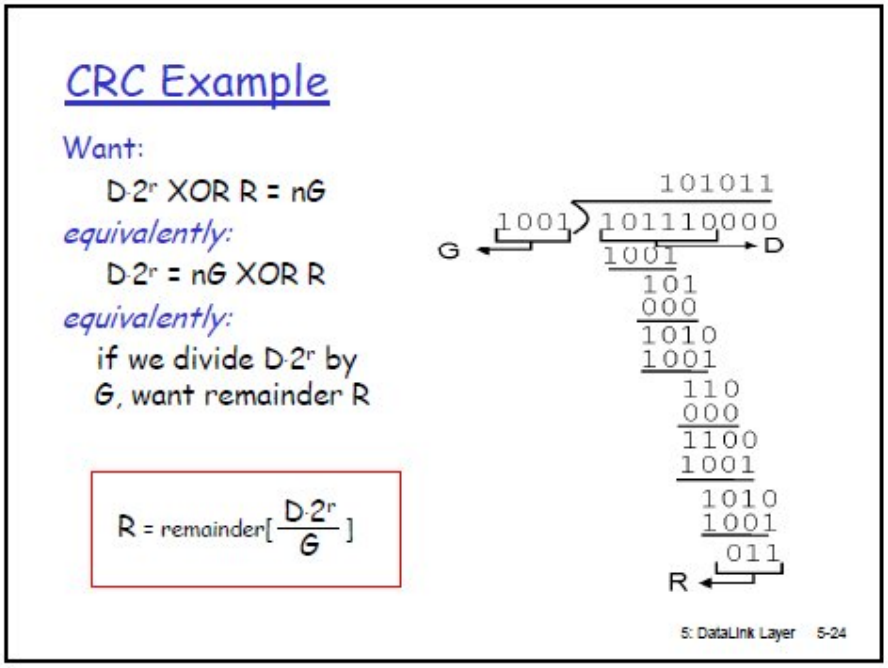
Методы вычисления контрольной суммы обрабатывают *d* разрядов данных как последовательность -разрядных целых чисел. Наиболее простой метод заключается в простом суммировании этих^-разрядных целых чисел и использовании полученной суммы в качестве битов определения ошибок. На этом методе основан алгоритм вычисления контрольной суммы, принятый в Интернете, — байты данных группируются в 16-разрядные целые числа и суммируются. Затем от суммы берется обратное значение (дополнение до 1), которое и помещается в заголовок сегмента. Получатель проверяет контрольную сумму, вычисляя дополнение до 1 от суммы полученных данных (включая контрольную сумму), и сравнивает результат с числом, все разряды которого равны 1.

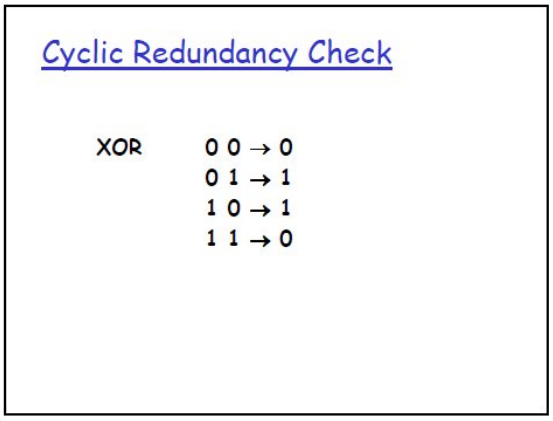
Если хотя бы один из разрядов результата равен 0, это означает, что произошла ошибка. В протоколах TCP и UDP контрольная сумма вычисляется по всем полям (включая поля заголовка и данных). В других протоколах, например XTP, вычисляются две контрольной суммы, одна для заголовка, а другая для всего пакета.

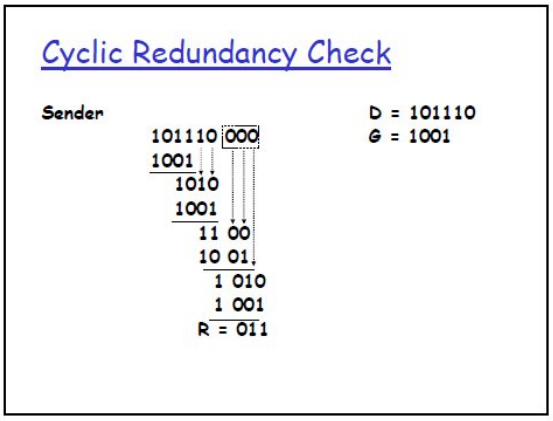
Метод вычисления контрольной суммы для пакетов требует относительно небольших накладных расходов. Например, в протоколах TCP и UDP для контрольной суммы используются всего 16 бит. Однако подобные методы предоставляют относительно слабую защиту от ошибок по сравнению с обсуждаемым далее методом контроля с помощью *циклического избыточного кода,* который часто используется на канальном уровне. Разумеется, возникает вопрос: почему на транспортном уровне применяют контрольные суммы, а на канальном уровне — циклический избыточный код? Вспомним, что транспортный уровень, как правило, реализуется на хосте программно как часть операционной системы хоста. Поскольку обнаружение ошибок на транспортном уровне реализовано программно, важно, чтобы схема обнаружения ошибок была простой. В то же время обнаружение ошибок на канальном уровне реализуется аппаратно в адаптерах, способных быстро выполнять более сложные операции по вычислению циклического избыточного кода.

Таким образом, основная причина использования контрольных сумм на транспортном уровне и более сложного метода вычисления циклического избыточного кода на канальном уровне состоит в том, что программно проще реализовать вычисление суммы.

**Циклический избыточный код**




Широко применяемый в современных компьютерных сетях метод обнаружения ошибок основан на *контроле при помощи циклического избыточного кода* (Cyclic Redundancy Check, CRC). Циклические избыточные коды также называют полиномиальными кодами, так как при их вычислении битовая строка рассматривается как многочлен (полином), коэффициенты которого равны 0 или 1, и операции с этой битовой строкой можно интерпретировать как операции деления и умножения многочленов.

Циклические коды работают следующим образом. Рассмотрим фрагмент данных Д состоящий из *d* разрядов, которые передающий узел хочет отправить принимающему узлу. Отправитель и получатель должны договориться о последовательности из r+ 1 бит, называемой образующим многочленом (или генератором), который мы будем обозначать *G.* Старший (самый левый) бит образующего многочлена *G* должен быть равен 1. Ключевую идею циклических избыточных кодов иллюстрирует рис. 5.7. Для заданного фрагмента данных *D* отправитель формирует г дополнительных разрядов *R,* которые он добавляет к данным Д так что получающееся в результате число, состоящее из *d* + *г* бит, делится по модулю 2 на образующий многочлен (число) *G* без остатка. Таким образом, процесс проверки данных на наличие ошибки относительно прост. Получатель делит полученные *d* + *г* бит на образующий многочлен *G.* Если остаток от деления не равен нулю, это означает, что данные повреждены. В противном случае данные считаются верными и принимаются. Все операции по вычислению CRC-кода производятся в арифметике по модулю 2

без переносов в соседние разряды. Это означает, что операции сложения и вычитания идентичны друг другу и эквивалентны поразрядной операции исключающего ИЛИ (exclusive OR, XOR). Например:
*1011 XOR 0101 = 1110*
*1001XOR 1101 = 0100*
Также мы получим:
*1011-0101 - 1110*
*1001 -1101 = 0100*
Каждый стандарт CRC-кода способен обнаруживать ошибочные пакеты длиной не более г разрядов. Кроме того, при соответствующих допущениях ошибочный пакет длиной более чем г разрядов будет обнаружен с вероятностью 1 - 0,5Г. Помимо этого каждый стандарт CRC-кода может обнаруживать ошибки нечетной кратности.

38. **Multipöördusprotokollid**

Нам всем известно понятие широковещательной рассылки, так как эта технология передачи данных используется в телевидении с момента его изобпетения. Но в традиционном телевидении широковещательная рассылка является односторонней, так как там один фиксированный узел передает информацию множеству получающих узлов. Между тем узлы компьютерной широковещательной сети могут как принимать данные, так и передавать их. Возможно, более близкой к компьютерной широковещательной сети аналогией является вечеринка с коктейлями, где множество людей собираются в большой комнате (средой широковещательного канала при этом является воздух), чтобы поговорить и послушать. Вторая хорошая аналогия, хорошо знакомая многим читателям, — классная комната, в которой преподаватель и студенты совместно используют один общий широковещательный носитель. Центральная проблема обоих сценариев заключается в том, чтобы решить, кто и когда получает право говорить (передавать по каналу). Для себя люди разработали сложный набор правил коллективного использования канала:

• дайте возможность поговорить каждому;
• не говорите, пока с вами не заговорят;
• не монополизируйте беседу;
• если у вас есть вопрос, поднимите руку;
• не прерывайте говорящего громким храпом.

Чтобы гарантировать высокую производительность канала при большом количестве активных узлов, необходимо каким-то образом координировать передачу ими кадров. За эту координацию отвечает протокол коллективного доступа. За последние 30 лет по теме протоколов коллективного доступа были написаны тысячи статей и защищены сотни докторских диссертаций.

За годы с помощью широкого спектра технологий канального уровня были реализованы десятки протоколов коллективного доступа. Тем не менее практически любой из этих протоколов мы можем отнести к одной из трех категорий: *протоколы разделения канала, протоколы произвольного доступа* и *протоколы последовательного доступа.* Мы обсудим эти категории протоколов коллективного доступа в следующих трех подразделах.

В заключение этого обзора отметим, что в идеальном случае протокол коллективного доступа для широковещательного канала со скоростью передачи данных *R* бит/с должен обладать следующими характеристиками.

• Когда данные для передачи есть только у одного узла, этот узел обладает пропускной способностью в *R* бит/с.
• Когда данные для передачи есть у *M* узлов, каждый из этих узлов обладает пропускной способностью в *R/M* бит/с. Это не означает, что каждый из *M* узлов в каждый момент времени может передавать данные со скоростью *R/M* бит/с, — это средняя скорость передачи данных каждого из узлов.

• Протокол является децентрализованным, то есть не существует управляющих узлов, выход из строя которых может остановить работу всей сети.
• Протокол прост и дешев в реализации.

**Протоколы разделения канала**

**TDM- и FDM-мультиплексирование**

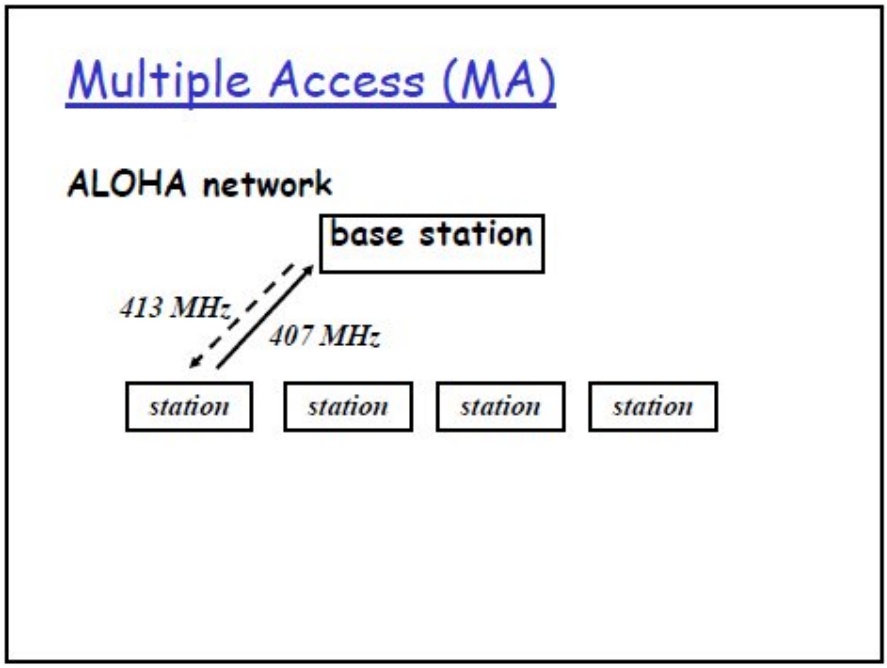
При временном разделении канала время делится на интервалы, называемые *кадрами,* каждый из которых делится на *N* элементарных интервалов времени, называемых *слотами.* Затем каждому из N узлов назначается один временной слот. Когда у узла есть кадр для отправки, он передает биты этого кадра в течение назначенного ему элементарного интервала времени. Как правило, длина слота выбирается таким образом, чтобы за этот интервал можно было передать один кадр. Привлекательность временного разделения канала заключается в том, что такая схема полностью устраняет коллизии и обладает идеальной справедливостью: каждый узел получает выделенную скорость передачи данных, равную *R/N* бит/с в течение каждого временного кадра. Однако у данной схемы есть два существенных недостатка. Во-первых, каждый узел ограничен средней скоростью передачи данных в *R/N* бит/с даже в том случае, когда этот узел единственный, кому нужно передавать данные в этот момент. Во-вторых, при передаче узел всегда должен ждать своей очереди, даже когда кроме него никто не передает данные. Таким образом, очевидно, что временное разделение канала плохо подходит для протокола коллективного доступа.

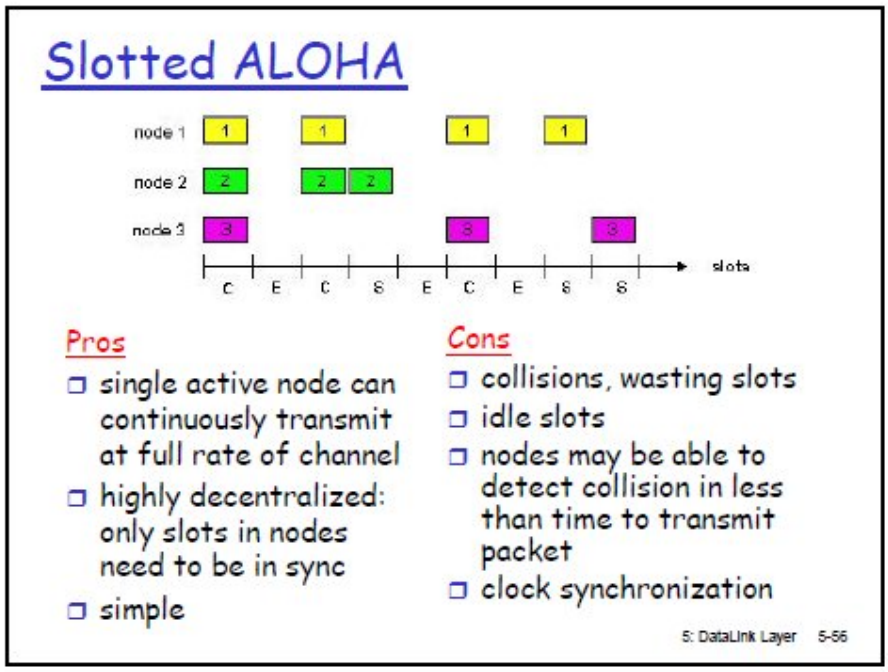
**Протокол CDMA**

*CDMA* (Code Division Multiple Access — множественный доступ с кодовым разделением). В отличие от схем мультиплексирования с частотным и временным разделением канала, предоставляющих узлам частотные диапазоны или временные интервалы, протокол CDMA назначает каждому узлу собственный *код.* Затем каждый узел использует этот уникальный код для кодирования передаваемых им данных. Как мы увидим, протокол CDMA позволяет нескольким узлам передавать данные *одновременно,* при этом получатели могут корректно принимать эти данные (при условии, что получателю известен код передатчика). Протокол CDMA в течение некоторого времени использовался в военных системах связи (благодаря своей устойчивости к попыткам подавления сигнала), а в настоящее время получает все более широкое распространение в гражданских беспроводных средствах связи коллективного доступа.

В протоколе CDMA при передаче каждый бит кодируется, для чего он умножается на некий сигнал (код), изменяющийся с частотой, в несколько раз превосходящей исходную скорость передачи данных.

**39. ALOHA ja CSMA/CD**




**Дискретный протокол ALOHA**

В нашем описании дискретной системы ALOHA мы будем предполагать следующее:

• все кадры состоят ровно из $L$ бит;

• время разделено на интервалы времени (слоты) длительностью $L/R$ секунд (это время, за которое передается один кадр);

• узлы начинают передачу кадров только в момент начала очередного слота;

• узлы синхронизируются так, что каждый узел знает, когда начинается слот;

• если в течение данного временного слота сталкиваются несколько кадров, тогда все узлы обнаруживают факт столкновения, прежде чем закончится данный слот.

Работа дискретного протокола ALOHA на каждом узле проста. Когда у узла есть новый кадр для передачи, он ждет, пока не начнется новый временной слот, после чего весь кадр передается в течение одного временного слота. Если передача проходит без коллизии, повторная передача кадра не требуется (узел может подготовить к передаче новый кадр). В случае коллизии узел обнаруживает факт коллизии,

прежде чем заканчивается данный слот. Затем при наступлении каждого последующего слота с вероятностью$p$ узел передает кадр повторно до тех пор, пока кадр не будет передан без коллизии.

Под повторной передачей кадра с вероятностью $p$ мы подразумеваем, что узел как бы подбрасывает монетку. При этом кадр передается повторно, только если выпадает решка, что происходит с вероятностью $p$. При выпадении орла, что происходит с вероятностью $(1 - p)$, узел пропускает данный временной интервал и подбрасывает монетку еще раз. Все узлы, вовлеченные в коллизию, подбрасывают свои монетки независимо друг от друга.

Может показаться, что дискретный протокол ALOHA обладает рядом достоинств. В отличие от протоколов мультиплексирования с разделением канала, дискретный протокол ALOHA позволяет единственному активному в сети узлу передавать кадры без перерыва с максимальной скоростью $R$ (узел называется активным, если у него есть кадр для передачи). Дискретный протокол ALOHA является в большой степени децентрализованным, так как каждый узел сам обнаруживает факт коллизии и независимо от других узлов принимает решение о времени повторной передачи. (Однако для использования дискретного протокола ALOHA требуется синхронизация узлов. Далее мы кратко обсудим непрерывную версию протокола ALOHA, а также протоколы CSMA, не требующие подобной синхронизации и, таким образом, являющиеся полностью децентрализованными.) Кроме того, ALOHA представляет собой чрезвычайно простой протокол.

Дискретный протокол ALOHA хорошо работает в тех ситуациях, когда имеется только один активный узел, но какова его эффективность при наличии нескольких активных узлов? Эффективность дискретного протокола ALOHA снижается благодаря двум факторам. Во-первых, как показано на рис. 5.13, когда в сети несколько активных узлов, определенная доля слотов тратится впустую из-за коллизий. (Как показано на рисунке, в первом слоте в коллизию вовлекаются три узла. Затем узлу 2 удается передать кадр в четвертом слоте, узлу 1 — в восьмом слоте, а узлу 3 — в девятом.) Во-вторых, другая доля слотов тратится напрасно, когда все активные узлы одновременно отказываются от передачи. Дискретный протокол ALOHA работает продуктивно только в те слоты, когда передавать требуется ровно одному узлу. Слот, на протяжении которого передает только один узел, называется *успешным слотом*. Эффективность дискретного протокола коллективного доступа определяется долей успешных слотов в ситуации большого количества активных узлов, у каждого из которых всегда есть большое количество кадров для передачи. Обратите внимание, что, если вообще не использовать протокола коллективного доступа и после коллизии немедленно повторять передачу каждым из узлов, эффективность сети была бы равна нулю. Несмотря на то что мгновенная пропускная способность канала равна 100 Мбит/с, его успешная пропускная способность составит менее 37 Мбит/с.

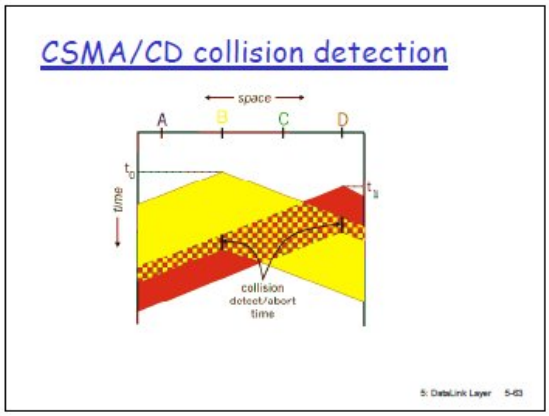
**Чистый протокол ALOHA**

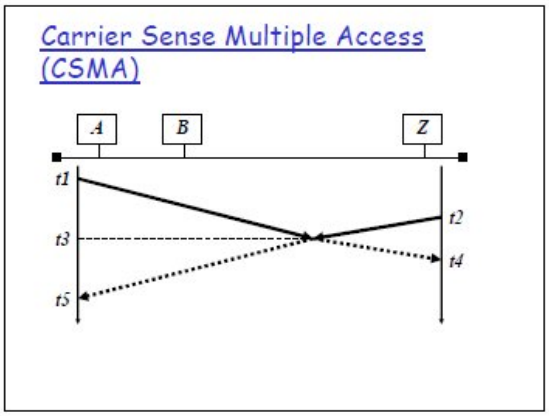
Дискретный протокол ALOHA требует, чтобы все узлы синхронизировали время начала передачи кадров. Собственно, первый протокол ALOHA не был дискретным, представляя собой полностью децентрализованный протокол. В так называемом чистом протоколе ALOHA, когда прибывает первый кадр (то есть дейтаграмма сетевого уровня передается на более низкий уровень передающего узла), узел немедленно передает весь кадр целиком в широковещательный канал. Если переданный кадр сталкивается с одним или несколькими другими кадрами, с вероятностью p узел немедленно передает кадр повторно. В противном случае узел выжидает в течение времени, необходимого для передачи одного кадра, после чего опять с вероятностью p передает кадр либо пережидает еще один интервал времени.

Чтобы определить максимальную эффективность чистого протокола ALOHA, сконцентрируем наше внимание на отдельном узле. Мы будем использовать те же допущения, что и в случае дискретного протокола ALOHA, и примем за единицу времени

интервал (слот), требующийся для передачи одного кадра. В любой момент времени вероятность того, что узел передает кадр, равна p.

**Протокол CSMA**




В обоих вариантах протокола ALOHA, дискретном и чистом, узел принимает решение о передаче кадра независимо от активности остальных узлов, присоединенных к широковещательному каналу. В частности, узел не обращает внимания на то, ведется ли в данный момент передача другими узлами, и не прекращает передачу в случае коллизий.

• *Слушайте, прежде чем говорить.* Если кто-то уже говорит, подождите, пока он не закончит. В мире компьютерных сетей это правило называется *контролем несущей* и заключается в том, что узел прослушивает канал перед тем, как начать передачу. Если по каналу передается кадр, узел выжидает (отступает) в течение случайного периода времени, после чего снова опрашивает канал. Если канал оказывается свободным, узел начинает передачу кадра. В противном случае узел ждет в течение еще одного случайного интервала времени и повторяет весь процесс.

• *Если кто-то начал говорить, прекращайте разговор.* В мире компьютерных сетей это правило называется *обнаружением* коллизий. Оно заключается в том, что во время передачи узел прослушивает канал. Если он обнаруживает, что другой узел в этот момент времени тоже ведет передачу, он прекращает свою передачу и с помощью протокола вычисляет время своей следующей попытки передачи.

Эти два правила реализованы в семействе протоколов *CSMA* (Carrier Sense Multiple Access — множественный доступ с контролем несущей) и *CSMA/CD* (CSMA with Collision Detection — множественный доступ с контролем несущей и обнаружением коллизий).

В разделе Протоколы коллективного доступа упоминался применяемый в Ethernet алгоритм коллективного доступа CSMA/CD (Carrier Sense Multiple Access with Collision Detection — множественный доступ с контролем несущей и обнаружением коллизий). Вспомним некоторые его свойства.

• Адаптер может начинать передачу в любое время, то есть время не делится на слоты.

• Адаптер никогда не начинает передачу, если он слышит передачу другого адаптера, то есть выполняется контроль несущей.

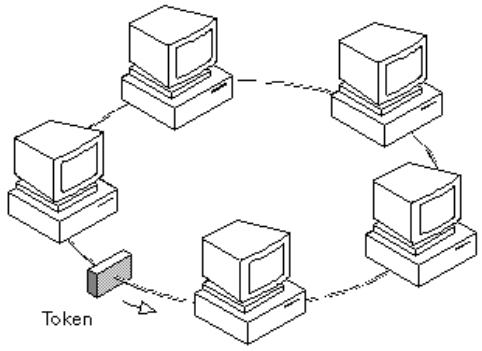
• Адаптер прерывает передачу, как только он обнаруживает, что другой адаптер также выполняет передачу, то есть имеет место обнаружение коллизий.

• Прежде чем пытаться повторить передачу после обнаружения коллизии, адаптер выжидает в течение интервала времени случайной длительности. Как правило, по сравнению со временем, необходимым для передачи кадра, этот интервал небольшой.

Благодаря этим свойствам производительность протокола CSMA/CD значительно превышает производительность дискретного протокола ALOHA. В самом деле,

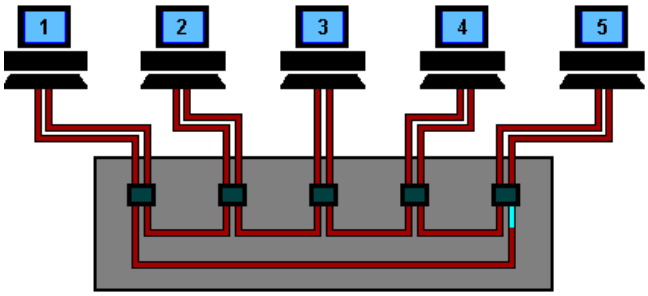
если максимальная задержка распространения сигнала между станциями невелика, эффективность протокола CSMA/CD может достигать 100 %. Однако обратите внимание, что адаптеру приходится решать две важные задачи: прослушивать передачу другого адаптера (контролировать несущую) и обнаруживать коллизии. Эти две задачи адаптеры выполняют, измеряя уровни напряжения на линии перед началом и во время передачи.
Протокол CSMA/CD работает на каждом адаптере независимо от других адаптеров локальной Ethernet-сети.
1. Адаптер принимает от своего родительского узла единицу обмена (PDU) сетевого уровня, формирует Ethernet-кадр и помещает его в буфер адаптера.
2. Если адаптер определяет, что канал свободен (то есть в нем не наблюдается сигнала), он начинает передавать кадр. Если же канал занят, адаптер ждет, пока сигнал в линии не прекратится (плюс еще время, необходимое для передачи 96 бит), после чего начинает передачу кадра.
3. Передавая кадр, адаптер отслеживает наличие напряжения от сигнала других адаптеров. Если адаптер успевает передать весь кадр, не обнаружив сигнала других адаптеров, передача кадра считается успешной.
4. Если адаптер обнаруживает сигнал других адаптеров во время собственной передачи, он прекращает передачу кадра и передает 48-разрядный сигнал коллизии (jam signal).
5. После этого адаптер входит в фазу *экспоненциального отката.* В частности, после *n* неудачных попыток повторить передачу одного и того же кадра адаптер выбирает значение *K* псевдослучайным образом из множества {0, 1, 2,..., *2m* - 1}, где *m* := т т ( я , 10). Затем адаптер выжидает в течение интервала времени, длительность которого равна *Kx* 512 длительностей передачи одного бита, после чего возвращается к шагу 2.

40. **Token ring** -тип сети, в кот. все компьютеры схематически объединены в кольцо, по кот. от компьютера к компьютеру передается спец. блок данных, - маркер (*token*). Когда какой-либо станции требуется передача данных, 3-битный маркер ею модифицируется в пакет данных и больше не распознается другими станциями, пока не дойдёт до адресата.
Стандарт IEEE 802.5
Сеть поддерживает два базовых типа фреймов : маркеры и фреймы с данными/коммандами.
Длина маркера 3 бита, и он состоит из стартового разделителя, контрольного бита доступа,и разделителя в конце маркера.
Фремы с данными имеют разный размер, в зависимости от размера передаваемой инфы.
**После отправки пакета с данными**
1. адресат принимает этот пакет и запускается новый маркер по кольцу.
2. если адресат не найден, то отправитель сам удаляет пакет когла тот пройдёт 1 полный круг
3. Машина со специальными полномочиями следит за гуляющими пакетами и в случае необходимости удаляет безадресные данные и запускает новый маркер.




**Минусы сети**

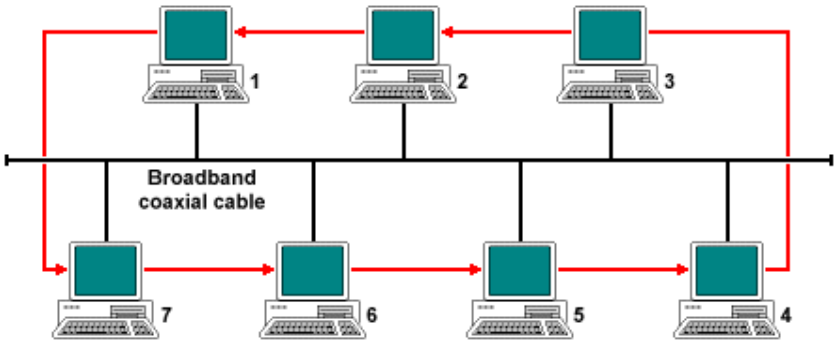
1. Если кабель накроется в каком-либо месте , то кольцо прервётся и перестанет работать,  одним и решением проблемы- использовать хабы (рис 2)
2. По кольцу может гулять только 1 маркер, поэтому только возможно только 1 передача за раз.
3. Ограничено кол-во  «нодов»( компов) в такой сети  - 250 макс




**Структура фрейма**

SD AC FC DA SA Data FCS ED FS
SD AC ED
Где
SD  Start Delimiter
AC  Access Control
ED  End Delimiter

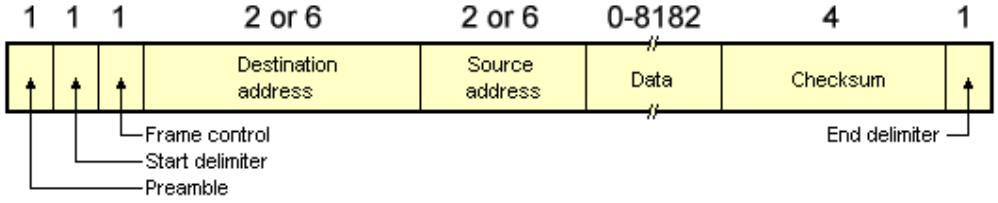
FC Frame Control  (Loa edastamine,Paketi saatmine, Loa prioriteet, Paketi prioriteet)
DA Destination Address (2/6)
SA Source Address (2/6)
Data (…5000)
FCS Frame Check Sequence (4)
FS Frame Status
**41.Token bus** – local area сеть (IEEE 802.4), в которой станции  на шине(кабеле) формируют «виртульаное кольцо». Маркет гуляет по сети,и только нод получивший маркер может передавать данные. Если ноду нечего передавать, то маркер передаётся дальше по виртуальному кольцу . Каждый нод должен знать своего соседа слева и справа, чтобы передать ему маркер по сети, поэтому используется спец протокол, который информирует все ноды о новых подкоючения или отключениях к/от сети.




**Token Bus MAC layer протокол**
При инициализиции кольца, маркер добавляется в сеть и передаётся станции с наибольшем номером. (Макер передаётся от станции с наибольшим номеров к наименьшему). Если станции получила маркер, у неё есть ограниченный период времени, за который она передавать фреймы с данными( кол-во переданных фремов зависит от их размера). Если станции нечего передать, то она передаёт маркер следующей станции в кольце без задержки.
Token bus определяет 4 класса приорететов при передаче данных (0,2,4,6)  6-  самый высокий приоритет, 0 имеет самый низкий.

У станции есть 4 очереди для передачи фреймов в зависимости от их приоритета. Когда станция получает маркер и права на передачу данных, она отправляет фреймы сначала из очереди с наибольшим приоритетом, затем с меньшим и ещё меньшим. На каждую очередь выделяется определённое время, за которое она передаёт фреймы. По истечению этого времени – возможность отправить фреймы передаётся следующей очереди,и так до последней. Если в очереди нет фреймов данного приоритета, но права на передачу отдаётся ниже стоящей по приоритету очереди.
Такая схема гарантирует, что пакеты с наибольшим приоритетом (6) будут отправлены раньше остальных, что весьма полезно, к примеру, при синхронизации работ машин( на фабриках, заводах)

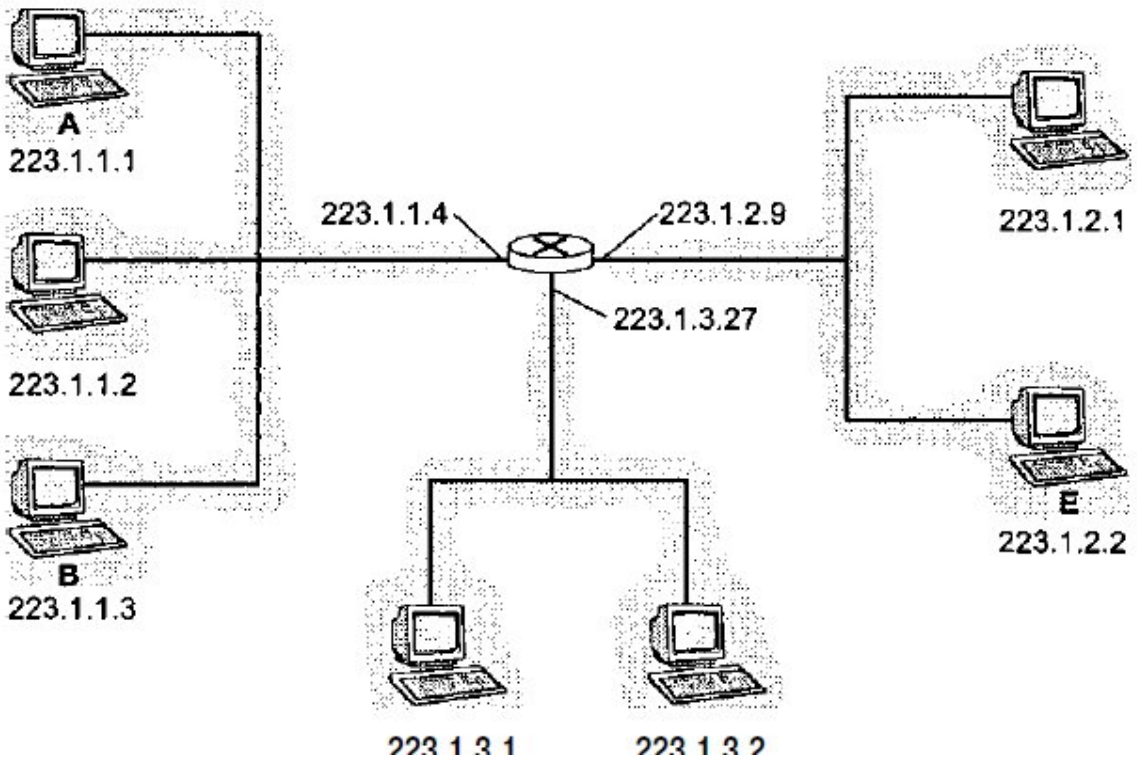
**Структура фрейма**




Preamble –для синхронизаче часов получателя
*Start Delimeter* **и** *End Delimeter* маркеруют начало и конец фрейма, и содержат значение отличное от 0 и 1 которое не может случайно возникнуть где-либо в данных ещё в данных во фрейме. Поэтому можно определить где начало и конец и не нужно знать длину фрейма.
*Frame Control* определяет тип фрейма – с данными или контольный фрейм. Для фреймов с данными он показывает приоритет фрейма . Для контрольного - показывает тип фрейма (конрольный фрейм может сожержать комманды Claim token (CT),Solicit successor (SolS),Who follows me (WF),Resolve contention (TS),Set successor (SetS),Token (T), Loa edastamine,Ringiga liitumine,Ringist lahkumine,Ringi käivitamine)
Destination **и** Source адреса содержат 2-битный или 6-битный адрес получателя и отправителя(сеть также должна иметь или 2-битную или 6-битную адресацию, что-то одно,не оба).
Data Field содержит информация,передаваему получателю.
Checksum(Frame Check Sequence) используется для выяления ошибок при передаче

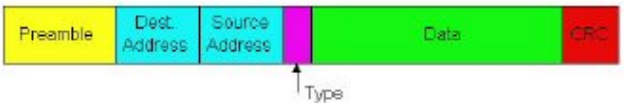
**42. Datagrammide edastus läbi võrkude**

Продвижение дейтаграмм зависит от того, располагаются ли оба хоста в одной и той же сети или в разных сетях.
1)Пусть сначала хост А хочет послать IP-дейтаграмму В, находящемуся в той же сети что и хост А.Это делается следующим образом. Сначала протокол IP хоста А заглядывает в свою внутреннюю IP-таблицу продвижения данных,и находит в ней сетевой адрес, совпадающий со старшими разрядами IP-адреса хоста В.Хост В расположен в той же самой сети, что и хост А. Таким образом, хост А узнает, что дейтаграмму хосту В можно послать прямо по выходному интерфейсу хоста А безо всяких промежуточных маршрутизаторов.Тогда хост А передает IP-дейтаграмму протоколу канального уровня, который передает ее хосту В.
2)Хосту А нужно передать дейтаграмму другому хосту, например хосту Е, находящемуся в другой сети.В этом случае посылают промежуточному маршрутизатору.Хост Л узнает, что предназначаемые хосту ? дейтаграммы следует направлять интерфейсу маршрутизатора с адресом, с которым интерфейс хоста Л соединен напрямую. Затем протокол IP хоста Л передает дейтаграмму канальному уровню, сообщая ему, что дейтаграмму следует отправить по IP-адресу. Здесь важно отметить, что, хотя дейтаграмма посылается (с помощью канального уровня) интерфейсу маршрутизатора, поле адреса получателя дейтаграммы, по-прежнему содержит адрес конечного получателя (хоста ?), а не адрес интерфейса промежуточного маршрутизатора.




**43. Ethernet -** определяет проводные соединения и электрические сигналы на физическом уровне, формат кадров и протоколы управления доступом к среде
Очень рапространнёная NIC (сетевая карта) проще и быстрее чем Маркерная сеть или ATM
большая скорость: 10 Mbps – 10 Gbps
Каждый Ethernet интерфейс имеет мак адресс оборудования, который указывается ещё при произвдстве
Пригодно при коаксиальных кабелях, витой паре и оптическом кабеле.

Структура протокола

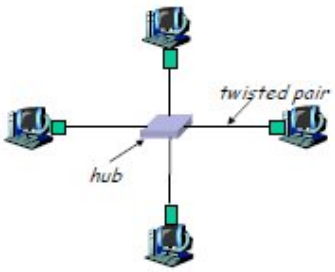


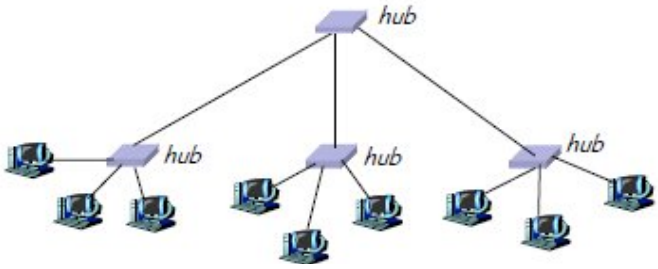

Preamble: 7 bytes вида 10101010, заними 1 byte вида 10101011 , нужен для синхронизации часов отправителя и получателя
**Addresses**: 6 bytes, Если адапатер получает фрейм с адресом машины- получателя, или адресом вещание(ARP пакет), то адапатер направляет фрейм на network layer протокол, иначе удаляет фрейм
**Type**: показывает протокол более высокового уровня, к примеру IP
**CRC**: проверяется получателем,если найдена ошибка , то фрейм удаляется

**Ethernet & CSMA/CD**
**Как передача работает**
1. NIC адапатер получает датагруму и сетевого уровня и создаёт фрейм
2. Если кабель свободен, то начинается передача фрейма
Если кабель занят, то подождать пока освободиться
3. Если адаптер переслал весь фрейм и не заметил столкновений, то передача закончена
4. Если адапатер заметил другую передачу, то посылает jam сигнал
5. После прекрашение адаптер ждёт,и вновь повторяет шаг 2

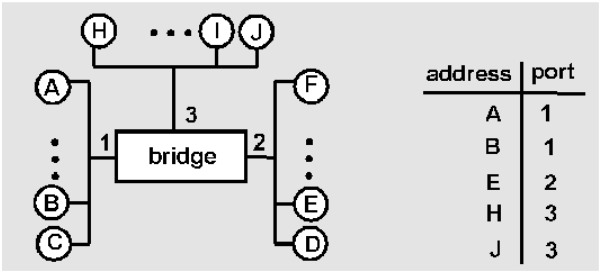
**44. Концентраторы(hub), мосты(brige), и коммутаторы(switch)**
**Hub или концентратор** - многопортовый повторитель сети с автосегментацией. Все порты концентратора равноправны. Получив сигнал от одной из подключенных к нему станций, концентратор транслирует его на все свои активные порты. Концентраторы можно использовать как

автономные устройства или соединять друг с другом, увеличивая тем самым размер сети и создавая более сложные топологии.

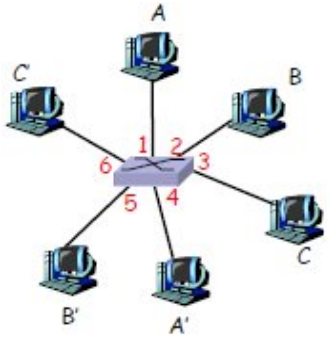



**Bridge (мост) -** Устройство, соединяющее две или несколько физических сетей и передающее пакеты из одной сети в другую. Мосты могут фильтровать пакеты(Смотрят заголовок ферйма и выборочно передают пакет(Если адрес получателя присутствует в таблице адресов моста, кадр передается только в тот сегмент или сеть, где находится получатель. )
Не заметны для соединёных сторон, не требуют настройки или установки, самообчающиеся (
У моста есть табилица, в которой хранится (Node LAN Address, Bridge Interface, Time Stamp),а устаревшие записи удаляются. )
Мост определяет через какой интерфейм можно достич тот или иной хост.
При получение нового фрейма,мосты запоминают место отправителя(его интерфейс) и добавляют запись в таблицу.
Если мост не знает в каком интрефейсе находится хост, то он оптравляет фрейм во все интерфейсы, кроме того откуда он пришёл. Если мост знает где получатель,то он передаёт фрейм сразу только в тот интерфейс.



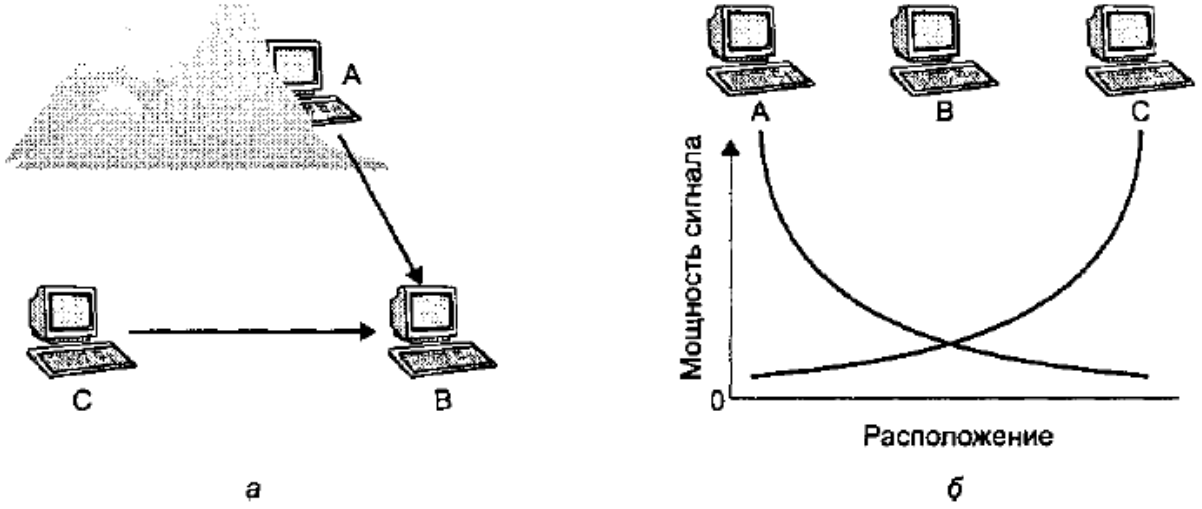

Предположим C отправляем пакет D,а D отвечает фреймом для C
Bridge получает фрейм от C, и добавляет запись что C находится на Интерфейсе 1
Т.к. D нет в таблице, то мост шлёт во все оставшиеся интерфейсы этот фрейм
D получает фрейм,и высылает обратно фрейм для C
Мост получает фрейм, запоминает место D на интерфейсе 2
Мост уже знает что C на Интерфейсе 1, поэтому сразу передаёт в интерфейс 1 этот фрейм
**Коммутаторы(switch)** позволяет пересылать пакеты между несколькими сегментами сети. Он является обучающимся устройством и действует по аналогичной технологии. Коммутатор анализирует адрес назначения в заголовке пакета и, сверившись с адресной таблицей, тут же (время задержки около 30-40 микросекунд) направляет этот пакет в соответствующий порт. Таким образом, когда пакет еще целиком не прошел через входной порт, его заголовок уже передается через выходной**.**




Многоинтерфейсный мост, перенаправляет фрейсы и фильтрует по LAN адресу
Связывает A-A` и B-B` паралельно, без столкновений

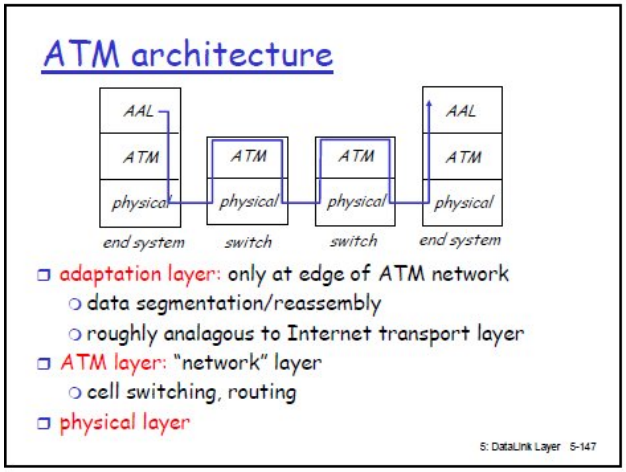
Не заметны для соединёных сторон, не требуют настройки или установки, самообчающиеся, Хосты имеют выделенное прямое соденинея со свитчем, свитч буферизирует пакеты, полная двух сторонняя передача
У свитча есть табилица, в которой хранится (MAC хоста, интерфейс,и метка времени)
Работает как мост, При получение нового фрейма, запоминает место отправителя(его интерфейс) и добавляет запись в таблицу. Если  не знает в каком интрефейсе находится хост, то он оптравляет фрейм во все интерфейсы, кроме того откуда он пришёл. Если свитч знает где получатель,то он передаёт фрейм сразу только в тот интерфейс.

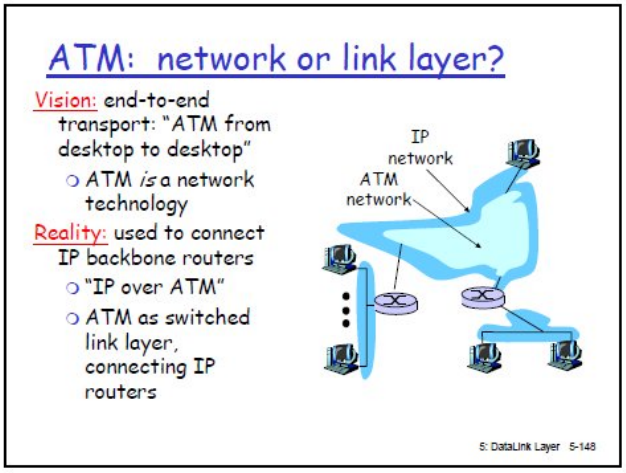
**45.CSMA/CA**
CSMA/CA-множественный доступ с контролем несущей и
предотвращением коллизий  протокол.
CSMA сначала прослушивает канал, чтобы определить, не занят ли канал другой
станцией, передающей кадр.Если канал оказывается свободным в течение времени,
большего чем распределенный интервал между кадрами станция получает разрешение на передачу.
Как и в любом протоколе
произвольного доступа, кадр будет успешно получен принимающей станцией, если
другие станции не начнут передачу и их сигнал не наложится на сигнал, передаваемый
данной станцией.Разработчики
стандарта IEEE 802.11 создали протокол доступа, основное назначение которого
состоит  в том, чтобы предотвращать коллизии-(CSMA/CA)




**46. ATM**
Сетевая технология, способная передавать как аудио и видео в реальном времени, так и текст, электронную почту и изображения. Данная технология получила название *ATM* (Asynchronous Transfer Mode — режим асинхронной передачи).
Стандартами ATM предусматривается коммутация пакетов с использованием виртуальных каналов (также иногда называемых виртуальными цепями) и определяются интерфейсы приложений с ATM. Таким образом, технология ATM предоставляет законченное сетевое решение для распределенных приложений.
**Основные характеристики ATM**
•Стандарт ATM определяет полный набор протоколов обмена данными от прикладного до физического уровня.
•Модели обслуживания ATM включают обслуживание с постоянной битовой скоростью (Constant Bit Rate, CBR), с переменной битовой скоростью (Variable Bit Rate, VBR), с доступной битовой скоростью (Available Bit Rate, ABR) и с неуказанной битовой скоростью (Unspecified Bit Rate, UBR).
•В ATM применяется коммутация пакетов фиксированной длины 53 байта. В терминах ATM эти пакеты называются *ячейками.* Каждая ячейка состоит из 5-байтового заголовка и 48-байтовой полезной нагрузки. Фиксированная длина ячеек и простота заголовков упрощают высокоскоростную коммутацию ATM-ячеек.

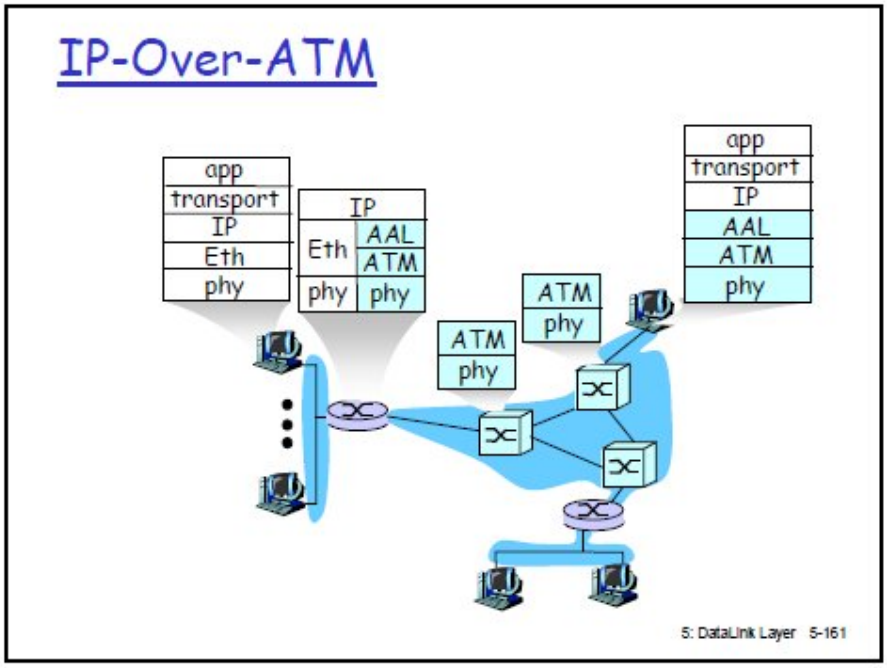
•В ATM-сетях используются *виртуальные каналы.* Номер виртуального канала, называемый *идентификатором виртуального канала* (Virtual Channel Identifier, VCI), помещается в специальное поле заголовка ATM-ячейки. Идентификаторы VCI используются коммутаторами для направления ATM-ячеек их адресатам.
•Технология ATM не предоставляет повторной передачи ячеек на канальном уровне. Если коммутатор обнаруживает ошибку в заголовке ATM-ячейки, он пытается исправить ее при помощи помехоустойчивых кодов. Если исправить ошибку не удается, коммутатор не запрашивает повторную передачу у предыдущего коммутатора, а просто отбрасывает ячейку.
•ATM-сети могут работать поверх практически любого физического уровня. Эта технология часто применяется поверх оптоволоконных кабелей, использующих стандарт SONET со скоростями передачи данных 155,52 Мбит/с, 622 Мбит/с и выше.
•*Физический уровень A TM* имеет дело с напряжениями, синхронизацией битов и формированием кадров, передаваемых по физическому носителю.
•*Уровень A TM* представляет собой ядро стандарта ATM. Он определяет структуру ATM-ячейки.
•*Уровень адаптации ATM* в определенной степени соответствует транспортному уровню стека протоколов Интернета.
На сегодня ATM в основном применяется как технология канального уровня в локализованных областях Интернета.




**Физический уровень ATM**

Физический уровень занимается передачей ATM-ячейки по одной физической линии. Физический уровень состоит из двух подуровней: подуровня PDM (Physical Medium Dependent —подуровень, зависимый от физического носителя) и подуровня TC (Transmission Convergence —подуровень конвергенции передачи).
Подуровень PMD располагается в самом низу стека протоколов ATM. Подуровень PMD отвечает за передачу и прием отдельных битов. Существуют два класса подуровней PMD: подуровни PMD, имеющие структуру кадров передачи (например, T1, T3, SONET или SDH), и подуровни PMD, не имеющие такой структуры. Если у подуровня PMD есть структура кадров передачи, тогда он отвечает за формирование кадров и определение их границ. (Употребляемый в данном разделе термин □кадр□ не следует путать с кадром канального уровня, упоминавшимся в начале этой главы. Кадр передачи представляет собой механизм физического уровня, вроде мультиплексирования с временным разделением, позволяющий организовать передачу битов по линии.) Подуровень PDM имеет дело с битами и не □видит□ ячеек.
Ниже перечислены некоторые возможные варианты подуровней PDM.
•SONET/SDH (Synchronous Optical Network/Synchronous Digital Hierarchy — синхронная оптическая сеть/синхронная цифровая иерархия) поверх одномодового оптоволоконного кабеля. Подобно линиям T1 и T3, технологии SONET и SDH обладают структурой кадров, позволяющей выполнять синхронизацию битов между передатчиком и приемником.




**Уровень ATM**

При работе протокола IP поверх ATM ATM-ячейки играют роль кадра канального уровня. Структура ATM-ячейки и значение каждого поля ячейки определяются уровнем ATM. Первые 5 байт ячейки составляют заголовок. Остальные 48 байт несут полезную нагрузку. Структура заголовка ATM-ячейки показана на рис. 5.43. Ниже перечислены поля заголовка ATM-ячейки.

•*VCI* (Virtual Channel Identifier —идентификатор виртуального канала). Определяет виртуальный канал, которому принадлежит ячейка. Как и в большинстве сетевых технологий, использующих виртуальные каналы, идентификатор виртуального канала ячейки транслируется от линии к линии (см. раздел Ядро компьютерных сетей в главе 1).

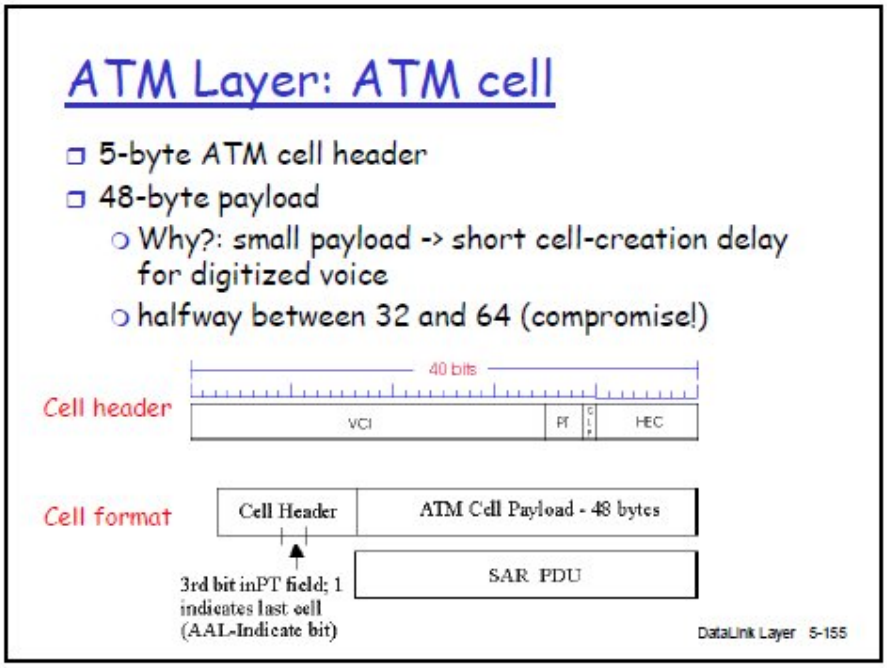
•*PT* (Payload Type —тип полезной нагрузки). Указывает тип полезной нагрузки, содержащейся в ячейке. Определены несколько типов полезной нагрузки для данных, еще несколько типов полезной нагрузки для управляющих ячеек, а также тип полезной нагрузки для холостых ячеек. (Вспомним, что холостые ячейки иногда бывают нужны физическому уровню для синхронизации.)

•*CLP* (Cell-Loss Priority —приоритет отбрасывания ячеек). Бит CLP может устанавливаться источником для разграничения высокоприоритетного и низкоприоритетного трафиков. Если возникает затор и ATM-коммутатору приходится отбрасывать ячейки, при помощи этого бита коммутатор может отфильтровывать ячейки низкоприоритетного трафика.

•*HEC* (Header Error Control —контрольная сумма заголовка). Восемь избыточных битов, позволяющих обнаруживать и исправлять некоторые ошибки в заголовке ячейки.

**Виртуальные каналы**

Прежде чем источник сможет начать передачу ячеек адресату, ATM-сеть должна установить *виртуальный канал* (VC) от источника до адресата. Понятие виртуального канала уже обсуждалось в разделе Ядро компьютерных сетей главы 1. Виртуальный канал образуется из линий связи, расположенных между отправителем и получателем. Каждая из линий, входящих в виртуальный канал, идентифицируется собственным номером виртуального канала. При установке или разрыве виртуального канала таблицы трансляции номеров виртуального канала должны обновляться (см. главу 1). Как уже отмечалось ранее, в ATM-магистралях Интернета часто применяются постоянные виртуальные каналы, что позволяет избавиться от необходимости устанавливать и разрывать виртуальный канал в динамическом режиме.




**Уровень адаптации ATM**

Назначение уровня AAL (ATM Adaptation Level —уровень адаптации ATM) заключается в том, чтобы позволить существующим протоколам (например, IP) и приложениям (например, потоковое видео с постоянной скоростью передачи данных) работать поверх ATM-сети. Как показано на рис. 5.44, уровень AAL реализуется только на оконечных точках ATM-сети. Данной оконечной точкой может быть хост (если ATM обеспечивает сквозное соединение от хоста к хосту) или IP-маршрутизатор (если ATM используется для соединения двух 1P-маршрутизаторов).
В этом случае уровень AAL аналогичен транспортному уровню в стеке протоколов Интернета.

47. **Võrkude turvalisus ja ohud** , безопасность сетей и опасности подстерегающие сети –

Свойства безопасных сетей :

1)*Конфиденциальность* – только отправитель и предполагаемый получатель должны знать содержимое передаваемых сообщений , следовательно сообщения должны быть зашифрованы , чтобы перехватив их нельзя было расшифровать . Для этого используются криптографические ключи, для шифрации и дешифрации.

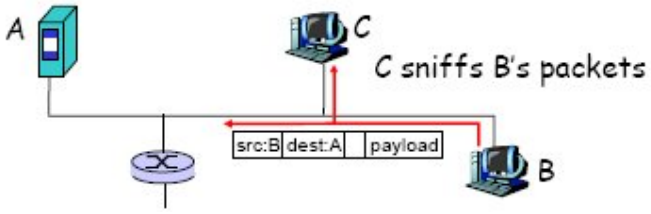
2) *Аутентификация* – отправитель и получатель должны идентифицировать и подтвердить личности(объекты ) друг друга .( методы с применением криптографии )

3)*Целостность сообщения* – отправитель и получатель должны быть уверены , что сообщения доставлены или получены без изменений , в оригинальном виде . Для того чтобы избежать измененных сообщений , используется подсчет контрольных сумм как в транспортных протоколах и в протоколах передачи данных , применяются методы криптографии…

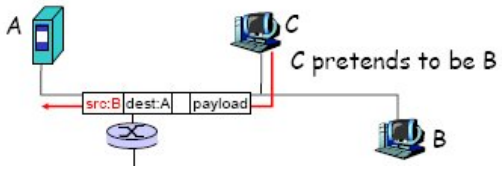
4)*Управление доступом* - хост или сеть должна быть доступна авторизированному пользователю , устойчива к атакам типа DDOS( отказ от обслуживания )

Это ключевые компоненты безопасной связи. Ещё один вариант - использование брандмауэра – устройства , которое расположено между внутренними и внешними сетями , защищающее их друг от друга и от попадания вредоносных пакетов в сеть. Основная идея – управление доступом пакетов и отправка пакетов из сети. Кроме того , помимо механизмов защиты есть и механизмы обнаружения атак и механизмы ответных действий .

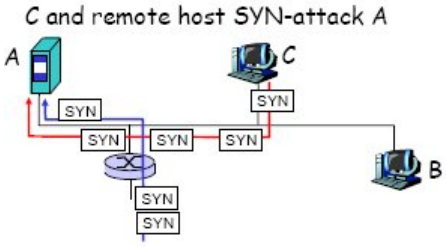
Опасности , подстерегающие в сети : 1) Packet sniffing – использование специальных программ sniffer`ов для чтения чужих пакетов




2) Ip spoofing – создание ложных ИП дейтаграмм для обмана получателя

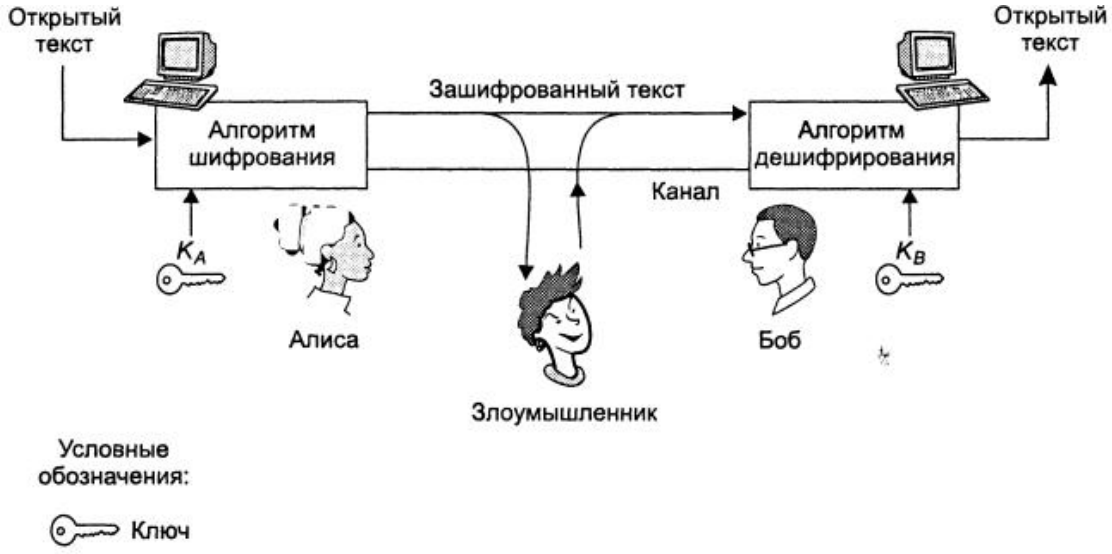



3) dDos ( атака кучи ботнетов какого-нибудь хоста с целью завалить его пакетами и вызвать отключение или отказ от работы и тп. )




48. **Krüptograafia , algoritmid , võtmed** - крипто, алгоритмы и ключи .

Криптография позволяет отправителю скрыть содержимое сообщений , чтобы злоумышленник не смог прочитать их не имея ключ для расшифровки . С другой же стороны , получатель должен всегда иметь возможность расшифровать сообщение с помощью ключа.




Основной идеей шифрования является 1) субституция – замена текущих символов на какие-либо другие символы при помощи алгоритма 2) транспозиция - изменение порядка расположения символов .

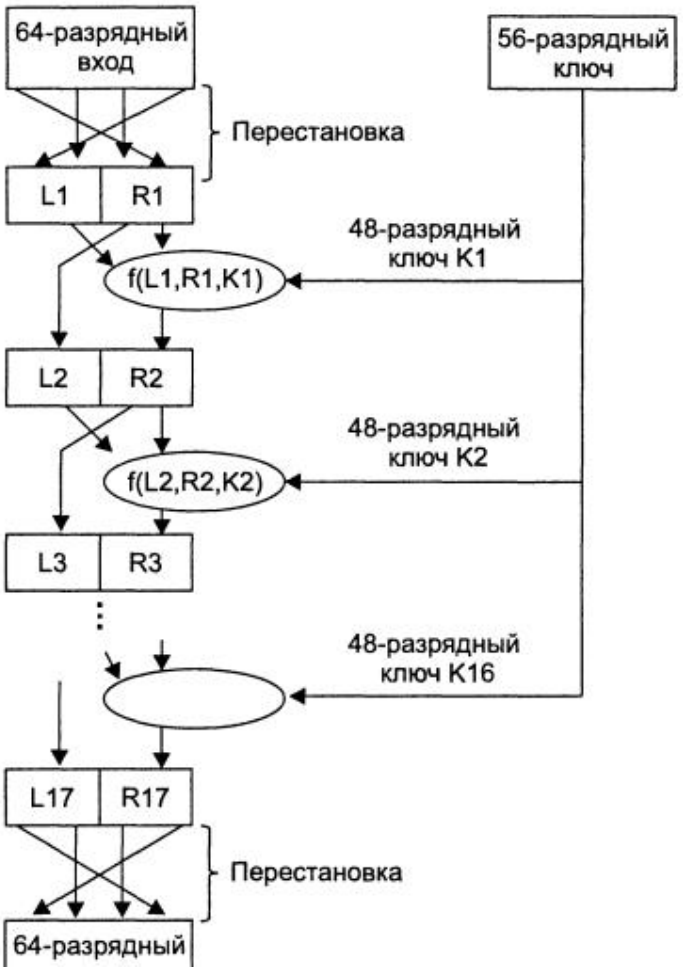
Различают 2 основных алгоритма шифрования : симметричный и открытый . Симметричные алгоритмы имеют меньшую длину ключа , работают на порядок быстрее открытых и в большей степени служат для шифрования данных .

Открытые алгоритмы же медленнее, однако применяются в основном при работе с ключами и различными криптографическими протоколами.

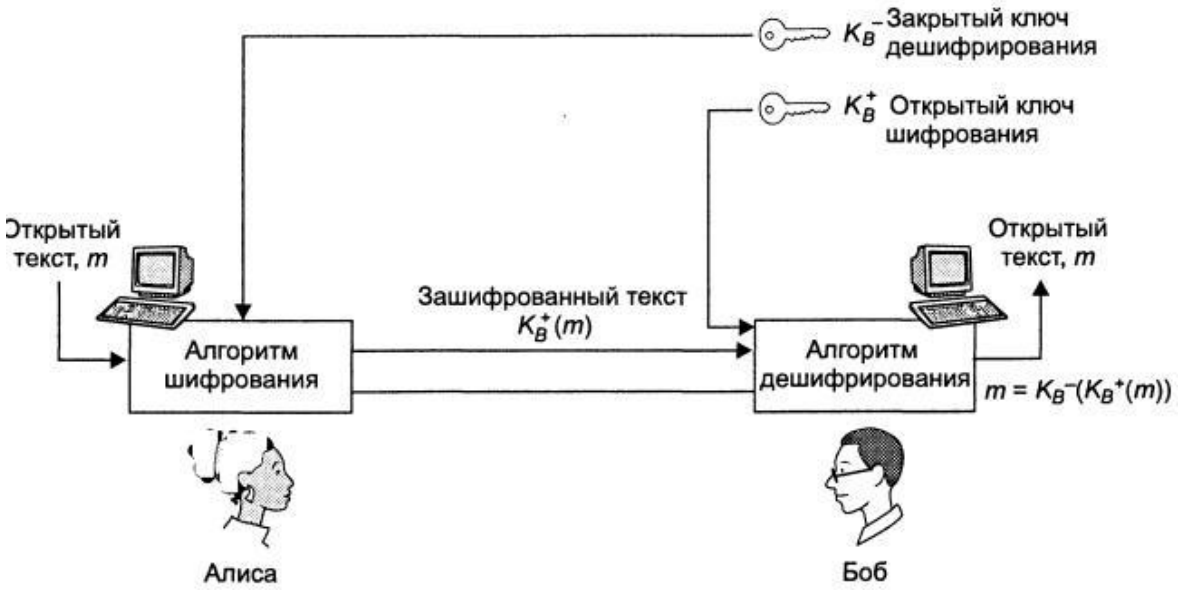
*Симметричные системы* используют идентичные секретные ключи , самым простым примером алгоритма является шифр Цезаря - в нем каждая буква сдвигается на фиксированное кол-во символов , например на +3. Усовершенствованная версия называется - моноалфавитный шифр, в нем также каждая буква алфавита заменяется другой . Однако не регулярными или фиксированным образом , а на любую другую букву при условии , что каждая буква замещается уникально

Существует также *полиалфавитный шифр* - идея в использовании нескольких моноалфавитных шифров , при чем комбинация зависит от позиции элемента.

Общепризнанным стандартом симметричного шифрования является DES , - длина 56бит (AES - 128)




*Открытый алгоритм шифрования* – не требует, чтобы стороны использовали один и тот же ключ . Поскольку отправитель и получатель мог не знать друг друга, а следовательно никогда не смогут передать уникальный секретный ключ для шифровки и дешифровки , возникла идея использовать 2 ключа – открытый и личный пользовательский ключ , тем самым нет надобности передавать секретный ключ друг другу .




*Самый известный алгоритм -RSA* , идея в использовании простых чисел и общих делителей. Сначала выбирается открытый ключ, шифруется сообщения – подбираются простые числа, передаются пара чисел для открытого ключа, другая пара остается у владельца . Получатель получает сообщение , берет свою пару чисел и расшифровывает ).

**49. Sümmeetrilise võtme krüptograafia, DES**
**Шифрование с симметричными ключами**
Все криптографические алгоритмы заменяют один текст другим: открытый текст — зашифрованным.
DES – представляет собой стандарт шифрования с симметричным ключом, опубликованный в 1997 году и обновленный в 1993 году национальным бюро стандартов США для шифрования коммерческой и несекретной государственной документации. Шифр DES кодирует 64-разрядные блоки кода при помощи 64-разрядного ключа. В действительности
длина ключа шифра DES составляет всего лишь 56 бит, так как 8 из 64 бит
представляют собой биты четности (по одному для каждого из восьми байтов).
Национальный институт стандартов и технологии (National Institute of Standards and Technology, NIST), который сменил национальное бюро стандартов (National Bureau of Standards, NBS), следующим образом формулирует назначение шифраDES : «Цель заключается в том, чтобы полностью скремблировать данные и ключ, так чтобы каждый бит данных в зашифрованном тексте зависел от каждого бита открытого текста и каждого бита ключа... Не должно быть никакой корреляции между зашифрованным текстом и оригинальными данными или ключом».

В алгоритме DES перестановка выполняется в два этапа (первый и последний шаги алгоритма), в которых 64 бита переставляются местами. Между этими двумя этапами выполняется еще 16 идентичных этапов. Входными данными для каждого из этих 16 этапов является результат предыдущего этапа.На каждом этапе левые 32 бита заменяются правыми 32 битами.Все 64 входных бита на г'-м этапе и 48-разрядный ключ z-го этапа (полученный из 56-разрядного ключа DES) подаются на вход функции, превращающей 4-разрядные блоки данных в 6-разрядные, складывающей эти 6-разрядные блоки по модулю 2 с полученными при помощи аналогичных операций 6-разрядными блоками 48-разрядного ключа *К»* а также операции замены и еще одного сложения по модулю 2 с левыми 32 битами входных данных. Полученные в результате этих вычислений 32 бита образуют правые 32 бита 64-разрядного результата (рис. 7.5). При дешифрировании используются обратные операции этого алгоритма.

**50. Avaliku võtme krüptograafia, RSA**
**Шифрование с открытым ключом**
В 1976 году Диффи и Хеллман продемонстрировали алгоритм (называемый теперь алгоритмом обмена ключа Диффи и Хеллмана),обеспечивающий такую возможность,как обмениваться по сети шифрованными сообщениями, не имея изначально общего секретного ключа — принципиально

отличный и элегантный метод безопасной связи, приведший к разработке современных криптографических систем с открытым ключом. Криптографические системы с открытым ключом обладают замечательными свойствами, благодаря которым могут использоваться не только для шифрования данных, но также для аутентификации и цифровых подписей.

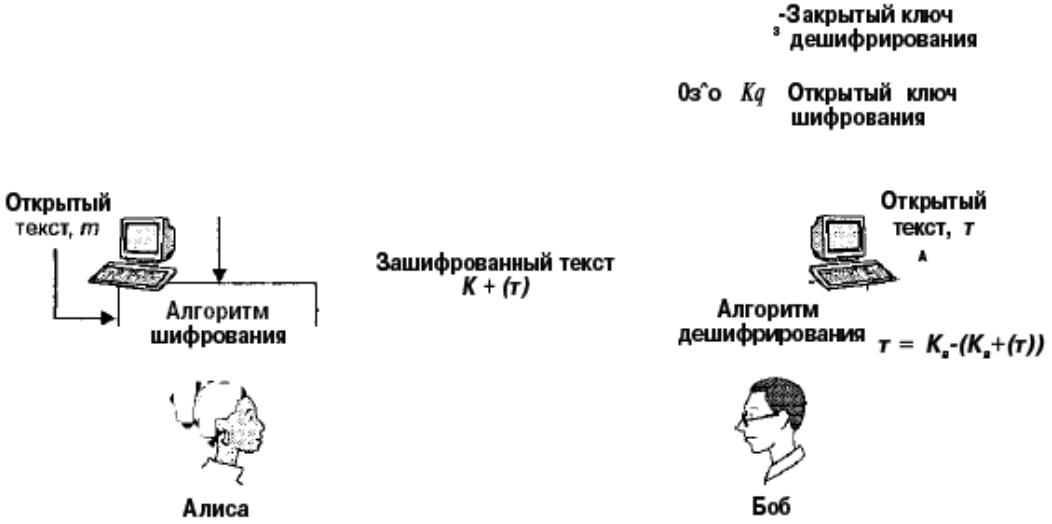



**Рис. 7.6. Шифрование с открытым ключом**

Хотя в системах шифрования с открытым ключом могут применяться самые разные алгоритмы, один такой алгоритм, *алгоритм RSA* стал почти синонимом шифрования с открытым ключом. Рассмотрим сначала, как работает алгоритм RSA, а затем поговорим о том, почему он работает. Алгоритм RSA подразумевает два тесно связанных этапа.

1. Выбор открытого и личного ключей.
2. Шифрование и дешифрировани

**Таблица 7.1. Этапы шифрования**

| Символ открытого текста | т:числовое представление | me | Зашифрованный текст: c = me mod n |
|---|---|---|---|
| l | 12 | 248 832 | 17 |
| o | 15 | 759 375 | 15 |
| v | 22 | 5 153 632 | 22 |
| e | 5 | 3 125 | 10 |

**Таблица 7.2. Этапы дешифрирования**

| Зашифрованный текст с | cd | т = = cd mod n | Открытый текст |
|---|---|---|---|
| 17 | 481968572106750915091411825223071697 | 12 | l |
| 15 | 127834039488589391112327575683593 75 | 15 | o |
| 22 | 851643319086537701956194499721106030592 | 22 | v |
| 10 | 10000000000000000000000000000000 | 5 | e |

Безопасность алгоритма RSA основывается на том факте, что алгоритм быстрого определения делителей числа, в данном случае алгоритм нахождения делителей/? и *q* числа *n,* пока не найден. Если вам известны значения/? и *q,* тогда, зная открытое число *e,* нетрудно вычислить секретный ключ *d.* С другой стороны, быстрые алгоритмы нахождения делителей числа всего лишь не найдены, отсутствие таких алгоритмов не доказано, поэтому, строго говоря, безопасность алгоритма RSA не гарантирована.

**51. Autentimine** - процесс распознования пользователей, а также сетевых элементов (пример: маршрутизатор). Осуществляться исключительно на основе сообщений, которыми обмениваются

участники протокола аутентификации. Как правило, протокол аутентификации запускается прежде, чем две общающиеся стороны запустят другой протокол (например, надежный протокол передачи данных, протокол обмена таблицами маршрутизации или протокол электронной почты). Сначала протокол аутентификации устанавливает аутентичность участников сеанса связи, и только после этого участники сеанса получают возможность продолжать сеанс связи. При аутентификации по сети общающиеся стороны не могут полагаться на биометрическую информацию, такую как внешний вид или тембр голоса.

Аутентификация часто требуется сетевым элементам, таким как маршрутизаторы и процессы клиентов и серверов. Таким образом, аутентификация должна осуществляться исключительно на основе сообщений, которыми обмениваются участники *протокола аутентификации.* Как правило, протокол аутентификации запускается *прежде,* чем две общающиеся стороны запустят другой протокол (например, надежный протокол передачи данных, протокол обмена таблицами маршрутизации или протокол электронной почты). Сначала протокол аутентификации устанавливает аутентичность участников сеанса связи, и только после этого участники сеанса получают возможность продолжать сеанс связи.

**Протокол аутентификации ap 1.0**

Возможно, самым простым протоколом аутентификации, который мы можем себе представить, является такой протокол, при котором Алиса просто посылает Бобу сообщение о том, что она Алиса

**Протокол аутентификации ap 2.0**

В том случае, если у Алисы есть известный сетевой адрес, используемый ею при связи (например, IP-адрес), Боб может попытаться аутентифицировать Алису по ее IP-адресу. Однако это может остановить только очень наивного злоумышленника. Узнать сетевой адрес не так уж и трудно (например, если у злоумышленника есть доступ к исходным текстам операционной системы, что, например, характерно для Linux и некоторых других операционных систем, он может построить собственное ядро операционной системы). В этом случае злоумышленник сможет создавать IP-дейтаграммы

и указывать в них IP-адрес нужного источника (например, IP-адрес Алисы).

**Протокол аутентификации ap 3.0**

Классический подход к аутентификации состоит в использовании паролей. У нас есть PIN-коды, позволяющие подтвердить нашу личность торговым автоматам, и пароли для входа в операционные системы. Пароль хранится в тайне от всех, и знают его только два участника процесса аутентификации — тот, кто подтверждает свою личность, и тот, кто ее проверяет. В протоколе аутентификации ap 3.0 Алиса посылает Бобу свой секретный пароль (рис. 7.9).Поскольку пароли так широко применяются, мы можем предположить, что протокол аутентификации ap 3.0 надежен. Однако это не так. И взломать этот протокол совсем несложно. Злоумышленнику нужно всего лишь перехватить сообщение, посылаемое Алисой, чтобы узнать ее пароль. При использовании протокола TELNET для работы на удаленном сервере пароль также пересылается в открытом виде.

**Протокол аутентификации ap 4.0**

Недостаток протокола аутентификации ap 3.1 заключается в том, что пароль не меняется. Один из методов решения проблемы состоит в том, чтобы каждый раз использовать новый пароль. Алиса и Боб могут договориться о последовательности отправки паролей (или алгоритме генерации паролей) и применять каждый пароль из последовательности всего один раз. Эта идея реализована в системе S/KEY , в которой для генерации последовательности паролей применяется метод Лампорта.

**Протокол аутентификации ap 5.0**

Естественный вопрос заключается в том, можем ли мы для решения проблемы аутентификации использовать нонсы и шифрование с открытым ключом (вместо шифрования с симметричным ключом). Шифрование с открытым ключом позволило бы устранить недостаток любой системы шифрования с симметричными ключами, заключающийся в проблеме передачи секретного ключа, который должны знать две стороны.

**52. Digitaalallkiri** - метод шифрования с открытым ключом предоставляет простой способ создания цифровой подписи документа, подписи, которую можно проверить, невозможно подделать и, следовательно, от которой нельзя отречься. Кроме того, эта же подпись защищает документ от последующей модификации. Процесс подписи документа выглядит следующим образом. На первом шаге строится специальная функция, напоминающая контрольную сумму - хэш-функция, она идентифицирует содержимое документа (создается "дайджест" документа). На втором шаге автор документа шифрует содержимое хэш-функции своим персональным закрытым ключом. Зашифрованная хэш-функция помещается в то же сообщение, что и сам документ. Цифровая

подпись является производной “дайджеста” и личного закрытого ключа, чем гарантируется её абсолютная уникальность. Алгоритм верификации электронной подписи состоит в следующем. На первом этапе получатель сообщения строит собственный вариант хэш-функции подписанного документа. На втором этапе происходит расшифровка хэш-функции, содержащейся в сообщении. На третьем этапе производится сравнение двух хэш-функций. Их совпадение гарантирует одновременно подлинность содержимого документа и его авторства.

**53. Sertifitseerimine** - обеспечению гарантии подлинности открытых ключей служит так называемый **сертификат** (**certificate**). сертификат содержит:

серийный номер ( уникальный для запрашивающий стороны )

информацию о владельце сертификата, включая алгоритм и значение самого ключа ( не показывается )

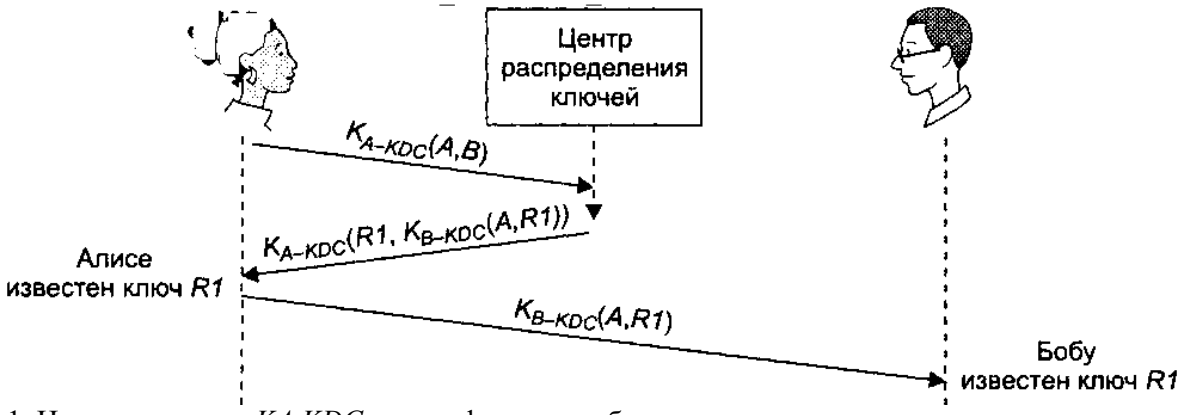
информацию о запрощике сертификата

даты действия

Сертификат “подписывается” закрытым ключом человека (или организации), который выпускает сертификаты. В схемах шифрования с открытым ключом доверенный посредник называется сертификационным центром (Certification Authority, С А). Сертификационный центр проверяет, является ли объект (человек, маршрутизатор и т. д.) тем, кем он себя объявляет. После проверки подлинности объекта сертификационный центр создает сертификат, который связывает открытый ключ объекта с идентификатором этого объекта. Сертификат содержит открытый ключ и глобально уникальный идентификатор владельца этого ключа (например, имя человека или его IP-адрес). Сертификат снабжается цифровой подписью сертификационного центра. Затем этот сертифицированный ключ может распространяться каким угодно способом, например через сервер открытых ключей, личную web-страницу или дискету. Среди компаний, предоставляющих подобные услуги, можно назвать Digital Signature Trust Company и Verisign .

**54. Võtmete jaotussüsteemid ja protokollid** - Центр распределения ключей представляет собой сервер, совместно использующий симметричные секретные ключи со всеми своими зарегистрированными пользователями. Эти ключи могут согласовываться с сервером вручную при первой регистрации пользователя. Центру распределения ключей известен секретный ключ каждого пользователя, и каждый пользователь может безопасно общаться с центром распределения ключей при помощи этого ключа.

Предположим, Алиса и Боб являются пользователями центра распределения ключей. Им известны только их собственные ключи *KA_KDC* и *KB_KDC.*




1. Используя ключ *KA.KDC* для шифрования обмена данными с центром распределенияключей, Алиса посылает центру сообщение, в котором указывает, что она *(A)* хочет установить связь с Бобом *(B).* Это сообщение обозначим как *KA_KDC (A, B).*

2. Зная ключ *KA_KDO* центр распределения ключей расшифровывает сообщение ***KA-KDC*** (А *B).* Затем центр распределения ключей генерирует случайное число *R1.* Это значение общего ключа, которым будут пользоваться Алиса и Боб для симметричного шифрования сообщений друг другу. Этот ключ называется *одноразовым ключом сеанса,* так как Алиса и Боб будут использовать его только в текущем сеансе. Чтобы сообщить Алисе и Бобу значение *R1,* центр распределения ключей посылает Алисе зашифрованное ключом *K^юс* сообщение, содержащее следующие данные.

• Одноразовый ключ сеанса *R1,* которым будут пользоваться Алиса и Боб.

• Пару значений *A* и *R1,* зашифрованных центром распределения ключей при помощи ключа Боба *KB_KDC.* Это сообщение мы будем обозначать как ***KB-KDC*** (А Важно отметить, что центр распределения ключей посылает

Алисе не только значение *R1,* но также значение *R1* и имя Алисы, зашифрованные при помощи ключа Боба. Алиса не может расшифровать эту пару значений в сообщении (ей не известен ключ Боба), но ей это и не требуется.„ Далее будет показано, что Алиса просто переправляет эту зашифрованную пару значений Бобу, который может их расшифровать.
Центр распределения ключей помещает эти значения в сообщение, зашифровав их при помощи ключа Алисы, и посылает это сообщение Алисе. Таким образом, сообщение, посылаемое центром распределения ключей Алисе, - это *KA_KDC(R1, KB_KDC* (А *R1)).* Алиса получает сообщение от центра распределения ключей, расшифровывает его и извлекает из него значение *R1.* Теперь Алисе известен одноразовый ключ сеанса *R1.* Кроме того, Алиса извлекает сообщение *KB_KDC (A, R1)* и переправляет
его Бобу. Боб расшифровывает полученное сообщение *KB_KDC* (Л, *R1)* с помощью ключа *KB_KDC* и извлекает из него *A* и *R1.* Теперь Бобу известен ключ сеанса *R1* и тот, с кем он будет обмениваться зашифрованными этим ключом данными. Разумеется, прежде чем продолжить обмен данными, Боб выполняет с Алисой процедуру аутентификации (проверяя ее личность) с помощью только что полученного ключа сеанса *R1.*

**55. Tulemüürid** - Анализатор пакетов представляет собой программу, работающую на присоединенном к сети устройстве и пассивно получающую все информационные кадры канального уровня, проходящие через сетевой адаптер устройства. В широковещательной среде, такой как локальная сеть Ethernet, это означает, что анализатор пакетов получает все пакеты, отправляемые всеми хостами локальной сети, а также все пакеты, получаемые этими хостами. Роль анализатора пакетов может исполнять любой хост с Ethernet-картой, так как в неупорядоченном режиме сетевой Ethernet-адаптер будет получать все проходящие по сети кадры. Эти кадры могут передаваться прикладной программе, извлекающей данные прикладного уровня. Анализ пакетов представляет собой «палку о двух концах» — он может быть очень полезен сетевому администратору для мониторинга сети и управления сетью и не менее полезен хакеру для его «черных» дел. Программное обеспечение анализаторов пакетов можно бесплатно загрузить с различных web-сайтов или купить. Для обнаружения анализаторов пакетов достаточно обнаружить сетевые интерфейсы, работающие в неупорядоченном режиме. На предприятии сетевые администраторы могут установить на всех компьютерах предприятия специальное программное обеспечение, предупреждающее о переходе сетевого интерфейса в неупорядоченный режим. Для дистанционного обнаружения сетевых интерфейсов, работающих в неупорядоченном режиме, могут применяться различные методы. Например, хост, отвечающий на эхо-запрос протокола ICMP с корректным IP-адресом, но некорректным MAC-адресом (адрес канального уровня), скорее всего, работает с интерфейсом в неупорядоченном режиме. Тем не менее можно обеспечить безопасную работу сети и в условиях несанкционированного анализа пакетов. Для этого нужно просто шифроватьвсе данные, передаваемые по сети (особенно пароли).
Например, в изображенном на рис. 7.25 сценарии хост С перехватывает запрос пароля, посылаемый хостом А хосту В, а также сам пароль, посылаемый хостом В хосту А.

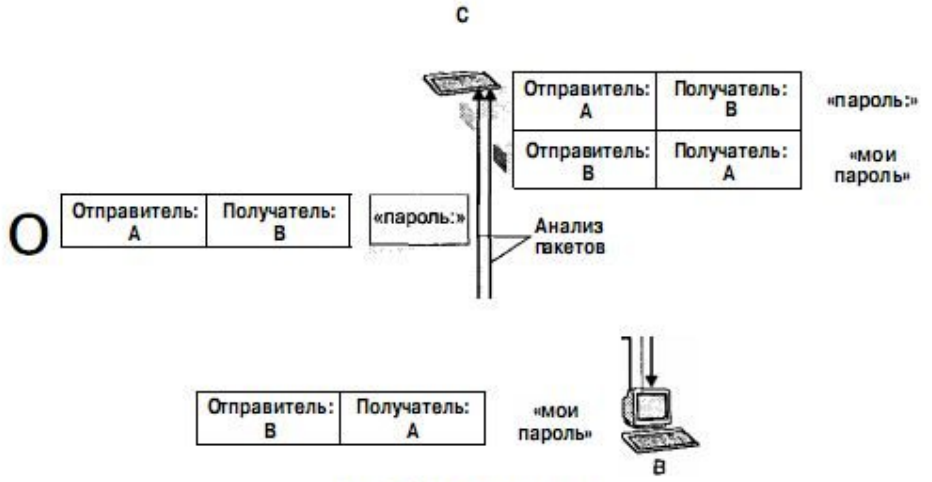



Рис. 7.25. Анализ пакетов

**56. Transpordikihi turvalisus, SSL** - Протокол SSL (Secure Sockets Layer — слой защищенных сокетов), разработанный компанией Netscape, предназначен для шифрования данных и аутентификации между web-клиентом и web-сервером. Протокол начинает свою работу с фазы

рукопожатия, при которой две стороны соединения договариваются об используемом алгоритме шифрования (например, DES или IDEA) и ключах. В этой же фазе производится аутентификация сервера и при необходимости клиента для сервера. После завершения фазы рукопожатия начинается передача прикладных данных. При этом все данные зашифровываются при помощи ключа сеанса, о котором стороны договариваются на стадии рукопожатия. Протокол SSL широко применяется в Интернет-коммерции. Он реализован почти во всех популярных web-браузерах и web-серверах. Кроме того, он составляет основу протокола TLS (Transport Layer Security — безопасность на транспортном уровне). Протокол SSL может применяться не только для оплаты покупок кредитными картами в Интернете, но и для других финансовых транзакций, включая банковские и биржевые операции. Сфера применения протоколов SSL и TLS не ограничивается Всемирной паутиной. Например, они также могут использоваться для аутентификации и шифрования данных в протоколе ШАР (Internet Mail Access Protocol — протокол доступа к почте Интернета). Протокол SSL может рассматриваться как некий слой между прикладным и транспортным уровнями. На передающей стороне протокол SSL получает данные (например, HTTP- или IMAP-сообщения от приложения), зашифровывает их и направляет в TCP-сокет. На принимающей стороне протокол SSL считывает данные из TCP-сокета, расшифровывает их и направляет приложению.

**57. E-kaubandus, SET - E-kaubandus** - это любая транзакция, которая осуществляется через компьютерную сеть и приводит к передаче собственности или прав на использование товаров или услуг. Системы E-kaubandus: B2C (бизнес-потребитель) - системы интернет-торговли, в которой в качестве продавца выступает юридическое лицо, а в качестве покупателя – физическое. B2B (бизнес-бизнес) – системы эектронной коммерции, в которой в качестве субъектов процессов продажи и покупки выступают юридические лица. Основными субъектами электронной коммерции выступают Продавец, Банк Продавца, Покупатель, Банк Покупателя. Протокол **SET** узко специализирован и применяется для обеспечения безопасности при осуществлении платежей с использоваанием пластиковых карт через Интернет. Все участники операций идентифицируются при помощи сертификатов. SET предоставляет самый защищенный на настоящее время механизм выполнения платежей. Преимущества использования SET: *продавцы защищены от покупок с помощью неавторизованной платежной карточки и от отказа от покупки; *банки защищены от неавторизованных покупок; *клиенты не пострадают от перехвата номера кредитки и от покупки у несуществующих продавцов. SET позволяет проводить проводить авторизацию, используя цифровые подписи, и одновременно защищает покупателей, обеспечивая механизм передачи номера карты для проверки непосредственно эмитенту, минуя промежуточные звенья.

**58. Võrgukihi turvalisus, IPsec** - Безопасность на сетевом уровне обеспечивает протокол IPsec (IP security — безопасный протокол IP), по сути представляющий собой набор протоколов. В набор протоколов IPsec входят два основных протокола: протокол AH (Authentication Header — заголовок аутентификации) и протокол ESP (Encapsulation Security Payload — безопасная инкапсуляция полезной нагрузки). Когда хост-отправитель посылает дейтаграмму хосту-получателю, он использует либо протокол AH, либо протокол ESP. Протокол AH обеспечивает аутентификацию источника и целостность данных, но не обеспечивает конфиденциальности. Протокол ESP обеспечивает аутентификацию источника, целостность данных и конфиденциальность. В процедуре аутентификации протокола IPsec (как для протокола AH, так и для протокола ESP) используется схема вычисления зашифрованного дайджеста сообщения. В схеме для аутентификации сообщений применяется общий секретный ключ, а нешифрование с открытым ключом.


Рис. 7.34. Поля протокола ESP в IP-дейтаграмме

AH-заголовок содержит несколько полей: Следующий заголовок, Индекс параметра безопасности (SPI), Порядковый номер (32-разрядное поле), данные аутентификации.

**59. Võrguhaldus, SNMP** - является протоколом прикладного уровня, предназначенным для облегчения обмена информацией управления между сетевыми устройствами. Пользуясь информацией SNMP (такой, как показатель числа пакетов в секунду и коэффициент сетевых

ошибок), сетевые администраторы могут более просто управлять производительностью сети и обнаруживать и решать сетевые проблемы. Протокол SNMP относится к прикладному уровню в стеке протоколов TCP/IP. Он работает по системе "менеджер-агент". Менеджер (серверная программа SNMP) посылает запросы агентам, агенты (т.е. программы SNMP объектов управления) устанавливаются в контролируемых узлах, они собирают информацию (например, о загрузке, очередях, временах совершения событий), и передают ее серверу для принятия нужных мер. В общем случае агентам можно поручить и обработку событий, и автоматическое реагирование на них. Для этого в агентах имеются триггеры, фиксирующие наступление событий, и средства их обработки. Команды SNMP могут запрашивать значения объектов MIB (Managment Information Base, база данных), посылать ответы, менять значения параметров. Для посылки команд SNMP используется транспортный протокол UDP. Одной из проблем управления по SNMP является защита агентов и менеджеров от ложных команд и ответов, которые могут дезорганизовать работу сети. Используется шифрование сообщений, но это снижает скорость реакции сети на происходящие события. Расширением SNMP являются протоколы RMON (Remote Monitoring) для сетей Ethernet и Token Ring и RMON2 для сетевого уровня. Преимущество RMON заключается в меньшем трафике, так как здесь агенты более самостоятельны и сами выполняют часть необходимых управляющих воздействий на состояние контролируемых ими узлов. На базе протокола SNMP разработан ряд мощных средств управления, примерами которых могут служить продукт ManageWISE фирмы Novell или система UnicenterTNG фирмы Computer Associates. С их помощью администратор сети может: 1) строить 2D изображение топологии сети, причем на разных иерархических уровнях, перемещаясь от региональных масштабов до подсетей ЛВС (при интерактивной работе); 2) разделять сеть на домены управления по функциональным, географическим или другим принципам с установлением своей политики управления в каждом домене; 3) разрабатывать нестандартные агенты с помощью имеющихся инструментальных средств.

**60. Sünkroniseerimine, asünkroon- ja sünkroonedastus** - При обмене данными на физическом уровне единицей информации является бит, поэтому средства физического уровня всегда поддерживают побитовую синхронизацию между приемником и передатчиком.Канальный уровень оперирует кадрами данных и обеспечивает синхронизацию между приемником и передатчиком на уровне кадров. В обязанности приемника входит распознавание начала первого байта кадра, распознавание границ полей кадра и распознавание признака окончания кадра.Обычно достаточно обеспечить синхронизацию на указанных двух уровнях - битовом и кадровом, - чтобы передатчик и приемник смогли обеспечить устойчивый обмен информацией. Однако при плохом качестве линии связи (обычно это относится к телефонным коммутируемым каналам) для удешевления аппаратуры и повышения надежности передачи данных вводят дополнительные средства синхронизации на уровне байт.Такой режим работы называется *асинхронным* или *старт-стопным*. Другой причиной использования такого режима работы является наличие устройств, которые генерируют байты данных в случайные моменты времени. Так работает клавиатура дисплея или другого терминального устройства, с которого человек вводит данные для обработки их компьютером. асинхронном режиме каждый байт данных сопровождается специальными сигналами «старт» и «стоп». Назначение этих сигналов состоит в том, чтобы, во-первых, известить приемник о приходе данных и, во-вторых, чтобы дать приемнику достаточно времени для выполнения некоторых функций, связанных с синхронизацией, до поступления следующего байта. Сигнал «старт» имеет продолжительность в один тактовый интервал, а сигнал «стоп» может длиться один, полтора или два такта, поэтому говорят, что используется один, полтора или два бита в качестве стопового сигнала, хотя пользовательские биты эти сигналы не представляют. синхронным описанный режим называется потому, что каждый байт может быть несколько смещен во времени относительно побитовых тактов предыдущего байта. Такая асинхронность передачи байт не влияет на корректность принимаемых данных, так как в начале каждого байта происходит дополнительная синхронизация приемника с источником за счет битов «старт». Более «свободные» временные допуски определяют низкую стоимость оборудования асинхронной системы.

При синхронном режиме передачи старт-стопные биты между каждой парой байт отсутствуют. Пользовательские данные собираются в кадр, который предваряется байтами синхронизации. Байт синхронизации - это байт, содержащий заранее известный код, например 0111110, который оповещает приемник о приходе кадра данных. При его получении приемник должен войти в байтовый синхронизм с передатчиком, то есть правильно понимать начало очередного байта кадра. Иногда применяется несколько синхробайт для обеспечения более надежной синхронизации

приемника и передатчика. Так как при передаче длинного кадра у приемника могут появиться проблемы с синхронизацией бит, то в этом случае используются самосинхронизирующиеся коды.