1. Базовая эталонная модель взаимодействия открытых систем (модель OSI)

OSI – Open System Interconnection; модель, определяющая уровни взаимодействия систем в сетях с коммутацией пакетов, дает им стандартные имена и указывает, какие функции должен выполнять каждый уровень.
Всего уровней 7:

1. Физический
2. Канальный
3. Сетевой
4. Транспортный
5. Сеансовый
6. Представления
7. Приложения

При пересылке сообщения от А к Б пакет проходит уровни 7 6 … 1 в А, а затем 1 2 … 7 в Б.

Взаимодействие уровней.
Служба-пользователь – уровень, который запрашивает функциональность другого уровня.
Служба-провайдер – предоставляет эту функциональность.
А: i+1 пользователь, i провайдер. Б: i-1 пользователь, i провайдер. Таким образом, уровень i в OSI взаимодействует только с уровнями i-1 и i+1.

Подробнее про каждый уровень.

1. Передача *битов* по физическим каналам связи. Характеристики сред передачи данных, тип кодирования, скорость передачи. Примеры: провод, сетевая плата
2. Организация канала связи. Проверка доступности физической среды. Формирование и распознавание каждого отдельного *кадра*. Проверка корректности передачи кадров по каналу при помощи контрольной суммы. Примеры: WiFi, Ethernet
3. Организация транспортной системы, определяет, куда именно хотим доставить *пакет*, но не организовывает саму доставку. Маршрутизация. Примеры: IP4, IP6

Доставка данных внутри сети: канальный уровень, между сетями: сетевой.

4. Обеспечивает передачу данных (*датаграммами* или *потоками*) с требуемыми гарантиями сохранения. Примеры: UDP, TCP
5. Обеспечивает синхронизацию и создание сеанса связи. Примеры: TCP, RPC
6. Представление информации: шифрование и сжатие.
7. Формирование *сообщения*. Пример: HTTP

2. Виды каналов связи: оптический, медный, беспроводной.

Кабельный (медный) вид связи.
Кабель состоит из проводников, заключенных в несколько слоев изоляции: электрической, механической и т. д.

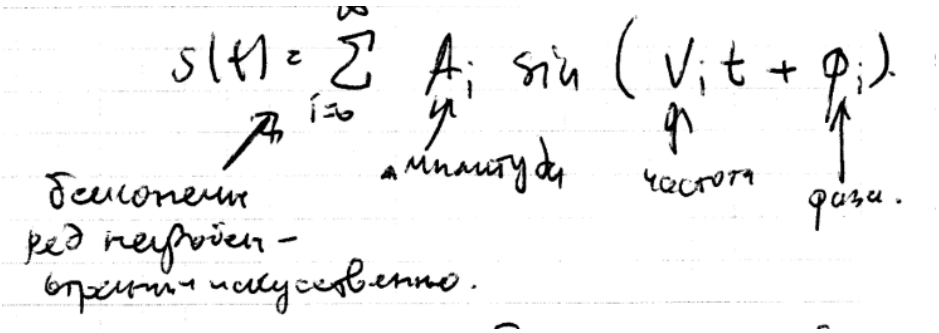
Симметричный медный кабель представляет из себя витую пару (скрученная пара одинаковых проводов). Скручивание снижает влияние внешних и взаимных помех на сигналы. Отличное соотношение качества к стоимости, простота монтажа.
Коаксиальный кабель состоит из несимметричных пар проводников. Каждая пара представляет собой внутреннюю медную жилу и соосную с ней внешнюю жилу, которая может быть полой медной трубой или изолированной оплеткой. Внешняя жила играет двойную роль: по ней и передаются сигналы, она же и служит защитой для внутренней жилы.
Медные кабели не очень используются, потому что медленновато и дороговато в сравнении с…

Оптический вид связи.
Оптоволокно состоит из тончайших (5-60 микрон) гибких стеклянных волокон, по которым распространяются световые сигналы. Он обеспечивает передачу данных с очень высокой скоростью (до 10 Гбит/с и выше) и к тому же лучше других типов передающей среды обеспечивает защиту данных от внешних помех. Но есть проблемы с монтажом, разъемами и стыками между кабелями.
По **одномодовому** кабелю распространяется только один луч света и очень пряменько, что позволяет передавать его дальше и устойчивее, но делать такой кабель дорого.
По **многомодовому** кабелю можно пускать сразу несколько лучей света под разным углом, получается мультиплекс. По таким кабелям далеко что-то передать не получится.
Схема кабеля похожа на колбасу!

Беспроводной вид связи.
Радиоканалы наземной и спутниковой связи образуются с помощью передатчика и приемника радиоволн. Существует большое разнообразие типов радиоканалов, отличающихся частотным диапазоном и дальностью канала.
Широкополосные частотные режимы (WAN) – только для лицензированных пользователей. Способ передачи цифровой, суть – симплекс (можно передавать только в одну сторону), топология – точка-точка.
Узкополосные частотные режимы (WLAN, например WiFi) – для кого угодно. Способ передачи аналоговый, суть – полудуплекс или дуплекс, топология – звезда.

3. Методы передачи данных

Сигнал S(t) – изменяющийся во времени физический процесс, отображающий передаваемое сообщение. Бывает непрерывный и дискретный.
Непрерывные сигналы можно разложить в ряд Фурье – получится **спектральное разложение**, которое характеризуется амплитудой, частотой и фазой.

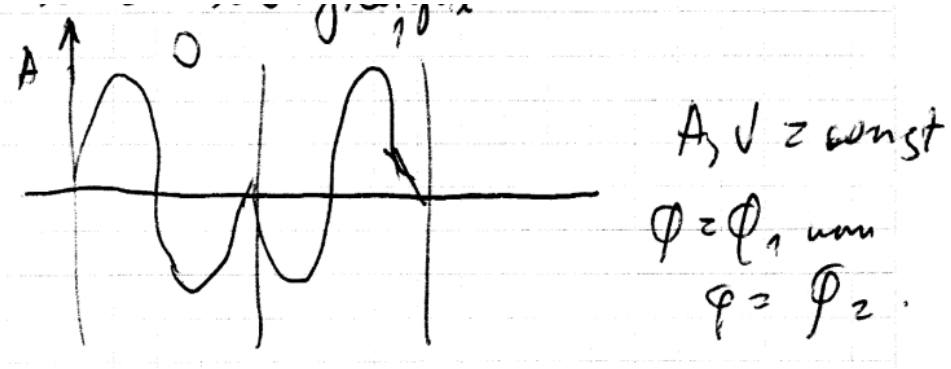


Выбор частоты

Амплитудно-частотная характеристика – отношение амплитуды на входе и выходе. Ей соответствует *коэффициент затухания* – 10 логарифмов ач характеристики. Соответственно **полоса пропускания** – промежуток частот, на которых ач характеристика достаточно большая.

Виды непрерывной модуляции

1. Амплитудная. При фиксированной частоте маленькая амплитуда = 0, большая = 1.
2. Частотная. При фиксированной амплитуде маленькая частота = 0, большая = 1.
3. Фазовая. Медленная и странная, так что ну ее.



4. Комбинированная. Будет проще находить ошибки. Например, А1 и ф1 = 1, А2 и ф2 = 0, а А1 и ф2 – ошибка.

Дискретная модуляция непрерывного сигнала

Замеряем, например, амплитуду с заданным периодом. Далее эти значения нужно дискретизировать – привести к заранее заданному дискретному множеству значений. Этим занимается аналого-цифровой преобразователь. Он решает систему уравнений по замерам (интересно, что это делается при помощи целых чисел, благодаря чему выигрываем в стоимости и скорости, но может быть погрешность). Замеры могут быть *абсолютными* и *относительными*.

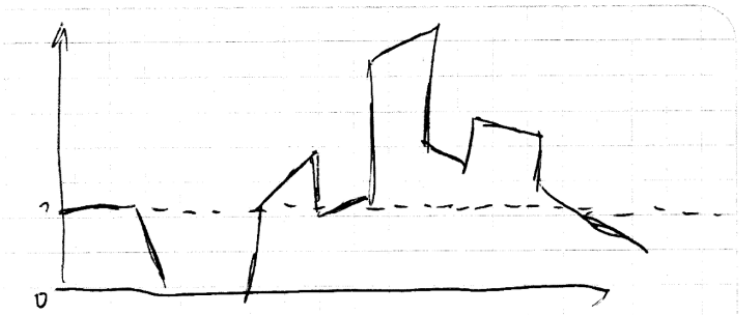
Дискретная модуляция основана на **теореме Найквиста-Котельникова**: аналоговая непрерывная функция, переданная в виде последовательности ее дискретных значений, может быть точно восстановлена, если частота дискретизации в два или более раз выше, чем частота самой высокой гармоники спектра исходной функции.

4. Цифровое кодирование

Несущая – периодический сигнал, параметры которого меняются во времени.
Кодирование – выбор способа представления информации в виде сигнала. Бывает потенциальное и импульсное.

Сигнал часто искажается. Причины:

1. Изменение параметров со временем (амплитуды и формы сигнала)
2. Погрешность дискретизации и внешние помехи
3. Блуждание несущей



4. Синхронизация. На больших расстояниях неравномерность скорости распространения сигнала может привести к тому, что тактовый импульс придет несколько позже или раньше соответствующего сигнала данных, и бит данных будет пропущен или считан повторно.

Задачи кодирования:

1. Минимизировать ширину спектра
2. Синхронизировать. Можно использовать *самосинхронизирующиеся* коды, сигналы которых содержат в себе информацию про момент считывания.
3. Распознавать ошибки
4. Минимизировать стоимость

Потенциальные коды

1. Без возврата к нулю (NRZ). Значения сигнала соответствуют битам. Много одинаковых значений подряд => синхронизация ломается.
2. С инверсией (NRZI). Сигнал поменялся = 1, остался прежним = 0. Тоже легко сломать синхронизацию.
3. Биполярный (AMI). 0 = 0, +v и -v = 1, причем при передаче последовательностей единиц +v и -v чередуется. Пофиксили длинные последовательности единиц, но не нулей.
4. MLT-3. Типа NRZI + AMI: меняем сигнал (циклически: 0, +v, 0, -v), если 1, не меняем если 0.
5. PAM-5. 4 уровня для сигнала, +1, обозначающий отсутствие сигнала. Передаем по два бита: 00, 01, 10, 11. Используется в Ethernet.

Импульсные коды

1. Манчестерское кодирование. Импульс вверх = 1, импульс вниз = 0. Если передаем 11, между импульсами нужно вернуться к 0. Вообще крутая штука: и

синхронизирует, и борется с блужданием несущей, и не нужно точных чисел/большого количества уровней.
2. Манчестерское кодирование с возвратом к 0. 1 = вверх-вниз, 0 = вниз-вверх.

Избыточные коды
Нужны, чтобы не было большого количества одинаковых символов подряд. Например, 4B/5B: заменяет каждые 4 бита на 5 новых, но в новой последовательности не больше 3 одинаковых символов подряд.

5. Управление каналом связи

Задачи канального уровня:
1. Формирование кадров
2. Распознавание кадров
3. Обработка ошибок
4. Управление потоком

Асинхронные протоколы оперируют не кадрами, а символами: <старт-символ> символ <стоп-символ>. Устаревшая штука. Каждый символ кидается отдельно.
Пример: XMODEM

Синхронные протоколы оперируют кадрами: <SYN>[Header][Data][Tail]. Кадры передаются потоком, что быстрее посимвольной передачи. Бывают символьно-ориентированные и бит-ориентированные протоколы.
Можно:
- Обозначать начало и конец каждого кадра. Чтобы в середине кадра символы-обозначители не встречались, используем экранирование в символьных протоколах и бит-стаффинг в битовых. Редко используется.
- Обозначать начало и конец каждого кадра запрещенными символами. Пример для манчестерского кодирования: J = неизменный низкий сигнал, K = неизменный высокий сигнал; старт кадра = JK0JK000, конец = JK1JK100. Недостаток способа – зависимость от типа кодирования.
- Обозначать начало и длину каждого кадра.

Протоколы могут устанавливать соединение перед передачей, а могут нет. Во втором случае кадры могут теряться, зато не нужно тратить время на открытие/разрыв соединения. В первом случае можно пользоваться выделенной линией (физический специальный канал) либо *коммутируемыми* протоколами (соединение устанавливается, когда надо, и разрывается по таймауту).

Что еще есть у коммутируемых протоколов: *аутентификация, callback* (клиент устанавливает соединение -> сервер рвет соединение -> сервер “перезванивает” клиенту и сам устанавливает соединение), *туннелирование* (передаем содержимое одного протокола через другой; можно объединить две сети, не зная, какие там протоколы).

6. Обнаружение и коррекция ошибок

Канальный уровень должен обнаруживать ошибки, связанные с искажением битов в принятом кадре или с потерей кадра, и по возможности их корректировать.
Большая часть протоколов умеет только находить ошибки, оставляя их исправление протоколам верхних уровней. Так работает, например, Ethernet. Вообще говоря, для сетей, в которых ошибки очень редки, разрабатываются протоколы, не предусматривающие процедур устранения ошибок.

Обнаружение ошибок
Основывается на передаче в составе кадра избыточной информации – **контрольной суммы**, по которой можно с некоторой степенью вероятности судить о достоверности принятых данных. Контрольная сумма вычисляется как функция от основной информации; принимающая сторона повторно вычисляет ее и в случае совпадения делает вывод о корректности переданных данных.

1. **Контроль четности**: XOR. Разбиваем данные на блоки, например, ксорим каждый байт или каждый восьмой бит (что лучше, потому что помехи часто затрагивают именно соседние биты). Важно: двойную ошибку мы упустим. Коэффициент избыточности для байтов: ⅛.
2. **Циклический избыточный контроль (CRC)**. Контрольная сумма – остаток от деления сообщения на образующий полином G(x). Обладает более высокой вычислительной сложностью, но его диагностические возможности гораздо выше + низкая избыточность. Например, для кадра Ethernet размером в 1024 байт контрольная информация длиной в 4 байт составит только 0,4%.

Алгоритм CRC

M(X) – сообщение, G(x) – образующий полином.

1. M1(x) = x^(deg G) * M(x)
2. R(x) = M1 mod G
3. T = M1 + R – отправляемое сообщение
4. R1 = T mod G. Если R1 == 0, ошибок нет.

Коррекция ошибок
Основана на повторной передаче, если кадр искажен. Отправитель нумерует отправляемые кадры и для каждого кадра ожидает от приемника **квитанции**. Время этого ожидания ограничено – если положительная квитанция не получена за фиксированное время, кадр считается утерянным.
Организация обмена квитанциями:

1. **С простоями**: источник ждет квитанцию и только после этого посылает следующий кадр или повторяет искаженный. Оченьб медленно.
2. **«Скользящее окно»**. Источник передает N кадров в непрерывном режиме без получения на эти кадры квитанций. Две стратегии:
    a. Выборочный повтор. Просим повторить только то, что не получили.
    b. Возврат на N. Просим повторить все, начиная с первого неполученного. Это оказывается лучше, потому что можно посылать меньше квитанций и менее сложные вычисления проводить.

7. Коммутация и мультиплексирование

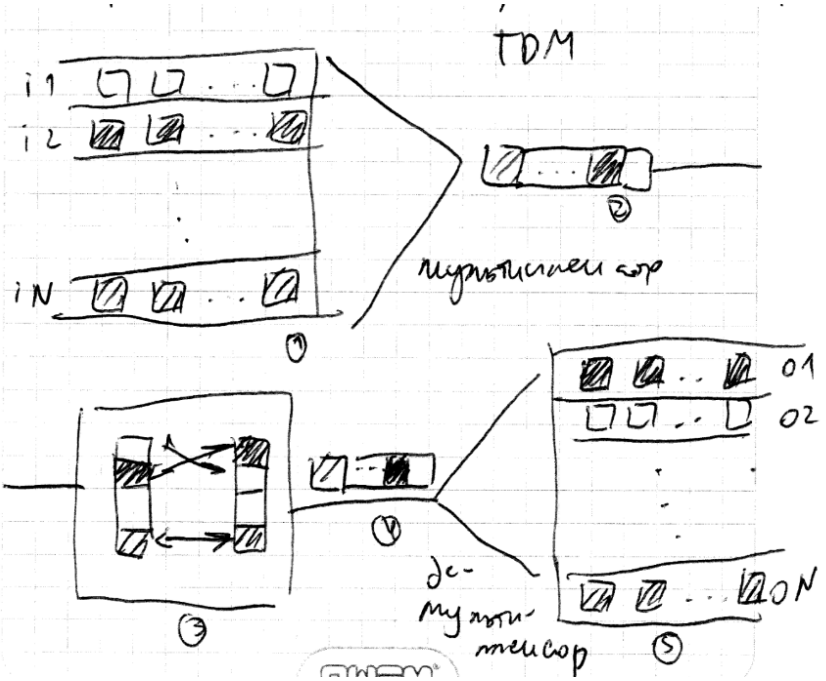
Задача соединения конечных узлов через сеть транзитных узлов называется задачей **коммутации**.
Схемы коммутации:

1. Каналов: создание между конечными узлами непрерывного составного физического канала из последовательно соединенных коммутаторами промежуточных канальных участков.
2. Пакетов: сообщения делятся на пакеты, которые передаются независимо. Отличие от каналов: коммутаторы имеют внутреннюю память для временного хранения пакетов, когда выходной порт занят. В этом случае пакет находится некоторое время в очереди в памяти, что позволяет эффективно передавать неравномерный трафик.
3. Сообщений: как пакетов, но передаем сообщения полностью (ждем, пока нам запишут сообщение, потом передаем). Медленно.

В сетях с **динамической** коммутацией разрешается устанавливать соединение по инициативе
пользователя. Коммутация выполняется на время сеанса связи, а затем связь разрывается. Благодаря этому каналы связи меньше простаивают.
В сетях со **статической** коммутацией паре пользователей разрешается заказать соединение на длительный период времени. Соединение устанавливается не пользователями, а персоналом, обслуживающим сеть. Режим постоянной коммутации в сетях с коммутацией каналов называется **сервисом арендуемых каналов**. В таких сетях не тратится время на установку соединения и аутентификацию.

Мультиплексирование – объединение нескольких каналов связи в рамках одного физического канала.
Типы:

- Частотное (FDM): разбиваем частотный диапазон на кусочки меньшего размера. В **уплотненном** канале между двумя FDM-коммутаторами одновременно передаются сигналы всех абонентских каналов, но каждый из них занимает свою полосу частот.
- По времени (TDM): каждый поток время от времени получает физический канал в свое распоряжение.



Мультиплексор принимает от каждого канала данные, составляет из них всех уплотненный кадр, отправляет дальше. TDM-коммутатор перетасовывает байты в уплотненном кадре, чтобы их порядок соответствовал порядку выходных соединений. Демультиплексор это распаковывает и каждому посылает по байту.

- По длине волны (WDM). Как FDM, только свет вместо тока.
- По коду (CDM). Каждый передатчик модулирует сигнал с применением присвоенного в данный момент пользователю числового кода, приемник, настроенный на аналогичный код, может вычленять из общей какофонии радиосигналов свою часть сигнала. Получается типа ортогональная система.

8. Основы Ethernet

Ethernet – самый распространенный на сегодняшний день стандарт локальных сетей. Для передачи информации по кабелю для вариантов, обеспечивающих пропускную способность 10 Мбит/с, используется манчестерский код. В более скоростных версиях применяются более эффективные избыточные логические коды.
Стандарты и модификации:

1. Network (1975)
2. II (1980) – стандарт IEEE 802.3
3. Fast Ethernet (1995) – стандарт 802.3u

4. Gigabit Ethernet (1998) – стандарт 802.3z

CSMA/CD

Метод доступа к среде передачи данных – метод коллективного доступа (MA) с опознаванием несущей (CS) и обнаружением коллизий (CD). Применяется в сетях с логической общей шиной. Все компьютеры такой сети имеют доступ к общей среде, поэтому она может быть использована для передачи данных между любыми двумя узлами сети. Одновременно все компьютеры могут немедленно получить данные, которые любой из компьютеров начал передавать в общую среду. Простота схемы подключения – один из факторов, определивших успех стандарта.

Алгоритм доступа к разделяемой среде:

1. Убедиться, что среда свободна, прослушав несущую частоту сигнала.
2. Если среда свободна, то узел имеет право начать передачу кадра.
3. Кадр всегда сопровождается *преамбулой*, которая состоит из 7 байтов 10101010, и 8-го байта 10101011 – начала кадра. Преамбула нужна для синхронизации.
4. Далее в кадре следует DEST и SRC адрес.
5. Также кадр содержит поле длины и контрольную сумму.
6. Все станции, подключенные к кабелю, могут распознать факт передачи кадра, и та, которая узнает свой адрес в заголовках кадра, записывает его содержимое в свой внутренний буфер, обрабатывает данные, а затем посылает по кабелю кадр-ответ.
7. После окончания передачи кадра все узлы сети обязаны выдержать паузу (Inter Packet Gap, *межкадровый интервал*).

Коллизии

Две станции одновременно пытаются передать кадр. *Коллизионный домен* – участок сети, где может возникнуть коллизия. В Ethernet это вся сеть. *Коллизионный диаметр* – расстояние между двумя самыми удаленными станциями.

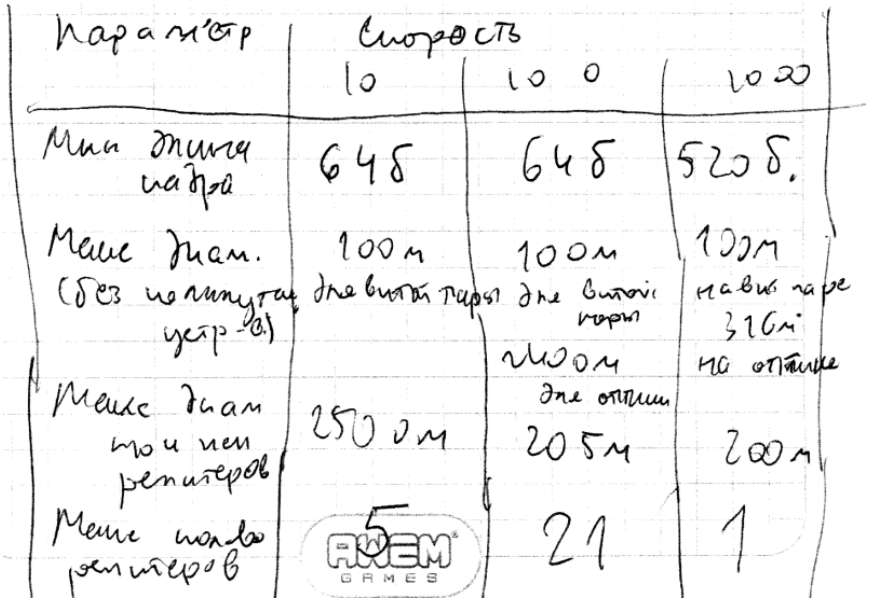
1. Если передаваемые и наблюдаемые сигналы для какой-то станции отличаются, то фиксируется обнаружение коллизии.
2. Станция, которая обнаружила коллизию, прерывает передачу своего кадра (в произвольном месте) и посылает в сеть специальный 32-битный сигнал – *jam-последовательность*.
3. Станция обязана прекратить передачу и сделать паузу в течение короткого случайного интервала времени.
4. Затем она может снова предпринять попытку захвата среды и передачи кадра.

Размер сети

Для надежного распознавания коллизий нужно, чтобы минимальное время передачи кадра было больше **времени двойного оборота** (время, за которое сигнал коллизии успеет распространиться до самого дальнего узла сети – по факту 2 * коллизионный диаметр). Тогда передающая станция успеет обнаружить коллизию, которую вызвал ее кадр, до того, как она закончит его передачу.

Исходя из этого выбираются ограничения на диаметр сети и размер кадра для разных скоростей передачи. Также требуется учитывать *затухание сигналов*; *повторители* увеличивают мощность передаваемых с сегмента на сегмент сигналов, в результате затухание уменьшается и можно использовать сеть большей длины. Есть также

микросегментация, *пакетная передача* (по много кадров сразу), *синхронная передача* (одновременно в две стороны).



Параметр	Скорость 10	100	1000
Мин длина кадра	64 б	64 б	520 б.
Макс длин. (без повторит. усилителей)	100 м для витой пары	100 м для витой пары	100 м на витой паре 316 м на оптике
		2000 м для оптики	
Макс длина тр и нет репитеров	250 м	205 м	200 м
Макс кол-во репитеров	5	21	1

Проблемы Ethernet:

- Безопасность
- Мобильность (на сетевом уровне)

9. Logical Link Control protocol

Протокол LLC (стандарт подкомитета 802.2) обеспечивает для локальных сетей транспортные функции (поддерживает передачу с и без установления соединения). Протоколы сетевого уровня передают данные для протокола LLC – свой пакет, адресную информацию об узле назначения, а также требования к качеству транспортных услуг. LLC помещает пакет в свой кадр, дополняя его служебными полями. Далее кадр передается протоколу уровня MAC, который упаковывает его в свой кадр (например, кадр Ethernet):
[Ethernet HEADER] [Кадр LLC] [Контрольная сумма] = кадр Ethernet

Типы процедур
LLC1 = UDP, LLC2 = TCP, LLC3 – что-то между (соединение не устанавливается, но есть подтверждение получения кадра).

Структура кадра
Кадры уровня LLC (блоки данных, PDU) бывают трех типов:

1. Информационные. LLC2, собственно данные. Нумеруются в режиме скользящего окна.
2. Управляющие. LLC2, квитанции.
3. Ненумерованные. LLC1, 3 – все, LLC2 – установление и разъединение соединения, информирование об ошибках.

Формат: [DSAP=destination service access point] [SSAP] [Control] [Data]

С помощью **DSAP и SSAP** в узле осуществляется демультиплексирование кадров: общий поток кадров, предназначенных для конкретного узла, распределяется далее по различным протоколам сетевого уровня.

Общение в LLC2

1. Ненумерованные кадры
    - Poll Bit: это запрос, на который нужно ответить, или сам ответ?
    - Команда ненумерованного кадра, например, запрос на установку и разрыв соединения.
2. После установления соединения данные и положительные квитанции начинают передаваться в информационных кадрах. Логический канал дуплексный.
    - В информационных кадрах имеется поле для номера отправленного кадра.
    - Скользящее окно размером в 127 кадров, для их нумерации циклически используются числа от 0 до 127.
    - Приемник помнит номер последнего принятого кадра и кадра, который ожидается следующим V. В информационных кадрах имеется для этого поле. Если приходит ответ, в котором помер посланного кадра совпадает с V, то он считается корректным (если корректна его контрольная сумма). Иначе кадр отбрасывается и посылается отрицательная квитанция с V.
    - При приеме отрицательной квитанции передатчик повторяет кадр V и все, что за ним.
3. В состав супервизорных кадров входят:
    - отказ
    - приемник не готов (RNR)
    - приемник готов (RR)

Команда RR с номером часто используется как положительная квитанция, а команда RNR – для замедления потока кадров, поступающих на приемник (иногда он не успевает обработать быстро приходящие кадры). Получив RNR, передатчик ждет, пока снова не получит RR. Таким образом управляем потоком данных, что важно для коммутируемых сетей, в которых нет других тормозящих механизмов.

10. Virtual LAN

Виртуальная сеть – группа узлов сети, трафик которой на канальном уровне полностью изолирован от других узлов сети, т. е. передача кадров между разными виртуальными сетями на канальном уровне невозможна. При этом обмен данными между устройствами внутри VLAN происходит так, будто они подключены к одному кабелю.

Основные функции:

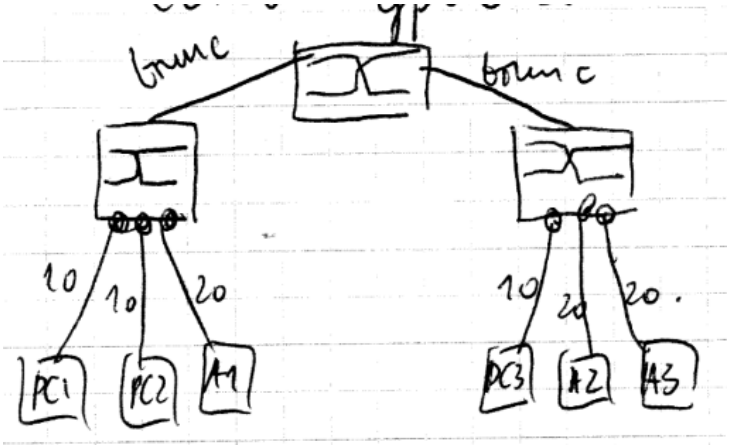
- Ускорение срочного трафика
- Разбиение на логические группы
- Упрощение управления

Можно реализовать на:

- Физическом уровне
- Канальном уровне
- Сетевом уровне (концентраторы + маршрутизаторы)

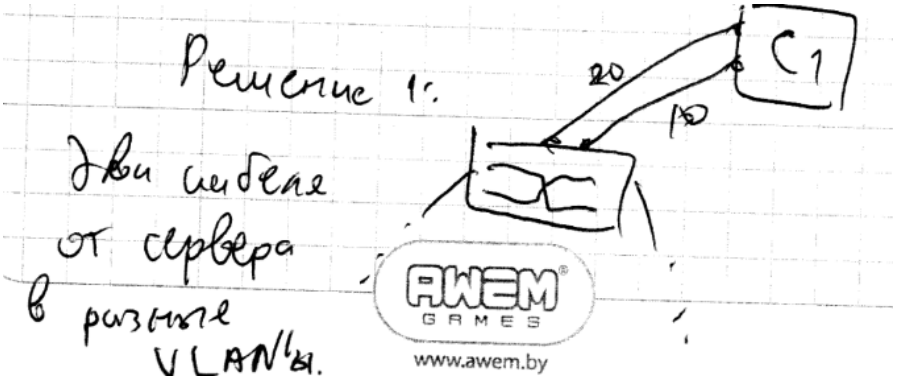
Управлением VLAN на канальном уровне занимаются **коммутаторы**.

Кадр при использовании VLAN содержит дополнительную информацию – тег виртуальной сети, который включает идентификатор VLAN, приоритет, номер сети. Эта информация добавляется в коммутаторе; конечные узлы про VLAN типа ничего не знают.



Trunk-соединение передает между коммутаторами только те пакеты, VLAN которых есть с обоих сторон соединения. Именно на этих соединениях используются теги кадров.

VLAN на коммутаторе
На портах коммутатора: Каждый порт приписывается той или иной виртуальной сети. Можно к одному порту подключить целый сегмент, построенный на основе повторителя (но все устройства в этом сегменте должны быть в одной виртуальной сети). Легко настраивается.
По MAC-адресам. Убирается зависимость от числа портов, коммутаторов, физического расположения устройств. Сложнее настраивать. Может быть проблема при необходимости дать доступ к нескольким сетям сразу.



Еще можно сказать конечным станциям, в каком они VLAN, или маршрутизатор подключить, но это не оч :(
VLAN-3: Через DHCP даем IP, потом по IP понимаем, какой VLAN (дорогое)
Native VLAN: типа дефолтный

11. FDDI

FDDI (Fiber Distributed Data Interface) – первая технология локальных сетей на оптоволокне. Основан на протоколе Token Ring. В основе доступа к общей среде лежит не случайный алгоритм, как в Ethernet, а детерминированный, основанный на передаче станциям права на использование в определенном порядке, с помощью кадра специального формата – **маркера**. Для кодирования информации используется 4B/5B.

Задачи:

- скорость передачи данных – 100 Мбит/с.
- покрытие большой территории – до 200 км.
- высокая отказоустойчивость сети.
- максимально эффективно использовать потенциальную пропускную способность сети.

Сеть FDDI строится на основе двух оптоволоконных колец, которые образуют основной и резервный пути передачи данных между узлами сети. Каждая станция непосредственно связана только с двумя своими соседями.
В нормальном (*“сквозном”*) режиме работы сети данные проходят только по первичному кольцу. Когда часть первичного кольца не может передавать данные, первичное кольцо объединяется со вторичным (режим *“сворачивания”*). Объединение производится средствами концентраторов и/или сетевых адаптеров FDDI. Для упрощения данные по первичному кольцу всегда передаются в одном направлении, а по вторичному – в обратном.
Также FDDI подразумевает работу с асинхронным и синхронным (срочным) трафиком.
Отличительной особенностью технологии FDDI является **уровень управления станцией** (SMT). Он выполняет функции по управлению и мониторингу всех остальных уровней стека прото-
колов FDDI. Все узлы обмениваются специальными кадрами SMT для управления сетью.

Алгоритм доступа к разделяемой среде

1. Станция получает маркер. Если ей нечего отправлять, передает его дальше.
2. Иначе станция захватывает маркер, что дает ей право доступа к физической среде для передачи своих данных.
3. Станция отправляет кадр данных, снабженный адресом назначения и источника. Кадр также содержит контрольную сумму и ограничители начала и конца.
4. Переданные данные проходят по кольцу от одной станции к другой; станции просто повторяют кадр.
5. Если кадр проходит через DEST, то эта станция копирует кадр в свой внутренний буфер и вставляет в кадр подтверждение приема, отправляя его дальше.
6. Станция, отправившая кадр, при получении с подтверждением приема изымает кадр из кольца и передает маркер, давая другим станциям сети возможность передавать данные.

Время владения разделяемой средой ограничивается временем удержания маркера, которое зависит от загруженности сети. Существует также алгоритм *раннего освобождения маркера*, где станция передает маркер, не дожидаясь возвращения кадра с подтверждением.
Существуют также управляющие кадры и *активный монитор*. Активный монитор следит за тем, чтобы маркер не терялся. Управляющие кадры позволяют подтверждать существование активного монитора и назначать новый в случае его отказа.
Физически топология сети может быть звездой при использовании концентраторов, объединяющих станции. Обычно концентраторы подсоединены к обоим кольцам, а станции – только к одному.

Асинхронная передача

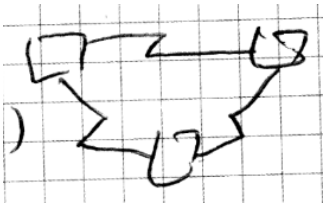
1. При инициализации сети задается максимальное время оборота маркера T0.
2. При приходе маркера станция, передающая асинхронный кадр, вычисляет фактическое время оборота кадра TRT.
3. Станция захватывает маркер только если TRT < T0, т. е. сеть несильно загружена.

Максимальное количество станций двойного подключения в кольце – 500, максимальный диаметр двойного кольца – 100 км. Максимальные расстояния между соседними узлами для многомодового кабеля равны 2 км, для одномодового 20-50 км.

12. Беспроводные сети

IEEE 802.11, Wi-Fi. Основной компонент – **сеть с базовым набором услуг** (BSS) – набор беспроводных устройств, разделяющих среду передачи и имеющих одинаковые характеристики доступа к среде (частота и схема модуляции каналов).

Два типа топологий BSS
Ad-Hoc – независимая, точка-точка. Алгоритм доступа децентрализованный (но в каждый момент времени существуют устройства, берущие на себя управляющую роль), сеть создается самопроизвольно на небольшой период времени. Нет средств взаимодействия с другими сетями.



Инфраструктурная – через базовую станцию (точку доступа). AP обычно подключается к другим сегментам сети проводом по Ethernet; как правило, они также выполняют роль коммутатора.

ESS (Extended …) – объединение нескольких точек доступа с организованной связью между ними. Получается распределенная система. Устройства могут переходить между разными точками доступа этой системы (мобильность).

Помехи
В беспроводных сетях влияние ошибок и помех особенно велико. Что делают в связи с этим:

- Расширение спектра. Каждый передаваемый бит информации представляется в виде псевдослучайной последовательности кодовых символов. Это реализуется сложением исходной последовательности битов с кодовой расширяющей последовательностью. В результате спектр сигнала расширяется в число раз, равное длине расширяющей последовательности. В итоге уменьшается влияние шума и других участников передачи на сигнал.
- Алгоритм доступа основан на методе простоя, а не скользящего окна.
- Используются избыточные коды с детекцией ошибок.
- Метод доступа к общей среде – CSMA/CA (Collision Avoiding)

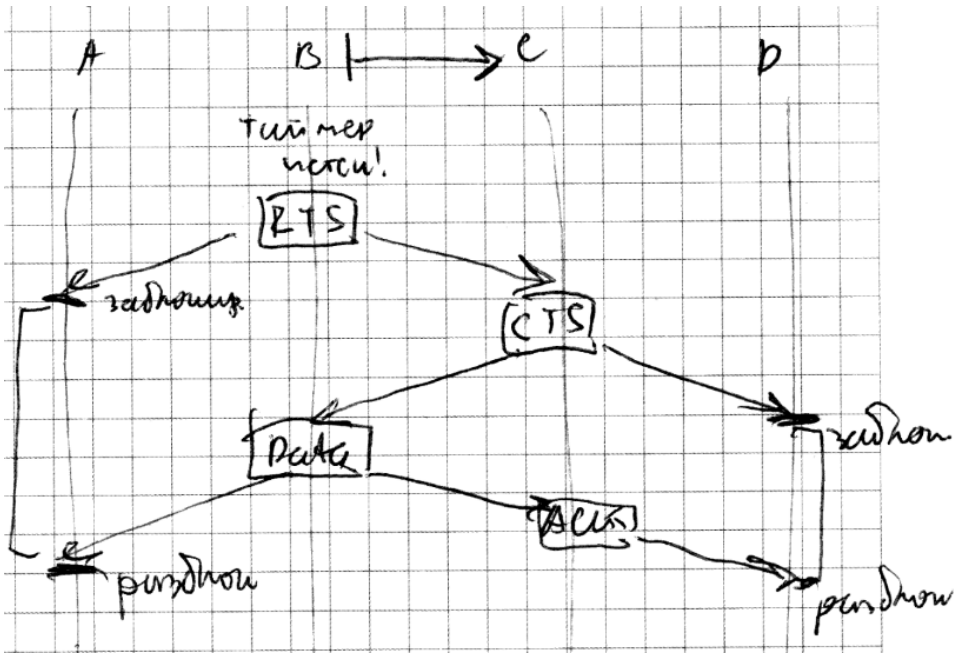
Разные стандарты отличаются скоростными режимами и используемыми частотами. Примеры:

- 802.11 1.2 Мбит/с
- 802.11b 11 Мбит/с
- 802.11ac (WiFi-5) 350-3500 Мбит/с в зависимости от частоты
- 802.11x (WiFi-6) 10.5 Гбит/с

CSMA/CA
В **распределенном режиме** доступа (DCF):

1. Слушаем несущую, пока не освободится среда
2. Запускаем таймер на максимальное время передачи кадра * рандомное число. За счет этого уменьшается вероятность коллизии (разные станции должны нарандомить одинаково). Интервал для рандомных чисел – *конкурентное окно* (чем больше, тем меньше вероятность коллизии).
3. По истечении таймера отправляем кадр, если среда свободна. Если пока таймер отсчитывался, кто-то начал передачу, таймер замораживается, пока среда не освободится.
4. Если коллизия все же случилась, повторяем всю процедуру заново.

Существует проблема *невидимой станции*: два отправителя находятся далеко друг от друга и одновременно будут думать, что среда свободна. Иногда это решается дополнительными квитанциями Ready To Send / Clear To Send



В **централизованном режиме** доступа (PCF) – если нужно передать срочный трафик. Во-первых, DCF челы в такой схеме должны ждать так долго, что гарантируется, что за это время срочный чел захватит среду, если это нужно. Во-вторых, точка доступа (*арбитр*) может передать флаг-начало контролируемого периода, во время которого DCF в принципе не захватывает среду.

13. Структуризация сети: мосты и коммутаторы, их виды

Для структуризации сети разделяемую среду разбивают на сегменты поменьше с меньшим количеством узлов, в каждом из которых своя разделяемая среда. Здесь важны: **полезная пропускная способность** (общая пропускная способность / число узлов) и **коэффициент использования** (трафик / max пропускная способность). Отдельно рассматривается физическая и логическая структуризация.
Структуризация возможна на *канальном* уровне (мосты и коммутаторы) и *сетевом* (маршрутизаторы). Часто используют комбинированный способ – небольшие сегменты объединяются канально в более крупные подсети, которые соединяются маршрутизаторами.

Мосты и **коммутаторы** выполняют одни и те же функции (структуризация и уменьшение конкурирования за трафик) при помощи одних и тех же алгоритмов; коммутатор = мультипроцессорный мост.
Прозрачный мост: не зависит от технологии LAN и его существование незаметно для участников сети. Строит *адресную таблицу* (устройство/порт/динамическая ли запись), по которой решает, что куда отправлять, на основании пассивного наблюдения за трафиком. Алгоритм построения:

1. Получаем пакет с порта A.
2. Вносим адрес отправителя пакета и порт A в адресную таблицу.

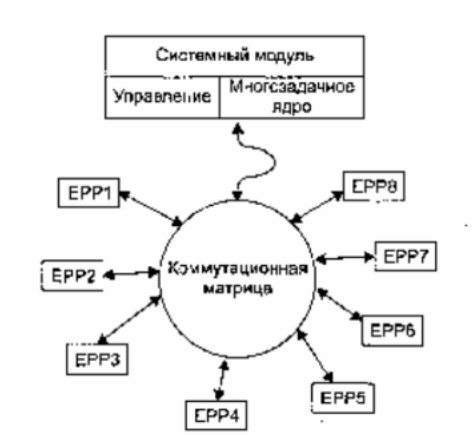
3. Если адрес получателя есть в таблице не на порте А, направляем его на нужный порт, если на порте А, то отбрасываем (происходит фильтрация трафика).
4. Если адреса нет, дублируем пакет на все порты, кроме А (*алгоритм затопления*)

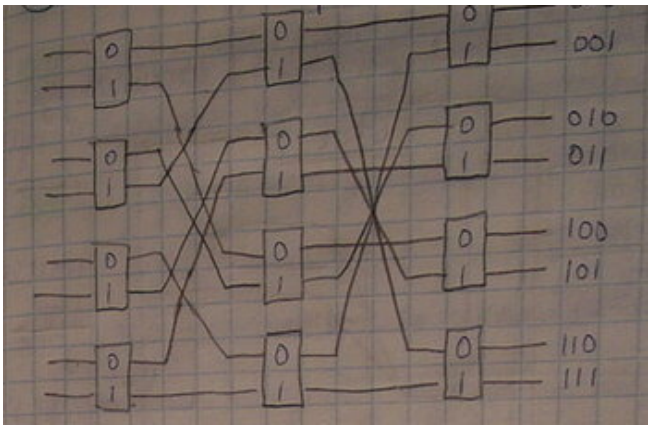
Мост с маршрутизацией от источника: станция-отправитель помещает в посылаемый кадр всю адресную информацию о промежуточных мостах, которые должен пройти кадр; мост просто распознает свой адрес и пересылает кадры по указаниям сверху.

Полная буферизация: в буфер помещается фул кадр, мы сначала принимаем его полностью, проверяем, а потом кидаем дальше. Получается долго, зато не транслируем коллизии и всякую фигню.
Буферизация на лету: в буфер помещается заголовок кадра, которого достаточно, чтобы понять, куда его кидать.

Мультипроцессинг в коммутаторах

1. Коммутационная матрица.





Из каждого порта есть путь в другой по своим транзисторам, не пересекающийся с другими путями. Це быстро, но много портов не получится сделать.

2. Общая шина. Сначала нужно фул прочитать кадр в буфер порта, потом положить на общую шину, добавив номер порта-получателя. Медленно, надо много памяти, зато просто.

3. Разделяемая память. Кадр пишется в общую память из порта. Довольно сложный механизм памяти, поэтому дорого.

Проблемы мостов
- Нет защиты от широковещательной фигни
- Нельзя сделать петлеобразную структуру сети

14. Коммутация с полной буферизацией и «на лету». Spanning Tree Protocol.

Буферизация и проблемы мостов выше.

Альтернативные пути между станциями нужны для резерва и для разгрузки сети. Но базовые протоколы канального и физического уровня поддерживают только древовидные, то есть не содер-
жащие замкнутых контуров топологии связей. Чтобы обеспечить альтернативные маршруты, нужен **ST**. Его реализуют прозрачные мосты, оставаясь при этом довольно простыми.
Алгоритм обеспечивает поиск в графе сети древовидной топологии связей с единственным путем между любыми двумя вершинами (коммутатор или сегмент сети), причем минимально возможного расстояния (но не оптимального в общем случае).

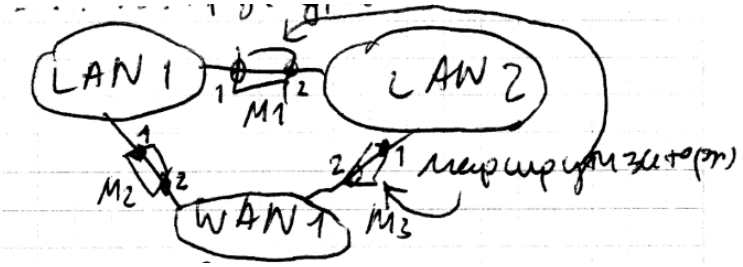
1. Выбирается коммутатор-корень дерева (произвольно или – лучше – вручную админом)
2. Каждый коммутатор находит корневой порт – порт с минимальным корня. Это делается при помощи управляющих пакетов BPDU, которые рассылает корень, а остальные коммутаторы повторяют, запоминая расстояние (типа bfs).
3. Из всех коммутаторов каждого сегмента выбирается чел с корневым портом с минимальным расстоянием – “назначенный мост”.
4. Все порты, кроме корневых и назначенных, блокируются.

Возможные состояния порта:
- Blocking – только слушаем только BDPU
- Listening – слушаем и отправляем только BDPU
- Learning – как Listening, но еще и можем вносить изменения во внутреннюю таблицу
- Forwarding – все слушаем и отправляем

BPDU
- тип (например, заявка на корень)
- poll флаг: это запрос или ответ?
- id корня, расстояние до корня, id текущего коммутатора, id порта
- время жизни сообщения, максимальное время жизни, интервал посылки BDPU

15. Идея Internetworking. Маршрутизация и функции маршрутизатора

Internetworking – межсетевое взаимодействие. Таким образом получаем сеть как совокупность подсетей меньшего размера.



Маршрутизатор – устройство, которое определяет направление движения пакета между сетями. **Маршрут** – последовательность маршрутизаторов. Маршрутизация – действия по перенаправлению пакетов.

Задачи, которые канальный уровень решить не может (подключается сетевой):

- унификация и абстракция взаимодействия подсетей
- разные технологии в подсетях
- масштабируемость системы

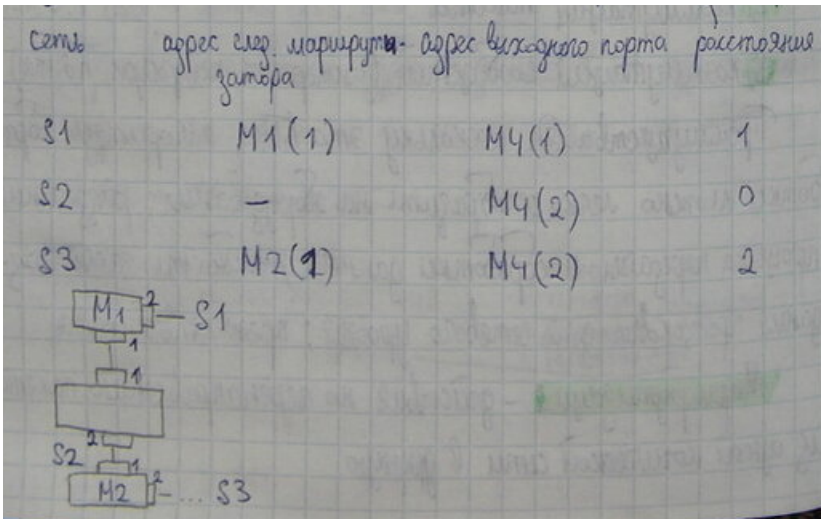
Основные принципы маршрутизации:

1. Каждой сети ставим в соответствие номер (адрес)
2. Каждому порту маршрутизатора ставим в соответствие номер присоединенной к нему подсети
3. В сети существует несколько маршрутов в одну точку

Функции маршрутизатора:

1. Принятие и распределение данных по портам
2. Инкапсуляция данных сетевого уровня
3. Принятие решения по направлению пакета; фильтрация
4. Ведение таблиц маршрутизации

Таблица маршрутизации
DestNet / Next Hop / Port / Metric



Маршрут по умолчанию – маршрут, который выбирается, если нет записи в таблице.

Типы маршрутизации

1. Многошаговая (от источника): станция сама говорит, как кидать
2. Одношаговая (распределенная): маршрутизаторы знают Next Hop

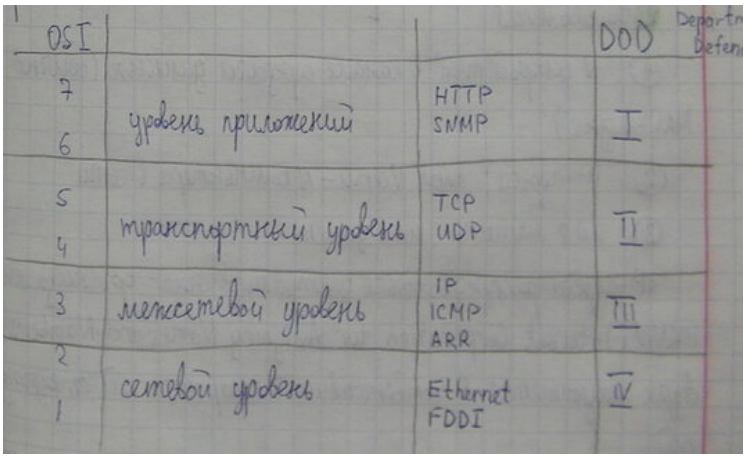
Методы построения таблицы (+ перечислить достоинства и недостатки каждого)

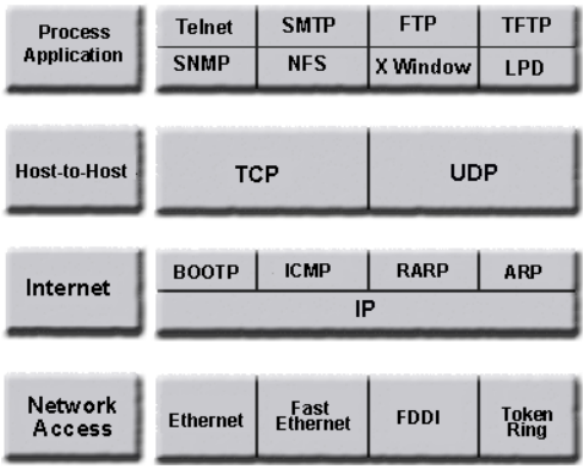
1. Статически
2. Просто (почти не нужна таблица маршрутизации)
    a. Случайный
    b. Лавинный (направляем всем, кроме отправителя)
    c. Предыдущий опыт (для обратных пакетов)
3. Динамически (обновляем таблицу маршрутизации)
    a. Дистанционно-векторный (пример: RIP)
    b. По состоянию каналов связи (пример: OSPF)

Можно комбинировать несколько сразу (но это сложнее)

16. Модель DOD. Классовая и бесклассовая адресация

Модель DOD основана на 4 уровнях





Классовая адресация

Основана на разделении всех IP-адресов на 5 классов.

Class	Leading bits	Size of *network number* bit field	Size of *rest* bit field	Number of networks	Addresses per network	Total addresses in class	Start address	End address
Class A	0	8	24	128 (2^7)	16,777,216 (2^{24})	2,147,483,648 (2^{31})	*0.0.0.0*	*127.255.255.255*[a]
Class B	10	16	16	16,384 (2^{14})	65,536 (2^{16})	1,073,741,824 (2^{30})	*128.0.0.0*	*191.255.255.255*
Class C	110	24	8	2,097,152 (2^{21})	256 (2^8)	536,870,912 (2^{29})	*192.0.0.0*	*223.255.255.255*
Class D (multicast)	1110	not defined	not defined	not defined	not defined	268,435,456 (2^{28})	*224.0.0.0*	*239.255.255.255*
Class E (reserved)	1111	not defined	not defined	not defined	not defined	268,435,456 (2^{28})	*240.0.0.0*	*255.255.255.255*[b]

В классе A есть зарезервированные адреса: 0.0.0.0 (маршрут по умолчанию) и 127.x.x.x (loopback). Также во всех сетях зарезервирован наибольший и наименьший адрес.
Сейчас классовая адресация почти не используется из-за недостаточной гибкости. В сетях классов A и B слишком много доступных узлов, такое как правило не нужно.

Бесклассовая адресация (CIDR)
Адреса сетей и машин теперь разделены при помощи маски переменной длины. Это позволяет, например, уменьшить размер таблиц маршрутизации (храним только адреса сетей), более эффективно распределять узлы по пользователям. **Маска** имеет битовый вид сначала x единиц, потом y нулей (для краткости можно задать просто числом единиц). Тогда из адреса a и маски m адрес сети получается как a & m. Поставщик адресов выделяет размер маски соответственно требуемому числу адресов (числу станций).

17. Протоколы IP (v4,v6) и ARP

IP4
IP – протокол межсетевого взаимодействия. Передает пакеты между сетями. В каждой сети, лежащей на пути перемещения пакета, он при помощи внутренних средств транспортировки этой сети передает пакет на маршрутизатор, ведущий к следующей сети, или получателю.
Работает *без установления соединения*. Нет надежной доставки сообщений: нет квитирования, нумерации пакетов, повторной передачи. Надежность обычно обеспечивает TCP, работающий над протоколом IP.
Важная особенность IP: динамическая фрагментация пакетов при передаче между сетями с разными максимальными значениями длины поля данных (MTU).

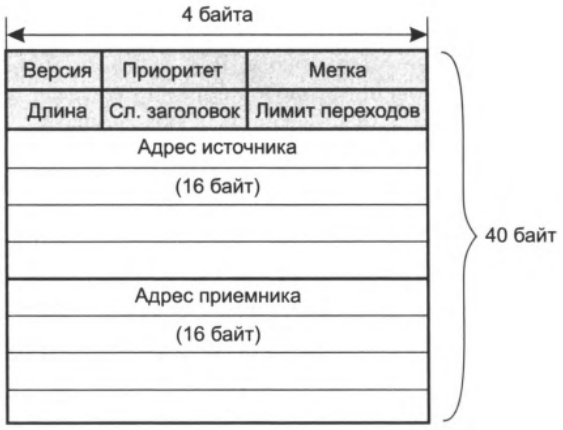
Заголовок IP

- Номер версии = 4 или 6
- Тип сервиса: приоритет + выбор, что максимизировать: D – скорость, T – пропускную способность, R – надежность
- Вся вторая строчка – для фрагментации. Фрагменты одного исходного пакета будут иметь одинаковый ID; флаги указывают, можно ли фрагментироать, и последний ли это фрагмент; смещение показывает место в исходном пакете
- Протокол верхнего уровня, примеры: 6 = TCP, 17 = UDP
- Параметры – используются в основном при отладке. Можно задать маршрутизацию от источника или записывать пройденный путь

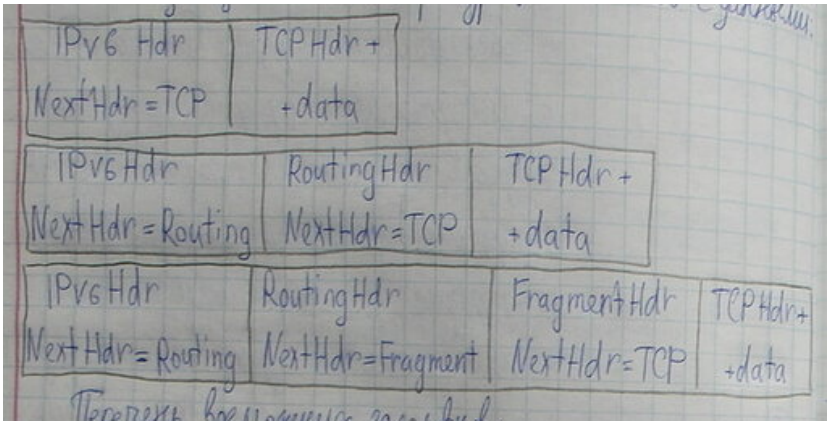
IP6

- 128-битное пространство адресов, 16-ричная система счисления. Например FEDC:0A98:0:0:0:0:1234:5678
- типы адресов (при этом, вообще говоря, все на CIDR):
    - Unicast
    - Anycast – один адрес на несколько интерфейсов (кто первым услышит, тот забирает)
    - Multicast – вещаем всем, но лучше чем broadcast
    - Также зарезервированы 0 и loopback
- Совмещение с IP4: 0:0:0:0:0:FFFF:[IP4-адрес]
- Каждый интерфейс может иметь несколько адресов для разных групп
- Область действия адреса: нельзя послать пакет с адресом частной сети в глобальный интернет
- Автоконфигурирование: можно настраивать IP интерфейсов даже без DHCP

Заголовок IP6
Основной заголовок:



Метка потока: можно настраивать поток пакетов, правила передачи которого отличаются от остальных.
Остальные части заголовка работают как конструктор и там много всяких дополнительных вещей:



ARP
Address Resolution – преобразовывает локальные адреса в глобальные. **ARP-таблица** – соответствие между MAC и IP (записи в ней могут быть статическими и динамическими). Если в ARP-таблице нет нужной записи, мы присылаем всем требуемый IP; если кто-то распознал в нем свой, он об этом сообщает. Таблица периодически сбрасывается, чтобы не хранить устаревшую информацию, и содержит только записи о тех, кому что-то приходит, а не о всех, поэтому ее еще называют *ARP-кэшем*.
Есть еще RARP (Reverse ARP) – кто-то раздает IP-адреса узлам, а не спрашивает их.

18. Протокол DHCP

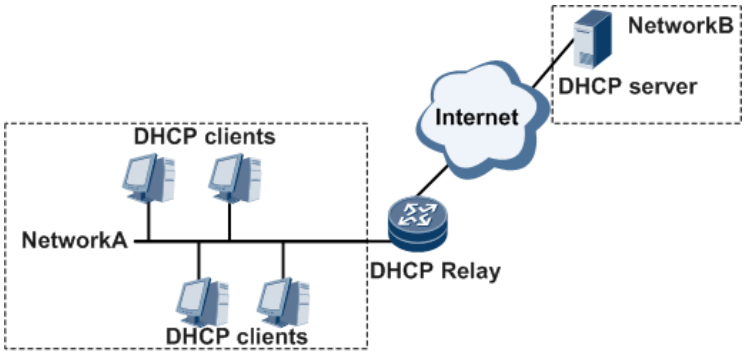
Dynamic Host Configuration Protocol – обеспечивает автоматическое динамическое распределение адресов.
Алгоритм общения:

1. Discover: клиент посылает широковещательный канальный запрос в поисках сервера.
2. Offer: сервер отвечает, говорит свой IP
3. Request: клиент просит дать ему IP

4. (N)Ack: сервер дает IP либо говорит, что есть проблемки
5. Еще типы запросов: Decline (клиент отказывается от выданного IP), Release (требование освободить адрес), Inform

Итого, что есть в пакете:
- тип операции (запрос/ответ), канальный протокол, длина заголовка
- случайный идентификатор связи (вместо IP, его же у клиента нет)
- адрес клиента (если есть), предлагаемый кленту адрес, адрес сервера, MAC-адрес, адрес relay-агента

DHCP relay позволяет поставить один DHCP-сервер на много подсетей:



Типы конфигурации DHCP-сервера
1. Ручная статическая
2. Автоматическая статическая: DHCP сам выбирает кому какой адрес давать, но делает это только один раз; один и тот же клиент всегда будет получать один адрес, и этот адрес нельзя дать кому-то еще
3. Автоматическая динамическая: адрес выдается на ограниченное время (*время аренды*), поэтому можно даже использовать пул адресов меньший, чем число клиентов

19. Методы маршрутизации. Функции маршрутизатора

Ну во-первых вопрос 15 :)

Протоколы маршрутизации могут быть:
распределенными: каждый маршрутизатор строит таблицу независимо от других на основании получаемое информации
централизованными: есть один маршрутизатор, который собирает информацию о топологии сети. потом он может отправить каждому таблицу, а может отправить узлам требуемые маршруты (типа маршрутизация от источника)

Требования к протоколам маршрутизации:
- рациональность (чтобы было +- кратчайшее расстояние)
- относительно простая реализация
- сходимость (получение корректной таблицы маршрутизации за конечное время)
- согласованность таблиц разных маршрутизаторов

Несколько уровней работы маршрутизатора:

1. Уровень протоколов маршрутизации
    - Ведение таблицы
2. Уровень сетевого протокола
    - Удаление поврежденных пакетов
        - Проверка чексуммы
        - Проверка TTL
    - Анализ и модификация заголовка (напр. TTL)
    - Фильтрация трафика
    - Ведение очереди (типа если сеть занята, пока не кидаем в нее пакет, а храним у себя)
    - Определение маршрута по таблице (не нашли – удаляем пакет)
3. Уровень сетевого интерфейса
    - Передача пакета на канальный уровень, где по номеру порта происходит коммутация. Интересное отличие маршрутизатора от коммутатора: у маршрутизатора у каждого порта свой MAC-адрес
    - Формирование кадра
    - Преобразование сетевого адреса в локальный
    - Отправка и распознавание кадров

20. Динамическая маршрутизация. Протокол RIP

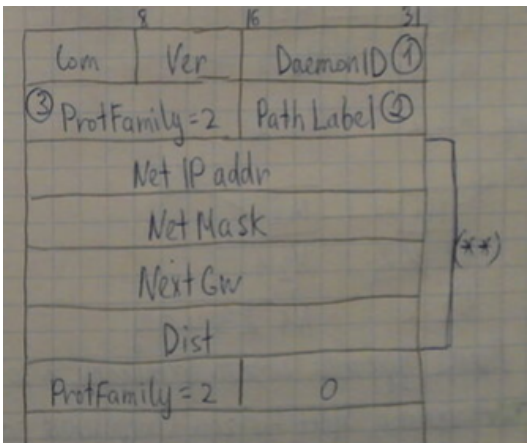
Динамическая маршрутизация в целом – вопрос 15.

RIP – дистанционно-векторный протокол маршрутизации. Каждый маршрутизатор периодически рассылает вектор расстояний до известных ему сетей. *Метрика* расстояния может быть числом хопов, может быть сложнее (например, учитывать пропускную способность). При получении вектора от соседа увеличиваем указанные в нем расстояния на расстояние до соседа, добавляем к вектору расстояния, о которых узнали из других мест, а затем рассылаем новое значение вектора по сети.
Из имеющегося вектора выбирается кратчайший по метрике маршрут до каждой сети. Чел, передавший информацию об этом маршруте, отмечается в таблице как next hop.
Чтобы адаптироваться к изменениям в сети, маршрутизаторы продолжают периодически рассылать векторы. Если информация о сети перестала поступать в течение определенного времени, то соответствующая запись из таблицы удаляется.
Дистанционно-векторные алгоритмы хорошо работают только в небольших сетях. В больших сетях они засоряют линии связи интенсивным служебным периодическим трафиком, к тому же изменения обрабатываются с задержкой и не всегда оптимально, так как нет точного представления о топологии в сети.

RIP1 и RIP2

- RIP1 не умеет в CIDR
- RIP1 со всем разговаривает broadcast-ом, а RIP2 умеет еще и в multicast

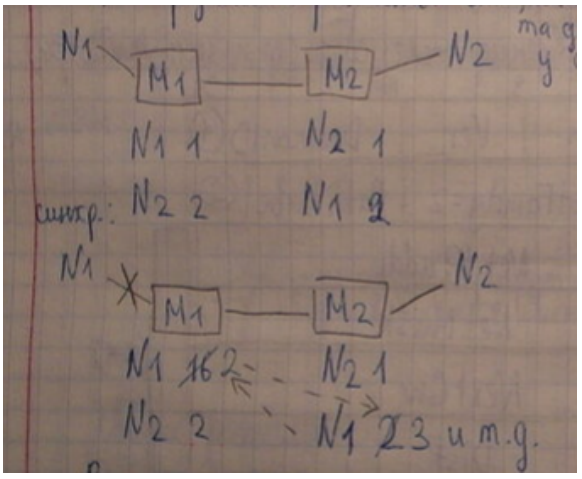
Пакет
Заголовок – часть UDP-пакета.



Этапы/режимы общения

1. Инициализация
2. Получение запроса (высылаем свою таблицу)
3. Получение отклика (корректируем таблицу)
4. Регулярная коррекция

Ложные маршруты



Split Horizon – не посылаем информацию о маршруте челу, от которого она получена. Пофиксит картинку выше, но если такая фигня будет в цикле из нескольких, проблемы останутся.

Триггерные обновления: получив данные об изменении метрики, не ждем истечения периода передачи таблицы, а передаем данные об изменившемся маршруте сразу. Триггерные объявления также делаются с некоторой задержкой (иначе очень сильно захламляется трафик). Из-за задержки все еще можно получить по жопе.

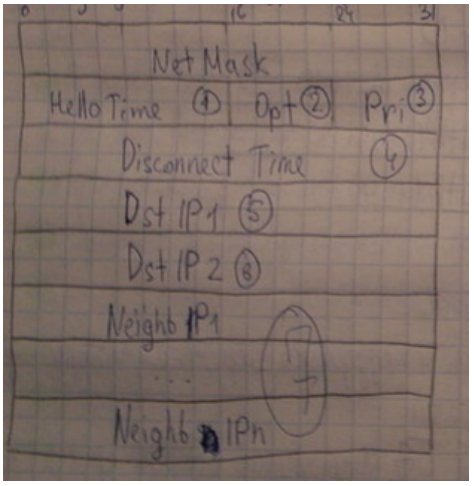
Замораживание изменений: (самый норм прием) тайм-аут на принятие новых данных о сети, которая только что стала недоступной. Это предотвращает принятие устаревших сведений от маршрутизаторов, находящихся на некотором расстоянии от отказавшей связи. Предполагается, что за тайм-аут они вычеркнут маршрут из своих таблиц, так как не получат о нем новых записей.

21. Протокол OSPF

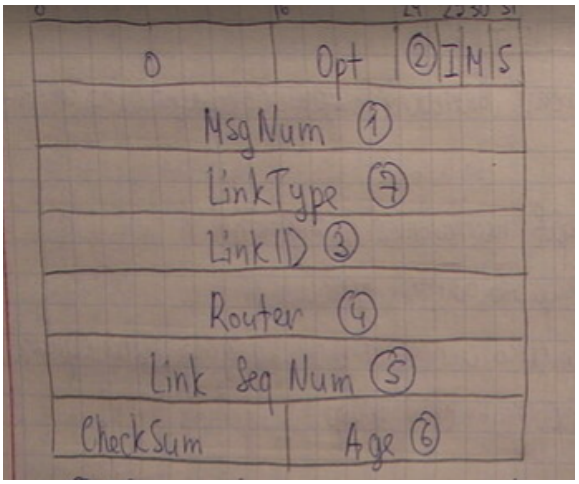
Open Shortest Path First – link-state протокол маршрутизации. Каждый маршрутизатор хранит всю топологию сети, кратчайшие маршруты находит дейкстрой.

Этапы работы и пакеты
Заголовки OSPF вложены в IP-пакеты.

1. HELLO: установка и поддержка соединения между маршрутизаторами. Не широковещательно, а от соседа к соседу.

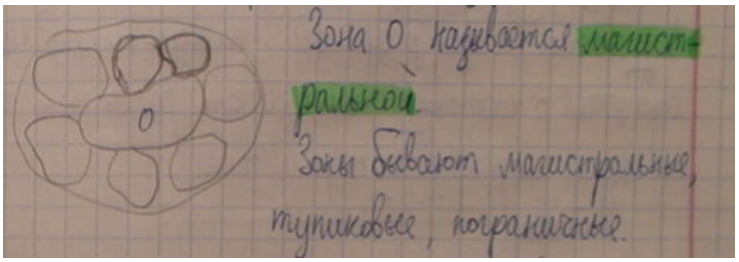


   Рассылаются свежие таблицы маршрутизации (сообщаем только своих соседей, никаких обновлений на основании сообщений других челов). В итоге у каждого маршрутизатора будет одинаковая информация о графе. Такие сообщения также посылаются при изменении состояния сети. Просто так ничего не кидаем, в отличие от RIP.
2. Постройка маршрутов дейкстрой. Бывает, что такое делает только один маршрутизатор, а потом всем рассылает их маршруты



3. Запрос на обновление состояния. Кидаем, когда информация о соседях считается устаревшей.
4. Ответ на запрос на обновление состояния. Еще кидается периодически (но гораздо реже по сравнению с RIP). В отличие от HELLO, может содержать более подробную информацию обо всех метриках (стоимость, надежность, скорость, тип связи).

Зонирование
Можно строить граф не всей сети, а только своей зоны + путь до *магистральной* зоны.



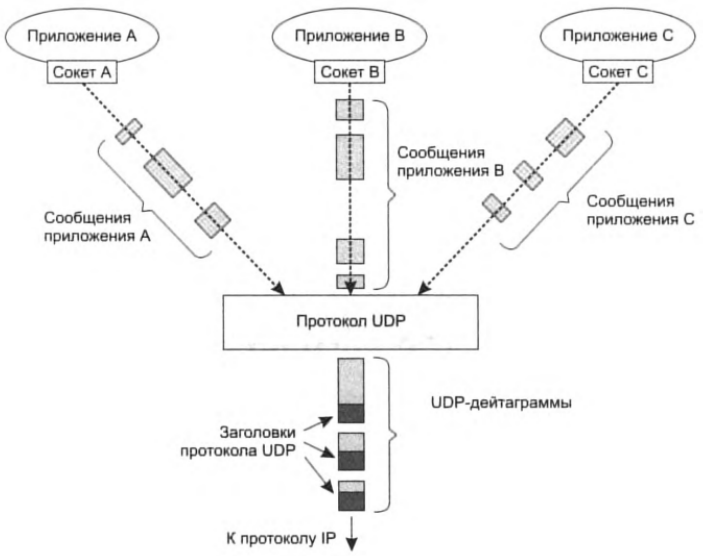
Получается, что маршрутизаторы могут быть внутренние (внутри зоны) и пограничные (соединяют с магистральной зоной). Построение маршрута делится на несколько этапов:

1. От отправителя до пограничного чела своей зоны
2. От своей зоны до зоны получателя
3. От пограничника зоны получателя до самого получателя

В каждой зоне можно сделать один маршрутизатор, который простраивает дейкстры, а остальные пусть только их получают.

22. Протокол UDP. Передача данных без установления соединения

Дейтаграммный протокол (дейтаграммы встраиваются в IP-пакеты), не гарантирует надежность доставки пакетов. Выполняет мультиплексирование данных: если на один сетевой порт поступают сообщения от нескольких приложений (сокетов = IP + порт), он идентифицирует их.



Заголовок UDP содержит UDP-порт отправителя и получателя, длину датаграмы и чексумму.

Алгоритм работы:

1. Модуль IP просматривает поле Protocol ID, если оно соответствует UDP, передает соответствующему модулю (убрав IP-заголовок).
2. UDP проверяет чек-сумму. Она считается не по всей датаграмме, а по псевдо-заголовку, включающему в себя IP отправителя/получателя (это нам дополнительно дает IP-протокол)
3. UDP проверяет порт. Если существует приложение на данном порте, помещаем в очередь сообщений для этого приложения. Иначе просто отбрасываем.

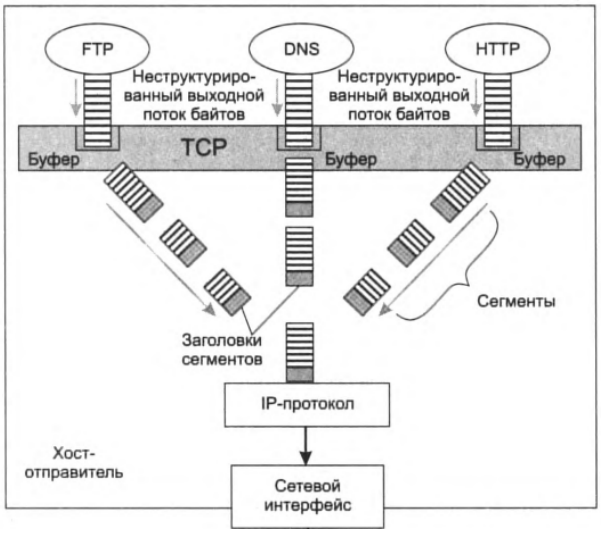
UDP не умеет фрагментировать свои датаграммы, поэтому накладывает ограничения на их размер.

Немного в код по стандарту POSIX

- `socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)`
- `connect(socket, dest, sizeof(dest))` – говорим, кому хотим посылать сообщения (ip+port)
- `bind(socket, &port_info, sizeof(port_info))` – говорим, что будем слушать сообщения на каком-то порту
- `sendto(socket, message, message_len, 0, dest, sizeof(dest))`
- `recvfrom(socket, buf, buf_len, 0, sender, &sender_len)`
- `closesocket(socket)`

23. Протокол TCP. Потоковая передача данных

Потоковый протокол – разбивает сообщение на сегменты и отправляет их один за другим.



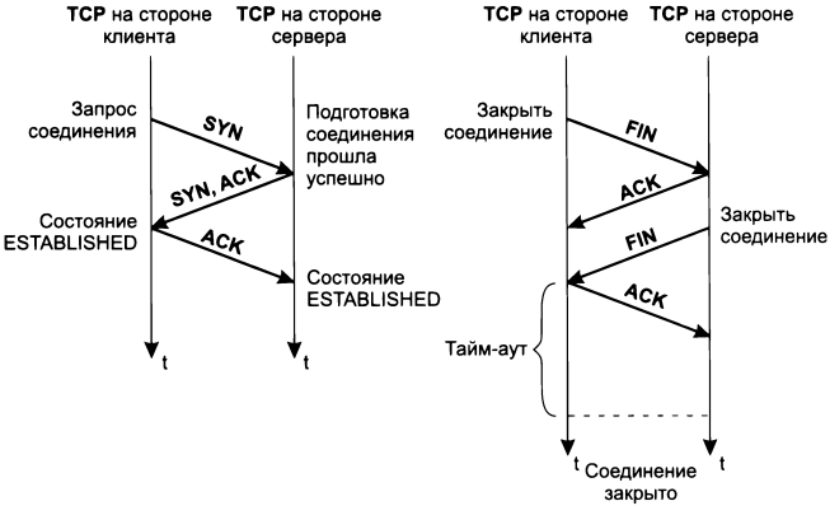
В отличие от датаграммного UDP, разбивает данные на кусочки не по смыслу, а по размеру (берет из буфера следующий непрерывный кусок данных).

Другие фичи TCP:

- Установление соединения
- Гарантия доставки и целостности сообщений
- Скользящее окно + медленный старт

Заголовок

Этапы жизни соединения



1. Установка соединения (состояния сокетов: LISTEN, SYN_SENT, SYN_RECV, ESTABLISHED)
2. Передача данных: зависит от заполнения буфера – если заполнился, отправляем, иначе ждем немножко, и отправляет вне зависимости от заполнения
3. Разрыв соединения (состояния сокетов: FIN_WAIT1, FIN_WAIT2, CLOSING, TIME_WAIT, CLOSE_WAINT, LAST_ACK, CLOSED – могут использоваться не все, есть разные схемы закрытия)

Контроль перегрузки TCP-сетей

Проблема малого пакета: мало данных относительно служебной инфы. Для этого и используется поточность и буфер. + можно использовать метод простоя иногда.
Проблема подавления источника: связан с плавающим окном. В буфере клиента осталось мало места => просим дослать нам мало данных => освобождаем буфер => но сервер уже шлет кучу пакетов где мало данных => буфер забивается малыми пакетами (см. проблему 1) и у нас снова остается мало места… Как фиксить: ждем между этим всем по таймеру; шлем пакеты со ставкой на то, что пока они дойдут, буфер освободится.
Определение размера окна: **алгоритм медленного старта**. Отправитель поддерживает у себя два размера окна: предложенный получателем и *окно перегрузки.* Ориентируясь на минимальное из окон, размер окна перегрузки меняется в зависимости от того, насколько быстро приходят квитанции. “Насколько быстро” определяет *round trip timer*, тик которого зависит от среднего RTT всех предыдущих пакетов и RTT последнего пакета, сложенных с какими-то коэффициентами.
Есть еще *таймер повторных передач* (через сколько считаем, что пакет утерян). Его величина тоже зависит от среднего RTT, умноженного на какой-то коэффициент (обычно от 1.5 до 2). В сетях с большим разбросом времени возвращения к подсчету этого таймера подключают дисперсию.

Немного в код по стандарту POSIX

- `socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)`
- `connect(socket, dest, sizeof(dest))` – говорим, кому хотим посылать сообщения (ip+port)
- `bind(socket, &port_info, sizeof(port_info))` – говорим, что будем слушать сообщения на каком-то порту
- `closesocket(socket)`

Здесь начинаются различия с UDP:

- `accept(socket, &conn_info, &info_len)` – чел, который забиндился, должен принять connect другого чела
- `send(socket, message, message_len, 0)` – не нужно явно указывать, кому посылать (потому что мы сконнектились)
- `recv(conn_socket, buf, buf_len, 0)`

24. Глобальные сети

Сеть, использующая средства дальнего действия для предоставления сетевых сервисов большому количеству абонентов, распределенных по большой территории. Бывают публичные и частные (корпоративные); *магистральные* (высокая пропускная способность и коэффициент готовности) и *сети доступа* (обеспечивают большому количеству абонентов доступ к магистральным сетям). Основная задача глобальных сетей (WAN) в целом – транзитные транспортные механизмы; остальное ложится на LAN (безопасность, организация доступа и т д).
Организации. **Владелец** сети – организует транспорт данных между точками доступа. **Оператор** сети – обеспечивает работоспособность сегмента. **Поставщик услуг** – обрабатывает данные, связующее звено между абонентами и WAN.

Организация WAN берет свое начало из организации глобальных телефонных сетей. Что интересно, от коммутации каналов в телефонах перешли к коммутации пакетов в WAN.
Технологии:

- PDH, плезиохронная цифровая иерархия – для магистральных сетей, 140 Мбит/с, первоначально телефонная технология
- SDH, синхронная – 10 Гбит/с
- DWDM, спектральное мультиплексирование – очень много Гбит/с

Инфраструктура
Сеть строится на основе некоммутируемых (выделенных) каналов связи, которые соединяют коммутаторы глобальной сети. Коммутаторы устанавливаются там, где нужно ответвление или слияние потоков данных конечных абонентов или магистральных каналов. Абоненты сети подключаются к коммутаторам помощью выделенных каналов связи (более слабеньких); могут быть и коммутируемые каналы, например телефонные сети, но это прям плохо.
В глобальных сетях используются те же основные типы кабелей, что и в локальных: на витых
медных парах, медный коаксиальный и волоконно-оптические (одномодовые).
Мосты и маршрутизаторы работают так же как в LAN. *Удаленные мосты* строят таблицу MAC-адресов на основании проходящего через них трафика и принимают решение о передаче кадра в удаленную сеть. В отличие от локальных мостов, используются и сегодня, потому что их не надо особо конфигурировать и поддерживать. *Маршрутизаторы* принимают решение на основании номера сети пакета и упаковывают его в кадр нужной сети, снабжают соответствующим аппаратным адресом следующего маршрутизатора и отправляют в глобальную сеть.

Типы взаимодействия
Пользователь-сеть. Строго стандартизирован, чтобы можно было пользоваться оборудованием любого производителя.
Сеть-сеть. Используется при взаимодействии двух территориальных сетей различных операторов; не нужно так строго стандартизировать.