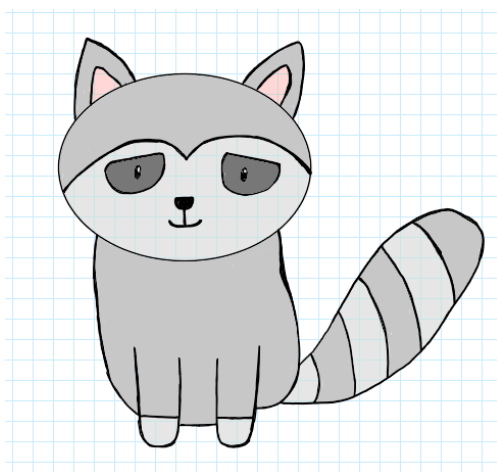


Лекция 1 (05.09)	3
Лекция 2 (12.09)	6
Лекция 3 (13.09)	11
Лекция 4 (11.10)	15
Лекция 5 (17.10)	18
31.10 Структура заголовка IPv4 / IPv6	23
Лекция 6 (24.10)	25
Лекция 7 (25.10)	29
Лекция 8 (31.10)	34
Лекция 9 (07.11)	36
Лекция 10 (08.11)	41
Лекция 11 (14.11)	44
Лекция 12 (28.11)	48
Лекция 13 (05.12)	51
Лекция 14 (06.12)	53
Лекция 15 (19.12)	56
Лекция 16 (20.12)	59



*Это енот, если кто не понял*

все, что выделено желтым или красным маркером - нуждается в дополнении или исправлении

Как всегда отдельное спасибо за рисунки главному редактору Софье)

## Лекция 1 (05.09)

### Методы передачи информации:

1. Симплексный метод - передача информации в одну сторону (радио, телевидение, датчик, интернет вещей).
2. Полудуплекс - передача в обе стороны, но попеременно (рация).
3. Дуплекс или полный дуплекс - передача в обе стороны одновременно (современные сети).

Метод передачи информации сильно зависит от устройства. Передача данных происходит с помощью *протокола* – набора соглашений логического уровня.

### Классификация протоколов:

- По наличию **соединения**

**Commented [1]:** Соединение – это гарантия доставки пакета.

1. Без установления соединения (покой-передача-покой);  
Отправка работает без подтверждения прослушивания => нет гарантии, что пакет дошел до получателя
2. С установлением соединения (покой-соединение-передача-разрыв-покой).  
Работает так, что передача данных начинается после получения положительного фидбека (то есть ответа, что информация получена/прочитана)

- По формату передаваемых данных

1. Битоориентированные (местоположение каждого бита строго определяет функционал).
2. Байтоориентированные (существуют управляющие комбинации, которые регламентируют передачу, в частности: начала и окончания передачи).  
Проблема именно этого типа - управляющие комбинации в теле сообщения. Решается *непрозрачной передачей данных* – когда гарантируется, что каким-то образом данные не совпадут с управляющей комбинацией, например:
  - a. байтстаффинг - есть определенная комбинация окончания, источник смотрит, что в данных такая комбинация есть и ставит перед ними условный лишний байт, о котором знает приемник и при считывании его вместе с комбинацией понимает, что это информация;
  - b. битстаффинг - источник видит, что данные совпали с комбинацией окончания и на определенное место ставит определенный символ, и получатель, зная о подобных вставках, делает обратное преобразование, получая "чистые" данные;

Пример: Пусть управляющая комбинация "01111110" (0, 6 единиц, 0). Если источник передает данные и обнаруживает, что было передано пять "1" подряд, то он автоматически вставляет дополнительный 0 после них (даже если после этих пяти 1 шёл 0). Поэтому последовательность 01111110 никогда не появится в поле данных кадра. Аналогичная схема работает в приемнике и выполняет

обратную функцию. Когда после пяти 1 обнаруживается 0, он автоматически удаляется из поля данных кадра...

- с. уникальная управляющая комбинация - работает с помощью *метода физического кодирования* – посредством регулирования выходного напряжения (см. лекция 3).

#### Типы трафика:

1. Синхронный - трафик, который критичен к задержкам (например, стримы) для него используется протокол без установления соединения. Если при передаче аналогового сигнала без установления соединения были потери, то с помощью аппроксимации можно их восполнить (см. теорему Котельникова, если интересно как именно);
2. Асинхронный - трафик, который не критичен относительно времени и задержек).

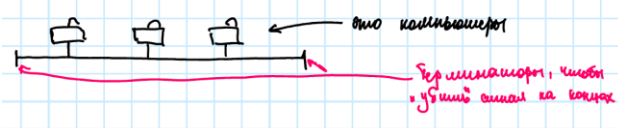
#### Виды сетей в общем смысле:

1. Локальные LAN – local area network
2. Глобальные WAN – wide area network

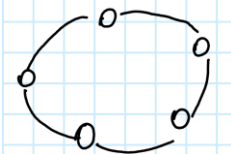
**Топология** – способ соединения (аналогия с графом, где вершины - источники сигналов, а ребра – среда передачи данных).

#### Варианты топологии:

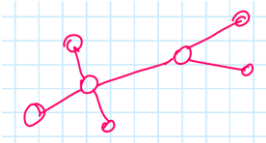
1. Шина - неустойчивая к разрывам, нужен терминатор (пример: устройства подключенные к коаксиальному кабелю, т.е. с одной жилой)



2. Кольцо (принцип: хоть где-то рухнет – рухнет все)

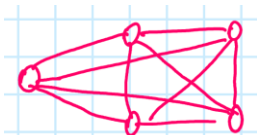


3. Звезда (принцип: рухнет только ребро – рухнет всё на нем, рухнет источник – рухнет всё)

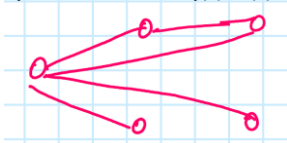


4. Полносвязный (абсолютно все источники соединены между собой → эта топология самая надежная, но и очень дорогая, т.к. требуется много сетевых адаптеров и кабелей → нужен фальшпол, фальшпотолок, коробка). Также существуют следующие проблемы настройки:

- *широковещательный шторм* – заикливание и накопление служебного трафика в сети, в результате чего объем потенциального пользовательского трафика снижается;
- *нестабильность таблицы mac-адресов* - в такой топологии много входных интерфейсов, через которые может подключаться один и тот же клиент, а за каждым клиентом надо закреплять единственный подключаемый интерфейс (а они каждый раз меняются, т.к. приходят через разные пути);
- *множественные копии кадров* - другими словами, дублирование данных, вытекает из той же причины, что и предыдущий – будет на 4-й лабе



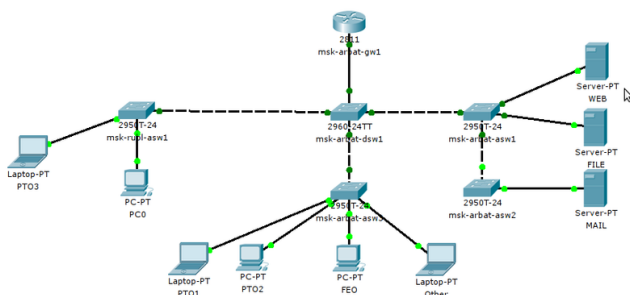
5. Частично связный - топология, при которой один узел соединен со всеми, а остальные произвольно (пример с филиалами и заводами). Эта топология – золотая середина, но необходимо знать критически важную точку (т.е. где будет точка множественного соединения).



Топологии также делятся на:

1. Физическая - как устройства подключены физически
2. Логическая - как логически происходит передача

Например wi-fi *физически* - звезда (есть маршрутизатор и к нему все подключаются):



а логически – шина: потому что они все работают на одной частоте (ну либо на 2-х) и если будет обрыв хоть в одном месте, то сеть “упадет”



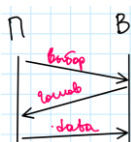
## Лекция 2 (12.09)

**Коллизия** – наложения 2 и более сигналов друг на друга.

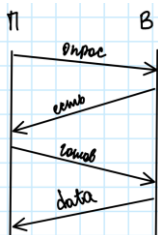
**Дисциплина передачи данных** (методы борьбы с коллизией):

1. Иерархическая – есть первичный узел и несколько вторичных, подчиняющихся первичному. 3 сценария развития событий:

- a. Первичный хочет передать вторичному (выбор - готов - данные);



- b. Вторичный хочет передать первичному (опрос - есть - готов - данные);



- с. Вторичный хочет передать вторичному (вторичный отдает первичному, тот смотрит получателя, видит, что не он получатель, и отправляет на указанный вторичный).

2. Одноранговая – все узлы равны между собой.

- а. Множественный доступ с контролем несущей и обнаружением коллизии (**CSMA/CD** – carrier sense multiple access/collision detection)

Если работаем с шиной, то следующий принцип работы: (шина с устройствами, высокочастотная модуляция и сигнал описанный ниже амплитудная модуляция)



, на генераторе сигнал высокой частоты с одинаковыми фазой, амплитудой и частотой).

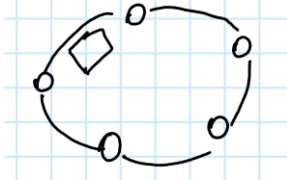
«С контролем несущей» = любой передающий узел просмотрит несущий (свободен – начнет передачу, занят - подождет).

Когда происходит коллизия, посылается Jam-сигнал и окончательно портит основной сигнал, чтобы все узлы точно поняли, что имеет место коллизия. При его появлении те, кто передавал, «замолкают», а затем каждый узел поочередно продолжает передачу, по окончании случайного таймера, чтобы не допустить очередную коллизия.

Коллизия возникает, если узлы передают одновременно или если данные еще передавались после завершения передачи данных источником, а на них было оказано воздействие другим заработавшим источником.

- б. Детерминированный (с избеганием коллизии) работает на топологии типа «кольцо» следующим образом: каждому узлу задается квант времени, в течение которого разрешается передача, реализуется следующим образом: по кольцу «бегает» блок данных, если станция его захватывает, то она может поместить данные в этот блок и передать их

далее по кольцу. Нет необходимости избегать коллизии, т.к. передача происходит строго поочередно.



## Эталонная модель взаимодействия открытых систем OSI / ISO - (The Open Systems Interconnection model)

Состоит из 7 уровней-моделей:

### 7. Приложений – основные задачи:

- связывает приложение со всей моделью;
- предоставление услуг UI – user interface.

Пример: адресант в электронной почте.

### 6. Представлений – основная задача:

- корректное отображение данных на получателе (вне зависимости от ОС, кодировки и прочего).

### 5. Сеансовый – основная задача:

- установление, поддержание и завершение сеансов – аутентификация.

### 4. Транспортный – основные задачи:

- сегментация данных на источнике и реорганизация в поток на приемнике;
- установка, поддержание и корректное завершение соединения (гарантирование надежной передачи). Здесь работает протокол **TCP** и **UDP** (первый работает с установлением соединения).

### 3. Сетевой – основная задача:

- поиск и выбор оптимального маршрута между сетями, географически удаленными друг от друга.

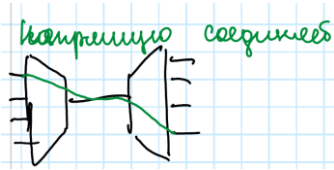
### 2. Канальный – основные задачи:

- определение формата данных для передачи (на этом уровне есть понятие *адреса источника* и *адреса получателя*),
- отвечает за метод контроля доступа в физической среде (т.е. отвечает за дисциплину передачи информации) и за надежную доставку данных.

Устройство, работающее на этом уровне – коммутатор (switch), раньше был мост (коммутатор работает на аппаратном уровне, а мост на программном, вследствие этого коммутатор более быстродействующий).

### 2 варианта коммутации:

1. *Коммутация каналов* - есть источник и получатель, которые соединяются напрямую (канал переключается по очереди на каждого "пользователя").

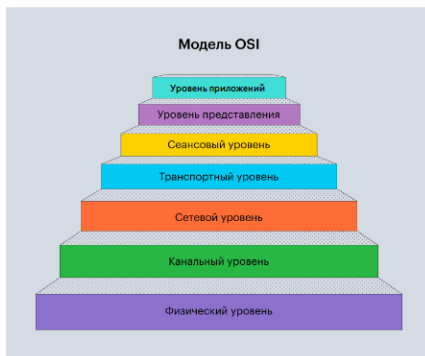


2. Коммутация пакетов, из плюсов: при незагруженности каналов можно увеличить количество передаваемых пакетов, из минусов: наличие пиковой нагрузки, которую нужно рассчитать (чтобы оборудование справлялось, и буферы входа и выхода не переполнялись).

1. *Физический* – электрические и механические, процедурные и функциональные характеристики соединения между устройствами (уровни напряжения и сопротивления, скорость передачи данных и т.п.). На этом уровне нет сведений о источнике и приемнике (глобальных). На нем работают сетевой адаптер, усилитель, концентратор (он же hub – один вход и раздача многим устройствам), медиаконверторы.

4-7 работают на самих устройствах (источнике и приемнике), обрабатывают данные.  
1-3 физически доставляют данные.

**Commented [2]:** 1-3 физически доставить, 4-7 логически обработать



Пример с метро (маршрут: Беговая - Нагатинская):

Сетевой – где спустились и где поднялись (знает крайние точки). Знает, что при поездке до Нагатинской нужно с Баррикадной пересесть на Краснопресненскую -> знает пересадки. Канальный – позволительно знать только ближайшие точки на текущей ветке метро.

Коммутатор соединяет в себе все станции одной ветки. На маршрутизаторе меньше.





## Модель OSI

Данные	Прикладной доступ к сетевым службам
Данные	Представления представление и кодирование данных
Данные	Сеансовый Управление сеансом связи
Сегменты	Транспортный безопасное и надежное соединение точка-точка
Пакеты	Сетевой Определение пути и IP (логическая адресация)
Кадры	Канальный MAC и LLC (Физическая адресация)
Биты	Физический кабель, сигнал, бинарная передача данных

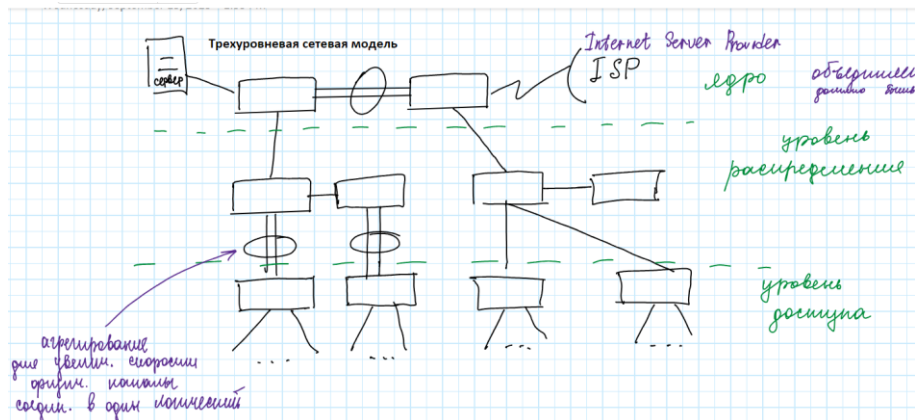
**Модель DOD** (он же стек **TCP/IP**) и соответствие ее уровней уровням OSI/ISO:

4. Прикладной – приложений, представлений, сеансовый
3. Транспортный - транспортный
2. Сетевой – сетевой
1. Канальный – физический и канальный

**Факт:** на данной модели работает интернет

## Лекция 3 (13.09)

**3-уровневая иерархическая модель** (пунктир – мысленное разделение на уровни)



\*Internet Service Provider

Уровни:

1. Доступ – предназначен для соединения всех конечных устройств, для разбиения домена коллизии (уменьшение количества коллизии). Работает только на уровне L2. Домен коллизий – совокупность устройств, борющихся за доступ к пропускному каналу.
2. Распределение – отвечает за маршрутизацию, качество обслуживания, безопасность (кто куда “лезет” и запрещать прохождение), агрегирование (на рисунке – «озеро с рельсами», оно же **etherchannel** – технология, осуществляющая объединение нескольких физических каналов связи в один логический). Отвечает также за переход от одной технологии к другой. Здесь работают L2, L3 устройства.
3. Ядро – самый сложный в реализации, на нем объединяется весь трафик. Здесь проходит весь трафик → оно должно быть скоростным. Помимо этого оно должно быть отказоустойчивым (дублирование компонент и т.д.) (и помехоустойчивым). Обеспечивает доступ в интернет или любые удаленные сервера. Здесь работают только L3 устройства.

Уровни доступа: L2 – канальный, L3 – сетевой

Канальный уровень. Детерминированный метод. Работа на кольце.

Несколько протоколов, работающих по данному методу:

**Протокол Token Ring – «маркерное кольцо».**

(Редко встречается, но многие другие созданы на основе него. Придуман IBM и поддерживается им)

**Два метода работы Token Ring:**

1. **«Передача маркера»** – по кольцу бежит только 1 pdu-шка пустая либо с данными (скорость в 4 мбит/с)
2. **«Раннее освобождение маркера»** когда кто-то захотел передать данные и первая pdu занята, то создается вторая, уже пустая и свободная, которая тоже бежит по кольцу, ее могут захватить и использовать, после чего продолжается прежний алгоритм создания новых pdu-шек (скорость от 64 мбит/с на ранних версиях до 1 гбит/с и более впоследствии).

Смешанного метода работы Token Ring NET.

Из всех станций выделяется одна с наиболее старшим mac-адресом - **активный монитор**.

**Mac-адрес** – уникальный адрес, созданный производителем, занимающий 48 бит: в первых 24 – ID организации, в остальных – ID устройства. Пишется в 16-ричном формате. Пример записи: 2A:5C:27:51:2C:A7, 2A5C:2751:2CA7. Можно переписать адрес, изменив второй бит с 0 → 1.

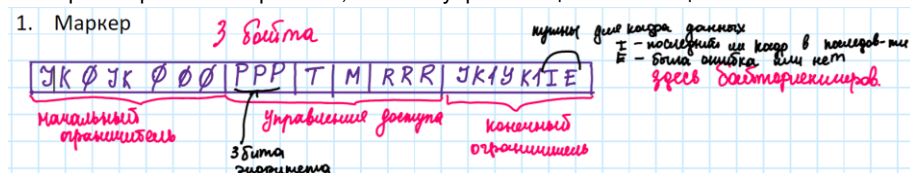
### Разновидность рdi-шек на кольце:

1. Маркер
2. Кадр данных
3. Прерывающая последовательность

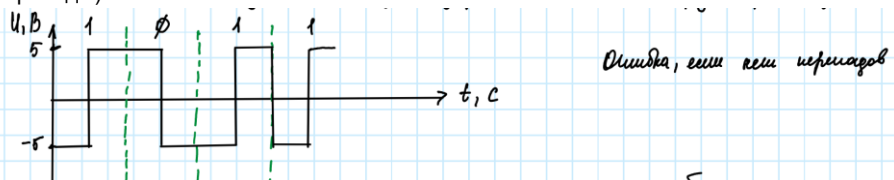
### Про маркеры:

**JK0JK000 | PPP | T | M | RRR | JK1JK1E** – маркер (весит 3 байта, состоит из следующих частей: 1 байт (красный) – начальный ограничитель, 2-й (оранжевый) – управление доступом, 3-й (зеленый) – конечный ограничитель) – это байториентированный протокол, т.к. есть управляющие комбинации.

1. Маркер

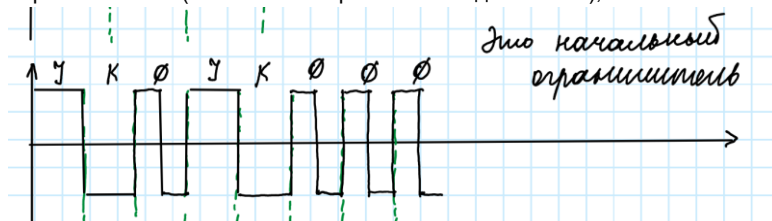


**Манчестерский код** – тип физического кодирования, то каким образом с помощью электрического напряжения передают данные. Перепад напряжения из - в + сообщает, что мы хотим передать 1. А служебный перепад будет происходить при повторной передаче одного и того же символа (у "0" и "1" будут противоположно направленные перепады).



\* ошибка, если нет перепадов

**Начальный ограничитель** в виде манчестерского кода (такты одинаковые) - тут J – верхняя ошибка (нет скачка напряжения на одном такте), K – нижняя ошибка.



Бит **I** – говорит, последний ли кадр в последовательности.

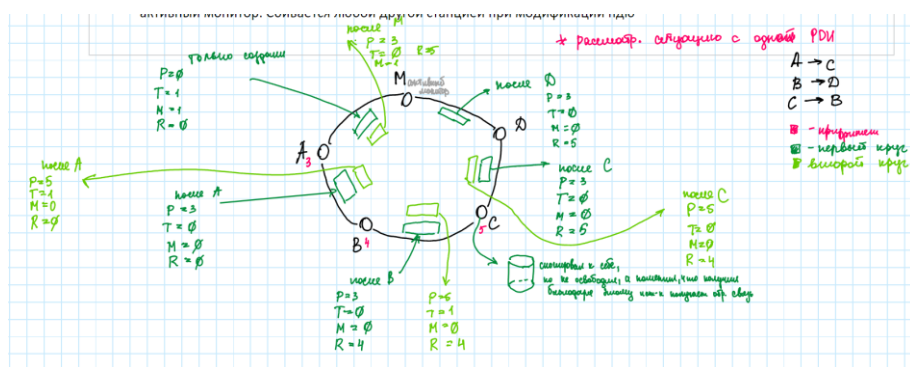
Бит **E** – говорит, была ли ошибка.

Бит **P** – бит приоритета (максимальный приоритет из 3-х таких бит - 7), маркер может захватить только та станция, у которой приоритет данных для передачи равен или больше наибольшего значения в этих битах.

Бит **R** – резервный приоритет, используется для записи в очередь на передачу, записаться может только та станция, приоритет данных которой выше значения, записанного в битах R.

Бит **T** – бит маркера (token) если выставлен, то это маркер (он пустой), иначе кадр данных.

Бит **M** – бит монитора, выставляется активным монитором при создании pdu и при прохождении pdu через активный монитор. Сбивается этот бит любой другой станцией при модификации pdu.



Топология - кольцо, на нем точки с приоритетами A - 3, B - 4, C - 5, D, M. Нужно осуществить передачу данных A → C, B → D, C → B)

Рассмотрим вариант с одной pdu:

-----первый круг-----

1. M создает маркер (P=0, T=1, M=1, R=0)
2. После прохождения A (P=3, T=0, M=0, R=0)
3. После прохождения B (P=3, T=0, M=0, R=4)
4. В C произошло копирование данных, но не освобождение.
5. После прохождения C (P=3, T=0, M=0, R=5)
6. После прохождения D (P=3, T=0, M=0, R=5)
7. После прохождения M (P=3, T=0, M=1, R=5)

-----второй круг-----

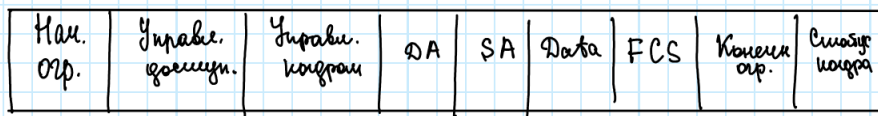
8. A копирует данные и освобождает приоритет
9. После прохождения A (P=5, T=1, M=0, R=0)
10. После прохождения B (P=5, T=1, M=0, R=4)
11. C копирует данные.
12. После прохождения A (P=5, T=0, M=0, R=4)

13. ... (там ещё B-D)

(«чужие данные мы не выкидываем, мы не из 90-х!»)

**Про кадр данных:**

Части кадра данных:



1. начальный ограничитель,
2. управление доступом,
3. управление кадром,
4. DA (destination address),
5. SA (source address),
6. Data,
7. FCS (frame control sequences) - для отлова ошибок,
8. конечный ограничитель,
9. статус кадра (AC\*\*AC\*\*, A = 1- данные дошли до получателя, C = 1 - данные были скопированы).

DA и SA только на этом этапе в таком порядке, так как важно быстро переслать pdu далее

## Лекция 4 (11.10)

### Управление кадрами

Для любого протокола есть 2 вида информации:

- Служебная
- Пользовательская

Управление кадром показывает, какой вид информации идет в сообщении.

1-й вариант служебной информации - существует *служебный монитор*, такой вид служебной информации (кадра) отправляется каждые 3 секунды (есть в Token Ring). Сигнализирует о том, что активный монитор существует.

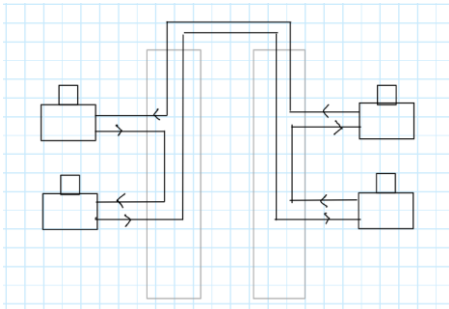
2-й вариант - существует *резервный монитор*, данный тип кадра отправляется станцией, готовой стать новым активным монитором. Резервный монитор ждет, когда заглохнет активный монитор.

"Король умер - да здравствует король"

3-й - *маркер заявки* - с помощью него происходит выбор активного монитора. Чтобы выбрать активный монитор, нужно сравнить мак-адреса. Пускаются маркеры с мак-адресами и каждая станция сравнивает свой адрес с другими. Активным монитор становится устройство с наибольшим мак-адресом.

4-й - *beacon (маяк)* - предназначен для поиска разрыва в кольце. Каждая станция связана напрямую только с двумя соседними, смысл кольца в заиклиивании. Отправляются по таймеру. Та, которая не получила, сообщает, что перед ней разрыв.

### Про прерывающую последовательность



слева и справа - компьютеры, 2 больших прямоугольника - специальные концентраторы, чтобы все работало по топологии кольца. **Концентраторы** бывают:

1. *Пассивные* (закрывают реле, если устройство выходит из строя)
2. *Активные* (умеют замыкать реле и ретранслировать сигнал, усиливая его - в результате появляется возможность увеличить сегмент между концентраторами).

### **FDDI - Fibre Distributed Data Interface (Protocol)**

(тоже каналный уровень, тоже детерминированный метод, тоже на кольце)

Отличие от Token Ring:

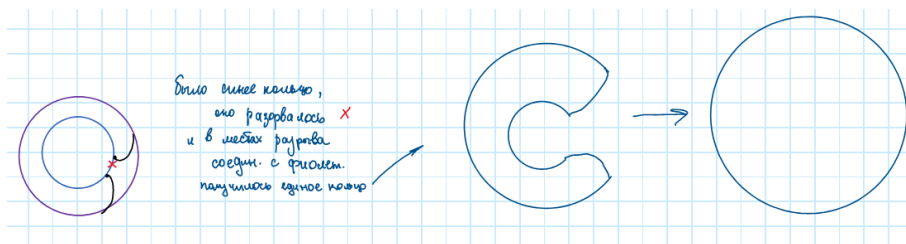
1. Работает на оптоволокне
2. Раннее освобождение маркеров -> ушёл приоритет
3. Синхронный и асинхронный трафик

Синхронный и асинхронный трафик - критичный и некритичный к задержкам, пытается и не пытается - сразу отправится и не ждать в буфере.

В этом протоколе считается, что если трафик синхронный, то смело можно захватить маркер. Если значение времени оборота маркера больше чем для передачи, то маркер захватывается, если меньше, то перехват маркера запрещен, выполняется ретрансляция.

***Ни одна станция не знает топологии системы!***

В синхронном типе трафика время передачи фиксировано - вроде 10 мс. Работает на основе 2-х колец -> отказоустойчивость от одного разрыва (кольца замыкаются, изолируя место разрыва и образуя один замкнутый контур). Переключения делают концентраторы.



### 3 причины создания локальной сети:

- совместное использование данных
- совместное использование ресурсов
- доступ к другим (как правило, глобальным) сетям

**Создание локальной сети** - это создание 4-х групп-компонентов:

1. **Сетевые устройства** - усилитель, коммутатор, маршрутизатор (любое промежуточное устройство).
2. **Соединительные устройства** - среда передачи данных - коаксиальный кабель, витая пара, оптика, wi-fi и прочее.
3. **Конечные устройства** - компы, сервера, периферия.
4. **Протоколы.**

### Иерархия снизу-вверх.

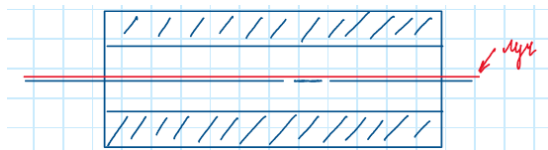
#### Типы соединительных устройств:

- **Коаксиал** (работает по шине, поэтому мало устройств и медленно; длина сегмента не превышала около 25м)
- **Витая пара** (категория 5E 100 мбит/с максимальная длина 90 метров - потом ошибки начинаются)

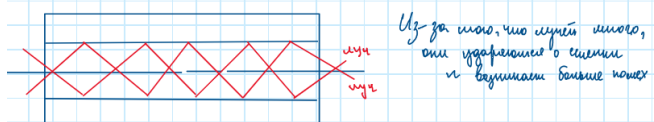


- **Оптический кабель.** Для запуска сигнала используется лазер или диод.
  - **одномодовый** - сигнал проходит вдоль сердцевины, дает большую длину сегмента (4 км).





- **многомодовый** - сигнал идет зигзагом (1 км).



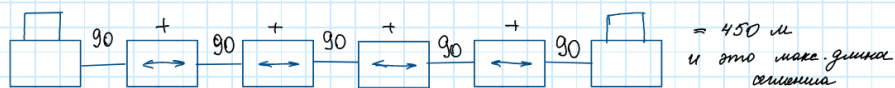
*Из-за того, что лучи много, они удаляются от центра и вызывают больше потерь*

### - Wi-Fi

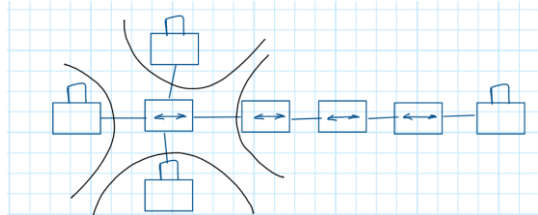
Для наращивания длины сегмента используются сетевые устройства, в частности хабы (концентраторы) (НО НА РК ПРО ЭТО НЕ НАДО ПИСАТЬ). Тут важно соблюдать правило 4-х хабов: между 2-мя конечными устройствами не может быть больше 4-х хабов – макс длина 450 метров.

Каждый хаб на физическом уровне усиливает сигнал, соответственно, добавляет туда ошибки.

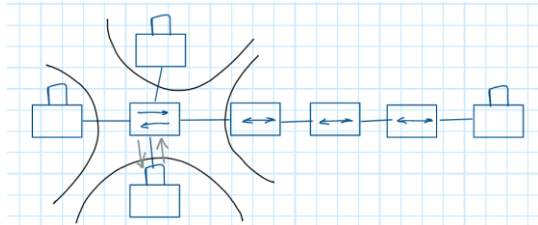
При добавлении хаба получается физически звезда, а логически шина → та же ошибка, что и на обычной шине.



Добавляем устройства → получаем домен коллизии → несколько устройств борются за одну среду передачи данных.

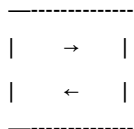


Крайний левый хаб меняем на коммутатор (рисунок с 2-мя стрелками), в результате чего получаем 4 домена коллизии.



## Лекция 5 (17.10)

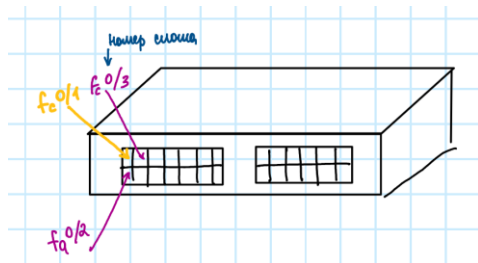
Обозначение свитча:



Свитч работает на канальном уровне, он может прочесть заголовок канального уровня и самостоятельно создать каналы для передачи данных (передачу он тоже сам выполняет).

**Таблица коммутации (адресации) (таблица мас-адресов)** позволяет свитчу соединять устройства, в ней хранятся номера интерфейсов и мак-адреса устройств, подключенных к ним.

Схема свитча:



верхние - нечетные

нижние - четные

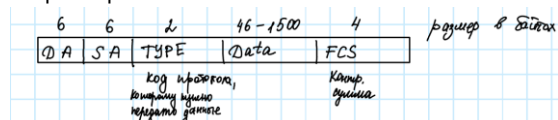
fa - fast ethernet (есть еще просто ethernet и giga ethernet)

Как свитч узнает кому, куда и т.д. →

**Протокол Ethernet 2** - протокол канального уровня

Формат его кадра: DA | SA | Type | Data | FCS

По размерам в байтах: 6 + 6 + 2 + 46-1500 + 4



Первые 3: DA, SA, Type - заголовок

DA - destination address

SA - source address

Type - код протокола, которому нужно передать то, что мы инкапсулировали (кодом зашифрован протокол уровня на 1 выше, например, ipv4)

Хвостовик FCS - контрольная сумма

Записи бывают:

- 1) *Статическими* - человек сел и сам записал мак-адреса (бессрочное время жизни)
- 2) *Динамическими* - запись формируется самим свитчем без участие человека в результате прослушивания каналов, у нее есть некоторое время жизни

Как свитч обрабатывает кадр - **3 метода коммутации**:

- 1) *С буферизацией*  
Весь кадр помещается в буфер и анализируется на наличие коллизий и ошибок, если их нет, то отправляется по DA - надежный, но долгий
- 2) *Без буферизации*  
Свитч берет **первые 6 байт**, смотрит, куда отправить и отправляет
- 3) *Бесфрагментный*  
В буфер попадает первые 64 байта, анализируются на наличие коллизии, если все ок - отправляется получателю (время на распространения 64 байтов достаточно, чтобы все в сети "услышали", что несущая занята - особенность CSMA/CD)

**Commented [4]:** на канальном destination address пишется первый для ускорения, на сетевом уже неважно

(Хаб работает на физическом уровне L1 - он усиливает и передает во все стороны, не делает проверку на коллизию)

Что свитч может сделать с кадром - **функции свитча**:

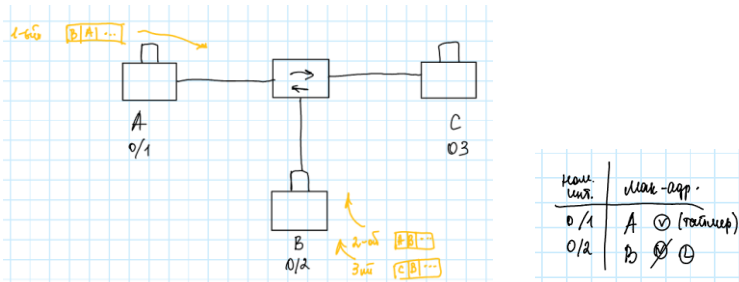
- 1) *Пересылка* - отправляет четко на интерфейс получателя, если известно на каком интерфейсе он подключен
- 2) *Лавинная рассылка - Flooding* - если неизвестно на каком интерфейсе получатель - свитч отправляет кадр на все интерфейсы кроме того, с которого он получил
- 3) *Фильтрация* - если кадр предназначается тому же сегменту (кружок на рисунке на доске - где блок с  $\leftrightarrow$ ) из которого пришел, то он уничтожается

Классификация по объему таблицы коммутаций:

- 1) *SOHO - Small Office Home Office* - 1k-8k коммутаций
- 2) *SMB - Small Medium Business* - 48k-64k коммутаций
- 3) *ISP - Internet Service Provider* - 128k

Если место в таблице коммутаций закончилось, а нужно записать новую коммутацию, то удаляется самая старая

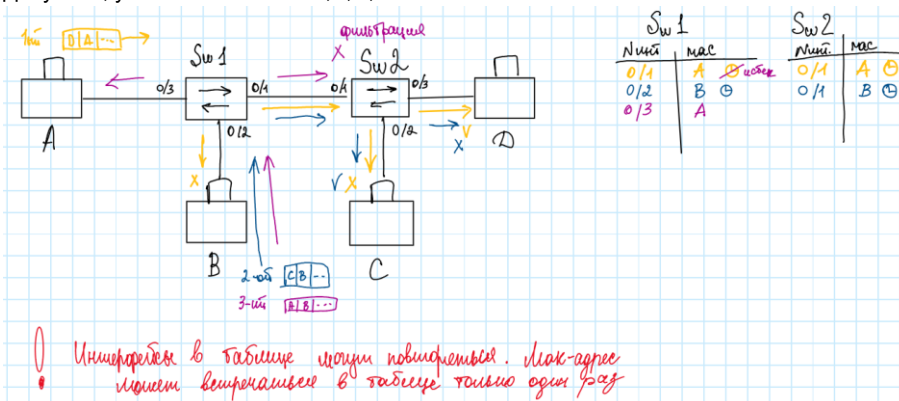
Допустим у нас есть 3 компа: А, В, С с 1-м свитчем



- идет передача из A в B - записывает, что A на интерфейсе 0/1, свитч делает лавинную рассылку
- передача из B в A - запоминаем, что B на интерфейсе 0/2, свитч делает пересылку в A
- передача из B в C - кадр попадает на свитч, он делает лавинную рассылку

крестик значит, что он не принял это, потому что это не ему

Допустим, у нас есть 4 компа A,B,C,D с 2-мя свитчами



- идет передача из A в D - свитч 1 запоминает, что A на интерфейсе 0/3, лавинная рассылка, попадает на свитч 2, он запоминает, что A на интерфейсе 0/1, лавинная рассылка
- идет передача из B в C - свитч 1 запоминает, что B на интерфейсе 0/2, лавинная рассылка, попадает на свитч 2, он запоминает, что B на интерфейсе 0/1, лавинная рассылка

Пример фильтрации - истек A, идет передача из B в A, свитч 1 делает лавинную рассылку и находит A, на 2-м свитче происходит фильтрация, т.к. с интерфейса 0/1 пришел сигнал о передаче в A, а у него в таблице A сохранено на интерфейсе 0/1 - т.е. передача должна произойти в тот же сегмент, из которого он пришел.

**Интерфейсы могут встречаться в таблице коммутаций (таблице мас-адресов) n-ое количество раз, а мак-адреса только 1 раз**

## Сетевой уровень

Его функционал - поиск и выбор оптимального маршрута между сетями, географически удаленными между собой. Здесь используется IP (internet protocol), который выдается программистом, поэтому применим в отличие от мак-адресов.

### IPv4

#### Сеть | Хост

IP-адрес состоит из **32 бита**, они разделены на 4 октета по 8 бит  
пример адреса - 192.168.1.1

#### Классы IP-адресов:

A - граница сети после 1-го октета

B - граница сети после 2-го октета

C - граница сети после 3-го октета

D - нет понятия хоста, предназначен для групповой рассылки, используются только как получатели, не могут быть источником

E - для дальнейших разработок, ими нельзя пользоваться, используется только 1 адрес (255.255.255.255 - локальный широковещательный адрес)

Как определить класс - по старшему биту первого октета:

0xxx - A (1-127)

10xxx - B (128-191)

110xxx - C (192-223)

1110xxx - D (224-239)

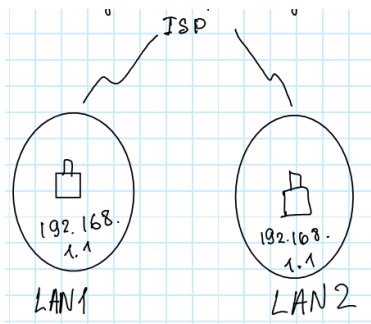
11110xxx - E (240-255)

Зарезервированные адреса:

- 1) *Адрес сети* **192.168.1.0** - адрес без указания хоста (сказали улицу - сеть, но не назвали номер дома - хост (все дома не многоквартирные))
- 2) *Направленный широковещательный адрес* **192.168.1.255** - все хвостовые биты такого адреса забиты единицами, производится рассылку пакетов в рамках одной сети (каждому устройству). При попытке выйти за пределы сети маршрутизатор убивает запрос.
- 3) *Локальный широковещательный адрес* **255.255.255.255** (класс E), все биты забиты единицами, используется при подключения к сети с целью получения IP-адреса
- 4) *Неопределенный адрес* - **0.0.0.0**
- 5) *IP-адрес автоконфигурации* - **169.254.X.X** - выдается, когда нет соединения с сервером (DHCP), и человек его не назначил
- 6) *Localhost* - **127.0.0.0** (127.X.X.X в 6-й версии (протокола)? не так)

Виды IP-адресов:

- 1) *Публичные* - белые - адреса, используемые для "хождения" в сеть
  - 2) *Частные* - серые - адреса, обслуживаемые в пределах локальной сети
- есть 2 устройства с одинаковыми IP → маршрутизатор посредством технологии NAT  
(Network Address Translation) заменяет частный адрес на публичный



#### Разновидности NAT

- 1) *Статический static* - пишем руками пары: IP частный - IP публичный, минус - нужно много публичных адресов
- 2) *Динамический dynamic* - задается пул публичных адресов, которые можем выдать, с помощью политики безопасности говорим, кто может выходить в публичную сеть, а кто нет. Минус - сколько хотим выпустить устройств в сеть, такого размера пул
- 3) *Перегруженный PAT (Port Address Translation)* - подменяет все частные IP-адреса на один публичный, но запоминает номера портов и потом понимает, куда вернуть

192.1.1.1:50136 → 1.1.1.1:50136

192.1.1.1:49723 → 1.1.1.1:49723

В каждом классе выделен пул публичных и частных адресов.

Частные:

A - 10.0.0.0

B - 172.16.0.0 - 172.32.0.0

C - 192.168.0.0 - 192.168.255.0

Все остальные - публичные

## 31.10 Структура заголовка IPv4

Важное требование к IPv4 - кратность 32 битам

4 байта	Номер версии (4 бита)	Длина заголовка (4 бита)	Тип сервиса (8 бит)	Длина пакета (16 бит)
---------	-----------------------	--------------------------	---------------------	-----------------------

4 байта	ID пакета (16 бит)		Флаги (3 бита)	Указатель фрагмента (13 бит)
4 байта	Время жизни	Протокол	Контрольная сумма	
4 байта	Адрес источника			
4 байта	Адрес получателя			
Переменная длина	Опции			_____ заполнитель (чтобы поле было кратно 4-м байтам)

Поле "Опции" может быть, а может и не быть. Имеет переменную длину

**Номер версии** – номер версии протокола (в IPv4 будет 4, в IPv6 будет 6)

Formatted: Indent: First line: 1,27 cm

**Длина заголовка** – сколько реально занимает заголовок, считается в словах (каждое слово - 4 байта) (длина заголовка IPv4, если слово (поле) "Опции" отсутствуют - 5 слов)

Тип сервиса:

P	P	P	D	T	R	ECN
---	---	---	---	---	---	-----

- P (Priority)** – биты приоритета (при отсутствии настройки качества обслуживания происходит работа по принципу FIFO, при настройке есть возможность разделить очередь на очереди с разными приоритетами - высоким, средним и низким, трафик в них можно разделить разными способами, например:
  - Если в высоком что-то присутствует, то средний и низкий молчат, потом, когда старший закончит, то начнет средний и т.д.
  - Проценты раздать - например: высокий 50%, средний 30%, низкий 20%
  - Смешанный - пока высокий не закончит, средний и низкий молчат. Как только высокий договорил, пропускная способность делится между средним и низким.
- D (Delay)** – бит задержки, если выставлен то выбирается маршрут с минимальной задержкой
- T (Throughput)** – если он выставлен то мы выбираем маршрут с наибольшей реальной пропускной способностью канала.
- R (Reliability)** – надежность, если выставлен, то ищется маршрут с наибольшей надежностью (наименьшей возможностью ошибки), надежность определяется на основе статистики утерянных пакетов, которая копится за каждым интерфейсом

**Commented [5]:** Bandwidth – пропускная способность канала передачи, заявленная производителем.  
Throughput – реальная пропускная способность. Стремится к bandwidth, но не превышает её  
Tổng cộng 4 phản ứng  
Michael Ovakimian đã phản ứng bằng 👍 lúc 2023-10-31 06:13 SA  
Ivan Pogiba đã phản ứng bằng 👍 lúc 2023-11-29 13:31 CH  
Serg đã phản ứng bằng 👍 lúc 2024-01-13 00:48 SA  
Валерий Полубояров đã phản ứng bằng 👍 lúc 2023-12-07 06:57 SA

- **ECN (Explicit Congestion Notification)** - явное сообщение о задержке, подается маршрутизатором источнику, чтобы тот остановил передачу, чтобы маршрутизатор успел обработать все и "не убил" трафик. Когда буфер освобождается, маршрутизатор сообщает источнику, что можно передавать.

**Длина пакета** - просто размер пакета в байтах (минимум 20 байт, максимум 65535)

**ID пакета** – связывает все фрагменты пакета

**Флаги** – говорят о том, разрешена ли фрагментация, 3 бита:

Х - без функционала

рз - можно ли фрагментировать (разрешено-запрещено)

пн - последний пакет или нет

**Указатель фрагмента** – определяет, в какое место какой пакет вклеивать, используя смещения относительно начала

**Время жизни (TTL - Time To Live)** – время, в течении которого пакет существует, указывается в "хопах" сетевых устройств, когда становится равен 0, то пакет уничтожается

**Протокол** – код протокола, которому необходимо передать поле данных пакета (т.е. протоколы L4 уровня, например TCP, UDP)

**Контрольная сумма** – анализируется маршрутизатором и в случае несоответствия заменяется (в IPv6 отсутствует)

**Опции** – может отсутствовать, приоритетнее типа сервиса. Его размер должен быть кратен 32 битам. Чтобы заполнить биты до кратных 32, используется **заполнитель**.

**Классы опций:**

0) Datagram'ы пользователя, т.е., например,. маршрут по которому надо отправить пакет

1) Зарезервированы

2) Отладка. Тут можно записать временные метки (timestamp-ы) прохождения через маршрутизаторы. Позволяет отследить маршрут пакета.

3) Зарезервированы

**Commented [6]:** количество промежуточных устройств уровня L3

## Лекция 6 (24.10)

### Бесклассовая адресация

Граница сеть/хост может не принадлежать классу

Аналогия:

сеть - название улицы

подсеть - номер дома

хост - номер квартиры

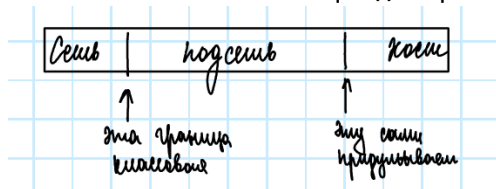
2 класса адресации (это не протоколы, это алгоритмы)



- 1) **VLSM** (Variable Length Subnet Masks) - *маски переменной длины* (на это у нас ДЗ)  
IP состоит из 3-х частей - **сеть|подсеть|хост**  
сеть | подсеть - граница классовая
- 2) **CIDR** (Classless Inter-Domain Routing) - бесклассовая адресация

## VLSM

IP состоит из 3-х частей - **Сеть | Подсеть | Хост**



Пример необходимости использования: нам дали одну сеть и сказали, что другую не используем (но к маршрутизатору подключено 3 свитча)



192.168.1.0/24

// Маска пишется через слеш как количество единиц, т.е. /24 эквивалентно 255.255.255.0

Оборудование ничего не знает про адресацию.

192.168.1. \_ \_ \_ \_ \_

Разобьем на 4 подсети: (синим указаны битовые комбинации)

192.168.1.00|000000 - 192.168.1.0/26

192.168.1.01|000000 - 192.168.1.64/26

192.168.1.10|000000 - 192.168.1.128/26

192.168.1.11|000000 - 192.168.1.192/26

После адреса сети в первых 2-х битах записан адрес подсети

Маска подсети - 26, потому что нужно еще 2 бита на адрес подсети

(широковещательная маска теперь из 26 единиц)

## Задача 1

дан адрес сети - 192.168.1.0/24

нужно разбить на 16 подсетей и написать адрес 5-го хоста во 2-й (0,1,2,...) подсети

Решение:

192.168.1. \_ \_ \_ \_ | \_ \_ \_ \_

16 подсетей =  $2^4$  подсетей → нужно 4 бита под подсеть, 2-я (нумерация с 0) подсеть имеет адрес - 0010, 5-й хост - 0101

192.168.1.0010 0101 / 28

192.168.1.37 / 28 - адрес 5-го хоста во 2-й подсети

("|" разделяет подсеть и хост)

### Задача 2

дан адрес сети - 192.168.1.0/24

тах количество подсетей, min 15 хостов в каждой, написать адрес 5-го хоста во 2-й подсети

Решение:

192.168.1.0/24

15 хостов + 1 широковещательный + 1 адрес для сети = 17 хостов → минимум 5 бит под адресацию → 3 бита для подсети

192.168.1. \_ \_ \_ | \_ \_ \_ \_ \_

192.168.1.010 00101 / 27

192.168.1.69 / 27

### Задача 3

дан адрес сети - 172.16.0.0/16

тах количество подсетей, min 300 хостов в каждой, написать широковещательный адрес во 2-й подсети

Решение:

$300+1+1 = 302 \rightarrow 9$  бит под адресацию

172.16.0.0/16

172.16. \_ \_ \_ \_ \_ | \_ . \_ \_ \_ \_ \_ \_ \_ \_

172.16.0000010 | 1 . 11111111 / 23

172.16.5.255

### Задача 4

дан IP адрес - 10.67.0.0 / 12

тах количество подсетей, min 1023 хоста в каждой, написать адрес 77-го хоста в 1-й подсети

**У двух разных IP адресов в одной сети мб маски разной длины**

Решение:

10.0100 | 0011.0000 0000.0000 0000 / 12 - дано

$1023+2 = 1025 \rightarrow 11$  бит под адресацию хоста

10.0100 | 0000.0000 0 | 000.0000 0000 / 21

77-й хост ( $64+8+4+1$ ) и 1-я подсеть

10.0100 | 0000.00001 | 000.0100 1101 / 21

10.64.8.77

## CIDR

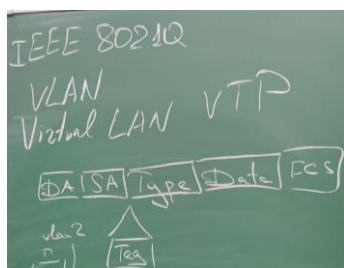
### Сеть | Хост

Маска может смещаться и влево, и вправо.

Пример: 10.0.0.0 / 5 - это адрес хоста  
000001 | 010.0.0.0  
8.0.0.0 / 5 - адрес сети

## VLAN - Virtual LAN

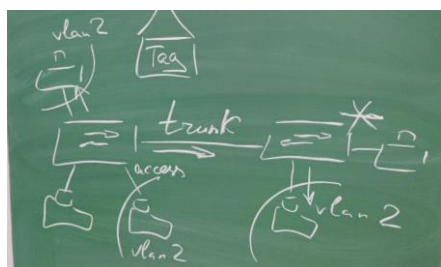
**Стандарт IEEE 802.1Q** - расширяет понятие ethernet и вводит понятие VLAN - Virtual LAN (позволяет создание подсетей на канальном уровне)



1-й VLAN существует всегда, но интерфейс VLAN1 в свитче - это мозг коммутатора, обращаясь к нему мы не пользуемся VLAN1, мы просто смотрим интерфейс.

стандартный адрес Ethernet 2 - DA / SA / Type / Data / FCS

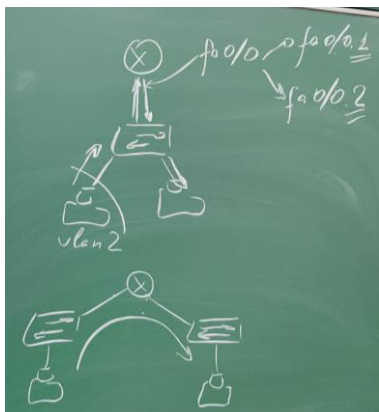
В соответствии с описанным стандартом происходит добавление Tag поэтому стандарту - DA / SA / **Tag** / Type / Data / FCS



в одном из полей Tag записывается VLAN

- 1) **access-интерфейс**: при входе кадра на интерфейс добавляет Tag, при выходе - удаляет  
1-й VLAN на тегируется (он native - нативный), все остальные тегируется
- 2) **trunk-интерфейс** (магистраль): не удаляет Tag кадра при выходе с интерфейса и не добавляет при входе. Позволяет настраивать прохождение определенных VLAN-ов (устанавливается между сетевыми устройствами)

Как маршрутизировать между VLAN-ами? - С помощью маршрутизатора



сверху - физическая топология, снизу - логическая топология

Виртуальные интерфейсы (саб-интерфейсы) - fa0/0.1 и fa0/0.2 - каждому назначается физический адрес и сеть (словно у нас логическая топология)

## Лекция 7 (25.10)

### Протокол VTP - *Vlan Trunking Protocol*

Занимается созданием, удалением и переименованием VLAN-ов.

В коммутаторах Cisco с поддержкой VLAN протокол STP по умолчанию выполняется независимо для каждой виртуальной сети.

в VTP у **свитчей** есть **3 режима работы**:

- 1) *Сервер* - создает, переименовывает и удаляет VLAN (режим по умолчанию)
- 2) *Клиент* - ничего нельзя сделать, но он применяет обновления от сервера и пересылает обновления дальше по своим интерфейсам
- 3) *Прозрачный* - изменения VLAN, внесенные на данном коммутаторе, распространяются только на данный коммутатор. Получаемые обновления от VTP сервера распространяются другим коммутаторам, но не применяются на данном коммутаторе.

Для начала работы нужно перевести свитч в режим 2 или 3, чтобы не возникало конфликта серверов, поскольку это может привести к затиранию таблицы VLAN-ов (по умолчанию все свитчи в режиме сервера).

Для работы протокола все устройства должны принадлежать одному и тому же домену (эта штука критична к регистрам) - как правило одна и та же подсеть.

Например, эту лекцию слушает только 3 курс ИУ6, ИУ6 3 курс получается именем домена.

Если пароль на свитче и на pdu отличаются, то pdu-шка не принимается

## Протокол STP - Spanning Trunking Protocol

Предназначен для убирания петель на канальном уровне. Причины возникновения петель:

- 1) Создана пользователем для организации топологии (например кольцо)
- 2) Когда патч-корд замкнут на свитч с 2-х сторон
- 3) Когда патч-корд замкнут на 2-х розетках

Из-за петель происходит:

- 1) *Широковещательный шторм* - размножение широковещательных сообщений активным сетевым оборудованием (одна pdu на коммутаторе размножается на все сразу, тот может передать следующему, который ее тоже размножит).
- 2) *Нестабильность таблиц коммутаций* (они же таблицы mac-адресов) - постоянная перезапись таблицы коммутаций, т.к. кадр с одного и того же mac-адреса приходит с разных портов
- 3) *Множественные копии кадров*. Один и тот же кадр рассылается в несколько сторон и к получателю приходит в нескольких копиях.

### Как работает STP и как разрешает конфликты

- строится дерево из имеющейся топологии (если она, например, замкнута)
- 1) *Выбирается 1 корневой мост (свитч)* - у каждого свитча есть Bridge ID, состоящий из Priority (=  $\text{vlan xxx} + 4096 * N$ , N-множитель назначается администратором сети ( $4096 * 8 = 32768$  default cost)) и mac-адреса, чем меньше приоритет и чем меньше mac, тем больше вероятность стать корневым мостом. (у корневого коммутатора все порты — назначенные)
  - 2) *Каждый свитч просчитывает кратчайший путь к корневому* - на каждом свитче выбирается корневой порт и предоставляется маршрут минимальной стоимости (пропускной способности) от данного свитча до корневого моста (с началом в корневом порту).  
(у любого некорневого свитча может быть только один корневой порт)  
(если два или более порта предоставляют маршруты минимальной стоимости пути до корневого моста, то корневым станет тот порт, который связывает с мостом, имеющим меньший Bridge ID)  
(возможны случаи, когда стоимость пути по двум и более портам коммутатора будет одинакова, тогда выбор корневого порта будет происходить на основании полученных от соседей приоритета и порядкового номера порта (Lowest Sender Port ID), например fa0/1, fa0/2, fa0/3 и корневым станет порт с наименьшим номером)
  - 3) *Происходит выбор назначенных мостов и портов* - на каждом сегменте сети, к которому присоединен более чем один мост (или несколько портов одного моста), просчитывается кратчайший путь к корневному мосту (порту). Мост, через который проходит этот кратчайший путь, становится **назначенным мостом** для этой сети (Designated Bridge), а соответствующий порт — **назначенным портом** (Designated port).  
(на одном сегменте может быть один назначенный порт)  
(если два или более порта имеют одинаковую стоимость пути от сегмента до корневого моста, то мост с наименьшим идентификатором моста (Bridge ID) выбирает свой порт в качестве назначенного порта)

**Commented [7]:** Если возникают вопросы, то смотрим тут [https://ru.wikipedia.org/wiki/STP#%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF\\_%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%8F](https://ru.wikipedia.org/wiki/STP#%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF_%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%8F)

Если вопросы остались, то смотрим тут и переводим [https://en.m.wikipedia.org/wiki/Spanning\\_Tree\\_Protocol](https://en.m.wikipedia.org/wiki/Spanning_Tree_Protocol)

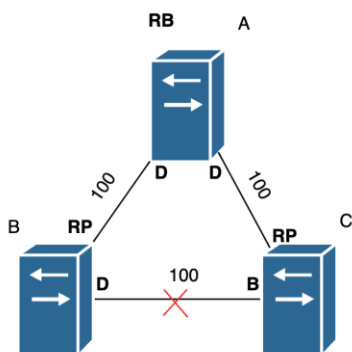
**Commented [8]:** на самом деле такое невозможно из-за особенности рассылки BPDU - Bridge Protocol Data Unit, которые каждые 2 секунды рассылаются RB  
эта рассылка положена в основу работы STP

**Commented [9]:** Если непонятны сноски по разрешению конфликтов, то рекомендую [https://en.m.wikipedia.org/wiki/Spanning\\_Tree\\_Protocol#Tiebreakers](https://en.m.wikipedia.org/wiki/Spanning_Tree_Protocol#Tiebreakers)  
буду рад, если перепишите в более читаемый вид

**Commented [10]:** Сегмент сети - это логически или физически обособленная часть сети, разделенная на участки для оптимизации сетевого трафика и/или повышения безопасности сети в целом. Сегменты сети могут быть ограничены сетевыми устройствами, такими как коммутаторы (2-й уровень в модели OSI) или маршрутизаторы (3-й уровень в модели OSI)

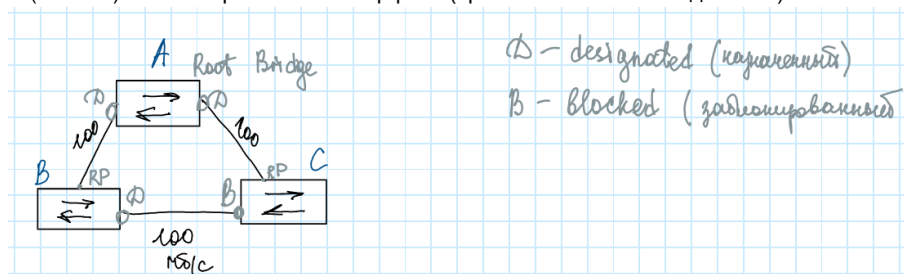
- 4) **Блокировка некорневых и неназначенных портов** - во всех сегментах, с которыми соединено более одного порта моста, мосты блокируют все порты, не являющиеся корневыми и назначенными.

В итоге получается древовидная структура (дерево) с вершиной в виде корневого свитча.



D (designated) - назначенный интерфейс (левый на нижнем соединении)

B (blocked) - заблокированный интерфейс (правый на нижнем соединении)



RB - Root Bridge

RP - Root Port

на каждом свитче только один порт будет корневым (на примере - верхний на B и C и везде одинаковая пропускная способность)

#### **Что делает любой интерфейс (порт)**

Blocking 20 sec -> Listening 15 sec -> Learning 15 sec -----> Forwarding

#### **Что делает заблокированный интерфейс (порт)**

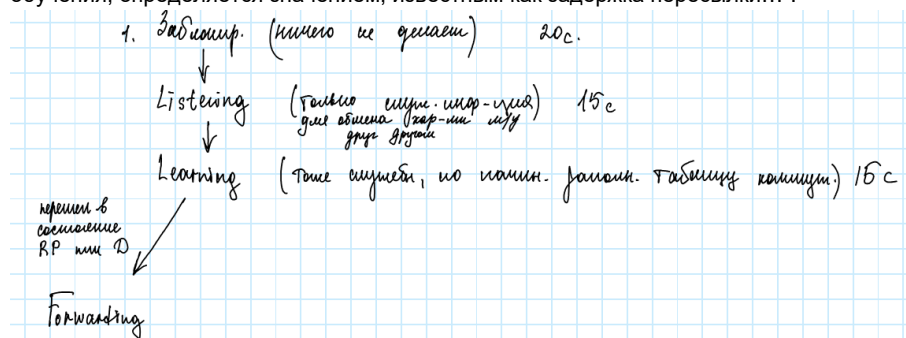
Blocking 20 sec -> Listening 15 sec -> Learning 15 sec -----> Blocking

Заблокирован 20 сек -> Слушает порт 15 сек -> Обучается (получает данные и обновляет таблицу mac-адресов) 15 сек -----> Пересылает пакеты (если перешел в состояние RP или DP)

весь цикл = 50 секунд

Все порты в STP последовательно проходят 4 состояния: *blocking* (прослушивают BPDU без передачи данных), *listening* (прослушивают и ретранслируют BPDU), *learning* (получают данные, обновляют MAC-таблицы), *forwarding* (рабочее состояние порта). С интервалами по умолчанию работа порта (*forwarding*) начинается через 30 сек. А вот почему:

“Когда устройство впервые подключается к порту коммутатора, оно не сразу начинает пересылку данных. Вместо этого оно пройдет ряд состояний, пока обрабатывает BPDU и определяется топологию сети. Порт, подключенный к хосту, такому как компьютер, принтер или сервер, всегда переходит в состояние пересылки (*forwarding*), хотя и после задержки в 30 секунд, пока он проходит состояния прослушивания (*listening*) и обучения (*learning*). Время, проведенное в состояниях прослушивания и обучения, определяется значением, известным как задержка пересылки...”



## Протокол RSTP - Rapid Spanning Tree Protocol

Блокировка и прослушка (*Blocking* и *Listening*) объединены в один этап *Discarding* - весь цикл = 30 секундам вместо 50

введено понятие *port-fast* (*port-edge*) смотрят только в сторону конечных устройств (там никогда не будет петли) и не имеют состояния *listening*

введены роли заблокированных интерфейсов

**Роли для заблокированных интерфейсов** - их “будущее”:

- 1) *Альтернативный* - замена корневому
- 2) *Резервный* - замена назначенному

Минус: если в сети есть VLAN, то он не смотрит на них и строит одно дерево (другими словами, построение дерева на зависит от VLAN-ов), как и STP

## Протокол PVST - Per-VLAN Spanning Tree

Также 30 сек, есть бонус в виде балансировки, если мы строим разные варианты дерева VLAN

Он для каждого VLAN строит свое дерево. Мы можем для разных VLAN-ов настроить разные корневые мосты. Для одного VLAN блокируется одна связь, для другого другая.

## DHCP - Dynamic Host Configuration Protocol

Протокол **прикладного** уровня (заработает только после запуска ОС)

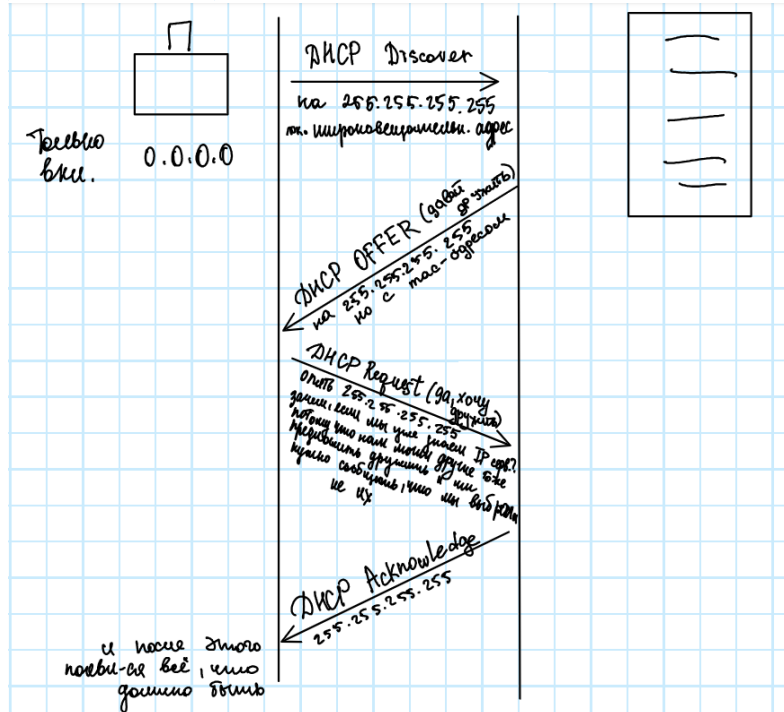
Работает поверх UDP, который на транспортном уровне (работает без установки соединения)

DHCP выдает:

- 1) IP / маска
- 2) dg - default gateway
- 3) Адрес DNS-сервера - того сервера, который будет преобразовывать доменные имена в IP-адреса

Если мы подключаемся к вайфай-точке, то маршрутизатор будет DHCP-сервером. Также DHCP-сервером может быть сервер.

Как это происходит:



**Commented [11]:** можно тут почитать подробно [https://ru.wikipedia.org/wiki/DHCP#%D0%9F%D1%80%D0%B8%D0%BC%D0%B5%D1%80\\_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%B0\\_%D0%BF%D0%BE%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F\\_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0](https://ru.wikipedia.org/wiki/DHCP#%D0%9F%D1%80%D0%B8%D0%BC%D0%B5%D1%80_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%B0_%D0%BF%D0%BE%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0)

Как работает (Устройство - Сервер):

- 1) Устройство появилось в сети 0.0.0.0 (комп включил ОС)
- 2) У-С. Подается DHCP Discover на 255.255.255.255 (светится мас-адрес, чтобы offer прилетел именно тебе)
- 3) С-У. Подается DHCP Offer на 255.255.255.255 с мас-адресом внутри



- 4) У-С. Подается DHCP Request на **255.255.255.255** (потому что кинуть оффер могло несколько серверов, но мы выбрали один(первый, который нам ответил), он получит сигнал для следующей передачи данных (Ack), а остальным мы сообщим, что выбрали другой -(выбрали первый, который был прислан. сообщаем, что выбрали тем, что кидаем этот request всем серверам, а в нем уже указан выбранный нами айпишник.)
- 5) получается, все теперь знают, что мы выбрали айпишник.)
- 6) С-У. Подается DHCP Ack на 255.255.255.255

### 3 режима работы DHCP-сервера

- 1) *Ручное назначение статических IP-адресов* - по паре mac-IP (удобно в случае с МФУ(Многофункциональное устройство), когда ее mac ставим в соответствие IP)
- 2) *Динамическое назначение статических адресов*: мы задаем пул адресов, которые хотим выдать, при проходе внешнего запроса сервер выдает адрес и запоминает, какому mac-адресу какой IP-адрес выдал
- 3) *Автоматическое назначение динамических адресов* (задаем пул IP, задаем время аренды - сколько времени устройство может пользоваться этим IP, пока устройство подключено и время аренды не вышло, то в сети видна пара IP-mac, в следующем подключении пара меняется)

### ARP - Address Resolution Protocol

протокол **сетевого** уровня

смысл - по известному IP-адресу получателя найти его mac-адрес

как выглядит пакет:

Hardware Type (говорит, что ищем mac-адрес)		Protocol Type (говорит, что ищем по IP)
Hardware Length (= 6 = размер mac)	Protocol Length (= 4 = размер IP)	Operation (определяет, запрос это или ответ)
Sender Hardware address (mac-адрес отправителя запроса)		
Sender Protocol address (IP адрес отправителя запроса)		
Target Hardware address (mac-адрес получателя запроса, в запросе будет 48 нулей, в ответе уже сам mac-адрес)		
Target Protocol address (IP адрес получателя запроса)		

Запрос широковещательный, ответ - юникастом

Пары mac-IP хранятся в ARP-таблицах в 2-х видах: статика (...) и динамика (заполняется посредством рассылок и обладает временем жизни)

**Commented [12]:** можно почитать тут подробнее  
<https://ru.wikipedia.org/wiki/ARP>

## Лекция 8 (31.10)

### ICMP – Internet Control Message Protocol

Не предназначен для пользователя

Между сетевым и транспортным уровнем, т.к. инкапсулируется в IP, но чаще всего его считают L3 протоколом.

#### Цели работы протокола:

- 1) *Диагностика сети* - проверка работоспособности (например команда `ping` для проверки прохождения пакета)
- 2) *Отправление сообщений об ошибках* (если пакет данных был уничтожен, то ICMP сообщит об этом источнику)

**Commented [13]:** работает в обе стороны, не надо пинговать по очереди с каждой стороны - в этом нет смысла

Тип (8 бит)	Код (8 бит)	Контрольная сумма (16 бит)
Данные		

**Тип** - причина, определяет что происходит (например, адресат недоступен)

**Код** - конкретика причины, подр

...

Save to Files

...

Download file

обности о ней

**Контрольная сумма** - ...

Примеры:

для `ping` - тип: echo, код: туда или обратно;

для *ошибки* - тип: есть ошибка, код: почему ошибка.

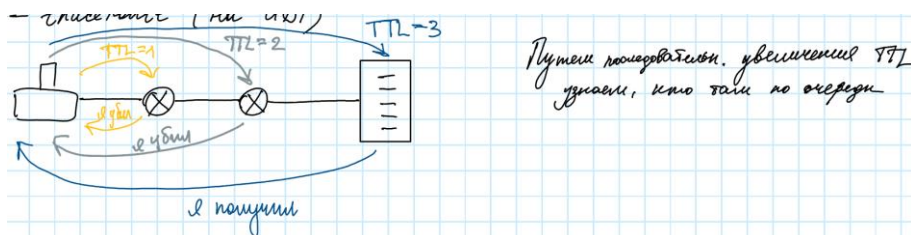
Исключения, когда ICMP не отрабатывает (с какой целью не создается)

- уничтожение широковещательной рассылки
- потеря ICMP
- потеря **не первого фрагмента** фрагментированного пакета (например, когда когда потерян 7-й, если до этого был потерян 3-й)

**Commented [14]:** не в смысле, что у него номер 1, а что он очередной

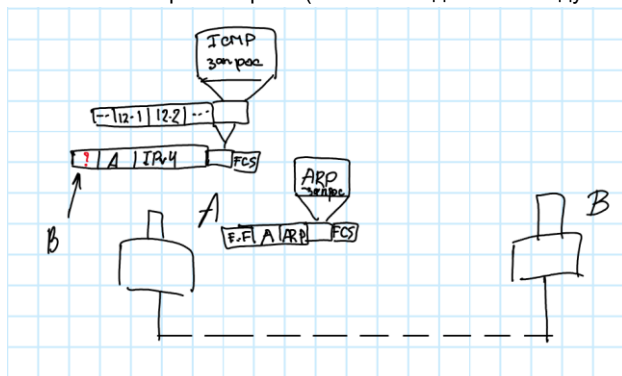
**Утилиты для трассировки** (возвращают путь, по которому идет пакет):

- `tracert` (работает на основе ICMP)
- `tracert` (работает на основе UDP (L4 уровень)) (но есть возможность работы и через ICMP)



### Примеры создания ICMP-запроса:

есть топология point-to-point (2 компа соединены между собой перекрестным кабелем),

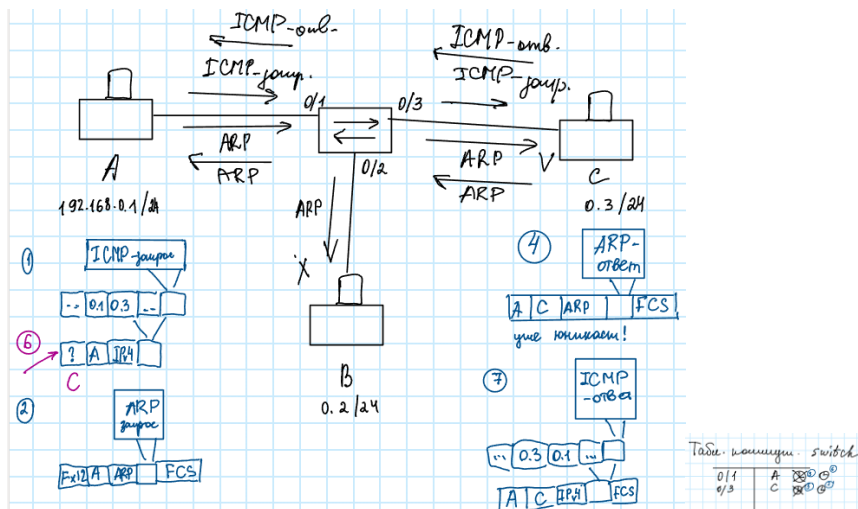


Организовываем возможность пингования их друг другом:

- 1) Задать IP для компов A и B - например 192.168.12.1 / 24 и 192.168.12.2 / 24
- 2) Работая из консоли устройства A, создаем ICMP-запрос, он инкапсулируется в IPv4
- 3) Надо инкапсулировать в Ethernet 2, но мы не знаем mac-адрес получателя, поэтому создаем ARP-запрос, чтобы узнать mac-адрес конечного устройства
- 4) ARP инкапсулируется в Ethernet 2 с широковещательным адресом (FF:FF:FF:FF) и отправляется
- 5) Устройство B дает ARP-ответ, заполняется ARP-таблица на B (появляется запись об A)
- 6) A получает ответ B, подставляет полученный mac-адрес в ранее созданный пакет Ethernet 2 и отправляет его на B.
- 7) B деинкапсулирует полученный пакет, обрабатывает ICMP-запрос и посылает ICMP-ответ на A.

## Лекция 9 (07.11)

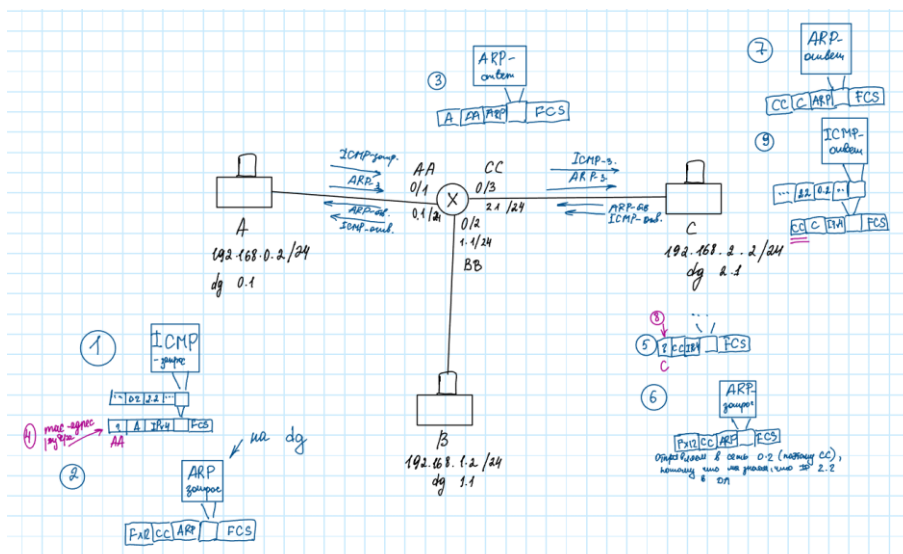
Пример 1 (со свитчем):



Делаем так, чтобы все пинговалось (рассмотри пинг из A в C):

- Выдаем IP-адреса
- Формируем ICMP-запрос, инкапсулируем в IPv4 (в заголовке 0.1 - источник | 0.3 - приемник), инкапсулируем в Ethernet, понимаем, что не знаем mac-адрес получателя, поэтому будем формировать ARP-запрос
- Формируем ARP-запрос, инкапсулируем в Ethernet (пакет выглядит следующим образом FF:FF:FF:FF:FF:FF | A | ARP | [ARP-запрос] | FCS, он идет через свитч на все устройства
- Свитч записывает, что A на интерфейсе 0/1; B видит, что в запросе не он
- C видит себя в запросе и формирует ARP-ответ (пакет выглядит следующим образом A | C | ARP | [ARP-ответ] | FCS)
- Свитч получает пакет с C, записывает, что C на интерфейсе 0/3 и пересылает его на A
- Получаем ARP-ответ на A, дописываем mac-адрес в ICMP-запрос из пункта 1 и отправляем его. При прохождении ICMP через свитч - свитч обновит данные о том, что A на интерфейсе 0/1.
- На C формируется ICMP-ответ, инкапсулируем в IPv4, у которого в заголовке 0.3 | 0.1 | ..., инкапсулируем в Ethernet, у которого в заголовке A | C | IPv4 | ... | FCS, который пойдет из C в A

**Пример 2 (с роутером):**



Каждый интерфейс маршрутизатора имеет свой мас-адрес, потому что каждый интерфейс имеет отдельную сетевую карту

AA и прочее - мас-адреса интерфейсов маршрутизатора

Общаются А и С:

- 0) Выдаем IP-адреса компам и интерфейсам роутера и мас-адреса интерфейса роутера
- 1) С компа А делаем ICMP-запрос
  - а) инкапсулируем в IPv4 (в заголовке ... | 192.168.0.2 | 192.168.2.2 | ... | [ICMP-запрос] )
  - б) инкапсулируем в Ethernet (в заголовке ? | А | IPv4 | [IPv4-пакет] | FCS)
- 2) на А формируем ARP-запрос на dg (default gateway)
  - а) инкапсулируем в Ethernet (в заголовке FF:FF:FF:FF:FF:FF | А | ARP | [ARP-запрос] | FCS)
  - б) запрос идет на роутер
- 3) Роутер формирует ARP-ответ
  - а) инкапсулирует в Ethernet (в заголовке А | АА | ARP | [ARP-ответ] | FCS)
- 4) В ICMP-запрос из пункта 1 подставляем полученный мас-адрес в Ethernet-кадре (АА вместо ?)
- 5) Роутеру пришел инкапсулированный ICMP-запрос
  - а) Он деинкапсулирует его до IPv4 (потому что на L2 мас-адрес его интерфейса АА)
  - б) хочет инкапсулировать IPv4-пакет в Ethernet, но не знает мас-адрес получателя (? | CC | IPv4 | [IPv4-пакет] | FCS).
- 6) Роутер формирует ARP-запрос (в заголовке Ethernet: FF:FF:FF:FF:FF:FF | CC | ARP | [ARP-запрос] | FCS) и отправляет в сеть 192.168.0.2; отправляется только

Commented [15]: это IP-адрес С

с CC, потому что в инкапсулируемом IPv4-пакете из 5b лежит IP-адрес 192.168.2.2

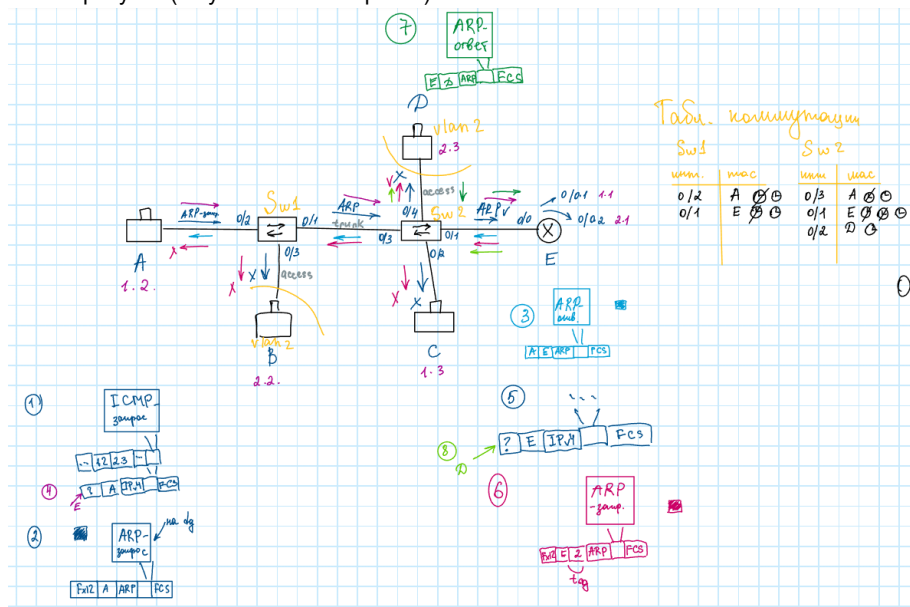
Commented [16]: я бы уточнил этот момент

- 7) С узнает себя по IP-адресу и формирует ARP-ответ (CC | C | ARP | [ARP-ответ] | FCS)
- 8) Роутер получает ARP-ответ, берет из него нужный мас-адрес
  - а) дополняет Ethernet-кадр из пункта 5b и отправляет его на С с инкапсулированным (C | CC | IPv4 | [IPv4-пакет] | FCS).
- 9) С обрабатывает полученный ICMP-запрос и формирует ICMP-ответ,
  - а) инкапсулирует в IPv4 ( ... | 192.168.2.2 | 192.168.0.2 | ... | [ICMP-ответ] )
  - б) инкапсулирует в Ethernet (CC | C | IPv4 | [IPv4-пакет] | FCS), который пойдет из С на роутер
- 10) Роутер деинкапсулирует до IPv4-пакета и увидит не свой IP-адрес
  - а) инкапсулирует в Ethernet (A | AA | IPv4 | [IPv4-пакет] | FCS) и отправляет с AA интерфейса

### Пример 3 (с роутером и VLAN-ами):

Фишка Cisco Packet Tracer: если не знает мак-адрес получателя, то роутер пропускает первый icmp и он умирает за время прохождения

Новый рисунок (он уже после настройки)



между Sw1 и Sw2, Sw2 и E - trunk-и  
между Sw1 и B vlan2, Sw2 и D vlan2 - access

A пингует D

- 1) На A создается ICMP-запрос, инкапсулируется в IPv4, инкапсулируется в Ethernet, выясняется, что не знаем мас-адрес

- 2) На А формируется широковещательный ARP-запрос на dg, чтобы узнать мас-адрес получателя
  - а) Sw1 и Sw2 передают его на роутер и записывают интерфейс, где А
- 3) Роутер получает ARP-запрос и формирует ARP-ответ на А, отправляет его
  - а) свитчи Sw1 и Sw2 передают его на А, записывая в таблицы, на каком интерфейсе Е
- 4) А получает ARP-ответ и подставляет найденный мас-адрес в запрос из пункта 1, отправляет его
  - а) Sw1 и Sw2 перезаписывают интерфейс, где А
- 5) Роутер получает ICMP-запрос, деинкапсулирует его до IPv4-пакета, видит не свой IP, а 192.168.2.3
  - а) роутер инкапсулирует IPv4-пакет в Ethernet, обнаруживает, что нет мас-адреса получателя
- 6) Роутер формирует широковещательный ARP-запрос с тегом 2-го VLAN, (? | Е | 2 | IPv4 | [IPv4-пакет] | FCS), отправляет его
  - а) Sw2 и Sw1 запоминают, на каком интерфейсе Е
- 7) Снимается тег 2, D получает ARP-запрос
  - а) формирует ARP-ответ (Е | D | ARP | [ARP-ответ] | FCS)
- 8) Пакет заходит на свитч 2, и свитч 2 добавляет тег
  - а) Роутер получает ARP-ответ
  - б) Роутер получает мас-адрес D в измененный кадр ICMP-запроса из пункта 5а
- 9) D получает ICMP-запрос и формирует ICMP-ответ на роутер с IP 192.168.1.2
- 10) Роутер подменяет канальный кадр, меняет мас-адрес получателя с Е на А, отправителя - с D на Е и отправляет
- 11) А получает ICMP-ответ

Commented [17]: это тег

Вторая топология - логическая (как передается)

## IPv6

проблема 4-й версии - мало публичных адресов

Изменения:

- 1) Занимает 128 бит против 32 в 4-й версии
- 2) Полная отмена широковещательной рассылки, используется только [multicast](#) ~~broadcast~~-рассылки (решена проблема широковещательного шторма)
- 3) Проще заголовков
- 4) Обязательно в стек включается шифрование
- 5) Появилось свойство plug&play - устройство может себе автоматически настроить IP-адрес из разрешенной сети

**Заголовок** (в скобках - размер в битах)

4 байта	Version (4 бита)	Traffic Class (8 бит)	Flow label (20 бит)	
4 байта	Payload length (16 бит)		Next Header (8 бит)	Hop Limit (8 бит)

16 байт	Source address(128 бит)
16 байт	Destination address(128 бит)

**Version** - написано 6

**Traffic Class** - см. "Тип сервиса" из 4-й версии

**Flow Label** - используется для ускорения обработки пакетов, случайное число, которое устанавливается на все пакеты одной транзакции, выставляется ОС(используется как ключ к кэшу, в нём содержится политика безопасности или другая метаданная информация)

**Payload Length** - длина пакета

**Next Header** - "Код протокола" из 4-й версии, там название следующего протокола

**Hop Limit** - "TTL"

## Лекция 10 (08.11)

### Виды IPv6 адресов

1. *Unicast* - рассылка один к одному
2. *Multicast* - групповой, мб только назначением, рассылка один ко многим
3. *Anycast* - (не было в 4-й версии), рассылка один к ближайшему устройству

09A3:2A30:057A:0000:0000:0000:02AC

9A3:2A30:57A::2AC

9A3:2A30:57A:0.0.0.0:02AC

Можно:

- 1) убирать старшие нули
- 2) использовать :: чтобы заместить n-ое количество нулей(:: - компилятор считает количество бит слева и справа и дополняет нужным кол-вом бит)
- 3) 4 нуля заменять 1-м

### Виды Unicast

1. Глобальный (как публичный)

начинается с 2000::

2. Локальный - состоит из

локального адреса *канала* (идет только до маршрутизатора, используется для рассылки служебки) - FE80::/10

локального адреса *площадки* (идет по всей локальной сети) - FD::/8

3. Локалхост - ::1 (все нули кроме 1)
4. Неизвестный адрес (при включении) - :: (все нули)

**Глобальный адрес, путешествующий в интернете**



реестр (23) | ISP (32) | префикс площадки (48) | префикс подсети (64) | ID интерфейса (127)

Организация IANA выдает реестр, региональный регистратор выдает ISP, локальный провайдер выдает префикс площадки

#### Локальный адрес площадки:

FD (8) | Global ID (48) | Subnet (64) | ID интерфейса (127)

FD - отметка, что это локальный адрес

Global ID - ID локальной сети

Subnet - ID локальной подсети

#### Локальный адрес канала

FE80(10) | (64) | ID интерфейса

#### Способы получения ID интерфейса:

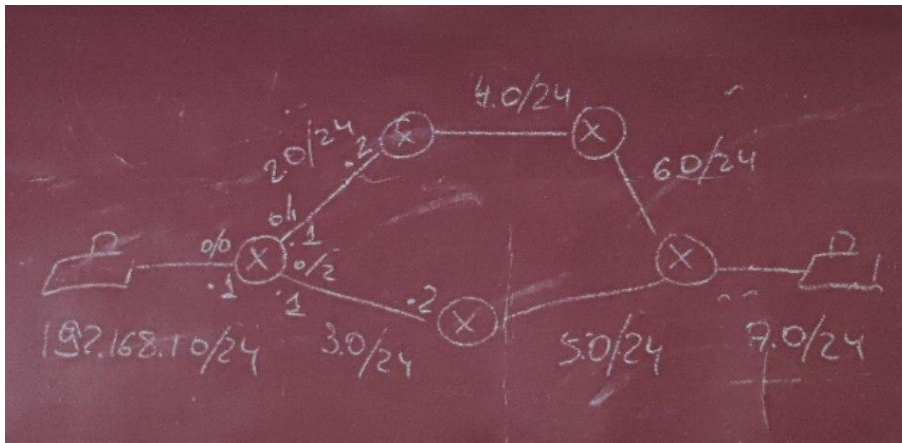
1. Настроен руками
2. Через DHCP
3. Автоматически получить через mac-адрес - используется EUI-64, который делает следующее: ID организации из mac-адреса | FFFE | ID устройства из mac-адреса
4. Самостоятельно, используя plug&play - канал определяется на устройстве, площадка определяется через multicast-запрос на маршрутизатор и ответ от него

## Маршрутизация

Таблица маршрутизации содержит *оптимальный маршрут* (оптимальность определяется по каким-либо критериям, например, по метрике - расстояние от данного маршрутизатора до сети назначения, по хопам, по пропускной способности, по надежности, по нагрузке каналов, по задержке)

Таблица маршрутизации содержит **2 типа записей**:

- 1) *Статические*
- 2) *Динамические* (источники адресов сетей)
  - а) *Напрямую подключенные сети* - как только мы даем ID интерфейсу маршрутизатора в таблице маршрутизации появляется запись об этом
  - б) *Динамические протоколы маршрутизации* - заставляют маршрутизаторы общаться между собой



(.1 и .2 - пронумерованные последние октеты IP-адресов)

dc - directly connected - напрямую соединенные

192.168.1.0/24 DC fa 0/0

192.168.2.0/24 DC fa 0/1

192.168.3.0/24 DC fa 0/2

маршрут до 7-й сети с минимальным количеством хопов:

192.168.7.0/24 {120/2} via 192.168.3.2 fa 0/2

120 - (маршрутное) административное расстояния - степень доверия источнику информации (чем меньше, тем больше доверяем, макс. 255)

2 через slash после административного расстояния - метрика - количество хопов

после via - IP next hop (следующего прыжка)

дальше - с какого интерфейса выпустить

### Параметры протоколов маршрутизации

1. *Оптимальность алгоритма* - характеризует способность алгоритма маршрутизации выбирать наилучший маршрут, который зависит от показателей (hop, доверие и др.) и веса этих показателей при расчете маршрута
2. *Низкие и непроизводительные затраты* - разрабатываются максимально простые алгоритмы маршрутизации для того, чтобы эффективно обеспечивать свои функциональные возможности, при минимальных затратах ПО
3. *Стабильность работы* - устойчивость в работе, т.е. четкое функционирование в случае неординарных и непредвиденных обстоятельств или некорректной настройке.
4. *Быстрая сходимость алгоритма* - (сходимость - процесс соглашения между всеми маршрутизаторами об оптимальных маршрутах)

# Лекция 11 (14.11)

## Классы протоколов маршрутизации

### 1-й вариант классификации:

1. *Статические* - человек сел и сам прописал, минус: не подстраивается под изменения в режиме настоящего времени
2. *Динамические* - работают на основе рассылки обновлений, за счет которых маршрутизатор строит таблицу маршрутизаций, минус: нужно время на обработку изменений

В жизни статика позволяет выйти в "мир" (глобальную сеть), а динамические в локальной сети

### 2-й вариант классификации

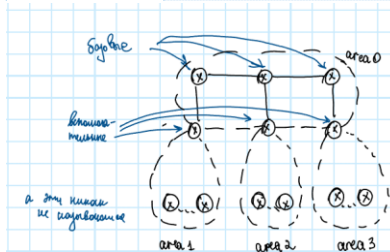
1. *Одномаршрутные* - всегда только один маршрут (оптимальный)
2. *Многомаршрутные* - n количество может храниться, по умолчанию 4, но можно руками перенастроить количество n. Если метрика будет одинаковой, то маршрут считается одинаковым. Все маршруты должны узнаваться одним и тем же протоколом - это значит, что оптимальность будет рассчитываться по одному критерию (пропускная способность и прочее).

2 варианта балансировки выбора маршрута (на маршрутизаторе используется для выбора маршрута для пакета когда есть нескольких оптимальных маршрутов и для того, что выполнить некоторые условия):

1. По количеству пакетов: один налево, другой направо (с целью распределения нагрузки по сети)
2. Смотрим на транзакции: все пакеты одной и той же транзакции идут по одному и тому же маршруту

### 3-й вариант классификации:

1. *Одноранговая* - все маршрутизаторы равны по правам, все маршрутизаторы знают о всех локальных сетях вашей сети (например: на предприятии есть локальная сеть с 10-ю роутерами, и все они знают все друг о друге)
2. *Иерархическая* - есть основные (базовые) и вспомогательный



Сеть делится на зоны.  
Если 0-вая (наш-остов/костяк).  
Если граничные маршруты, связывающие зоны.  
Зоны 1, 2, 3 и т.д. связываются только через зону 0.  
Внешняя часть сети делает вид, что зона 0.  
Это снижает нагрузку на сеть.  
Маршрутизаторы внутри зон.

**Commented [18]:** реализуется на основе одного протокола, в отличие от стандартного разделения сетей

в area 0 - базовые  
граничные - вспомогательные (они организуют связь зон (area-й))  
area 0 над остальными сетями

есть 3 граничных маршрутизатора, которые не пропускают служебную информацию  
маршрутизаторы знают только о тех маршрутизаторах, которые в одной сети с ним, в том числе и о граничных маршрутизаторах, которые позволяют выйти в area 0  
**рисунок**

#### 4-й вариант классификации (только динамические):

1. *Дистанционно-векторные* - на регулярной основе отправляют обновления своим соседям, внутри обновления чаще всего таблица маршрутизации (таблица оптимальных маршрутов). Обновление соседа тоже пересылается.  
-: "глухой телефон" - информация может искажаться  
+: Быстрые, т.к. добавляют свои сведения в общие таблицы.
2. *Состояния канала* - отправляют обновления (не обязательно соседним маршрутизаторам), которые содержат обновления только о своих собственных каналах.  
-: работает медленнее  
+: Надежные, поскольку роутер верит только себе и рассказывает информации только о себе, маршрутизатор сам строит оптимальные маршруты (не на основе сформированных другими, а на основе своих сведений)

## Протокол RIP - *Routing Information Protocol*

Динамический, одноранговый, многомаршрутный, дистанционно-векторный, административное расстояние - 120, в качестве метрики - хопы(максимум 15 хопов), протокол прикладного уровня, работает поверх UDP, инкапсулируется в IP

Т.к. он дистанционно-векторный (работает на основе 3-х таймеров:

1. *Таймер регулярной рассылки*(30 сек) - как часто будет кидать рассылку
2. *Таймер таймаут* - запускается, когда запись появляется в таблице маршрутизации; обновляется, когда приходит апдейт с этой записью; а если по истечении времени таймера апдейт не пришел, то накладывает на маршрут "бесконечную метрику" - 16 хопов)
3. *Таймер garbage collector* - запускается, если таймер таймаута истек - по истечении времени запись удаляется

Сообщение RIP-а 1-й версии:

Команда (8 бит)	Версия (8 бит)	Ø
ID адресного семейства		Ø
IP адреса сети		

Ø
Ø
Метрика

**Команда** - определяет функционал (что делает данное сообщение), варианты:

1. Запрос (состоит из 1-й строчки) - маршрутизатор появляется в сети и использует это для запроса таблиц маршрутизации со всех своих настроенных интерфейсов
2. Ответ - ответ на запрос
3. Регулярная рассылка - выглядит как ответ

**Версия** - версия протокола

**ID адресного семейства** - создавалось из расчета, что будут работать не по TCP/IP

**IP адреса сети** - сеть назначения, т.е. о каком IP эта запись (IP из таблицы маршрутизации)

**Метрика** - сколько нужно пройти для достижения сети из IP адреса сети

Команда (8 бит)	Версия (8 бит)	Ø
ID адресного семейства	Тег маршрута	
IP адрес сети		
Маска (для поддержки бесклассовой маршрутизации)		
IP next hop (содержит IP адрес маршрутизатора к месту назначения. Значение 0.0.0.0 — хопом к месту назначения является отправитель пакета. Необходимо, если протокол RIP не может быть запущен на всех маршрутизаторах)		
Метрика		

последние 5 строк повторяются для накопления данных из таблицы маршрутизации, в 1-й версии так же

**Тег маршрута** - ID протокола. Предназначен для разделения «внутренних» маршрутов от «внешних», взятых, например, из IGRP или EIGRP

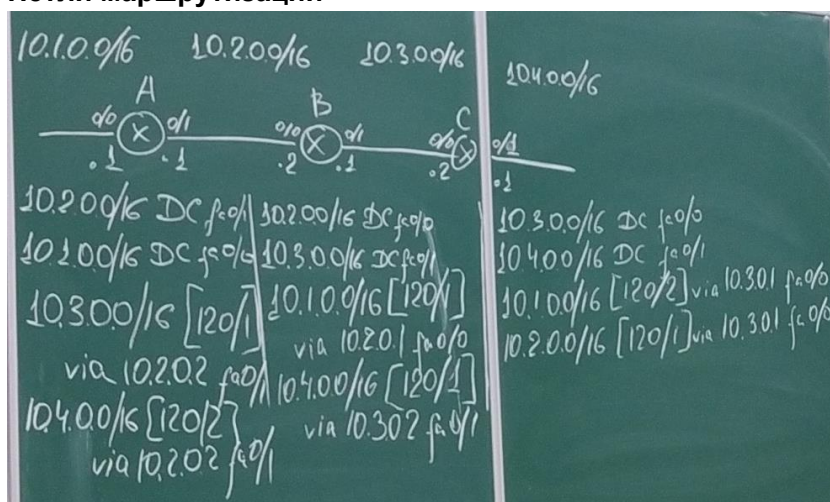
На основе него определяется, какой будет маршрут

1. Внутренний - необходимо работать в рамках одного протокола
2. Внешний - необходимо связываться с другим протоколом, связь будет статической

**Убрали сетевую рассылку**

Чтобы избежать ненужной нагрузки на hosts, не участвующие в маршрутизации, RIPv2 [рассылает](#) multicast-ом всю таблицу маршрутизации всем соседним маршрутизаторам по адресу [224.0.0.9](#), в отличие от RIPv1, который использует [широковещательную рассылку](#). Одноадресная адресация (unicast) по-прежнему разрешена для специальных приложений.

## Петли маршрутизации



### таблица маршрутизации А

10.2.0.0/16 DC fa0/1  
10.1.0.0/16 DC fa0/0

### таблица маршрутизации В

10.2.0.0/16 DC fa0/0  
10.3.0.0/16 DC fa0/1

### таблица маршрутизации С

10.3.0.0/16 DC fa0/0  
10.4.0.0/16 DC fa0/1

нужна 2-я версия RIP, т.к. бесклассовая адресация (маска 16 и сеть - 10.)

### таблица маршрутизации А

10.2.0.0/16 DC fa0/1  
10.1.0.0/16 DC fa0/0  
10.3.0.0/16 [120/1] via 10.2.0.2 fa 0/1  
10.4.0.0/16 [120/2] via 10.2.0.2 fa 0/1

### таблица маршрутизации В

10.2.0.0/16 DC fa0/0  
10.3.0.0/16 DC fa0/1

10.1.0.0/16 [120/1] via 10.2.0.1 fa 0/0  
10.4.0.0/16 [120/1] via 10.3.0.2 fa 0/1

#### таблица маршрутизации С

10.3.0.0/16 DC fa0/0  
10.4.0.0/16 DC fa0/1  
10.2.0.0/16 [120/1] via 10.3.0.1 fa 0/0  
10.1.0.0/16 [120/2] via 10.3.0.1 fa 0/0  
Добавим проблему - отвалилась сеть 10.4.0.0

#### таблица маршрутизации С

~~10.4.0.0/16 DC fa0/1 (зачеркнуто)~~

10.4.0.0/16 [120/2] via 10.3.0.1 fa0/0 (добавилась строка)

(у нас РИП еще не дошел, роутер, от которого отвалилось, знает об изменении)

#### На РК2:

про сетевой уровень, IPv4, IPv6, STP, ARP, ICMP, DHCP, RIP, классификация протоколов маршрутизации и петли, OSPF  
3-я задача во всех билетах - посчитать IP с использованием CIDR, VLSM

## Лекция 12 (28.11)

### Метод борьбы с петлями маршрутизации (для любого дистанционно-векторного протокола)

Метод настраивается одновременно на всех роутерах  
(примеры приводятся для топологии из прошлой лекции)

#### 1. Расщепление горизонта

Информация не отправляется назад источнику информации. В при формировании апдейта для С не будет формировать информацию о 4-й подсети, тем самым информация о 4-й подсети в 2-х хопх исчезает на роутере С. (потому что В про 4-ю подсеть рассказал С)

то есть он не присылает апдейты тому, от кого они получены

#### 2. Отравление маршрута

Маршрут до сети назначения, который вышел из строя, будет отправлен с бесконечной метрикой. Когда умерла (сломался или выключился интерфейс) 4-я подсеть, то С отправляет апдейт на В с бесконечной метрикой (в РИПе метрика - количество хопов).

#### 3. Обратное отравление маршрута

(Не может работать с расщеплением горизонта).

Отправляем маршрут источнику информации с бесконечной метрикой. Обычно: 4-я сеть упала, С не успел сообщить В, тогда С узнает от него, что 4-я подсеть в 2-х хопх (это стандартная ситуация); в случае с обратным отравлением: С узнает от В, что 4-я

подсеть с бесконечной метрикой ("бесконечная" = максимальное количество хопов в РИПе)

#### 4. Таймер удержания

Запускается на маршрут, который вышел из строя, и в течение работы данного таймера не принимается маршрут с метрикой, хуже имеющейся.

Когда 4-я подсеть рухнет, на С встает таймер удержания, В присылает апдейт о том, что подсеть 4 в 1-м хопе от нее, следовательно для С она в 2-х хопов, но С не примет апдейт, т.к. у него 4-я подсеть записана с 1-м хопом, а  $2 > 1$  (срабатывает таймер удержания).

Таймер должен быть больше времени жизни любого маршрута в таблице маршрутизации, чтобы мы не приняли апдейт.

#### 5. Триггерное сообщение

Триггерное сообщение отправляется сразу после изменения в сети, а не по таймеру регулярной рассылки

При одновременной работе 2 и 5 методов может возникать ошибка, потому что, если С отправит В сообщение о смерти 4-й подсети, а В в этот момент может прислать апдейт С, что знает о 4-й подсети в 1-м хопе (для С - 2 хопов)

EIGRP - можно посмотреть информацию о том, как реализованы методы борьбы с петлями маршрутизации

## OSPF - Open Shortest Path First

Commented [19]: <https://ru.wikipedia.org/wiki/OSPF>

Протокол состояния канала - означает, что маршрутизаторы в своих апдейтах обмениваются не таблицами маршрутизации, а своими линками и не на регулярной основе (только по запросу или при изменении).

административное расстояние = 110

метрика  $M = 10 \text{ Мбит/с} \div \text{Пропускная способность}$

**Роутер сам ищет оптимальный маршрут** на основе алгоритма Дейкстры для графов, потому что не верит другим роутерам о том, что они нашли оптимальный маршрут.

OSPF - иерархический протокол - автономная система делится на зоны и маршрутизаторы строят графы только внутри своей зоны. Тем самым мы уменьшаем количество служебки в каналах и тем самым увеличиваем максимальную скорость в каналах

в лабе когда прописываешь network, то прописываешь и area

Внутри одной зоны мб много сетей, но каждая сеть должна находиться только в одной зоне. (1 зона =  $\infty$  сетей, 1 сеть = 1 зона)



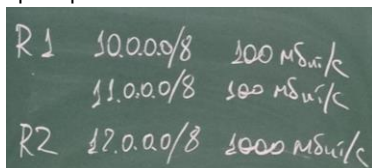
#### Заголовок сообщения OSPF

Версия (8 бит)	Тип (8 бит)	Длина сообщения (16 бит)
ID маршрутизатора (32 бит)		
ID области (32 бит)		
Контрольная сумма (16 бит)	Тип идентификации (16 бит)	
Идентификация		
Идентификация		

**Версия** - версия OSPF, всего есть 3, для IPv4 используется версия 2, для IPv6 используется версия 3

**Тип** - функционал сообщения (5 разновидностей):

1. Hello (единственное рассылается на регулярной основе) - смысл сообщения "я живой, не убирай меня из своего графа"
2. Data Base Description - описание топологической базы данных, в данном сообщении вся база данных зоны, включая все маршрутизаторы с их линками, используется для того, чтобы новый роутер получил всю необходимую информацию  
пример DBD:



```

R1  10.0.0.8  100 Мбит/с
    11.0.0.8  100 Мбит/с
R2  12.0.0.8  1000 Мбит/с
  
```

3. Link State Request - запрос, который отправляется, когда 1. ты новенький, 2. когда время жизни подходит к окончанию и нужно узнать, сохранять ли его
4. Link State Update - сообщение, генерируемое в следующих случаях:
  - a. Новый роутер хочет рассказать о себе (о линках и пропускной способности),
  - b. Произошло какое-то изменение на роутере и он изменяется,
  - c. В ответ на запрос, который отправлялся в ответ на запрос по таймеру
5. Link State Acknowledgement - подтверждение получения базы данных. (есть таймер на повторную отправку БД, если Ask не был получен)

**ID маршрутизатора (Router ID)** - занимает 32 бита, записывается как IP-адрес, выбирается несколькими способами:

1. ручная настройка;

Commented [20]: время жизни маршрута?

2. самостоятельная настройка роутером на основе IP-адресов виртуальных интерфейсов (loopback): берет максимальный,
3. если виртуальных нет, то смотрит физические и берет максимальный (продолжение 2-го)

**ID области** - то, в какой области остается сообщение -> за какую область его кидать нельзя

**Контрольная сумма** - определяет наличие ошибки

**Тип идентификации** -

1. как в RIPv2, нет никакой идентификации,
2. идентификация открытым текстом (на маршрутизаторе настраивается пароль и этот же пароль заносится в сообщение в идентификацию),
3. способ шифрования MD5 - алгоритм однозначного шифрования

**Commented [21]:** 1. Если явно задан ID маршрутизатора (с помощью команды `router-id`), используется назначенный вручную ID.  
2. Если ID маршрутизатора не задан, то присваивается больший адрес из настроенных на маршрутизаторе loopback интерфейсов.  
3. При отсутствии loopback интерфейсов принимается больший адрес из всех включенных на маршрутизаторе

## Лекция 13 (05.12)

### IGRP - Interior Gateway Routing Protocol

Дистанционно-векторный протокол, проприетарное решение Cisco (другие маршрутизаторы его не понимают), работает на основе тех же 3-х таймеров, что и рип. Таймер регулярной рассылки = 90 секунд.

**Commented [22]:** <http://citforum.ru/nets/ito/24.shtml>

**Commented [23]:** <https://www.cisco.com/c/en/us/support/docs/ip/interior-gateway-routing-protocol-igrp/26825-5.html>

Метрика считается как (списывает из телефона):

$$M = [K1*B + K2*B / (256 - L) + K3*D] * K5 / (K4 + R)$$

K1-K5 - коэффициенты, которые могут изменяться программно

B - bandwidth - пропускная способность

L - load - загрузка канала

D - delay - задержка

R - reliability - надежность

Если значения K по умолчанию (K1=K3=1, K2=K4=K5=0), то метрика считается как **M = B + D**

Как выглядит IGRP (в скобках значение крайней правой границы в битах):

Version (4)	Opcode(8)	Edition(16)	AS number (24)	
Number of interior routes(16)			Number of system routes (32)	
Number of exterior routes			Checksum	
IP address (24)				

Delay			
Bandwidth			
MTU			
Reliability	Load	Hop Count	

**Version** - версия протокола

**Opcode** - определяет функционал (два вида:

1. запрос - только 1-я строка (когда появляется новый маршрутизатор и ему нужен список роутеров сети),
2. пакет корректировки - регулярный update / ответ на запрос)

**Edition** - контроль "версии" пакетов (просто число, которое будет инкрементироваться при каком-либо изменении в сети), нужен для применения актуальных апдейтов (пакеты со старым edition не применяются)

**AS number** - номер **автономной системы** (совокупность сетей под общей политикой администрирования)

**Number of interior routes** - количество **внутренних маршрутов** в таблице маршрутизации

**Commented [24]:** маршрут определенный посредством самого протокола

**Number of system routes** - количество статических маршрутов в таблице маршрутизации

**Number of exterior routes** - количество **внешних маршрутов** в таблице маршрутизации

**Commented [25]:** маршруты, определенные другими протоколами

**Checksum** - контрольная сумма

**IP address** (24 бита, т.к. протокол работает на классовой маршрутизации, т.к. сеть всегда там указана в 3-х октетах), Delay, Bandwidth () - для расчета метри

**MTU** (maximum transmission unit) - максимальный размер данных, которые можно передать через канал за единицу времени (за одну транзакцию)

**Commented [26]:** [https://ru.wikipedia.org/wiki/Maximum\\_transmission\\_unit](https://ru.wikipedia.org/wiki/Maximum_transmission_unit)

**Reliability** - надежность

**Load** - загрузка канала

**Hop Count** - количество хопов

## EIGRP - Enhanced Interior Gateway Routing Protocol

гибридный протокол - одновременно и дистанционный-векторный и состояния канала метрика - считается как в IGRP

новшества:

1. Уменьшенная конвергенция (быстрая сходимость сети), за счет того, что помнит не только оптимальные маршруты, но и все возможные

2. Сниженное потребление полосы пропускания - на регулярной основе только Hello сообщения, все генерируется на групповой адрес (multicast рассылка)
3. Позволяет балансировать не только между маршрутами с одинаковой метрикой, но и между маршрутами с разной метрикой (можно указать погрешность метрики, в то время, как в других она должна быть одинаковой)
4. В остальных протоколах автосуммирование работает по классовому признаку, а здесь не только по классовому признаку

### Рисунок

Группировка (не на основе классов):

192.168.1.0 / 25 -> 0/0

192.168.1.128 / 25 -> 0/1

Группировка (на основе классов, как в РИПе):

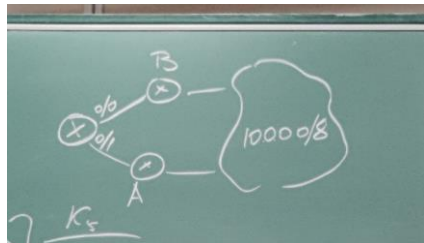
192.168.1.0 / 24 -> 0/0

192.168.1.0 / 24 -> 0/1

все это производится на основе 3-х таблиц:

1. таблица соседей
2. таблица топологий - все возможные маршруты
3. таблица оптимальных маршрутов (маршрутизации)

Как это работает: есть роутер, подсоединенный к 2-м другим



Таблицы для самого левого маршрутизатора:

Таблица соседей:

Router A	0/1
Router B	0/0

Таблица топологий:

сеть	метрика (от текущего роутера до сети назначения)	объявленная метрика (от соседа до сети назначения)	сосед
10.0.0.0/8	2500	1500	Router A
10.0.0.0/8	2000	1000	Router B

Таблица маршрутизации - можем сами нарисовать

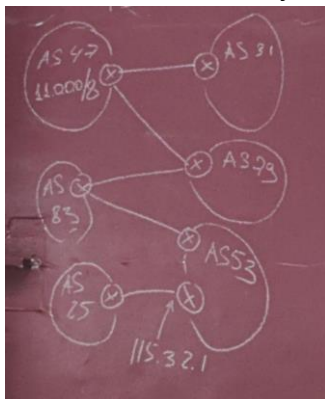
**Commented [27]:** feasible route - второй лучший после оптимального (в таблицу маршрутизации не заносится)

**Commented [28]:** Тогда для пущей точности Advertised Distance и Feasible Distance

**Commented [29]:** берется, как оптимальный

## Лекция 14 (06.12)

### BGP - Border Gateway Pr.



AS - автономная система (совокупность сетей под общей политикой администрирования)

**BGP** - протокол прикладного уровня, обеспечивает классовую маршрутизацию в глобальных сетях, оперирует автономными системами. Инкапсулируется в TCP (порт 179). Записывает все сети, через которые надо пройти, чтобы достичь сеть назначения. В примере выше хотим добраться из AS25 в AS47.

*Формат таблицы маршрутизации:*

маршрут из автономных систем, IP-next hop, адрес сети назначения.

При написании маршрута исходная AS в таблицу маршрутизации (см. ниже) не записывается.

AS53, AS83, AS79, AS47,	115.3.2.1,	11.0.0.0/8
маршрут,	IP-next hop,	куда (адрес сети назначения)

### Разновидности BGP

**Commented [30]:** [https://ru.wikipedia.org/wiki/Border\\_Gateway\\_Protocol#:~:text=BGP%20%D1%8F%D0%B2%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D0%BE%D0%BC%20%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE%20%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F,%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F%20TCP%20\(%D0%BF%D0%BE%D1%80%D1%82%20179\)](https://ru.wikipedia.org/wiki/Border_Gateway_Protocol#:~:text=BGP%20%D1%8F%D0%B2%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D0%BE%D0%BC%20%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE%20%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F,%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F%20TCP%20(%D0%BF%D0%BE%D1%80%D1%82%20179))

1. eBGP, Внешние - между маршрутизаторами, которые в разных автономных системах
2. iBGP, Внутренний - нужен для связи между граничными маршрутизаторами одной автономной системы

-----  
\\на заметку\\

Если я хочу поймать трафик чужой автономной системы, то я в своей автономной системе напишу свою новую сеть, и поймаю пакет с ней от атакуемой системы.

Средства защиты трафика, которые используются провайдером

1. Электронные подписи,
2. Регистрация автономных систем,
3. Шифрование трафика,
3. Накат IPsec.

-----  
Типы сообщений в BGP:

1. Установление связи
2. Ответ
3. Изменение состояния
4. Мониторинг на ошибки (проверка правильности работы протокола)

// дополнительно

#### Типы сообщений BGP

BGP использует при работе четырех типа сообщений:

- OPEN - Устанавливает и настраивает смежность BGP.  
Сообщение OPEN используется для установки смежности BGP. Обе стороны согласовывают вероятности сеанса до установки пиринга. Сообщение OPEN содержит номер версии BGP, ASN исходного маршрутизатора, время удержания, идентификатор BGP и другие дополнительные параметры, которые определяют возможности сеанса.
- UPDATE - Объявляет, обновляет или отменяет маршруты.  
Сообщение Update объявляет любые возможные маршруты, отменяет ранее объявленные или может делать и то, и другое. Сообщение UPDATE включает информацию о доступности сетевого уровня (NLRI), которая включает префикс и связанные с ним PA BGP при объявлении префиксов. Изъятые NLRI включают только префикс. Сообщение UPDATE может действовать как Keepalive для уменьшения ненужного трафика.
- NOTIFICATION - Указывает на состояние ошибки соседу BGP.  
Сообщение NOTIFICATION отправляется, когда в сеансе BGP обнаруживается ошибка, такая как истечение таймера удержания, изменение возможностей соседей или запрос сброса сеанса BGP. Это сообщение приводит к закрытию BGP-соединения.
- KEEPALIVE - Обеспечивает работоспособность соседей BGP.  
BGP не полагается на состояние TCP-соединения, чтобы гарантировать, что соседи все еще работают. Сообщения Keepalive обмениваются каждые треть таймера удержания, согласованного между двумя маршрутизаторами BGP. Если время удержания установлено равным нулю, сообщения Keepalive между соседями BGP не отправляются.

## Транспортный уровень

## TCP - Transmission Control Pr.

протокол с установлением соединения, для адресации используется номер порта (виртуальная "история", которая выделяется ОС для всех приложений, которые используют сеть)(данные уже на оборудовании). Понимает, какому приложению на устройстве надо отправить.

Подгруппы номеров портов:

1. Well-known ("хорошо известные") - 1-1023, все зарегистрированы за определенными протоколами (серверные порты)
2. Зарезервированные - 1024-49151, существование зависит от используемой ОС
3. Динамические - 49152-65535, выделяются ОС на клиентской части

Заголовок

(в скобках правая граница раздела)

Порт источника		Порт назначения	
Номер последовательности			
Номер подтверждения			
Длина заголовка (4)	Зарезервировано (10)	Флаги (16)	Размер окна (32)
Контрольная сумма		Указатель важности	
Опции - поле переменной длины			

**Порт источника**

**Порт назначения** (получателя)

**Номер последовательности** - если SYN=1, то это начальное значение последовательности (ISN), если SYN=0, то это номер 1-го байта передаваемых данных

**Номер подтверждения** - если SYN=1, то номер последовательности+1, если SYN=0, то это номер последовательности, ожидаемой в следующий раз (то есть мы тут плюсуем размер полученного сегмента и еще 1 байт)

**Длина заголовка** - ...

**Зарезервировано** -

CWR - Control Window Reduced - был изменен размер окна (останавливается или возобновляется передача)

ECE - Echo-ECN - явное сообщения об ошибке (отметка, что устройство его понимает)

**Флаги** -

URG - "важность", если выставлен, то нужно смотреть на указатель важности

ACK - "подтверждение", если выставлен, то данный сегмент является подтверждением получения, предыдущего сегмента

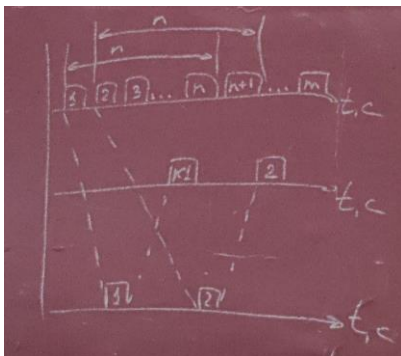
PSH - если выставлен, то данные нам доставят немедленно

RST - аварийное закрытие соединения, используется, когда таймер работоспособности выходит из строя

SYN- "установление соединения", по нему находится начало передачи, выставлен при установлении соединения

FIN - "корректное закрытие соединения", корректное закрытие программы, работающей с сервером

**Размер окна** - объем данных, который можно передать, до получения подтверждения получения предыдущего



(на данном рисунке  $n=1$ )

2 верхних оси - отправитель

нижняя - получатель

размер окна -  $n$

1 - сегмент, дошедший до получателя

K1 - подтверждение получения (на каждый сегмент)

Когда пришло подтверждения, то мы увеличиваем размер окна на 1 (если хороший канал связи), или уменьшаем на 1 (если плохой канал связи)

**Контрольная сумма** - ищет ошибки

**Указатель важности** - работает только с флагом, показывает последний байт важных данных

**Опции** - поле переменной длины, сюда можно поместить TimeStamp (метка, когда сегмент был отправлен), Noop (типа Hello-сообщение) и пр.

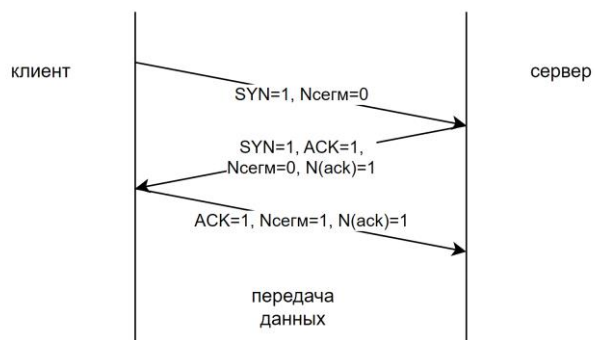
Тип (1 байт)	Длина (1 байт)	Данные (как получится)	Заполнитель (16 - как получится)
--------------	----------------	------------------------	----------------------------------



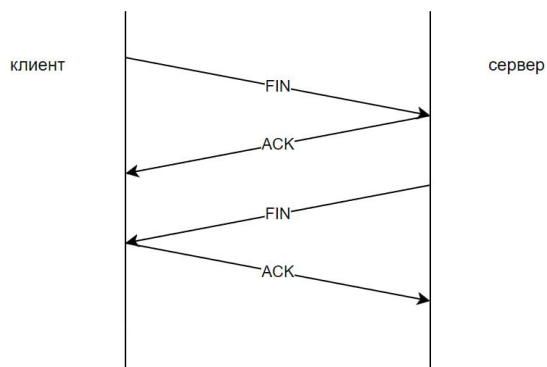
## Лекция 15 (19.12)

### Установление соединения

Установление TCP-соединения называется *трехсторонним рукопожатием* или *трехсторонним квитированием*

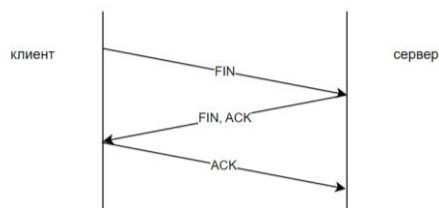


“Древний” вариант завершения соединения:



не рисуется флаг ack на первой линии вместе с сегментом FIN, но он там есть (подтверждает передачу, не имеет отношения к закрытию соединения)

“Современный вариант”



## Таймеры

1) Таймер повторной передачи - запускается в момент отправки сегмента. Если по истечении работы не пришло подтверждение, происходит повторная передача. (подтверждение может не прийти по 3-м причинам)

1. Ack отправлено, но где-то затерялось по пути к отправителю
2. Отправили данные, но получатель не получил и не знает, что мы в принципе передавали
3. Произошло изменение нагрузки на сеть, ack не дошел вовремя -> таймер увеличивается на значение по формуле (примерно 10-12 раз)

Таймер не постоянен, подстраивается под настройки сети)

2) Таймер контроля работоспособности - таймер, контролирующий работу самого соединения, без него соединение работало бы вечно и не было бы аварийных завершений сигнала (по умолчанию 2 часа). Сервер кидает запрос "клиент, ты живой?", варианты развития событий:

1. Клиент живой, таймер обновляется
2. Клиент выключен или перезагружается или неработоспособен
3. Клиент перезагрузился, запрос от сервера дошел, но приложение уже не работает и порт закрыт -> ОС говорит, что можно закрыть соединение
4. Клиент работоспособен, но недостижим -> клиент нормально функционирует, но пропал из сети

(2 и 3 одинаково интерпретируются сервером)

3) Таймер запросов - запускается, когда приходит сообщение о размере окна, равному 0, что означает, что передача остановлена. Если по истечению работы таймера не пришла информация о том, что размер окна изменился - отправляется запрос на размер окна. Если пришел равный 0 - сброс таймера, если не 0 - повторное установление передачи.

4) 2MSL - удвоенное время жизни сегмента (2 x maximum segment line). Запускается после завершения соединения и в течение работы таймера ресурсы старого соединения не выдаются новому. Смысл в том, что если клиент быстро закрыл соединение, и не все сегменты до него дошли, следовательно, они будут уничтожены,

## UDP - User Datagram Pr.

Протокол транспортного уровня без установления соединения

В основном используется для стримов.

Как выглядит пакет

Порт источника (16)	Порт получателя (16)
Длина сегмента	Контрольная сумма

(в скобках размеры в битах)

Wildcard - используется для установления соединения,

192.168.1.0 0.0.0.255 -> выполнить действие permit

под это действие попадает любой хост из сети 192.168.1

192.168.1.0 0.0.255.255 -> выполнить действие deny

под это действие попадает любой ip-адрес из 192.168.0.0

192.168.64.0 0.0.63.255 -> выполнить действие permit

0. 0.63.255

Проверим выполнение для 192.168.69.73

если бит = 0 -> важен, должен совпасть с айпи слева и надо сверить

если бит = 1 -> не важен, пропускаем

**0**100 0000 = 64

0011 1111 = 63 - wildcard

**0**100 0101

выделенное жирным совпало, следовательно для 192.168.69.73 действие выполнится, как и для любого ip-адреса, принадлежащего [64, 127]

## ACL вставка из лаб

Спасибо Павлу Смирнову за подготовку данной теории

**ACL** (Access Control List) — это набор текстовых выражений, которые что-то разрешают, либо что-то запрещают. Обычно ACL разрешает или запрещает IP-пакеты,

но помимо всего прочего он может заглядывать внутрь IP-пакета, просматривать тип пакета, TCP и UDP порты

Виды фильтраций:

1. На интерфейсе: пакетная фильтрация (наша лаба)
2. На линии Telnet: ограничения доступа к маршрутизатору
3. VPN: какой трафик нужно шифровать
4. QoS: какой трафик обрабатывать приоритетнее
5. NAT: какие адреса транслировать

**Пакетная фильтрация** - работают на интерфейсах роутерах. Сначала создаются отдельно, потом связываются с определенным интерфейсом.

Тот трафик, который входит в маршрутизатор называется входящим, тот который из него выходит — исходящий. Соответственно ACL размещаются на входящем или на исходящем направлении.

Пример: есть 2 интерфейса - fa0/0 (слева) и fa0/1 (справа). Нужно ограничить прохождение трафика слева направо. Существует 2 подхода:

1. Настроить ACL на fa0/0 как входящую фильтрацию
2. Настроить ACL на fa0/1 как исходящую фильтрацию

Выбор зависит от типа трафика и топологии. Для ограничения доступа из внешней сети (например, ISP на fa0/0 а локалка на fa0/1), лучше выбрать 1-ый способ (подумайте почему).

Сам же ACL представляет собой набор текстовых выражений, в которых написано permit (разрешить) либо deny (запретить), и обработка ведется строго в том порядке в котором заданы выражения.

ACL разделяются на два типа:

- **Стандартные** (Standard) (от 1 до 99, от 1300 до 1999): могут проверять только адреса источников
- **Расширенные** (Extended) (от 100 до 199, от 2000 до 2699): могут проверять адреса источников, а также адреса получателей, в случае IP ещё тип протокола и TCP/UDP порты - работают медленнее

**Нельзя разместить более 1 списка доступа на интерфейс, на протокол, на направление!**

То есть для IP на fa0/1 in - максимум 1 ACL, для IP на fa0/1 out - тоже максимум 1 ACL и тп

**ACL не действует на трафик, сгенерированный самим маршрутизатором**

### Настройка ACL (Стандартный)

```
access-list <номер списка> {permit | deny } {address | any | host} [source-wildcard]
```

- permit: разрешить
- deny: запретить
- address: запрещаем или разрешаем сеть
- any: разрешаем или запрещаем всё
- host: разрешаем или запрещаем хосту
- source-wildcard: WildCard маска сети

Пример

```
access-list 1 permit 192.168.1.0 0.0.0.255
```

где 192.168.1.0 0.0.0.255 - источник

### Настройка ACL (Расширенный)

```
access-list {номер_списка} {разрешение | запрещение} {протокол} {источник_адрес}  
{маска_подсети_источника} {назначение_адрес} {маска_подсети_назначения}
```

Например:

```
access-list 101 permit tcp 192.168.1.0 0.0.0.255 10.0.0.0 0.255.255.255
```

```
access-list 100 deny tcp host 192.168.1.3 host 192.168.3.2 eq 80
```

## Лекция 16 (20.12)

### Прикладной уровень

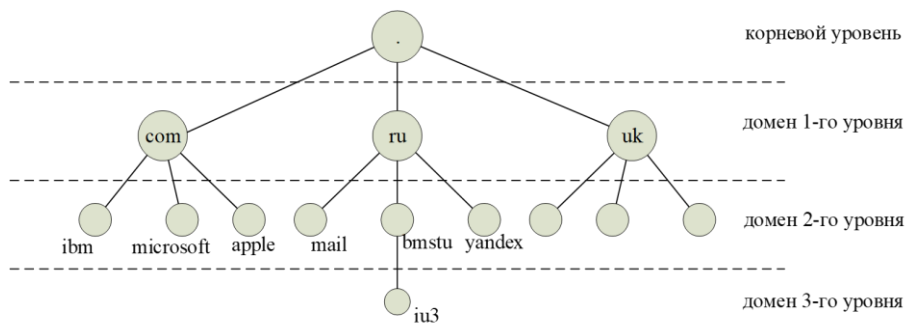
**DNS** - *Domain Name Server*

Относится к семейству TCP/IP

Протокол прикладного уровня, работающий на **53** порту.

Позволяет браузерам и иным клиентам разрешать доменное имя в IP-адрес, (еще балансирует нагрузку между разными сервисами на один IP-адрес).

Имеет иерархическую структуру, где каждый домен (уровень) знает только на один уровень вниз (нижестоящие, подключенные к нему).



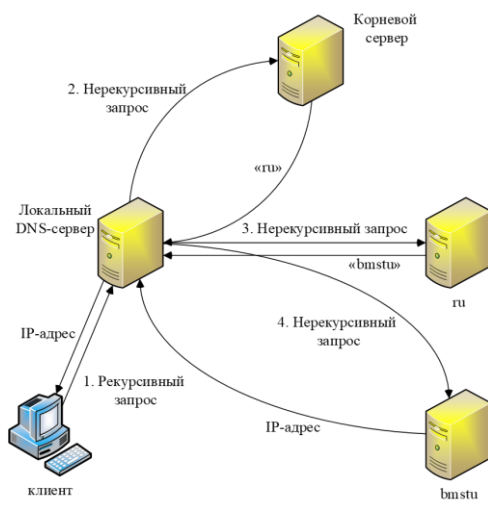
DNS – иерархия, дерево. По домену в каждой стране, по домену в каждой организации. Корень – сервер в мире (раньше их количество было равно 13).

### Особенности DNS

- Распределенность администрирования – ответственность за разные части иерархической системы лежит на разных организациях (людях)
- Распределенность хранения информации – каждый узел (сервер) хранит только те данные, которые входят в его зону ответственности (на 1 уровень вниз)
- Кэширование – узлу разрешено определенное время хранить информацию не из своей зоны своей ответственности разрешенное количество времени (хозяин записи определяет время хранения ее в кэше)
- Резервирование – на случай выхода из строя первичного сервера есть резервный сервер
- Иерархическая структура – узел может самостоятельно решить проблему или делегировать задачу вышестоящему узлу

### Алгоритмы поведения

- **Рекурсивный** - локальный DNS-сервер опрашивает сервер следующего уровня, тот следующего и так далее; на клиент возвращается готовый ответ
- **Нерекурсивный** - каждый узел, сообщает, что не может ответить на запрос и советует место, в котором нужно искать. Клиент отправляет запрос другому серверу... Так повторяется, пока какой-либо из серверов не даст ответ.



### Состав DNS-записи

**name** – доменное имя, к которому принадлежит данная ресурсная запись;

**ttl** – допустимое время хранения данной ресурсной записи в кэше неответственного DNS-сервера;

**type** – тип (формат) (назначение данной ресурсной записи):

запись A (address record), связывающая доменное имя устройства и IPv4-адреса;

запись AAAA, связывающая доменное имя устройства и IP-адрес версии 6;

запись CNAME (canonical name record), предоставляющая **каноническое имя** для перенаправления на другое доменное имя;

запись NS (name server), указывающая на ответственный DNS-сервер.

**class** – тип сети, в которой применяется данная ресурсная запись (по умолчанию TCP/IP);

**rdlen** – (resource data length) длина поля «данные»;

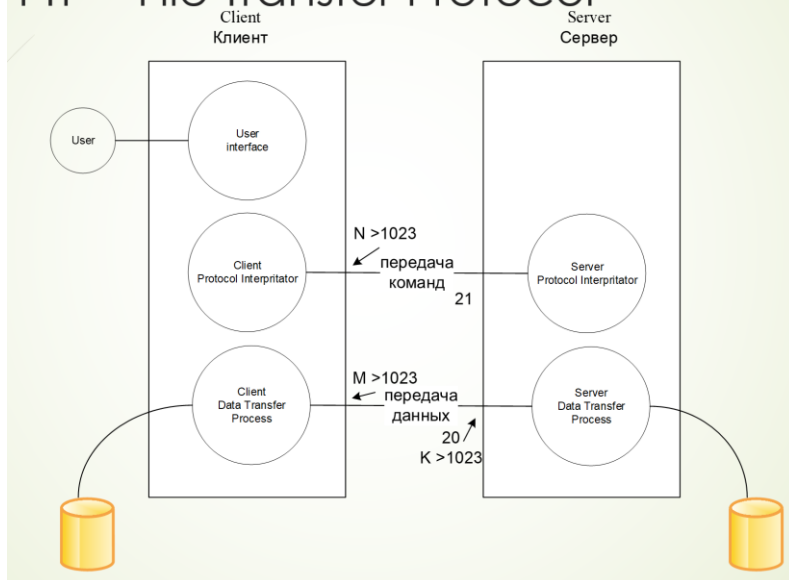
**rdata** – поле «данные», формат и содержание которых определяет значение поля «тип» данной ресурсной записи.

**Commented [31]:** Каноническое имя (запись CNAME) – это тип записи DNS, которая привязывает псевдоним к действительному (каноническому) доменному имени.

### FTP - File Transfer Protocol

Протокол прикладного уровня, для передачи файлов (на сервер, с сервера). Работает поверх TCP (сразу двух TCP соединений) на **21** и **20** портах

# FTP – File Transfer Protocol



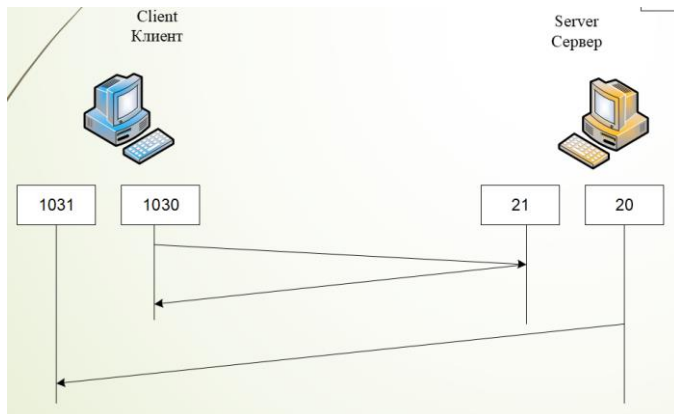
- 1) **PI (Protocol Interpreter)** – для передачи команд (TCP:21). Работает все время пока мы общаемся (21 порт).
- 2) **DTP (Data Transfer Process)** – для передачи данных (TCP:20). Открывается на одну транзакцию. После передачи закрывается. На сервере либо 20-й порт, либо высокий (зависит от режима работы сервера), на клиенте – высокий (>1023).

**Соединение для передачи команд всегда открывает клиент**

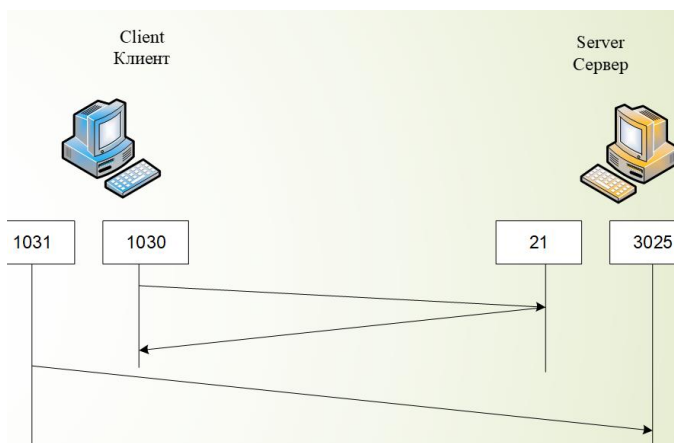
**Режимы работы сервера:**

1. Активный - у сервера DTP на 20-м порту, и он открывает соединение для передачи данных (отправляет флаг SYN=1)





2. Пассивный - клиент открывает соединение (отправляет флаг SYN=1) и для передачи команд, и для передачи данных



Проблема: брандмауэр блокирует высокие порты, и их нужно программно открывать, изменяя настройки брандмауэра, ведь порты по умолчанию заблокированы.

### Команды FTP

#### 1) Управление доступом (нет шифрования):

- user, pass – отправляем логин, пароль;
- cwd – смена рабочей директории;
- rein – реинициализация (откат настроек до заводских);
- quit – закрытие соединения для передачи команд.

#### 2) Управление потоком (управление параметрами соединения для передачи данных):

формат - h1,h2,h3,h4,p1,p2 (h – IP, p – порт)

пример: 192,168,1,1,4,1; номер порта – 1025 (00000100,00000001 = 4,1)

- pasv – перевод сервера в пассивный режим;
- port – перевод сервера в активный режим (указывает на активного участника соединения).

3) **FTP-сервис** (для работы с папками и файлами с помощью этих команд):

- retr zzz.txt – копирует файл с сервера на клиента;
- store zzz.txt – заливает файл с клиента на сервер;
- rnfr z1.txt – переименовывает «из»;
- rnfo z2.txt – переименовывает «в» ;
- dele zzz.txt – удаляет файл;
- list – содержимое директории;
- mkd name\_dir – создает папку.
- rmd name\_dir – удаляет папку.

На заметку: ftp не дает удалять директорию, где есть хоть один файл.

\list - посмотреть, что есть в директории

## SMTP - Simple Mail Transfer Protocol

Протокол прикладного уровня для отправки почты, поверх **TCP**, по умолчанию – **25** порт, из-за шифрования порт может меняться.

Назначение - отправка почты

**Принцип работы:** источник отправляет на сервер, сервер отправляет на сервер входящей почты получателя (а как забирать с этого сервера - уже работа другого протокола) -> источник и получатель должны быть онлайн для общения.

слева - как мы общаемся с сервером

справа - как сервер с нами общается

Как мы с ним общаемся	Как он с нами общается
<ul style="list-style-type: none"> <li>■ Установка TCP-соединения</li> <li>■ <code>helo bmsu.ru</code> (вход)</li> <li>■ <code>auth login</code> (авторизация) <ul style="list-style-type: none"> <li>■ ввод логина (зашифровано по алгоритму base64 поэтому еще обычно сверху шифруется)</li> <li>■ ввод пароля (зашифровано)</li> </ul> </li> <li>■ <code>mail from zzz@bmsu.ru</code></li> <li>■ <code>rcpt to yyy@bmsu.ru</code></li> <li>■ Data (ввод данных) <ul style="list-style-type: none"> <li>■ <code>blablabla.</code></li> <li>■ <code>.</code></li> </ul> </li> <li>■ <code>quit</code></li> </ul>	<ul style="list-style-type: none"> <li>■ <code>2__</code> - все хорошо</li> <li>■ <code>3__</code> - неоконченное действие</li> <li>■ <code>5__</code> - неуспешно</li> </ul> <p>Примеры:</p> <ul style="list-style-type: none"> <li>■ <code>250 OK</code></li> <li>■ <code>354 Start mail input, end with &lt;CRLF&gt;. &lt;CRLF&gt;</code></li> <li>■ <code>550 No such user here</code></li> </ul>

// в helo нет опечатки, это реальная команда

- 3.. - неоконченное действие (например, пароль введен не до конца)
- 5.. - ошибка (опечатка в команде, ошибка авторизации)

## **POP3 - Post Office Protocol**

Работает на порту **TCP:110 клиента**

Назначение – **получение** почты

Ответы сервера:

- + OK
- ERR
- + текст сообщения, что ему не нравится

Режимы работы:

- 1) Authorization - ввод почты и пароля (без шифрования), команды:
  - user – открытым текстом
  - pass – открытым текстом
  - quit
- 2) Transaction, команды:
  - stat – кол-во сообщений и общий объем в символах
  - list [n] – информация о сообщениях
  - retr [n] – содержимое сообщения
  - dele [n] – отмечает, как «удаленное»
  - rset – снимает пометки «удаленное»
- 3) Update - освобождает все ресурсы и разрывает соединение

## **IMAP4 - Internet Message Access Protocol**

Протокол прикладного уровня, поверх **TCP:143 на сервере**

**Получение** почты. Работает с почтой на сервере (а POP3 – на клиенте).

Используется, когда есть стабильный интернет, есть несколько устройств.

Разница: POP3 работает на клиенте а IMAP4 на сервере. Рассмотрим пример: есть два компа (один дома, другой на работе), оба с почтовиками и оба настроены на POP3.

Синхронизации между ними не будет. Все что сделали на работе – останется там.

А если IMAP4, то всё синхронизируется - то, что сделали на работе, дома тоже отобразится.

Преимущество POP3 в том, что он работает на клиенте. Когда он появился, с сетью были реальные проблемы. Достаточно на немного получить доступ в сеть, чтобы всё перекинулось к нам и мы могли с этим работать

IMAP4 не даст прочесть письмо если нет соединения, а мы во время наличия соединения не посмотрели письмо, потому что оно хранится на сервере

## Telnet

Протокол прикладного уровня, поверх **TCP:23**

Предназначение: управление удаленным устройством

### Режимы передачи:

- 1) Полудуплексный  
По факту не функционирует. Смысл в том что мы печатаем команду и видим на мониторе но отправляем только по команде от другого устройства
- 2) Символьный  
Смысл в том, что мы печатаем команду и каждый символ отправляется на устройства, там применяется, отправляется нам назад и мы видим его на экране. Соответственно каждую букву нужно упаковать в 4 уровня и отправить. Нельзя стереть ничего
- 3) Условно-строчный  
напечатанное отправляется по нажатию enter, отсутствует echo-печать (не видно, что напечатали), удобно использовать для ввода паролей
- 4) Строчный  
echo-печать от клиента, можем стирать и отправлять целиком по нажатию enter

В жизни он хорош для обучения и немного для траблшутинга в локалке (в жизни для удаленного доступа используется SSH)

Telnet - также клиент (можно подключаться в определенному протоколу)

```
C:\> telnet 192.168.1.1
```

заходим на устройство с указанным IP удаленно

```
C:\> telnet 192.168.1.1 25
```

заходим и получаем функционал SMTP, если на устройстве с этим ip запущен smtp-сервер

```
C:\> telnet 192.168.1.1 21
```

заходим и получаем функционал FTP, в частности Protocol interpreter FTP, т.к. 21-й порт

## HTTP - Hyper Text Transfer Protocol

(версии 1.0 / 1.1) работает на **TCP:80**

Назначение: **представление файлов, хранящихся на сервере**, в определенном формате

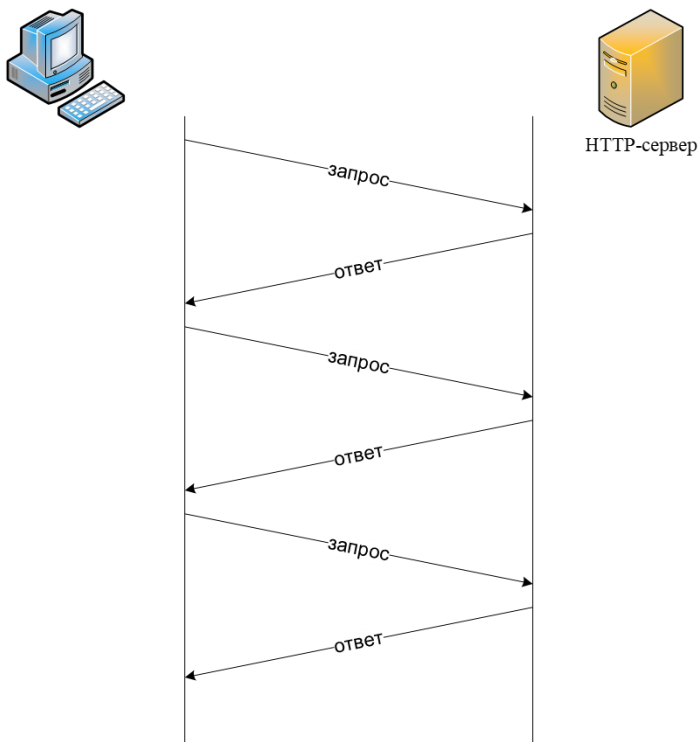
Между клиентом и сервером могут быть промежуточные устройства - например,

- 1) Proxy ("главные выходные ворота") - обладает следующим функционалом: взятие результата из кэша, с целью не пустить запрос дальше
- 2) Gateway (не пускают злоумышленника в нашу сеть, за это отвечает PAT). Эти 2 штуки вместе - следующий тип
- 3) Firewall/Tunnel

Существует чтобы когда мы передавали данные через устройства которые не могут, корректно интерпретировать наши данные, был тоннель

Типы соединений:

- 1) Непостоянное соединение 1.0, 1.1 - идет открытие 1-го TCP соединения, для каждого объекта, полученного из соединения начинается свое скачивание, а значит для каждого открывается свое TCP-соединения -> нерациональное использование ресурсов
- 2) Постоянное 1.1 - все скачивается в рамках одной сессии
  - а) Без конвейеризации (клиент ждет ответа на запрос перед отправкой нового запроса) и
  - б) С конвейеризации (не ждет получение первого, кидает запросы на следующие -> ускорение)



GET-запрос

страница, сервер, тип подключения, браузер

Ответ

версия протокола, и все оставшееся

Стандарт MIME используется для стандартного отображения

Номер порта	Протокол	Приложение
20	TCP	FTP data
21	TCP	FTP control
22	TCP	SSH
23	TCP	Telnet
25	TCP	SMTP
53	UDP, TCP	DNS
67	UDP	DHCP Server
68	UDP	DHCP Client
69	UDP	TFTP
80	TCP	HTTP (WWW)
110	TCP	POP3
161	UDP	SNMP
443	TCP	SSL
514	UDP	Syslog

Commented [32]: 143 - TCP - IMAP4