1. Базовая модель взаимодействия сетевых приложений. Понятия сервиса, интерфейса и протокола.

**Сервис** - это услуги, которые нижерасположенный уровень оказывает по запросам вышерасположенного. В этой модели нижерасположенный уровень свои услуги может предоставлять только вышерасположенному уровню.

**Интерфейс** - набор операций и структур данных, с помощью которых уровень предоставляет доступ к своим операциям вышележащему уровню.

**Протокол** - правила и соглашения по установлению соединения, его поддержанию и  обмену данными по нему между активностями, расположенными на одинаковом уровне на разных машинах.

2. Виды задержек передачи данных при пакетной коммутации и их свойства.

**Пакет** – единица данных, несущая достаточно информации, чтобы быть доставленной к месту назначения

**Коммутация пакетов**: для каждого поступающего пакета независимо выбирается канал для отправки. Если канал свободен, то пакет отправляют, если нет – буферизуют пока канал не освободится.

Идея коммутации пакетов состоит в том, чтобы разбить массив данных на порции, и каждую порция (пакет) передавать независимо от других порций. Каждый пакет должен нести достаточно информации для его маршрутизации. Каждый пакет маршрутизируется независимо от других.

Когда мы отправляем пакет, его передают hop\_by\_hop от коммутатора к коммутатору. Каждый пакетный коммутатор смотрит заголовок и определяет по своей локальной таблице линию, по которой надо передать пакет.

**Производительность** – количество PDU, которое сеть способна передать между двумя абонентскими машинами. В ее формировании ключевую роль играют такие величины как задержка распространения и задержка пакетизации, которые нам понадобятся для определения сквозной задержки. Чем больше задержка тем меньше производительность.

* **Задержка распространения**

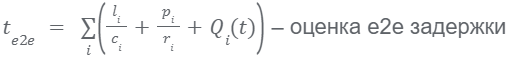
Время распространения одного бита по каналу длины l со скоростью c

* 
* эта задержка есть всегда, она связана с тем, что есть конечная скорость распространения фронта волны. Т.е. электромагнитная волна посредством которой передаем бит через носитель имеет конечную скорость распространения c.
* c зависит от среды, в которой передается сигнал, то есть от физических свойств канала
* не зависит от пропускной способности канала
* **Задержка пакетизации**

Время, за которое все биты пакета с первого до последнего переданы в канал (от момента поступления 1-ого бита на линию до момента поступления последнего)

* 
* задержка связана с тем что есть конечная скорость передачи каждого бита r, то есть пакет передается не мгновенно
* скорость передачи = пропускная способность – определяется сетевым интерфейсом
* этот вид задержки определяет то, как быстро биты помещают в линию
* это функция от длины пакета и скорости адаптера, при условии, что пропускная способность линии соответствует скорости адаптера
* **Задержка буферизации**:
  + Q(t)= A(t) - D(t)
  + Эта задержка переменная и может меняться внезапно, поэтому ее сложно оценить. Оценки в основном строятся на основе ее среднего значения
  + Причина: буферизация, т.е. пакеты находятся в очереди
* **Сквозная задержка (e2e)**

Это время от момента когда первый бит пакета попадает на линию до момента когда последний бит пакета поступает к получателю

* равна сумме по всем линиям маршрута (задержка распространения + задержка пакетизации + задержка буферизации)
* ****

3. Устройство пакетного коммутатора. Методы оценки сквозной задержки при пакетной коммутации. Метод справедливой взвешенной очереди и его свойства, условия его применимости.

**Пакетный коммутатор** работает на канальном уровне. Содержит  в себе хеш-таблицу, в которой каждому MAC-адресу сопоставляется выходной интерфейс.

Общий алгоритм работы коммутатора:

 1. Lookup Address – анализ адреса-получателя, чтобы понять на какой порт необходимо посылать пакет. Выполняется с помощью поиска соответствия MAC-адреса и порта в таблице коммутации

 2. Update Header - если потребуется, то может быть произведено изменение заголовка

 3. Queue Packet - постановка пакета в очередь и буферизация, если это необходимо

Формирование таблицы:

● На коммутатор поступает пакет

● Анализируется MAC-адрес отправителя, и порт, с которого пришел этот пакет заносится в таблицу.

● Анализируется MAC-адрес получателя (Destination Address - DA)

Если адрес DA есть в таблице коммутации, то кадр передают на надлежащий  выходной порт

Если адрес DA нет в таблице, кадр рассылается по всем портам, кроме того на который пришел.

Когда придет ответ на разосланный пакет, то по его адресу отправителя и номеру порта, с которого он пришел, мы узнаем , куда надо направлять пакеты с такими адресами получателей

**Оценка сквозной задержки при пакетной коммутации:**

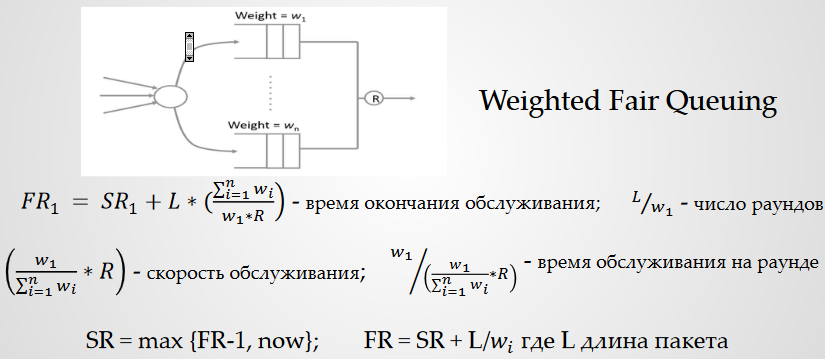
* задержка распространения – время распространения одного бита по каналу длины l со скоростью c
  + 
  + не зависит от пропускной способности канала
* задержка пакетизации – время, за которое все биты пакета с первого до последнего переданы в канал
  + 
  + скорость передачи = пропускная способность – определяется сетевым интерфейсом
* задержка буферизации:
  + Q(t)= A(t) - D(t)
  + Причина: буферизация, т.е. пакеты находятся в очереди
* сквозная задержка (e2e) – время от момента когда первый бит пакета попадает на линию до момента когда последний бит пакета поступает к получателю
  + сумма задержек(задержка распространения + задержка пакетизации + задержка буферизации)
  + 

**Взвешенная справедливая очередь** - механизм планирования пакетных потоков данных с различными приоритетами. Регулирование использования одного канала несколькими конкурирующими потоками.

Алгоритм обслуживания:

У нас есть некоторая магическая очередь, состоящая из n очередей. За один проход мы берем из каждой очереди число бит пропорциональное весу потока, пока не дойдем до маркера конец пакета. Как только мы перешлем последний бит пакета, этот пакет магически будет собран в магической очереди из находящихся там бит и отправлен по линии.

Условия применимости:

* пакеты не должны сбрасываться
* дисциплина обслуживание очереди – FIFO

Свойства:

* Время окончания обработки может быть

определено в момент поступления пакета

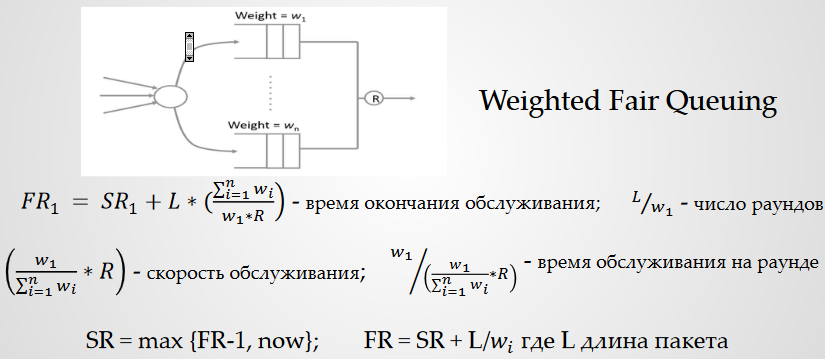
* Пакеты обрабатываются в соответствии

с весами один за другим

4. Коммутация пакетов: методы управления задержкой пакета в сети.

Если поток ограничен текущим буфером и маршрутизатор использует

дисциплину обслуживания WFQ, то можно гарантировать е2е задержку:



**WFQ**(Взвешенная справедливая очередь)(в 5 билете подробнее) позволяет каждому потоку обеспечить гарантированный сервис, планируя их в порядке bit-by-bit finishing time в предположении:

* Дисциплина обслуживания очереди – FIFO
* Нет потери пакетов

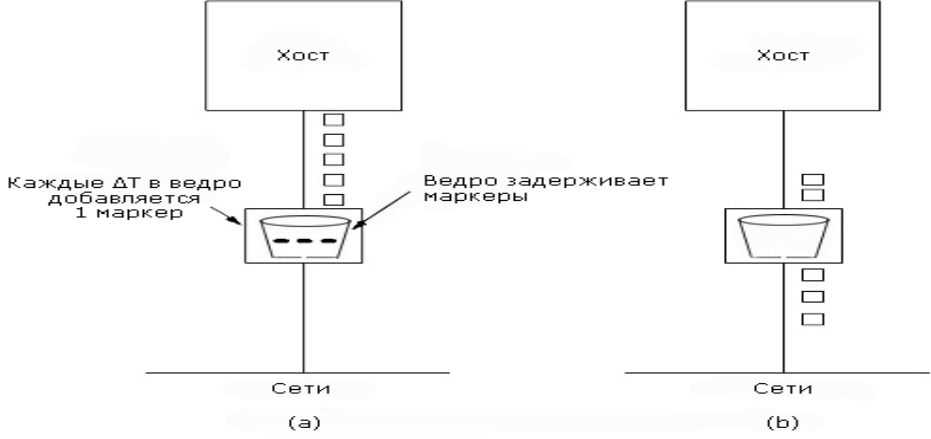
Поскольку мы не можем управлять процессом поступления, его можно ограничить. Пусть число бит, которые могут поступить за период t ограничены q + pt .Где, например, q=B , p=R1 – размер канала.

Тогда мы не столкнемся с проблемой переполнения буфера (для конкретного случая).

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

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

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



Несмотря на то, что технически это возможно, лишь некоторые сети могут управлять е2е задержкой.

Причины:

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

5. Устройство пакетного коммутатора. Виды буферизации в пакетных коммутаторах. Различия в работе коммутатора и маршрутизатора

**Пакетный коммутатор** работает на канальном уровне. Содержит  в себе хеш-таблицу, в которой каждому MAC-адресу сопоставляется выходной интерфейс.

Общий алгоритм работы коммутатора:

 1. Lookup Address – анализ адреса-получателя, чтобы понять на какой порт необходимо посылать пакет. Выполняется с помощью поиска соответствия MAC-адреса и порта в таблице коммутации

 2. Update Header - если потребуется, то может быть произведено изменение заголовка

 3. Queue Packet - постановка пакета в очередь и буферизация, если это необходимо

Формирование таблицы:

● На коммутатор поступает пакет

● Анализируется MAC-адрес отправителя, и порт, с которого пришел этот пакет заносится в таблицу.

● Анализируется MAC-адрес получателя (Destination Address - DA)

Если адрес DA есть в таблице коммутации, то кадр передают на надлежащий  выходной порт

Если адрес DA нет в таблице, кадр рассылается по всем портам, кроме того на который пришел.

Когда придет ответ на разосланный пакет, то по его адресу отправителя и номеру порта, с которого он пришел, мы узнаем , куда надо направлять пакеты с такими адресами получателей

**Буферизация на выходе.**

Этот тип буферизации используется в выходных портах - для каждого отдельного выходного порта создается свой буфер. Пакет обрабатывается и направляется в очередь на соответствующий порт в ожидании отправки. Работа с памятью при такой буферизации должна быть очень быстрой.

**Буферизация на входе.**

При буферизации на входе буферы организуются на входных портах и пакеты ожидают на входе, пока их порты не освободятся. Такой механизм приводит к блокировке на входе. Когда приходят несколько первых пакетов на разных входах, которые должны быть отправлены через один интерфейс, то они могут блокировать передачу пакетов следующих за первыми. Решением этой проблемы может служить **механизм виртуальной очереди на выходе**. Когда в каждом буфере создаются очереди для каждого выходного интерфейса. В таком случае блокировок не возникает и линии используются оптимально

**Интернет маршрутизатор**

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

**По сути, Ethernet коммутаторы и маршрутизаторы выполняют одинаковые действия, но поиск адреса происходит по разному:**

Коммутатор(Ethernet):

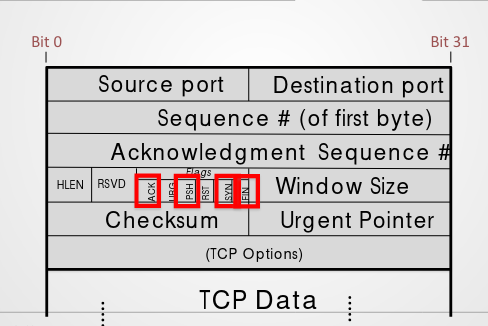
* Адреса хранятся в хэш-таблице.
* Ищем в хэш-таблице точное совпадение.

Маршрутизатор(IP):

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

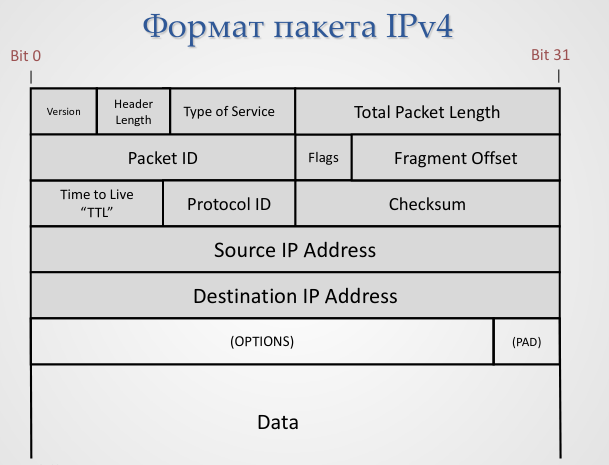
6. Заголовки IP, TCP. Фрагментация PDU и управление ею. Методы управления потоком

**TCP заголовок:**



Заголовок состоит из номера порта отправителя, номера порта получателя, порядковый номер первого байта в поле данных, номер подтверждения, длина TCP заголовка, целый ряд флагов, размер окна (размер буфера), контрольная сумма (средство обнаружения ошибок при передаче пакета) + всякие опции, которые позволяют расширить заголовок. TCP соединение устанавливается двунаправленно (от А к Б и обратно).

**IP заголовок:**



Заголовок имеет переменную длину (от 20 байт до 60)

Состоит из: версия протокола (IPv4 или IPv6), длина заголовка, тип сервиса (передается ли голос, видео или просто данные), общая длина IP пакета, уникальный идентификатор ID у каждого пакета (задается если надо фрагментировать IP пакет, т.е. разбить его на более мелкие протокольные единицы данных), смещение соответствующего фрагмента в общем пакете, набор флагов, TTL (время жизни пакета), транспортный протокол(TCP, UDP), контрольная сумма (ТОЛЬКО для заголовка, чтобы дошло куда надо), IP адрес отправителя и получателя + разные опции (проложить весь маршрут для передачи пакета, указать маршрутизаторы через которые должен пройти маршрут, каждый маршрутизатор оставляет свой ID, таким образом можно узнать маршрут, временная метка, чтобы узнать, где блуждал пакет подозрительно долго (борьба с хакерством))

Фрагментация PDU и управление ею:

Транспорт: TCP получает поток байтов и разделяет его на TCP сегменты

Сетевой: IP пакет может оказаться слишком коротким для TCP сегмента

Канальный: IP пакет длиннее кадра (Ethernet кадр = 1,5кб IP packet = 64кб)

Следует избегать IP фрагментацию всегда, когда можно, потому что при ip фрагментации пакеты не будут собираться на следующем хопе и будут существовать как отдельные единицы, тем самым нагружая сеть, так как у каждого фрагмента свой заголовок.

В случае TCP можно выбрать размер сегмента так чтобы избежать фрагментации

чтобы определить MTU (maximum transmission unit) используется DF флаг и оценивается получение ICMP сообщения

MTU -  максимальный размер блока данных, который может быть передан протоколом без фрагментации.

В компьютерных сетях неизбежны потери пакетов данных, в частности, из-за переполнения буферной памяти хотя бы одного из узлов, расположенных на пути от источника к приемнику, включая последний.

Такие потери, связанные с переполнениями, в дальнейшем именуются **перегрузками** узлов сети. Существует множество способов предотвращения и устранения перегрузок; эти способы, в большинстве своем, основаны на **управлении потоками данных**. Особое место занимает обслуживание пакетов с учетом их приоритетов.

**Способ 1** Управление потоком данных регулировкой длительности пауз между пакетами(Stop and Wait).

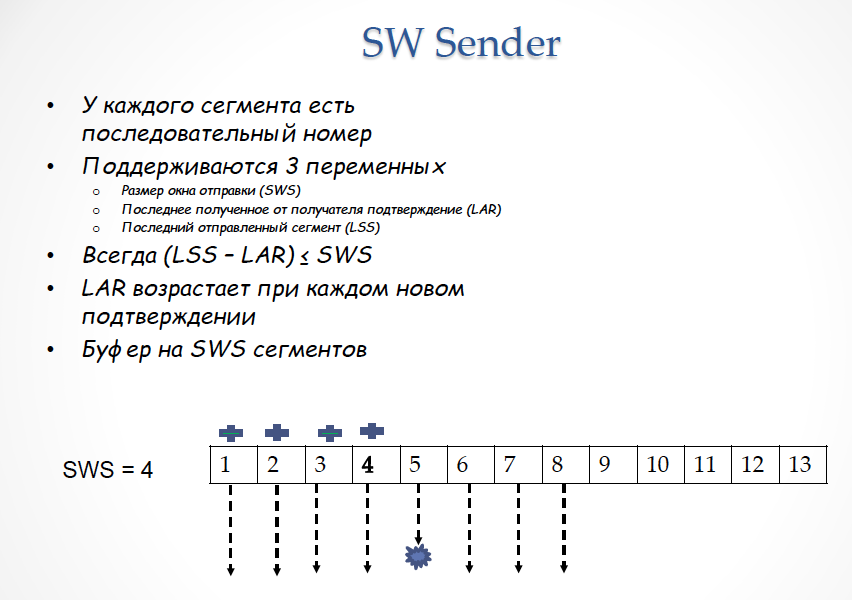
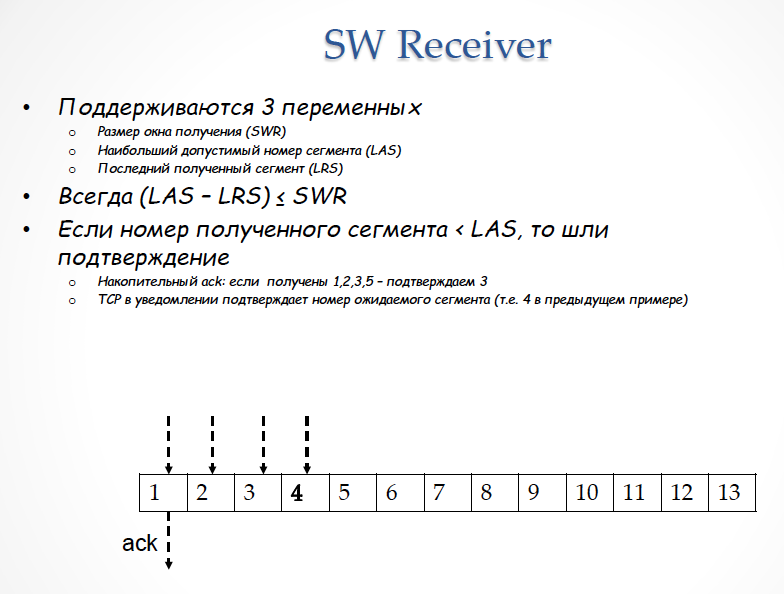
Отправляем данные, ждем ack, отправляем след данные, если timeout, то еще раз посылаем те же данные, получатель получается только шлет ack и все.

Недостаток: медленно работает, так как отправляем по одному пакету.

если получатель медленно отправляет подтверждение могут возникать дубли данных в сети (подтверждение приходит после тайм аута).

**Способ 2** Схема скользящего окна.

Отправитель не может отправлять данных больше чем указано в размер окна пока не получит ответ об их получении отправителем. В отличие от stop and wait отправитель не ждет подтверждение на каждый пакет а отправляет сразу несколько - суммарным размером в размер окна или меньше, при поступлении очередного подтверждения отправитель отмечает что пакет отправлен и подтвержден и может передавать следующие.(размер окна выбирается min{WSS, WSD} из 2 хостов)

****

7. Явление перегрузки, причины ее возникновения и основные методы борьбы с ней.

**Перегрузка** — это явление, при котором из-за нерегулярности потоков данных нагрузка на выходных линиях коммутатора начинает превосходить пропускную способность этих линий.

**Управление перегрузками** — это процесс регулирования потоков данных в транспортной среде, при котором эти потоки не превышают ее пропускной способности. Эта глобальная проблема затрагивает поведение всех абонентских машин и всех маршрутизаторов в сети.

Причины возникновения перегрузки:

* Источники ничего не знают о распределении пропускных способностей каналов - суммарная скорость поступления пакетов в маршрутизатор превышает скорость выходной линии
* Слишком много пользователей используют один и тот же канал в одно и то же время
* Столкновение двух пакетов в маршрутизаторе - на один и тот же порт поступает несколько пакетов

Причина перегрузки – несбалансированность пропускных способностей каналов в сравнении со скоростью передаваемых данных

Почему перегрузки неизбежны?

* Коммутацию пакетов используют потому, что она позволяет эффективно использовать пропускную способность каналов. Поэтому буферы в маршрутизаторах часто заполнены.
* Если буферы не заполнены, то задержки малы, но интенсивность использования сети низкая.
* Если буферы постоянно заполнены, задержки возрастают, но интенсивность использования сети также возрастает

**Методы, предотвращающие перегрузки.**

Методы ориентированы на минимизацию перегрузок при первых признаках их проявлений, а не на борьбу с перегрузками, когда они уже случились.

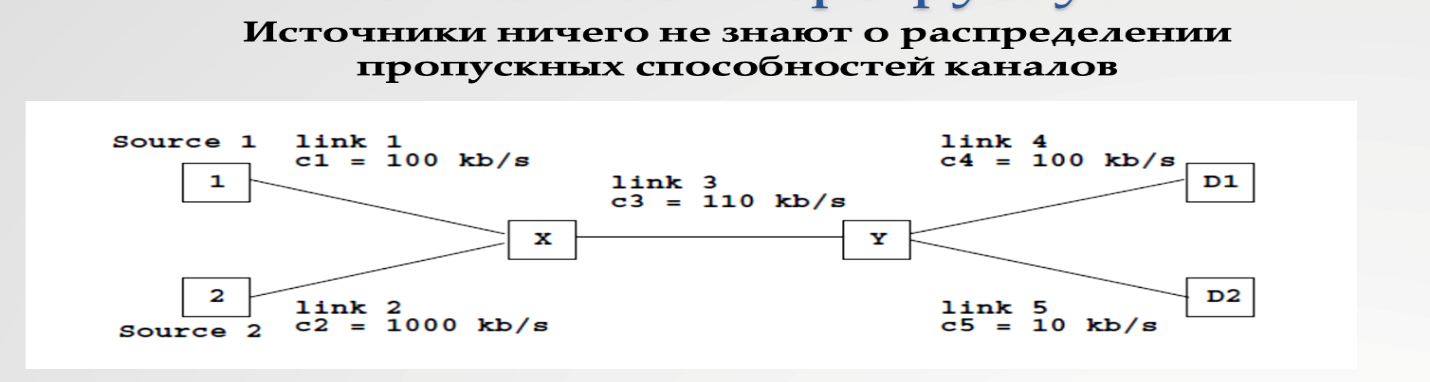
Существуют следующие методы:

* Сдерживание нарастания трафика посредством управления потоком (скользящее окно)
* Корректная организация очередей на коммутаторах и маршрутизаторах
* Выбор метода сброса пакетов
* Регулирование времени жизни пакета в сети

Для распределения пропускной способности перегруженного канала необходимо ввести понятие справедливости, чтобы решить как потоки будут разделять ресурсы этого канала.

Определение:

**Распределение max-min** справедливо если нельзя увеличить скорость какого-нибудь потока, не снизив скорости другого потока с меньшей скоростью.

**Математические модели возникновения перегрузки: **

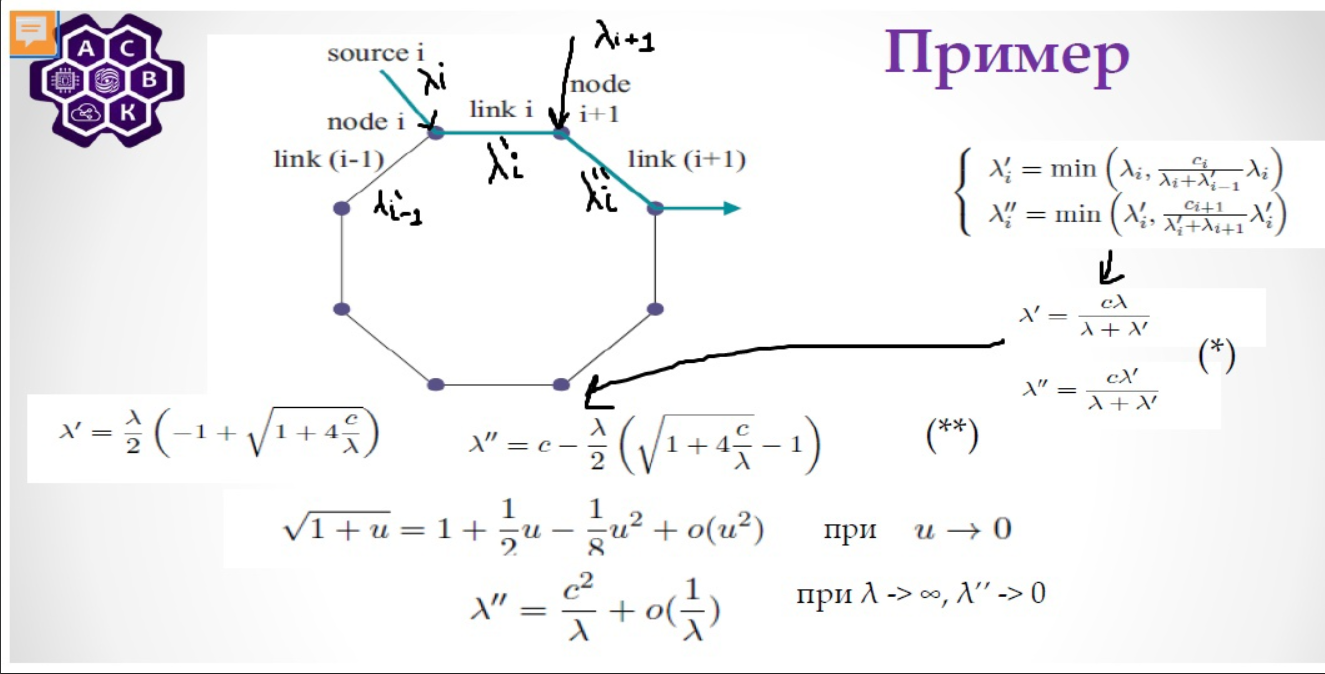
1.  Пусть надо связать s1 – d1, s2 – d2. Предполагаем, что у источников s1 и s2 нет обратной связи с сетью. Тогда s1 будет слать пакеты на скорости с1, а s2 – c2. Но на линии l3 эти потоки не пройдут.

Пакеты от s2 будут поступать чаще и они займут большую часть пропускной способности l3. Поэтому скорее всего распределение потоков будет 10 для s1 : 100 для s2, т.е. из каждой тысячи пакетов от s2 пройдет только 100.

Однако после Y от этих 100 пройдет только 10. И от s1 пройдет до d1 только 10. Итого 20 пакетов Если бы s2 «знал», что более 10 пакетов в секунду от него не пройдет, то он и не стал бы слать.

В этом случае s1 смог бы передать все свои 100 пакетов в секунду. Итого: 110 пакетов.

2.



Эта математическая модель показывает, что если источник будет гнать пакеты в сеть без какой-либо обратной связи, то сеть очень быстро заткнется. На этом примере мы работаем с асимптотикой. Мы смотрим, что будет происходить при устремлении соответствующих параметров к бесконечности и видим, что сеть перестает работать.

Топология представляет собой кольцо. Есть узлы и линки, пронумерованные 0, 1, …, l-1. Источник i передаёт данные в узел i+2, через узел i+1, используя линии [(i+1) mod l] и [(i+2) mod l], и покидает сеть в узле i+2 mod l.

Обозначим ci - пропускную способность линии i. Предположим, что источник i отправляет λi пакетов в секунду. Назовем λi’ скоростью, достигнутой источником i для линии [(i+1) mod l], а λi’’ - скорость для источника i, достигнутая на линии  [(i+2) mod l]. Это соответствует для каждого источника выбору кратчайшего пути к месту назначения. В оставшейся части этого примера мы опускаем “mod l”, когда контекст понятен.

Разрешаем λ’ и λ’’ по λ из (\*) получаем (\*\*), откуда с учетом разложения 1+u получаем результат.

**Вывод:** в сети с коммутацией пакетов источник должен регулировать свою скорость вброса пакетов в сеть в зависимости от состояния сети. В противном случае наступит перезагрузка.

8. Алгоритмы управления перегрузкой: AIMD в случае одного потока и в случае нескольких потоков. Способы обнаружения перегрузки и виды алгоритмов управления перегрузкой.

*1 Поток:*

Если мы увеличиваем число байт в сети, то вероятность перегрузки растет. Если перегрузка произошла, то надо сокращать число байт в сети, т.е. размер окна. Таким образом регулируя количество байт в сети мы можем влиять на перегрузки. И что самое интересное это все происходит на конечном хосте без помощи сети.

 • Если пакет получен успешно: 𝑊 ← 𝑊 + 1/𝑊

 • Если пакет был сброшен: 𝑊 ← W/2

Для оценки размера окна перегрузки (cwnd) используют алгоритм AIMD.

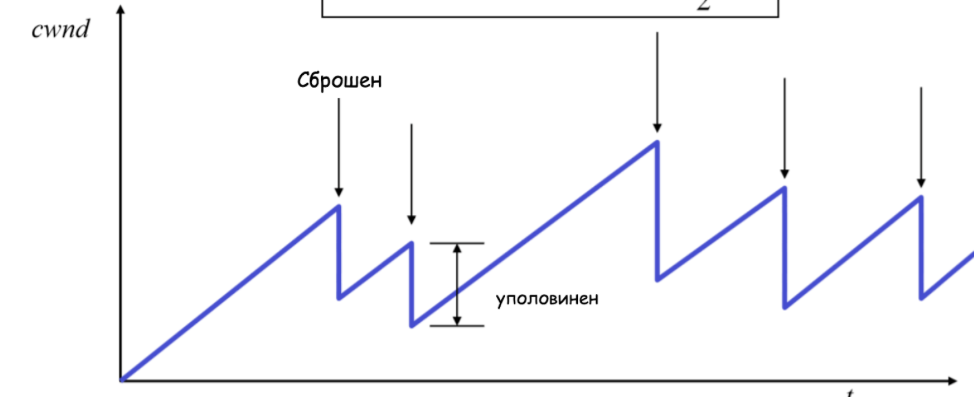
Это классический прием для управления перегрузками в сетях. Его идея в том, чтобы как можно быстрее нащупать предел имеющегося ресурса, после чего прекратить наращивать скорость выброса пакетов в сеть.

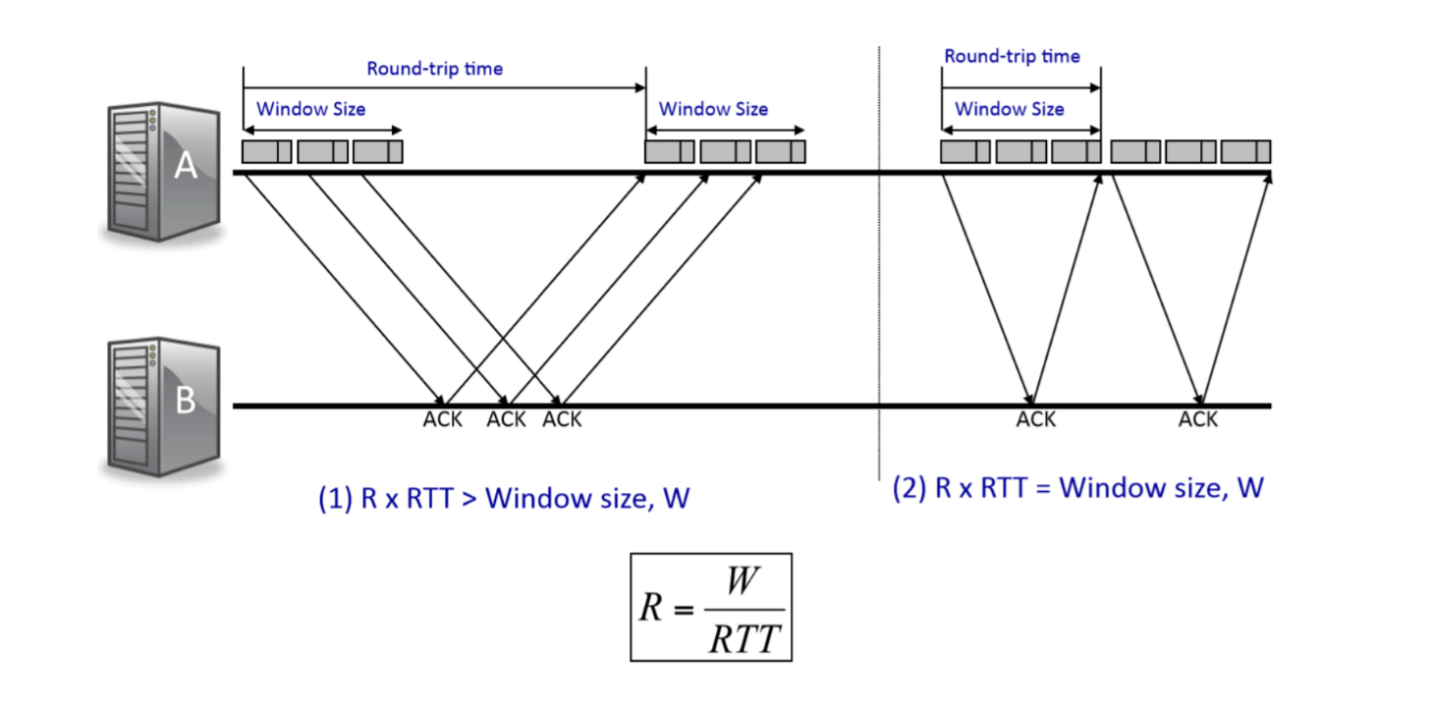
Начнем с аддитивного увеличения. Каждый раз когда пакет был получен корректно (АСК), отправитель увеличивает окно на 1/W, где W – текущий размер окна перегрузки. Увеличиваем окно на 1/W для каждого пакета, а пакетов в окне - W, то по исчерпании окна, его размер будет увеличен на 1. Это медленное увеличение размера окна, если все идет хорошо. Если же есть осложнения, если пакет был утерян, что воспринимается как признак перегрузки, то размер окна резко сокращают вдвое.

Итак, окно растет пока все идет хорошо, как только пакет утерян, сокращаем окно вдвое. Рассмотрим подробнее что происходит на каждом зубце.

*(Указать на графике где additive increase где multiple decrease.)*

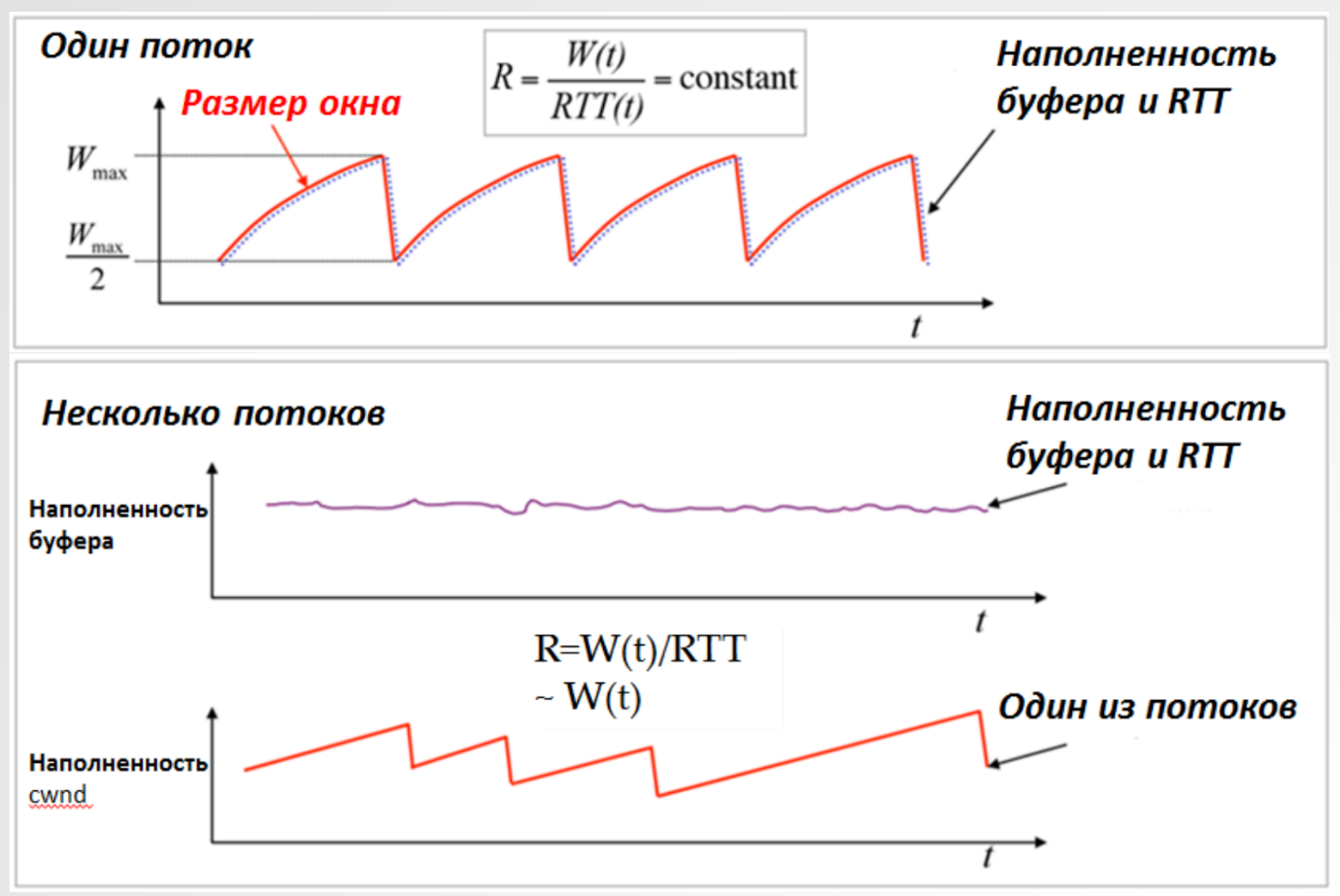
Фактически у нас не плавная монотонная прямая а ступеньки. Мы увеличиваем окно на 1 каждое RTT.





cwnd и RTT ведут себя одинаково потому что отношение R = W / RTT имеет постоянное значение. Это так потому что это самосинхронизирующийся алгоритм. Размер окна перегрузки и RTТ меняются синхронно.

* **Один поток vs много потоков**

****

Для одного потока пила плавно изгибается (для одного потока RTT растет с каждым увеличением W – поэтому кривая загибается), в случае множественных потоков RTT постоянна и получаем ровные ступенечки - прямая. Дело в том, что один поток не сразу заполняет буфер, а постепенно. Поэтому RTT постепенно увеличивается и кривая изгибается.

В случае множества потоков считаем, что буфер все время под нагрузкой (почти заполнен). Поэтому RTT постоянна и рост окна на 1 – линейная функция с фиксированным углом.

Уравнение показывает что при постоянном RTT скорость пропорциональна размеру окна, обратно пропорциональна RTT.

9. Управление передачей в ТСР: алгоритм управления перегрузкой Tahoe и алгоритм управления перегрузкой Reno – основные отличия.

Используемые значения:

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

Стратегия **TCP Tahoe** заключается в следующем:

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

**Медленный старт.**

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

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

**Предотвращение перегрузки**.

Когда CWND становится равным или превышает ssthresh, происходит переход с медленного старта на режим предотвращения перегрузки. В этом режиме значение CWND увеличивается на MSS2/𝐶𝑊𝑁𝐷 для каждого из подтвержденных о доставке 𝐶𝑊𝑁𝐷/𝑀𝑆𝑆 пакетов. Т.е. за каждый RTT окно перегрузки увеличивается на MSS. Получаем линейный рост для размера окна перегрузки. Смысл данного режима заключается в том, что значение окна перегрузки «удерживается» в области максимально возможных значений (где сеть загружена, но не происходит потеря пакетов).

В TCP Tahoe есть три типа сигналов:

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

**Повторные уведомления.**

Для отправителя существует быстрый способ узнать, что один из его пакетов потерян. По мере того как пакеты, следующие за потерянным пакетом, прибывают на приемник, они инициируют отправку подтверждений, которые приходят к отправителю. Все они имеют один и тот же номер подтверждения и называются повторными уведомлениями. Каждый раз, когда отправитель получает дубликат подтверждения, есть вероятность, что другой пакет уже пришел, а потерянный – нет. В TCP условно считается, что три повторных уведомления сигнализируют о потере пакета. Также по номеру подтверждения можно узнать, какой пакет утерян (это следующий по порядку пакет).

**При возникновении time\_out снова начинается медленный старт.**

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

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

10. Алгоритмы маршрутизации в Интернет: основные подходы, маршрутизация по вектору расстояния.

Основной задачей сетевого уровня является маршрутизация пакетов.

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

**Корректность** – свойство алгоритма маршрутизации, определяющее, что при любых обстоятельствах этот алгоритм либо найдет маршрут для доставки пакета адресату, либо выдаст сообщение о невозможности его доставки.

**Простота** – свойство, определяющее вычислительную сложность алгоритма маршрутизации.

**Устойчивость** – свойство алгоритма маршрутизации сохранять работоспособность независимо от каких-либо сбоев, или изменений топологии.

**Справедливость** – означает, что все пакеты независимо оттого, из какого канала они поступили, будут обслуживаться равномерно, для всех абонентов будет всегда выбираться оптимальный маршрут.

Алгоритмы маршрутизации можно разбить на два больших класса: адаптивные и неадаптивные.

**Неадаптивные алгоритмы** не принимают в расчет текущую загрузку сети и ее текущую топологию. Все возможные маршруты вычисляются заранее и загружаются в маршрутизаторы при загрузке сети. Такая маршрутизация называется статической.

**Адаптивные алгоритмы**, наоборот, определяют маршрут исходя из текущей загрузки и топологии транспортной среды. Адаптивные алгоритмы различаются способом получения информации, временем изменения маршрута и метрикой, используемой при оптимизации.

**Алгоритм маршрутизации по вектору расстояния** работает следующим образом: у каждого маршрутизатора в транспортной среде есть таблица расстояний до всех других маршрутизаторов, принадлежащих этой транспортной среде. Периодически каждый маршрутизатор обменивается этой информацией со своими соседями и обновляет информацию в своей таблице. Каждый элемент этой таблицы включает в себя два поля:

первое – номер канала, по которому следует отправлять пакеты, чтобы достичь нужного места

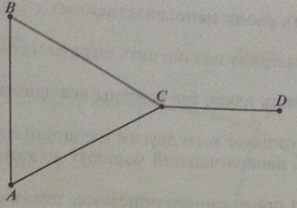
второе – значение задержки до места назначения, которое может измеряться в разных единицах

Каждые T секунд маршрутизатор шлет своим соседям свой вектор задержек до всех маршрутизаторов в транспортной среде. В свою очередь, он получает такие же векторы от своих соседей. Кроме того, он постоянно замеряет задержки до своих соседей, следовательно, имея векторы расстояний от соседей и зная расстояние до них, маршрутизатор всегда может вычислить наикратчайший маршрут до определенного места в транспортной среде.

**Проблема бесконечного счетчика задержки.**

Алгоритм маршрутизации по вектору расстояния теоретически работает хорошо, но у него есть один недостаток: он очень медленно реагирует на разрушения каналов в транспортной среде. Информация о появлении хорошего маршрута в транспортной среде распространяется более или менее быстро, а вот данные о потере, разрушении какого-то маршрута распространяются значительно медленнее.

**Разделение направлений**. Одним из решений проблемы бесконечного счетчика задержки является следующее. Алгоритм маршрутизации по вектору расстояния работает так, как было описано ранее, но при передаче вектора по линии, по которой направляются пакеты для маршрутизатора X, т.е. по которой достижим маршрутизатор X, расстояние до X указывается как бесконечность.

Однако и в алгоритме разделения направлений есть «дыры». Рассмотрим пример, показанный на рисунке. Если линия между С и D будет разрушена, то С сообщит об этом А и В. Однако А знает, что у В есть маршрут до D, а В знает, что такой маршрут есть и у А. И опять мы «сваливаемся» в проблему бесконечного счетчика. 

11. Алгоритмы маршрутизации в Интернет: основные подходы, маршрутизация по состоянию канала.

Основной задачей сетевого уровня является маршрутизация пакетов.

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

**Корректность** – свойство алгоритма маршрутизации, определяющее, что при любых обстоятельствах этот алгоритм либо найдет маршрут для доставки пакета адресату, либо выдаст сообщение о невозможности его доставки.

**Простота** – свойство, определяющее вычислительную сложность алгоритма маршрутизации.

**Устойчивость** – свойство алгоритма маршрутизации сохранять работоспособность независимо от каких-либо сбоев, или изменений топологии.

**Справедливость** – означает, что все пакеты независимо оттого, из какого канала они поступили, будут обслуживаться равномерно, для всех абонентов будет всегда выбираться оптимальный маршрут.

Алгоритмы маршрутизации можно разбить на два больших класса: адаптивные и неадаптивные.

**Неадаптивные алгоритмы** не принимают в расчет текущую загрузку сети и ее текущую топологию. Все возможные маршруты вычисляются заранее и загружаются в маршрутизаторы при загрузке сети. Такая маршрутизация называется статической.

**Адаптивные алгоритмы**, наоборот, определяют маршрут исходя из текущей загрузки и топологии транспортной среды. Адаптивные алгоритмы различаются способом получения информации, временем изменения маршрута и метрикой, используемой при оптимизации.

**Маршрутизация по состоянию канала.**

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

1. Определить своих непосредственных соседей и их сетевые адреса.
2. Измерить задержку иди оценить затраты на передачу до каждого соседа.
3. Сформировать пакет, где укатаны все данные, полученные на шаге 2.
4. Послать этот пакет всем другим маршрутизаторам.
5. Вычислить наикратчайший маршрут до каждого маршрутизатора.

**Определение соседей.**

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

**Оценка затрат.**

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

**Формирование пакетов состояний каналов.**

В пакетах состояний указываются: отправитель, последовательное число, возраст, список соседей и задержки до них.

**Распространение пакетов состояния каналов.**

Рассмотрим сначала базовый алгоритм, в котором СК-пакеты распространяются методом лавины, т.е. СК-пакет рассылается всем соседям. Однако чтобы не потерять контроль и не вызвать неограниченного дублирования СК-пакетов, каждый маршрутизатор ведет счетчик последовательных номеров СК-пакетов, которые он сгенерировал. Все маршрутизаторы запоминают пары маршрутизатор – последовательное число, которые они уже встречали среди полученных СК-пакетов. Если маршрутизатор уже встречал такой пакет, то этот пакет сбрасывается и никуда не дублируется.

**Первая проблема** – размер поля последовательных номеров пакетов. Если это поле будет недостаточно велико, то его переполнение приведет к повтору номеров, а следовательно, к некорректной работе всего алгоритма. Решением здесь является использование достаточно большого поля, например 32-разрядного.

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

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

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

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

**Вычисление нового пути.**

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

12. Маршрутизация в Интернет: OSPF протокол для внутренней маршрутизации.

Внутри каждой сети работают свои алгоритмы маршрутизации и управления. При этом сама сеть называется автономной системой (АС). Алгоритмы маршрутизации, применяемые внутри АС, называются внутренними протоколами шлюзов. Алгоритмы маршрутизации, применяемые для маршрутизации между АС, называются внешними протоколами шлюзов. Наличие стандартов на протоколы маршрутизации позволяет преодолеть различия во внутренней организации автономных систем и обеспечить их совместное функционирование.

Изначально в качестве внутреннего протокола шлюзов использовался протокол по вектору расстояния. По мере роста АС он стал работать все хуже и хуже. Проблемы «бесконечного счетчика» и медленной сходимости не получили удовлетворительного решения.

OSPF - Open Shortest Path First

Свойства OSPF

1. Обновления состояния ссылок происходит используя наводнения (когда каждый роутер отправляет информацию каждому доступному роутеру) когда нужно
2. RFC 2328 в этом стандарте он описан
3. Все роутеры работает по алгоритму Дейкстры
4. Авторизированные обновления
5. Используется много где, сложно

Требования к  протоколу OSPF

1. Алгоритм должен быть опубликован в открытой литературе (отсюда open).
2. Алгоритм не должен быть собственностью какой-либо компании.
3. Алгоритм должен уметь работать с разными метриками маршрутов (расстоянием, пропускной способностью, задержкой и т.п.) и при этом быть динамическим, т.е. реагировать на изменение топологии сети автоматически и быстро.
4. Алгоритм должен поддерживать разные виды сервиса, а также маршрутизацию для трафика в реальном времени одним способом, а для других типов трафика — другим. Для этого в ТСР-пакете есть поле Туре of service, которое не использовалось существующими в то время протоколами.
5. Алгоритм должен обеспечивать балансировку нагрузки во избежание перегрузки и при необходимости направлять потоки по разным каналам (все предыдущие протоколы использовали только один канал — наилучший).
6. Алгоритм должен поддерживать иерархическую маршрутизацию.
7. В алгоритме должна быть усилена безопасность маршрутизаторов для обеспечения защиты от злоумышленников и просто любопытствующей публики, например студентов.
8. В алгоритме должна быть обеспечена возможность маршрутизаторам общаться с помощью туннелирования.

Виды соединений в OSPF

Протокол OSPF поддерживает три вида СПД (системы передачи данных).

1. Каналы типа точка—точка между двумя маршрутизаторами.
2. СПД на основе каналов с множественным доступом и вещанием (большинство ЛВС).
3. СПД на основе коммутации каналов или коммутации пакетов (например, региональные сети с коммутацией пакетов).

При этом протокол OSPF абстрагируется от конкретных сетей и строит их модель в форме ориентированного графа, каждая дуга в котором имеет вес, представляющий собой определенную метрику канала: задержку, расстояние и т.п. В этом графе кратчайший путь определяется на основе весов дуг. Последовательный дуплексный канал между узлами изображается двумя дугами, которые могут иметь разный вес. СПД с множественным доступом представляются звездообразным графом, в котором центральный узел соединен дугами, имеющими нулевой вес с другими узлами (рис. 2.28, 6).

Многие АС сами по себе представляют большие сети. OSPF позволяет разбивать их на области, где каждая область — это либо сеть, либо последовательность сетей. Эти области не пересекаются. Область — это обобщение понятия подсети. Извне топология области не видна. Каждая АС имеет магистральную область, называемую областью 0. Все области одной АС соединяются через магистральную область, возможно с помощью туннелирования. Туннель представляется в графе взвешенной дугой. Любой маршрутизатор, соединенный с двумя или более областями, является частью магистральной области. Топология магистральной области так же, как и топологии других областей, не видна извне.

Внутри области у всех маршрутизаторов одинаковая база данных состояний каналов, и все они используют одинаковый алгоритм нахождения наилучшего пути. Задача маршрутизатора — вычислить наилучший путь до другого маршрутизатора этой области, включая маршрутизатор, соединенный с магистральной областью. Маршрутизатор, соединенный с двумя областями, должен иметь две базы данных и два алгоритма поиска наилучшего пути.

Для поддержания разных типов сервисов OSPF  использует три графа: первый с разметкой, где веса представляют собой задержку, второй — пропускную способность каналов, третий — надежность каналов. Хотя все три графа и требуют соответствующих вычислений, но зато при этом получаем три маршрута, оптимизированных по задержке, пропускной способности и надежности. Во время работы сети возникают три вида маршрутов: внутри области, между областями и между АС. Внутри области вычислить маршрут просто — это наикратчайший путь до маршрутизатора получателя. Маршрутизация между областями всегда выполняется в три этапа: от источника до магистральной области, от магистральной области до области назначения и внутри области назначения. Этот алгоритм навязывает звездообразную топологию АС: магистральная область — это центр (ось), остальные области — лучи (спицы). Пакеты маршрутизируются без изменений за исключением случая, когда область получателя соединена с магистральной областью туннелем.

Маршрутизаторы в OSPF

1. Внутренние, находящийся целиком внутри одной области.
2. Пограничные, соединяющие несколько областей.
3. Магистральные, принадлежащие магистральной области.
4. Пограничные, соединенные с маршрутизаторами других АС.

Маршрутизаторы разных классов могут пересекаться. Например, все пограничные маршрутизаторы являются магистральными, а маршрутизатор из магистральной области, но не находящийся на ее границе — внутренним.

Когда маршрутизатор загружается, он рассылает сообщение HELLO всем своим соседям на линиях типа точка—точка и группам маршрутизаторов в ЛВС (Локальная вычислительная сеть) с множественным доступом, чтобы получить информацию о своем окружении.

В OSPF маршрутизатор обменивается данными не со всеми маршрутизаторами внутри области, а лишь с теми, которые объявлены выделенными маршрутизаторами. Выделенный маршрутизатор смежен со всеми другими выделенными маршрутизаторами. В целях обеспечения надежности каждому выделенному маршрутизатору сопоставляется дублер, который имеет ту же информацию, что и основной маршрутизатор.

Периодически в ходе нормальной работы каждый маршрутизатор рассылает всем своим смежным маршрутизаторам сообщение LINK STATE UPDATE. В этом сообщении он передает информацию о состоянии своих линий и их стоимости в разных метриках для базы данных, описывающей топологию соединений. Это сообщение в целях обеспечения надежности передается с подтверждением (LINK STATE АСК). Каждое такое сообщение имеет номер, позволяющий определить, несет ли пришедшее сообщение новую информацию по сравнению с той, которая имеется в его базе. Рассылают эти сообщения, когда у маршрутизаторов появляются новые линии, разрушаются старые или изменяется вес линии в той или иной метрике.

Сообщение DATABASE DESCRIPTION содержит описание состояния всех каналов из базы данных отправителя. Получатель, сравнивая свои значения с теми, которые имеются у отправителя, может определить, у кого из них наиболее свежая информация.

Используя сообщение LINK STATE REQUEST, маршрутиризатор может в любой момент запросить информацию о любой линии у другого маршрутизатора. Все сообщения передаются как IР-пакеты.

В магистральной области маршрутизаторы выполняют все указанное ранее, а также обмениваются информацией с пограничными маршрутизаторами в целях обеспечения вычисления наилучшего маршрута от любого маршрутизатора магистральной области до любого другого маршрутизатора.

*короткая версия билета:*

**Автономные системы** – сети или группа сетей, которая находится под единым административным управлением и в которой реализуется единая политика маршрутизации.  
Алгоритмы маршрутизации, применяемые внутри АС, называются внутренними протоколами шлюзов. Алгоритмы маршрутизации, применяемые для маршрутизации между АС, называются внешними протоколами шлюзов. Наличие стандартов на протоколы маршрутизации позволяет преодолеть различия во внутренней организации автономных систем и обеспечить их совместное функционирование.  
Изначально в качестве внутреннего протокола шлюзов использовался протокол по вектору расстояния(RIP). По мере роста АС он стал работать все хуже и хуже. Проблемы «бесконечного счетчика» и медленной сходимости не получили удовлетворительного решения.  
**OSPF - Open Shortest Path First**

**Свойства OSPF**

- разбивает АС(автономная система) на зоны  
- изменения состояний линии рассылается лавиной  
- все маршрутизаторы в области используют алгоритм Дейкстры(алгоритм наикратчайшего пути)  
- аутентификация изменений(чтобы изменять систему нужны спец права)  
зоны:  
- магистральные(соединена с каждой немагистральной)  
- немагистральные(передают между собой пакеты только через магистральную)

**работа протокола:**  
1. Маршрутизаторы обмениваются маленькими HELLO-пакетами

2. Обменявшись пакетами, они устанавливают соседские отношения, добавляя каждый друг друга в свою локальную таблицу соседей

3. Маршрутизаторы собирают состояния всех своих линков (связей с соседями), включающие в себя id маршрутизатора, id соседа, сеть и префикс между ними, тип сети, стоимость линка (метрику) и формируют пакет, называемый LSA (Link State Advertisement).

4. Маршрутизатор рассылает LSA своим соседям, те распространяют LSA дальше.

5. Каждый маршрутизатор, получивший LSA добавляет в свою локальную табличку LSDB (Link State Database) информацию из LSA.

6. В LSDB скапливается информация обо всех парах соединённых в сети маршрутизаторов, *то есть каждая строчка таблицы — это информация вида: «Маршрутизатор A имеет соединение со своим соседом маршрутизатором B, между ними сеть такая-то с такими-то свойствами».*

7. После обмена LSA, каждый маршрутизатор знает про все линки, на основании пар строится полная карта сети, включающая все маршрутизаторы и все связи между ними.

8. Среди всех своих соседей маршрутизатор выбирает смежного соседа (designated) , через которого он будет рассылать LSA сообщения в другие области

9. На основании этой карты каждый маршрутизатор индивидуально ищет кратчайшие с точки зрения метрики маршруты во все сети и добавляет их в таблицу маршрутизации

Маршрутизаторы в OSPF

В OSPF различают четыре класса маршрутизаторов.  
1. Внутренние, находящийся целиком внутри одной области.  
2. Пограничные, соединяющие несколько областей.  
3. Магистральные, принадлежащие магистральной области.  
4. Пограничные, соединенные с маршрутизаторами других АС.  
Маршрутизаторы разных классов могут пересекаться.  
Например, все пограничные маршрутизаторы являются магистральными, а маршрутизатор из магистральной области, но не находящийся на ее границе — внутренним.

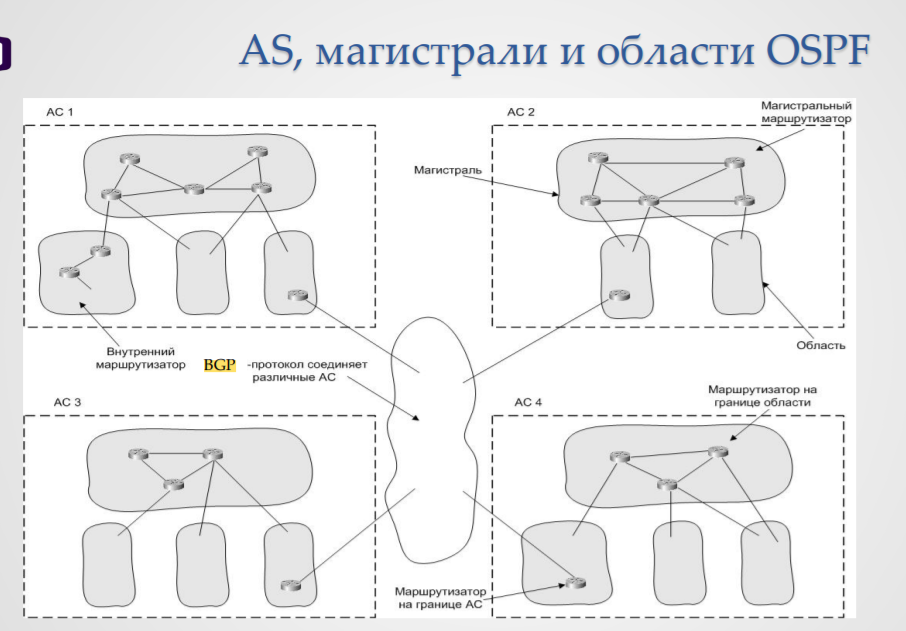
13. Маршрутизация в Интернет: структура Интернета, понятие автономной системы, протокол EBGP внешней маршрутизации.

Интернет состоит из сетей, управляемых разными организациями. Внутри каждой такой сети работают свои алгоритмы маршрутизации и управления. При этом сама сеть называется автономной системой (АС). АС - это сеть, в рамках которой применяется единая политика маршрутизации. Алгоритмы маршрутизации, применяемые внутри АС, называются внутренними протоколами шлюзов. Алгоритмы маршрутизации, применяемые для маршрутизации между АС, называются внешними протоколами шлюзов.

Для маршрутизации пакетов между АС используется протокол пограничных шлюзов — BGP (Border Gateway Protocol).

Все автономные системы связаны между собой либо непосредственно, либо через IXP – Internet eXchange Point по протоколу BGP.

* Каждый маршрутизатор знает все префиксы внутри АС
* Пакеты для других АС пересылаются на ближайший BGP шлюз-маршрутизатор
* BGP маршрутизатор – пограничный шлюз для других АС



Все АС взаимодействуют, используя протокол BGP-4. BGP-4 был разработан чтобы решить следующие проблемы:

* Топология: Интернет – плохо структурированная смесь разнообразных АС
* Автономия АС: каждая АС по-своему определяет стоимость линии, поэтому невозможно построить путь с наименьшей стоимостью
* Доверие: некоторые АС не могут доверять тем маршрутам, которые предлагают другие АС (два конкурирующих провайдера, защита конфиденциальности через территорию неприятеля)
* Политика: разные АС преследуют разные цели (мин. число скачков vs предпочтение одного провайдера перед другими)

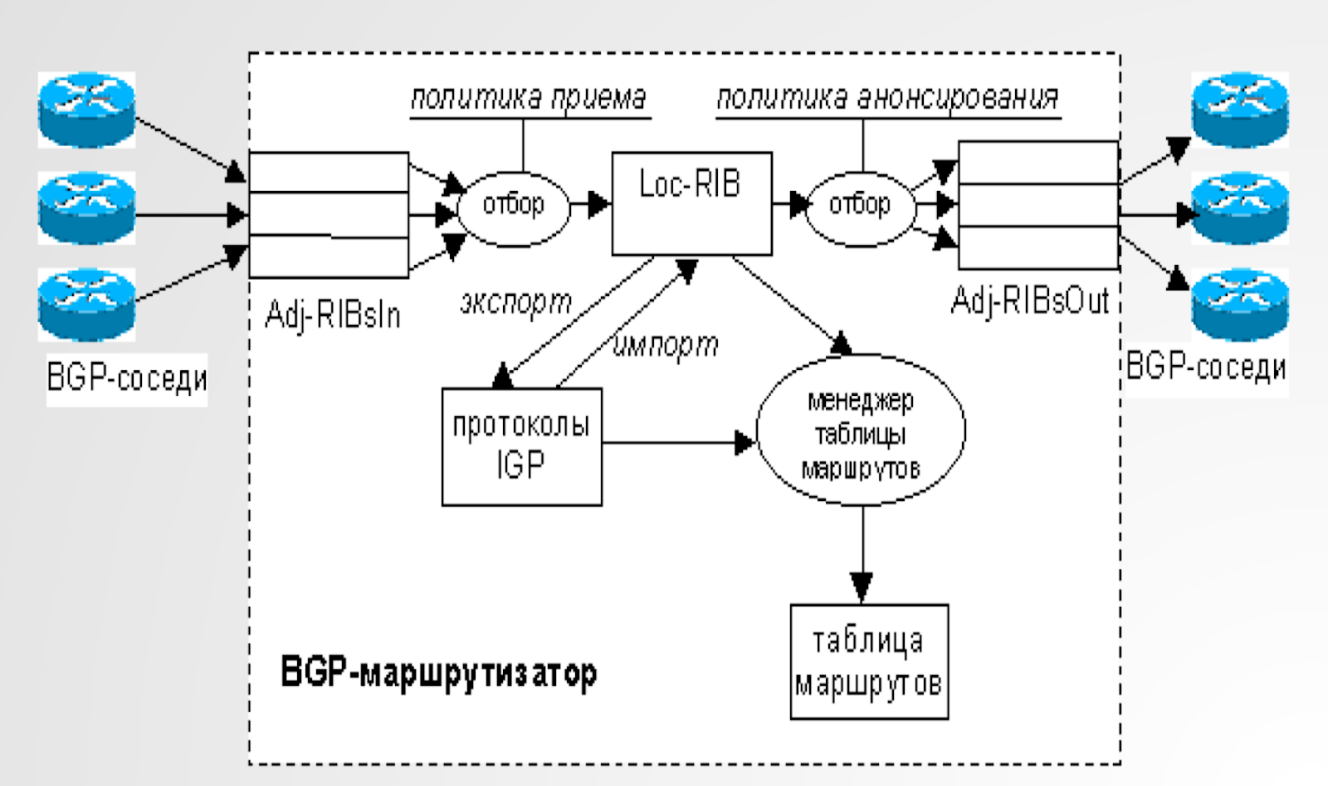
С позиции ВGР-маршрутизатора весь мир состоит из ВGР-маршрутизаторов, соединенных между собой.

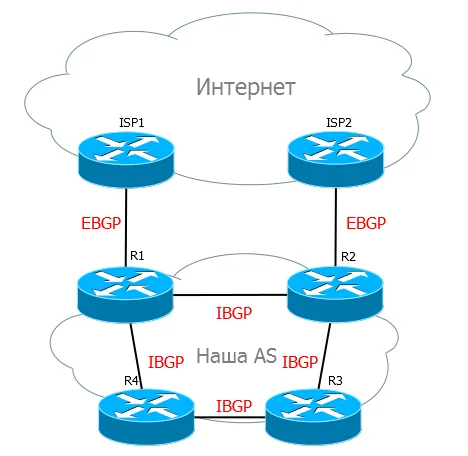
Два ВGР-маршрутизатора соединены, если у них есть общая сеть. Для ВGР-маршрутизатора все сети делятся на три категории: тупиковые сети, которые никуда не ведут и не могут использоваться для транзита пакетов через сеть; сети с множественными соединениями, которые могут использоваться для транзита, если его допускает политика определенной АС; транзитные сети, предназначенные для транзита трафика, возможно с некоторыми ограничениями.

Два ВGР - маршрутизатора взаимодействуют через транспортное ТСР-соединение, что обеспечивает надежность передачи информации и скрывает все подробности от сетей, через которые она проходит.

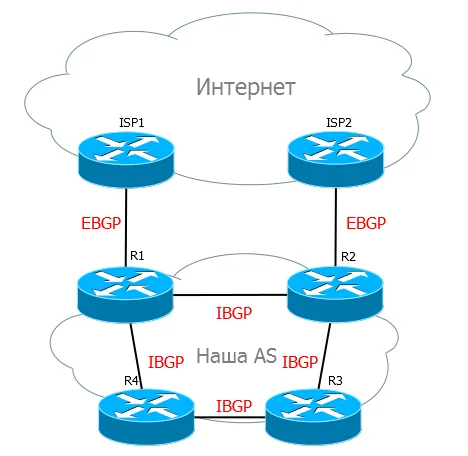
Основы:

* BGP использует «вектор пути»
* Каждый BGP маршрутизатор рассылает список АС, доступных через данную АС - AS\_PATH
  + Например: к сети 171.64/16 можно пройти по пути {AS7,AS52,AS13}
* Наличие цикла в маршруте определяется локально и такие маршруты игнорируются
* Из множества доступных маршрутов выбирается тот, которые наиболее всего соответствует политике АС
* Если маршрутизатор/линии вышли из строя, то маршрут изымается из списка

traceroute -a <destination> даст информацию о номера



14. Понятие автономной системы, ее структуры. Протоколы EBGP и IBGP – назначение и основные различия.



**Автономная система** – это система IP-сетей и маршрутизаторов, управляемых одним или несколькими операторами, имеющими единую политику маршрутизации.

**Типы АС** – оконечные (с одним входом и выходом), и транзитные.

В АС может быть несколько точек входа и выходы ( несколько BGP шлюзов). Если у AS несколько BGP шлюзов, то эти шлюзы

взаимодействуют через IBGP протокол. Все остальные

маршрутизаторы одной и той же AS взаимодействуют по OSPF

протоколу.

* Каждый маршрутизатор знает все префиксы внутри АС
* Пакеты для других АС пересылаются на ближайший IBGP шлюз-маршрутизатор
* EBGP маршрутизатор – пограничный шлюз для других АС

**Основы Border Gateway Protocol** (BGP-4)**:**

* BGP использует «вектор пути»
* Каждый BGP маршрутизатор рассылает список путей (путь – список АС)
  + *AS\_PATH*
* Наличие цикла в маршруте определяется локально и такие маршруты игнорируются
* Из множества доступных маршрутов выбирается тот, которые наиболее всего соответствует политике АС Если маршрутизатор/линии вышли из строя, то маршрут изымается из списка

В этом протоколе легко обнаруживаются циклы на пути – повторение вершин. Такие пути отбрасываются и игнорируются.

**Политика** – это набор правил/ограничений, которыми BGP протокол должен руководствоваться при выборе пути. (ограничение по стоимости, надежности, географические признаки и т.д.)

BGP поддерживает бесклассовую адресацию является протоколом прикладного уровня, использует ТСР.

BGP-4 был разработан чтобы решить следующие проблемы:

* **Топология**: Интернет плохо структурированная смесь разнообразных АС
* **Автономия АС**: каждая АС по-своему определяет стоимость линии, поэтому невозможно построить путь с наименьшей стоимостью
* **Доверие**: некоторые АС не могут доверять тем маршрутам, которые предлагают другие АС (два конкурирующих провайдера, защита конфиденциальности через территорию неприятеля)
* **Политика**: разные АС преследуют разные цели (мин. число скачков vs предпочтение одного провайдера перед другими)

**IBGP** (Interior BGP) – протокол между BGP маршрутизаторами внутри одной АС.

**EBGP** (Exterior BGP) – протокол между BGP маршрутизаторами между различными АС.

(EBGP маршрутизатор – пограничный шлюз для других АС)

**Различия IBGP и EBGP:**

* EBGP устраняет циклы с помощью AS-Path. Если в списке уже был номер локальной AS, этот маршрут отбрасывается.
* Когда маршрут объявляют внутри автономной системы, AS-Path не меняется. IBGP использует полносвязную сеть (mash).
* В этом случае маршрут, полученный от соседа IBGP, не объявляется другим соседям IBGP. Это позволяет всем маршрутизаторам иметь все маршруты и при этом избегать петель.

15. Маршрутизация в Интернет: взаимосвязь протоколов OSPF и BGP

**OSPF протокол:**

1. Маршрутизаторы обмениваются маленькими HELLO-пакетами
2. Обменявшись пакетами, они устанавливают соседские отношения, добавляя каждый друг друга в свою локальную таблицу соседей
3. Маршрутизаторы собирают состояния всех своих линий (связей с соседями), включающие в себя id маршрутизатора, id соседа, сеть и префикс между ними, тип сети, стоимость линии (метрику) и формируют пакет, называемый LSA (Link State Advertisement).
4. Маршрутизатор рассылает LSA своим соседям, те распространяют LSA дальше.
5. Каждый маршрутизатор, получивший LSA добавляет в свою локальную табличку LSDB (Link State Database) информацию из LSA.
6. В LSDB скапливается информация, обо всех парах соединённых в сети маршрутизаторов, т.е. каждая строка таблицы — это информация вида: «Маршрутизатор A имеет соединение со своим соседом маршрутизатором B, между ними сеть такая-то с такими-то характеристиками».
7. После обмена LSA, каждый маршрутизатор знает про все линии, на основании пар строится полная карта сети, включающая все маршрутизаторы и все связи между ними.
8. Среди всех своих соседей маршрутизатор выбирает смежного соседа (designated) , через которого он будет рассылать LSA сообщения в другие области
9. На основании этой карты каждый маршрутизатор индивидуально ищет кратчайшие по метрике маршруты во все сети из других областей и добавляет их в таблицу маршрутизации.

***подробно про OSPF - билет 12***

**Основы Border Gateway Protocol** (BGP-4)**:**

* BGP использует «вектор пути»
* Каждый BGP маршрутизатор рассылает список путей (путь – список АС)
  + *AS\_PATH*
* Наличие цикла в маршруте определяется локально и такие маршруты игнорируются
* Из множества доступных маршрутов выбирается тот, которые наиболее всего соответствует политике АС Если маршрутизатор/линии вышли из строя, то маршрут изымается из списка

***подробно про BGP - билет 14***



**Взаимосвязь протоколов OSPF и BGP:**

Внутри АС маршрутизация происходит по протоколу OSPF, а различные АС взаимодействуют по протоколу BGP

16. Понятие групповой маршрутизации, протоколы групповой маршрутизации.