# **Экзаменационные вопросы к курсу "Введение в компьютерные сети" 2022 год.**

1. Принципы построения Интернет. Модели сетевого взаимодействия OSI ISO и TCP/IP. Базовая модель взаимодействия сетевых приложений. Понятия сервиса, интерфейса и протокола.

2. Способы коммутации потоков данных в Интернете. Виды задержек передачи данных при пакетной коммутации и их свойства. Обоснование необходимости обратной связи при передачи данных в сети на математической модели.

3. Простая модель очереди и свойства очередей. Формула Литтла для стационарных потоков с фиксированной средней скоростью. Условия ее применения.

4. Простая модель очереди и свойства очередей. Формула Литтла для случайных потоков с пуассоновским распределением поступления заявок. Условия ее применения.

5. Как устроен и работает пакетный коммутатор. Методы оценки сквозной задержки при пакетной коммутации. Метод справедливой взвешенной очереди и его свойства, условия применимости.

6. Методы оценки сквозной задержки при пакетной коммутации. Понятие max-min справедливости. Теорема о необходимых и достаточных условиях max-min справедливости.

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

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

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

10. Явление перегрузки, причины ее возникновения и основные методы борьбы с ней. Математические модели перегрузки.

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

12. Определение размера окна перегрузки и величины timeout в TCP.

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

14. Управление потоком в ТСР: Оценка зависимости скорости потока от RTT и вероятности сброса пакета.

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

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

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

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

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

20. Маршрутизация в Интернет: взаимосвязь протоколов OSPF и BGP.

21. Понятие групповой маршрутизации, протоколы групповой маршрутизации.

22. Маршрутизация на L2, ее отличие от маршрутизации на L3. Протокол STP, алгоритм построения ST дерева коммутатором.

23. Представление о коммутации по меткам – MPLS протокол. Понятие VPN – Virtual Private Network.

24. Теоретические основы передачи данных (ограничения на пропускную способность передачи сигналов, взаимосвязь пропускной способности канала и ширины его полосы пропускания). Среды передачи (магнитные носители, витая пара, среднеполосный и широкополосный кабели, оптоволокно, сравнение кабелей и оптоволокна).

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

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

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

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

29. Физические среды передачи данных. Беспроводная связь (электромагнитный спектр, радиопередача, микроволновая передача, видимое излучение). Протоколы МАСА.

30. Семейство протоколов IEEE 802.11. Система передачи данных WiFi: принципы организации, структура кадра, алгоритм функционирования.

31. Принципы организации и функционирования семейства протоколов IEEE 802.3: математическая модель и оценка производительности.

32. Проблемы передачи данных на канальном уровне. Сервис, предоставляемый сетевому уровню. Обнаружение и исправление ошибок (Коды исправляющие ошибки, коды обнаруживающие ошибки).

33. Математическая модель и оценки числа состязаний в протоколе множественного доступа к каналу (динамическое vs статическое выделение канала). Мат. модель системы ALOHA. Сравнение производительности систем: чистая ALOHA, слотированная ALOHA. Протоколы множественного доступа с обнаружением несущей (настойчивые и не настойчивые CSMA, CSMA с обнаружением коллизий).

34. Протокол IEEE 802.3 и система передачи данных Ethernet (кабели, способ физического кодирования, понятие коллизии, алгоритм вычисления задержки, МАС подуровень, структура кадра, LLC подуровень).

35. Сетевые коммутаторы: организация, основные функции, принципы функционирования. Коммутатор канального уровня с обучением. Виртуальные сети на основе протокола IEEE 802.1Q.

36. Сетевой уровень в Интернет: адресация IPv4 и IPv6, протокол маршрутизации IP, протоколы ARP, RARP, DHCP.

37. Транспортный уровень: адресация, установление соединения, разрыв соединения, управление потоком и буферизацией, восстановление последовательности сегментов.

38. Ключевые функции системы безопасности компьютерных систем.

39. Безопасность информации в сетях: основные понятия (угрозы, информация, документы, уязвимость, нарушитель, информационная безопасность, целостность, конфиденциальность, доступность, атака).

40. Понятия угрозы и уязвимости в компьютерных сетях, классификация угроз.

41. Понятия идентификации, аутентификации и авторизации. Примеры алгоритмов.

42. Функции монитора безопасности. Объектно-субъектная модель управления доступом.

43. Понятия и виды политики безопасности.

44. Основные виды шифрования. Алгоритмы шифрования с закрытым ключом. Примеры.

45. Основные виды шифрования. Алгоритмы шифрования с открытым ключом. Примеры.

46. Информационная безопасность: основные задачи. Протоколы установления подлинности на основе закрытого ключа, протокол Диффи-Хеллмана. Электронная подпись.

47. Информационная безопасность: контроль доступа и защита от компьютерных атак. Межсетевые экраны и их виды. Системы обнаружения и предотвращения компьютерных атак (метод аномалий и метод злоупотреблений).

48. Служба DNS: основные функции, структуры данных, принципы функционирования. Режим адресации anycast.

49. Организация, функционирование и основные протоколы почтовой службы и прикладной протокол FTP.

50. NAT: основные функции, типы и принципы функционирования, влияние на приложения.

51. Устройство ЦОД. Понятие облачных вычислений. Виртуализация и масштабирование.

52. Современные проблемы компьютерных сетей Программно Конфигурируемые Сети (ПКС): структура, принципы функционирования, протокол Open Flow.

53. Протокол Open Flow, организация и принципы работы ПКС коммутатора, маршрутизация в ПКС сетях

## **1. Принципы построения Интернет. Модели сетевого взаимодействия OSI ISO и TCP/IP. Базовая модель взаимодействия сетевых приложений. Понятия сервиса, интерфейса и протокола.**

**OSI** - модель взаимодействия открытых систем , была разработана для определения международных стандартов компьютерных сетей. Модель OSI включает в себя семь уровней: физический, канальный, сетевой, транспортный, сессии, представления и прикладной:

**TCP/IP** - сетевая модель передачи данных. Модель описывает способ передачи данных от источника информации к получателю. Состоит из 4 уровней: прикладной, транспортный, межсетевой, канальный.

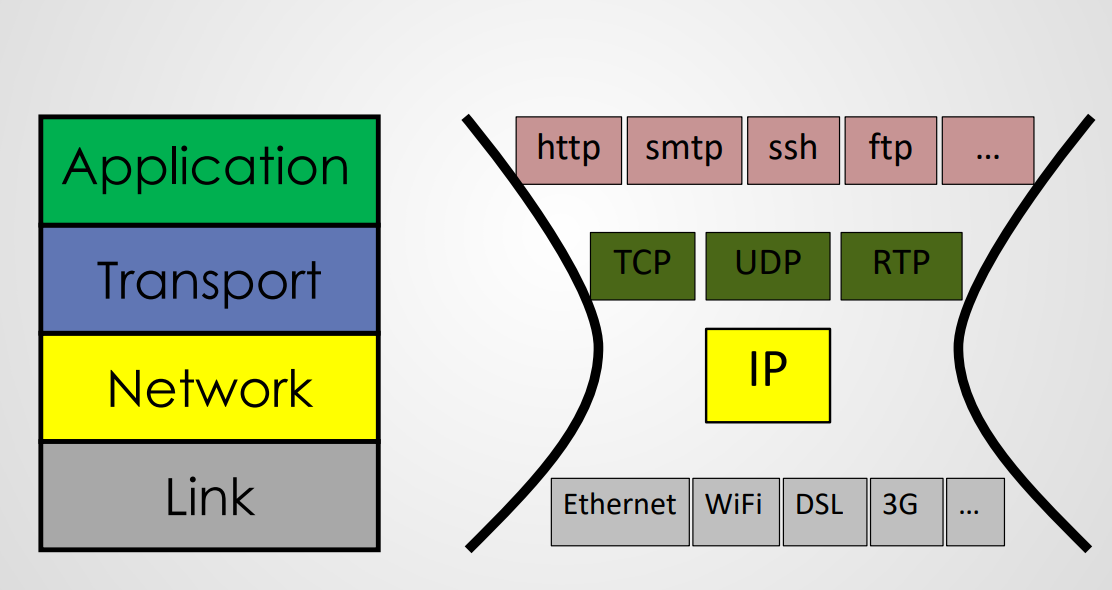
**Сервис** - это услуги, которые нижерасположенный уровень оказывает по запросам вышерасположенного. В этой модели нижерасположенный уровень свои услуги может предоставлять только вышерасположенному уровню.

**Интерфейс** определяет формирование и передачу запроса на услугу.

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

**Функции уровней**:

***Сетевой*** - выбор оптимального маршрута для пакета от отправителя к получателю. ***Транспортный*** - принять данные с вышерасположенного уровня, по необходимости разделить, обеспечить доставку до адресата, передать на сетевой уровень. ***Сессии(сеансовый)*** - позволяет пользователям устанавливать между собой сессии. ***Представления*** - обеспечивает решение часто возникающих проблем, связанных с представлением данных при передаче. ***Приложений*** - обеспечивает работу часто используемых приложений, например передачу файлов.



## **2. Способы коммутации потоков данных в Интернете. Виды задержек передачи данных при пакетной коммутации и их свойства. Обоснование необходимости обратной связи при передачи данных в сети на математической модели.**

Одним из принципов построения Интернета является принцип коммутации пакетов.

Есть два основных типа коммутации потоков данных: коммутация пакетов и коммутация каналов.

Сеть с **коммутацией каналов** подразумевает создание соединения (канала) между двумя узлами, прежде чем они начнут обмен данными. У каждого соединения точка-точка своя гарантированная, пропускная способность, доступная тому кто передает по ней.

Три этапа соединения при коммутации каналов: установка соединения, передача, разрыв.

Проблемы коммутации каналов:

* Неэффективность: взаимодействие между компьютерами в сети очень неравномерно. Если каждая коммуникация будет использовать выделенное соединение, это будет крайне не эффективно.
* Большой разброс скоростей взаимодействия (набор текста или просмотр видео с web)
* Управление состоянием взаимодействия: все коммутаторы, задействованные в соединении должны поддерживать согласовано состояние взаимодействия

**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.

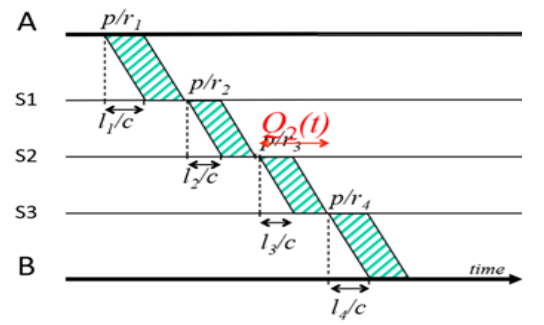
**Задержка распространения** – время *t* необходимое одному биту, чтобы пройти по линии длины *l* со скоростью *с*. *(t\_l = l / c*).

**Задержка пакетизации** - время от момента когда первый бит пакета поступает на линию до момента когда последний бит пакета поступает на линию. *(t\_p = p / r*) Этот вид задержки определяет то как быстро биты помещают в линию.

**Сквозная задержка (e2e delay)** – это время от момента когда первый бит пакета попадает на линию до момента когда последний бит пакета поступает к получателю.

Формула в случае одного потока:

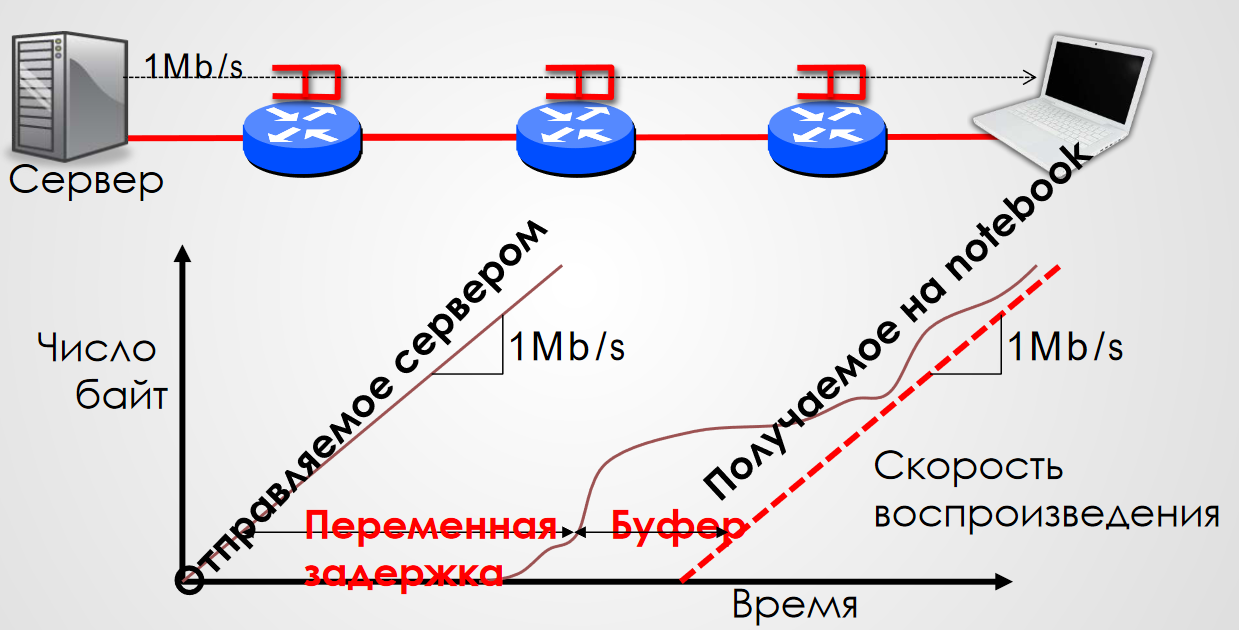
Маршрутизаторы в Интернете работают по принципу «store and forward», т.е. они получают весь пакет целиком, анализируют заголовок, принимают решение по какой линии пересылать пакет далее и, только после этого, начинают передачу пакета далее. Некоторые коммутаторы могут начинать пересылку пакета сразу, как только они увидели заголовок пакета, но обычно маршрутизаторы так не поступают.



Пусть есть несколько потоков, на какой-то коммутатор пришло сразу два пакета. Один надо буферизовать, а второй можно протолкнуть на линию. Красным выделена **задержка буферизации**. Если места в буфере нет, то пакет будет потерян. Все предыдущие задержки детерминированы, кроме задержки на буферизацию.

Формула в случае нескольких потоков:

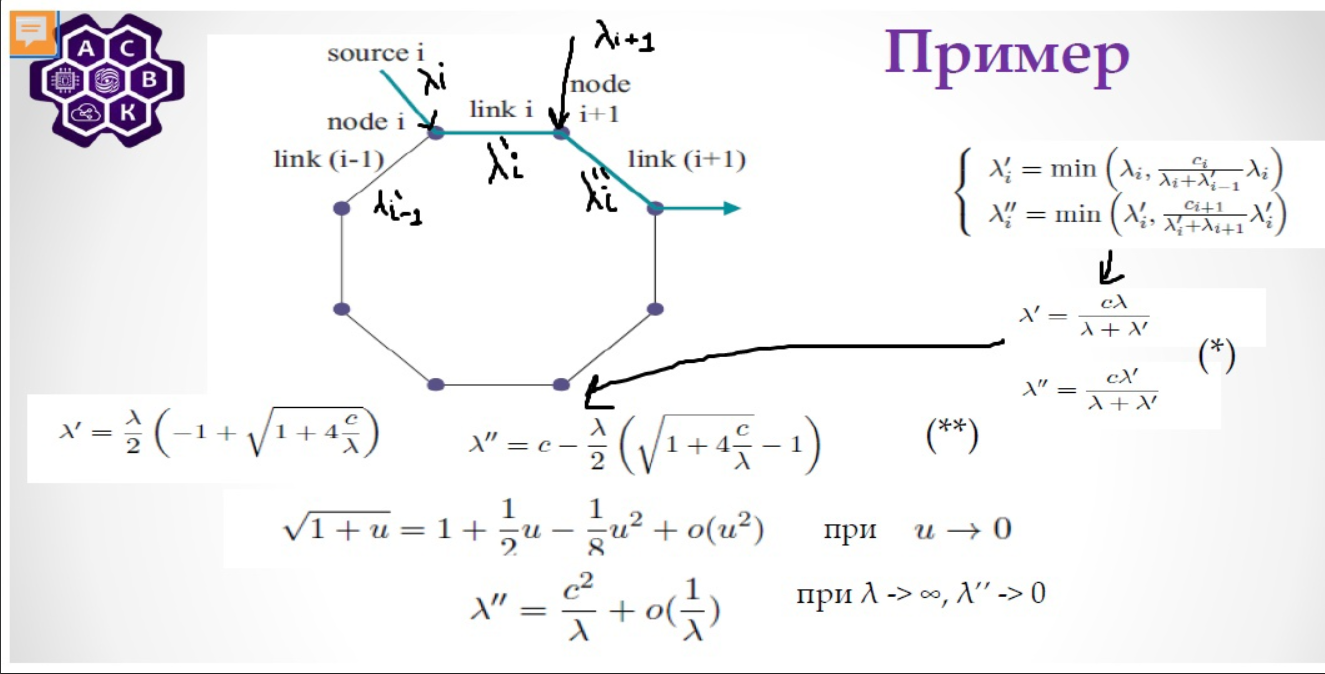
Когда несколько пакетов должны пройти от S2 к S3, то некоторые из них должны быть временно сохранены в буфере или очереди. В этом случае говорят о **заторе на линии**, т.к. у нее образуется очередь. Буфер очереди позволяет избежать немедленного сброса пакетов. Чем больше буфер, тем меньше вероятность что нам придется сбросить пакет, ожидающий передачи.



е2е задержка не может быть меньше суммы задержек на распространение и пакетизацию по всем линиям маршрута. Верхняя граница для е2е задержки – учет всех задержек в очередях на всех коммутаторах. Так как буфер в каждом маршрутизаторе конечен и обслуживается FIFO, то мы можем оценить наихудший случай. Скорость поступления бит в буфер воспроизведения не может превосходить скорости последней линии.

Красная пунктирная линия показывает скорость воспроизведения видео – она в точности такая же что и скорость скачивания с сервера – 1Мб/с.

**Итого сквозная задержка состоит из трех: распространения, пакетизации, буферизации.**



Пример. 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, без обратной связи от сети. 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.**

## **3. Как устроен и работает пакетный коммутатор. Методы оценки сквозной задержки при пакетной коммутации. Метод справедливой взвешенной очереди и его свойства, условия 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

Адреса хранятся в хэш-таблице и ищутся по точному совпадению.

**Оценка сквозной задержки при пакетной коммутации:**

* задержка распространения – время распространения одного бита по каналу длины со скоростью

  + не зависит от пропускной способности канала
* задержка пакетизации – время, за которое все биты пакета с первого до последнего переданы в канал

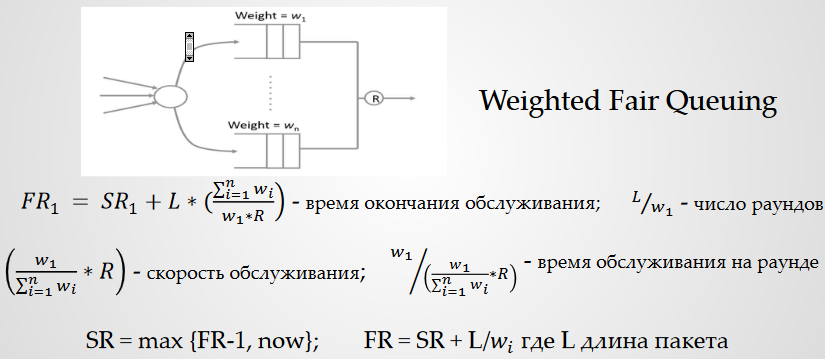
  + скорость передачи = пропускная способность – определяется сетевым интерфейсом
* задержка буферизации:
  + Q(t)= A(t) - D(t)
  + Причина: буферизация, т.е. пакеты находятся в очереди
* сквозная задержка (e2e) – время от момента когда первый бит пакета попадает на линию до момента когда последний бит пакета поступает к получателю
  + сумма задержек(задержка распространения + задержка пакетизации + задержка буферизации)
  + – **оценка e2e задержки**

**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

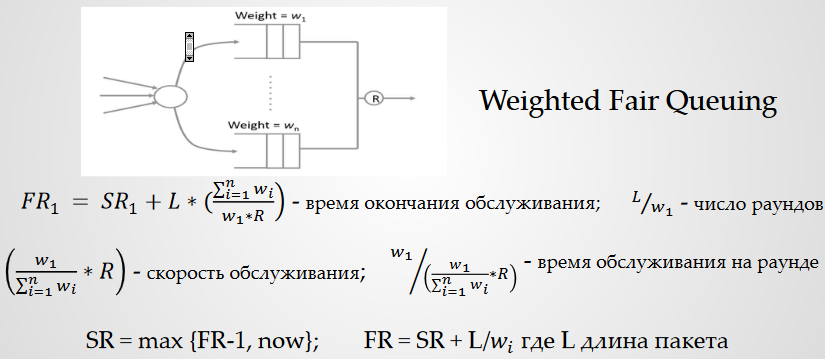
* Пакеты обрабатываются в соответствии

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:**

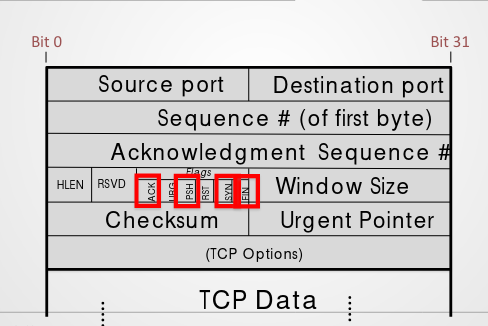
Коммутатор(Ethernet):

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

Маршрутизатор(IP):

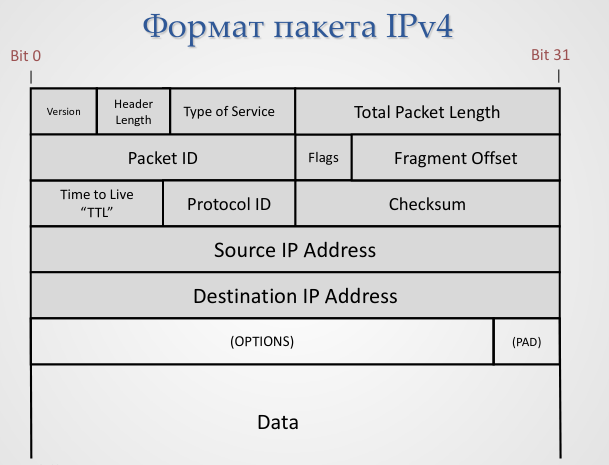
* Поиск адреса: IP.
* Ищут совпадение по самому длинному префиксу, а не точное совпадение.

## **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).

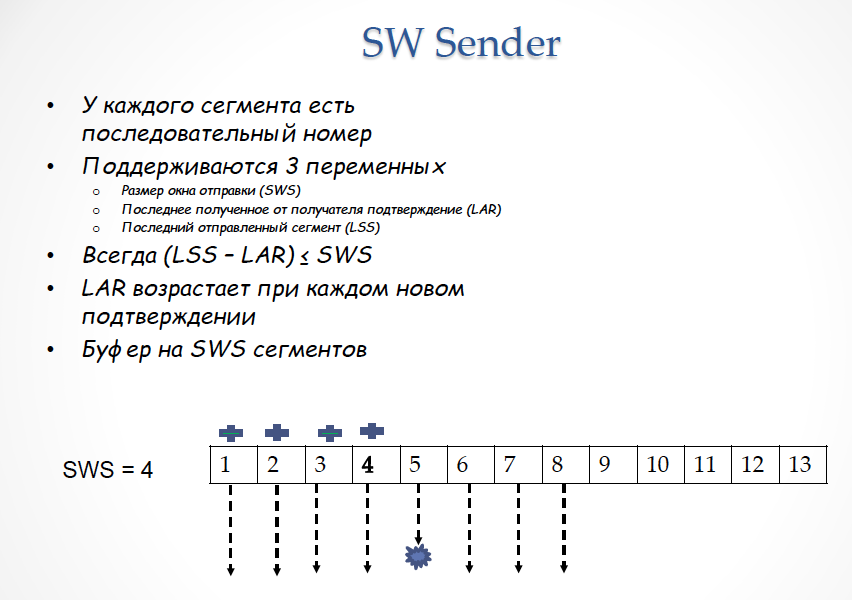
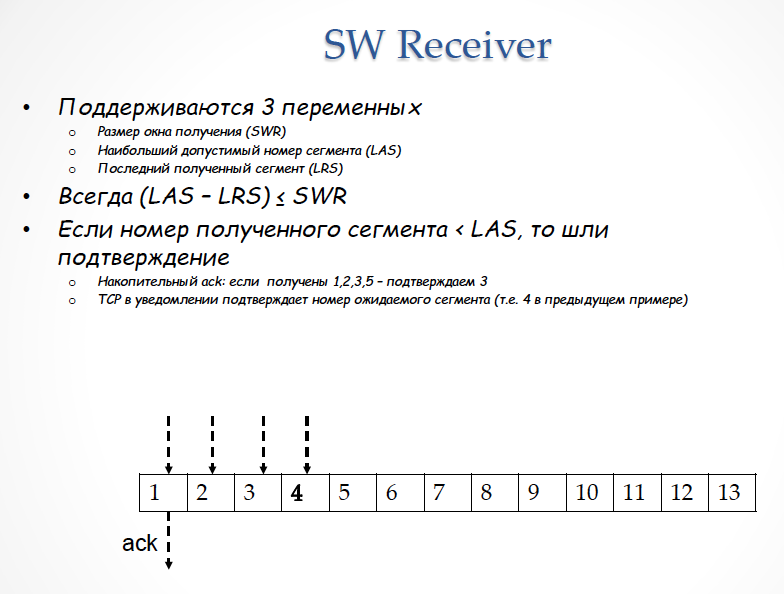
Отправляем данные, ждем ack, отправляем след данные, если timeout, то еще раз посылаем те же данные, получатель получается только шлет ack и все.

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. Математические модели перегрузки.**

**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.

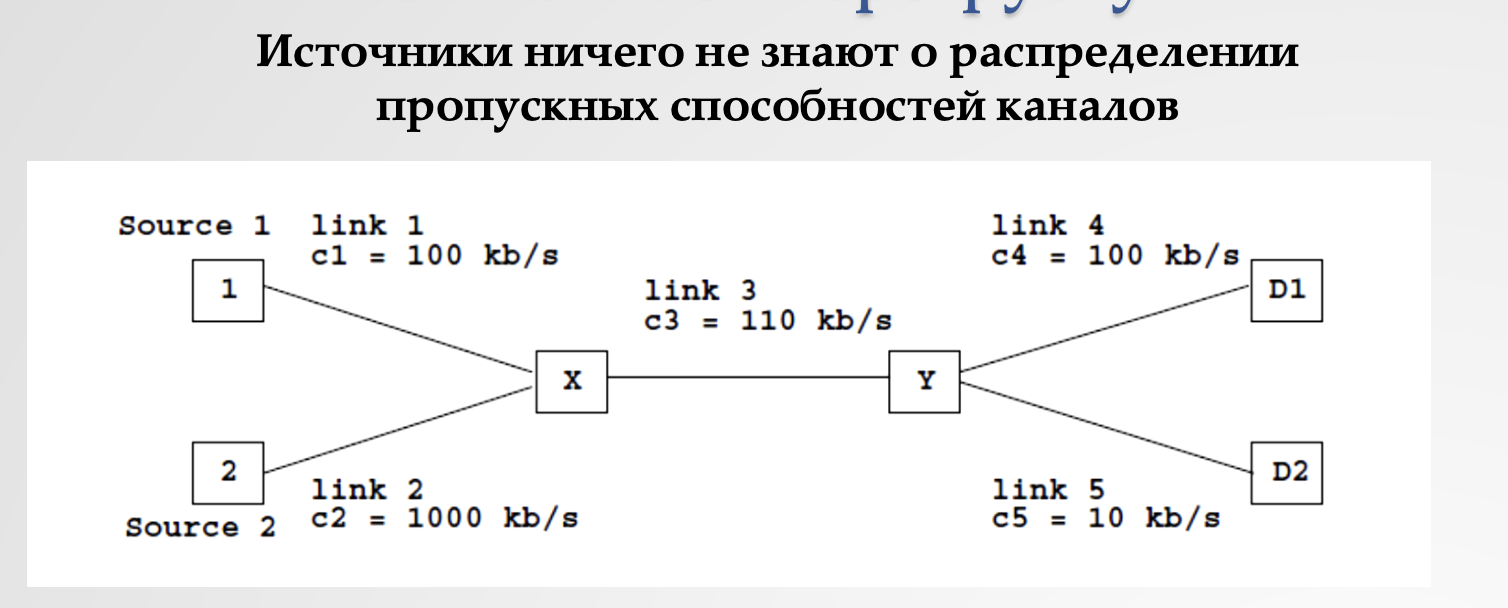
Существуют следующие методы:

* 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: **

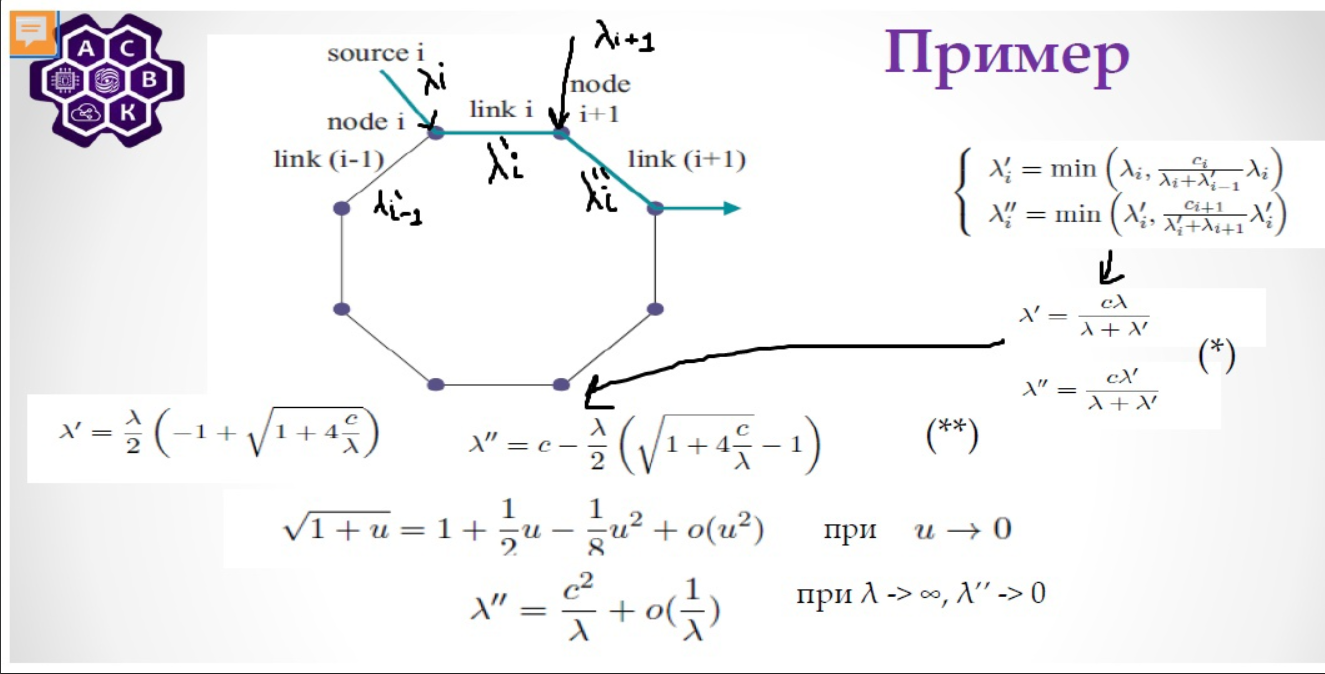
1. Пусть надо связать s1 – d1, s2 – d2. Предполагаем, что у источников s1 и s2 нет обратной связи с сетью. Тогда s1 будет слать пакеты на скорости с1, а s2 – c2. Но на линии l3 эти потоки не пройдут.

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.

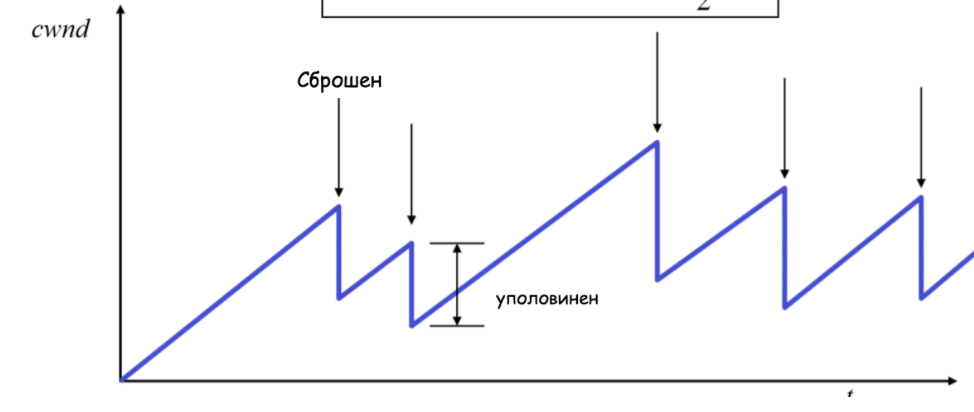
Это классический прием для управления перегрузками в сетях. Его идея в том, чтобы как можно быстрее нащупать предел имеющегося ресурса, после чего прекратить наращивать скорость выброса пакетов в сеть.

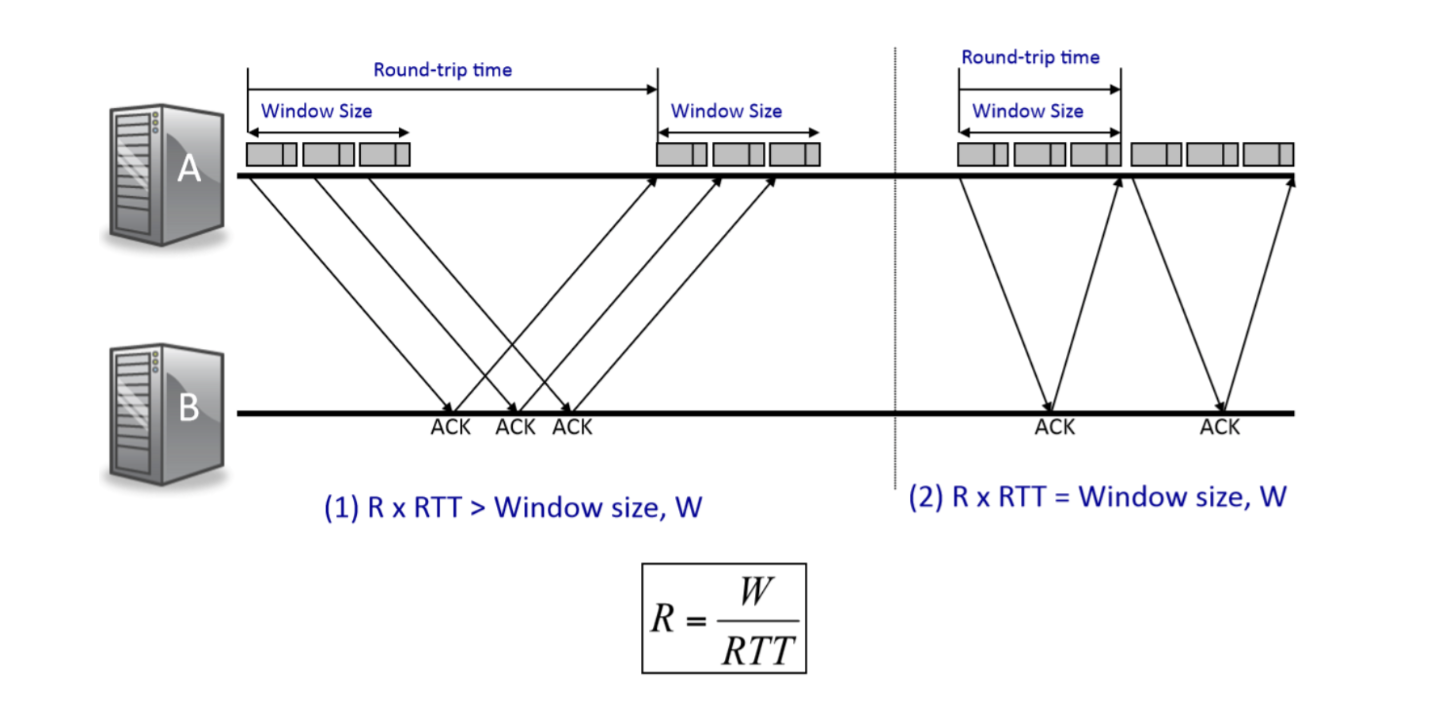
Начнем с аддитивного увеличения. Каждый раз когда пакет был получен корректно (АСК), отправитель увеличивает окно на 1/W, где W – текущий размер окна перегрузки. Увеличиваем окно на 1/W для каждого пакета, а пакетов в окне - W, то по исчерпании окна, его размер будет увеличен на 1. Это медленное увеличение 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.)*

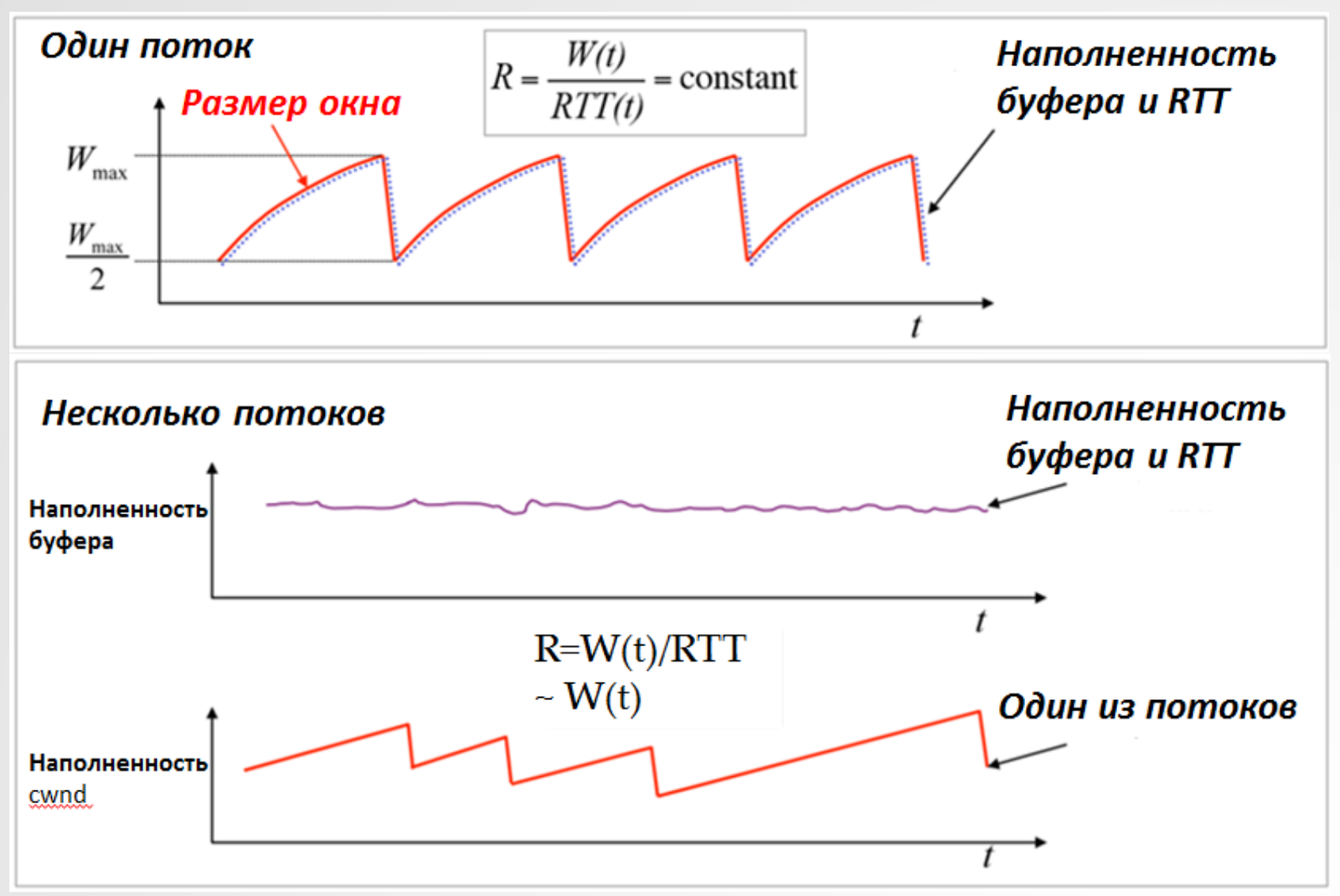
Фактически у нас не плавная монотонная прямая а ступеньки. Мы увеличиваем окно на 1 каждое RTT.





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 /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

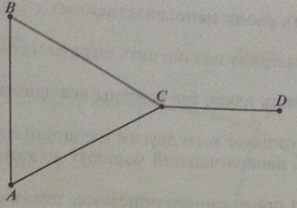
второе – значение задержки до места назначения, которое может измеряться в разных единицах

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.**

***основные понятия - билет 15***

**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.**

* АС можно разбивать на области
* Изменения состояний линии рассылаются лавиной по необходимости
* Каждый маршрутизатор использует алгоритм Дейкстры
* Изменения аутентифицируются

**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.

Типы маршрутизаторов:

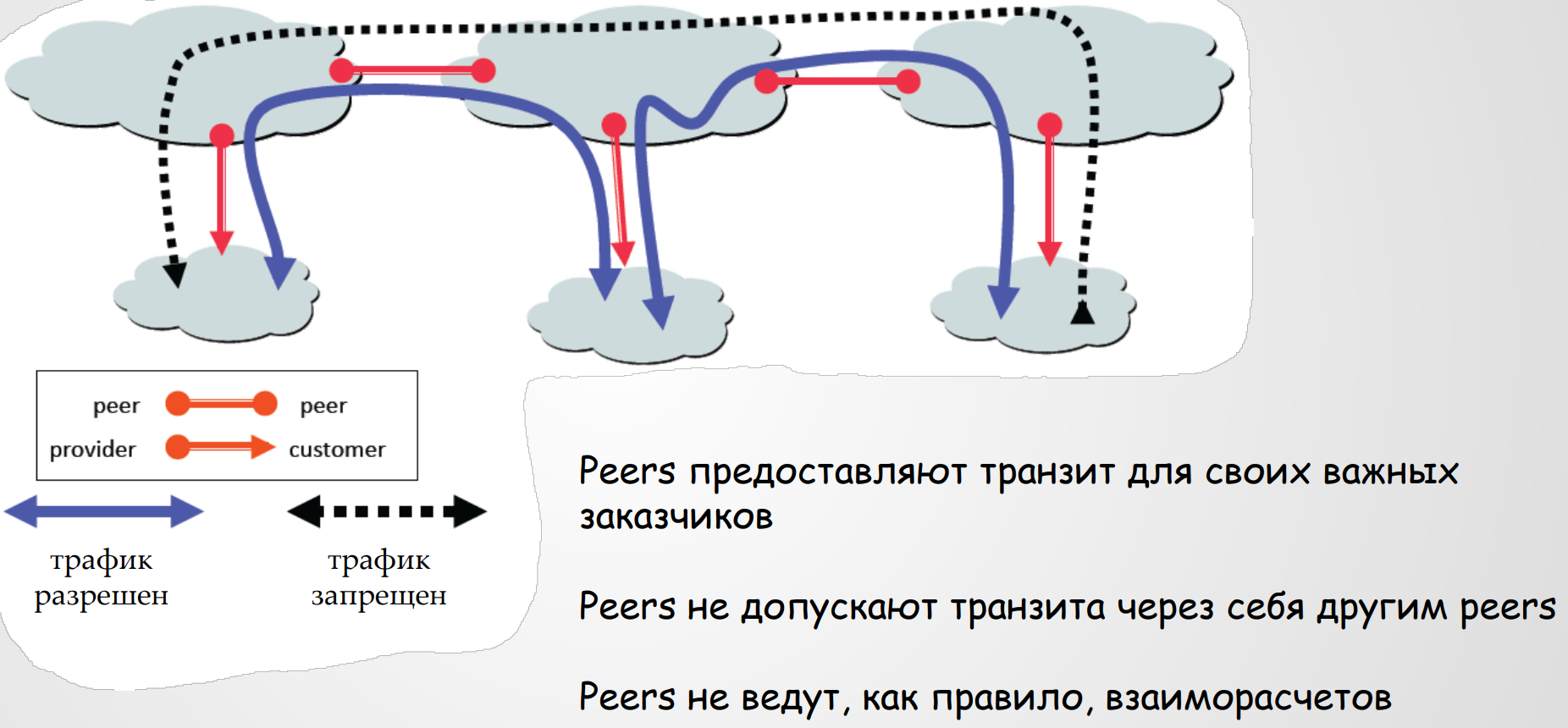
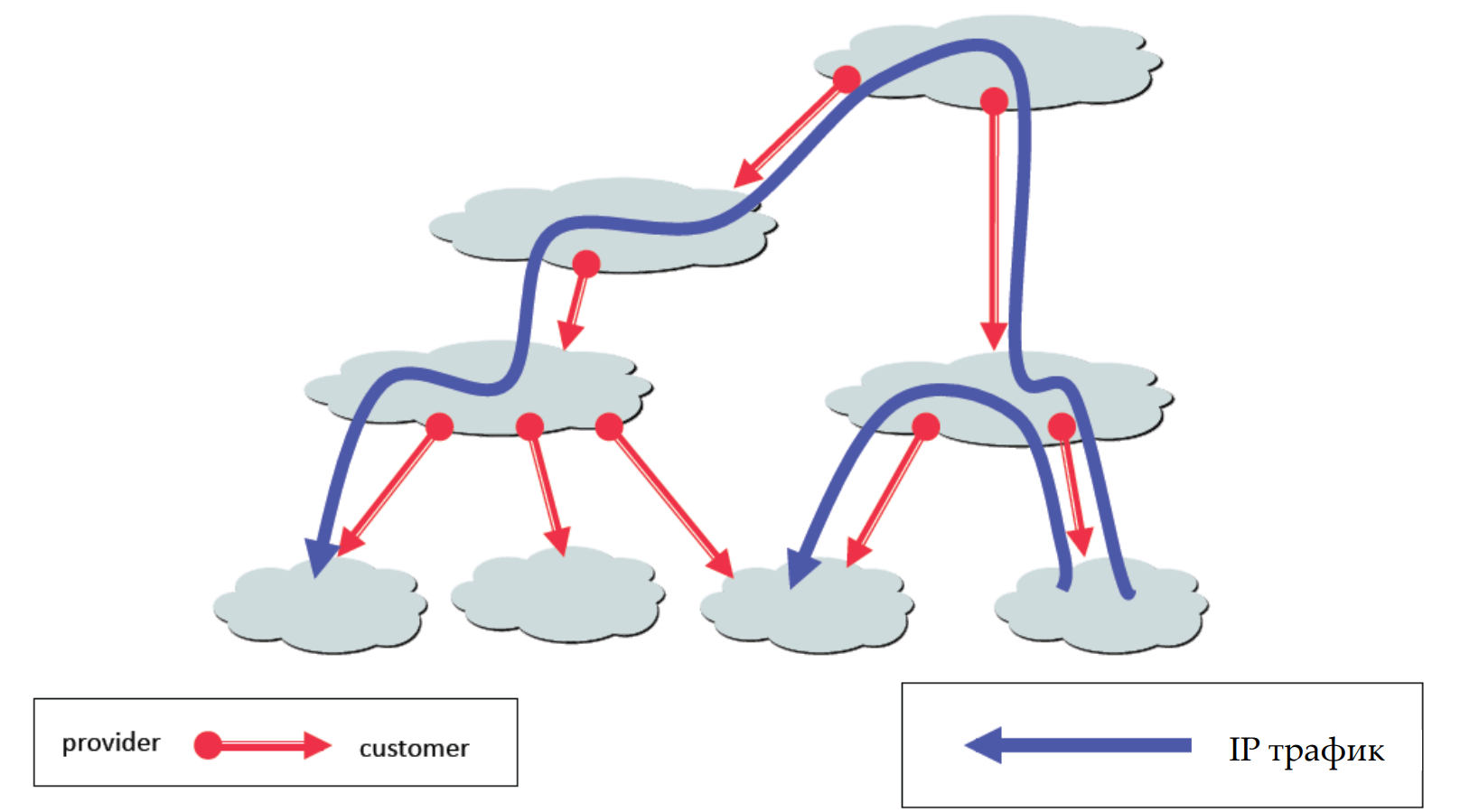
* **Внутренний маршрутизатор** (internal router) — маршрутизатор, все интерфейсы которого принадлежат одной области. У таких маршрутизаторов одинаковая база данных состояния каналов.
* **Пограничный маршрутизатор** (area border router, ABR) — соединяет одну или больше областей с магистральной областью и выполняет функции шлюза для транзитного трафика.
* **Магистральный маршрутизатор** (backbone router) —маршрутизатор у которого всегда хотя бы один интерфейс принадлежит магистральной области.
* **Пограничный маршрутизатор автономной системы** (AS boundary router, ASBR) — это маршрутизатор, один из портов которого находится в домене OSPF протокола, а другой в домене любого из внутренних шлюзовых протоколов (например RIP или IGRP).

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

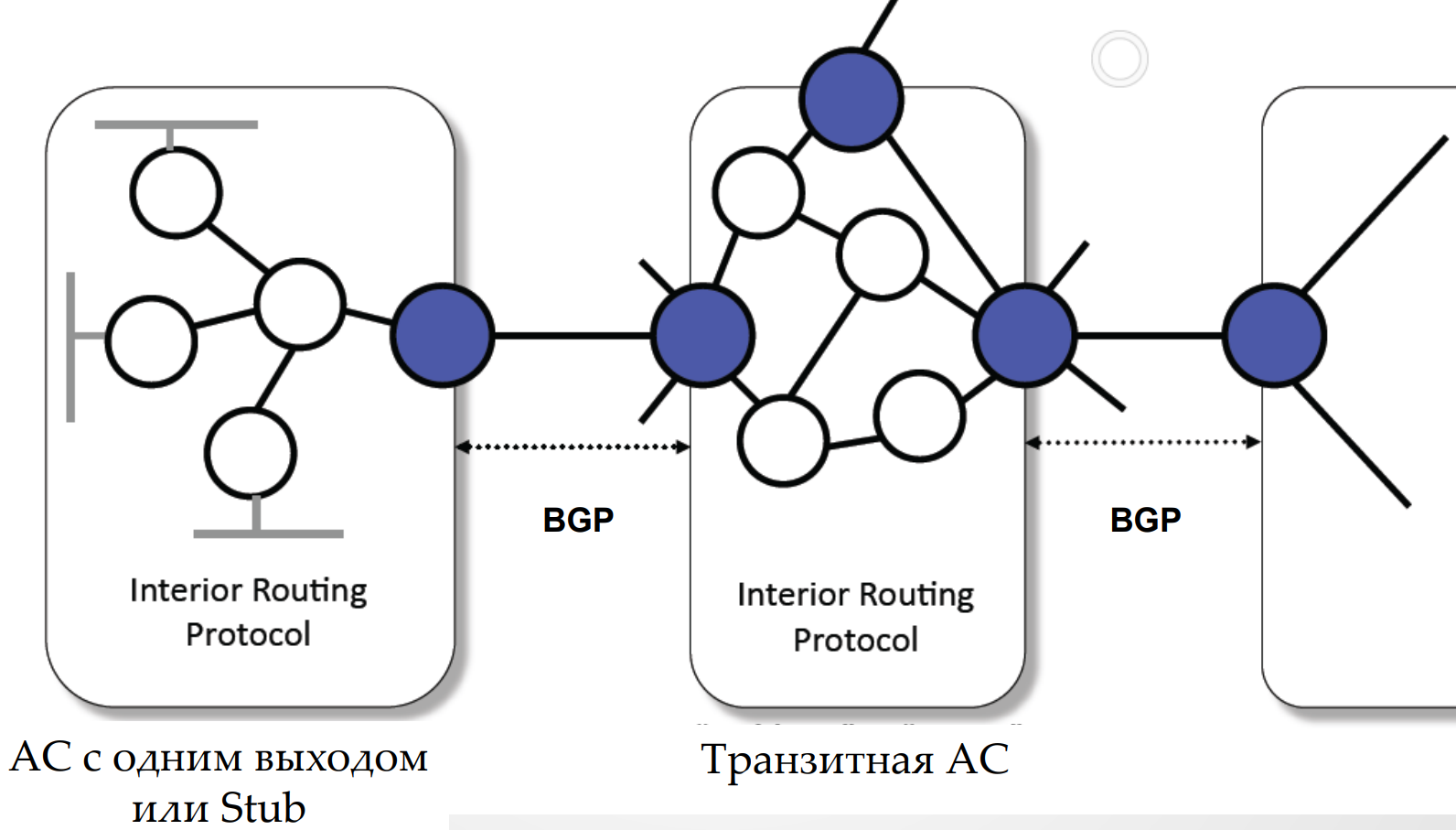
Интернет имеет иерархическую структуру.

Цель иерархии – масштабирование (простота наращивания размера сети без дополнительных переделок существующего и доп. инвестиций)

*Конкуренция среди провайдеров и поставщиков оборудования.*

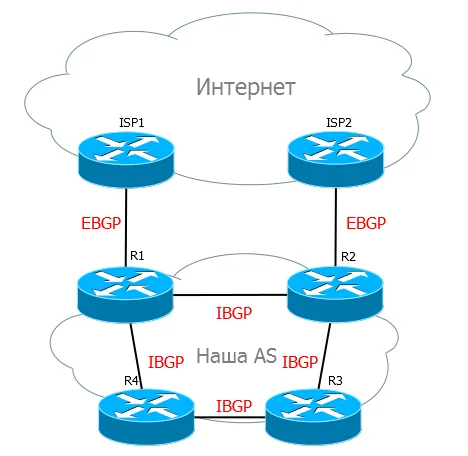
Иерархия заказчиков и провайдеров: Peering:

* Интернет состоит из множества независимо управляемых АС
* Каждая АС использует свой внутренний протокол маршрутизации на уровне областей и IGP протокол между областями
* Оконечные АС используют простую маршрутизацию по умолчанию
* Транзитные АС должны сами определять какой выход использовать
* Для взаимодействия АС должны использовать BGP-4 протокол

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

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

(EBGP маршрутизатор – пограничный шлюз для других АС)



## **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 17***

**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 19***



**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.**

## Групповая маршрутизация - multicast routing. Потребность в таком виде маршрутизации возникает когда нам надо направить байтовые потоки разным приложениям одновременно.

В протокол IPv4 (этот протокол в виде примера) есть несколько типов адресов:

* unicast -тот адрес, которое показывает конкретное устройство
* multicast - показывает группу устройств сети
* broadcast - адрес широковещательной рассылки

Если мы хотим сформировать группу, то надо используется протокол IGMP, при помощи которого устройства могут заявить, что они хотят принадлежать группе и получать трафик, адресованный этой группе. Так решается вопрос об определение членов одной группы.

**Рассылка лавиной** - маршрутизаторы отправляют пакеты по всем направлениям.

Очевидные недостатки:

* Дублирование пакетов
* Склонность к зацикливанию пакетов
* Крайне неэффективное использование ресурсов сети

## **Вещание по обратному пути**

* Источник потока имеет обычный IP.
* Роутеры, предназначенные для групповой маршрутизации, дополнительно получают IP класса D. Основной никуда не пропадает
* Появляются таблицы групповой маршрутизации
* То же самое, что и просто таблицы маршрутизации, только для групповых адресов
* Таблица соответствий вида “групповой адрес” → “адреса маршрутизаторов данной группы”

Работа алгоритма:

1. Прежде чем А начнёт групповую рассылку, должно быть построено соединяющее дерево минимальной стоимости от А до всех членов группы (как строить такие деревья мы уже знаем - например Дейкстрой)
2. Теперь все роутеры знают маршрут минимальной стоимости до А
3. А может использовать этот факт, чтобы доставлять пакеты всем кому надо. Для этого воспользуемся режимом вещания с некоторым ограничением:
4. А посылает пакет с групповым адресом и своим адресом
5. Каждый маршрутизатор, получивший такой пакет, рассылает его по всем интерфейсам кроме того, по которому находится маршрут минимальной стоимости от маршрутизатора до А
6. Пакеты, пришедшие с направления, не совпадающего с направлением маршрута минимальной стоимости до А, сбрасываются
7. Чтобы исключить кого-то, отправляются сообщение обрезки
8. Сообщение посылает хост или маршрутизатор, который понял, что трансляция не для него.
9. Сообщение можно трактовать как “я не хочу получать пакеты этой трансляции”
10. Сообщение посылается А, адрес известен, так как в пакете о трансляции содержится адрес А
11. Все через кого прошло такое сообщение, будут исключать маршрут до его отправителя из таблиц групповой маршрутизации

Что позволяет получить этот несложный алгоритм:

* Каждому хосту пакеты доставляют без циклов
* Маршрутизаторы, через которые нет путей к нужным маршрутизаторам и хостам, шлют сообщения обрезки
* Результирующее дерево - дерево соединений минимальной стоимости от источника к группе интересующих хостов

Если источников вещания несколько, как быть?

* Деревья минимальной стоимости для разных источников вещания могут получиться различными
* Можно запоминать свое дерево для каждой трансляции
* Для небольших групп подойдёт другой вариант:
* Выбираем роутер-рандеву - роутер, который будет отвечать за рассылку трансляций (для рисунка выше подойдет R5, например)
* Все источники потока шлют пакеты выбранному роутеру
* А он уже рассылает данные конечным получателям

В реальности используются следующие **протоколы**:

* IGMP (Internet Group Management Protocol)
* Действует между хостом и непосредственно подсоединенным роутером на уровне L2
* Хосту не надо строить с-дерево: он лишь сообщает специальному роутеру, к какой группе он хотел бы подключиться
* Хосты подписываются на пакеты определенной группы
* Маршрутизаторы периодически опрашивают хосты, к каким группам они хотели бы быть подключенными
* Если отклика нет, то членство в группе прекращается по timeout
* Клиенты одного цвета подписываются на одну и ту же группу. Каждая группа характеризуется фиксированным IP сервиса класса D, на который подписываются. Этому IP соответствует МАС. Multicast MAC адресов может быть несколько на одном сетевом интерфейсе. Подписка в двух режимах: по приглашению и по необходимости По запросу – все кому нужно выбирают случайный интервал и шлют запрос на подписку. Остальные претенденты на эту группу слышат запрос и запоминают MAC на который пойдет вещание.
* Отключение – при отключении клиенты/отписки маршрутизатор опрашивает тех кто остался. Когда никого не осталось – pruning сообщение.
* DVMRP (Distance Vector Multicast Routing Protocol)
* Первый протокол групповой маршрутизации в Интернете
* Используется внутри АС
* Использует RPF с обрезками (примерно как описано выше)
* PIM (Protocol Independent Multicast). Имеет два режима работы:
* Dense (DM)
  + Используется для небольших групп с высокой плотностью получателей
  + Трафик передается на ближайший маршрутизатор, и этот маршрутизатор рассылает копии этого трафика на все активные интерфейсы, которые смотрят на маршрутизаторы, поддерживающие групповую маршрутизацию. Остальные маршрутизаторы также будут копировать трафик в следующие устройства. Таким образом все эти маршрутизаторы получат этот трафик. Если обнаруживается, что один из маршрутизаторов не имеет потребителей, которым необходим этот трафик, то ветка с этим устройством обрезается (prune).
  + Почти эквивалентен DVMRP
* Sparse (SM)
  + Используются роутеры-рандеву (Rendezvous Point - RP). Все маршрутизаторы в multicast-домене маршрутизации должны знать,кто у них является RP.
  + От каждого такого роутера строится однонаправленное дерево с корнем в RP для каждой мультикастинг-группы.
  + Если одно из устройств хочет получать multicast-трафик, то оно отправляет запрос к ближайшему маршрутизатору, который в свою очередь обращается за этим трафиком к RP. RP должен знать, где находятся источники этого трафика.
  + Подходит для сетей с произвольной разреженностью получателей и ограниченной пропускной способностью каналов
  + Получатели сами строят минимальный маршрут до RP, чтобы разгрузить сеть
  + Источник потока соединен с RP и шлет трафик ему

## 

## **17. Маршрутизация на L2, ее отличие от маршрутизации на L3. Протокол STP, алгоритм построения ST дерева коммутатором.**

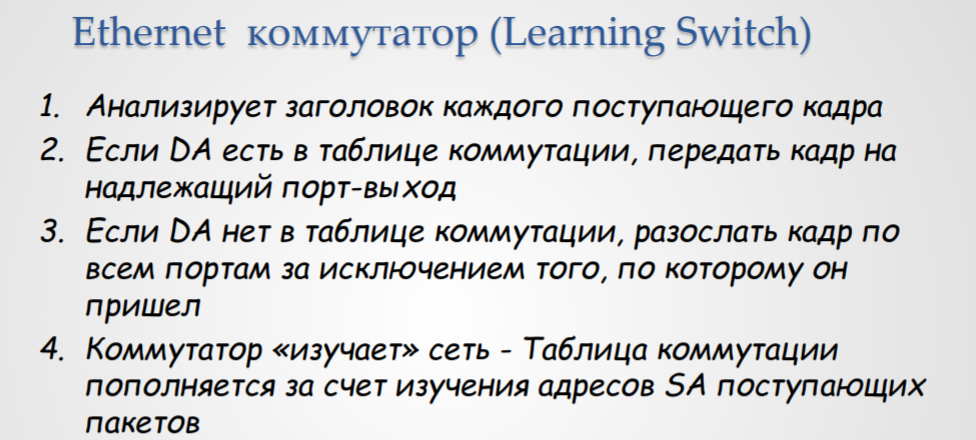
L2 может быть не просто физ. линией. L2 может обладать достаточно сложной структурой, работающая по определенным протоколам, правилам. Примером такой системы может быть Ethernet – СПД с множественным доступом. Этот протокол используют Ethernet коммутаторы.

Множественный доступ – это доступ, когда все, подсоединенные к одному и тому же сегменту, имеют право доступа для передачи данных.

**Маршрутизация** на L2 никак не связана с маршрутизацией на L3.

Основное различие между уровнем 2 и уровнем 3 заключается в функции маршрутизации. Коммутатор уровня 2 работает только с MAC-адресами и не заботится об IP-адресе или любых элементах более высоких уровней. Коммутатор уровня 3 или многослойный коммутатор может делать всю работу, что коммутатор уровня 2. Кроме того, он может выполнять статическую и динамическую маршрутизацию. Это означает, что коммутатор уровня 3 имеет таблицу MAC-адресов и таблицу IP-маршрутизации, а также обрабатывает связь внутри VLAN и маршрутизацию пакетов между различными VLAN. Помимо маршрутизации пакетов, коммутаторы уровня 3 также включают в себя некоторые функции, которые требуют способности понимать информацию IP-адреса данных, поступающих в коммутатор, таких как маркировка трафика VLAN на основе IP-адреса вместо ручной настройки порта. Коммутаторы уровня 3 имеют повышенную мощность и безопасность в соответствии с требованиями.

Когда задерживаются между коммутаторами уровня 2 и уровня 3, вы должны подумать о том, где он будет использоваться. Если у вас есть чистый домен уровня 2, вы можете просто перейти на коммутатор уровня 2. Чистый домен уровня 2 - это место, где подключены хосты, поэтому коммутатор уровня 2 будет работать там нормально. Обычно это называется уровнем доступа в топологии сети. Если вам нужен коммутатор для агрегирования нескольких коммутаторов доступа и маршрутизации внутри VLAN, тогда необходим коммутатор уровня 3. Это известно как уровень распространения в топологии сети.



Вся эта последовательность действий верна в предположении что пакеты не попадают в циклы.

Что они поступают вдоль с-дерева.

**Предотвращение зацикливаний - STP** (протокол соединяющего дерева - spanning tree protocol)

* Топология коммутаторов – граф
* Протокол STP находит подграф, в котором все вершины соединены без циклов

Как работает протокол:

* После включения коммутаторов в сеть, по умолчанию каждый коммутатор считает себя корневым (root).
* Каждый коммутатор начинает посылать по всем портам конфигурационные Hello BPDU пакеты раз в 2 секунды. (BPDU (Bridge PDU)– ID отправителя, ID корня, расстояние от отправителя до корня)). Изначально все считают себя корнем (расстояние = 0).
* Если коммутатор получает BPDU с идентификатором Bridge ID меньшим, чем свой собственный, он прекращает генерировать свои BPDU и начинает ретранслировать BPDU с этим идентификатором. Таким образом в конце концов в этой сети Ethernet остаётся только один коммутатор, который продолжает генерировать и передавать собственные BPDU. Он и становится корневым (root bridge).
* Остальные коммутаторы ретранслируют BPDU корневого, добавляя в них собственный идентификатор и увеличивая счетчик пути (path cost).
* Для каждого сегмента сети, к которому присоединены два и более портов коммутаторов, происходит определение root port, потом designated port — порта, через который BPDU, приходящие от корневого коммутатора, попадают в этот сегмент.
* После этого все порты в сегментах, к которым присоединены 2 и более портов коммутаторов, блокируются за исключением root port и designated port.
* Корневой хост продолжает посылать свои Hello BPDU раз в 2 секунды.

## **18. Представление о коммутации по меткам – MPLS протокол. Понятие VPN – Virtual Private Network.**

**MPLS** (MultiProtocol Label Switching) — технология быстрой коммутации пакетов в транспортной среде мультипротокольных сетей, основанная на использовании меток.

**Цель:** высокоскоростные IP (и не только) магистрали над произвольной СПД.

**Требования** к такой системе:

* Высокая пропускная способность
* Малая задержка
* Хорошая масштабируемость
* Доступ к интегрированным сервисам сети
* Организация виртуальных частных систем (VPN)

**Архитектура:**

В основе лежит принцип обмена меток. Любой пакет несет метку **класса эквивалентности   
при коммутации** (FEC)**.** Значение метки должно быть уникальным только для пути между соседними узлами MLPS, которые также называют **LSR - маршрутизаторами, коммутирующими по меткам**. Метка передается в составе каждого пакета.

Маршрутизатор с помощью алгоритмов OSPF, BGP и IS-IS получает топологию сети. Затем он взаимодействует со смежными коммутаторами, распределяя метки. Обмен метками может быть произведен как с помощью алгоритма **LDP**, так и модифицированными версиями других протоколов.

В итоге получим множество путей внутри MPLS - LSP. Каждый маршрутизатор имеет таблицу - соответствие пары (входной интерфейс, входная метка) вектору <префикс адреса получателя, выходной интерфейс, выходная метка>. LSR, получив пакет, по номеру входного интерфейса и метке определяет выходной интерфейс, заменяет старую метку новой, и пакет отправляется по новому пути.

Это эффективно, так как нужно всего лишь одно сопоставление - входной метки и выходной, а не поиск best matching адреса в традиционной маршрутизации.

MPLS делится на две части:

* **Ядро** — устройства, которые должны просто поддерживать MPLS и осуществлять маршрутизацию трафика. Они занимаются только коммутацией
* **Граничная часть** — устройства которые занимаются классификацией пакетов по FEC, реализуют доп сервисы, такие как фильтрация, явная маршрутизация, выравнивание нагрузкой и управление трафиком

Главная особенность MPLS — отделение процесса коммутации от процесса анализа IP-адресов в его заголовке.

**Плюсы:**

* Можно легко реализовать разные приоритеты обслуживания,
* Легко поддерживается качество обслуживания,
* Легко контролировать ресурсы и изолировать трафик отдельных пользователей.

**Метки**

Метка — короткий идентификатор, определяющий класс. Перед включением в состав пакета метка определенным образом кодируется. Например в случае IP она помещается в специальный “тонкий” заголовок пакета, инкапсулирующего IP. В других случаях метка пишется в заголовок протокола канального уровня или в виде значения в заголовке уровня ATM. Метка в IPv6 может размещаться в поле идентификатора потока

**Стек меток**

В MPLS можно передавать больше одной метки. Операции добавления/изъятия метки подобны операциям на стеке push и pop. Результат коммутации определяет лишь верхняя метка стека. Таким образом можно выстраивать иерархию потоков в MPLS и организовывать туннельные передачи

**Привязка и распределение меток**

Привязка - соответствие класса FEC и значение метки для данного сегмента LSP. Обмен информацией о привязке меток осуществляется между соседними LSR с помощью протокола распространения меток.

**Есть два режима распределения меток:**

* Независимый — можно уведомить верхний узел о привязке данного класса от нижнего узла.
* Упорядоченный — можно выслать такое уведомление только после получения указанной информации от нижнего узла, кроме случая, когда маршрутизатор LSR является выходным для этого класса FEC

**Построение маршрута**

Рассмотрим протокол LDP:

1. Определение “соседей” с помощью многоадресной рассылки LDP-сообщений
2. Открытие TCP соединений на транспортном уровне между участниками сеанса
3. Передача запросов на установку привязки (меток) и сама привязка
4. Периодически с помощью тестовых сообщений проверяется работоспособность друг друга

**Лектор о VPN:**

“Сейчас достаточно часто используют в офисах и бизнесе это то, что называют **Virtual Private Network**. С Virtual Private Network вы открываете безопасное соединение к сети, которой вы доверяете, такая как у вас в офисе, например. Когда вы выходите в Интернет и шлете IP пакеты, они идут не как обычно, а в рамках этого VPN соединения. Таким образом все пакеты идут в офисную сеть. Это позволяет обеспечивать безопасный доступ и защищать сетевые ресурсы своего офиса. Вместо того, чтобы обеспечивать защиту по всей сети, достаточно обеспечить защиту в точке доступа через VPN. Безопасный доступ осуществляется через шлюз, компьютер, запрашивающий доступ через VPN, получает доступ и посылает свои пакеты в офисную сеть. Но как это происходит? Предположим, что получаю доступ к веб сайту компании. Мой браузер строит HTTP GET. Как всегда это инкапсулируется в TCP и в IP, который предназначен для внутреннего веб-сервера компании. Но прежде чем инкапсулировать IP пакет в сетевой кадр (link layer frame) (Я не могу напрямую взаимодействовать с внутренним веб-сервером) мой компьютер положит IP пакет внутрь TLS (Transport Layer Security) сегмента. TLS защищает содержимое пакета и держит его в секрете. TLS сессия находится внутри TCP потока, который ограничивается шлюзом VPN. Таким образом TCP находится внутри IP пакета, предназначенного для шлюза VPN. Выглядит это так: HTTP внутри TCP, внутри IP, внутри TLS, внутри TCP, внутри IP, внутри Ethernet”

## **19. Теоретические основы передачи данных (ограничения на пропускную способность передачи сигналов, взаимосвязь пропускной способности канала и ширины его полосы пропускания). Среды передачи (магнитные носители, витая пара, среднеполосный и широкополосный кабели, оптоволокно, сравнение кабелей и оптоволокна).**

**Данные** - это то, с помощью чего мы описываем явления или объекты.

**Передача** - процесс взаимодействия передатчика и приемника в целях получения приемником сигналов от передатчика.

**Сигнал** - представление данных. Сигнал можно рассматривать в двух видах:

1. Как функцию во времени смотрим, например, как изменяется со временем напряжение, сила тока сигнала
2. Как функцию частоты

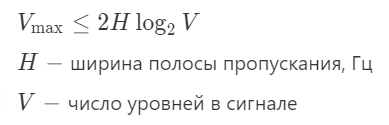
**Полоса пропускания канал** - спектр частот, которые наша физическая среда (провода, воздух и т.п) пропускает без существенного понижения мощности сигнала.

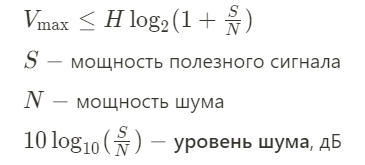
* “**Существенное понижение**” определяется в каждом конкретном случае по-своему. Но обычно это потеря 50% мощности
* **Чем больше полоса пропускания, тем больше наши потери**. Поэтому любая физическая среда, приводя к потерям, накладывает ограничения на частоту гармоник
* **Чем больше полоса пропускания, тем выше скорость передачи данных**

**Пропускная способность** - максимальная скорость, с которой канал способен передавать сигнал

* Сигналы не могут передаваться без потери энергии
* Чем шире полоса пропускания, тем больше потери
* Ширина полосы пропускания ограничивает то, какие гармоники возможно использовать
* Чем больше набор гармоник, тем больший набор функций мы можем довольно точно аппроксимировать

**Формула Найквиста-Шеннона-Котельникова (не учитывает шум)**Устанавливает взаимосвязь между пропускной способностью канала и шириной его полосы пропускания



**Формула Шеннона-Хартли (учитывает шум)**  


**Среды пере(с)дачи**

**Назначение физического уровня** - передавать данные в виде потока битов от одной машины к другой. При этом для передачи данных можно использовать различные физические среды.

**Некоторые важные характеристики физических сред:** ширина полосы пропускания, пропускная способность, задержка сигнала, стоимость, сложность прокладки, сложность обслуживания, затухание, помехоустойчивость, достоверность передачи данных и т.д.

**Магнитные носители**  
Берем магнитный диск (жесткий диск, ленту или барабан) и физически переносим его в пункт назначения.

**Витая пара**  
**Структура:** 2 медных изолированных провода, один из которых обвивает второй. Вьющийся провод нужен для устранения помех, возникающих от соседних витых пар  
Пример использования: широко используется в телефонии. Длина линии может при этом составлять до нескольких километров (между абонентом и автоматизированной телефонной станцией). В России к примеру средняя длина в городских условиях равна 3.5 километрам  
***Важно:*** *витая пара передает как цифровые, так и аналоговые сигналы*  
**Скорость:** зависит от длины кабеля. На коротких расстояниях (до сотни метров) можно достичь скорости до 1 Гбит/с, на больших расстояниях (несколько км) скорость не превышает 4 Мбит/с  
Еще пример использования: витая пара стоит довольно дешево. И поэтому ее любят применять при создании ЛВС (локальные вычислительные системы (в рамках одной комнаты, здания или нескольких близко расположенных зданий))  
**Объединение:** витые пары обычно объединяют в многопарные кабели

* Кабель категории 3: 4 витые пары, полоса пропускания до 16 МГц
* Кабель категории 5: 4 витые пары (но с более плотной навивкой), полоса пропускания до 100 МГц

**Коаксиальный кабель**  
**Структура:** по центру толстый медный проводник, окруженный изолятором, после чего вся эта конструкция помещается внутрь второго проводника (обычно это плетеная металлическая сетка, выполняющая роль экрана) и покрывается пластиком  
Полоса пропускания: от 1 до 500 МГц  
***Важно:*** *передает как цифровые, так и аналоговые сигналы***Примеры использования:**

* 1. кабельное телевидение
  2. передача нескольких телефонных разговоров одновременно
  3. подключение компьютерных периферийных устройств на небольших расстояниях

**Проблемы:** основные ограничения скорости и расстояния передачи без усиления - это затухание, тепловой и интермодуляционный шум.

**Оптоволокно**  
**Структура:**

1. Центральный осевой элемент - для жесткости и чтобы провод не ломался
2. Самое важное - **оптические волокна**, передающие сигнал. В кабеле их обычно от 2 до 144. Оптические волокна бывают разной толщины. Поговорим про них чуть подробнее:

* **Одномодовый шнур** (малая толщина, 8-10 мкм) - силиконовый шнур, имеющий толщину, близкую к длине волны источника света. Зачем так мало? Чтобы не было потерь на внутреннее отражение. По такому шнуру можно передавать данные со скоростью несколько гигабит в секунду на сотню километров без промежуточного усиления
* **Многомодовый шнур** (чуть потолще, 50-100 мкм): такой шнур проводит несколько лучей разной длины

1. Все остальные элементы (3-7) играют роль разного рода защиты (от влаги, от трения и т.п.), так как оптические волокна - вещь крайне хрупкая

**Длины волн:** на практике используются волны длиной 0.85, 1.3 и 1.55 мкм

* 1.3 и 1.55 мкм: их затухание < 5% на километр (хорошее свойство)
* 0.85 мкм: имеют бОльшее затухание, но они лучше соответствуют возможностям лазерных источников

**Ширина полосы пропускания:** от 25000 до 30000 ГГц

**Специальная форма:** чтобы не было потерь на дисперсию (потеря по мере распространения сигналом начальных форм и размеров) сигналам придают специальную форму (оставим подробности экспертам, нам они не важны). Сигналы такой специальной формы называются **солитонами.**

**Источники сигнала:** есть 2 источника света - это **светодиод** (LED) и **полупроводниковый лазер**, у которых разные свойства, но если коротко, то LED похуже будет.

**О точках соединения:** разные кабеля соединяют с помощью специальных коннекторов, механически прижимая один край к другому, либо сваркой. При этом в точке соединения теряется от 5 до 20% мощности сигнала

**Сравнение оптоволокна с обычными кабелями**:

1. Ширина полосы пропускания оптоволокна > ШПП других кабелей
2. Оптоволокно компактнее, тоньше и имеет меньшую массу (1 км 1000-парника весит 8т, 1 км оптоволокна той же пропускной способности - 100кг)
3. Затухание сигнала в оптоволокне существенно меньше, чем в коаксиальном кабеле и витой паре и остается постоянным для широкого диапазона частот
4. Оптоволокно невосприимчиво к внешним электромагнитным излучениям и не излучает энергию (значит, оно не влияет на работу другого оборудования и его сложнее обнаружить)
5. Чем меньше используется репитеров, тем дешевле система передачи и меньше источников ошибок. С этой позиции оптоволоконные системы достигли большего совершенства. Среднее расстояние между репитерами у них в разы больше, чем у коаксиального кабеля и витой пары
6. Оптоволокно сложнее монтировать, нужны специально подготовленные инженеры

## **20. Теоретические основы передачи данных (ограничения на пропускную способность передачи сигналов, взаимосвязь пропускной способности канала и ширины его полосы пропускания). Передача аналоговых данных по цифровым сигналам. Передача цифровых данных по цифровым сигналам.**

***Теоретические основы смотрите в 24 билете***

Преобразование аналоговых данных в цифровой сигнал можно представить как преобразование аналоговых данных в цифровую форму. Этот процесс называют оцифровкой данных. Выполнив его, мы можем передать цифровые данные цифровым или аналоговым сигналом. Устройство АЦП (Аналогово-Цифровой Преобразователь) преобразует аналоговые данные в цифровую форму, а устройство ЦАП (Цифро-Аналоговый преобразователь) выполняет обратную процедуру. Устройство, объединяющее в себе функции и АЦП и ЦАП, называют кодеком (кодер-декодер).

**Импульсно-кодовая модуляция** (ИКМ) основана на следствии из теоремы Найквиста, которое утверждает, что если изменять параметры сигнала f(t) через регулярные интервалы времени с частотой не меньше, чем удвоенная частота самой высокочастотной составляющей сигнала, то полученная серия измерений будет содержать всю информацию об исходном сигнале и этот сигнал может быть восстановлен.

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

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

У метода Дельта модуляции есть два параметра: величина шага 𝛿 и частота замеров или шаг квантования. Выбор шага 𝛿 – это баланс между ошибкой квантования и ошибкой перегрузки по крутизне.

Когда исходный сигнал изменяется достаточно медленно, то возникает только ошибка квантования, чем больше 𝛿, тем больше эта ошибка. Если же сигнал изменяется резко, то рост ступенчатой функции может отставать. Это вид ошибки растет с уменьшением 𝛿.

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

***Теоретические основы смотрите в 24 билете***

**Цифровой сигнал** - это дискретная последовательность импульсов по напряжению, каждый из которых имеет ступенчатую форму. Каждый импульс - это единичный сигнал. В общем случае, данные в двоичной форме при передаче кодируются так, что один бит данных отображается в несколько единичных сигналов. В простейшем случае это соответствие имеет однозначный характер: один бит - один сигнал.

**Скорость передачи данных** — это количество бит в секунду, которые передают с помощью сигналов. Эту скорость также называют битовой скоростью.

**Продолжительность или длина бита** - это интервал времени, который нужно передатчику, чтобы испустить надлежащий единичный сигнал. При скорости передачи данных R бит/сек, длина бита равна 1/𝑅.

Прежде всего, приемник должен быть строго настроен на длину бита. Он должен уметь распознавать начало и конец передачи каждого бита. Уметь распознавать уровень сигнала: низкий или высокий. Из-за шума на линии при этом могут возникать ошибки.

Есть три важных фактора влияющие на правильность передачи:

* уровень шума
* скорость передачи данных
* ширина полосы пропускания канала.

Есть еще один фактор, влияющий на передачу данных: это способ представления (кодировки) данных на физическом уровне.

Основными критериями сравнения различных способов кодирования данных на физическом уровне являются:

**Ширина спектра сигнала** (чем меньше высокочастотных составляющих в сигнале, тем уже ширина полосы пропускания может быть при передаче), **Синхронизация между приемником и передатчиком** (приемник должен точно определять начало и конец битового интервала), **Обнаружение ошибок**, **Чувствительность к шуму**, **Стоимость и скорость.**

**Потенциальный код NRZ**: 0 – высокий потенциал. 1 – низкий потенциал.

**Биполярный код NRZI**: 0 – нет перепада уровня сигнала в начале битного интервала. 1 – перепад уровня сигнала в начале интервала.

**Биполярный код AMI**: 0 – отсутствие сигнала. 1 – положительный или отрицательный потенциал, обратный по отношению к потенциалу в предыдущий период.

**Манчестерский код**: 0 – переход с высокого на низкий потенциал в середине интервала. 1 – переход с низкого на высокий потенциал в середине интервала.

**Потенциальный код 2B1Q**: Использует 4 уровня сигналов, значение уровня определяется значением пары битов данных.

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

**Потенциальный NRZ код.** Основным недостатком этого кода является отсутствие синхронизации. На длинных последовательностях нулей или единиц потенциал на линии не меняется, и может произойти рассинхронизация между приемником и передатчиком, что приведет к ошибкам.

**Биполярный код AMI.** У этого метода есть несколько существенных преимуществ по сравнению с NRZ кодами. **Во-первых**, в случае длительной последовательности 1 рассинхронизации не происходит. Каждая единица сопровождается изменение потенциала устойчиво распознаваемом приемником. Поскольку каждая единица сопровождается изменением потенциала, то не возникнет постоянной составляющей. Однако длинная последовательность 0 остается проблемой, и требуются дополнительные усилия, которые позволили бы избежать ее появления. **Во-вторых**, спектр сигнала здесь уже, чем у NRZ кодов. **И, наконец**, четко определенное правило чередования уровней позволяет обнаруживать единичные ошибки.

## **21. Теоретические основы передачи данных (ограничения на пропускную способность передачи сигналов, взаимосвязь пропускной способности канала и ширины его полосы пропускания). Передача аналоговых данных по аналоговым сигналам. Передача цифровых данных по аналоговым сигналам.**

***Теоретические основы смотрите в 24 билете***

**Модуляция**, т.е. объединение исходного сигнала m(t) и несущей частоты 𝑓𝑐, позволяет нужным образом изменять параметры исходного сигнала и тем самым упростить решение ряда технических проблем. Кроме этого, модуляция позволяет использовать методы мультиплексирования.

**Три способа модуляции:** амплитудная модуляция, частотная и фазовая.

При **амплитудной модуляции** форма результирующего сигнала определяется формулой:

𝑆(𝑡) = [1 + 𝑥(𝑡)]𝑐𝑜𝑠(2𝜋𝑡), где – частота несущей, – индекс модуляции, который определяют как отношение амплитуды исходного сигнала к амплитуде несущего сигнала.

Форма результирующего сигнала при **частотной модуляции** определяется следующим выражением: 𝑆(𝑡) = 𝑐𝑜𝑠(2𝜋𝑡 + 𝑚(x)dx), где – индекс частотной модуляции.

Сигнал, получаемый **фазовой модуляцией**, определяет соотношение: 𝑆(𝑡) = 𝑐𝑜𝑠(2𝜋𝑡 + 𝑚(t)), где – индекс частотной модуляции.

Хотя все эти три вида модуляции порождают сигнал S(t), спектр которого симметричен относительно , но в случае амплитудной модуляции он проще по составу. В случае частотной и фазовой модуляций требуется, в общем случае, более широкая полоса пропускания.

Широко распространенным случаем аналоговой модуляции является метод квадратичной амплитудной модуляции QAM. Метод QAM – это комбинация амплитудной и фазовой модуляций. Идея этого метода состоит в том, что можно по одной и той же линии послать одновременно два разных сигнала с одинаковой несущей частотой, но сдвинутых по фазе друг относительно друга на . Каждый сигнал генерируется методом амплитудной модуляции. ***Теоретические основы смотрите в 24 билете***

Примером передачи данных в цифровой форме с помощью аналоговых сигналов является использование телефонных сетей для передачи цифровых данных.

**Модем** (МОдулятор–ДЕМодулятор) – прибор, который преобразует цифровой сигнал в аналоговый и наоборот в надлежащем диапазоне частот.

Аналоговая модуляция заключается в преобразовании одного или нескольких параметров из трех основных параметров несущего сигнала: амплитуды, частоты и фазы.

Есть три основных метода модуляции для преобразования цифровых данных в аналоговую форму:

* амплитудная модуляция,
* частотная модуляция,
* фазовая модуляция.

В случае **амплитудной модуляции** двоичные 0 и 1 представлены аналоговым сигналом на частоте несущей, но разной амплитуды. Обычно 0 соответствует сигнал с нулевой амплитудой.

Таким образом, при амплитудной модуляции сигнал S(t) имеет вид 𝑆(𝑡) =, где - несущий сигнал с амплитудой A. Метод амплитудной модуляции не очень эффективен по сравнению с другими методами, т.к. он очень чувствителен к шумам.

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

**Фазовая модуляция** состоит в представлении цифровых данных сдвигом фазы несущего сигнала.

Эффективность использования полосы пропускания можно существенно повысить, если единичный сигнал будет кодировать несколько бит. В общем случае скорость модуляции 𝐷 = 𝑅/𝑏 = 𝑅/𝐿 , где D – скорость модуляции (сигнальная скорость), R – битовая скорость (скорость передачи данных), L – число разных уровней единичных сигналов, b – число бит на единичный сигнал.

## **22. Физические среды передачи данных. Беспроводная связь (электромагнитный спектр, радиопередача, микроволновая передача, видимое излучение). Протоколы МАСА.**

**Беспроводная связь** востребована для мобильных вычислительных средств и там, где прокладка

любого кабеля затруднительна либо невозможна (горы, старые здания), либо если требуется быстрое создание коммуникации.

**Электромагнитный спектр**

При определенных условиях волны распространяются в строго определенном направлении. В этом

случае антенна приемника должна быть должным образом ориентирована в пространстве по отношению

к антенне передатчика, чтобы принимать сигналы. При других условиях антенна передатчика распространяет электромагнитные волны во всех направлениях.

Для передачи информации из всего спектра частот используют только следующие диапазоны: радиодиапазон, микроволновый, инфракрасный, видимый и частично ультрафиолетовый.

Количество данных, передаваемых электромагнитной волной, определяется ее шириной, т.е. спектром частот гармоник, составляющих эту волну. При определенных условиях на низких частотах можно

закодировать несколько бит данных на 1 Гц, но на высоких частотах это число можно довести до 40

бит.

Задав некоторую полосу длин волн, получим полосу частот, откуда затем найдем скорость передачи

для этой полосы частот. Чем шире полоса частот, тем выше битовая скорость, что следует из формулы,

связывающей ширину полосы пропускания и битовую скорость передачи. (Теорема Шеннона – Хартли)

, где C – пропускная способность канала, бит/с; B – полоса пропускания канала, Гц;   
S – полная мощность сигнала над полосой пропускания, Вт или В²; N – полная шумовая мощность над полосой пропускания, Вт или В²; S / N – отношение мощности сигнала к шуму (SNR)

**Радиопередача**

Радиоволны распространяются на большие расстояния, легко преодолевая преграды. Поскольку радиоволны распространяются во всех направлениях, то принимающая и передающая антенны не требуют дополнительной настройки и регулирования взаимного расположения.

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

до источника. На высоких частотах радиоволны распространяются по прямой, но хуже преодолевают

препятствия.

На любых частотах радиоволны чувствительны к помехам от электрических устройств. В силу перечисленных причин лицензирование, т.е. право использования частот в радиодиапазоне, находится под

жестким контролем государства.

Длинные и средние волны могут огибать поверхность Земли и распространяться на большие расстояния. Короткие полны хотя и поглощаются земной поверхностью, но за счет отражения от ионосферы также могут распространяться на большие расстояния.

**Микроволновая передача**

Частоты свыше 10 МГц представляют собой область микроволнового диапазона. Волны в этом диапазоне распространяются в строго определенном направлении и могут быть сфокусированы с помощью параболической антенны, имеющей вид телевизионной тарелки. Однако приемная и передающая антенны при этом должны быть тщательно ориентированы в пространстве по отношению друг к другу. Такая направленность позволяет, построив цепочку ретрансляторов, передавать сигнал на большие расстояния.

Микроволны не проходят сквозь здания так же хорошо, как низкочастотные волны.

**Видимое излучение**

Видимый диапазон также используется для передачи сигналов. Обычно источником света в этом случае является лазер. Монохромное когерентное излучение легко фокусируется. Однако смог, загрязнение атмосферы, дождь или туман портят дело. Передачу такого излучения способны нарушить даже конвекционные потоки воздуха на крыше, возникающие в жаркий день, которые вызывают дрожание луча вокруг приемника, ухудшая качество передачи.

**МАСА** (**Multiple Access with Collision Avoidance** – множественный доступ с предотвращением столкновений). Идея, лежащая в основе этого протокола, заключается в том, что отправитель заставляет получателя передать короткий кадр, чтобы окружающие станции могли услышать эту передачу и воздержаться от действий на время, требуемое для приема большого информационного кадра.

## **23. Семейство протоколов IEEE 802.11. Система передачи данных WiFi: принципы организации, структура кадра, алгоритм функционирования.**

**IEEE 802.11** – набор стандартов связи для коммуникации в беспроводной локальной сетевой зоне частотных диапазонов 0,9; 2,4; 3,6; 5 и 60 ГГц.

**Wi-Fi (Wireless Fidelity)** – технология беспроводной локальной сети с устройствами на основе стандартов IEEE 802.11

**Физический уровень.**

Первой и ключевой технологией стандарта 802.11 является технология расширения спектра передачи методом прямой последовательности (Direct Sequence Spread Spectrum – DSSS). Использование DSSS позволяет беспроводным интерфейсам передавать данные со скоростью от

1 до 2 Мбит/c.

**Канальный уровень.**

Минимально сеть wifi может содержать всего два устройства. В этом случае

организуется выделенная есть, в которую входят все беспроводные интерфейсы этих устройств. Данное

соединение можно сравнить с соединением типа точка-точка в проводной связи.

Для решения задачи совместимости в сети устанавливается выделенный узел – точка доступа, представляющая собой устройство, имеющее проводной интерфейс для подключения к проводной сети, а

также антенну, образующую вокруг себя зону покрытия точки доступа. Радиус покрытия точки доступа

от 90 до 150 м.

Когда абонентское устройство включается внутри сети, оно начинает прослушивать эфир в поисках

совместимого устройства, с которым оно могло бы взаимодействовать. Этот этап называется сканированием. При активном сканировании генерируется широковещательный запрос от абонентского устройства, обязательно включающий в себя идентификатор сети (SSID), к которой он хочет присоединиться.

Когда запрос достигает точки доступа, имеющий запрашиваемый идентификатор, эта сеть генерирует ответ на запрос. При пассивном сканировании абонентское устройство слушает эфир и ожидает появления кадров-маяков, которые периодически рассылаются точками доступа. Когда абонентское устройство

получает кадр-маяк, в котором указан SSID, оно пытается присоединиться к указанной сети. Пассивное

сканирование – постоянный процесс, при котором устройства могут присоединиться к точке доступа или

отсоединяться по мере изменения мощности радиосигнала.

**Алгоритм передачи в IEEE 802.11**

1. При потребности передачи выбираем случайным образом номер из 0-31 – количество слотов ожидания(W)

2. Слушаем эфир

A. если эфир занят - идет передача

a) пропустили Duration того кто передает, то ждем конца передачи и к шагу B по сформированному W

b) поймали Duration, то W = W + Duration (услышанный) + ack + 1 slot переход к шагу B

B. Countdown W: обратный отсчет W до 0

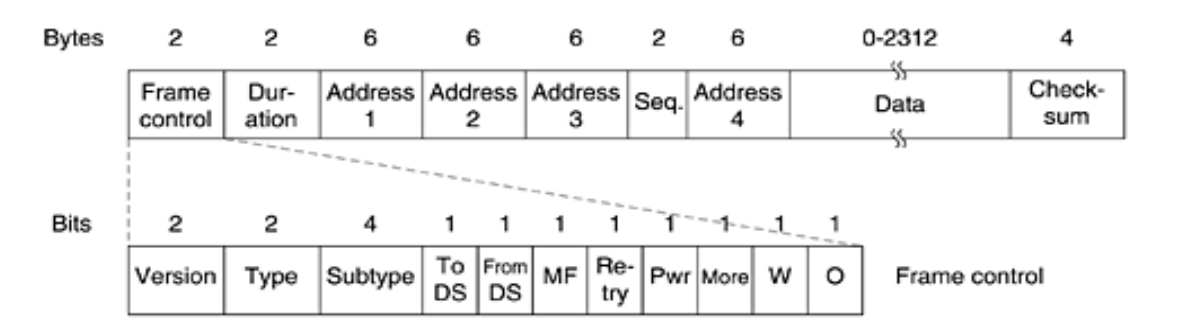
С. если эфир свободен, то передаем и ждем ack

2.1. если асk не пришел, то коллизия и переход к 1

2.2. если ack пришел, то кадр передан и переход к 1

3. Если W достиг 0, но в этот момент кто-то начал передачу, то либо ждем, пока он не закончит, либо формируем W как в b) и обратный отсчет.

**Структура кадра WiFi**



**Кадр WiFi:**

**Поле “Frame Control” (является составным):**

* **Поле «Version»** - версия протокола, в данном стандарте – 0.
* **Поле «Туре»** - тип фрейма. В рамках стандарта определено три типа фреймов:
  + **Management (00)** – фреймы для передачи служебной информации (Beacon, Probe Request, Authentication и т.д.);
  + **Control (01)** – используются для контроля доступа к среде передачи, например RTS, CTS, АСК;
  + **Data (02)** – служат для передачи полезной информации. Каждый тип фрейма делится на подтипы, определяемые значением поля «Subtype».
* **Бит «То DS»** установлен в единицу, если фрейм адресован точке доступа для передачи его в обычную сеть (с точки зрения стандарта – DS) или другому абоненту из данного BSS. Бит «From DS» установлен в единицу, соответственно, если фрейм направлен из DS.
* **Бит «More Frags»** установлен в единицу, если фрейм разбит на фрагменты и данный фрагмент не последний.
* **Поле «Retry»** указывает на то, что данный фрейм – повторная передача предыдущего фрейма, что позволяет принимающей станции распознавать повторяющиеся фреймы, возникающие из-за потери подтверждений.
* **Бит «Power»** означает, что после передачи данного фрейма станция переходит в режим энергосбережения из активного режима или наоборот.
* **Бит «More Data»** используется точкой доступа для того, чтобы сообщить станции, что для нее имеются данные (в буфере в точке доступа).
* **Бит «WEP»** указывает на то, что фрейм зашифрован по протоколу WEP.
* **Бит «Order»** указывает на то, что фрейм не нуждается в обеспечении QoS, т.е. порядок следования этого фрейма в потоке не важен (его можно обгонять).

**Duration** - количество временных слотов для передачи данных на скорости абонента

**МАС1** – отправитель

**МАС2** – Точка доступа 1 (отправителя)

**МАС3** – Точка доступа 2 (получателя)

**МАС4** – получатель

**seq** – последовательный номер кадра (защита от повторов)

**Checksum** – защищает и заголовок и данные

## **31. Принципы организации и функционирования семейства протоколов IEEE 802.3: математическая модель и оценка производительности.**

Стандарт **IEEE 802.3 (Ethernet)** – среда с множественным доступом типа CSMA/CD

**CSMA/CD – Carrier Sense Multiple Access with Collision Detection**, т.е. протокол множественного доступа с контролем несущей и обнаружением коллизий. Протоколы этого класса широко используется в локальных сетях. В их основе лежит следующая модель работы. В момент t0 станция заканчивает передачу очередного кадра. Все станции, у которых есть кадр для передачи, слушают шину и, если там только несущая, то начинают передачу. За время тау заголовок дойдет до самой удаленной станции и если она также начнет в этот момент передавать, то станция, начавшая передавать первой, увидит, что на шине мощность сигнала выше, чем та с которой она передает. Это для нее означает коллизию. Она останавливает передачу и генерирует 48 разрядный код оповещающий всех о наступлении коллизии. Обнаружив коллизию, станция сразу прекращает передачу на случайный интервал времени и генерирует специальный сигнал об обнаружении коллизии, после чего все начинается сначала. Таким образом, в работе протокола CSMA/CD можно выделить три стадии: состязания, передачи и ожидания, когда нет кадров для передачи. Длину этот период состязаний определяет длина заголовка кадра. Она должна быть такой, чтобы занять весь сегмент (домен коллизий).

**IEEE 802.3: алгоритм задержки**

* При возникновении коллизии время разбивается на слоты длиной, соответствующей наибольшему времени распространения сигнала в оба конца.
* Алгоритм двоичной экспоненциальной задержки
* При первой коллизии станции, участвовавшие в ней случайно выбирают 0 или 1 слот для ожидания.
* Если коллизия возникнет опять, то выбор происходит среди чисел 0-, где i - порядковый номер очередной коллизии.
* После 10 коллизий число слотов достигает 1023 и далее не увеличивается.
* После 16 коллизий Ethernet контроллер фиксирует ошибку и сообщает о ней машине, т.е. более высокому уровню стека протоколов.

**IEEE 802.3 : оценка числа слотов состязаний**

* Плотная и постоянная нагрузка: есть k станций всегда готовых к передаче.
* При коллизиях в каждом слоте повторная передача с постоянной вероятностью.
* Если каждая станция участвует в каждом слоте состязания с вероятностью p , то вероятность А, что некоторая станция захватит канал в этом слоте, равна при , при
* Средний интервал состязания
* эффективность канала равна , где m - среднее время передачи кадра

Из этой формулы видно, что чем длиннее кабель, т.е. больше , тем хуже эффективность канала, т.к. растёт длительность периода состязаний.

## **24. Проблемы передачи данных на канальном уровне. Сервис, предоставляемый сетевому уровню. Обнаружение и исправление ошибок (Коды исправляющие ошибки, коды обнаруживающие ошибки).**

В разных средах характер ошибок разный. Ошибки могут быть одиночные, а могут возникать группами, сразу несколько. Недостатком групповых ошибок является то, что их труднее обнаруживать и исправлять, чем одиночные.

Для надежной передачи кодов было предложено два основных метода.

Первый – внести избыточность в форме дополнительных битов в передаваемый блок данных так, чтобы, анализируя полученный блок, можно было бы указать, где и какие возникли искажения. Это, так называемые, **коды с исправлением ошибок**.

Второй – внести избыточность, но лишь настолько, чтобы анализируя полученные данные, можно было сказать, есть в переданном блоке ошибки или нет. Это, так называемые, **коды с обнаружением ошибок**.

Пусть данные занимают m разрядов и мы добавляем r разрядов, как контрольные разряды. Нам надо передать слово длины n (n = m + r), которое называют n-битовым кодословом. Количество разных битов в двух кодословах называется **расстоянием Хемминга**. Если мы хотим обнаруживать d ошибок, то надо чтобы кодослова отстояли друг от друга на расстояние d + 1. Если мы хотим исправлять ошибки, то надо, чтобы кодослова отстояли друг от друга на 2d + 1. Простым примером кода с обнаружением одной ошибки является код с битом четности.

Оценим минимальное количество контрольных разрядов, необходимое для исправления одиночных ошибок. Пусть у нас есть код из m бит сообщения и r контрольных бит. Каждое из правильных сообщений имеет n неправильных кодослов на расстоянии 1. Таким образом, с каждым из кодослов связано n + 1 кодослов. Так как общее число кодослов , то (n+1) ≤ , учитывая, что n = m + r, получаем: (𝑚 + 𝑟 + 1) ≤ .

**Коды обнаруживающие ошибки.** Применение техники четности «в лоб» в случае групповых ошибок не даст нужного результата. Однако ее можно скорректировать. Пусть нам надо передать n слов по k бит. Расположим их в виде матрицы n на k. Для каждого из n столбцов вычислим бит четности и разместим их в дополнительной строке. Получившаяся матрица затем передается по строкам. По получению матрица восстанавливается и если нарушен хоть один бит, то весь блок передается повторно. Этот метод позволяет обнаружить одиночный пакет ошибок длины n. Против групповых ошибок длины n+1 он бессилен.

Поэтому на практике применяют другую технику, которая называется полиномиальным кодом или циклическим избыточным кодом (Cyclic Redundancy Code) или CRC кодом. CRC коды построены на рассмотрении битовой строки как строки коэффициентов полинома. k битовая строка – коэффициенты полинома степени k - 1. Самый левый бит строки – коэффициент при старшей степени. Например, строка 110001 представляет полином .

Использование полиномиальных кодов при передаче заключается в следующем. Отправитель и получатель заранее договариваются о конкретном генераторе полиномов G(x), у которого коэффициенты при старшем члене и при младшем члене должны быть равны 1. Пусть степень G(x) равна r. Для вычисления контрольной суммы блока из m бит надо, чтобы обязательно m > r. Идея состоит в том, чтобы добавить контрольную сумму к передаваемому блоку, рассматриваемому как полином М(х) так, чтобы передаваемый блок с контрольной суммой был кратен G(x). Когда получатель получает блок с контрольной суммой, он делит его на G(x). Если есть остаток, то были ошибки при передаче. Алгоритм вычисления контрольной суммы (здесь r степень G(x)):

1. Добавить r нулей в конец блока так, что он теперь соответствует полиному M(x);

2. Разделить по модулю 2 полином M(x) на G(x) и получить остаток;

3. Вычесть по модулю 2 остаток (длина которого всегда не более r разрядов) из строки M(x).

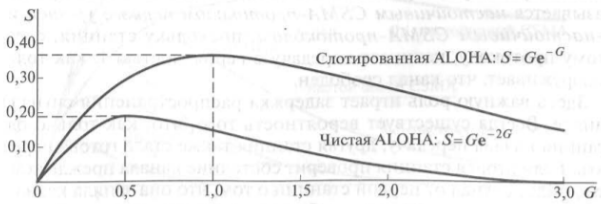
Результат и есть блок с контрольной суммой (назовем его Т(х)). Этот метод позволяет отлавливать одиночные ошибки. Групповые ошибки длины не более r. Нечетное число отдельных ошибок.

Идея обоснования этих утверждений состоит в следующем. Ошибки при передаче означают в терминах полиномов, что мы получим после передачи не Т(х), а Т(х) + E(x). Если степень E(x) меньше степени G(x), то остаток от деления никогда не будет равен 0.

## **25. Математическая модель и оценки числа состязаний в протоколе множественного доступа к каналу (динамическое vs статическое выделение канала). Мат. модель системы ALOHA. Сравнение производительности систем: чистая ALOHA, слотированная ALOHA. Протоколы множественного доступа с обнаружением несущей (настойчивые и не настойчивые CSMA, CSMA с обнаружением коллизий).**

**Чистая ALOHA.** Идея чистой ALOHA проста: любой пользователь, желающий передать сообщение, сразу пытается это сделать. Благодаря тому, что в вещательной среде у него всегда есть обратная связь, т. е. он может определить, пытался ли кто-то еще передавать сообщение на его частоте, отправитель может установить возникновение конфликта при передаче. Отправитель при этом должен «слушать» среду передачи до тех пор, пока последний бит его сообщения не достигнет самого отдаленного получателя. Обнаружив конфликт, отправитель ожидает некоторый случайный отрезок времени, после чего повторяет попытку передачи. Интервал времени на ожидание должен быть случайным, иначе конкуренты, повторяя попытки передачи вызовут коллизию снова. Системы, в которых пользователи конкурируют за получение доступа к общему каналу, называются системами с состязаниями.

**Слотовая ALOHA.** Модификация чистой ALOHA, в которой все время работы канала разделяется на слоты. Размер слота при этом должен быть равен максимальному времени кадра. Такая организация работы канала требует синхронизации. Кто-то, например одна из станций, испускает сигнал начала очередного слота. Поскольку передачу теперь можно начинать не в любой момент, а только по специальному сигналу, то время на обнаружение коллизии сокращается вдвое



S – пропускная способность, G – среднее значение распределения по закону Пуассона вероятности за время кадра сделать k попыток передачи, как новых, так и ранее не переданных из-за коллизий кадров

**Типы CSMA:**

1. **1-настойчивый CSMA.** Когда станция готова к передаче данных, она прослушивает канал. Если канал оказывается свободным, происходит передача кадра. Если же канал занят, то станция ждет, пока канал не освободится, затем сразу начинает передачу данных. В случае возникновения коллизии станция ждет в течение случайного интервала времени, затем снова прослушивает канал и, если он свободен, пытается передать кадр еще раз.
2. **Ненастойчивый CSMA.** Когда станция готова к передаче данных, она прослушивает канал. Если канал оказывается свободным, происходит передача кадра. Если же канал занят, то станция ждет в течение случайного интервала времени, а затем прослушивает линию снова. При возникновении коллизии станция поступает также, как и в случае 1-настойчивого CSMA.
3. **p-настойчивый CSMA.** Этот протокол используется в дискретных каналах. Когда станция готова передавать, она опрашивает канал. Если канал свободен, она с вероятностью *p* начинает передачу. С вероятностью *1-p* она отказывается от передачи и ждет начала следующего такта. Процесс повторяется до тех пор, пока кадр не будет передан или другая станция не начнет передачу. В последнем случае станция поступает также, как и при возникновении коллизии. Если при первом прослушивании канала он оказывается занят, станция ждет следующего интервала времени, после чего применяется тот же алгоритм.

## **26. Протокол IEEE 802.3 и система передачи данных Ethernet (кабели, способ физического кодирования, понятие коллизии, алгоритм вычисления задержки, МАС подуровень, структура кадра, LLC подуровень).**

Относится к 1-настойчивым протоколам CSMA/CD

Прежде, чем начать передачу, станция, использующая такой протокол, опрашивает канал:

* Если канал занят, то станция ждёт и как только он освободится, сразу начинает передачу.
* Если несколько станций одновременно начали передачу, то возникает **коллизия** и передача тут же прекращается. Станции переходят в этап состязаний - ожидают некоторый случайный промежуток времени, вычисляемый по алгоритму двоичной экспоненциальной задержки, и всё начинается сначала.

**Кабели**

* Толстый коаксиал
  + Часто называют Толстый Ethernet
  + Коаксиальный кабель с отметками через каждые 2.5 м, указывающими, где можно производить подключение.
  + Подключение выполняется через специальные розетки, которые монтируются прямо на кабеле.
  + Трансивер встраивался в эти розетки
  + Кабель обеспечивает пропускную способность 10 Мбит/с, а максимальная длина равна 500 м.
* Тонкий коаксиал
  + Это более простой в употреблении коаксиальный кабель с простым подключением через BNC-коннектор, представляющий собой Т-образное соединение коаксиальных кабелей.
  + Его сегмент не должен превышать 200м и объединять более 30 машин.
  + Трансивер расположен на контроллере (см. ниже) в компьютере
* Витая пара
  + Конфигурация на основе витой пары - каждая машина соединяется витой парой со специальным устройством – хабом.
  + Трансиверы в 10Base5 размещаются прямо на кабеле и соединяются с компьютером трансиверным кабелем, длина которого не может превышать 50м.
    - Трансиверный кабель состоит из пяти витых пар.
    - Две из них используются для передачи данных к компьютеру и от него, две служат для передачи управляющей информации в обе стороны, а пятая – для подачи питания на трансивер.
  + Трансиверный кабель подключается к контроллеру в компьютере через интерфейс AUI. В контроллере имеется специальная микросхема, отвечающая за:
    - прием кадров и их отправку,
    - проверку и формирование контрольной суммы.
    - управление буферами на канальном уровне,
    - управление очередью буферов на отправку,
    - обеспечивает прямой доступ к памяти машины,
    - а также решает другие вопросы доступа к сети.
* Оптоволокно
  + Недостатки
    - Относительно дорогое
  + Преимущества
    - Обеспечение низкого уровня шума
    - Большая длина одного сегмента

**Способ физического кодирования**

Для физического кодирования используется манчестерское кодирование

* Манчестерский код
  + Биполярный импульсный код
  + 0 - переход с высокого на низкий потенциал в середине интервала
  + 1 - переход с низкого на высокий потенциал в середине интервала
  + У всех биполярных импульсных кодов сигнальная скорость в два раза выше, чем у потенциальных кодов
  + Преимущества:
    - самосинхронизация
    - отсутствие постоянной составляющей
    - обнаружение единичных ошибок
* Дифференциальный манчестерский код
  + В середине интервала всегда происходит смена потенциала - она не несет никакой информации
  + Смотреть нужно на стыки интервалов
    - 0 - смена потенциала произошла
    - 1 - смены потенциала не было

**Алгоритм вычисления задержки:**

Алгоритм двоичной экспоненциальной задержки:

* При первой коллизии все станции, участвовавшие в ней, случайно выбирают слот из диапазона [0,], т.е. 0 или 1
* Возможны 2 исхода
  + Минимальное значение (0) выбрала лишь одна станция
    - Коллизия разрешена
  + Несколько станций выбрали минимальное значение (0) или все выбрали максимальное значение (1)
    - Возникает повторная коллизия с номером 1
    - Происходит новый выбор среди чисел из слотов в диапазоне [0,], где i – порядковый номер очередной коллизии.
* Важные нюансы
  + После 10-ти коллизий число слотов достигает 1023 и далее не увеличивается.
  + После 16 коллизии Ethernet-контроллер фиксирует ошибку и сообщает о ней более высокому уровню стека протоколов.

**Канальный уровень разделен на 2 подуровня:**

* **LLC (Logical Link Control)** - подуровень логической передачи данных
* **MAC (Media Access Control)** - подуровень управления доступом к среде

**LLC:**

Основное назначение протокола **LLC (стандарт 802.2)** – обеспечение требуемого качества услуг системы передачи данных посредством передачи своих кадров одним из двух способов (дейтаграммным или с помощью процедур с установлением соединения и восстановлением кадров), а также обеспечение независимости вышерасположенных уровней стека протоколов от конкретного типа физической среды канала с множественным доступом.

Этот протокол занимает уровень между сетевыми протоколами и протоколами уровня MAC.

**Кадр LCC**



1. Адрес точки входа сервиса назначения (Destination Service Access Point, DSAP).
2. Адрес точки входа сервиса источника (Source Service Access Point, SSAP).
   1. Поля DSAP и SSAP указывают службу верхнего уровня, которая передает и принимает пакеты данных
   2. Программному обеспечению узлов сети при получении кадров канального уровня необходимо распознать, какой протокол вложил свой пакет в поле данных поступившего кадра, для того, чтобы передать извлеченный из кадра пакет нужному протоколу для последующей обработки.
3. Управляющее поле (Control, CTRL).
   1. Поле управления (один байт) используется для обозначения типа кадра данных – информационный, управляющий или ненумерованный.
   2. Кроме этого, в этом поле указываются порядковые номера отправленных и успешно принятых кадров, если подуровень LLC работает по процедуре LLC2 с установлением соединения.
4. Поле данных (Data).
   1. Поле данных кадра LLC предназначено для передачи по сети пакетов протоколов верхних уровней - IP, IPX, AppleTalk, DECnet, и др.
   2. Поле данных может отсутствовать в управляющих кадрах и некоторых ненумерованных кадрах.

Поля обрамляются двумя однобайтовыми полями «Флаг», имеющими значение «01111110». Флаги используются на MAC-уровне для определения границ кадра.

**MAC:**

Является интерфейсом между подуровнем LLC и физическим уровнем. Подуровень MAC ответственен за:

* формирование кадра Ethernet,
  + Структура кадра представлена ниже
* адресацию
  + Называется физической адресацией или MAC-адресами
  + MAC-адрес представляет собой уникальный серийный номер из 12-ти шестнадцатизначных цифр (6 байт), который присваивается каждому сетевому устройству во время изготовления, и позволяет однозначно определить его среди других сетевых устройств в мире.
  + Это гарантирует, что все устройства в сети будут иметь различные MAC-адреса, что делает возможным доставку пакетов данных в место назначения внутри подсети (сегмента)
* доступ к каналам - получение доступа к разделяемой среде передачи данных и отправку с помощью физического уровня кадра по физической среде узлу назначения.
  + В соответствии с CSMA/CD

LLC помещает протокол верхнего уровня в свой кадр, который дополняется необходимыми служебными полями. Далее, через межуровневый интерфейс передает свой кадр вместе с адресной информацией об узле назначения соответствующему протоколу уровня MAC, который упаковывает кадр LLC в свой кадр

**Структура кадра в IEEE 802.3**



* Кадр начинается с преамбулы – 7 байт вида 10101010, которая в манчестерском коде на скорости 10 МГц обеспечивает интервал времени 5.6 мкс для синхронизации приемника и передатчика.
* Затем следует стартовый байт 10101011, обозначающий начало передачи. Отличается от байта преамбулы последним битом - 1, а не 0.
* Адрес назначения и адрес источника
  + Допускает двух- и шести байтовые адреса назначения
  + НО! Для 10Base (один из стандартов Ethernet) используются только шестибайтовые.
  + Классификация адресов
    - По получателям
      * Нуль в старшем бите адреса получателя указывает на обычный адрес
      * Единица в старшем бите – это признак группового адреса. Групповой адрес позволяет обращаться сразу к нескольким станциям одновременно.
      * Адрес получателя, состоящий из одних единиц, – вещательный адрес, т.е. этот кадр должны получить все станции в сети.
    - По области
      * На то, какой адрес используется, указывает 46-й бит.
      * Если этот бит равен 1 – это локальный адрес, который устанавливает сетевой администратор, и вне данной СПД этот адрес смысла не имеет.
      * Глобальный адрес устанавливает IEEE, гарантируя при этом, что нигде в мире нет второго такого адреса. С помощью 46 бит можно получить 7 \* 1013 глобальных адресов.
* Длина поля данных содержит информацию о числе байт в поле “Данные”
* Данные
  + В кадре может занимать от 0 до 1500 байт.
* Важно! Поле “Данные” длиной 0 создаёт проблему для обнаружения коллизий, поэтому IEEE 802.3 предписывает, что кадр не может быть короче 64 байт (подробнее см. ниже). Если длина поля данных недостаточна, то поле “Заполнение” компенсирует нехватку длины. Этот приём называется расширением носителя.
* Контрольная сумма
  + Формируется с помощью CRC- кода.

Ограничение длины кадра связано со следующей проблемой. Если кадр короткий, то:

* станция может закончить передачу прежде чем начало этого кадра достигнет самого отдаленного получателя;
* пока начало кадра не достигнет самого отдаленного получателя, может произойти коллизия;
* однако станция, закончив отправку сообщения, пропустит коллизию, ошибочно считая, что кадр доставлен благополучно - это проблема
* в IEEE 802.3 (при 2,5 км и четырех репитерах) минимальное время обнаружения коллизии равно 51,2 мкс, что соответствует 64 байт. Поэтому введено такое ограничение

## **27. Сетевые коммутаторы: организация, основные функции, принципы функционирования. Коммутатор канального уровня с обучением. Виртуальные сети на основе протокола IEEE 802.1Q.**

**Сетевые коммутаторы**

Сетевой коммутатор представляет собой устройство с несколькими портами, к которым можно подключать сегменты СПД КМД (системы передачи данных на основе каналов с множественным доступом), например, сегменты 802.3.

На основании таблицы коммутации, расположенной в памяти коммутатора, кадры с входного порта коммутатор передает на надлежащий выходной порт. Тем самым трафик на отдельных сегментах существенно ниже, чем на коммутаторе в целом. Коммутаторы работают на более высоких скоростях, чем мосты, и функционально более гибкие.

По сравнению с мостами у коммутаторов больше портов. Достаточно распространены коммутаторы с 24 и 48 портами со скоростями 10 Мбит/с и 100 Мбит/с. Коммутаторы на крупных предприятиях могут поддерживать сотни портов. У коммутаторов больше размер буферного пространства для сохранения принимаемых кадров.

Большие буферы кадров. Возможность сохранять принимаемые кадры. Это весьма полезно, особенно если порты, серверы и другие элементы сети перегружены.

## Скорости портов. В зависимости от стоимости коммутатора возможна поддержка носителей с разными скоростями: 10 Мбит/с, 100 Мбит/с, 1 Гбит/с или 10 Гбит/с.

**Основная идея построения виртуальных сетей**

Виртуальной сетью LAN называют группу узлов сети, в которой весь трафик, включая и широковещательный, полностью изолирован на канальном уровне от других узлов сети. Это означает, что передача кадров между узлами сети, относящимися к различным виртуальным сетям, на основании адреса канального уровня невозможна (хотя виртуальные сети могут взаимодействовать друг с другом на сетевом уровне с использованием маршрутизаторов). Основным средством структурирования систем передачи данных на основе каналов с множественным доступом в виде системы виртуальных сетей являются коммутаторы.

Виртуальные сети способствуют повышению производительности сети, локализуя широковещательный трафик в пределах виртуальной сети. Изоляция виртуальных сетей друг от друга на канальном уровне позволяет повысить безопасность сети, делая часть ресурсов для определенных категорий пользователей недоступной.

Виртуальные сети на основе группировки портов (Port-based) обычно реализуются в так называемых управляемых коммутаторах, дополнение к возможности организации VLAN на базе стандарта IEEE 802.1Q.

Этот способ создания виртуальных сетей достаточно прост. Каждый порт коммутатора приписывается к той или иной виртуальной сети, то есть порты группируются в виртуальные сети. Адресация кадра в этой сети основывается на MAC-адресе получателя и ассоциированного с ним порта. Если к порту в определенной виртуальной сети,, подключить ПК, этот ПК автоматически будет принадлежать этой сети. Если к данному порту подключается коммутатор, то все порты этого коммутатора также будут принадлежать этой сети.

При использовании технологии группировки портов один и тот же порт может быть одновременно приписан к нескольким виртуальным сетям, что позволяет реализовывать разделяемые ресурсы между пользователями различных виртуальных сетей.

К достоинствам такого построения виртуальной сети можно отнести простоту конфигурации виртуальных сетей(При этом не требуется, чтобы конечные узлы сети поддерживали стандарт IEEE 802.1Q), можно создавать разделяемые сетевые ресурсы.

Однако, эта технология ограничивает возможности по наращиванию как числа VLAN, так и количество узлов к каждой виртуальной сети. В большинстве случаев эта технология ограничивается лишь одним коммутатором.

Существует несколько способов построения виртуальных сетей, но сегодня в коммутаторах главным образом реализуется технология группировки портов или используется спецификация IEEE 802.1Q.

В виртуальных сетях, основанных на стандарте IEEE 802.1Q, информация о принадлежности передаваемых Ethernet-кадров к той или иной виртуальной сети встраивается в сам передаваемый кадр. Таким образом, стандарт IEEE 802.1Q определяет изменения в структуре кадра Ethernet, позволяющие передавать информацию о VLAN по сети.

## **28. Сетевой уровень в Интернет: адресация Ipv4 и IPv6, протокол маршрутизации IP, протоколы ARP, RARP, DHCP.**

Каждая машина в Internet имеет уникальный IP адрес. Он состоит из адреса сети и адреса машины в этой сети. Все IP адреса имеют длину 32 разряда. Все адреса разделяются на классы. Всего есть пять классов адресов: А, B, C, D, E. Адреса выделяет только NIC – Network Information Center.

**Подсети.** Все машины одной сети должны иметь одинаковый номер сети в адресе. По мере роста сети приходится менять класс адреса. Перенос машины из одной сети в другую требует изменения маршрутизации. Решением этих проблем является разделение сети на части, которые извне видны как единое целое, но внутри каждая часть имеет свой адрес. Эти части называются подсети. Подсеть – это часть сети, не видимая извне. Изменение адреса подсети или введение новой подсети не требует обращения в NIC или изменений какой-либо глобальной базы данных.

**Адресация.** Есть две таблицы. Первая показывает как достичь интересующей сети. Вторая – как достичь узел внутри сети. Когда поступает IP пакет, маршрутизатор ищет его адрес доставки в таблице маршрутизации. Если это адрес другой сети, то пакет передают дальше тому маршрутизатору, который отвечает за связь с этой сетью. Если это адрес в локальной сети, то маршрутизатор направляет пакет прямо по месту назначения. Если адреса нет в таблице, то маршрутизатор направляет пакет специально выделенному по умолчанию маршрутизатору, который должен разобраться с этим случаем с помощью более подробной таблицы. Теперь записи в таблице имеют форму (эта\_сеть, подсеть, 0) и (эта\_сеть, эта\_подсеть, машина). Таким образом, маршрутизатор подсети в данной локальной сети знает, как достичь любую подсеть в данной локальной сети, и как найти конкретную машину в своей подсети.

**Протоколы управления межсетевым взаимодействием.**

**Internet Control Message Protocol.** Управление функционированием Internet происходит через маршрутизаторы с помощью протокола ICMP. Этот протокол выявляет и рассылает сообщения о десятках событий.

**Address Resolution Protocol** – протокол определения адреса.

Ethernet адрес. Этот адрес имеет 48 разрядов. Сетевая карта знает только такие адреса и ничего об 32-разрядных IP. Как отобразить 32-разрядный IP адрес на адреса канального уровня, например, Ethernet адрес. Для отображения IP адреса на Ethernet адрес, в подсеть посылается запрос у кого такой IP адрес. Машина с указанным адресом шлет ответ. Протокол, который реализует рассылку запросов и сбор ответов - ARP протокол. Практически каждая машина в Internet имеет этот протокол. Для того чтобы узнать адрес в другой сети есть два решения - есть определенный маршрутизатор, который принимает все сообщения, адресованные определенной сети или группе адресов - proxy ARP. Этот маршрутизатор знает как найти адресуемую машину. Другое решение - выделенный маршрутизатор, который управляет маршрутизацией удаленного трафика. Машина определяет, что обращение идет в удаленную сеть и шлет сообщение на этот маршрутизатор.

**Reverse Address Resolution Protocol** — обратный протокол определения адреса. Иногда возникает обратная проблема – известен Ethernet адрес, какой IP адрес ему соответствует. Он посылает запрос к RARP серверу: Мой Ethernet адрес такой то, кто знает соответствующий IP адрес? RARP сервер отлавливает такие запросы и шлет ответ.

**DHCP** – сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Данный протокол работает по модели «клиент-сервер». Для автоматической конфигурации компьютер-клиент на этапе конфигурации сетевого устройства обращается к так называемому серверу DHCP, и получает от него нужные параметры. Сетевой администратор может задать диапазон адресов, распределяемых сервером среди компьютеров. Это позволяет избежать ручной настройки компьютеров сети. Протокол DHCP используется в большинстве сетей TCP/IP.

**Отличия IPv6:**

1. Более длинный адрес – 16 байт. Это решает одну из главных задач – неограниченное расширение Internet.

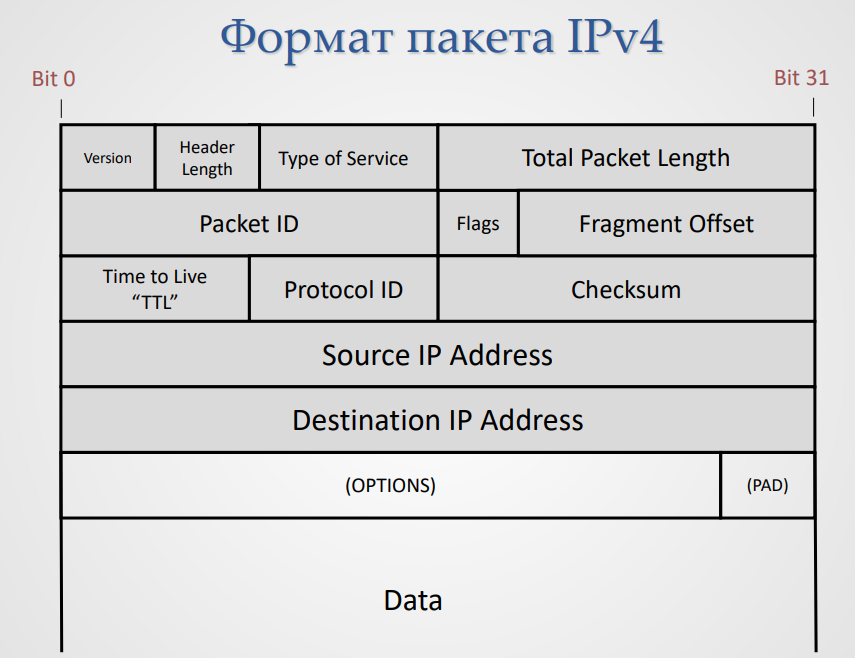
2. Заголовок стал проще (всего 7 полей), что ускорило обработку и маршрутизацию.

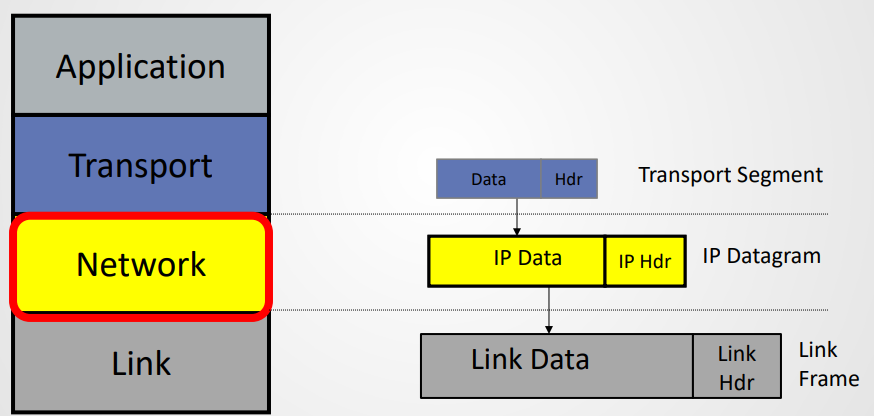
3. Он лучше поддерживает варианты в заголовке, что делает работу с ним более гибкой, позволяя опускать ненужные поля и вводить необходимые.

4. Серьезно улучшена безопасность протокола. Идентификация и конфиденциальность - ключевые возможности нового IP.

5. Существенно улучшена работа с типом сервиса, особенно учитывая возрастающий multimedia трафик. Существуют различные типы адресов IPv6: одноадресные (Unicast), групповые (Anycast) и многоадресные (Multicast).

**IP протокол** - network уровень модели TCP/IP.





## **29. Транспортный уровень: адресация, установление соединения, разрыв соединения, управление потоком и буферизацией, восстановление последовательности сегментов.**

Основная цель **транспортного уровня** — обеспечить эффективный и надежный сервис для пользователей на прикладном уровне. Задачи транспортного уровня выполняет **транспортный агент.**

Задачи сетевого уровня и транспортного в части адресации и управления потоком сильно схожи, тогда зачем новый уровень - дело в том, что сетевой уровень является частью среды передачи данных, которой управляет провайдер и у пользователя нет средств для ее контроля.

**Примитивы транспортного уровня.**

Под примитивами здесь подразумеваются инструменты, которые представляет транспортный агент пользователю (программисту) для управления соединением, такие как: LISTEN, CONNECT, SEND и RECEIVE. Соединение предполагает модель клиент-сервер, где одна машина LISTEN - ждет CONNECT клиента, после чего происходит обмен сообщениями через SEND и RECEIVE.

При использовании этих примитивов, машины обмениваются **TPDU (Transport Protocol Data Unit) сообщениями (сегментами).**

**Адресация**.

Для ассоциации транспортного подключения в сети с машиной используется транспортный адрес или **TSAP (Transport Service Access Port)**. Его структура устроена так, что в ней содержится вся информация, необходимая для подключения клиента к нужному серверу приложения на другой машине. Так же возможно использование постоянных TSAP, которые связаны с конкретными приложениями и никогда не меняются. Еще один подход - это использовать сервер процессов, указывающий при подключении к нему, для какого приложения, какой TSAP.

**Установление соединения.**

Установить соединения сложно, так как сетевой уровень ненадежен. Для борьбы с потерей или дублей сегментов у каждого из них есть время жизни t. Все машины оснащены синхронизированными часами, которые считаются достаточно точными. Сегмент становится недействительным по истечению времени T = λt, где λ > 1 - некий коэффициент, определяющийся протоколом. Само соединение устанавливается непосредственно тройным рукопожатием: пусть машина А хочет связаться с машиной Б. Тогда А посылает Б запрос на соединение, Б посылает подтверждение запроса, А отправляет Б, что получил запрос и дальше отправляет сами данные.

**Разрыв соединения.**

Разрыв может быть **асимметричным** (одна сторона разрывает соединение не дожидаясь согласия другой) и **симметричным** (одна сторона ожидает подтверждения на разрыв от другой). Первый тип разрыва может привести к потере данных. Но и второй тип не гарантирует отсутствие потерь при неизвестном количестве данных (*проблема двух армий*).

**Управление потоком и буферизацией**.

Надо гарантировать доставку всех сегментов, поэтому до получения подтверждения отправителю нужно хранить сегмент в каком-то буфере, а для борьбы с дубликатами стоит организовать буфер и у получателя. Тут возникает проблема размера буфера. Решением является сделать буфер динамического размера и менять этот размер в зависимости от текущей загруженности.

Хотя на данный момент память для буферов достаточно дешевая, скорость передачи по сети все еще ограничена, и если ее превысить в сети начнутся перегрузки. Проблема перегрузок решается динамически с помощью управляющих сообщений, которые контролируют количество отправляемых сегментов.

Механизм управления потоком располагается на стороне отправителя и учитывая пропускную способность среды контролирует отправку пакетов, размеры и типы буферов.

**Восстановление последовательно сегментов.**

По сети сегменты могут идти совершенно разными путями, и вследствие приходить получателю в разное время. Задачей транспортного агента - собрать их в необходимой последовательности.

## **30. Ключевые функции системы безопасности компьютерных систем (КС).**

**Безопасность информации** – состояние информационных ресурсов и информационных систем, при котором с требуемой вероятностью обеспечивается защита информации от утечки, хищения, утраты,

несанкционированного уничтожения, искажения, модификации, копирования и блокирования

**Ключевые функции системы безопасности:**

* Опознавание (идентификация и аутентификация)
* Регистрация
* Управление доступом
* Изоляция
* Контроль неизменности

**Опознавание и Регистрация (см. билет 41):**

* **Идентификация** – это процесс распознавания элемента объекта/субъекта
* **Аутентификация** – это проверка подлинности объекта/субъекта
* **Авторизация** – процедура предоставления субъекту определенных прав доступа к ресурсам системы
* **Администрирование** – процесс управления доступом субъектов к ресурсам системы.
  + создание идентификатора субъекта (создание учетной записи пользователя) в системе;
  + управление данными субъекта, используемыми для его аутентификации (смена пароля, издание сертификата и т. п.);
  + управление правами доступа субъекта к ресурсам системы.
* **Аудит** – процесс контроля (мониторинга) доступа субъектов к ресурсам системы

**Управление доступом (см. билет 42, 43):**

* **Монитор безопасности (МБ)** – субъект, необходимый для разделения всего множества потоков в КС на подмножества L и N
* **Политика безопасности** - общий принцип (методология, правило, схема) безопасной работы (доступа) коллектива пользователей с общими информационными ресурсами
* **МБ средство реализации политики безопасности в КС.**

**Изоляция (см. билет 44, 45):**

* **Шифрование** – обратимое преобразование информации в целях сокрытия от неавторизованных лиц с предоставлением в это же время авторизованным пользователям доступа к ней. Главным образом шифрование служит задачей соблюдения конфиденциальности передаваемой информации. Важной особенностью любого алгоритма шифрования является использование ключа, который утверждает выбор конкретного преобразования из совокупности возможных для данного алгоритма

**Контроль неизменности (см. билет 46):**

* **Электронная цифровая подпись.** Подлинность многих юридических, финансовых и прочих документов устанавливается наличием подписи уполномоченного лица. Имеются способы, позволяющие отличить фотокопии от подлинника. Подпись на документе — это факт, подтверждающий, что лицо, подписавшее документ, либо является автором документа, либо знакомо с документом.

## **31. Безопасность информации в сетях: основные понятия (угрозы, информация, документы, уязвимость, нарушитель, информационная безопасность, целостность, конфиденциальность, доступность, атака).**

**Угроза** – это потенциально возможное преднамеренное или непреднамеренное происшествие, которое может оказать нежелательное воздействие на саму систему в сети ЭВМ, а также привести к потере безопасности информации, хранящейся в ней.

**Информация** – сведения о фактах, событиях, процессах и явлениях, о состоянии объектов (их свойствах, характеристиках) в некоторой предметной области, необходимые для оптимизации принимаемых решений в процессе управления данными объектами.

**Конфиденциальная информация** – документированная информация, доступ к которой ограничен в

соответствии с законодательством Российской Федерации

**Документированная информация (Документ)** – зафиксированная на материальном носителе информация с реквизитами, позволяющими ее идентифицировать.

**Уязвимость** – это некая характеристика сети, которая делает возможным возникновение

угрозы безопасности информации.

**Нарушитель** – физическое лицо (субъект), случайно или преднамеренно совершающее действия, следствием которых является нарушение безопасности информации при ее обработке техническими средствами в информационных системах.

**Информационная безопасность** – практика предотвращения несанкционированного доступа, использования, раскрытия, искажения, изменения, исследования, записи или уничтожения информации. Основная задача информационной безопасности — сбалансированная защита конфиденциальности, целостности и доступности данных, с учетом целесообразности применения и без какого-либо ущерба производительности организации

**Целостность** – гарантия сохранности данными правильных значений, которая обеспечивается запретом для неавторизованных пользователей каким-либо образом изменять, модифицировать, разрушать или создавать данные.

**Конфиденциальность** – гарантия того, что секретные данные будут доступны только тем пользователям, которым этот доступ разрешен (такие пользователи называются авторизованными).

**Доступность** – гарантия того, что авторизованные пользователи всегда получат доступ к данным.

**Атака** – это действие, предпринимаемое злоумышленником, которое заключается в поиске и использовании уязвимости.

**Иранская АЭС подверглась атаке компьютерного вируса**

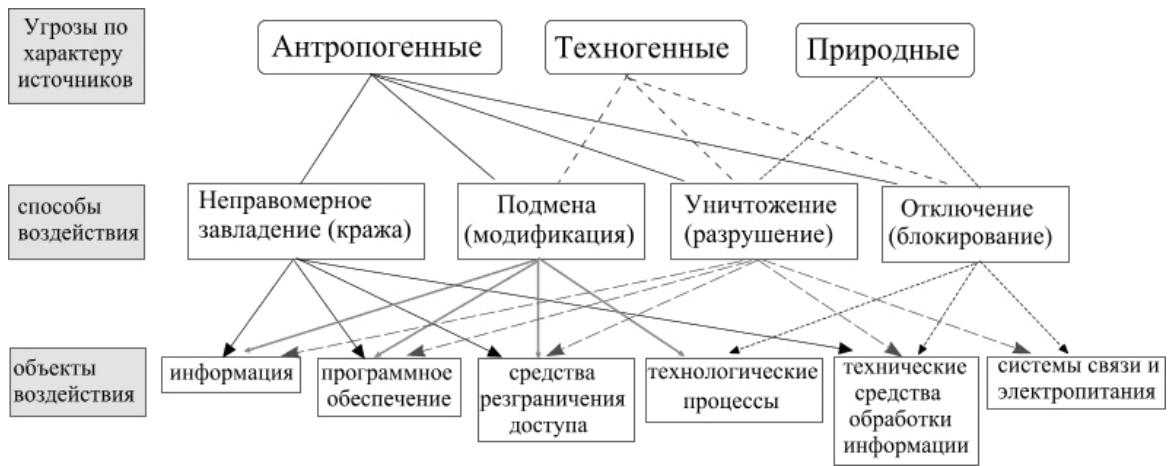
**Казино взломали через аквариум**

**Завод заразили через кофеварки**

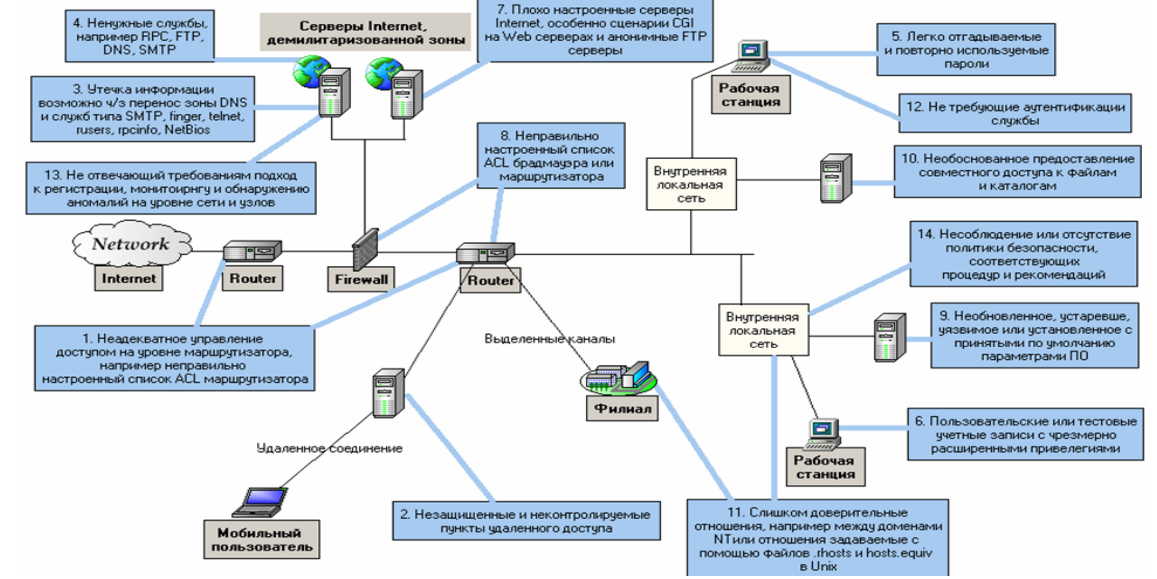
## **32. Понятия угрозы и уязвимости в компьютерных сетях, классификация угроз.**

**Опр. понятий угрозы и уязвимости в билете 39**

**Классификация угроз**





**Угрозы и уязвимости в компьютерных сетях**

## **33. Понятия идентификации, аутентификации и авторизации. Примеры алгоритмов.**

**Идентификация** – это процесс распознавания элемента объекта/субъекта

**Аутентификация** – это проверка подлинности объекта/субъекта

**Авторизация** – процедура предоставления субъекту определенных прав доступа к ресурсам системы

**Протоколы аутентификации**

Протоколы установления подлинности (аутентификации) позволяют процессу убедиться, что он взаимодействует с тем, кто должен быть, а не с тем, кто лишь представляется таковым.

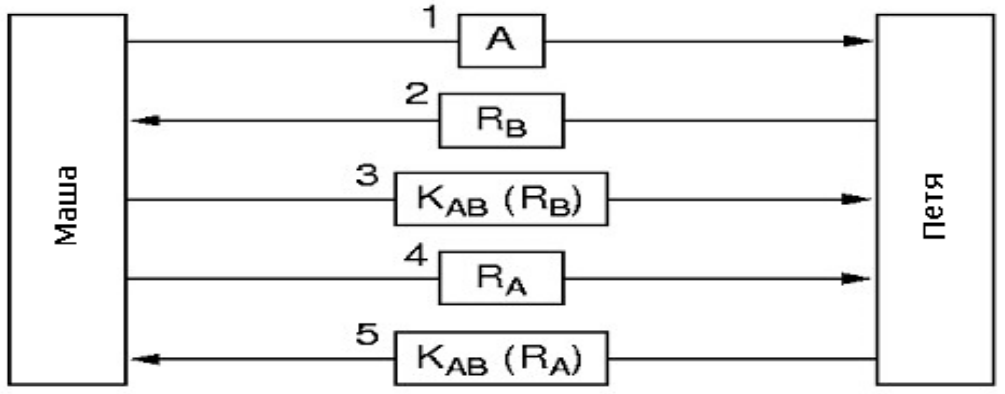
* Не путать с проверкой прав и полномочий.

**Общая схема всех протоколов аутентификации:**

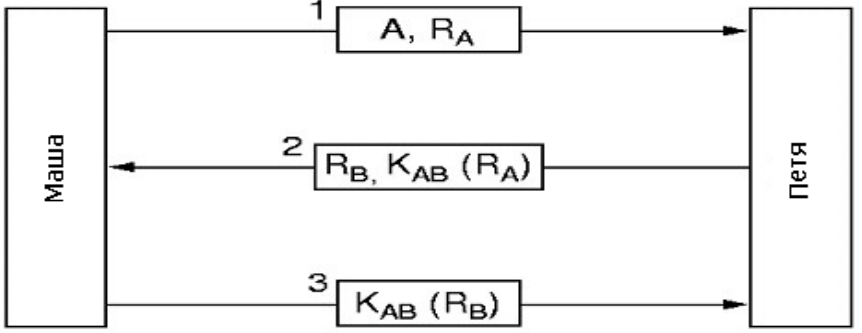
* А и В начинают обмениваться сообщениями между собой или с Центром раздачи ключей (ЦРК). ЦРК всегда надежный партнер.
* Протокол аутентификации должен быть устроен так, что даже если злоумышленник перехватит сообщения между А и В, то ни А ни В не спутают друг друга с злоумышленником.

**Аутентификация на основе секретного ключа**

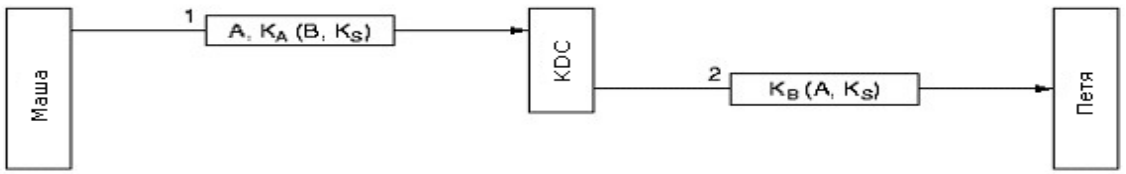
– общий ключ, , - вызовы



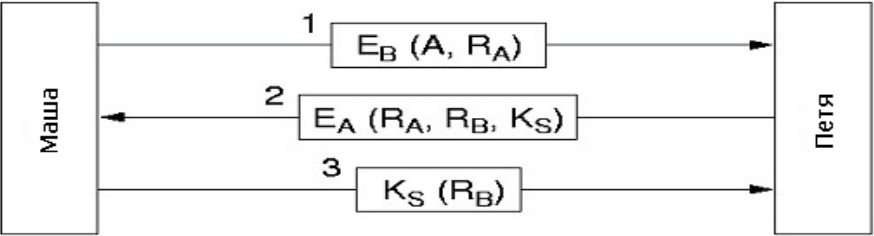
**Сокращенная двусторонняя аутентификация**



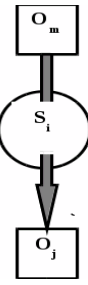
**Проверка подлинности через центр раздачи ключей**



**Взаимная аутентификация на основе открытых ключей**



## **34. Функции монитора безопасности. Субъектно-объектная модель управления доступом.**

* Р – множество потоков при фиксированной декомпозиции КС (компьютерная система) на субъекты и объекты во все моменты времени (все множество потоков является объединением потоков по всем моментам дискретного времени);
* N – подмножество потоков, характеризующее несанкционированный доступ;
* L – подмножество потоков, характеризующих легальный доступ.

**Субъектно-объектная модель КС**

1. В КС действует дискретное время.

2. В каждый фиксированный момент времени tk КС представляет собой конечное множество элементов, разделяемых на два подмножества:

* подмножество субъектов доступа S;
* подмножество объектов доступа O.

3. Пользователи КС представлены одним или некоторой совокупностью субъектов доступа, действующих от имени конкретного пользователя.

4. Субъекты КС могут быть порождены из объектов только активной сущностью (другим субъектом).

5. Все процессы в КС могут быть описаны через доступ субъектов к объектам, вызывающими потоки информации.

6. В защищенной сети в любой момент времени все объекты и все субъекты могут быть идентифицированы и аутентифицированы.

7. В КС есть **активный субъект (монитор безопасности)** с ассоциированным **объектом-источником (политика безопасности)**, который управляет и контролирует доступ субъектов к объектам.

**Потоком информации между объектом Оm и объектом Oj** называется произвольная операция над объектом Oj, реализуемая в субъекте Si и зависящая от Оm. Stream(Si Оm) → Oj (см. рис.)

**Поток всегда инициируется (порождается) субъектом.**

**Доступом субъекта Si к объекту Oj** будем называть порождение потока информации между некоторым объектом (например, ассоциированным с субъектом Si объектом Оm и объектом Oj.

**Монитор безопасности**

* Для разделения всего множества потоков в КС на подмножества L и N необходим субъект **Монитор безопасности (МБ)**
  + Субъект, активизирующийся при возникновении потока от любого субъекта к любому объекту
  + Распознающий потоки из L и N
  + Который разрешает поток, только из L - множества легального доступа.
* **Политика безопасности** - общий принцип (методология, правило, схема) безопасной работы (доступа) коллектива пользователей с общими информационными ресурсами
* **МБ средство реализации политики безопасности в КС.**

**Требования к Монитору безопасности:**

* **Полнота:** Монитор безопасности должен контролировать каждый доступом любого субъекта к любому объекту, и не должно быть никаких способов его обхода.
* **Изолированность:** Монитор безопасности должен быть защищен от отслеживания и перехвата своей работы.
* **Верифицируемость:** Монитор безопасности должен быть проверяемым (само- или внешне тестируемым) на предмет работоспособности и корректного выполнения своих функций.
* **Работоспособность:** Монитор безопасности должен функционировать в любых штатных и нештатных, в том числе и аварийных ситуациях

**Следствие 1.** В защищенной КС существуют особая категория субъектов (активных сущностей), которые не инициализируют и которыми не управляют пользователи системы. Это - системные процессы (субъекты), присутствующие (функционирующие) в системе изначально.

**Следствие 2.** Ассоциированный с монитором безопасности объект, содержащий информацию по системе разграничения доступа, является наиболее критическим с точки зрения безопасности информационным

ресурсом в защищенной КС.

**Следствие 3.** В защищенной системе может существовать доверенный пользователь (администратор системы), субъекты которого имеют доступ к ассоциированному с монитором безопасности объекту-данным (политика безопасности) для управления политикой разграничения доступа.

## **35. Понятия и виды политики безопасности.**

* **Р** – множество потоков при фиксированной декомпозиции КС (компьютерная система) на субъекты и объекты во все моменты времени (все множество потоков является объединением потоков по всем моментам дискретного времени);
* **N** – подмножество потоков, характеризующее несанкционированный доступ;
* **L** – подмножество потоков, характеризующих легальный доступ.

**Политика безопасности** - общий принцип (методология, правило, схема) безопасной работы (доступа) коллектива пользователей с общими информационными ресурсами

**Базовые политики безопасности** – дискреционная (матричная) и мандатная (полномочий).

* **Дискреционная политика** – определяет L в явном виде как набор троек   
  «Пользователь(субъект) – поток(операция) – объект» (матрица доступа).
* **Мандатная политика** – определяет L неявным образом через введение для пользователей-субъектов дискретной характеристики доверия (уровня допуска), а для объектов некоторой дискретной характеристики конфиденциальности (грифа секретности), и наделение на этой основе пользователей-субъектов полномочиями порождать определенные потоки в зависимости от соотношения   
  «уровень допуска – поток(операция) – уровень конфиденциальности».

**Производные политики безопасности:**

* **Политика тематического доступа** (вариант мандатной политики) – Множество безопасных (разрешенных) доступов L задается неявным образом через введение для пользователей-субъектов некоторой тематической характеристики - разрешенных тематических информационных рубрик, а для объектов аналогичной характеристики в виде набора тематических рубрик, информация по которым содержится в объекте, и наделение на этой основе субъектов-пользователей полномочиями порождать определенные потоки в зависимости от соотношения   
  «набор тематических рубрик субъекта – набор тематических рубрик объекта».
* **Политика ролевого доступа** –множество безопасных (разрешенных) доступов L задается через введение в системе дополнительных абстрактных сущностей - ролей, выступающих некими "типовыми" ролевыми субъектами доступа, с которыми ассоциируются конкретные пользователи (в роли которых осуществляют доступ), и наделение ролевых субъектов доступа на основе дискреционного или мандатного принципа правами доступа к объектам системы.

## **36. Основные виды шифрования. Алгоритмы шифрования с закрытым ключом. Примеры.**

Существует два основных вида шифрования: **симметричное и асимметричное**.

**Симметричное шифрование** для шифрования и дешифрования данных использует один и тот же криптографический ключ.

**Асимметричное шифрование** для шифрования и дешифрования данных использует различные криптографические ключи.

Алгоритмы с секретными ключами. (симметричное шифрование)

Все алгоритмы шифрования с позиции использования ключа подразделяются на алгоритмы с секретным ключом и алгоритмы с открытым ключом. Алгоритмы с секретным ключом используют один ключ и для шифрования, и для дешифрования, поэтому их часто называют симметричными алгоритмами шифрования. Этот ключ является строжайшим секретом, известным только тому, кто шифрует сообщение, и тому, кто это сообщение расшифровывает. Слабым местом этих шифров является этап установки общего секретного ключа.

Примеры:

**DES**. Одним из широко известных шифров с секретным ключом является шифр DES (Data Encryption Standard). Алгоритм состоит из 19 этапов. На первом этапе исходный текст разбивается на блоки по 64 бит каждый, и над каждым блоком выполняется перестановка. Последний этап является инверсией первой перестановки. Предпоследний этап заключается в обмене местами 32 самых левых битов и 32 самых правых битов. Для этапов со 2-го по 17-й с помощью специального преобразования исходного 56-разрядного ключа строятся 16 частных ключей, которые используются для преобразования данных. У алгоритма DES имеется два недостатка. Во-первых, он представляет собой моноалфавитное замещение с 64-разрядным символом, а всегда при подаче одних и тех же 64 разрядов исходного текста на вход, те же самые 64 разряда получают на выходе. DES сохраняет структуру сообщения, т. е. одни и те же поля исходного текста попадут в одни и те же места шифр-текста, чем может воспользоваться злоумышленник. Зная структуру исходного сообщения и длину его полей, он просто переставит необходимые поля в шифр-тексте, чтобы несанкционированно изменить сообщение. Во-вторых, для начала шифрования надо иметь сразу весь 64-разрядный блок исходного текста, а это не совсем удобно, если речь идет об интерактивных приложениях.

**AES**. Шифруемые данные представляют в виде двухмерных байтовых массивов размером 4 на 4 байт. Все операции производятся над отдельными байтами массива, независимо над столбцами и строками. На каждой итерации алгоритма выполняются следующие преобразования массива:

1. Операция Sub Bytes, представляющая собой замену каждого байта массива данных в соответствии со специальной таблицей кодирования.

2. Операция Shift Rows, представляющая собой циклический сдвиг влево всех строк массива данных за исключением нулевой. Сдвиг i-й строки массива (для i = 1, 2, 3) производится на i байт.

3. Операция MixColumns, выполняющая умножение каждого столбца массива данных, который рассматривается как полином степени , на фиксированный полином (𝑥) =. Умножение выполняется по модулю .

4. Операция Add Round Key, преобразующая массив данных с расширенным ключом итерации (наложение ключа).

Процедура получения расширенного ключа для каждой итерации такова: над i-м столбцом массива данных (i = 0...3) побитово выполняется логическая операция XOR с расширенным ключом , где r – номер текущей итерации алгоритма, начиная с 1. Количество итераций R алгоритма зависит от длины ключа. Перед первой итерацией алгоритма AES выполняется предварительное наложение расширенного ключа 𝑊0...𝑊3 на открытый текст первых четырех слов итерации с помощью операции AddRoundKey. Последняя итерация отличается от предыдущих тем, что в ней нет операции MixColumns. В алгоритме AES используются ключи шифрования трех фиксированных размеров: 128, 192, и 256 бит.

## **37. Основные виды шифрования. Алгоритмы шифрования с открытым ключом. Примеры.**

Существует два основных вида шифрования: **симметричное и асимметричное**.

**Симметричное шифрование** для шифрования и дешифрования данных использует один и тот же криптографический ключ.

**Асимметричное шифрование** для шифрования и дешифрования данных использует различные криптографические ключи.

Алгоритмы с открытыми ключами. (асимметричное шифрование)

Пусть имеются алгоритмы шифрования Е и дешифрования D которые удовлетворяют следующим требованиям:

* D(E(P)) равно исходному тексту Р,
* чрезвычайно трудно получить D, зная Е,
* Е нельзя вскрыть через анализ исходных текстов.

Алгоритм шифрования Е и его ключ (открытый ключ), публикуют или помещают таким образом, чтобы каждый мог их получить. Алгоритм D также публикуют, чтобы подвергнуть его изучению и проверке на стойкость, а вот ключ к нему хранят в секрете. Это секретный (или закрытый) ключ. Взаимодействие двух абонентов А и В происходит следующим образом.

Пусть А хочет послать В текст Р. Абонент А шифрует текст 𝐸(𝑃), зная алгоритм и открытый ключ абонента В для шифрования. Абонент В, получив текст 𝐸(𝑃) и использовав секретный ключ и алгоритм 𝐷, вычисляет 𝐷(𝐸(𝑃)) = 𝑃. Никто не прочтет текст Р кроме абонентов А и В, так как по условию алгоритм 𝐸 нераскрываем, а алгоритм 𝐷 нельзя вывести из 𝐸.

Пример:

**алгоритм RSA**. Общая схема этого алгоритма следующая:

1. Выберем два больших (больше ) простых числа р и q.

2. Вычислим n = р \* q и z = (p - 1) (q - 1).

3. Выберем простое d, взаимно простое по отношению к z.

4. Найдем е, удовлетворяющее условию (е \* d) mod z = 1.

Разобьем исходный текст на блоки длиной s таким образом, чтобы каждый блок как число не превосходил n. Для этого выберем наибольшее k, при котором выполняется условие s = < 𝑛. Шифр сообщения s получим, вычислив шифрованный текст 𝐶 = (𝑚𝑜𝑑 𝑛). Для расшифровки найдем 𝑃 = (𝑚𝑜𝑑 𝑛).

Для шифрования требуются числа e, n, представляющие собой открытый ключ, а для дешифрования – числа d, n – закрытый ключ.

Безопасность, обеспечиваемая применением этого метода шифрования, основана на высокой вычислительной сложности операции разложения на простые множители больших чисел.

Один из основных недостатков алгоритма RSA – медленная работа.

## **38. Информационная безопасность: основные задачи. Протоколы установления подлинности на основе закрытого ключа, протокол Диффи-Хеллмана. Электронная подпись.**

Протоколы установления подлинности (аутентификации) позволяют процессу убедиться, что он взаимодействует с тем, с кем должен, а не с тем, кто лишь представляется таковым. Авторизация — проверка прав на выполнение тех или иных операций.

Общая схема всех протоколов аутентификации следующая: сторона А и сторона В начинают обмениваться сообщениями между собой или с центром раздачи ключей (ЦРК). При этом предполагается, что ЦРК — всегда надежный партнер, т.е. его нельзя фальсифицировать. Протокол аутентификации должен быть устроен таким образом, чтобы даже в случае перехвата злоумышленником сообщения между А и В, ни А, ни В не спутают друг друга со злоумышленником.

**Аутентификация на основе закрытого разделяемого ключа.**

Протокол ответа по вызову:

одна сторона посылает некоторое число (вызов), а другая сторона, получив это число, преобразует его по определенному алгоритму и отправляет обратно. Увидев результат преобразования и зная исходное число, инициатор может определить, правильно сделано преобразование или нет. Алгоритм преобразования является общим секретом взаимодействующих сторон.

Существует несколько общих правил построения протоколов аутентификации:

* Инициатор передачи должен доказать, кто он есть, прежде чем вы пошлете ему какую-либо важную информацию.
* Инициатор и отвечающий должны использовать разные ключи.
* Инициатор и отвечающий должны использовать начальные вызовы из разных непересекающихся множеств.

**Установка общего закрытого ключа. Протокол Диффи-Хеллмана.**

Прежде всего, Маша и Петя должны договориться об использовании двух больших простых чисел 𝑛 и 𝑔, удовлетворяющих определенным условиям, причем эти числа могут быть общеизвестны. Затем Маша выбирает большое число, например 𝑥, и хранит его в секрете, а Петя выбирает число 𝑦 и также держит его в секрете. Маша посылает Пете сообщение вида (𝑛, 𝑔, 𝑚𝑜𝑑 𝑛), а Петя отвечает сообщением вида ( 𝑚𝑜𝑑 𝑛). Теперь Маша выполняет операцию , а Петя — операцию . Теперь они имеют общий ключ ( 𝑚𝑜𝑑 𝑛). Злоумышленник следит за всем этим, и единственное, что мешает ему вычислить 𝑥 и 𝑦, это то, что неизвестен алгоритм с приемлемой сложностью вычисления логарифма от модуля для простых чисел.

Слабое место – «чужой в цепочке».

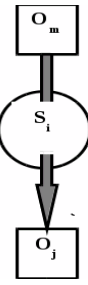
**Электронная цифровая подпись.** Подлинность многих юридических, финансовых и прочих документов устанавливается наличием подписи уполномоченного лица. Имеются способы, позволяющие отличить фотокопии от подлинника. Подпись на документе — это факт, подтверждающий, что лицо, подписавшее документ, либо является автором документа, либо знакомо с документом. Проблема создания электронного аналога ручной подписи:

* Получатель должен иметь возможность удостовериться в подлинности отправителя.
* Отправитель не должен иметь возможность отречься от документа.
* Получатель не должен иметь возможность подделать документ

**Подпись с секретным ключом.** Одно из решений проблемы электронной подписи — наделить полномочиями третью сторону, которую знают все, которая знает всех и которой верят все. Единственная слабость такого решения заключается в том, что злоумышленник может скопировать диалог между отправителем и получателем через СД и позже его повторить. Механизм временных меток позволяет ослабить эту проблему.

**Подпись с открытым ключом.** Предположим 𝐸(𝐷(𝑃)) = 𝑃 дополнительно к 𝐷(𝐸(𝑃)) = 𝑃 (этим свойством обладает алгоритм шифрования RSA). В этой схеме имеются два недостатка. Оба основаны на том, что схема работает до тех пор, пока пользователь либо умышленно не рассекретит свой ключ, либо не изменит его в одностороннем порядке.

## **39. Субъектно-объектная модель управления доступом.**

* Р – множество потоков при фиксированной декомпозиции КС (компьютерная система) на субъекты и объекты во все моменты времени (все множество потоков является объединением потоков по всем моментам дискретного времени);
* N – подмножество потоков, характеризующее несанкционированный доступ;
* L – подмножество потоков, характеризующих легальный доступ.

**Субъектно-объектная модель КС**

1. В КС действует дискретное время.

2. В каждый фиксированный момент времени tk КС представляет собой конечное множество элементов, разделяемых на два подмножества:

* подмножество субъектов доступа S;
* подмножество объектов доступа O.

3. Пользователи КС представлены одним или некоторой совокупностью субъектов доступа, действующих от имени конкретного пользователя.

4. Субъекты КС могут быть порождены из объектов только активной сущностью (другим субъектом).

5. Все процессы в КС могут быть описаны через доступ субъектов к объектам, вызывающими потоки информации.

6. В защищенной сети в любой момент времени все объекты и все субъекты могут быть идентифицированы и аутентифицированы.

7. В КС есть **активный субъект (монитор безопасности)** с ассоциированным **объектом-источником (политика безопасности)**, который управляет и контролирует доступ субъектов к объектам.

**Потоком информации между объектом Оm и объектом Oj** называется произвольная операция над объектом Oj, реализуемая в субъекте Si и зависящая от Оm. Stream(Si Оm) → Oj (см. рис.)

**Поток всегда инициируется (порождается) субъектом.**

**Доступом субъекта Si к объекту Oj** будем называть порождение потока информации между некоторым объектом (например, ассоциированным с субъектом Si объектом Оm и объектом Oj.

**Монитор безопасности**

* Для разделения всего множества потоков в КС на подмножества L и N необходим субъект **Монитор безопасности (МБ)**
  + Субъект, активизирующийся при возникновении потока от любого субъекта к любому объекту
  + Распознающий потоки из L и N
  + Который разрешает поток, только из L - множества легального доступа.
* **Политика безопасности** - общий принцип (методология, правило, схема) безопасной работы (доступа) коллектива пользователей с общими информационными ресурсами
* **МБ средство реализации политики безопасности в КС.**

**Требования к Монитору безопасности:**

* **Полнота:** Монитор безопасности должен контролировать каждый доступом любого субъекта к любому объекту, и не должно быть никаких способов его обхода.
* **Изолированность:** Монитор безопасности должен быть защищен от отслеживания и перехвата своей работы.
* **Верифицируемость:** Монитор безопасности должен быть проверяемым (само- или внешне тестируемым) на предмет работоспособности и корректного выполнения своих функций.
* **Работоспособность:** Монитор безопасности должен функционировать в любых штатных и нештатных, в том числе и аварийных ситуациях

**Следствие 1.** В защищенной КС существуют особая категория субъектов (активных сущностей), которые не инициализируют и которыми не управляют пользователи системы. Это - системные процессы (субъекты), присутствующие (функционирующие) в системе изначально.

**Следствие 2.** Ассоциированный с монитором безопасности объект, содержащий информацию по системе разграничения доступа, является наиболее критическим с точки зрения безопасности информационным

ресурсом в защищенной КС.

**Следствие 3.** В защищенной системе может существовать доверенный пользователь (администратор системы), субъекты которого имеют доступ к ассоциированному с монитором безопасности объекту-данным (политика безопасности) для управления политикой разграничения доступа

**40. Протокол установления секретного ключа Диффи-Хеллмана**

Прежде всего, Маша и Петя должны договориться об использовании двух больших простых чисел 𝑛 и 𝑔, удовлетворяющих определенным условиям, причем эти числа могут быть общеизвестны. Затем Маша выбирает большое число, например 𝑥, и хранит его в секрете, а Петя выбирает число 𝑦 и также держит его в секрете. Маша посылает Пете сообщение вида (𝑛, 𝑔, 𝑚𝑜𝑑 𝑛), а Петя отвечает сообщением вида ( 𝑚𝑜𝑑 𝑛). Теперь Маша выполняет операцию , а Петя — операцию . Теперь они имеют общий ключ ( 𝑚𝑜𝑑 𝑛). Злоумышленник следит за всем этим, и единственное, что мешает ему вычислить 𝑥 и 𝑦, это то, что неизвестен алгоритм с приемлемой сложностью вычисления логарифма от модуля для простых чисел.

Слабое место – «чужой в цепочке».

**Электронная цифровая подпись.** Подлинность многих юридических, финансовых и прочих документов устанавливается наличием подписи уполномоченного лица. Имеются способы, позволяющие отличить фотокопии от подлинника. Подпись на документе — это факт, подтверждающий, что лицо, подписавшее документ, либо является автором документа, либо знакомо с документом. Проблема создания электронного аналога ручной подписи:

* Получатель должен иметь возможность удостовериться в подлинности отправителя.
* Отправитель не должен иметь возможность отречься от документа.
* Получатель не должен иметь возможность подделать документ

**Подпись с секретным ключом.** Одно из решений проблемы электронной подписи — наделить полномочиями третью сторону, которую знают все, которая знает всех и которой верят все. Единственная слабость такого решения заключается в том, что злоумышленник может скопировать диалог между отправителем и получателем через СД и позже его повторить. Механизм временных меток позволяет ослабить эту проблему.

**41. Протокол RSA с открытым ключом**

Суть **шифрования с открытым ключом** заключается в том, что для шифрования данных используется один ключ, а для расшифрования другой (поэтому такие системы часто называют асимметричными).

Пусть у нас есть алгоритмы Е и D, которые удовлетворяют следующим требованиям:

– D(E(P))=P;

– Чрезвычайно трудно получить D, зная E;

– Е нельзя вскрыть через анализ исходных текстов.

Алгоритм шифрования Е и его ключ публикуют или помещают так, что каждый может их получить, алгоритм D также публикуют, чтобы подвергнуть его изучению, а вот ключи к последнему хранят в секрете.

Примером такого алгоритма является **алгоритм RSA**, предложенный Ривестом, Шамиром и Адлеманом в 1978 году. Общая схема этого алгоритма такова:

– Выберем два больших (более 10^100) простых числа p и q.

– Вычислим n = p•q и z = (p-1)•(q-1).

– Выберем d относительно простое к z. Таких чисел несколько, какое выберем – это секрет.

– Вычислим e такое, что e•d = 1 (mod z).

– Разбиваем исходный текст на блоки длины Р так, чтобы каждый блок, как число не превосходил n. Для этого выбираем наибольшее k такое, чтобы P=2k•n. Вычисляем С=(P^e) (mod n), чтобы зашифровать сообщение P.

– Для расшифровки вычисляем P=(C^d) (mod n). Для шифрования нам нужна пара (e,n) – это открытый ключ, для расшифровки – пара (d,n), это закрытый ключ. Можно доказать, что для любого Р в указанном выше диапазоне, функция шифрования и дешифрования взаимно обратны.

Безопасность этого метода основ ана на высокой вычислительной сложности операции разложения на множители больших простых чисел.

## **43. Информационная безопасность: контроль доступа и защита от компьютерных атак. Межсетевые экраны и их виды. Системы обнаружения и предотвращения компьютерных атак (метод аномалий и метод злоупотреблений).**

**Сканирование портов – предвестник атаки**

**Межсетевой экран (МСЭ)** – это устройство обеспечения безопасности сети, которое осуществляет мониторинг входящего и исходящего сетевого трафика и на основании установленного набора правил безопасности принимает решение: пропустить или заблокировать конкретный трафик.

**Типы межсетевых экранов (в порядке их появления):**

* Пакетные фильтры
* Шлюзы уровня приложений
* Шлюзы уровня соединения

**Недостатки межсетевых экранов**

* **Пакетные фильтры:**
  + Ограниченные возможности контроля доступа
  + Уязвимость к атакам подмены значений полей заголовка
* **Шлюзы уровня приложений:**
  + Высокая вычислительная сложность, нагрузка на аппаратуру
  + Сравнительно низкая пропускная способность
* **Общие:**
  + Сложность защиты новых протоколов
  + Возможность обхода
  + Незащищенность от вредоносного программного обеспечения и компьютерных атак

**Система обнаружения атак (intrusion detection)** – это система, которая идентифицирует вредоносную активность в сети

**Основные классы атак**

* Несанкционированный доступ
* Отказ в обслуживании

**Методы обнаружения атак**

* **Обнаружение аномалий**
  + Мониторинг сетевого трафика и сравнение с некоторым профилем нормальной работы сети
  + Оповещение администратора, если текущий трафик существенно отличается от нормы
  + Ошибки первого и второго рода
  + Как правило, такие методы склонны к ложным срабатываниям
* **Сигнатурные методы (Также известны как обнаружение злоупотреблений)**
  + Мониторинг сетевого трафика, и сравнение содержимого (или атрибутов) с базой описаний ранее известных реализаций атак
  + Реализуется аналогично большинству антивирусных сканеров
  + Менее склонны к ложным срабатываниям по сравнению с подходом на основе аномалий
  + Не могут обнаруживать даже варианты реализации известных атак, которых нет в базе описаний

## **44. Служба DNS: основные функции, структуры данных, принципы функционирования. Режим адресации anycast.**

**DNS (англ. Domain Name System «система доменных имён»)** — компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты и/или обслуживающих узлах для протоколов в домене.

**Задача:**

* Сопоставление символического имени IP адресу
* Автоматическая резолюция адресов
* DNS (Domain Name System)
* Распределенная база данных, поддерживающая иерархическую систему имен для идентификации узлов в сети Internet
* Механизм взаимодействия машин в сети Internet с этой базой данных

**DNS: ресурсная запись**

* **Имя (NAME)** — доменное имя, к которому привязана данная ресурсная запись
* **TTL (Time To Live)** — допустимое время хранения данной ресурсной записи в кэше неответственного DNS-сервера
* **Тип (TYPE) ресурсной записи** — определяет формат и назначение данной ресурсной записи
* **Поле данных (RDATA)**, формат и содержание которого зависит от типа записи.

**DNS: типы ресурсных записей**

* **Запись A (address record)** или запись адреса связывает имя хоста с адресом IP
* **Запись AAAA (IPv6 address record)** связывает имя хоста с адресом протокола IPv6
* **Запись CNAME (canonical name record)** или каноническая запись имени (псевдоним) используется для перенаправления на другое имя
* **Запись MX (mail exchange)** или почтовый обменник указывает сервер(ы) обмена почтой для данного домена
* **Запись NS (name server)** указывает на DNS-сервер для данного домена
* **Запись PTR (pointer)** или запись указателя связывает IP хоста с его каноническим именем
* **Запись SOA (Start of Authority)** или начальная запись зоны указывает, на каком сервере хранится эталонная информация о данном домене, содержит контактную информацию лица, ответственного за данную зону
* **Запись TXT** содержит не интерпретируемую текстовую информацию

**Поиск адреса по доменному имени**

* Приложение формирует запрос к DNS-серверу, где работает DNS-служба. Эта служба – приложение, обладающее соответствующей базой данных, с помощью которой оно обслуживает такого рода запросы.
* Обработка имени DNS-сервером выполняется справа налево, т.е. сначала производится поиск адреса в самой верхней группе иерархии, а затем он постепенно опускается по иерархии, тем самым сужая область поиска. В целях сокращения поиска сначала опрашивается локальный узел DNS.
  + Местный сервер знает адрес, потому что этот адрес содержится в его базе данных.
  + Местный сервер знает адрес, потому что кто-то недавно уже запрашивал его, и он сохранил у себя в кэш-памяти этот адрес (время хранения в кэш-памяти определяет TTL).
  + Местный сервер адрес не знает. В этом случае запускают ранее описанную процедуру опроса DNS-серверов доменов, указанных в имени справа налево.

**DNS: заключительные замечания**

* Доменная система именования указывает на то, кто ответственен за поддержку имени, но не где эта машина находится (несмотря на коды стран)
* Понятия доменного имени и адрес сети вообще говоря не связаны: две машины одного домена имен могут не принадлежать к одной сети
* У машины может быть много имен. В частности, это верно для машин, предоставляющих какие-либо услуги, которые в будущем могут быть помещены под опеку другой машины

**Anycast** (англ. any cast — «посылка/отправка данных кому угодно») — метод рассылки пакетов, позволяющий устройству посылать данные ближайшему из группы получателей. Реализован, в частности, в протоколе IPv6.

В протоколе IP anycast реализован путём публикации одинакового маршрута из различных точек сети через протокол BGP. Одним из основных критериев выбора маршрута в BGP является AS-path — набор (список) номеров автономных систем, через которые должен пройти пакет; выбирается маршрут с самым коротким списком AS-path. При получении анонса маршрутов из двух и более точек, будет выбран самый короткий.

Из-за особенностей топологии сети или её политики ближайший узел не обязательно будет географически ближайшим.

В настоящее время anycast используется в интернете для уменьшения времени отклика и для балансировки нагрузки корневых DNS-серверов. Например, корневой DNS-сервер K имеет множество инсталляций, в том числе в Амстердаме, Лондоне, Токио, Дели, Майами, Рейкьявике, Новосибирске, Хельсинки и других городах

## **45. Организация, функционирование и основные протоколы почтовой службы и прикладной протокол FTP.**

**Основные функции почтовой службы:**

* **Композиция** –обеспечивает создание сообщений и ответов
* **Передача** – обеспечивает передачу сообщения от отправителя к получателю без вмешательства пользователей.
* **Отчет о доставке.**
* **Отображение сообщения,** включая вопросы форматирования и кодировки
* **Размещение** – вопросы хранения сообщений, поиска среди них, повторной отправки или переадресации и т.п.

**Элементы краткого заголовка сообщения:**

* **Return-Path** — обратный адрес
* **Received** — строчка журналирования прохождения письма. Каждый почтовый сервер (MTA) помечает процесс обработки этим сообщением
* **Формат тела: MIME-Version** — версия MIME, с которым это сообщение создано
* **From:** — Имя и адрес отправителя.
* **Sender:** — Отправитель письма. Добавлено для возможности указать, что письмо от чьего-то имени (from) отправлено другой персоной (например, секретаршей от имени начальника)
* **To:** — Имя и адрес получателя. Может содержаться несколько раз (если письмо адресовано нескольким получателям). Может не совпадать с полем SMTP RCPT TO
* **cc:** — (от carbon copy). Содержит имена и адреса вторичных получателей письма, к которым направляется копия
* **bcc:** — (от blind carbon copy). Содержит имена и адреса получателей письма, чьи адреса не следует показывать другим получателям.

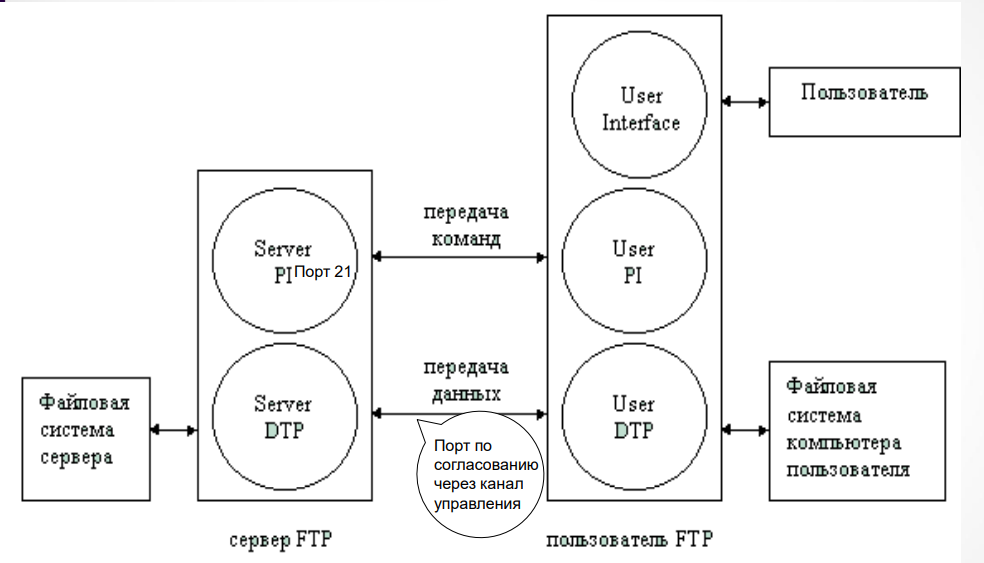
**Полный заголовок сообщения:**

* **Reply-To:** — имя и адрес, куда следует адресовать ответы на это письмо. Если, например, письмо рассылается ботом, то в качестве Reply-To будет указан адрес персоны, готовой принять ответ на письмо
* **Message-ID:** — уникальный идентификатор сообщения. Состоит из адреса узла-отправителя и номера (уникального в пределах узла). Выглядит примерно так: [AAB77AA217@example.com](mailto:AAB77AA2175ADD4BA@example.com). Вместе с другими идентификаторами используется для поиска прохождения конкретного сообщения по журналам почтовой системы и для указания на письмо из других писем
* **Content-Type:** — тип содержимого письма. С помощью этого поля указывается тип (HTML, RTF, Plain text) содержимого письма и кодировка, в которой создано письмо
* **In-Reply-To:** — указывает на Message-ID, для которого это письмо является ответом (с помощью этого почтовые клиенты могут легко выстраивать цепочку переписки)
* **Date:** — дата написания письма

**MIME (Multipurpose Internet Mail Extension)** – Поддержка различных алфавитов, включая нелатинские. Поддержка передачи нетекстовых данных

**SMTP (Simple Mail Transfer Protocol)** – передача письма между почтовыми серверами

**Для взаимодействия почтового агента с сервером:**

* **РОР3 (Post Office Protocol)** – простой протокол для изъятия почты из удаленного почтового ящика . Он позволяет забирать почту с сервера и хранить ее на машине пользователя
* **IMAP (Interactive Mail Access Protocol)** – позволяет одному и тому же пользователю заходить с разных машин на сервер, чтобы прочесть, отправить почту

**FTP (File Transfer Protocol)** – протокол передачи файлов по сети

**Задача:**

* Доступ к файлам на удаленных машинах
* Надежная передача файлов
* Независимость клиента от файловой системы удаленной машины

**Обозначения:**

* **User Interface** – пользовательский интерфейс работы с FTP;
* **User PI (User Protocol Interpretator)** – интерпретатор команд пользователя. Эта программа взаимодействует и с Server-PI, чтобы обмениваться командами управления передачей данных по каналу передачи команд, и с модулем User DTP, который осуществляет непосредственную передачу данных по каналу передачи данных;
* **User DTP (User Data Transfer Process)** – модуль, осуществляющий обмен данными между клиентом и сервером FTP по каналу передачи данных по командам от модуля User PI. Этот объект взаимодействует с файловой системой пользователя и объектом Server DTP;
* **Server PI (Server Protocol Interpretator)** – модуль управления обменом данных со стороны сервера по каналу передачи команд;
* **Server DTP (Server Data Transfer Process)** – модуль обмена данными со стороны сервера по каналу передачи данных;
* **Сервер FTP** – собственно сервер FTP, который состоит из модуля Server PI управления передачей и модуля Server DTP, осуществляющего передачу;
* **Пользователь FTP** – модуль клиента FTP, состоящий из-модуля управления передачей User PI и модуля, осуществляющего передачу, User DTP.

**Алгоритм работы FTP:**

1. Сервер FTP устанавливает в качестве управляющего соединение с портом 21 TCP, который всегда

находится в состоянии ожидания соединения со стороны FTP-клиента.

2. После установки управляющего соединения модуля User PI с модулем Server PI, клиент может

отправлять на сервер команды. FTP-команды определяют параметры соединения передачи: роли

участников соединения (активная или пассивная), порт соединения (как для User DTP, так и для

Server DTP), тип передачи, тип передаваемых данных, структуру данных и управляющие директивы, обозначающие действия, которые пользователь хочет совершить, например сохранить, считать,

добавить или удалить данные или файл.

3. После согласования всех параметров работы канала передачи данных один из участников соединения, который является пассивным (например, клиентский модуль User DTP), переходит в режим

ожидания открытия соединения на заданный для передачи данных порт. После этого активный

модуль (например, Server DTP) открывает соединение и начинает передачу данных.

4. После окончания передачи данных соединение между Server DTP и User DTP закрывается, но

управляющее соединение Server PI – User PI остается открытым. Пользователь, не закрывая сессии

FTP, может еще раз открыть канал передачи данных, передать необходимую информацию и т.д.

## **46. NAT: основные функции, типы и принципы функционирования, влияние на приложения.**

**NAT (Network Address Translation)** – в общем случае технология изменения source или destination IP

адреса в пакете на какой-то другой. NAT позволяет нескольким хостам выходить в Интернет, используя один и тот же реальный IP адрес.

**Основные функции:**

* Позволяет сэкономить IP-адреса, транслируя несколько внутренних IP-адресов в один внешний публичный IP-адрес (или в несколько, но меньшим количеством, чем внутренних).
* Позволяет предотвратить или ограничить обращение снаружи к внутренним хостам, оставляя возможность обращения из внутренней сети во внешнюю. Если для пакетов, поступающих из внешней сети, соответствующей трансляции не существует (а она может быть созданной при инициации соединения или статической), они не пропускаются.
* Позволяет скрыть определенные внутренние сервисы внутренних хостов/серверов. Тем самым, снаружи, на внешнем IP-адресе после трансляции адресов на сайт (или форум) для осведомленных посетителей можно будет попасть по адресу <http://dlink.ru:54055>, но на внутреннем сервере, находящимся за NAT, он будет работать на обычном 80-м порту.

**Базовые концепции трансляции адресов:**

**Статический NAT** – Отображение незарегистрированного IP-адреса на зарегистрированный IP-адрес

на основании один к одному. Особенно полезно, когда устройство должно быть доступным снаружи сети.

**Динамический NAT** – Отображает незарегистрированный IP-адрес на зарегистрированный адрес

от группы зарегистрированных IP-адресов. Динамический NAT также устанавливает непосредственное

отображение между незарегистрированным и зарегистрированным адресом, но отображение может меняться в зависимости от зарегистрированного адреса, доступного в пуле адресов, во время коммуникации.

**Перегруженный NAT (NAPT, NAT Overload, PAT, маскарадинг)** – форма динамического NAT, который отображает несколько незарегистрированных адресов в единственный зарегистрированный IP-адрес, используя различные порты.

**Типы:**

**Симметричный NAT (Symmetric NAT)** — трансляция, при которой каждое соединение, инициируемое парой «внутренний адрес: внутренний порт» преобразуется в свободную уникальную случайно выбранную пару «публичный адрес: публичный порт». При этом инициация соединения из публичной сети невозможна.

**Cone NAT, Full Cone NAT** — однозначная (взаимная) трансляция между парами «внутренний адрес: внутренний порт» и «публичный адрес: публичный порт». Любой внешний хост может инициировать соединение с внутренним хостом (если это разрешено в правилах межсетевого экрана).

**Address-Restricted cone NAT, Restricted cone NAT** — постоянная трансляция между парой «внутренний адрес: внутренний порт» и «публичный адрес: публичный порт». Любое соединение, инициированное с внутреннего адреса, позволяет в дальнейшем получать ему пакеты с любого порта того публичного хоста, к которому он отправлял пакет(ы) ранее.

**Port-Restricted cone NAT** — трансляция между парой «внутренний адрес: внутренний порт» и «публичный адрес: публичный порт», при которой входящие пакеты проходят на внутренний хост только с одного порта публичного хоста — того, на который внутренний хост уже посылал пакет.

**Минусы NAT:**

* Не все протоколы могут "преодолеть" NAT.
* Из-за трансляции адресов "много в один" появляются дополнительные сложности с идентификацией пользователей и необходимость хранить полные журналы записей о трансляциях.
* Атака DoS со стороны узла, реализующего NAT – если NAT используется для подключения многих пользователей к одному и тому же сервису, это может вызвать иллюзию DoS-атаки на сервис (множество успешных и неуспешных попыток).

## **47. Устройство ЦОД. Понятие облачных вычислений. Виртуализация и масштабирование.**

**Центр обработки данных (ЦОД)** – это отказоустойчивая комплексная централизованная система,

обеспечивающая автоматизацию бизнес-процессов с высоким уровнем производительности и качеством

предоставляемых сервисов. В настоящее время в коммерческой среде наряду с определением **Центр**

**обработки данных (ЦОД)** используется термин – **Дата Центр (ДЦ)**.

**Типичный дата-центр состоит из:**

* **Информационной инфраструктуры,** включающей в себя серверное оборудование и обеспечивающей основные функции дата-центра — обработку и хранение информации;
* **Телекоммуникационной инфраструктуры,** обеспечивающей взаимосвязь элементов дата-центра, а также передачу данных между дата-центром и пользователями;
* **Инженерной инфраструктуры,** обеспечивающей нормальное функционирование основных систем дата-центра.

**Основные задачи, решаемые установленными в ЦОД инженерными системами:**

* **Обеспечение функционирования технологического оборудования:** системы электроснабжения, вентиляции и кондиционировании, Структурированные Кабельные Системы (СКС).
* **Защита от технических сбоев:** системы автоматического оповещения и тушения пожара, система автоматизации и диспетчеризации.
* **Защита от несанкционированных действий человека:** охранная сигнализация, видеонаблюдение, контроль доступа.

**Облачные вычисления (Cloud computing)** – технология распределенной обработки данных, при которой некие масштабируемые информационные ресурсы и мощности предоставляются как сервис для многочисленных внешних клиентов посредством Интернет-технологий.

**Концепция облачных вычислений (Модели обслуживания):**

* IaaS (Infrastructure as a Service или «Инфраструктура как сервис») – компьютерная инфраструктура, как правило, представленная в форме виртуализации.
* PaaS (Platform as a Service или «Платформа как сервис») – интегрированная платформа для разработки, развертывания, тестирования и поддержки web-приложений.
* SaaS (Software as a service или «ПО как сервис») – представляет собой бизнес-модель лицензионного использования ПО, которая подразумевает разработку и поддержку программного обеспечения поставщиком.

**Достоинства облачных вычислений:**

* Снижаются требования к вычислительной мощности ПК; Отказоустойчивость; Безопасность; Высокая скорость обработки данных; Снижение затрат на аппаратное и программное обеспечение, на обслуживание и электроэнергию; Экономия дискового пространства

**Недостатки облачных вычислений:**

* Зависимость сохранности пользовательских данных от компаний, предоставляющих услугу cloud computing и Появление новых («облачных») монополистов.

**Модели развёртывания:**

* **Частное облако (англ. private cloud)** – инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей
* **Публичное облако (англ. public cloud)** – инфраструктура, предназначенная для свободного использования широкой публикой. Публичное облако может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций
* **Общественное облако (англ. community cloud)** – вид инфраструктуры, предназначенный для использования конкретным сообществом потребителей из организаций, имеющих общие задачи
* **Гибридное облако (англ. hybrid cloud)** – это комбинация из двух или более различных облачных инфраструктур, связанных между собой стандартизованными или частными технологиями передачи данных и приложений

**Виртуализация** – это динамическая подмена физической подсистемы с определенными сервисами, на объект, имеющий такой же интерфейс и реализующий тот же самый набор сервисов.

**Масштабирование** – это способность системы, сети или процесса увеличивать свою производительность при добавлении ресурсов. **Вертикальное масштабирование** – увеличение производительности каждого компонента системы; **Горизонтальное масштабирование** – разбиение системы на более мелкие структурные компоненты и разнесение их по отдельным физическим машинам, и (или) увеличение количества серверов, параллельно выполняющих одну и ту же функцию.

## **48. Современные проблемы компьютерных сетей Программно Конфигурируемые Сети (ПКС): структура, принципы функционирования, протокол Open Flow.**

**Сеть** - платформа по предоставлению широкого спектра услуг.

**Проблемы** современных компьютерных сетей:

* **Современные сети проприетарны**: диктат производителя. Программное обеспечение тесно связано с железом (т. е. сетевым оборудованием, на котором оно развернуто), следовательно, производитель телекоммуникационного оборудования диктует оператору связи, как ему следует организовать работу, влияет на конкуренцию операторов между собой.
* **Закрытость для инноваций**: внесение любых изменений трудоемко, дорогостояще, длительно по срокам. Для разработки нового сервиса и его внедрения в сеть необходимо оплатить труд разработчиков, поэтому возникает проблема увеличения срока ROI - Return of Investment.
* **Сложность**: свыше 600 используемых протоколов, более 10 000 RFC, (Request for Comments = стандарты, технические спецификации, применяемые в сети)
* **Число middleboxes растет постоянно** (middleboxes - специальное сетевое оборудование, проверяющее и преобразующее трафик для целей, примерами являются различные сетевые экраны, трансляторы адресов, балансировщики нагрузки, системы обнаружения вторжений, анализаторы пакетов, транскодеры)
* **Нет надежных решений по безопасности**
* **Невозможно контролировать** и надежно предвидеть **поведение таких сложных объектов**, как глобальные компьютерные сети (ping, traceroute - классические методы контроля). В традиционных сетях уровень автоматизации работы достаточно низкий.

**ПКС** (Программно-конфигурируемые сети ) — это концепция построения сети, в которой **контур управления сетью** (control-plane) централизован и отделен от **контура передачи данных** (data-plane). Согласно концепции ПКС вся логика управления сетью сосредоточена в контроллере — программной платформе, которая работает на выделенном типовом сервере и способна отслеживать работу всех сетевых устройств (коммутаторов, портов, линий связи и т. п.) в контуре передачи данных сети и программно управлять ими. Контроллер агрегирует информацию о характеристиках коммутаторов, топологии сети, загрузке физических линий и устройств, об установленных соединениях между устройствами пользователей в сети.

В отличие от традиционных сетей в ПКС-сети у контроллера есть полная информация о состоянии сети. На рис. 1.1 представлена структура ПКС-сети. Контроллер образует централизованную программную платформу для реализации логики управления сетью. В среде контроллера, подобно традиционным операционным системам, работают приложения, которые, используя интерфейс контроллера, управляют параметрами состояния устройств в контуре передачи данных. Приложения контроллера могут предоставлять разнообразные сервисы управления сетью, например управлять маршрутизацией пакетов, контролировать доступ к сетевым ресурсам, балансировать нагрузку в сети и т. п. Ядро контроллера, образующего контур управления, предоставляет приложениям унифицированный, открытый интерфейс и службы по управлению программируемой сетевой инфраструктурой.

## 

У ПКС-контроллера можно выделить три вида интерфейсов: **южный** — для связи с устройствами в контуре передачи данных; **северный** — для приложений контроллера; **западно-восточный** — для администрирования сети и связи между контроллерами.

Архитектура ПКС-сети состоит из контуров управления и контура данных:

**Контур данных (Data Plane)** отвечает за обработку пакетов при передаче данных по маршруту, который был определен в контуре управления. Операции контура данных включают пересылку пакета на определенный порт коммутатора, изменение заголовка пакета либо сброс пакетов.

**Контур управления (Control Plane)**, который может быть распределенным (соединяются сервера, на которых крутятся экземпляры OS, каждая такая конфигурация управляет своим контуром передачи данных), отвечает за настройку и программирование сетевых устройств, принятие решений о том, как и какие пакеты должны пересылать и на какие сетевые устройства.

**Блок администрирования (Management Plane)** отвечает за мониторинг, настройку и администрирование сетевых устройств (например, принятие решений относительно состояния сетевого устройства, его конфигурации).

**Блок приложений (Application Plane)** охватывает приложения и службы, определяющие поведение устройств в контуре данных.

**Сетевое устройство** — это объект, принимающий пакеты на своих портах и выполняющий над этими пакетами определенные действия. Например, сетевое устройство может пересылать принятые пакеты, сбрасывать их, изменять у них поля заголовков и т. п. В каждом сетевом устройстве должен быть агент для связи с контуром управления и блоком администрирования.

OpenFlow-протокол используется для программирования и администрирования сетевых устройств: с помощью этого протокола осуществляется взаимодействие контроллера с коммутатором в ПКС.

(Подробнее в следующем билете)

## 

**Основное преимущество ПКС-сети** — гибкость управления, которое позволяет (за счет разделения контура управления и контура передачи данных) эффективно маршрутизировать, точечно выбирая маршрут, избегать пересечений маршрутов, избегать широковещательных рассылок по сети, существенно снизить время сходимости в сети. ПКС-контроллер видит  топологию сети как на L2-уровне, так и на L3-уровне, что принципиально невозможно в традиционной сети.

## **49. Протокол Open Flow, организация и принципы работы ПКС коммутатора, маршрутизация в ПКС сетях.**

**Openflow** – протокол управления процессом обработки данных, передающихся по сети передачи

данных маршрутизаторами и коммутаторами, реализующий технологию программно-конфигурируемой

сети. (стандарт для взаимодействия коммутатора с контроллером)

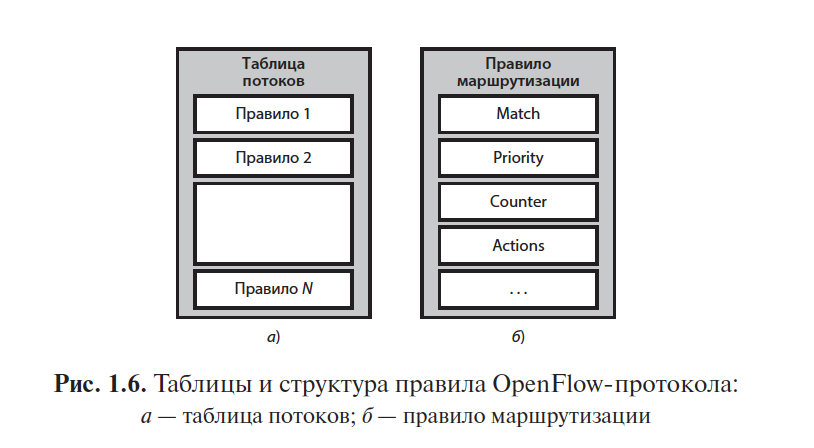
Протокол используется для управления сетевыми коммутаторами и маршрутизаторами с центрального устройства — контроллера сети (например, с сервера или даже персонального компьютера). Это управление заменяет или дополняет работающую на коммутаторе (маршрутизаторе) встроенную программу,

осуществляющую построение маршрута, создание карты коммутации и т.д. Контроллер используется

для управления таблицами потоков коммутаторов, на основании которых принимается решение о передаче принятого пакета на конкретный порт коммутатора. Таким образом в сети формируются прямые

сетевые соединения с минимальными задержками передачи данных и необходимыми параметрами.

Протокол OpenFlow предполагает наличие на коммутаторах одной или нескольких таблиц коммутации (рис. 1.6), в которые контроллер с помощью этого протокола может загружать правила обработки и передачи пакетов (рис. 1.6, a), проходящих через коммутатор. Такие таблицы называются таблицами потоков (flow-таблицами). У коммутатора может быть несколько таблиц потоков, которые образуют конвейер и нумеруются, начиная с нуля. Пакет, проходя через такой конвейер, подвергается обработке согласно правилам обработки и передачи, заранее загруженными контроллером в таблицы. Запись в каждой таблице потоков коммутатора содержит правило. Каждое правило состоит из поля признаков заголовка пакетов потока (шаблон заголовка), счетчиков и набора инструкций (рис. 1.6, б), т. е. действий, которые должны быть выполнены над пакетами, заголовки которых совпадают с полем шаблон заголовка.



При получении первого пакета нового потока OpenFlow-коммутатор  вырезает  из него битовую строку, которая охватывает все заголовки пакета от уровня L2 до уровня L4 в коммутаторе. Поиск записи для входящего пакета начинается с таблицы с номером 0. Если соответствующая запись в этой таблице найдена, то обновляются счетчики, соответствующие этой записи, и выполняются действия над пакетом. Далее возможен переход к следующей таблице, где процедура поиска записи повторяется, либо возможен выход из конвейера с выполнением набора действий (action set).

В случае если запись о потоке не была найдена в таблице 0, то в зависимости от конфигурации коммутатора он может выполнить одно из следующих действий:

* пакет отправляется на контроллер;
* пакет сбрасывается;
* продолжается поиск записи в следующей по номеру таблице.

Поиск в таблицах может быть реализован разными средствами. Наиболее распространенным является использование TCAM-памяти — ассоциативной аппаратной памяти. Ассоциативное ЗУ, АЗУ (англ. associative memory, content-addressable memory, CAM) — вид памяти, в котором адресация осуществляется на основе содержания данных, а не их местоположения.

Протокол OpenFlow поддерживает три типа сообщений:

1. **сообщения контроллер—коммутатор**. Контроллер с помощью этих сообщений конфигурирует и мониторит состояния коммутаторов, включая и управление таблицами правил коммутатора. Основными сообщениями этого типа являются:

* **Features** — запрос текущих параметров коммутатора: число таблиц правил, поддерживаемые действия и т. п.;
* **Configuration** — запрос и изменение настроек коммутатора, например изменение скорости работы портов;
* **Modify-State** (flow-mod) — добавление/удаление/изменение правил в таблицах коммутатора;
* **Read-State** (multipart request) — запрос статистики по портам коммутатора, списка правил в таблицах и др.;
* **Packet-out** — запрос на отправку одного конкретного пакета с указанного порта коммутатора;
* **Role-Request** — сообщение о смене роли контроллера по отношению к коммутатору. Разрешаемые роли — master, slave, equal. Сообщения этого типа нужны для организации работы распределенного контура управления

1. **симметричные сообщения** — это сообщения, которые может отправлять как контроллер, так и коммутатор. Служат для инициализации установки соединения и для обнаружения проблем соединения контроллера с коммутатором. Основные сообщения:

* **Hello** — первое сообщение при начале общения контроллера с коммутатором. При инициализации соединения коммутатор посылает контроллеру сообщение Hello, указав поддерживаемые версии протокола OpenFlow. В ответ контроллер посылает ответное сообщение Hello, указав выбранную для общения версию протокола. Дальше контроллер и коммутатор начинают общаться, используя остальные сообщения OpenFlow, например, как правило, следом всегда идет запрос от контроллера Feature Request для получения от коммутатора в Feature Reply поддерживаемого устройством функционала;
* **Echo** — служебное сообщение для проверки состояния соединения.

1. **асимметричные сообщения** отправляет коммутатор контроллеру с целью сообщить об изменении своего состояния. Основные виды сообщений этого типа:

* **Packet-in** — одно из самых важных сообщений протокола OpenFlow. Оно информирует контроллер о приходе пакета, для которого не нашлось записи в таблицах правил. Это происходит, например, в случае установки нового сетевого соединения пользователем в сети (например, открытие новой страницы сайта).
* **Flow-removed** — информирование контроллера об удалении правила с коммутатора. Это возможно, если коммутатор находится под управлением нескольких контроллеров или в случае истечения срока жизни правила, когда при установке правила был установлен срок его жизни: hard\_timeout — время в секундах, через которое надо удалить правило с коммутатора;
* **Port-status** — коммутатор сообщает об изменении состояния своих сетевых портов: включен/выключен административно из консоли устройства, есть ли несущая на канале (вставлен ли сетевой провод или есть ли кто-то на его другом конце);
* **Error** — информация о любой ошибке в работе коммутатора.