ОТВЕТЫ К ЭКЗАМЕНУ

по курсу "Архитектура компьютерных сетей"

1. Компьютерные сети - определение, структура, составные части, топология, связность, маршруты

Определение:

Сеть - система узлов (nodes), связанных друг с другом звеньями (links)
Строго говоря, компьютерной сетью считается любое соединение компьютеров (хотя бы двух), позволяющее передавать между ними данные.
Main функция сети - доставлять payload до заданных endpoints.

Структура (то, как устроена):

Сеть - система нодов, связанных друг с другом звеньями (links)

Точка присоединения узла к звену называется (сетевой) интерфейс
Способ соединения узлов звеньями называется топологией сети
В зависимости от топологии у разных узлов может быть различное количество интерфейсов, присоединённых к соответствующим звеньям.
На узлах сети могут располагаться конечные точки (endpoints), через которые в сеть может поступать (для доставки) полезная нагрузка (payload, груз)

Основная функция сети - доставлять payload до заданных endpoints.

Составные части:

Узел - это любое устройство, подключенное к сети и способное отправлять, получать или пересылать данные.
Endpoint (конечная точка) — это специфический узел в сети, который служит конечной точкой для связи.
Звено (link) — это физическая или логическая связь между двумя узлами в сети.
Интерфейс - это точка соединения между звеном и узлом, которая позволяет передавать данные

Топология:

Топология сети - это физическое расположение устройств в сети и способ их соединения.
Сетевая тополо́гия (анг. Network design) — это конфигурация графа, вершинам которого соответствуют конечные узлы сети- (компьютеры и маршрутизаторы), а рёбрам — физические или информационные связи между вершинами.
то есть как взаимодействуют эндпоинты и линки внутри сети.

Элементы топологии:

Линейная (Daisy chain)
Дерево (Tree) - сеть имеет иерархическую структуру, где каждый уровень связан с родительским уровнем.
Кольцо (Ring) - устройства соединены между собой кольцевой линией.
Шина (Bus) - все устройства подключены к одной линии передачи данных.

Связность:

Узлы называются связными (have connectivity) если между ними имеется, по крайней мере, один путь передачи трафика по сети.
Связность - это свойство сети, определяющее степень доступности каждого устройства в сети. Чем выше связность, тем более надежна сеть и лучше она способна передавать данные между компьютерами.
Если любая пара узлов сети обладает связностью, то сеть называется полносвязной (full connectivity).
Многосвязность - это наличие нескольких альтернативных маршрутов

Топология сети **связная**, если между любой парой узлов существует маршрут.

Маршрут

Маршрут в связной сети — это последовательность узлов и связей (каналов передачи данных), по которым данные могут передаваться от одного узла к другому.
Маршрут обеспечивает возможность обмена информацией между любыми двумя узлами сети.
Маршрут - описание пути передачи трафика по сети от одного узла до другого
Узлы называются связными (have connectivity) если между ними имеется, по крайней мере, один путь передачи трафика по сети.

2. Назначение компьютерных сетей: сетевые службы, принципы их устройства, роль компьютерных сетей в их функционировании

Сервис - это:

1. Оказание помощи или выполнение работы для кого-либо (ресторан, салон красоты,автомойка...)
2. Система, обеспечивающая какую-либо потребность, востребованную обществом.

Цель существования комьютерных сетей - предоставить возможность пользователям использовать сетевые сервисы.

Сетевые службы (сервисы)

Компьютерные сети реализуют базовый сервис по доставке данных между endpoints, располагающимися на территориально распределенных узлах

Endpoint - это объект ОС, динамически создаваемый программным кодом в процессе исполнения.

Функционирующие на злах сети программные модули (приложения, сервисы) через Endpoint имеют возможность отправлять и получать порции данных, а сеть обеспечивает транспортировку и доставку этих порций до указанных Endpoints.
Благодаря этому разрабатывается сетевое ПО и его пользователи (люди) получают полезные сетевые службы (Эл. почта, WEB-сайты, поисковые системы, соц. сети, интернет-магазины)

Компьютерные сети (aka сети передачи данных) реализуют функционал (сервис) по
оперативной доставке между endpoint-ами данных, представленных в цифровом
виде и, обычно, несущих информацию обладающую некоей ценностью для
пользователей (подробнее в следующем пункте)

Принципы их устройства

Специальные процессы операционной системы (демоны, службы) создают «слушающий» сокет и «привязывают» его к определённому порту (пассивное открытие соединения),
обеспечивая тем самым возможность другим компьютерам обратиться к данной службе. Клиентская программа или процесс создаёт запрос на открытие сокета с указанием
IP-адреса и порта сервера, в результате чего устанавливается соединение, позволяющее взаимодействовать двум компьютерам с использованием соответствующего
сетевого протокола прикладного уровня.

Роль компьютерных сетей в их функционировании

Взаимодействие компьютеров между собой, а также с другим активным сетевым оборудованием, в TCP/IP-сетях организовано на основе использования сетевых служб,
которые обеспечиваются специальными процессами сетевой операционной системы (ОС) — демонами в UNIX-подобных ОС, службами в ОС семейства Windows и т. п.
Примерами сетевых сервисов являются веб-серверы (в т.ч. сайты всемирной паутины), электронная почта, FTP-серверы для обмена файлами,
приложения IP-телефонии и многое другое.

В кратце - компьютерные сети реализуют лишь базовый сервис по доставке данных между Endpoints, располагающимися на территориально распределенных узлах

3. Понятие и цель передачи данных: информация vs данные, представление данных, способы передачи данных

Компьютерные сети (aka сети передачи данных) реализуют функционал (сервис) по
оперативной доставке между endpoint-ами данных, представленных в цифровом
виде и, обычно, несущих информацию обладающую некоей ценностью для
пользователей

Понятие передачи данных

✔ Носитель данных — материальное тело или физическое явление, которое
специальным образом изменяется/модифицируется для того, чтобы представлять
данные
✔ Передача данных - процесс/процедура, в результате которой какой-либо набор
данных становится известным в другой точке пространства:
➔ путём физического перемещения носителя данных, с нанесённым на него
набором данных (к сетям передачи данных это не относится!)
➔ путём распространения по среде передачи специально изменяющиегося во
времени физического сигнала-носителя

Информация vs Данные

```
✔ Информация – знания относительно фактов, событий, вещей,
идей, понятий и
т. п. Информация нематериальна, она существует только в виде
идей в мозге и
строго говоря, неотделима от субъекта (human beings)
✔ Данные – способ (форма) представления/регистрации информации
в виде,
пригодном для восприятия (интерпретации), хранения, обработки и
коммуникации (т. е. социального поведения, в результате
которого одни субъекты узнают информацию от других субъектов)
```

Представление данных

Данные могут быть представлены в различных форматах, которые зависят от типа данных и способов их использования. Некоторые основные форматы включают:

```
Числовые данные: Целые числа, вещественные числа и другие
числовые представления.
Текстовые данные: Символы, строки, документы.
Графические данные: Изображения, видео, графики.
Звуковые данные: Аудиофайлы, звуковые записи.
Бинарные данные: Данные, представленные в виде двоичных чисел
(например, файлы, исполняемые программы).
```

Способы передачи данных

Передача данных – процесс/процедура, в результате которой какой-либо набор
данных становится известным в другой точке пространства:
➔ путём физического перемещения носителя данных, с нанесённым

на него
набором данных (к сетям передачи данных это не относится!)
➔ путём распространения по среде передачи специально изменяющиегося во
времени физического сигнала-носителя

4. Системы (звенья) передачи данных: структура, общие принципы их функционирования, задействованные процессы (передача, приём, распространение), роль в составе компьютерных сетей

Передача данных (информации) – процедура, в результате которой какой-либо
набор информации становится известным в другой точке пространства
Способы передачи данных (информации):
• Физическое перемещение носителя информации
• Распространение физического сигнала-носителя, который промодулирован по
среде передачи

Звено сети передачи данных – техническое устройство, выполняющее передачу
данных (информации) между соединяемыми им узлами сети без необходимости
перемещения носителя

Принцип устройства звена передачи данных:
• Передатчик-модулятор, формирующий сигнал-носитель, модулированный

последовательными порциями данных
• Среда передачи данных, протянутая между узлами, по которой распространяется модулированный сигнал-носитель
• Приёмник-демодулятор, выделяющий данные из принятого сигнала-носителя

Общие принципы функционирования систем передачи данных включают следующие процессы:

```
Передача: Отправка данных от источника к приёмнику.
Приём: Получение данных приёмником.
Распространение: Перемещение данных по медиа передачи.
```

Роль в составе компьютерных сетей:

```
Связь: Обеспечение связи между пользователями и системами для обмена информацией.
Доступ к ресурсам: Обеспечение удалённого доступа к базам данных, веб-сайтам и другим ресурсам.
Управление и контроль: Централизованное управление устройствами и системами.
```

Звено передачи данных является "мостом" между узлами для транспортировки данных по маршруту передачи

5. Системы (звенья) передачи данных: техническое устройство, метод передачи данных, варианты (синхронные/асинхронные, симлексные/дуплексные/

полудуплексные), скорость передачи данных, задержка доставки данных

Устройство

Логически звено передачи данных (data link) рассматривается как совокупность:
➔ двух модемов (DCE — data communication equipment)
➔ линии связи по которой между узлами распространяется модулированный
сигнал

Модем состоит из модулятора/передатчика, обеспечивающего формирование
сигнала носителя из передаваемого потока данных и демодулятора/приёмника,
обеспечивающего распознавание в принятом сигнале модулированных данных и
формирование принимаемого потока данных

Метод передачи данных:

Модулятор генерирует и передаёт в линию последовательность различимых сигналов
определённой длительности T_s. Каждый сигнал это символ из заранее определённого
алфавита {S1,...SK}.

✔ Какие именно символы из алфавита передаются в последовательности,
определяется содержимым передаваемого потока данных
✔ Частота формирования символов определяется модулятором и называется

частотой манипуляции Vm=1/TS [измеряется в Бодах]
✔ При размере алфавита K каждый символ может кодировать E=log2(K) бит входных
данных (обычно K выбирается из ряда 2, 4, 8, 16 ...). Таким образом, эффективная
скорость передачи данных по звену составляет V=Vm•log2(K) [бит/сек]

Варианты передачи данных:

По допустимости пауз в передаваемом потоке символов звенья разделяются на:
➔ асинхронные — допускающие паузы произвольной длительности в потоке
передачи
➔ синхронные — работающие без пауз: после активации такого звена, передача
осуществляется с постоянной скоростью V (даже если нет трафика всё равно
нужно заполнять канал каким-то «холостым» потоком битов)

Сетевой обмен данными, как правило, двухсторонний: необходимо и принимать
и передавать данные. Для реализации такой возможности применяют:
➔ Дуплексные звенья, использующие две отдельные линии (среды передачи) —
по одной в каждом направлении (такие звенья способны одновременно
передавать и принимать разные данные встречных направлений)
➔ Полудуплексные звенья, использующие одну среду, но с переключением
приём ⇿ передача
Симплексные звенья: Передача данных только в одном направлении (например, телевизионное вещание).

Скорость передачи данных:

```
Скорость передачи данных: Измеряется в битах в секунду (bps), влияет на объём передаваемой информации за единицу времени.
```

Чаще всего скорость передачи данных V [бит/сек] у конкретного звена фиксирована (например, у обычного Ethernet бывает 10 мбит/сек, 100 мбит/сек
или 1 гбит/сек). Исключения наблюдаются в тех случаях когда модем имеет
функцию автоподстройки скорости под условия линии связи (например, ADSL)

Задержка (Ping)

```
Задержка доставки данных: Время, необходимое для перемещения данных от отправителя к получателю. Влияет на качество связи, особенно в реальном времени
```

6. Мультиплексирование потоков данных: назначение, принципы реализации. Варианты статического мультиплексирования: TDM, FDM.

Определение

Мультиплексирование – объединение нескольких отдельных независимых потоков
в единый комбинированный поток для передачи по звену

Смешать, но не перемешивать – комбинированный поток должен обладать
различимой структурой, позволяющей без потерь разделить его на исходные потоки в демультиплексоре

Назначение

В связи с тем, что вычислительные сети используются для передачи данных на большие расстояния,
то стремятся минимизировать количество проводов в кабеле, в целях экономии
. Поэтому разрабатывались технологии, которые позволяют передавать, по одному
и тому же каналу связи, сразу несколько потоков данных.

Принципы реализации

Статическое мультиплексирование — фиксированное разбиение одного
канала со скоростью V [бит/сек] на n подканалов со скоростью V/n: TDM, FDM

Статистическое (aka динамическое) мультиплексирование — передача по
очереди на полной скорости порций данных разных подканалов

Варианты статического мультиплексирования

TDM (Time division multiplexing) - мультиплексирование по времени - используется для передачи данных в
последовательности, где каждый канал имеет определенный интервал времени

для передачи данных.
Каждый поток данных прерывается через равные промежутки времени,
чтобы позволить другим потокам данных использовать канал.
В результате, все потоки данных передаются
последовательно на одном канале.

Высокоскоростной изохронный комбинированный поток битов со скоростью V
[бит/сек] формируется методом поочерёдной передачи битов из (n) менее
скоростных потоков со скоростями Vi = V/n
✔ Комбинированный поток представляет собой непрерывную последовательность
циклов, каждый из которых содержит фиксированное количество битов каждого
из исходных потоков в определённом порядке (обычно в форме чередования
одиночных битов каждого из потоков по порядку их номеров)
✔ Для корректной работы звена необходимо, чтобы приёмник и передатчик были
синхронизированы, т. е. воспринимали начало цикла в один и тот же момент
✔ Преобразование потоков может осуществляться в цифровом виде (т. е. на уровне
демодулированного потока данных, а не аналогового сигнала)

FDM (frequency division multiplexing) - частотное мультиплексирование использует
различные частотные диапазоны для передачи нескольких потоков данных.
Каждый поток данных имеет свой уникальный частотный диапазон, который отличается

от других потоков данных. Эти диапазоны разделены, чтобы не пересекаться с
другими
диапазонами потоков данных, что позволяет каждому потоку данных передаваться
через физический канал параллельно.

Ресурс (полоса пропускания) широкополосного тракта распространения
сигналов делится на несколько полос, внутри каждой из которых организуется
отдельный тракт для передачи узкополосного сигнала.
Преобразование осуществляется на уровне сигнала (в аналоговом виде!)

7. Динамическое (статистическое) мультплексирование: принцип действия, отличия от статического мультиплексирования ---(буферизация, store & forward, инкапсуляция)

Определение

Мультиплексирование – объединение нескольких отдельных независимых потоков
в единый комбинированный поток для передачи по звену

Статистическое мультиплексирование - это метод передачи данных, который позволяет разделять полосу пропускания физического канала на различные потоки данных в зависимости от текущей загрузки. Он используется для оптимального использования доступной пропускной способности канала и увеличения эффективности передачи данных.

Принципы действия

Сетевой трафик складывается из множества индивидуальных потоков данных
(каждый между определёнными endpoint-ами)

✔ каждый поток данных рассматривается как транспортировка через сеть
законченных порций данных — сообщений (генерируемых endpoint-ами)
✔ сообщения между узлами по звеньям передаются только целиком: с начала и до
конца без пауз
✔ разные сообщения могут передаваться по звену последовательно друг за другом:
пока идёт передача данных текущего сообщения, другие сообщения (если они
имеются) должны ожидать своей очереди на отправку
✔ пропускная способность звеньев динамически распределяется между потоками:
паузы в одних потоках могут заполняться сообщениями других потоков

Суть решения:
✔ узлы оборудуются (промежуточной) памятью, в которой выделяются буферы,
способные хранить целое сообщение
✔ любое поступающее сообщение (от соседнего узла через внешний интерфейс,
либо от локального endpoint-а) сначала размещается в буфере и только потом
коммутируется
✔ применяется метод Store and Forward

Отличия между статистическим и статическим мультиплексированием

Буферизация

Порции трафика могут поступать в произвольные моменты времени и приниматься со скоростью, не совпадающей со скоростью передачи исходящего звена

Из-за чего возникают конфликты:

Конфликт 1: передача порции трафика по исходящему звену может начинаться только после завершения приема всей единицы от входящего звена из-за различия скоростей

Конфликт 2: единицы трафика (C,D) не могут быть немедленно переданы по исходящему звену, поскольку оно в эти моменты занято передачей единицы трафика (P1)

Для разрешения этих конфликтов порции трафика сначала принимаются в буферную память и только затем перенаправляются в очередь соответствующего исходящего звена или Endpoint-a → Store & Forward

Буфер - место в памяти, способное хранить целое сообщение
Таким образом, любое поступающее сообщение (от соседнего узла через внешний
интерфейс, либо от локального endpoint-a) сначала размещается в

буфере и
только потом коммутируется

Store and Forward

✔ принимаемые по звену данные, байт за байтом, размещаются во входном буфере
(Store — запомни/сохрани)
✔ в момент, когда последний байт порции (сообщения) принят и размещён в
буфере, выполняется процедура коммутации: анализируются данные принятого
сообщения и, в зависимости от избранного варианта, буфер либо ставится в
очередь требуемого исходящаго звена, либо доставляется в локальный endpoint,
либо сбрасывается (Forward – направь)
✔ передатчики работают асинхронно: последовательно передавая порции,
находящиеся в выходной очереди данного исходящего канала (пока она не пуста)

Инкапсуляция

Если передавать исходные единицы трафика в канал “как есть”, то приемник не
сможет отделить порции друг от друга (это будет просто сплошной поток битов с
возможными паузами)
Кроме того, коммутатору на приемном узле требуется информация о том, к какому
потоку относится эта порция и куда далее её отправлять. Поэтому

передатчику
необходимо добавлять в поток служебную информацию

К каждой порции добавляется спе-
реди заголовок (header), а сзади –
хвостовик (trailer) определенного
известного формата и размера

Заголовок и хвостовик выступают в роли конверта в который вкладываются пере-
даваемые (полезные) данные. Такой процесс называется инкапсуляция (encapsu-
lation). Коммутатор-приемник, ориентируясь на информацию содержащуюся в
заголовках и хвостовиках, осуществляет коммутацию

8. Транспортировка данных (полезной нагрузки) в сетях: понятие сетевого трафика, методы доставки трафика через сеть, роль и функции элементов сети в процессе доставки трафика

Понятие

Сетевой трафик – это поток полезной нагрузки (payload), транспортируемой по сети (аналогия: автомобильный трафик – это поток пассажиров и грузов, путешествующий по сети дорог)

Сетевой трафик это не сама полезная нагрузка, а процесс её транспортировки (аналогия: автомо-
бильный трафик это не сами автомобили, а про-
цесс их перемещения по сети дорог)

В компьютерных сетях (в отличие от автомобильных) всю активную роль
выполняют узлы сети, а звенья лишь пассивно транслируют данные, передаваемые узлами

Все узлы компьютерной сети должны согласованно выполнять свою часть
работы по доставке единиц полезной нагрузки, таким образом, совместно
организуя сетевой трафик

Узлы сети ведут себя (относительно) независимо друг от друга, а вся сеть
представляет собой систему массового обслуживания

Трафик состоит из множества отдельных единиц полезной нагрузки (units of
payload), одновременно транспортируемых по элементам сети: узлам и
звеньям

Метод доставки трафика через сеть

Процесс передачи сетевого трафика по звену начинается с того, что данные отправляются с одного узла или компьютера через канал связи в сети. Затем эти данные проходят через несколько промежуточных устройств, таких как маршрутизаторы и коммутаторы, чтобы достичь конечного узла назначения.
Принципы передачи трафика в сети зависят от режима передачи, который используется в данной ситуации. Режим передачи может быть прямой, косвенный или транзитный.
В режиме прямой передачи данные передаются от отправителя к получателю напрямую через выделенную связь. Этот режим используется, например, при звонке по телефону.

В режиме косвенной передачи данные передаются через несколько узлов сети, каждый из которых перенаправляет данные следующему узлу до тех пор, пока они не достигнут получателя. Этот режим используется, например, при отправке электронной почты.
В режиме транзитной передачи данные передаются через сеть, но не для доставки конечному получателю. Вместо этого они перенаправляются на другой узел сети для обработки или дальнейшей передачи. Такой режим передачи используется, например, при пересылке глобальных информационных пакетов в Интернете.

А поподробнее?

(См. картинку на стр. 19 АКС-2024-1a.pdf)

Node A и Node C могут транспортировать трафик между endpoint A3 ⇿ C1 напрямую, через общее звено A-C:
➔ Node A передаёт payload от A3 (→ сигнал распространяется) через звено A-C
➔ Node C принимает payload из звена A-C и доставляет его в C1
✔ Между Node A и Node F нет общего звена. Тем не менее передавать трафик между endpoint A1 ⇿ F1 возможно с транзитом на Node D:
➔ Node A передаёт payload от A1 (→ сигнал распространяется) через звено A-D
➔ Node D принимает payload из звена A-D и перенаправляет его в звено D-F (осуще-ствляет транзит «чужого» трафика)
➔ Node F принимает payload из звена D-F и доставляет его в F1

Таким образом, доставка трафика между endpoint требует согласованного выпол-
нения узлами трёх процессов: приема и передачи данных на интерфейсах звеньев,
а также коммутации трафика внутри узлов

9. Понятие коммутации: назначение и роль в сетях, варианты (inbound, outbound, transit...), методы коммутации

Коммутация в сетях связи представляет собой процесс, в ходе которого данные передаются от источника к получателю через сеть по определённым маршрутам. Основное назначение коммутации — обеспечение эффективной передачи данных между различными узлами сети.

Каждый сетевой узел должен обеспечивать
обработку поступающих порций трафика:
• при получении новых порций payload от
местных endpoint-ов
• при приёме порций от внешних сетевых
интерфейсов, подключенных к звеньям

Назначение и роль в сетях:

```
Назначение: Обеспечивать передачу данных по сети от источника к получателю, минимизируя задержки и потери данных.
Роль: Улучшение эффективности использования сетевых ресурсов, обеспечение надёжности и скорости передачи данных, управление потоками данных.
```

Варианты коммутации:

Примеры на рисунке (АКС-2024-1a.pdf, стр. 22):

При поступлении очередной порции возможен один из 4 вариантов её коммутации:
➢ *входящий (inbound) — порция доставляется для одного из местных endpoint*
➢ исходящий (outbound) — местный endpoint отправляет в сеть очередную порцию
➢ *транзитный (transit) — принятая порция перенаправляется на другой узел*
➢ заблокированый (blocked) — передача порции блокируется (это ошибка доставки)

Выбор того или иного варианта осуществляется для каждой поступившей порции трафика исходя из её даных и настроек узла.

Методы(технологии) коммутации:

Коротко:
Коммутация каналов: Устанавливается физический канал связи между двумя узлами на всё время сеанса связи.
Коммутация сообщений: Сообщения передаются целиком от одного узла к другому, при этом каждый узел принимает и передаёт их дальше.
Коммутация пакетов: Данные разбиваются на пакеты, которые передаются через сеть независимо друг от друга, и каждый пакет может следовать своим маршрутом до конечного пункта назначения.

Подробно:

Коммутация каналов

(Стационарная проводная) телефоная сеть (ТфОП - PSTN), появившаяся ещё в XIX
веке, стала первой коммуникационной сетью и в её рамках была создана и
развилась исторически первая технология коммутации каналов

Узлами ТфОП являются:
➔ телефонные аппараты
➔ телефонные станции

Звеньями ТфОП являются
телефонные линии разных видов -
АЛ, СЛ, МГЛ..

Единицей payload телефонной сети
выступает телефонное соединение
— ограниченный период времени,
в течение которого через сеть уста-
новлен сквозной разговорный тракт
между телефонами двух абонентов.
Именно за это время (до сих пор),
как правило, начисляется оплата

Принципы коммутации каналов

Перед началом соединения телефонные станции
устанавливают перемыч ки, формируя временный
разговорный тракт из цепи по маршруту между
абонентами

В течение всего периода соединения задействованные
линии остаются занятыми, через разговорный тракт

свободно распространяется сигнал между абонентами.

По окончании соединения перемычки снимаются, а задействованные сетевые ресурсы (линии, аппараты АТС) освобождаются для последующих соединений

Коммутация пакетов

Узел-отправитель разбивает исходное сообщение произвольного размера на
последовательность пронумерованных фрагментов – пакетов
✔ Размер пакета не может превышать некую зафиксированную величину P
max (чаще всего около 1500 байтов). Пакеты меньшего размера – допускаются
✔ Каждый пакет отправляется по маршруту и коммутируется независимо от других
пакетов сообщения как автономная единица трафика.
✔ Сборку исходного сообщения осуществляет узел-получатель в процессе приема
пакетов - фрагментов.
✔ Транзитным узлам нет необходимости буферизовать все сообщение. Каждый
успешно полученный фрагмент может быть немедленно поставлен в очередь на
отправку по дальнейшему маршруту (конвейерная передача).

Коммутация сообщений

Сообщения передаются целиком от одного узла к другому, при этом каждый узел принимает и передаёт их дальше.
Сети с коммутацией сообщений используют метод Store & Forward и динамическое мультиплексирование

✔ В качестве порций трафика выступают сообщения (messages) – «естественные»
порции данных прикладных сетевых служб
✔ В зависимости от вида службы (услуги) сообщения могут быть очень разными:
➔ SMS (140 (70) символов);
➔ Электронное письмо (от сотен байт до десятков мегабайт);
➔ Видеофайл (до десятков гигабайт).
✔ Большой (потенциальный) размер сообщения создает серьезные технические
проблемы:
➢ Буферная память каждого узла должна вмещать несколько (в зависимости от
глубины очереди) сообщений наибольшего возможного размера
➢ Передача длинного сообщения задерживает все остальные передачи на
данном звене

10. Коммутация каналов: устройство сетевых узлов, принцип действия, характеристики эффективности

В лекции рассмотрен пример телефонной сети

Для нормального функционирования сети каждый узел обязан выполнять
коммутацию трафика (единиц payload). Это именно то, что делает совокупность
отдельных узлов и звеньев связной сетью

✔ (Стационарная проводная) телефоная сеть (ТфОП - PSTN), появившаяся ещё в XIX

веке, стала первой коммуникационной сетью и в её рамках была создана и
развилась исторически первая технология коммутации каналов

Узлами ТфОП являются:
➔ телефонные аппараты
➔ телефонные станции

✔ Звеньями ТфОП являются
телефонные линии разных видов -
АЛ, СЛ, МГЛ...

✔ Единицей payload телефонной сети
выступает телефонное соединение
— ограниченный период времени,
в течение которого через сеть уста-
новлен сквозной разговорный тракт
между телефонами двух абонентов.
Именно за это время (до сих пор),
как правило, начисляется оплата

И ещё

Принцип КК появился задолго до появления компьютеров – телефонная сеть. Единица
трафика – сквозное соединение, обеспечивающее разговор между абонентами.

Сеть состоит из коммутационных
центров (АТС) связанных пучками
унифицированных соединительных
линий. Абонентские аппараты
присоединяются к ближайшей АТС
индивидуальными абонентскими

линиями.
Пропускная способность сети – количество одновременных соединений в час наибольшей нагрузки (ЧНН). Определяется количеством доступных линий в пучках.

Принцип коммутации каналов

Перед началом соединения телефонные станции устанавливают перемычки, формируя временный разговорный тракт из цепи АЛ, СЛ, МГЛ по маршруту между абонентами

В течение всего периода соединения задействованные линии остаются занятыми, через разговорный тракт свободно распространяется сигнал между абонентами

По окончании соединения перемычки снимаются, а задействованные сетевые ресурсы (линии, аппараты АТС) освобождаются для последующих соединений

Характеристики эффективности

Соединения, предоставляемые сетью КК, могут быть только стандартной скорости (в телефонной сети это узкополосные голосовые соединения 64000 бит/сек). Это Не универсально. В большинстве случаев ПД - слишком медленно.

Соединение точка-точка позволяет осуществлять обмен данными только с одним конкретным узлом. Для того, чтобы одновременно общаться с несколькими узлами необходимо открывать и поддерживать несколько соединений.
Типовой вариант использования СПД в стиле «WEB-surfing» затруднителен

Пока соединение установлено оно занимает ресурсы сети вне зависимости от того разговаривают абоненты или молчат.
Неэффективно используется пропускная способность сети, особенно при неравномерном трафике.

11. Коммутация сообщений: устройство сетевых узлов, принцип действия,

характеристики эффективности по отношению с коммутацией каналов

Сети с коммутацией сообщений используют метод Store & Forward и динамическое мультиплексирование
✔ В качестве порций трафика выступают сообщения (messages) – «естественные»
порции данных прикладных сетевых служб
✔ В зависимости от вида службы (услуги) сообщения могут быть очень разными:
➔ SMS (140 (70) символов);
➔ Электронное письмо (от сотен байт до десятков мегабайт);
➔ Видеофайл (до десятков гигабайт).
✔ Большой (потенциальный) размер сообщения создает серьезные технические
проблемы:
➢ Буферная память каждого узла должна вмещать несколько (в зависимости от
глубины очереди) сообщений наибольшего возможного размера
➢ Передача длинного сообщения задерживает все остальные передачи на
данном звене

Принцип действия

Коммутация сообщений (КС, message switching) — разбиение информации на сообщения, которые передаются последовательно к ближайшему транзитному узлу, который, приняв сообщение, запоминает его и передаёт далее сам таким же образом. Получается нечто вроде конвейера. Данные могут задерживаться на узлах до тех пор, пока не появится рабочее соединение.

Коммутация сообщений
Под коммутацией сообщений понимается передача единого блока данных между транзитными компьютерами сети с временной буферизацией этого блока на диске каждого компьютера. Сообщение в отличие от пакета имеет произвольную длину, которая определяется не технологическими соображениями, а содержанием информации, составляющей сообщение (текстовый документ, файл с кодом программы, электронное письмо). Транзитные компьютеры могут соединяться между собой как сетью с коммутацией пакетов, так и сетью с коммутацией каналов. Сообщение хранится в транзитном компьютере на диске, причем время хранения может быть достаточно большим, если компьютер загружен другими работами или сеть временно перегружена. По такой схеме обычно передаются сообщения, не требующие немедленного ответа, чаще всего сообщения электронной почты. Режим передачи с промежуточным хранением на диске называется режимом «хранение-и-передача» (store-and-forward). Техника коммутации сообщений появилась в компьютерных сетях раньше техники коммутации пакетов, но потом была вытеснена последней, как более эффективной по критерию пропускной способности сети.

Сравнение с КК

Потребность в объеме буферной памяти для КС - очень высокая (мб, гб), а вот для
КК буферная память не требуется
Макс. полоса пропускания, предоставляемая абоненту - для КС ограничивается
только каналами пердачи, а для КК всегда низкая (потому что соединение с фикс.
скоростью)
Работа с неравномерным трафиком КС поддерживает а КК нет

Вносимая задержка доставки сообщения у КС очень высокая (особенно для длинных
сообщений и/или большом количестве транзитов), а вот КК задержку не вносит
Степень утилизации пропускной способности каналов передачи - у КС - самая высокая среди всех коммутаций, а вот у КК низкая, так как из-за простаивающих соединений ресурсы не могут быть задействованы.
Гарантия наличия полосы пропускания между узлами - у КС не обеспечивается
(из-за монополизации каналов длинными сообщениями), а у КК обеспечивается
архитектурой сети
Возможность параллельного обмена данными с несколькими узлами - у КС не
ограничена, а у КК крайне ограничено - 1 или 2.

Может пригодиться

Устройство сетевых узлов:

```
Узлы, участвующие в коммутации сообщений, включают коммутаторы и маршрутизаторы, способные принимать, хранить и передавать целые сообщения.
Каждый узел имеет буферы для хранения сообщений до их полной передачи.
```

Принцип действия:

```
Сообщение передаётся целиком от источника к ближайшему узлу.
Узел принимает сообщение целиком, хранит его и затем передаёт следующему узлу, пока оно не достигнет получателя.
```

Характеристики эффективности:

```
Преимущества:
    Гарантированная доставка сообщений в порядке их отправки.
    Хорошая защита от ошибок, так как сообщения могут быть
проверены и переправлены заново в случае ошибок.
Недостатки:
    Высокие задержки при передаче сообщений из-за необходимости
их полного приёма и хранения на каждом узле.
    Большие требования к объёму памяти узлов, так как
необходимо хранить целые сообщения.
```

По сравнению с коммутацией каналов, коммутация сообщений более гибка и надёжна, но имеет большие задержки и требует больше ресурсов на хранение данных.

12. Коммутация пакетов: принцип действия, особенности, по сравнению с коммутацией сообщений

Принцип действия:

Узел-отправитель разбивает исходное сообщение произвольного размера на
последовательность пронумерованных фрагментов – пакетов
✔ Размер пакета не может превышать некую зафиксированную величину P
max (чаще всего около 1500 байтов). Пакеты меньшего размера – допускаются
✔ Каждый пакет отправляется по маршруту и коммутируется независимо от других
пакетов сообщения как автономная единица трафика.

✔ Сборку исходного сообщения осуществляет узел-получатель в процессе приема
пакетов - фрагментов.
✔ Транзитным узлам нет необходимости буферизовать все сообщение. Каждый
успешно полученный фрагмент может быть немедленно поставлен в очередь на
отправку по дальнейшему маршруту (конвейерная передача).

Особенности:

- Кардинальное снижение требований к необходимому объему буферной
  памяти в узлах
- Передача длинных сообщений не задерживает передачу другого трафика
  надолго
- Уменьшение общего времени доставки длинного сообщения за счет эффекта
  конвейерной передачи
  – Более сложный алгоритм работы оконечных узлов (проблемы корректной
  сборки сообщения из фрагментов)
  – Выше накладные расходы за счет добавления служебной информации к
  каждому пакету
  Преимущества настолько существенны, что на сегодняшний момент все функциони-
  рующие сети передачи данных построены по методу коммутации пакетов.
  Даже традиционная телефонная сеть сегодня развивается в направлении

технологии коммутации пакетов для передачи голосового трафика.

По сравнению с коммутацией сообщений:

Потребность в объеме буферной памяти: у КП умеренная (кб/мб), а у КС очень высокая (мб/гб)
Макс. полоса пропускания предоставляемая абоненту: у КС и КП ограничение только каналами передачи
Работа с неравомерным трафиком - оба поддерживают
Вносимая задержка доставки сообщения - у КС высокая, у КП минимальная за счет
конвейерного эффекта
Степень утилизации пропускной способности каналов передачи - у КС - самая высокая, а у КП проста высокая (потому что требуется передавать несколько больше
служебного трафика, чем при КС)
Гарантия наличия полосы пропускания между узлами - у КС не обеспечивается (из-за
монополизации каналов длинными сообщениями), а у КП может быть обеспечена
доп. средствами резервирования
Возможность параллельного обмена данными с несколькими узлами: у КС не ограничена, также как и у КП.

13. Характеристики (метрики) функционирования сетей передачи данных: объём трафика, пропускная способность, время доставки, вероятность

потери данных. Факторы, влияющие на величину этих характеристик

Характеристики:

```
Объём трафика: Количество данных, передаваемых через сеть за определённый период времени.
Пропускная способность: Максимальное количество данных, которое может быть передано через сеть за единицу времени.
Время доставки: Время, необходимое для передачи данных от источника к получателю.
Вероятность потери данных: Вероятность того, что данные будут потеряны во время передачи через сеть.
```

Факторы, влияющие на характеристики:

Как правило, улучшение одной характеристики сети может быть достигнуто за
счет ухудшения других. Правило «выберите не больше двух качеств из трех
возможных».
В целом, сеть представляет собой сложную систему массового обслуживания,
поведение которой имеет вероятностный характер, а характеристики являются
случайными величинами, для которых, в большинстве случаев, можно лишь
оценить параметры распределения (мат.ожидание, дисперсию и т.п.)

```
Объём трафика: Зависит от числа активных пользователей и приложений, требующих передачи данных.
```

```
Пропускная способность: Определяется характеристиками
оборудования (коммутаторов, маршрутизаторов), шириной канала
передачи данных, технологией передачи данных.
Время доставки: Влияет расстояние между узлами, число
промежуточных узлов, задержки на обработку данных каждым узлом,
состояние сети (загруженность, наличие ошибок).
Вероятность потери данных: Зависит от качества каналов связи,
надёжности оборудования, наличия помех, перегрузок сети.
```

14. Логическая модель сетевого сервиса: «черный ящик», SAP, адресация, жизненный цикл SAP. Дейтаграммный и потокоориентированный режимы.

Прикладные процессы, использующие сетевой API, рассматривают сеть
целиком как некий «черный ящик» предоставляющий услугу (service) по
передаче данных между точками доступа к услугам (service access points - SAP).
При этом все детали реализации остаются скрытыми внутри черного ящика.

Внутренняя структура черного ящика

«чёрный ящик» сетевого сервиса реализуется совокупностью модулей сетевого
обеспечения узлов, взаимодействующих друг с другом через общие звенья
передачи данных
✔ как правило, программные модули Netware обслуживают Endpoint-

ы своего узла,
а также выполняют функции коммутации поступающих порций трафика, в том
числе и транзит между сетевыми интерфейсами
✔ низкоуровневые процессы приёма и передачи данных через звенья, реализуются
аппаратными адаптерами сетевых интерфейсов в асинхронном режиме

SAP

SAP это логический объект ОС, напоминающий файловый дескриптор. Процессы
ОС могут динамически создавать и закрывать SAP посредством сетевого API ОС
✔ SAP «олицетворяют» (represent) Endpoint-ы в сети при транспортировке трафика
между ними:
➢ порции трафика поступают в сеть через SAP процесса отправителя
➢ сеть доставляет порции трафика через SAP процесса получателя
✔ Для разных применений, ОС может реализовывать несколько методов
транспортировки трафика по сети через SAP соответствующего типа:
➢ дейтаграммный: независимая транспортировка отдельных (обособленных)
порций данных ограниченного размера
➢ потокоориентированный: транспортировка (потенциально бесконечного)
упорядоченного потока байтов и др.
✔ Коммутация трафика на транзитных узлах осуществляется без участия приклад-
ных процессов ОС и поэтому наличие SAP (на этих узлах для

транспортировки
транзитного трафика) не требуется

Адресация

✔ На уровне сети каждому действующему SAP присваивается уникальный адрес.
Обычно он формируется из трёх компонентов:

<Тип SAP> : <Сетевой адрес узла/интерфейса> : <Номер порта>

Жизненный цикл SAP

SAP динамически создаются и удаляются в процессе работы сетевого узла по запросам от прикладных процессов. Одновременно на узле могут существовать
тысячи SAP разных типов.

Логическая модель сетевого сервиса:

```
«Черный ящик»: Модель, в которой внутренняя реализация сервиса скрыта, и известны только входы и выходы.
SAP (Service Access Point): Точка доступа к сетевому сервису, через которую приложение взаимодействует с сетью.
Адресация: Процесс назначения уникальных адресов устройствам и узлам в сети для обеспечения правильной маршрутизации данных.
Жизненный цикл SAP: Включает создание, использование и уничтожение точки доступа к сервису.
```

Режимы:

```
Дейтаграммный режим:
    Данные передаются в виде отдельных пакетов (дейтаграмм),
```

```
которые могут следовать разными маршрутами.
    Пакеты могут приходить в разном порядке или теряться,
требует механизма для их правильной сборки.
    Примеры: UDP (User Datagram Protocol).
Потокоориентированный режим:
    Данные передаются как непрерывный поток байтов.
    Сессия устанавливается перед началом передачи данных,
гарантируется порядок доставки и целостность данных.
    Примеры: TCP (Transmission Control Protocol).
```

15. Процедура обмена данными через сеть в среде ОС: организация обмена данными, примитивы send/receive, индикации, роль буферов, логические соединения

Взаимодействующие приложения функционируют в виде процессов ОС на узлах
сети, где развёрнут сетевой сервис и имеются подключения к сетевой инфраструктуре, обладающей связностью
✔ В каждом приложении исполняется программный код, который манипулирует
данными в памяти своих процессов ОС и имеет возможность вызывать методы
API сетевого сервиса
Для обмена данными по сети каждое из приложений организует в своей памяти
буферы данных:
➔ буферы передачи S, куда код приложения должен скопировать данные, которые
сетевой сервис будет забирать для передачи по сети
➔ буферы приёма R, куда сетевой сервис будет копировать

принимаемые по сети
данные и откуда код приложения сможет их забрать для обработки
По сути, эти буферы и выступают в роли Endpoint-ов при доставке трафика по сети

Программный код приложений через сетевой API запрашивает создание SAP.
Сетевой сервис по этому запросу регистрирует логические Endpoint-ы в рамках
своего узла
✔ При создании SAP им присваивается адрес Endpoint в сети, включающий:
➢ уникальный (в пределах сети) адрес, присвоенный данному узлу/интерфейсу
➢ номер порта, который сейчас не назначен другому SAP внутри данного узла
Программный код одного из приложений (A) инициирует установку логического
соединения между SAP:
➔ Запрос Connect(addr) – установить логическое соединение с Endpoint, адрес
которого указан в параметре (addr)
Программный код второго приложения (B):
➔ получает индикацию NewConn — запрос установки нового соединения
➔ вызывает функцию Accept — принять новое логическое соединение
Программный код приложений через SAP вызывает функции API:
➔ Запрос Send – отправить данные из буфера S партнеру по соединению
➔ Запрос Recv (receive) – поместить в буфер R данные, принятые от партнера
Сетевой сервис узлов A и B, а также транзитных узлов по маршруту, выполняет в

асинхронном режиме транспортировку (копирование) данных по сети из буферов
передачи отправителя в буферы приёма получателя. Этот процесс происходит без
участия программного кода приложений
По завершению копирования данных сетевой сервис через механизмы ОС опове-
щает процессы приложений своего узла о завершении ранее запрошенных
действий:
➔ Индикация SCmp (send complete) – ранее отправленные данные переданы
➔ Индикация DAvl (data available) - в буфере R доступны новые принятые
данные

Организация обмена данными:

```
Обмен данными в операционной системе (ОС) происходит через
сетевые сокеты, которые предоставляют интерфейс для отправки и
получения данных.
ОС использует различные протоколы и примитивы для управления
передачей данных.
```

Примитивы send/receive:

```
send: Примитив для отправки данных через сокет. Вызывает
передачу данных на другой конец соединения.
receive: Примитив для приёма данных из сокета. Используется для
получения данных, отправленных другим узлом.
```

Индикации:

```
Сигналы или уведомления, используемые ОС для указания состояния передачи данных, например, готовность сокета для отправки или приёма данных.
Эти индикации могут быть использованы для асинхронного ввода-вывода (I/O) и управления потоком данных.
```

Роль буферов:

```
Буферы используются для временного хранения данных до их передачи или после приёма.
Они помогают управлять различиями в скорости передачи данных между отправителем и получателем, предотвращая потерю данных.
```

Логические соединения:

```
Представляют собой абстракцию, которая используется для установления, управления и завершения сеансов передачи данных.
Логические соединения обеспечивают надёжную передачу данных между узлами, гарантируя, что данные доставляются в правильном порядке и без потерь.
```

16. Реализация сетевого сервиса в среде ОС: Netware, его внешняя архитектура (интерфейсы, SAP), сетевой API. Принципы функционирования Netware в составе ОС, разновидности интерфейсов

Netware

Netware (модуль сетевого обеспечения) – часть «черного ящика», располагающаяся на одном узле

Роль netware – обеспечивать сервис обмена сообщениями между SAP (service access points), возможно, располагающихся на разных узлах

Сетевой интерфейс –точка подключения звена передачи данных к модулю netware конкретного узла. Имеет имя, которое известно только в рамках «своего» узла
Netware = Software + Hardware

в составе ядра ОС каждого узла сети функционирует модуль Netware (min один экземпляр...)
✔ в соответствии с топологией сети в модуле Netware конфигурируется необходимое количество сетевых интерфейсов, среди которых:
➔ аппаратные сетевые адаптеры с помощью которых узел может присоединяться к физическим звеньям передачи данных
➔ внутренние логические интерфейсы loopback
➔ разновидности виртуальных интерфейсов, позволяющих программно формировать и/или обрабатывать трафик
✔ состав и конфигурация интерфейсов, при необходимости, может динамически изменяться
✔ с помощью сетевого API процессы ОС могут динамически создавать и управлять SAP через которые netware реализует доставку сетевого трафика из/в Endpoint-ы процессов

Сетевой API

sock_hdl = socket (...) – создать SAP и розетку (socket) для доступа к ней
▲ sock_hdl – будет содержать socket handle («ручку от розетки»), которая
соответствует созданной в ОС точке доступа к сетевым услугам (SAP)
connect (sock_hdl, dst_addr ...) – соединить данную розетку с другой SAP
▲ dst_addr – адрес буфера в памяти приложения, где записаны координаты
вызываемой SAP (что-то вроде набираемого номера телефона)
send (sock_hdl, buf, len ...) – отправить порцию данных
▲ buf – адрес первого отправляемого байта в памяти приложения
▲ len – количество отправляемых байт
recv (sock_hdl, buf, len ...) – запросить получение порции принятых данных
▲ buf – адрес приемного буфера в памяти приложения куда должна будет
записана порция принятых данных
▲ len – размер приемного буфера в байтах (максимальный размер получаемой порции)

Внешняя архитектура:

```
Интерфейсы: Определяют взаимодействие между компонентами
системы и пользователями. Netware предоставляет множество
интерфейсов для различных уровней взаимодействия.
SAP (Service Access Point): Точки доступа к сетевым сервисам,
которые позволяют приложениям взаимодействовать с сетевыми
функциями.
```

Сетевой API:

```
Набор функций и вызовов, предоставляемых Netware для разработки
сетевых приложений.
API обеспечивает доступ к различным сервисам, таким как
файловый доступ, управление пользователями и сетевыми
ресурсами.
```

Принципы функционирования:

```
Netware работает как сетевой сервер, предоставляющий доступ к
общим ресурсам, таким как файлы и принтеры, для клиентов в
сети.
ОС использует клиент-серверную архитектуру, где сервер
управляет ресурсами и обеспечивает их доступность для клиентов.
```

Разновидности интерфейсов:

```
NCP (Netware Core Protocol): Основной протокол для
взаимодействия между клиентами и серверами Netware.
IPX/SPX (Internetwork Packet Exchange/Sequenced Packet
Exchange): Сетевые протоколы, используемые Netware для
маршрутизации и передачи данных.
```

17. Специфические проблемы при реализации netware и методы их решения: функциональная декомпозиция по слоям, гибкая модульная структура, стандартизация протоколов.

Netware – модуль сетевого обеспечения, предоставляющий сетевой сервис процессам через соответствующий

API.

Проблемы при реализации модулей netware

Модули netware – являются актерами в распределенных алгоритмах → являются протокольными модулями.

Необходимы специальные методы разработки.

Функциональная совместимость (interoperability) модулей netware друг с другом. Т.е. способность netware
разных компьютерных архитектур (Intel, ARM, MIPS и т.п.), разных операционных систем (Windows, Linux,
Android, iOS и т.п.), написанных разными программистами в разное время успешно функционировать вместе
в рамках общих распределенных алгоритмов.

Большое разнообразие возможных конфигураций сетевых узлов (разное число интерфейсов и типов
применяемых звеньев передачи данных) требует применения модульных подходов внутри netware с
обеспечением взаимозаменяемости разных модулей.

Необходимость реализации большого количества сложной функциональности (динамическое
мультиплексирование каналов, буферизация, контроль ошибок и восстановление данных, коммутация,
обеспечение транзита и т.д.) максимально прозрачно для прикладных процессов.

Все это требует применения особого подхода для разработки netware.

Модуль A и модуль B функционально совместимы (interoperable) в том случае, если они могут
взаимодействовать (выступать в роли актеров) в рамках некоторого протокола X (распределенного
алгоритма разновидности X).

Спецификация протокола – это формализованное,
однозначно понимаемое описание правил ведения
диалога определенного вида

Виды стандартов

• Стандарты де-факто (de-facto standards) Это некие технологические традиции, описываемыми не
формальными нормативными спецификациями, а широко известным know-how, позволяющими
добиваться совместимости независимо-выпускаемой продукции.

• Частные стандарты (proprietary standards) Не могут быть использованы без согласия правообладателя
(спецификации держатся в тайне и/или используется защита патентами). Обычно правообладатель
(владелец интеллектуальной собственности) либо единолично пользуется исключительным правом
выпускать продукцию, в соответствии со своим стандартом, либо собирает лицензионные отчисления
(royalties).

• Открытые стандарты (open standards) Имеют правовой статус, позволяющий использовать их любому
заинтересованному лицу. Обычно, в качестве автора и правообладателя открытых стандартов
выступает нейтральная некоммерческая организация – орган стандартизации (Standard Body),
которая способствует широкому применению выпускаемых ею стандартов. Текст открытых
стандартов публично доступен (бесплатно или за умеренную фиксированную плату).

Преимущества послойной архитектуры

• Реализация netware в виде модулей, которые взаимодействуют друг с другом по хорошо определенным интерфейсам → независимость модулей друг от друга

• Взаимозаменяемость модулей при условии
сохранения интерфейсов.

• Упрощение процесса разработки и тестирования

отдельного модуля.
• Возможность гибко строить netware конкретного узла из набора стандартных модулей (как конструктор Lego)
• Возможность гибкого расширения функциональности netware за счет добавления новых модулей.

Специфические проблемы:

```
Сложность управления: Управление большими и сложными сетевыми системами требует значительных ресурсов и точной координации.
Совместимость: Обеспечение совместимости между различными устройствами и протоколами может быть сложной задачей.
Надёжность и отказоустойчивость: Сети должны быть надёжными и устойчивыми к сбоям, чтобы обеспечивать непрерывный доступ к ресурсам.
```

Методы решения:

```
Функциональная декомпозиция по слоям: Разделение функций системы на отдельные слои позволяет упростить управление и улучшить модульность системы. Например, модель OSI (Open Systems Interconnection) делит сетевые функции на 7 уровней.
Гибкая модульная структура: Позволяет легко добавлять или заменять отдельные компоненты системы, улучшая адаптивность и расширяемость.
Стандартизация протоколов: Использование стандартизированных протоколов обеспечивает совместимость и интероперабельность между различными устройствами и системами. Например, использование IPX/SPX в Netware обеспечивает стандартизированный подход к маршрутизации и передаче данных.
```

18. Эталонные функциональные модели сетевой архитектуры: ЭМВОС, Internet Protocol Suite (TCP/IP). Перечисление и функциональное описание уровней этих моделей.

ЭМВОС - всеми нами знакомая сетевая модель OSI.

Сетевая модель OSI (базовая эталонная модель взаимодействия открытых систем, англ. Open Systems Interconnection Basic Reference Model) — абстрактная сетевая модель для коммуникаций и разработки сетевых протоколов.

- ЭМВОС — эталонная модель взаимосвязи открытых систем
- OSI - Open Systems Interconnection

Её уровни:

Физический уровень - (англ. physical layer)

Нижний уровень модели, который определяет метод передачи данных, представленных в двоичном виде, от одного устройства (компьютера) к другому. Функции физического уровня реализуются на всех устройствах, подключенных к сети. Со стороны компьютера функции физического уровня выполняются сетевым адаптером или последовательным портом. К физическому уровню относятся физические, электрические и механические интерфейсы между двумя системами. Физический уровень определяет такие виды сред передачи данных как оптоволокно, витая пара, коаксиальный кабель, спутниковый канал передач данных и т. п.

Канальный уровень (англ. data link layer)

Предназначен для обеспечения взаимодействия сетей на физическом уровне и контроля ошибок, которые могут возникнуть. Полученные с физического уровня данные, представленные в битах, он упаковывает в кадры, проверяет их на целостность и, если нужно, исправляет ошибки (либо формирует повторный запрос повреждённого кадра) и отправляет на сетевой уровень.
На этом уровне работают коммутаторы, мосты и другие устройства. Эти устройства используют адресацию второго уровня (по номеру уровня в модели OSI).
В программировании этот уровень представляет драйвер сетевой платы, в операционных системах имеется программный интерфейс взаимодействия канального и сетевого уровней между собой.

Сетевой уровень (англ. network layer)

Предназначен для определения пути передачи данных. Отвечает за трансляцию логических адресов и имён в физические, определение кратчайших маршрутов, коммутацию и маршрутизацию, отслеживание неполадок и «заторов» в сети.
Протоколы сетевого уровня маршрутизируют данные от источника к получателю. Работающие на этом уровне устройства (маршрутизаторы) условно называют устройствами третьего уровня (по номеру уровня в модели OSI).

Транспортный уровень (англ. transport layer)

предназначен для обеспечения надёжной передачи данных от отправителя к получателю. При этом уровень надёжности может варьироваться в широких пределах. Существует множество классов протоколов транспортного уровня, начиная от протоколов,

предоставляющих только основные транспортные функции (например, функции передачи данных без подтверждения приёма), и заканчивая протоколами, которые гарантируют доставку в пункт назначения нескольких пакетов данных в надлежащей последовательности, мультиплексируют несколько потоков данных, обеспечивают механизм управления потоками данных и гарантируют достоверность принятых данных. Например, UDP ограничивается контролем целостности данных в рамках одной датаграммы и не исключает возможности потери пакета целиком или дублирования пакетов, нарушение порядка получения пакетов данных; TCP обеспечивает надёжную непрерывную передачу данных, исключающую потерю данных или нарушение порядка их поступления или дублирования, может перераспределять данные, разбивая большие порции данных на фрагменты и наоборот, склеивая фрагменты в один пакет.

Сеансовый уровень (англ. session layer)

Обеспечивает поддержание сеанса связи, позволяя приложениям взаимодействовать между собой длительное время. Уровень управляет созданием/завершением сеанса, обменом информацией, синхронизацией задач, определением права на передачу данных и поддержанием сеанса в периоды неактивности приложений.

Уровень представления (англ. presentation layer)

Обеспечивает преобразование протоколов и кодирование/декодирование данных. Запросы приложений, полученные с прикладного уровня, на уровне представления преобразуются в формат для передачи по сети, а полученные из сети данные преобразуются в формат приложений. На этом уровне может осуществляться сжатие/распаковка или шифрование/дешифрование,

а также перенаправление запросов другому сетевому ресурсу, если они не могут быть обработаны локально.
Уровень представлений обычно представляет собой промежуточный протокол для преобразования информации из соседних уровней. Это позволяет осуществлять обмен между приложениями на разнородных компьютерных системах прозрачным для приложений образом.
Уровень представлений обеспечивает форматирование и преобразование кода. Форматирование кода используется для того, чтобы гарантировать приложению поступление информации для обработки, которая имела бы для него смысл. При необходимости этот уровень может выполнять перевод из одного формата данных в другой.
Чтобы понять, как это работает, представим, что имеются две системы. Одна использует для представления данных расширенный двоичный код обмена информацией EBCDIC, например, это может быть мейнфрейм компании IBM, а другая — американский стандартный код обмена информацией ASCII (его использует большинство других производителей компьютеров). Если этим двум системам необходимо обменяться информацией, то нужен уровень представлений, который выполнит преобразование и осуществит перевод между двумя различными форматами.
Другой функцией, выполняемой на уровне представлений, является шифрование данных, которое применяется в тех случаях, когда необходимо защитить передаваемую информацию от доступа несанкционированными получателями. Чтобы решить эту задачу, процессы и коды, находящиеся на уровне представлений, должны выполнить преобразование данных. На этом уровне существуют и другие подпрограммы, которые сжимают тексты и преобразовывают графические изображения в битовые потоки, так, что они могут передаваться по сети.

Прикладной уровень (уровень приложений; англ. application layer)

Верхний уровень модели, обеспечивающий взаимодействие пользовательских приложений с сетью:

```
позволяет приложениям использовать сетевые службы:
    удалённый доступ к файлам и базам данных,
    пересылка электронной почты;
отвечает за передачу служебной информации;
предоставляет приложениям информацию об ошибках;
формирует запросы к уровню представления.
```

TCP/IP

Семейство TCP/IP имеет три транспортных протокола: TCP, полностью соответствующий OSI, обеспечивающий проверку получения данных; UDP, отвечающий транспортному уровню только наличием порта, обеспечивающий обмен датаграммами между приложениями, не гарантирующий получения данных; и SCTP, разработанный для устранения некоторых недостатков TCP, в который добавлены некоторые новшества. В семействе TCP/IP есть ещё около двухсот протоколов, самым известным из которых является служебный протокол ICMP, используемый для внутренних нужд обеспечения работы; остальные также не являются транспортными протоколами.

Стек протоколов TCP/IP включает в себя четыре уровня:

```
Прикладной уровень (Application Layer),
Транспортный уровень (Transport Layer),
Межсетевой уровень (Сетевой уровень) (Internet Layer),
Канальный уровень (Network Access Layer).
```

Протоколы этих уровней полностью реализуют функциональные возможности модели OSI. На стеке протоколов TCP/IP построено всё взаимодействие пользователей в IP-сетях. Стек является независимым от физической среды передачи данных, благодаря чему, в частности, обеспечивается полностью прозрачное взаимодействие между проводными и беспроводными сетями.

19. Организация функционирования множества протокольных модулей в составе Netware: понятие протокольного стека, структура протокольного стека узла. Взаимодействие модулей внутри стека и с паритетными модулями других узлов.

Стек протоколов — это иерархически организованный набор сетевых протоколов, достаточный для организации взаимодействия узлов в сети. Протоколы работают в сети одновременно, значит работа протоколов должна быть организована так, чтобы не возникало конфликтов или незавершённых операций. Поэтому стек протоколов разбивается на иерархически построенные уровни, каждый из которых выполняет конкретную задачу — подготовку, приём, передачу данных и последующие действия с ними.

Количество уровней в стеке меняется в соответствии с конкретным стеком протоколов. Протоколы нижних уровней часто реализуются комбинацией программных и аппаратных средств, а протоколы верхних уровней, как правило, программными средствами.

Важно различать модель OSI и стек протоколов OSI. В то время как модель OSI является концептуальной схемой взаимодействия открытых систем, стек OSI представляет собой набор спецификаций конкретных протоколов.

В отличие от других стеков протоколов, стек OSI полностью соответствует модели OSI, включая спецификации протоколов для всех семи уровней взаимодействия, определённых в этой модели:

Структура протокольного стека

(На примере TCP/IP)
В стеке TCP/IP определены четыре уровня (см табл.1). Каждый из них несет на себе некоторую долю нагрузки по решению основной задачи - организации надежной и производительной работы составной сети, части которой построены на основе разных сетевых технологий.

Табл.1. Уровни стека TCP/IP
Уровень I Прикладной уровень
Уровень II Основной (транспортный) уровень
Уровень III Уровень межсетевого взаимодействия
Уровень IV Уровень сетевых интерфейсов

Описание каждого из уровней:

Уровень межсетевого взаимодействия

Стержнем всей архитектуры является уровень межсетевого взаимодействия, или сетевой уровень, который реализует концепцию передачи пакетов в режиме без установления соединений, то есть дейтаграммным способом. Именно этот уровень обеспечивает возможность перемещения пакетов по сети, используя тот маршрут, который в данный момент является наиболее рациональным. Этот уровень также называют уровнем internet, указывая, тем самым, на основную его функцию - передачу данных через составную сеть. Основным протоколом уровня (в терминологии модели OSI) в стеке TCP/IP является протокол IP. Этот протокол изначально проектировался как протокол передачи пакетов в составных сетях,

состоящих из большого количества локальных сетей, объединенных как локальными, так глобальными связями. Поэтому протокол IP хорошо работает в сетях со множеством топологий, рационально используя наличие в них подсистем и экономно расходуя пропускную способность низкоскоростных линий связи. Так как протокол IP является дейтаграммным протоколом, он не гарантирует доставку пакетов до узла назначения, но старается это сделать.
К уровню межсетевого взаимодействия относятся все протоколы, связанные с состоянием и модификацией таблиц маршрутизации, такие как протоколы сбора маршрутной информации RIP и OSPF, а также протокол межсетевых управляющих сообщений ICMP. Последний протокол предназначен для обмена информацией об ошибках между маршрутизаторами сети и удаленным источником пакета. С помощью специальных пакетов ICMP сообщает о невозможности доставки пакета, о превышении времени жизни или продолжительности сборки пакета из фрагментов, об аномальных величинах параметров, об изменении маршрута пересылки и типа обслуживания, о состоянии системы и т. п.

Основной уровень

Поскольку на сетевом уровне не устанавливается соединение, то нет никаких гарантий того, что все пакеты будут доставлены в место назначения целыми и невредимыми или придут в том же порядке, в котором они были отправлены. Эту задачу - обеспечение надежности информационной связи между двумя конечными узлами - решает основной уровень стека TCP/IP, называемый также транспортным.
На этом уровне функционируют протокол управления передачей TCP и протокол дейтаграмм пользователя UDP. Протокол TCP обеспечивает надежную передачу сообщений между удаленными прикладными процессами за счет образования логических соединений. Этот протокол позволяет равноранговым объектам на компьютере-

отправителе и на компьютере-получателе поддерживать обмен данными в дуплексном режиме. TCP позволяет без ошибок доставлять сформированный на одном из компьютеров поток байт в любой другой компьютер, входящий в составную сеть. TCP делит поток байт на части - сегменты и передает их нижележащему уровню межсетевого взаимодействия. После того, как эти сегменты будут доставлены в пункт назначения, протокол TCP снова соберет их в непрерывный поток байт.
Протокол UDP обеспечивает передачу прикладных пакетов дейтаграммным способом, как и главный протокол уровня межсетевого взаимодействия IP, и выполняет только функции связующего звена (мультиплексора) между сетевым протоколом и многочисленными системами прикладного уровня, или пользовательскими процессами.

Прикладной уровень

Прикладной уровень объединяет все службы, представляемые системой пользовательским приложениям. За долгие годы использования в сетях различных стран и организаций стек TCP/IP накопил большое число протоколов и служб прикладного уровня. Прикладной уровень реализуется программными системами, построенными в архитектуре клиент-сервер, базирующейся на протоколах нижних уровней.
В отличие от протоколов остальных трех уровней, протоколы прикладного уровня занимаются деталями конкретного приложения и "не интересуются" способами передачи данных по сети. Этот уровень постоянно расширяется за счет присоединения к старым, прошедшим многолетнюю эксплуатацию сетевым службам типа Telnet, FTP, TFTP, DNS, SNMP, сравнительно новых служб, таких, например, как протокол передачи гипертекстовой информации HTTP.

Уровень сетевых интерфейсов

Идеологическим отличием архитектуры стека TCP/IP от многоуровневой организации других стеков является интерпретация функций самого нижнего уровня - уровня сетевых интерфейсов. Протоколы этого уровня должны обеспечивать интеграцию в составную сеть других сетей, причем задача ставится так: сеть TCP/IP должна иметь средства включения в себя любой другой сети, какую бы внутреннюю технологию передачи данных эта сеть не использовала. Отсюда следует, что этот уровень нельзя определить раз и навсегда. Для каждой технологии, включаемой в составную сеть подсети, должны быть разработаны собственные интерфейсные средства. К таким интерфейсным средствам относится протокол инкапсуляции IP-пакетов межсетевого взаимодействия в кадры локальных технологий.

Уровень сетевых интерфейсов в протоколах TCP/IP не регламентируется, но он поддерживает все популярные стандарты физического и канального уровней: для локальных сетей - это Ethernet, Token Ring, FDDI, Fast Ethernet, Gigabit Ethernet, 100VG-AnyLAN, для глобальных сетей - протоколы соединений "точка-точка" SLIP и PPP, протоколы территориальных сетей с коммутацией пакетов X.25, frame relay. Разработана также специальная спецификация, определяющая использование технологии ATM в качестве транспорта канального уровня.

Структура связей протокольных модулей (На примере TCP/IP)

Рассмотрим потоки данных, проходящие через стек. В случае использования протокола TCP, данные передаются между прикладным процессом и модулем TCP. Типичным прикладным процессом, использующим протокол TCP, является модуль FTP. Стек протоколов в этом случае будет FTP/TCP/IP/Ethernet. При использовании протокола

UDP, данные передаются между прикладным процессом и модулем UDP. Например, SNMP пользуется транспортными услугами UDP. Его стек протоколов выглядит так: SNMP/UDP/IP/Ethernet.

Модули TCP, UDP и драйвер Ethernet являются мультиплексорами n x 1. Действуя как мультиплексоры, они переключают несколько входов на один выход. Они также являются демультиплексорами 1 x n. Как демультиплексоры, они переключают один вход на один из многих выходов в соответствии с полем типа в заголовке протокольного блока данных. Другими словами, происходит следующее:

```
Когда Ethernet-кадр попадает в драйвер сетевого интерфейса
Ethernet, он может быть направлен либо в модуль ARP, либо в
модуль IP. На то, куда должен быть направлен Ethernet-кадр,
указывает значение поля типа в заголовке кадра.
Если IP-пакет попадает в модуль IP, то содержащиеся в нем
данные могут быть переданы либо модулю TCP, либо UDP, что
определяется полем "протокол" в заголовке IP-пакета.
Если UDP-датаграмма попадает в модуль UDP, то на основании
значения поля "порт" в заголовке датаграммы определяется
прикладная программа, которой должно быть передано прикладное
сообщение.
Если TCP-сообщение попадает в модуль TCP, то выбор прикладной
программы, которой должно быть передано сообщение,
осуществляется на основе значения поля "порт" в заголовке TCP-
сообщения.
```

Мультиплексирование данных в обратную сторону осуществляется довольно просто, так как из каждого модуля существует только один путь вниз. Каждый протокольный модуль добавляет к пакету свой заголовок, на основании которого машина, принявшая пакет, выполняет демультиплексирование.

Пример передачи данных на другой узел (На примере TCP/IP)

Для иллюстрации рассмотрим передачу данных от одного узла к другому с использованием стека TCP/IP:

```
Приложение на узле A (например, веб-браузер) отправляет HTTP-запрос на узел B (веб-сервер).
HTTP-запрос передается на транспортный уровень, где протокол TCP сегментирует данные и добавляет заголовки с портами источника и назначения.
Сегменты TCP передаются на сетевой уровень, где протокол IP добавляет заголовки с IP-адресами источника и назначения.
IP-пакеты передаются на канальный уровень, где протокол Ethernet добавляет заголовки с MAC-адресами источника и назначения.
Ethernet-кадры передаются по физической сети к узлу B.
На узле B кадры Ethernet принимаются и передаются вверх по стеку протоколов: сначала на сетевой уровень (IP), затем на транспортный уровень (TCP), и, наконец, на уровень приложений (HTTP).
Веб-сервер на узле B обрабатывает HTTP-запрос и отправляет ответ обратно на узел A по аналогичной цепочке.
```

20. Описание процесса передачи данных через протокольный стек. Понятие PDU, его структура. Преобразования данных: сегментирование, сборка, инкапсуляция, декапсуляция. Мультиплексирование протоколов и экземпляров протоколов.

Процесс передачи данных через протокольный стек включает несколько уровней, каждый из которых добавляет свою специфическую информацию для обеспечения корректной передачи данных от отправителя к получателю.

Процесс передачи данных через стек протоколов начинается с отправки данных на прикладном уровне. Данные проходят через все уровни стека протоколов, где осуществляются необходимые преобразования и обработка данных на каждом уровне. Например, на транспортном уровне данные могут быть разделены на пакеты или объединены из пакетов, а на сетевом уровне могут быть добавлены заголовки с адресами маршрутизации. Затем данные отправляются на физический уровень, где они конвертируются в биты и передаются по физическому каналу связи.

PDU

PDU - порция данных в форме массива байтов некоторой длины
PDU, как правило, формируется модулем-отправителем путём записи определённых значений в байты буфера передачи
На каждом уровне OSI PDU имеет своё представление в виде данных - например, на канальной уровне - это кадр (frame), а вот на сетевом - это пакет (packet).

Структурно, PDU может включать до трех логических частей:
заголовка (header) - содержимое полностью определяется рассматриваемым протоколом
полезной нагрузки (payload) - содержащей инкапсулированные данные, транспортируемые по сети (для протокола это просто набор байтов)

концевика (trailer, footer) - обычно содержит служебные данные, необходимые для контроля правильности передачи

Преобразования данных, выполняемые на каждом уровне стека протоколов:

Сегментирование - разделение данных на более мелкие блоки для передачи на транспортном уровне.
Сборка – объединение фрагментов данных на транспортном уровне.
Инкапсуляция - добавление заголовков и другой информации к данным на каждом уровне стека протоколов для передачи.
Декапсуляция - удаление заголовков и другой информации с данных на каждом уровне стека протоколов после их получения.
Мультиплексирование протоколов и экземпляров протоколов позволяет передавать несколько потоков данных одновременно через один физический канал связи. Это достигается за счет того, что каждый протокол имеет свой уникальный номер порта или идентификатор, который используется для маршрутизации данных на приемной стороне. Кроме того, в протокольном стеке можно использовать несколько экземпляров протоколов, где каждый экземпляр обрабатывает данные отдельно друг от друга, что позволяет изолировать различные потоки данных и предотвращать их взаимное влияние

21. Понятие протокола и его составные части: актеры, диалог, сообщения, локальные действия. Протокольные термины: спецификация, реализация, экземпляр, таймаут.

Протокол - это набор правил и процедур, которые определяют способ обмена информацией и управляют взаимодействием между различными устройствами или приложениями в сети. Протоколы используются для того, чтобы гарантировать корректность передачи данных и обеспечить совместимость между различными системами.

Составные части протокола:
Актеры (участники) - это сущности, которые участвуют в обмене сообщениями по протоколу. Актеры могут быть как программными, так и аппаратными компонентами.
Диалог - это последовательность сообщений, которые передаются между актерами в рамках протокола.
Сообщения - это данные, которые передаются между актерами в рамках диалога. Сообщения содержат информацию о команде, запросе или ответе.
Локальные действия - это действия, которые выполняются каждым актером в ответ на получение или отправку сообщений. Например, актер может сохранять полученные данные в свою локальную базу данных или вычислять результат на основе полученных запросов.

Протокольные термины:
Спецификация - это набор требований к протоколу, который определяет его поведение и основные параметры. Спецификация описывает формат сообщений, правила обмена, требования по надежности и т.д.
Реализация - это реализация протокола в виде программного кода или аппаратного устройства. Реализация должна соответствовать спецификации протокола, чтобы обеспечить совместимость между различными системами.
Экземпляр - это конкретный экземпляр протокола, который запущен на определенных устройствах или приложениях. Каждый экземпляр протокола может иметь свои уникальные параметры и настройки.
Экземпляр протокола - это конкретный случай использования

протокола в определенной ситуации, когда два или более устройства обмениваются данными в соответствии с правилами, заданными протоколом. Это может быть одноразовое соединение между двумя устройствами или длительная сессия обмена данными.
Примером экземпляра протокола может служить передача электронной почты через протокол SMTP (Simple Mail Transfer Protocol). В таком случае, для отправки сообщения используется экземпляр протокола SMTP, который включает информацию о сервере и его параметры, а также данные о самом сообщении (кому адресовано, тема, содержание и т.д.).
При передаче электронной почты через протокол SMTP, каждый компонент сообщения разбивается на пакеты данных, которые затем передаются по сети от отправителя к получателю. SMTP гарантирует доставку сообщения до получателя, а также обеспечивает контроль ошибок и возможность повторной отправки сообщения в случае неудачной попытки отправки.
Таким образом, экземпляр протокола SMTP позволяет пользователям отправлять электронную почту, используя стандартные правила и процедуры, заданные протоколом SMTP.
Таймаут - это время ожидания ответа на запрос или сообщение перед тем, как актер считается недоступным. Если ответ не получен в течение заданного времени, то протокол может перейти в другое состояние или инициировать другую последовательность действий.

22. Способ изображения протокола. Роли актеров в рамках протокола. Модель «клиент-сервер».

Способ изображения протокола - это диаграмма, которая показывает последовательность сообщений, передаваемых между актерами в рамках протокола. Существует несколько способов изображения протокола, но наиболее распространенными являются диаграммы

последовательностей и диаграммы состояний.
Диаграмма последовательности показывает последовательность сообщений, передаваемых между актерами в виде вертикальных линий (актеры) и горизонтальных стрелок (сообщения). Диаграмма состояний показывает состояния, через которые проходит каждый актер в рамках протокола и переходы между этими состояниями.
Роли актеров в рамках протокола могут быть различными в зависимости от конкретного протокола. Например, в протоколе HTTP (HyperText Transfer Protocol), актеры могут быть клиентом (браузером) и сервером (веб-сервером), а в протоколе SMTP (Simple Mail Transfer Protocol) - отправителем и получателем электронной почты.

Модель "клиент-сервер" - это модель взаимодействия между компьютерными системами, которая базируется на принципе распределенной обработки. В этой модели одна компьютерная система (сервер) предоставляет определенный сервис или ресурс для других систем (клиентов), которые запрашивают этот сервис или ресурс.
В рамках модели "клиент-сервер" клиент и сервер взаимодействуют по определенному протоколу, который определяет формат запросов и ответов, правила аутентификации и шифрования, а также другие параметры. Клиент и сервер могут быть расположены на разных компьютерах или даже в разных сетях, что позволяет эффективно использовать доступные ресурсы и увеличивать масштабируемость системы.
Модель "клиент-сервер" имеет несколько преимуществ, таких как:
Централизованное управление: сервер является центральным элементом системы, который управляет доступом к ресурсам и контролирует работу клиентов.
Удобство использования: клиенты могут получать доступ к ресурсам с помощью стандартизированных протоколов, что делает использование системы более удобным и простым.
Масштабируемость: система может быть масштабирована путем

добавления новых серверов или распределения нагрузки между существующими серверами, что повышает производительность и увеличивает ее масштабируемость.

23. Сетевой протокол IPv4 и его базовые понятия: сеть (network), интер-сеть (internet), узел (host), интерфейс, multihomed host, шлюз. Виды сетей: P2P и multipoint. Режимы передачи: unicast, broadcast и multicast. Принцип передачи пакетов по интер-сети.

Протокол IPv4 является одним из наиболее распространенных сетевых протоколов, используемых в Интернете и других компьютерных сетях. Он предназначен для управления адресами и маршрутизации данных между устройствами внутри сети и между различными сетями (интер-сеть).
Сеть (network) в терминах IPv4 обычно описывается как группа узлов (host), связанных между собой через общие каналы передачи данных. Каждый узел имеет свой уникальный IP-адрес, который позволяет ему быть идентифицированным и общаться с другими узлами в этой сети.
Интер-сеть (internet) - это объединение нескольких локальных сетей в одну большую сеть. Каждая сеть в интер-сети может иметь свои собственные IP-адреса и свою собственную схему адресации, но все они должны соответствовать общим принципам IPv4, чтобы обеспечить корректную маршрутизацию данных между сетями.
Узел (host) - это устройство в сети, которое может отправлять и принимать данные. Это может быть компьютер, сервер или любое другое сетевое устройство.
Интерфейс - это средство подключения узла к сети, которое обычно

представлено в виде сетевой карты или другого аналогичного устройства. Каждый интерфейс имеет свой собственный уникальный IP-адрес, который используется для идентификации узла в сети. Multihomed host - это узел, который имеет более одного интерфейса и, следовательно, более чем один IP-адрес. Это может быть полезно для обеспечения надежности и повышения производительности связи между узлами.

Когда устройство имеет только один сетевой интерфейс и подключено к одной сети, оно зависит от этой сети для связи с другими устройствами и ресурсами. Если эта сеть выходит из строя, то устройство теряет связь с другими устройствами и перестает функционировать.
В случае с multihomed host, устройство имеет несколько сетевых интерфейсов, каждый из которых может быть подключен к разным сетям. Это позволяет использовать несколько путей для связи с другими устройствами и ресурсами, что повышает надежность и доступность соединений.
Например, если одна из сетей выходит из строя, устройство может продолжать функционировать, используя другую сеть. Также multihomed host может использоваться для повышения скорости передачи данных, например, при использовании двух сетевых интерфейсов параллельно.
Multihomed host также может быть использован для разделения сетевых трафиков, например, для разделения сети на группы по сетевой безопасности или для разделения рабочих и личных сетей.
В целом, multihomed host предоставляет более высокий уровень доступности и надежности соединений, а также большую гибкость в управлении сетью.
Шлюз (gateway) - это устройство, которое используется для соединения двух или более сетей разных типов. Шлюз позволяет узлам в одной сети общаться с узлами в другой сети, пересылая данные между этими сетями.

Когда узел в одной сети хочет отправить данные на узел в другой сети, он посылает эти данные своему шлюзу. Шлюз принимает данные, проверяет их адрес назначения и определяет, в какую сеть нужно отправить данные. Затем шлюз пересылает данные в сеть назначения, где они доставляются до узла-получателя.
Процесс пересылки данных между сетями возможен благодаря тому, что шлюз имеет минимум два сетевых интерфейса, каждый из которых подключен к разным сетям. Шлюз может использовать информацию, содержащуюся в заголовках пакетов данных, чтобы определить, какие данные должны быть пересланы в какую сеть.
Для обеспечения безопасности и контроля над передаваемыми данными, шлюз также может выполнять функции маршрутизации, фильтрации пакетов по адресам и протоколам, аутентификации и шифрования данных. Это позволяет создавать более надежные и безопасные сетевые соединения между различными узлами и сетями.
В целом, использование шлюза позволяет узлам в разных сетях общаться друг с другом и обеспечивает эффективную передачу данных между различными сетями.
Существует два типа сетей в терминах IPv4 - это P2P (point-to-point) сети и multipoint (многоточечные) сети.
P2P (point-to-point) сеть является прямой связью между двумя узлами. Это обычно соединение двух компьютеров через кабель, как в случае с использованием Ethernet-кабеля.
Multipoint (многоточечные) сети - это сеть, которая имеет более одного узла и используется для обмена данными между ними. Такие сети часто используются в офисах, где несколько компьютеров подключены через один роутер или коммутатор.
Существует три режима передачи данных в терминах IPv4 - unicast, broadcast и multicast.
Unicast - это режим передачи данных, при котором данные отправляются только на один конкретный узел в сети.
Broadcast - это режим передачи данных, при котором данные

отправляются на все узлы в данной сети. Этот режим используется для широковещательных сообщений, таких как запросы ARP (Address Resolution Protocol) или DHCP (Dynamic Host Configuration Protocol).
Multicast - это режим передачи данных, при котором данные отправляются на группу узлов в сети, которые были заранее выбраны для этого. Каждая группа имеет свой уникальный IP-адрес, который используется для идентификации группы узлов. Этот режим используется для потокового мультимедийного контента, онлайн-игр и других приложений, когда несколько узлов должны получать одинаковый поток данных.
Принцип передачи пакетов по интер-сети в IPv4 основан на использовании маршрутизаторов. Каждый маршрутизатор имеет таблицу маршрутизации, которая содержит информацию о том, какие сети доступны через данный маршрутизатор и какие маршруты лучше использовать для доставки пакетов. Когда пакет отправляется из одной сети в другую, он проходит через различные маршрутизаторы на пути к конечному узлу.
Каждый пакет содержит IP-адрес источника и назначения, а также другую информацию о пакете. Когда маршрутизатор получает пакет, он сравнивает IP-адрес назначения с таблицей маршрутизации и отправляет пакет по соответствующему маршруту. Если нужный маршрут не найден, маршрутизатор может либо отбросить пакет, либо отправить его на шлюз по умолчанию для дальнейшей обработки.

24. Адресация в сетях IPv4: назначение, адресуемый элемент, структура адресов IPv4, способ записи. Диапазоны и блоки IP адресов. Свойства адресов, образующих блок, префиксы и суффиксы. Способы записи блоков адресов: netmask и CIDR.

Адресация в сетях IPv4 используется для идентификации устройств, подключенных к компьютерной сети, а также для маршрутизации пакетов данных между этими устройствами. В IPv4 каждое устройство в сети имеет уникальный 32-битный адрес (четыре октета), который записывается в десятичной системе счисления.
IPv4 адрес состоит из двух частей: сетевой и хостовой. Сетевая часть адреса определяет сеть, к которой принадлежит устройство, а хостовая - устройство внутри этой сети. Для разделения сетевой и хостовой частей используется специальная маска подсети, которая указывается после IPv4-адреса и содержит набор единиц и нулей.
Адрес IPv4 записывается в виде четырех чисел, разделенных точками. Каждое число представляет собой значение байта (8 бит), и может быть от 0 до 255. Например, адрес 192.168.1.1 является допустимым IPv4-адресом
Структура адресов IPv4 выглядит следующим образом:

XXX.XXX.XXX.XXX

Где каждый XXX представляет один октет адреса, который может принимать значения от 0 до 255. Например, адрес 192.168.1.1 является допустимым IPv4-адресом.

IPv4-адреса можно разделить на группы, или диапазоны, в зависимости от их назначения. Некоторые наиболее распространенные диапазоны IPv4-адресов включают:
Существует несколько наиболее распространенных диапазонов IP-адресов, которые используются для различных целей. Рассмотрим каждый из них более подробно:

Приватный диапазон адресов 10.0.0.0 - 10.255.255.255
Этот диапазон адресов используется для приватных сетей и не маршрутизируется в Интернете. Это означает, что устройства, подключенные к сети с такими адресами, не могут быть напрямую

доступны из глобальной Интернет- сети, а только из локальных сетей, использующих тот же диапазон.

Приватный диапазон адресов 172.16.0.0 - 172.31.255.255
Этот диапазон адресов также используется для приватных сетей и не маршрутизируется в Интернете. Он предназначен для использования в больших организациях, которые нуждаются в большем количестве адресов, чем доступно
в диапазоне 10.0.0.0/8.

Приватный диапазон адресов 192.168.0.0 - 192.168.255.255
Этот диапазон адресов также используется для приватных сетей и не маршрутизируется в Интернете. Он широко используется в домашних и небольших офисных сетях, поскольку предоставляет достаточное количество адресов для
таких сетей.

Диапазон адресов 169.254.0.0 - 169.254.255.255 (APIPA - Automatic Private IP Addressing)
Этот диапазон адресов используется, когда устройство не может получить IP-адрес от DHCP-сервера. В этом случае устройство автоматически назначает себе адрес из этого диапазона, чтобы поддерживать связь в локальной сети.
APIPA - это сокращение от Automatic Private IP Addressing. Это технология автоматического назначения адреса в локальной компьютерной сети, которая используется, когда устройство не может получить IP-адрес от DHCP-сервера.
Если компьютер не может связаться с DHCP-сервером, то он автоматически присваивает себе IP-адрес из диапазона 169.254.0.0/16 (от 169.254.0.0 до 169.254.255.255), который является зарезервированным для APIPA. При использовании APIPA устройства могут продолжать работать в локальной сети, не зависимо от наличия DHCP-сервера, что обеспечивает повышенную отказоустойчивость и надежность сети. Однако, так как адреса APIPA не уникальны в

глобальной Интернет-сети, они не могут быть использованы для доступа к ресурсам в Интернете или на других сетях.

Отказоустойчивость - это способность системы продолжать работу в случае ее частичного или полного отказа. Она означает, что система может сохранять свою функциональность и доступность для пользователей даже при возникновении непредвиденных сбоев, ошибок или атак.
DHCP-сервер (Dynamic Host Configuration Protocol) - это сервер, который автоматически назначает IP-адреса и другие настройки сетевых параметров устройствам в компьютерной сети. DHCP-сервер позволяет управлять динамическим распределением IP-адресов, что упрощает конфигурацию сетевых устройств и уменьшает вероятность ошибок при настройке сети вручную.
DHCP-серверы используются практически во всех типах сетей, от локальных домашних сетей до крупных корпоративных сетей. Когда устройство подключается к сети, оно отправляет запрос на получение IP-адреса на DHCP-сервер, который затем назначает ему свободный адрес из пула доступных адресов. Кроме того, DHCP-сервер может также предоставлять информацию о шлюзе по умолчанию, DNS-серверах и других параметрах сети.
Каждый из этих диапазонов имеет свои особенности и предназначен для определенных целей. Приватные диапазоны адресов обеспечивают безопасность и конфиденциальность данных в локальных сетях, а диапазон APIPA помогает устройствам продолжать работу в сети даже в случае отсутствия доступного DHCP-сервера.
Диапазоны IP-адресов, которые используются для выхода в Интернет, называются публичными адресами. Эти адреса представляют собой уникальные идентификаторы, которые могут быть использованы для доступа к ресурсам в Интернете.
Публичные IP-адреса, как правило, выдаются провайдерами интернет-услуг, которые резервируют определенный диапазон адресов для своих клиентов. Некоторые из наиболее распространенных

диапазонов публичных IP-адресов, которые используются для выхода в Интернет, включают:
0.0.0.0 - 126.255.255.255
127.0.0.0 - 191.255.255.255
192.0.0.0 - 223.255.255.255
Эти диапазоны IP-адресов были выделены Интернет-ассигнационным блоком (IANA) для использования в публичных сетях и доступа к Интернет.
Но следует отметить, что эти диапазоны IP-адресов не являются единственными публичными адресами, используемыми для выхода в Интернет. Провайдеры интернет-услуг могут использовать и другие диапазоны IP-адресов в зависимости от региона, страны или других факторов.
IPv4-адреса также могут быть объединены в блоки, которые представляют собой набор адресов, которые могут быть использованы в одной или нескольких сетях. Блок адресов может быть записан в виде начального и конечного адресов или в виде адреса и маски подсети.

Адреса, образующие блок, имеют ряд свойств:

Все адреса в блоке имеют одинаковый префикс (начальную часть), который определяет сеть.
Размер блока задается количеством бит, которые занимает префикс. Количество адресов в блоке можно вычислить как 2 в степени числа бит в хостовой части.

Адреса, образующие блок, используются для распределения IP-адресов на подсети.

Блоки могут быть разделены на более мелкие блоки при необходимости увеличения количества доступных адресов или

уменьшения размера подсетей.
Блоки IPv4 адресов могут быть классифицированы в соответствии с их размером и назначением (например, A, B, C и т.д.).
Блоки могут быть использованы для организации VLAN (Virtual Local Area Network) в сетях Ethernet.
VLAN (Virtual Local Area Network) - это технология, которая позволяет разделить одну физическую сеть на несколько логических, независимых друг от друга виртуальных сетей.
В каждой VLAN может быть своя собственная настройка безопасности и управления доступом. VLAN позволяют группировать устройства в сети по функциональной принадлежности, а не только по физическому расположению в сети.
Маска подсети - это набор битов, которые определяют, какие биты IPv4-адреса являются сетевыми, а какие - хостовыми. Маска подсети записывается так же, как и IPv4-адрес, в виде 32-битного числа, которое состоит из четырех октетов (например, 255.255.255.0). Эта маска означает, что первые 24 бита IPv4-адреса являются сетевыми, а последние 8 бит - хостовыми.
Префикс и суффикс - это термины, которые используются для обозначения частей блока адресов. Префикс - это часть блока адресов, которая определяет сеть, к которой принадлежат устройства. Суффикс - это часть блока адресов, которая определяет количество доступных хостов в этой сети.
Пример записи блока адресов: 192.168.1.0/24.
Здесь 192.168.1.0 является адресом сети, а /24 - маской подсети, указывающей, что первые 24 бита IPv4-адреса являются сетевыми, а оставшиеся 8 бит - хостовыми. Это означает, что в данном блоке адресов может быть до 256 доступных хостов (2^8), так как последний октет адреса используется для идентификации конкретного устройства.
IPv4-адресация является одним из ключевых компонентов компьютерных сетей. Она позволяет устройствам обмениваться

данными между собой и использовать Интернет-ресурсы. Для правильной настройки сети необходимо понимание структуры IPv4-адресов, префиксов и масок подсетей, а также способов их использования при настройке маршрутизации данных в сети. Существуют два основных способа записи блоков адресов в IP-сетях - это запись netmask и запись CIDR.
Запись netmask (маска сети)

Netmask

Netmask представляет собой битовую маску, которая определяет диапазон IP-адресов, принадлежащих заданной сети. Она записывается в виде четырёх чисел от 0 до 255, разделённых точками, например: 255.255.255.0. Каждый из этих чисел представляет собой количество битов, которые используются для определения адреса сети. Таким образом, маска сети 255.255.255.0 означает, что первые 24 бита адреса используются для определения адреса сети, а оставшиеся 8 битов - для определения адресов устройств внутри этой сети.
В записи Netmask, количество первых бит, отведенных под сетевую часть IP-адреса, определяется значением маски, где единицы обозначают сетевую часть, а нули - хостовую.
Например, в маске "255.255.255.0" (в двоичной системе - "11111111.11111111.11111111.00000000") первые 24 бита (три октета) отведены под сетевую часть, а последние 8 бит (один октет) отведены под хостовую часть. Таким образом, в данном случае количество первых бит, отведенных под сетевую часть, равно 24.

Запись CIDR (Classless Inter-Domain Routing)

CIDR - это метод записи диапазона IP-адресов в более компактной форме. Он состоит из IP-адреса и числа, которое обозначает количество битов, используемых для определения адреса сети.

Например, 192.168.1.0/24 означает, что
первые 24 бита адреса используются для определения адреса сети. В этой записи число после слеша (/) обозначает количество битов в маске сети.
CIDR-запись может быть использована для записи любого диапазона IP-адресов, включая как одиночные адреса, так и блоки адресов различных размеров. Например, 192.168.1.0/24 представляет собой блок из 256 адресов (от
192.168.1.0 до 192.168.1.255), а 10.10.0.0/16 - блок из 65 536 адресов (от 10.10.0.0 до 10.10.255.255).
В целом, CIDR-запись более удобна и компактна, чем запись netmask, поэтому она широко используется в современных сетях. Однако, для понимания работы сетей, важно иметь понимание и описательную способность обеих
систем записи блоков адресов.
CIDR - это аббревиатура от английского выражения "Classless Inter-Domain Routing", что можно перевести как "Безклассовая междоменная маршрутизация". Это метод разбиения IP-адресов на более мелкие подсети без использования классов адресации, который был применяем в ранних версиях протокола IP. Использование CIDR позволяет более гибко и эффективно использовать доступные адреса IP, улучшает масштабируемость сетей и повышает эффективность маршрутизации.
Класс адресации - это метод, используемый для разделения IP-адресов на группы, которые затем могут использоваться для определения подсетей и маршрутизации трафика в сети. Ранее существовали три класса адресации: класс A, класс B и класс C. Каждый класс имел определенный диапазон IP-адресов, который был предназначен для определенных типов сетей.

25. Виды адресов IPv4: обычные и особые. Разновидности обычных адресов: глобально уникальные (белые), частные (серые) и специального назначения. Метод обеспечения уникальности «белых» IP адресов: IANA->RIR->ISP->потребитель. Правила назначения блоков адресов на сети.

IPv4-адреса могут быть разделены на две категории: обычные и особые.

Обычные адреса - это IPv4-адреса, которые используются для идентификации конкретного устройства в сети и для маршрутизации данных в Интернете. Обычные адреса могут быть разделены на три категории: глобально уникальные (белые), частные (серые) и специального назначения.
1.1. Глобально уникальные (белые) адреса
Глобально уникальные адреса, также известные как белые адреса, являются уникальными в рамках всего Интернета. Они предоставляются организацией IANA (Internet Assigned Numbers Authority) и распределяются региональными регистраторами интернет-ресурсов (RIR), которые затем передают эти адреса поставщикам услуг Интернета (ISP) и другим сторонам, нуждающимся в них. Затем ISP назначает эти адреса своим клиентам.
1.2. Частные (серые) адреса
Частные адреса, также называемые серыми адресами, используются в локальных сетях и не могут быть использованы для доступа к Интернету напрямую. Они предназначены для использования внутри сети, и служат для идентификации устройств в локальной сети. Частные адреса подразделяются на три диапазона: 10.0.0.0 до

10.255.255.255, 172.16.0.0 до 172.31.255.255 и 192.168.0.0 до 192.168.255.255.

1.3. Специального назначения адреса

Существует ряд IPv4-адресов, которые зарезервированы для специальных нужд. Некоторые из них используются для тестирования и отладки сетей, а другие - для определенных задач, таких как маршрутизация или мультикаст.

Задача мультикаста - это передача одного и того же сообщения (данных) от одного отправителя к нескольким получателям одновременно в рамках сети. В отличие от unicast, где данные передаются только одному адресату, и broadcast, где данные передаются всем узлам сети, multicast позволяет выбрать определенную группу узлов для передачи данных. Задача мультикаста широко используется в различных приложениях, таких как видеоконференции, онлайн-игры, потоковое видео и другие.

Особые адреса - это адреса, которые используются для специальных целей, например, для широковещательной передачи данных или для идентификации устройства, на котором выполняется запрос. Они не используются для идентификации конкретного устройства в сети и не могут быть назначены устройствам в качестве обычных адресов.

Широковещание (broadcast) - это метод передачи данных в компьютерных сетях, при котором сообщение отправляется одним узлом и доставляется всем остальным узлам в сети. Данные отправляются на специальный адрес, который называется широковещательным адресом (broadcast address), например, 255.255.255.255 для IPv4.

Метод обеспечения уникальности «белых» IP-адресов

Процесс распределения IP-адресов начинается с выделения блоков адресов организацией IANA (Internet Assigned Numbers Authority), которая ответственна за глобальное управление IP-адресами. Затем эти блоки адресов передаются региональным реестрам интернет-адресов (RIR - Regional Internet Registries):

RIPE Network Coordination Centre (RIPE NCC) для Европы, Ближнего Востока и некоторых стран бывшего Советского Союза;
RIR затем распределяет эти блоки адресов между поставщиками услуг Интернет (ISP - Internet Service Provider). ISP получают блоки адресов от RIR и используют их для своих потребностей. Например, один ISP может использовать блок адресов для сети своих клиентов.
Клиенты провайдеров, в свою очередь, получают IP-адреса от своих поставщиков услуг Интернет и используют их для подключения к сети, например, через свой домашний роутер или компьютер.
Таким образом, процесс распределения IP-адресов включает несколько уровней, начиная с глобальных организаций и заканчивая конечными пользователями. Каждый уровень отвечает за свою часть процесса и обеспечивает уникальность IP-адресов в пределах своей зоны ответственности.

Правила назначения блоков адресов на сети зависят от региона, в котором находится организация или пользователь, нуждающийся в назначении IP-адреса.

Обычно, ISP выделяют своим клиентам блоки адресов, достаточных для их потребностей. Это может быть блок из нескольких IPv4-адресов для домашнего пользователя или блок из тысяч IPv4-адресов для крупной компании.
В случае частных (серых) IP-адресов, правила использования не так жесткие. Частные адреса могут использоваться в любой локальной сети без необходимости получения их от IANA, RIR или ISP. Любая организация может использовать частные адреса для своих локальных сетей. Рекомендуется использовать частные адреса только для локальных сетей и использовать механизм Network Address Translation (NAT) для доступа к Интернету.

Механизм NAT - это способ, который позволяет устройствам в локальной сети выходить в Интернет через единственный общий IP-адрес.
Вот как это работает: когда ваше устройство отправляет запрос на сервер в Интернете, оно использует свой локальный IP-адрес (например, 192.168.1.2). Но этот адрес не уникален и не может быть использован для доступа к Интернету.
Важно отметить, что наличие уникального IPv4-адреса становится все более необходимым, поскольку число доступных адресов быстро исчерпывается. Поэтому большинство провайдеров интернет-услуг переходят на использование IPv6, которые обеспечивают гораздо больший адресный пространство, позволяя значительно расширить возможности подключения к Интернету.

26. Структура пакета IPv4 и назначение его основных полей (какую функцию они выполняют).

IPv4 (Internet Protocol version 4) является протоколом маршрутизации пакетов данных в Интернете. Пакет IPv4 состоит из заголовка и полезной нагрузки.

Заголовок IPv4 имеет фиксированную структуру и содержит следующие поля:
Версия (Version) - определяет версию протокола (IPv4 или IPv6). Размер поля 4 бита.
Длина заголовка (Header Length) - указывает размер заголовка. Размер поля 4 бита.
Тип сервиса (Type of Service) - используется для определения приоритета пакета и качества обслуживания. Размер поля 8 бит.
Приоритет пакета и качество обслуживания (Quality of Service, QoS) -

это параметры, которые используются для управления трафиком в сети, чтобы обеспечить более эффективное использование ресурсов.

Существует несколько типов приоритетов пакетов, включая:
High Priority (Высокий приоритет): такие пакеты получают наивысший приоритет и должны быть обработаны первыми. Это может включать голосовой или видео трафик, который требует низкой задержки и минимального пакетной потери.
Normal Priority (Нормальный приоритет): это наиболее распространенный тип приоритета пакетов. Они не имеют особого приоритета и обрабатываются по мере возможности. Данные с Normal Priority могут включать электронные письма, файлы, загрузки веб-страниц и другие типы трафика, которые не требуют быстрой обработки и доставки.
Low Priority (Низкий приоритет): такие пакеты получают наименьший приоритет и могут быть отложены до обработки других пакетов высшего приоритета. Данные с Low Priority могут включать резервирование пропускной способности для создания резервной копии данных, обновления программного обеспечения или любые другие процессы, которые не критически зависят от времени. В случае низкого приоритета такие данные будут обрабатываться после всех остальных пакетов, что может привести к задержкам.

Качество обслуживания определяется политиками управления трафиком, которые могут учитывать приоритетность пакетов, скорость передачи и другие факторы.

Некоторые типы качества обслуживания включают:
Best Effort (Лучшие усилия): это означает, что пакеты обрабатываются по мере возможности без какого-либо приоритета.
Guaranteed (Гарантированное): пакеты с гарантированным качеством обслуживания будут обработаны первыми и иметь наименьшую задержку при передаче данных в сети.
Controlled Load (Контролируемая нагрузка): это политика управления

трафиком, которая предоставляет определенный уровень качества обслуживания для определенного потока трафика, чтобы минимизировать задержки и потери пакетов.
Различные типы приоритетов пакетов и качества обслуживания используются для оптимизации производительности сети в зависимости от ее конкретных требований.
Длина пакета (Total Length) - указывает общий размер пакета (заголовок + полезная нагрузка) в октетах. Размер поля 16 бит.
Идентификатор (Identification) - уникальный идентификатор пакета, используемый для объединения фрагментированных пакетов. Размер поля 16 бит.
В сетевой связи, пакеты данных могут быть разделены на фрагменты для передачи по каналу связи.
Структура пакета IPV4 состоит из:
Флаги (Flags) - используются для указания наличия фрагментации пакета и определяют положение фрагмента в исходном пакете. Размер поля 3 бита.
Смещение фрагмента (Fragment Offset) - смещение фрагмента относительно начала пакета в 8-байтных блоках. Размер поля 13 бит.
Время жизни (Time to Live) - задает время жизни пакета (количество промежуточных маршрутизаторов, которые могут обработать пакет). Размер поля 8 бит.
Протокол (Protocol) - указывает протокол верхнего уровня (TCP, UDP, ICMP), который используется для передачи данных внутри пакета. Размер поля 8 бит.
Контрольная сумма заголовка (Header Checksum) - используется для проверки целостности заголовка пакета. Размер поля 16 бит.
IP-адрес источника (Source IP Address) - IP-адрес отправителя пакета. Размер поля 32 бита.
IP-адрес назначения (Destination IP Address) - IP-адрес получателя пакета. Размер поля 32 бита.

После заголовка следует полезная нагрузка, которая может содержать данные прикладного уровня (например, HTTP, FTP, SMTP).

27. Коммутация пакетов IPv4: таблицы маршрутов, алгоритм коммутации, варианты действий, выполняемых IP модулем.

51 слайд (2018)!!!

Коммутация пакетов IPv4 - это процесс передачи пакетов между узлами сети, который осуществляется на уровне сетевого протокола. Для того чтобы пакеты могли быть доставлены от отправителя к получателю, необходимо использовать таблицы маршрутов и алгоритм коммутации.
Таблицы маршрутов содержат информацию о том, каким образом следует доставлять пакеты в конечный узел. В таблице маршрутов для каждого возможного адреса назначения указывается следующий шаг, который должен выполнить коммутатор для передачи пакета. Например, если пакет должен быть доставлен из одной сети в другую, то в таблице маршрутов будет указан адрес следующего коммутатора, через который должен быть передан пакет.
Алгоритм коммутации включает в себя несколько этапов:
Получение пакета - коммутатор получает пакет на свой интерфейс и проверяет его заголовки.
Определение адреса назначения - коммутатор проверяет адрес назначения пакета и сравнивает его с записями в таблице маршрутов.
Выбор порта - на основе информации из таблицы маршрутов коммутатор выбирает порт, через который должен быть передан пакет.
Отправка пакета - коммутатор отправляет пакет на выбранный порт и переходит к следующему пакету.

В зависимости от ситуации, IP модуль может выполнить одно из нескольких действий:
Прямая доставка - если адрес назначения пакета принадлежит устройству, на котором работает коммутатор, то пакет передается напрямую, без поиска в таблице маршрутов.
Доставка по подсети - если адрес назначения пакета принадлежит устройству, находящемуся в той же подсети, что и отправитель, то коммутатор использует таблицу маршрутов для определения порта, через который нужно передать пакет.
Доставка по маршруту - если адрес назначения пакета находится в другой подсети, то коммутатор использует таблицу маршрутов для определения следующего коммутатора, через который нужно передать пакет.
Отбрасывание пакета - если запись в таблице маршрутов не найдена или возникла ошибка при обработке пакета, коммутатор может отбросить пакет или отправить его обратно к отправителю с сообщением об ошибке.
В целом, коммутация пакетов IPv4 осуществляется на основе таблиц маршрутов и алгоритма коммутации. IP модуль выполняет различные действия в зависимости от адреса назначения пакета и информации из таблицы маршрутов, что позволяет передавать пакеты по всей сети Интернет.

28. MTU. Функции фрагментации пакетов IPv4 и Path MTU discovery.

MTU (Maximum Transmission Unit) - это максимальный размер пакета, который может быть передан через сеть без фрагментации. Размер MTU определяется техническими характеристиками сетевых устройств и протоколов, которые используются в сети.
Фрагментация пакетов IPv4 - это процесс разделения большого пакета на несколько меньших фрагментов для передачи через сеть с

меньшим MTU. Фрагментация позволяет передавать большие пакеты по сети, но она также может вызывать проблемы, например, задержки в доставке пакетов и потерю данных.

Path MTU discovery (PMTUD) - это процесс автоматического определения максимального размера пакета, который может быть передан без фрагментации через сеть от источника к назначению. PMTUD используется для избежания фрагментации IP-пакетов во время их прохождения через различные сети с разными MTU (Maximum Transmission Unit).
В ходе PMTUD исходный узел начинает отправку IP-пакетов с определенным значением DF (Don't Fragment) - это битовый флаг в заголовке IP-пакета, которое указывает на запрет фрагментации пакета. Если пакет не может пройти через какую-то сеть на своем пути из-за ограничения MTU, сетевое устройство на этом участке отправляет обратно сообщение ICMP с ошибкой "Fragmentation Required and Don't Fragment was Set", содержащее MTU этой сети. Исходный узел затем изменяет размер пакета до максимально допустимого значения и повторяет процедуру до тех пор, пока пакет не достигнет своего назначения.
Функции фрагментации пакетов IPv4 включают следующие шаги:
При получении пакета коммутатор проверяет его размер и сравнивает его с MTU порта, через который пакет должен быть отправлен.
Если размер пакета больше, чем MTU порта, то коммутатор разделяет пакет на несколько фрагментов по размеру MTU порта.
Каждый фрагмент получает уникальный идентификатор, который используется для объединения фрагментов обратно в оригинальный пакет.
Фрагменты отправляются через сеть к адресу назначения.
Получатель собирает все фрагменты обратно в единый пакет на основе их идентификаторов.

Таким образом, фрагментация пакетов IPv4 и Path MTU discovery являются важными функциями для эффективной передачи данных в сети. Фрагментация позволяет передавать большие пакеты через сеть, но она также может вызывать проблемы. Path MTU discovery помогает избежать фрагментации и определяет оптимальный размер MTU в сети.

29. Обмен служебными сообщениями в процессе функционирования протокола IPv4. Протокол ICMP: виды сообщений, формат сообщений. При каких обстоятельствах применяются.

Протокол IPv4 (Internet Protocol version 4) используется для обмена пакетами данных в сетях TCP/IP. В процессе функционирования этого протокола могут передаваться различные служебные сообщения с помощью протокола ICMP (Internet Control Message Protocol)

Протокол ICMP предназначен для отправки сообщений об ошибках и событиях, происходящих в процессе передачи данных. Он может использоваться для проверки связи между устройствами в сети и определения наилучшего маршрута для передачи данных.
Виды сообщений, которые могут быть отправлены через протокол ICMP, включают в себя:
Echo Request (Ping) - запрос эхо-сообщения для проверки доступности удаленного узла. Это один из наиболее часто используемых типов ICMP-сообщений.
Echo Reply - ответ на запрос эхо-сообщения. Если удаленный узел доступен и работает корректно, то он должен отправить этот тип ICMP-сообщения в ответ на запрос эхо-сообщения.
Destination Unreachable - сообщение об ошибке, которое отправляется

обратно отправителю, если удаленный узел недоступен или если маршрут для доставки пакета не может быть найден.
Time Exceeded - сообщение об ошибке, которое отправляется обратно отправителю, если пакет не может быть доставлен в срок. Это может произойти, например, если пакет зациклился в сети или если его маршрут был изменен.
Redirect - сообщение, которое указывает на более эффективный маршрут для доставки пакета.

Формат сообщений ICMP состоит из заголовка и данных. Заголовок содержит информацию о типе и коде сообщения, а также контрольную сумму для проверки целостности сообщения. Данные могут содержать дополнительную информацию, связанную с сообщением.
Применение протокола ICMP зависит от конкретных обстоятельств, но обычно он используется для диагностики сетевых проблем, проверки доступности удаленных узлов и определения наилучшего маршрута для передачи данных.
Например, команда ping использует протокол ICMP для отправки запросов эхо-сообщений и получения ответов от удаленного узла. Также протокол ICMP может быть использован для приема и обработки сообщений об ошибках, когда возникают проблемы в процессе передачи данных.

Протокол ICMP (Internet Control Message Protocol) используется в сетевых технологиях для обеспечения коммуникации между различными устройствами, а также для диагностики и устранения неполадок в сетях.

Основное применение протокола ICMP:
Проверка доступности хостов: Одним из наиболее распространенных способов проверки доступности удаленного хоста является использование команды ping, которая отправляет эхо-запросы с помощью протокола ICMP. Когда удаленный хост получает эти запросы, он должен отправить в ответ эхо-ответы.

Обнаружение маршрутов: Протокол ICMP может быть использован для обнаружения маршрутов в сети. Если происходит изменение маршрута, то протокол ICMP может отправить сообщение об ошибке "Redirect" с информацией о новом маршруте.
Определение MTU: Протокол ICMP также может быть использован для определения максимального размера передаваемых данных (MTU). Если пакет имеет размер большего, чем MTU, то он может быть разделен на несколько фрагментов. В этом случае протокол ICMP будет отправлять сообщения "Fragmentation needed", указывающие на необходимость фрагментации пакета.
Обнаружение ошибок: Протокол ICMP может использоваться для обнаружения ошибок в процессе передачи данных. Если возникают проблемы при отправке или получении пакетов, то устройства могут отправлять сообщения "Destination Unreachable" или "Time Exceeded", чтобы оповестить отправителя о возникших проблемах.
Расширенные функции: Существуют и другие типы сообщений ICMP, которые могут быть использованы для выполнения расширенных функций, таких как определение IP-адреса ближайшего маршрутизатора (traceroute).
Протокол ICMP играет важную роль в обеспечении стабильности и надежности сетевых соединений. Благодаря своей гибкости и широкому спектру функций, он может быть использован в различных ситуациях для диагностики и устранения проблем в сетях.

30. Средства элементарной диагностики IP сетей: ping и traceroute. Описать принцип действия и приблизительную методологию применения.

Средства элементарной диагностики IP сетей - это инструменты, которые помогают определить работоспособность сетевого устройства

и проблемы в сетевой связи. Два наиболее распространенных инструмента для этой цели - это ping и traceroute.
Ping - это утилита, которая используется для проверки доступности узла в сети и измерения времени отклика устройства на отправленный ему запрос. Принцип работы Ping заключается в отправке ICMP-пакета (Internet Control Message Protocol) на указанный IP-адрес и ожидании ответа. Если устройство получает ICMP-запрос, то оно отправляет в ответ ICMP-ответ, содержащий время прохождения пакета до устройства и обратно. При этом Ping может отправлять несколько пакетов за раз и выводить статистику по их получению. В результате, если устройство отвечает на ICMP-запросы, значит, оно доступно, а время отклика позволяет определить скорость передачи данных между устройствами.

Методология применения Ping очень проста. Его можно использовать следующим образом:
Открыть командную строку (в Windows) или терминал (в MacOS/Linux).
Ввести команду ping и IP-адрес устройства, которое требуется проверить (например, ping 192.168.0.1).
Дождаться завершения выполнения команды Ping и прочитать результаты.

Traceroute - это сетевая утилита, которая используется для определения маршрута, по которому проходят пакеты данных в сети Интернет. Она работает по принципу последовательной отправки ICMP-запросов (Internet Control Message Protocol) на удаленный узел с постепенным увеличением значения TTL (Time To Live), начиная с единицы.
Каждый маршрутизатор на пути следования пакета уменьшает значение TTL на единицу. Если значение достигает нуля, то маршрутизатор отбрасывает пакет и отправляет обратно сообщение "Time exceeded" (время жизни истекло) отправителю. Traceroute фиксирует время прохождения каждого маршрутизатора и выводит

информацию о задержке (ping time) и IP-адресе каждого узла на маршруте.
Таким образом, Traceroute позволяет узнать количество и порядок маршрутизаторов, через которые проходит пакет до достижения конечного узла, а также определить проблемные сегменты сети, где возможны задержки или потери пакетов. Это делает эту утилиту полезной при диагностике проблем с сетью и выборе оптимального маршрута для передачи данных.
Методология применения Traceroute включает следующие шаги:
Открыть командную строку (в Windows) или терминал (в MacOS/Linux).
Ввести команду traceroute и IP-адрес устройства, до которого требуется определить маршрут (например, traceroute 192.168.0.1).
Дождаться завершения выполнения команды Traceroute и прочитать результаты.
Изучить список устройств (маршрутизаторов), через которые проходит пакет, и время, затраченное на его прохождение через каждое из устройств.
В целом, Ping и Traceroute являются важными инструментами для диагностики IP-сетей. При правильном использовании они помогают определить доступность устройства, проверить скорость передачи данных и определить маршрут передачи данных в сети.
TTL (Time To Live) - это поле в заголовке IP-пакета, которое указывает на количество маршрутизаторов или сетевых устройств, через которые может пройти пакет, прежде чем он будет отброшен. Каждый раз, когда пакет проходит через маршрутизатор, значение TTL уменьшается на единицу. Если значение TTL достигает нуля, то маршрутизатор должен отбросить пакет и отправить обратно ICMP-сообщение об ошибке.
Таким образом, значение TTL задает максимальное количество сетевых устройств, которые пакет может пройти, и предотвращает зацикливание пакетов в сети. Обычно значение TTL устанавливается в начале маршрутизации и увеличивается по мере прохождения пакета

через сеть. Значение TTL может быть изменено при пересылке пакета через другой маршрутизатор.
Значение TTL может быть различным для разных типов пакетов и настраивается в зависимости от конкретных потребностей сети. Например, если TTL установлен очень высоким значением, то это может привести к затратам ресурсов сети, так как пакет может проходить через большое количество маршрутизаторов. Если же TTL установлен слишком низким, то пакет могут быть отброшен раньше времени и не достигнуть своего назначения.

31. Особенности функционирования IP на многоточечных сетях (звеньях). MAC адреса: структура, назначение, особые адреса. Роль MAC адресов при передаче кадров по многоточечному звену в различных режимах: unicast, broadcast, multicast.

IP (Internet Protocol) является протоколом сетевого уровня, который обеспечивает маршрутизацию пакетов данных через интернет и другие сети. Однако для передачи пакетов данных по физической сети используются другие протоколы, включая Ethernet.
Ethernet - это стандартный протокол локальных сетей (LAN), который использует метод доступа CSMA/CD (Carrier Sense Multiple Access with Collision Detection). В Ethernet каждое устройство имеет свой уникальный адрес MAC (Media Access Control), который используется для идентификации устройства на физическом уровне сети.
Метод доступа CSMA/CD (Carrier Sense Multiple Access/Collision Detection) - это один из методов, используемых в сетях Ethernet для контроля доступа к среде передачи данных. Он обеспечивает доступ к

среде передачи для нескольких устройств, позволяя им конкурировать за передачу данных.
Принцип работы метода CSMA/CD заключается в том, что перед началом передачи каждое устройство производит прослушивание среды передачи на наличие других передач. Если среда свободна, то устройство начинает передачу своих данных. Если же в момент передачи данные другого устройства уже находятся на среде передачи, возникает коллизия (collision).
Каждое устройство, обнаружив коллизию, немедленно прекращает передачу и отправляет специальный сигнал (jam signal), оповещая другие устройства о возникновении коллизии. После этого все устройства должны подождать случайный интервал времени перед следующей попыткой передачи. Это помогает избежать повторной коллизии, так как вероятность возникновения коллизии на очередной попытке передачи будет меньше.

MAC-адрес состоит из 48 бит (6 байт) и записывается в формате шестнадцатеричного числа, разделенного двоеточиями. Первые три байта (24 бита) представляют производителя оборудования, а оставшиеся три байта (24 бита) уникальны для каждого устройства производителя.
Назначение MAC-адреса заключается в том, чтобы обеспечить уникальную идентификацию устройства на физическом уровне, что позволяет определять отправителя и получателя кадров данных в сети Ethernet. Каждый раз, когда устройство отправляет кадр данных, оно указывает MAC-адрес получателя в заголовке кадра.

При передаче кадров данных по многоточечной сети Ethernet (звену) используются различные режимы: unicast, broadcast и multicast. Unicast - это режим передачи данных, при котором кадр отправляется только одному устройству с определенным MAC-адресом получателя. В этом случае отправитель указывает MAC-адрес получателя в заголовке кадра. Broadcast - это режим передачи данных, при котором

кадр отправляется всем устройствам в сети Ethernet. Для этого используется специальный широковещательный MAC-адрес (FF:FF:FF:FF:FF:FF), который указывается в заголовке кадра.
Multicast - это режим передачи данных, при котором кадр отправляется группе устройств, имеющих определенный MAC-адрес. Для этого используется специальный MAC-адрес мультикастной группы, который указывается в заголовке кадра. Устройства, которые принадлежат к этой группе, получают копию кадра данных.
Когда компьютеры общаются в локальной сети, они используют протокол Ethernet. Каждое устройство в этой сети имеет уникальный идентификатор - MAC-адрес. Когда один компьютер хочет отправить данные другому компьютеру, он указывает MAC-адрес получателя в специальном поле заголовка пакета данных.
Но порой возникает необходимость отправлять данные нескольким устройствам одновременно. Например, когда нужно транслировать видео на всех компьютерах в классной комнате, или когда компьютеры должны получить обновление программного обеспечения из централизованного источника. В этом случае используется механизм мультикастинга.
Мультикастинг - это режим передачи данных, при котором один и тот же пакет данных отправляется одновременно нескольким устройствам. Мультикастинговые пакеты имеют специальный IP-адрес, называемый мультикастным адресом. Этот адрес определяет группу устройств, которая должна получить эти данные.
Каждая группа устройств, подписавшаяся на мультикастный адрес, имеет свой уникальный MAC-адрес, который называется MAC-адресом мультикастной группы. Он начинается с префикса "01:00:5E", за которым следует 23 бита номера группы. При передаче мультикастингового пакета компьютер указывает этот MAC-адрес в качестве получателя. Таким образом, все устройства, подключенные к этой группе и имеющие соответствующий MAC-адрес, получают эти данные одновременно.

В стандарте Ethernet и связанных сетевых технологиях существует несколько особых MAC-адресов. Вот несколько примеров:
MAC-адрес широковещательной группы (broadcast): FF:FF:FF:FF:FF:FF. Он используется для передачи данных всем устройствам в сети.
MAC-адрес мультикастной группы: начинается с префикса "01:00:5E", за которым следуют еще 23 бита, представляющие номер группы. Он используется для передачи данных нескольким устройствам одновременно, подключенным к определенной группе.
MAC-адрес Loopback-интерфейса: 00:00:00:00:00:00. Он используется для локального тестирования сетевых приложений и оборудования.
MAC-адрес настроенного интерфейса: 02:XX:XX:YY:YY:YY. Он используется для определения настроенных MAC-адресов виртуальных машин или других вспомогательных механизмов.
Виртуальные машины обычно имеют виртуальные сетевые адаптеры, которые могут быть сконфигурированы для использования определенных MAC-адресов. Для этого используются различные методы в зависимости от конкретной системы виртуализации. Например, в системе виртуализации VMware можно задать MAC-адрес виртуального адаптера вручную или автоматически генерировать его. В Hyper-V от Microsoft также предусмотрена возможность задания статического MAC-адреса для виртуальных адаптеров.
Кроме того, вспомогательные механизмы, такие как контейнеры, также могут иметь свои собственные сетевые интерфейсы с уникальными MAC-адресами, которые могут быть настроены при создании контейнера.
В целом, настройка MAC-адресов виртуальных машин и других вспомогательных механизмов будет зависеть от выбранной системы виртуализации и специфических требований к инфраструктуре.
MAC-адрес собственного интерфейса: это MAC-адрес сетевого интерфейса, присвоенный производителем оборудования. Он используется для уникальной идентификации сетевого оборудования в локальной сети.

32. Протокол ARP: назначение, место в протокольном стеке, формат пакетов. ARP таблица. Взаимодействие с IP модулем. Выполнение процедуры поиска MAC адреса (ARP-resolve). Алгоритм обработки ARP-пакета.

Протокол ARP (Address Resolution Protocol) используется для разрешения (определения) MAC-адреса устройства по его IP-адресу в локальной сети. Это необходимо для передачи данных между устройствами на канальном уровне.

ARP работает на сетевом уровне модели OSI и является частью протокольного стека TCP/IP. Он позволяет обеспечить соответствие между логическими адресами (IP-адресами) и физическими адресами (MAC-адресами) в локальной сети.
Формат пакетов ARP состоит из следующих полей:
Тип аппаратного адреса (Hardware Type) - определяет тип сетевого интерфейса: Ethernet, Token Ring, FDDI, и т. д.
Тип протокола (Protocol Type) - указывает тип сетевого протокола, для которого выполняется разрешение адреса.
Длина аппаратного адреса (Hardware Address Length) - размер аппаратного адреса в байтах.
Длина протокольного адреса (Protocol Address Length) - размер протокольного адреса в байтах.
Код операции (Operation Code) - определяет тип ARP-сообщения: запрос или ответ.
Аппаратный адрес отправителя (Sender Hardware Address) - MAC-адрес

отправителя.
Протокольный адрес отправителя (Sender Protocol Address) - IP-адрес отправителя.
Аппаратный адрес получателя (Target Hardware Address) - MAC-адрес получателя. В ARP-запросах это поле не заполняется, а в ARP-ответах содержит MAC-адрес устройства, для которого запрашивался IP-адрес.
Протокольный адрес получателя (Target Protocol Address) - IP-адрес устройства, для которого выполняется разрешение MAC-адреса.

ARP таблица - это таблица, в которой хранятся соответствия между IP-адресами и MAC-адресами устройств, доступных в локальной сети. ARP таблица обычно заполняется автоматически при обмене данными между устройствами, но может быть также заполнена вручную.

Взаимодействие с IP-модулем происходит следующим образом: при передаче пакета на уровне IP, IP-модуль проверяет, находится ли получатель в локальной сети. Если да, он выполняет процедуру ARP-resolve для определения MAC-адреса получателя и добавляет его в заголовок пакета.
Алгоритм обработки ARP-пакета выглядит следующим образом:
Получение ARP-пакета.
Проверка типа аппаратного адреса (обычно Ethernet).
Проверка длины адресов.
Проверка типа протокола (обычно IPv4 или IPv6).
Проверка длины протокольных адресов.
Обновление ARP таблицы, если необходимо.
Отправка ответного ARP-пакета, если получен запрос на свой IP-адрес.
Процедура поиска MAC адреса (ARP-resolve) выполняется в следующем

порядке:
Проверка ARP таблицы на наличие записи для нужного IP-адреса.
Если запись найдена, отправляется запрос на этот MAC-адрес.
Если запись не найдена, отправляется широковещательный ARP-запрос (broadcast). В запросе содержится IP-адрес запрашиваемого устройства.
Устройства, которые имеют соответствующий IP-адрес, отвечают собственным MAC-адресом.
Полученный MAC-адрес добавляется в ARP таблицу.

33. Функциональное описание канального уровня ЭМВОС - перечислить и описать функции канального уровня, описать взаимодействие с другими уровнями ЭМВОС, привести классификацию модулей канального уровня.

Канальный уровень в модели ЭМВОС (OSI) - это второй уровень, который отвечает за передачу данных между соседними узлами сети. Основная задача канального уровня состоит в обеспечении надежной передачи данных, управлении потоком и обнаружении ошибок.
Функции канального уровня:
Контроль доступа к среде передачи данных
Формирование кадров данных
Управление потоком передачи данных
Обнаружение и исправление ошибок в переданных данных
Канальный уровень взаимодействует с другими уровнями ЭМВОС в следующих аспектах:
Связывается с физическим уровнем для получения доступа к физической среде передачи данных.
Передает данные на сетевой уровень, формируя кадры данных и

добавляя информацию для контроля ошибок.
Получает от сетевого уровня данные для передачи и управления потоком.

Модули канального уровня могут быть классифицированы по различным параметрам, таким как:
Режим работы: полудуплексный или дуплексный.
Полудуплексный режим подразумевает, что передача данных возможна только в одном направлении за один период времени. Например, если два устройства обмениваются данными через кабель, каждое из них должно ждать своей очереди на передачу.
В дуплексном режиме связи данные могут передаваться в обоих направлениях одновременно, что повышает эффективность обмена информацией между устройствами.
Тип среды передачи данных: проводное или беспроводное.
Проводная среда передачи данных использует кабели и соединители для передачи информации между устройствами.
Беспроводная среда передачи данных использует радиоволны или инфракрасные лучи для передачи данных без использования проводов.

34. Кадрирование - описание проблемы, методы ее решения. Кодопрозрачность и способы ее достижения при различных методах кадрирования.

В контексте компьютерных сетей, кадрирование может использоваться для изменения размера передаваемых данных. Например, когда данные передаются через сеть, часто необходимо уменьшить их размер, чтобы ускорить процесс передачи и сэкономить пропускную способность сети. Однако, при этом могут возникнуть некоторые проблемы.

Одной из главных проблем кадрирования в контексте компьютерных сетей является потеря информации. Если мы обрезаем данные, то теряем часть информации, которая могла бы быть важна для корректной обработки или анализа этих данных. Кроме того, при изменении размера данных могут возникать ошибки искажения или повреждения данных.
Для решения проблемы потери информации при кадрировании в компьютерных сетях можно использовать различные методы. Один из таких методов - сжатие данных. Сжатие позволяет уменьшить размер данных без его обрезания, сохраняя все данные в исходном виде. Для этого используются различные алгоритмы сжатия данных, например, GZIP или LZ77.
Кроме того, можно использовать технику фрагментации данных. Фрагментация позволяет разбить данные на более мелкие части, которые затем передаются по сети. Это может помочь уменьшить нагрузку на сеть и ускорить процесс передачи данных. Однако, при использовании этого метода необходимо обеспечить целостность данных, чтобы они были корректно восстановлены после передачи.
Кодопрозрачность также является важным аспектом кадрирования в контексте компьютерных сетей. Кодопрозрачность означает, что изменения, сделанные при кадрировании, не должны повлиять на код или данные, связанные с изображением. Для достижения кодопрозрачности можно использовать сжатие данных с сохранением формата, либо использовать протоколы передачи данных, которые поддерживают целостность данных, например, TCP.

35. Контроль правильности передачи данных - принцип контроля, кодовое пространство, метод обнаружения ошибок, способ построения механизма

контроля. Эффективность контролирующего механизма – избыточность, обнаруживающая способность.

Контроль правильности передачи данных - это процесс проверки целостности и достоверности передаваемых данных. Цель контроля правильности передачи данных состоит в том, чтобы обнаруживать и исправлять ошибки в данных, которые могут возникнуть в процессе передачи по сети или хранения на устройствах.
Один из основных принципов контроля правильности передачи данных - использование контрольных сумм.
Контрольная сумма - это число, которое вычисляется на основе содержимого передаваемых данных. При получении данных, контрольная сумма вычисляется повторно и сравнивается с исходной контрольной суммой. Если они не совпадают, значит, данные были повреждены, и необходимо запросить их повторную передачу.
Для вычисления контрольной суммы используют кодовые пространства, которые представляют множество всех возможных кодов, которые могут быть использованы для передачи или хранения информации. Каждый код представляет собой последовательность символов (обычно битов), которая соответствует определенному значению или символу. Кодовое пространство для вычисления контрольной суммы - это множество всех возможных последовательностей символов (обычно битов), которые могут быть использованы для создания контрольной суммы. Используемое кодовое пространство зависит от способа вычисления контрольной суммы и требуемой степени защиты данных. Например, одним из наиболее распространенных кодовых пространств является CRC (Cyclic Redundancy Check), который используется для обнаружения ошибок в бинарных данных.
Существует несколько методов обнаружения ошибок с

использованием контрольных сумм. Один из таких методов - одиночная проверка. При этом методе контрольная сумма вычисляется только один раз, после чего данные передаются по сети. Если при получении данных контрольная сумма не совпадает с исходной, то происходит повторная передача данных.
Другой метод - многократная проверка. При этом методе контрольная сумма вычисляется несколько раз в течение передачи данных. Это позволяет обнаруживать ошибки в более коротких интервалах времени и уменьшить количество повторных передач.
Механизм контроля правильности передачи данных может быть построен на основе различных протоколов передачи данных, таких как UDP (User Datagram Protocol) или TCP (Transmission Control Protocol).
Протокол UDP не гарантирует доставку и целостность передаваемых данных, что позволяет уменьшить нагрузку на сеть. Однако, при использовании UDP могут возникнуть ошибки в передаче данных, например, данные могут быть повреждены или потеряны. Для обнаружения таких ошибок в протоколе UDP используется контрольная сумма.

Протокол TCP гарантирует доставку и целостность передаваемых данных, следит за порядком их доставки, а также обеспечивает повторную передачу потерянных пакетов данных. Контроль правильности передачи данных в протоколе TCP реализуется с помощью ACK-пакетов, которые отправляются для подтверждения получения пакетов данных и определения недостающих пакетов. Также в протоколе TCP используется проверка целостности и аутентичности данных с помощью специального поля в заголовке пакета - контрольной суммы.
ACK-пакет (сокр. от acknowledgment packet) - это пакет данных, который используется для подтверждения получения другого пакета данных в сетевой связи между устройствами. Обычно ACK-пакет отправляется после получения пакета данных от другого устройства,

чтобы сообщить об успешной передаче или получении данных. Он содержит информацию о номере подтверждаемого пакета и может также содержать дополнительные данные, такие как временные метки или контрольные суммы для обеспечения целостности данных. ACK-пакеты широко используются в протоколах TCP/IP (Transmission Control Protocol/Internet Protocol), которые обеспечивают надежную доставку данных в Интернете.

Аутентичность данных (data authenticity) - это свойство информации, которое гарантирует, что данные были созданы или изменены только допустимым и авторизованным источником, а не поддельным или недобросовестным.

Другими словами, аутентичность данных означает, что данные являются подлинными и соответствуют тому, что заявлено.

Другие способы построения механизма контроля правильности передачи данных могут включать в себя использование дополнительных битов для обнаружения ошибок или применение кодовых слов, которые позволяют исправлять ошибки в передаваемых данных.

Дополнительные биты для обнаружения ошибок могут быть добавлены к передаваемым данным с использованием различных алгоритмов. Один из таких алгоритмов - это циклический избыточный код (CRC), который является широко используемым механизмом контроля правильности передачи данных.

Алгоритм CRC работает следующим образом:

ųИсходные данные рассматриваются как последовательность битов.

ųФормируется блочный код, состоящий из исходных данных и дополнительных контрольных битов. Для создания контрольных битов используется операция деления многочлена.

ųПолученный блочный код передается по каналу связи.

ųПолучатель вычисляет контрольные биты на основе принятого блочного кода, используя тот же самый многочлен. Если полученная контрольная сумма совпадает с переданной контрольной суммой, то

данные считаются доставленными без ошибок.
Многочлен, который используется для генерации контрольных битов, называется генераторным многочленом.
При использовании CRC для каждого блока данных создается уникальная проверочная сумма, которая содержит информацию о последовательностях битов и другой дополнительной информации. Получатель данных вычисляет проверочную сумму, основываясь на принятых данных, и сравнивает ее со значением, полученным вместе с переданными данными. Если значения не совпадают, то это указывает на возможное изменение данных в процессе передачи, и получатель может запросить повторную передачу.
Кодовые слова, которые позволяют исправлять ошибки в передаваемых данных, являются другим способом обеспечения надежности при передаче данных. Например, код Хемминга использует специальные кодовые слова, которые позволяют исправлять одну или несколько ошибок в передаваемых данных. Это достигается путем добавления дополнительных битов информации (кодовых слов) к передаваемым данным. Код Хемминга - это метод обнаружения и исправления ошибок в передаваемых данных. В коде Хемминга используются кодовые слова, которые состоят из информационных битов и проверочных битов, добавленных для обнаружения и исправления ошибок.
Каждый проверочный бит определяется как побитовая сумма некоторых информационных битов. Количество информационных битов в каждом блоке данных зависит от количества проверочных битов: чем больше проверочных битов, тем меньше информационных битов.
При передаче данных кодируются с помощью кодовых слов. Если произошла ошибка в передаче, то приемник вычисляет проверочные биты и сравнивает их со значениями, полученными вместе с информационными битами. Если значение проверочного бита не соответствует вычисленному значению, то произошла ошибка и

приемник может использовать проверочную информацию для определения места ошибки.
Если возможно исправить одну ошибку, то приемник может изменить единственный бит, который был ошибочным. Если ошибок было несколько, то приемник может определить биты, которые содержат ошибки, но не может точно установить, какие же значения этих битов должны быть.
По мере приема данных получатель вычисляет контрольные суммы и/или проверяет кодовые слова, чтобы обнаружить ошибки. Если были обнаружены ошибки, то получатель может попросить отправителя повторно передать данные или исправить ошибки входных данных, используя специальные алгоритмы.
Эффективность контролирующего механизма зависит от его избыточности и обнаруживающей способности.
Избыточность контролирующего механизма означает, что он должен быть способен обнаруживать как можно больше ошибок, чтобы минимизировать количество повторных передач.
Обнаруживающая способность контролирующего механизма определяет его способность обнаруживать различные типы ошибок.
Чем выше избыточность и обнаруживающая способность, тем более эффективным будет механизм контроля правильности передачи данных.

36. Контроль по паритету, матричный код, код Хэмминга. Исправляющая способность. Недостатки паритетных кодов.

Контроль по паритету – это метод обнаружения ошибок в передаче данных, основанный на проверке четности битового потока. Для этого добавляется дополнительный бит, называемый битом четности или

битом проверки четности (Parity Check Bit), который определяет, должно ли количество единиц в передаваемом сообщении быть четным или нечетным. Например, для контроля по четности с четным числом единиц добавляется "0", а для контроля по четности с нечетным числом единиц добавляется "1".
Матричный код – это метод кодирования информации, основанный на использовании матриц. В матричных кодах каждое сообщение представлено в виде вектора из битовых символов, а для его кодирования используется матрица. Существует несколько различных типов матричных кодов, но одним из наиболее распространенных является блочный матричный код. В блочном матричном коде информационное сообщение разбивается на блоки фиксированной длины, которые затем кодируются отдельно. Для создания матрицы кодирования используются генерирующие матрицы, которые позволяют преобразовать информационные данные в кодовые слова. Кодовые слова обычно имеют большую длину, чем исходное сообщение,
что позволяет исправлять ошибки при передаче данных. При декодировании матричного кода, полученное кодовое слово умножается на проверочную матрицу, которая помогает определить наличие ошибок и их местонахождение. Если была обнаружена ошибка, то декодер может использовать синдром, который вычисляется на основе проверочной матрицы, чтобы определить, какой бит был изменен при передаче данных, и исправить его. Матричные коды используются в различных областях, включая телекоммуникации, компьютерные сети, цифровое телевидение и др. Они позволяют эффективно использовать доступную пропускную способность канала связи и обеспечивают надежную передачу данных.

Простейшее объяснение алгоритма кода Хэмминга на пальцах здесь:

https://www.boyarkirk.ru/?go=all/prostoy-kod-hemminga-praktika/

Код Хэмминга – это тип блочного кодирования (частный случай), который используется для исправления одиночных ошибок в передаваемых данных. Он был разработан Ричардом Хэммингом в 1950-х годах. Код Хэмминга расширяет информационный блок данных, добавляя дополнительные биты контроля ошибок. Каждый бит контроля ошибок используется для проверки определенной комбинации битов данных. Это позволяет обнаруживать и исправлять одиночные ошибки в передаваемых данных. Например, если в сообщении произошла ошибка в одном из битов данных, проверочные биты могут использоваться для определения, какой бит был изменен, и исправить его.
В коде Хэмминга используется матрица проверки четности, которая определяет, какие биты данных должны быть проверены на наличие ошибок. Если при передаче данных происходит ошибка, то полученный код будет несоответствующим одному из возможных кодов, которые можно сформировать с помощью матрицы проверки четности. Таким образом, принимающая сторона может определить, что произошла ошибка, и исправить ее.
Общая формула для построения матрицы проверки четности имеет вид:
H = [PT | I],
где P - матрица перестановок (порядок битов данных), T - транспонированная матрица паритетных чеков (какие комбинации битов проверять на наличие ошибок), I - единичная матрица размерности k x k (k - количество бит данных).
Каждый столбец транспонированной матрицы соответствует определенному биту данных, а каждая строка - определенному биту контроля ошибок.

Исправляющая способность – это количество ошибок, которое может быть обнаружено и исправлено с помощью данного кода. Например,

код Хэмминга сможет исправить одиночные ошибки в блоках данных, но не сможет корректировать множественные ошибки.
Недостатки паритетных кодов – это то, что они не могут обнаруживать ошибки, если число ошибок в передаваемом сообщении больше одной. Также паритетные коды не могут исправлять ошибки, только обнаруживать их. Кроме того, добавление дополнительного бита четности увеличивает объем передаваемых данных на 1 бит на каждые 7 бит информации, что неэффективно при передаче больших объемов данных.

37. Циклические избыточные проверки (CRC) - описание, принцип действия, показатели эффективности.

Циклические избыточные проверки (CRC) – это метод обнаружения ошибок в передаваемых данных, который используется во многих протоколах связи и хранения данных.
При использовании CRC данные представляются в виде битового потока, к которому добавляется некоторое количество контрольных битов, вычисленных с помощью алгоритма циклического кодирования. Контрольные биты являются функцией данных и рассчитываются на основе полиномиальной формулы, которая выбирается заранее.
При приеме данных полученный битовый поток проверяется на соответствие этой же полиномиальной формуле. Если результат не соответствует ожидаемому значению, то произошла ошибка в передаче данных.
Особенностью метода CRC является его способность обнаруживать не только одиночные ошибки, но и групповые ошибки, при условии что их количество не превышает размер контрольной суммы (т.е. число контрольных бит).
Кроме того, метод CRC отличается достаточно высокой скоростью

обработки данных и простотой реализации.
Недостатком метода CRC является то, что он не позволяет исправлять ошибки, а только обнаруживать их. Кроме того, возможно появление ложных срабатываний в случае использования не подходящей полиномиальной формулы или недостаточного количества контрольных бит.
Принцип действия CRC заключается в добавлении к передаваемому сообщению контрольной суммы, которая рассчитывается на основании содержимого сообщения. Получатель сообщения также вычисляет контрольную сумму и сравнивает ее со значением, полученным от отправителя. Если значения не совпадают, то это означает, что произошла ошибка при передаче данных.
Контрольная сумма вычисляется путем деления передаваемого сообщения на заданный полином и взятия остатка от этого деления. Этот остаток является контрольной суммой, которая добавляется к сообщению и передается вместе с ним. Получатель сообщения выполняет тот же самый расчет контрольной суммы и сравнивает ее со значением, полученным от отправителя.
Показатели эффективности CRC зависят от выбора полинома, используемого для расчета контрольной суммы. Некоторые полиномы имеют более высокую степень защиты от ошибок, чем другие. Важно выбирать полиномы с высокой степенью защиты от ошибок, чтобы обеспечить надежность передачи данных.
Также следует учитывать, что использование CRC приводит к увеличению размера передаваемых данных из-за добавления контрольной суммы. Однако эта дополнительная нагрузка на систему обычно оправдана за счет повышенной надежности передачи данных.

38. Восстановление ошибочно переданных данных: FEC и ARQ. Описание протокола простого ARQ (stop and wait):

подтверждение, восстановление от ошибок, таймаут, предотвращение дублей. Эффективность ## простого ARQ.

При передаче данных по каналу связи могут возникать ошибки, из-за которых данные могут быть искажены или не доходить до получателя вовсе. Для исправления таких ошибок используют два метода: прямое кодирование с помощью передачи дополнительной информации (Forward Error Correction, FEC) и обратная связь с автоматическим повтором запросов (Automatic Repeat reQuest, ARQ).
FEC - это метод, при котором данные дополняются специальным кодом, который позволяет восстановить ошибки без необходимости отправки дополнительных запросов на повтор передачи. Однако данный метод требует заранее известного количества ошибок, которые могут возникнуть в процессе передачи данных.
ARQ - это метод, при котором получатель отправляет подтверждение (ACK) о правильном приеме данных и запрос на повтор передачи (NAK) в случае обнаружения ошибок. Существуют различные реализации протокола ARQ, одной из которых является простой протокол "stop and wait".
Протокол "stop and wait" работает следующим образом:
Отправитель отправляет блок данных.
Получатель принимает блок данных и проверяет его на наличие ошибок. Если блок содержит ошибки, то получатель отправляет запрос на повтор передачи.
Отправитель повторно передает те же данные.
Получатель повторно принимает данные и отправляет подтверждение о правильном приеме.
Отправитель переходит к отправке следующего блока данных.

Протокол ARQ "stop and wait" - это простой протокол обратной связи с автоматическим повтором запросов, который используется для

восстановления ошибочно переданных данных.
Основные механизмы:
Подтверждение (ACK) - получатель отправляет подтверждение о правильном приеме данных. Данное сообщение позволяет отправителю знать, что данные были успешно доставлены.
Восстановление от ошибок - если получатель обнаруживает ошибки в переданных данных, он отправляет запрос на повтор передачи (NAK). Отправитель получает этот запрос и повторно отправляет данные.
Таймаут - отправитель устанавливает таймер, который запускается после отправки блока данных. Если за определенное время не будет получено подтверждение о правильном приеме, то отправитель повторно отправляет данные.
Предотвращение дублей - каждый блок данных содержит уникальный номер, который позволяет получателю проверить, были ли уже получены данные с таким же номером. Если данные уже были получены, то получатель игнорирует их.
В целом, протокол ARQ "stop and wait" является достаточно простым, но эффективным способом обеспечения надежности передачи данных. Однако он имеет свои ограничения и может быть неэффективным при высокой вероятности возникновения ошибок или при больших задержках в передаче данных. Для более эффективной обработки ошибок существуют более сложные протоколы ARQ, такие как "go-back-n" и "selective-repeat".
Эффективность простого ARQ зависит от нескольких факторов:
Вероятность возникновения ошибок - чем меньше вероятность возникновения ошибок в передаваемых данных, тем более эффективен будет протокол ARQ "stop and wait". Если же вероятность ошибок высока, то скорость передачи данных может быть существенно замедлена из-за повторных запросов на повторную передачу.
Время ожидания подтверждений - время ожидания подтверждений может быть значительно большим в условиях низкой пропускной

способности канала связи. Это также может привести к увеличению времени передачи данных и снижению эффективности протокола ARQ.
Размер блока данных - если размер блока данных слишком велик, то повторная передача целого блока может занять слишком много времени. В этом случае протокол ARQ "stop and wait" может быть неэффективным и более сложные протоколы ARQ должны использоваться вместо него.
Задержки в сети - задержки в сети могут привести к тому, что получатель не успевает отправить подтверждение вовремя. В таком случае отправитель повторно передает данные, что может занять слишком много времени и увеличить время передачи данных.

39. Описание протокола ARQ со скользящим окном (Go-Back-N): понятие окна, процедура нормальной передачи, примитив NAK, правила продвижения окна, процедура повторной передачи.

Протокол ARQ (Automatic Repeat Request) со скользящим окном – это протокол обеспечения надежной доставки пакетов между двумя узлами сети. Он использует механизм повторной передачи для обеспечения корректной доставки пакетов. Протокол ARQ со скользящим окном имеет две основные процедуры: нормальную передачу и повторную передачу.
Окно - это диапазон последовательных номеров пакетов, которые может отправить отправитель, не дожидаясь подтверждения от получателя. Размер окна определяется параметрами протокола и обычно является константой.

Процедура нормальной передачи:
Отправитель посылает пакеты от N до N+W-1, где N - номер первого

пакета в окне, а W - размер окна.
Получатель получает пакеты и отправляет подтверждение (ACK) с номером следующего ожидаемого пакета.
Отправитель продвигает окно на количество успешно доставленных пакетов, т.е. если получатель подтвердил доставку пакета с номером N+1, то отправитель продвигает окно на один пакет и начинает передачу пакета с номером N+W.
Примитив NAK используется, когда получатель обнаруживает ошибку в принятом пакете. Получатель посылает NAK с номером наибольшего непринятого пакета. Это заставляет отправителя повторно передать все пакеты, начиная с номера, указанного в примитиве NAK.

Правила продвижения окна:
Если получатель отправляет ACK, равный номеру ожидаемого пакета, то отправитель продвигает окно на один пакет.
Если получатель отправляет ACK, меньший номера ожидаемого пакета, то это может быть дубликат подтверждения и отправитель проигнорирует его.
Если получатель отправляет NAK, то отправитель сбросит окно до номера, указанного в NAK.

Процедура повторной передачи:
Если отправитель не получает подтверждение доставки пакета в течение определенного времени, он считает, что произошел сбой и запускает процедуру повторной передачи. Он повторно передает все пакеты, находящиеся в текущем окне, начиная с первого пакета.
Получатель должен корректно обработать повторно переданные пакеты и отправить подтверждения, как при нормальной передаче.
Если отправитель не получает подтверждения после определенного количества повторных передач, он считает, что соединение недоступно и прерывает передачу.

Таким образом, протокол ARQ со скользящим окном гарантирует доставку пакетов в правильном порядке и без потерь при условии, что

размер окна выбран правильно и задержки в сети не слишком большие.

40. Эффективность Go-Back-N ARQ. Выбор оптимального размера окна. Метод повышения эффективности путем выборочного повтора.

Go-Back-N ARQ (Automatic Repeat Request) - это протокол управления ошибками, который используется для обеспечения надежной доставки данных в сетях передачи пакетов. Он работает по принципу отправки нескольких пакетов подряд без ожидания подтверждения от приемника, а затем ожидания подтверждения и повторной отправки всех пакетов, начиная с последнего успешно доставленного.
Эффективность Go-Back-N ARQ зависит от многих факторов, таких как задержка в сети, размер окна, скорость передачи данных и вероятность возникновения ошибок. Оптимальный размер окна выбирается на основе этих факторов, чтобы минимизировать потери пакетов и избежать перегрузки сети. Существует несколько методов для выбора оптимального размера окна, один из которых - это использование формулы:
W = sqrt(2 *RTT* C / L)
где W - размер окна, RTT - время задержки в сети, C - скорость передачи данных, L - длина пакета данных.
Другой метод повышения эффективности Go-Back-N ARQ - это выборочный повтор. В этом случае, если приемник обнаруживает ошибку в каком-то пакете, он не отправляет подтверждение для всех последующих пакетов до тех пор, пока отправитель не повторит только те пакеты, которые были потеряны или повреждены. Это сокращает количество лишних повторных передач и уменьшает задержку в сети.

Однако, следует отметить, что использование выборочного повтора может привести к увеличению задержки для приемника, так как ему нужно ждать повторной передачи определенных пакетов. Поэтому, выбор между использованием этого метода или стандартного Go-Back-N ARQ должен зависеть от конкретных условий сети и требований к надежности и скорости передачи данных.

41. Функции управления звеном передачи данных: управление потоком (Flow control)

Управление потоком (Flow control) в терминологии передачи данных означает процесс регулирования скорости передачи данных между устройствами, чтобы избежать переполнения буфера приемника. Это особенно важно в случаях, когда скорость передачи данных на приемнике меньше, чем на передатчике. Для управления потоком используются различные методы, включая программное управление потоком и аппаратное управление потоком.
Программное управление потоком предполагает отправку специальных сигнальных символов (контрольных символов), которые информируют устройство-приемник о доступности буфера для получения новых данных.
Аппаратное управление потоком, с другой стороны, предусматривает использование дополнительной аппаратуры, например, линий RTS/CTS, которые сигнализируют об уровне занятости буфера приемника.

Управление потоком (Flow control) - это процесс контроля скорости передачи данных между устройствами. Один из способов программного управления потоком - это использование особого символа Xoff для временной приостановки передачи данных, когда буфер приемника заполнен до определенного уровня. При получении

символа Xoff устройство-отправитель прекращает отправку данных и ожидает появления символа Xon, сигнализирующего о доступности буфера приемника.

Управление потоком передачи данных (англ. Flow Control) — в компьютерных сетях, механизм, который притормаживает передатчик данных при неготовности приёмника.

Различают три основных способа:

```
аппаратный, при котором сигналы «готов/занят» передаются по
отдельным физическим линиям связи. Наиболее известна такая
реализация в интерфейсе RS-232.
программный, при котором программный флажок «готов/занят»
взводится и сбрасывается вставкой в поток данных специальной
уникальной последовательности (XOn/XOff). Применяется в
программных драйверах интерфейса RS-232 как альтернатива
аппаратному контролю потока в случаях неполного соединительного
кабеля.
протокольный, при котором программный флажок «готов/занят»
взводится и сбрасывается специальными соглашениями в рамках
протокола обмена данными. На сегодня является практически
единственным применяемым способом контроля потока. Наиболее
известный пример — реализация контроля потока в протоколе TCP
методом скользящего окна.
```

Импакт:
Управление потоком позволяет улучшить работу сетевого адаптера в режиме полного дуплекса с коммутатором. При работе в полном дуплексе (при этом требуется непосредственное подключение к коммутатору) и при угрозе переполнения буфера данных коммутатора, сетевой адаптер получит специальный кадр паузы. Последующий промежуток времени защищает буфер от переполнения и

предотвращает потерю данных. Эта технология может улучшить общую производительность сети, предотвращает потерю данных и помогает достичь оптимальной производительности в сети.

42. Управление доступом к среде передачи данных: описание сути проблемы и условий, при которых она проявляется. Понятие коллизии. Режим управления звеном Master/Slave и его свойства.

Управление доступом к среде передачи данных - это процесс регулирования доступа различных устройств к общей среде передачи данных. Эта проблема возникает при работе в сетевой среде, где несколько устройств пытаются передавать данные одновременно. Если доступ к среде не будет регулироваться, то возможны коллизии - случаи, когда два или более устройства начинают передачу одновременно и данные перепутываются.

Коллизия - это конфликтный случай, который возникает, когда два или более устройства пытаются передавать данные в одном и том же временном интервале. В результате, данные от этих устройств могут столкнуться на линии связи и образовать ошибки в передаче.
Для решения проблемы управления доступом к среде передачи данных был разработан режим управления звеном Master/Slave. В этом режиме одно из устройств (Master) контролирует доступ к среде передачи данных и определяет, какое устройство должно передавать данные в данный момент. Остальные устройства (Slave) получают команды от Master и передают данные только по указанию Master.

Свойства режима управления звеном Master/Slave:
Организация сети по принципу мастер-раб.

Равномерное использование ресурсов сети.
Уменьшение количества коллизий при передаче данных.
Возможность контроля доступа к среде передачи данных.
Снижение времени задержки на передачу данных.

Конкретным примером может быть использование режима Master/Slave в сети Ethernet. В этом случае, коммутатор работает в режиме Master, а все подключенные к нему устройства - в режиме Slave. Коммутатор контролирует передачу данных между устройствами и управляет потоком данных, чтобы избежать коллизий и обеспечить максимальную производительность сети.
В режиме Master/Slave, мастер имеет полный контроль над рабочими устройствами, которые выполняют только то, что им поручает мастер. Например, в сети Ethernet, мастер коммутатор может определять порты, на которые должны быть направлены данные, контролировать поток данных и устанавливать приоритеты передачи данных.
Этот режим также может использоваться в других типах сетей, таких как CAN (Controller Area Network), где управляющее устройство (мастер) управляет передачей данных между другими устройствами в сети (рабочие).

43. Управление доступом к среде в режиме соперничества. Усовершенствованный метод доступа Ethernet МДПН/ОК (CSMA/CD): детектор несущей, детектор коллизий, регламент доступа станции к среде. Механизм возникновения коллизии в CSMA/CD и факторы, влияющие на её вероятность.

Управление доступом к среде в режиме соперничества используется для управления доступом к общей среде передачи данных, которая может быть использована несколькими станциями одновременно. Одним из методов управления доступом является Ethernet МДПН/ОК (CSMA/CD), который используется в локальных вычислительных сетях. Ethernet МДПН/ОК (CSMA/CD) расшифровывается как "Множественный доступ с прослушиванием несущей/обнаружением столкновения" (Carrier Sense Multiple Access with Collision Detection). Это технология передачи данных по кабельным линиям, которая используется для создания локальных сетей (LAN). Она позволяет нескольким устройствам подключаться к сети одновременно и обмениваться данными. В процессе передачи данных каждое устройство слушает линию на наличие других передач, и в случае обнаружения конфликта прерывает свою передачу.
CSMA/CD - это протокол управления доступом в сетях Ethernet, который основывается на трех компонентах: детекторе несущей, детекторе коллизий и регламенте доступа станции к среде.

Детектор несущей отслеживает наличие сигнала на среде передачи данных. Если сигнал обнаружен, то станция ожидает окончания передачи данных другой станции и только после этого начинает передачу своих данных.
Детектор несущей - это устройство в сетевых технологиях, которое служит для определения наличия несущей частоты на кабельной линии. Он используется для контроля за передачей данных и гарантирует, что только одно устройство находится в активном режиме передачи в каждый момент времени. Если детектор несущей обнаруживает, что линия занята, то устройство откладывает свою передачу до тех пор, пока линия не будет свободна.
Несущая частота - это высокочастотный сигнал, который используется для передачи информации в радиосвязи или кабельных сетях. Он представляет собой сигнал, который не содержит непосредственно передаваемую информацию, а служит только для передачи этой

информации на носительной волне.
Носительная волна - это высокочастотный сигнал, который используется для передачи информации в радиосвязи и кабельных сетях. Он представляет собой электромагнитную волну, которая не несет информацию напрямую, но служит для переноса модулирующего сигнала.
Для передачи данных через кабельную линию, модулирующий сигнал (например, данные) накладывается на несущую частоту, чтобы создать модулированный сигнал, который может быть передан через среду передачи. Приемник на другом конце линии извлекает (демодулирует) оригинальный сигнал из модулированного сигнала, вычитая несущую частоту.
Детектор коллизий следит за тем, что происходит на среде передачи данных, и определяет, если две или более станций пытаются передать данные одновременно. В этом случае происходит коллизия.
Регламент доступа станции к среде определяет, как станция получает доступ к среде передачи данных. Станции получают доступ к среде только тогда, когда она свободна. Если станция не обнаруживает наличия сигнала на среде передачи данных, то она может начать передачу своих данных.
Коллизия в CSMA/CD возникает, когда две или более станции начинают передачу данных одновременно на общей среде передачи данных. Это происходит потому, что станции не могут одновременно обнаружить наличие других передач на среде передачи данных, и поэтому они могут начинать свою передачу данных, что приводит к коллизии.
Механизм возникновения коллизии в CSMA/CD можно объяснить следующим образом:

```
Станция A начинает передачу данных.
Станция B также начинает передачу данных практически
одновременно с станцией A, но не успевает обнаружить наличие
```

```
сигнала на среде передачи данных, который был создан станцией
A.
Оба сигнала доходят до всех станций в сети, и все станции
обнаруживают наличие коллизии.
Станции отбрасывают свои пакеты и ждут случайное время перед
повторной передачей.
```

Факторы, которые влияют на вероятность возникновения коллизии в CSMA/CD, включают в себя:
Высокая загрузка среды передачи данных - если много станций пытаются передавать данные одновременно, то вероятность коллизии возрастает.
Большое количество станций, которые пытаются передавать данные одновременно - чем больше станций, тем выше вероятность коллизии.
Большое расстояние между станциями - если расстояние между станциями большое, то время задержки при передаче данных увеличивается, что приводит к более высокой вероятности коллизии.
Задержки в передаче данных, вызванные проблемами сетевого оборудования - если сетевое оборудование работает неправильно или есть другие проблемы, то это может привести к задержке при передаче данных и увеличить вероятность коллизии.
Использование старых или некачественных кабелей для передачи данных - плохие кабели могут создавать шум на среде передачи данных, что может привести к повышению вероятности коллизии.
Низкий уровень шума на среде передачи данных - если уровень шума на среде передачи данных слишком низкий, то станции могут определить наличие свободной среды передачи данных неправильно, что может привести к коллизии.
Уровень шума в среде передачи данных - это мера того, насколько сильно сигналы, передаваемые по каналу связи, искажаются различными помехами. Он выражается в децибелах и показывает

отношение мощности сигнала к мощности шума.
В целом, чтобы снизить вероятность возникновения коллизии в CSMA/CD, необходимо правильно настроить сеть и использовать высококачественное сетевое оборудование.

44. Семейство протоколов HDLC - место в протокольном стеке, назначение. Виды и формат кадров. Режимы обмена данными. Реализация ARQ в HDLC. Методы управления потоком.

Лекция + Wiki

Место в протокольном стеке, назначение

High-Level Data Link Control (HDLC) — бит-ориентированный протокол канального уровня сетевой модели OSI, разработанный ISO.
Его основным назначением является обеспечение надежной передачи данных между компьютерами через сеть связи. Протокол HDLC может быть
использован для передачи данных между точками одной линии связи (точка-точка) или между группами устройств (многопользовательский режим).

Человечество поэксперементировало в области канальных уровней, придумало один вариант, на базе которого основываются дальнейшее. Большинство канальных протоколов построены на HDLC - способ передачи кадра [порция данных, идущая по звену]

Кадр

Кадры HDLC можно передавать, используя синхронные и асинхронные соединения. В самих соединениях нет механизмов определения начала и конца кадра, для этих целей используется уникальная в пределах протокола битовая последовательность (FD — Frame Delimiter) '01111110' (0x7E в шестнадцатеричном представлении), помещаемая в начало и конец каждого кадра. Уникальность флага гарантируется использованием битстаффинга в синхронных соединениях и байтстаффинга в асинхронных. Битстаффинг — вставка битов, здесь — бита 0 после 5 подряд идущих битов 1. Битстаффинг работает только во время передачи информационного поля (поля данных) кадра. Если передатчик обнаруживает, что передано подряд пять единиц, то он автоматически вставляет дополнительный ноль в последовательность передаваемых битов (даже если после этих пяти единиц и так идёт ноль). Поэтому последовательность 01111110 никогда не появится в поле данных кадра. Аналогичная схема работает в приёмнике и выполняет обратную функцию. Когда после пяти единиц обнаруживается ноль, он автоматически удаляется из поля данных кадра. В байтстаффинге используется escape-последовательность, здесь — '01111101' (0x7D в шестнадцатеричном представлении), то есть байт FD (0x7E) в середине кадра заменяется последовательностью байтов (0x7D, 0x5E), а байт (0x7D) — последовательностью байтов (0x7D, 0x5D).

Структура кадра `[Address(8б)]``[Control(кратно 8б)]``[Inform(8 или 16б)]``[CRC(16б)]` (в википедии, помимо указанных, в начале и в конце изображены FD флаги, о них рассказано позже)
контрол - 1 или 2 байта (от режима зависит) определяет тип кадра.

В стандарте 3 вида кадра iframe[info], sframe[supervisor], uframe[unnumbered]
Контрол - физически представляет собой 1 байт и в зависимости от типа кадра выглядит так:
кадр определяется парой битов -

0 0 - i,
0 1 - s,
1 1 - u

nr - номер кадра, который станция-отправитель желает принять следующим.
ns - номер кадра, который сейчас передается (только в i-frame -> передача данных в режиме ARQ)
Бит P/F (poll/finish) - если p, то мастер-станция отправляет s-frame с признаком 1.
Если это ответ, то признак говорит передачу закончил, пакета не будет.
Если посмотреть iframe, то он весь состоит из номеров (номер ожид, получ, p/f).
У s и u есть доп. подтипы:
s: 2 бита и 4 вариации -
rr - Готов к приёму,
rnr - Не готов к приёму,
reject(NAC) - Неприём,
selective-reject - Выборочный неприём

У кадра типа u их куча (18) и каждый из них отвечает либо за установление режима, либо за инициализацию, либо для объявления об ошибке

Все это -> примитивы ARQ протокола. Кадр из-за чего довольно легкий.
u - 32 бит число
U-кадры предназначены для установления и разрыва логического соединения, а также информирования об ошибках.

В HDLC есть несколько режимов обмена данных - канального уровня: протокол potpo, multipo, datagram mode - станция отправила и всё - так работает Eth. Этот датаграммный режим не требует подтвержения

на канальном уровне, передает тип ненумерованого кадра и кадр будет иметь биты B1=1 B0=1

ff03f:
ff-адрес (бродкаст)
03 - датаграмма
остальное - данные

ARQ

В том режиме, когда идет обмен нумерованными кадрами - режим ARQ. Он двусторонний (в двух направлениях). Кадры противоположного направления выступают в виде ack, из-за чего можно при помощи данных приема подтверждать передачу. Как и в любом arq нужно вначале инициировать счетчики.

Есть несколько видов ARQ - nrm (master-slave)
ABM - async balance mode - двуточечный дюплекс вариант.

Как ABM - (potpo передают данные друг другу)

1. Установка соединения (хэндшейкинг) - передать ненумерованный кадр.
2. Если станция 2 согласна, то отвечает UA - unnumb ack.
3. В обеих станциях счетчики сбрасываются в 0 и начинается передача данных.

Передача данных:

Лекция

передает iframe сразу. У этого кадра ns=0 и nr=0. То есть отправлю и ожидаю кадр=0. Приняв кадр данных станция должна ответить примитивом rr (receiver ready) готов к приему первого кадра. Станция

s2 может отправить свой кадр с ns=0, а nr=2 (так как первый принят и ожидается второй). Третий кадр - nr=3.
в rr всегда ожидаемый (следующий кадр)

В HDLC 2 варианта ack. станция 2, получив кадр, передает rnr (я receive not ready - принял 2 кадр, но ещё не готов принять следующий). Это реализация Start-Stop flow control. Чтобы буфер не был переполнен.

Wiki:

HDLC поддерживает три режима логического соединения, различающиеся ролями взаимодействующих устройств:

```
Режим нормального ответа (Normal Response Mode, NRM) требует инициации передачи в виде явного разрешения на передачу от первичной станции. После использования канала вторичной станцией (ответа на команду первичной), для продолжения передачи она обязана ждать другого разрешения. Для выбора права на передачу первичная станция проводит круговой опрос вторичных. Используется в основном в соединениях точка-многоточка.

Режим асинхронного ответа (Asynchronous Response Mode, ARM) даёт возможность вторичной станции самой инициировать передачу. В основном используется в соединениях типа кольцо и многоточечных с неизменной цепочкой опроса, так как в этих соединениях одна вторичная станция может получить разрешение на передачу от другой вторичной и в ответ начать передачу. То есть разрешение на передачу передаётся по типу маркера (token). За первичной станцией сохраняются обязанности по инициализации линии, определению ошибок передачи и логическому разъединению. Позволяет уменьшить накладные расходы, связанные с началом передачи.
```

Асинхронный сбалансированный режим (Asynchronous Balanced Mode, ABM) используется комбинированными станциями. Передача может быть инициирована с любой стороны, может происходить в полном дуплексе. В режиме ABM оба устройства равноправны и обмениваются кадрами, которые делятся на кадры-команды и кадры-ответы.

45. Протокол (протокольное семейство) PPP - назначение, разновидности, место в протокольном стеке. Формат кадров PPP и способы инкапсуляции. Внутренняя структура протоколов PPP: управляющие протоколы xCP, протоколы аутентификации xAP и сетевые протоколы. Процедура установки соединения PPP.

PPP (англ. Point-to-Point Protocol) — двухточечный протокол канального уровня (Data Link) сетевой модели OSI. Обычно используется для установления прямой связи между двумя узлами сети, причём он может обеспечить аутентификацию соединения, шифрование (с использованием ECP, RFC 1968) и сжатие данных.

Существуют подвиды протокола PPP, такие, как Point-to-Point Protocol over Ethernet (PPPoE), используемый для подключения по Ethernet, и иногда через DSL; и Point-to-Point Protocol over ATM (PPPoA), который используется для подключения по ATM Adaptation Layer 5 (AAL5), который является основной альтернативой PPPoE для DSL.

Формат кадров

Каждый кадр PPP всегда начинается и завершается байтом 0x7E. Затем следует байт адреса и байт управления, которые тоже всегда равны 0xFF и 0x03, соответственно. В связи с вероятностью совпадения байтов внутри блока данных с зарезервированными флагами существует система автоматической корректировки «проблемных» данных с последующим восстановлением.

Флаг 0x7E Адрес 0xFF Управление 0x03 Данные Контрольная сумма Флаг 0x7E
1 1 1 1494 2 1

Поле «Данные» PPP-кадра, в свою очередь, разбиты ещё на два поля: флаг протокола (который определяет тип данных до конца кадра) и сами данные.
Протокол 0xXXXX Данные
1 или 2 0 и более

```
Флаги протокола от 0x0XXX до 0x3XXX идентифицируют протоколы
сетевого уровня. Например, популярному IP-протоколу
соответствует флаг 0x0021, а Novell IPX — 0x002B.
Флаги протокола от 0x4XXX до 0x7XXX идентифицируют протоколы с
низким уровнем трафика.
Флаги протокола от 0x8XXX до 0xBXXX идентифицируют протокол
управления сетью (NCP).
Флаги протокола от 0xCXXX до 0xEXXX идентифицируют управляющие
протоколы. Например, 0xC021 обозначает, что кадр содержит
данные протокола управления соединением LCP.
```

Внутренняя структура протоколов PPP (а где инфа об этом?)

Внутренняя структура протоколов PPP включает три типа протоколов: управляющие протоколы (Control Protocols, xCP), которые отвечают за установление, поддержание и разрыв логического соединения; протоколы аутентификации (Authentication Protocols, xAP), которые используются для проверки подлинности пользователей или устройств.
сетевые протоколы, такие как IP, ICMP и другие, которые передают данные между точками соединения.

Процедура установки соединения PPP

PPP сделан абстрактно - есть 2 пира, которые желают обмениваться данными друг с другом. Давайте способ обмена данными сформулируем в виде набора опций (option value)
Есть набор значений у одного и второго пира.
Есть код управляющего кадра, id (на что пришел запрос)

При установке соединения установлен порядок:
LCP - link control protocol - набор опций линка
Требуется Auth
Согласование Auth
Согласование NCP
Передача данных

Процедура установки соединения PPP включает следующие шаги:
Установка физического соединения между двумя устройствами.
Отправка инициализационных кадров связи с использованием LCP (Link Control Protocol). Этот протокол определяет параметры соединения, такие как скорость передачи данных, размер кадра и другие.
утентификация пользователя или устройства (если необходимо) с помощью протоколов аутентификации xAP.
Установка сетевого соединения с использованием протоколов NCP

(Network Control Protocol).
Обмен данными между устройствами через логическое соединение PPP.

46. Транспортный уровень ЭМВОС: назначение и выполняемые функции. Режимы транспортного сервиса и протоколы, их реализующие (UDP и TCP).

Транспортный протокол отвечает за доставку данных между эндпоинтами, которые соотв. SAP. Узел 1, а SAP может быть множество. В общем - Доставка данных между SAP на оконечных узлах, возможно с гарантией
SAP могут быть на одном узле. Также являются динамически создаваемыми.
Выполняемые функции:
Сегментация данных
Управление темпом передачи
Восстановление данных
Транспортный уровень (англ. Transport layer) — 4-й уровень сетевой модели OSI, предназначен для доставки данных. При этом неважно, какие данные передаются, откуда и куда, то есть, он предоставляет сам механизм передачи. Блоки данных он разделяет на фрагменты, размеры которых зависят от протокола: короткие объединяет в один, а длинные разбивает. Протоколы этого уровня предназначены для взаимодействия типа точка-точка. Пример: TCP, UDP,

Режимы транспортного сервиса и протоколы, их реализующие (UDP и TCP)

Режимы транспортного сервиса определяют тип соединения между узлами сети:
Соединение на основе установления: этот режим используется в протоколе TCP. Передача данных начинается только после установления соединения между узлами сети, и данные передаются в определенном порядке.
Режим без установления соединения: этот режим используется в протоколе UDP. Передача данных начинается сразу после отправки пакета, и данные могут быть доставлены в любом порядке.

TCP

В сети файлы не передаются целиком, а дробятся и передаются в виде относительно небольших сообщений. Далее они передаются другому устройству — получателю, где повторно собираются в файл.

Например, человек хочет скачать картинку. Сервер обрабатывает запрос и высылает в ответ требуемое изображение. Ему, в свою очередь, необходим путь или канал, по которому он будет передавать информацию. Поэтому сервер обращается к сетевому сокету для установки требуемого соединения и отправки картинки. Сервер дробит данные, инкапсулирует их в блоки, которые передаются на уровень TCP получателя при помощи IP-протокола. Далее получатель подтверждает факт передачи.

У протокола TCP есть несколько особенностей:

```
Система нумерации сегментов. TCP отслеживает передаваемые и
принимаемые сегменты, присваивая номера каждому из них. Байтам
данных, которые должны быть переданы, присваивается
определенный номер байта, в то время как сегментам
присваиваются порядковые номера.
Управление потоком. Функция ограничивает скорость, с которой
```

```
отправитель передает данные. Это делается для обеспечения
надежности доставки, в том числе чтобы компьютер не генерировал
пакетов больше, чем может принять другое устройство. Если
говорить простым языком, то получатель постоянно сообщает
отправителю о том, какой объем данных может быть получен.
Контроль ошибок. Функция реализуется для повышения надежности
путем проверки байтов на целостность.
Контроль перегрузки сети. Протокол TCP учитывает уровень
перегрузки в сети, определяемый объемом данных, отправленных
узлом.
```

UDP

Если нам очень важна скорость передачи, а вот потеря пакетов не так критична (как, например, в голосовом или видеотрафике), то лучше использовать UDP, или User Datagram Protocol. В отличие от TCP он обеспечивает передачу данных без получения подтверждения от пользователя. Проще говоря, просто отправляет пакеты и не ждет ничего в ответ. Из-за этого достигается высокая скорость в ущерб надежности.

Чаще всего UDP применяется в чувствительных ко времени службах, где потерять пакеты лучше, чем ждать. Звонки в Skype или Google Meet, стриминг видео, онлайн-трансляции используют этот протокол из-за того, что они чувствительны ко времени и рассчитаны на определенный уровень потерь. Вся голосовая связь через интернет работает по протоколу UDP. Также UDP очень часто используется в онлайн-играх. Аналогичная история с DNS-серверами, поскольку они должны быть быстрыми и эффективными.

Особенности UDP

```
Поддерживает службу без соединения
Отправляет пакеты в большом количестве
В основном используется для потоковых служб и других служб,
таких как DNS и NFS
Отсутствие механизма контроля ошибок
Нет подтверждения после отправки или получения пакета
В IP встроена только адресация между процессами и контрольная
сумма
Отсутствие механизма управления потоком
Более быстрая связь, чем TCP
```

47. Идентификация SAP внутри одного узла на примере TCP/UDP. Номер порта. Явное и автоматическое назначение номера порта на SAP.

В SAP каждый экземпляр протокла транспортного уровня должен иметь идентификацию (как составной ключ в бд) из трех элементов: протокол транспорта, айпи адрес, номер порта.
Что такое номер порта? Это просто число - 65535 max. Первые 1024 номера были признаны привелигированы. их привелигированность определяется их сокет с таким номером может открыть только root user (например, с портом 50).
У нас не получится создать сокет с айпи адресом соседнего узла. Просто открыть сокет, открыв адрес, приближающийся к адресу гугла нельзя, только на локалке. Есть узел А, на нем есть 4 сокета. Комбинация - сокет внутри узла уникален. Нельзя создать перекрывающие друг друга сокеты
S1 -> UDP: 0.0.0.0:80 - такой сокет будет отзываться на любой айпи адрес этого узла

S2 -> TCP: 0.0.0.0:26432 - любой адрес, но не любой порт.
функция bind() - содержит какой назначить айпи и порт. Если программист бинд() не вызывал, но вызвал конект(), то когда конект() поймет что сокет ненумерованный, то сокету автоматом назначится адрес.
Эфемералы - эфемерные порты - существуют на период открытия сокета, потом освобождаются. Заранее сказать адрес сокета непросто, если его сами не присвоили.
UDP: 172.234.39.11:80 открыть не получится, так как он пересечется с S1

48. Протокол UDP: назначение, формат сегмента. Сценарий использование UDP в программе.

UDP (англ. User Datagram Protocol — протокол пользовательских датаграмм) — один из немногих ключевых элементов набора сетевых протоколов для Интернета. С UDP компьютерные приложения могут посылать сообщения (в данном случае называемые датаграммами) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных.

UDP использует простую модель передачи, без явных «рукопожатий» для обеспечения надёжности, упорядочивания или целостности данных. Датаграммы могут прийти не по порядку, дублироваться или вовсе исчезнуть без следа, но гарантируется, что если они придут, то в целостном состоянии. UDP подразумевает, что проверка ошибок и исправление либо не нужны, либо должны исполняться в приложении. Приложения, чувствительные ко времени, но не чувствительные к данным, часто используют UDP, так как предпочтительнее сбросить пакеты, чем ждать задержавшиеся пакеты, что может оказаться

невозможным в системах реального времени. При необходимости исправления ошибок на сетевом уровне интерфейса приложение может задействовать TCP или SCTP, разработанные для этой цели.

Формат сегмента (ну какой сегмент? Это же не TCP! Правильно -> Формат пакета или датаграммы)

UDP не предоставляет никаких гарантий доставки сообщения для вышестоящего протокола и не сохраняет состояния отправленных сообщений. По этой причине UDP иногда называют Unreliable Datagram Protocol (англ. — Ненадёжный протокол датаграмм).

UDP обеспечивает многоканальную передачу (с помощью номеров портов) и проверку целостности заголовка и существенных данных (с помощью контрольных сумм). Надёжная передача в случае необходимости должна реализовываться пользовательским приложением.

Биты 0 - 15 16 - 31
0-31 Порт отправителя (Source port) *Порт получателя (Destination port)*
32-63 Длина датаграммы (Length) Контрольная сумма (Checksum)
64-... Данные (Data)

Заголовок UDP состоит из четырёх полей, каждое по 2 байта (16 бит). Два из них необязательны к использованию в IPv4 (* ячейки в таблице), в то время как в IPv6 необязателен только порт отправителя.

Экспленейшен пакета:
Порт отправителя - В этом поле указывается номер порта отправителя. Предполагается, что это значение задаёт порт, на который при необходимости будет посылаться ответ. В противном же случае значение должно быть равным 0. Если хостом-источником является клиент, то номер порта будет, скорее всего, динамическим. Если

источником является сервер, то его порт будет одним из «хорошо известных».
Порт получателя - Это поле обязательно и содержит порт получателя. Аналогично порту отправителя, если хостом-получателем является клиент, то номер порта динамический, если получатель — сервер, то это будет «хорошо известный» порт.
Длина датаграммы - Поле, задающее длину всей датаграммы (заголовка и данных) в байтах. Минимальная длина равна длине заголовка — 8 байт. Теоретически, максимальный размер поля — 65535 байт для UDP-датаграммы (8 байт на заголовок и 65527 на данные). Фактический предел для длины данных при использовании IPv4 — 65507 (помимо 8 байт на UDP-заголовок требуется ещё 20 на IP-заголовок).
Контрольная сумма - Поле контрольной суммы используется для проверки заголовка и данных на ошибки. Если сумма не сгенерирована передатчиком, то поле заполняется нулями. Поле не является обязательным для IPv4.

Сценарий использования UDP в программе.

сценарий прост ->
сначала функция socket с параметрами AF_INET - хотим сокет в пространстве Ipv4
Когда нужен bind() - когда нужно управлять адресом сокета. Когда программируем сервер, чтобы другие могли указывать конкретный адрес для соединения.
Что делать с номерами портов? Во всех 65000 есть well-known port - локальный порт 80 - протокол HTTP. Это не закон. Это лишь традиция.

Клиенты bind() не используют, но могут.

теперь начинается передача sendto()

UDP - просто ip пакет. с заголовком 20+байтов. к нему добавляется UDP header с размером 8 байт. В этих 8 байт 4 поля есть source port и destination port

общая длина датаграммы 16 бит, контрольная сумма дейтаграммы(используется далеко не всегда, тогда заполняется 0000)

UPD дает создание сокета и просит операционку отправить что-то куда-то, а ОС сообщает ответ, дошло ли или нет.

49. Протокол TCP: назначение, структура передаваемых данных и способ двойной асинхронной буферизации. Сценарий использования TCP в программе. Механизм клонирования socket-а в сервере.

TCP (англ. Transmission Control Protocol — протокол управления передачей) — один из основных протоколов передачи данных интернета. Пакеты в TCP называются сегментами. В стеке протоколов TCP/IP выполняет функции транспортного уровня модели OSI. Механизм TCP предоставляет поток данных с предварительной установкой соединения, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета, гарантируя тем самым (в отличие от UDP) целостность передаваемых данных и уведомление отправителя о результатах передачи.
Реализации TCP обычно встроены в ядра ОС.

Бит 0 — 15 16 — 31
0 Порт источника, Source Port Порт назначения, Destination Port
32 Порядковый номер, Sequence Number (SN)

64 Номер подтверждения, Acknowledgment Number (ACK SN)
Бит 0 — 3 4 — 6 7 — 15 16 — 31
96 Длина заголовка, (Data offset) Зарезервировано Флаги Размер Окна, Window size
Бит 0 — 15 16 — 31
128 Контрольная сумма, Checksum Указатель важности, Urgent Point
160 Опции (необязательное, но используется практически всегда)
160/192+ Данные

Порт источника/назначения

```
Эти 16-битные поля содержат номера портов — числа, которые
определяются по специальному списку.
Порт источника идентифицирует приложение клиента, с которого
отправлены пакеты. Ответные данные передаются клиенту на
основании этого номера.
Порт назначения идентифицирует порт, на который отправлен
пакет.
```

Порядковый номер

```
Sequence number (32 бита) — измеряется в байтах, и каждый
переданный байт полезных данных (payload) увеличивает это
значение на 1.
Если установлен флаг SYN (идёт установление сессии), то поле
содержит изначальный порядковый номер — ISN (Initial Sequence
Number). В целях безопасности это значение генерируется
случайным образом и может быть равно от 0 до 232-1
(4294967295). Первый байт полезных данных в устанавливающейся
сессии будет иметь номер ISN+1.
```

```
В противном случае, если SYN не установлен, первый байт данных, передаваемый в данном пакете, имеет этот порядковый номер.
```

Номер подтверждения
Acknowledgment Number (ACK SN) (32 бита) — если установлен флаг ACK, то это поле содержит порядковый номер октета, который отправитель данного сегмента желает получить. Это означает, что все предыдущие октеты (с номерами от ISN+1 до ACK-1 включительно) были успешно получены.

Длина заголовка (смещение данных)

```
Длина заголовка (Data offset) занимает 4 бита и указывает значение длины заголовка, измеренное в 32-битовых словах. Минимальный размер составляет 20 байт (пять 32-битовых слов), а максимальный — 60 байт (пятнадцать 32-битовых слов). Длина заголовка определяет смещение полезных данных относительно начала сегмента. Например, Data offset равное 11112 говорит о том, что заголовок занимает пятнадцать 32-битных слова (15 строк* 32 бита в каждой строке/8 бит = 60 байт).
```

Зарезервировано

```
Зарезервировано (3 бита) для будущего использования и должно устанавливаться в ноль.
```

Флаги (управляющие биты)

Это поле содержит 9 битовых флагов:
NS (ECN-nonce) — Устойчивый механизм сигнализации насыщения с

помощью ECN-nonce (RFC 3540)
CWR (Congestion Window Reduced) — Поле «Окно перегрузки уменьшено» — флаг установлен отправителем, чтобы указать, что получен пакет с установленным флагом ECE (RFC 3168)
ECE (ECN-Echo) — Поле «Эхо ECN» — указывает, что данный узел способен на ECN (явное уведомление перегрузки) и для указания отправителю о перегрузках в сети (RFC 3168)
URG — поле «Указатель важности» задействовано (англ. Urgent pointer field is significant). Когда узел отправляет сегмент с URG флагом, то узел-получатель принимает его на отдельном канале.
ACK — поле «Номер подтверждения» задействовано (англ. Acknowledgement field is significant)
PSH — (англ. Push function) инструктирует получателя протолкнуть данные, накопившиеся в приёмном буфере, в приложение пользователя. API для установки PSH флага нет. Обычно он устанавливается ядром, когда оно очищает буфер. Дело в том, что когда узел отправляет информацию, TCP сохраняет её в буфере и не передает её сразу другому узлу, ожидая, захочет ли узел-отправитель передать ещё. Такая же схема работает и у узла-получателя. Когда он получает информацию, TCP сохраняет её в буфере, чтобы не тревожить приложение из-за каждого байта полученной информации. Если узел отправляет сегмент с PSH флагом, это значит, что он отправил все, что было нужно.
RST — оборвать соединения, сбросить буфер (очистка буфера) (англ. Reset the connection)
SYN — синхронизация номеров последовательности (англ. Synchronize sequence numbers)
FIN (англ. final, бит) — флаг, будучи установлен, указывает на завершение соединения (англ. FIN bit used for connection termination).

Размер окна

Window Size самостоятельно определяет количество байт данных (payload), после передачи которых отправитель ожидает подтверждения от получателя, что данные получены. Иначе говоря, получатель пакета располагает для приёма данных буфером длиной «размер окна» байт.

Контрольная сумма (Checksum)

Поле контрольной суммы — это 16-битное дополнение к сумме всех 16-битных слов заголовка (включая псевдозаголовок) и данных. Если сегмент, по которому вычисляется контрольная сумма, имеет длину не кратную 16-битам, то длина сегмента увеличивается до кратной 16-ти за счёт добавления к нему справа нулевых битов заполнения. Биты заполнения (0) не передаются в сообщении и служат только для расчёта контрольной суммы. При расчёте контрольной суммы значение самого поля контрольной суммы принимается равным 0.

Указатель важности (Urgent pointer)

16-битовое значение положительного смещения от порядкового номера в данном сегменте. Это поле указывает порядковый номер октета, которым заканчиваются важные (urgent) данные. Поле принимается во внимание только для пакетов с установленным флагом URG. Используется для внеполосных данных.

Опции

Могут применяться в некоторых случаях для расширения протокола. Иногда используются для тестирования. На данный момент в опции

практически всегда включают 2 байта NOP (в данном случае 0x01) и 10 байт, задающих timestamps. Вычислить длину поля опции можно через значение поля смещения.

Двойная асинхронная буферизация

Способ двойной асинхронной буферизации в TCP позволяет оптимизировать процесс передачи данных. Он основан на использовании двух буферов: один буфер для чтения и другой для записи. Когда данные передаются по сети,
они сначала помещаются в буфер записи, а затем извлекаются и отправляются по сети. Это позволяет приложению продолжать работу без ожидания окончания передачи данных.

В отличие от традиционной альтернативы — UDP, который может сразу же начать передачу пакетов, TCP устанавливает соединения, которые должны быть созданы перед передачей данных. TCP-соединение можно разделить на 3 стадии:

```
Установка соединения
Передача данных
Завершение соединения
```

Сценарий использования TCP

у клиента вызывается функция socket(), bind можно сделать для указания желаемого адреса, затем идет connect с адресом destination socket. Функция connect блокирует процесс. Для того чтобы начать соединение адрес сервера и порт(мб там well-known?) нужно знать. Сервер должен быть построен на уровне цикла приема запроса присоединения. socket(), bind(), listen() - listen устанавливает 2 бита для ядра, создающий socket(), но это лишь образцовый паттерн, а не сокет.

accept возвращает новый сокет. Поскольку это все в цикле, на выходе будет создан сокет на каждого клиента.

50. Протокол TCP: формат сегмента, описание полей. Процедуры установки и закрытия соединения TCP. Общее понятие об автомате состояний TCP модуля.

TCP (англ. Transmission Control Protocol — протокол управления передачей) — один из основных протоколов передачи данных интернета. Пакеты в TCP называются сегментами. В стеке протоколов TCP/IP выполняет функции транспортного уровня модели OSI. Механизм TCP предоставляет поток данных с предварительной установкой соединения, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета, гарантируя тем самым (в отличие от UDP) целостность передаваемых данных и уведомление отправителя о результатах передачи.
Реализации TCP обычно встроены в ядра ОС.

Бит 0 — 15 16 — 31
0 Порт источника, Source Port Порт назначения, Destination Port
32 Порядковый номер, Sequence Number (SN)
64 Номер подтверждения, Acknowledgment Number (ACK SN)
Бит 0 — 3 4 — 6 7 — 15 16 — 31
96 Длина заголовка, (Data offset) Зарезервировано Флаги Размер Окна, Window size
Бит 0 — 15 16 — 31
128 Контрольная сумма, Checksum Указатель важности, Urgent Point
160 Опции (необязательное, но используется практически всегда)
160/192+ Данные

Порт источника/назначения

```
Эти 16-битные поля содержат номера портов — числа, которые определяются по специальному списку.
Порт источника идентифицирует приложение клиента, с которого отправлены пакеты. Ответные данные передаются клиенту на основании этого номера.
Порт назначения идентифицирует порт, на который отправлен пакет.
```

Порядковый номер

```
Sequence number (32 бита) — измеряется в байтах, и каждый переданный байт полезных данных (payload) увеличивает это значение на 1.
Если установлен флаг SYN (идёт установление сессии), то поле содержит изначальный порядковый номер — ISN (Initial Sequence Number). В целях безопасности это значение генерируется случайным образом и может быть равно от 0 до 232-1 (4294967295). Первый байт полезных данных в устанавливающейся сессии будет иметь номер ISN+1.
В противном случае, если SYN не установлен, первый байт данных, передаваемый в данном пакете, имеет этот порядковый номер.
```

Номер подтверждения
Acknowledgment Number (ACK SN) (32 бита) — если установлен флаг ACK, то это поле содержит порядковый номер октета, который отправитель данного сегмента желает получить. Это означает, что все предыдущие октеты (с номерами от ISN+1 до ACK-1 включительно) были успешно получены.

Длина заголовка (смещение данных)

```
Длина заголовка (Data offset) занимает 4 бита и указывает значение длины заголовка, измеренное в 32-битовых словах. Минимальный размер составляет 20 байт (пять 32-битовых слов), а максимальный — 60 байт (пятнадцать 32-битовых слов). Длина заголовка определяет смещение полезных данных относительно начала сегмента. Например, Data offset равное 11112 говорит о том, что заголовок занимает пятнадцать 32-битных слова (15 строк* 32 бита в каждой строке/8 бит = 60 байт).
```

Зарезервировано

```
Зарезервировано (3 бита) для будущего использования и должно устанавливаться в ноль.
```

Флаги (управляющие биты)

Это поле содержит 9 битовых флагов:
NS (ECN-nonce) — Устойчивый механизм сигнализации насыщения с помощью ECN-nonce (RFC 3540)
CWR (Congestion Window Reduced) — Поле «Окно перегрузки уменьшено» — флаг установлен отправителем, чтобы указать, что получен пакет с установленным флагом ECE (RFC 3168)
ECE (ECN-Echo) — Поле «Эхо ECN» — указывает, что данный узел способен на ECN (явное уведомление перегрузки) и для указания отправителю о перегрузках в сети (RFC 3168)
URG — поле «Указатель важности» задействовано (англ. Urgent pointer field is significant). Когда узел отправляет сегмент с URG флагом, то узел-получатель принимает его на отдельном канале.
ACK — поле «Номер подтверждения» задействовано (англ.

Acknowledgement field is significant)
PSH — (англ. Push function) инструктирует получателя протолкнуть данные, накопившиеся в приёмном буфере, в приложение пользователя. API для установки PSH флага нет. Обычно он устанавливается ядром, когда оно очищает буфер. Дело в том, что когда узел отправляет информацию, TCP сохраняет её в буфере и не передает её сразу другому узлу, ожидая, захочет ли узел-отправитель передать ещё. Такая же схема работает и у узла-получателя. Когда он получает информацию, TCP сохраняет её в буфере, чтобы не тревожить приложение из-за каждого байта полученной информации. Если узел отправляет сегмент с PSH флагом, это значит, что он отправил все, что было нужно.
RST — оборвать соединения, сбросить буфер (очистка буфера) (англ. Reset the connection)
SYN — синхронизация номеров последовательности (англ. Synchronize sequence numbers)
FIN (англ. final, бит) — флаг, будучи установлен, указывает на завершение соединения (англ. FIN bit used for connection termination).

Размер окна

```
Window Size самостоятельно определяет количество байт данных (payload), после передачи которых отправитель ожидает подтверждения от получателя, что данные получены. Иначе говоря, получатель пакета располагает для приёма данных буфером длиной «размер окна» байт.
```

Контрольная сумма (Checksum)

```
Поле контрольной суммы — это 16-битное дополнение к сумме всех 16-битных слов заголовка (включая псевдозаголовок) и данных.
```

Если сегмент, по которому вычисляется контрольная сумма, имеет длину не кратную 16-битам, то длина сегмента увеличивается до кратной 16-ти за счёт добавления к нему справа нулевых битов заполнения. Биты заполнения (0) не передаются в сообщении и служат только для расчёта контрольной суммы. При расчёте контрольной суммы значение самого поля контрольной суммы принимается равным 0.

Указатель важности (Urgent pointer)

16-битовое значение положительного смещения от порядкового номера в данном сегменте. Это поле указывает порядковый номер октета, которым заканчиваются важные (urgent) данные. Поле принимается во внимание только для пакетов с установленным флагом URG. Используется для внеполосных данных.

Опции

Могут применяться в некоторых случаях для расширения протокола. Иногда используются для тестирования. На данный момент в опции практически всегда включают 2 байта NOP (в данном случае 0x01) и 10 байт, задающих timestamps. Вычислить длину поля опции можно через значение поля смещения.

Установка TCP соединения

Three way handshaking - трехтороннее рукопожатие - клиент в тот момент времени когда вызван connect() формирует TCP сегмент с флажком SYN - в этом сегменте данных может не быть, в качестве sequence (номер) выставляется isn - initial sequence number. Он рандомный, нужен для защиты. Из-за чего соединение соединяется

через случайный номер. Хакеру тяжело перехватить этот пакет.
isn - номер байта который бы посылался. Сервер, получив sin, берет accept, и отправляет sin и ack. клиент подтвержает isnc, отправив isnc+1, а также посылает isns (свой рандомный номер). Сервер его должен принять.
Почему 3 хендшейкинга? инициализируется 2 счетчика isnc, isns. sin - приглашение.

Состояния

Это state machine. Есть несколько логический состояний переключючаются при совершении определнных событий.
Первое состояние - closed - закрыт.
Если вызвана listen(), то у сокета будет состояние listen.
Состояние SYNSENT - с момента первой отправки до момента получения ответа из этого состояния выйдет, если получено syn + ack состояние ESTABLISHED. Для состояния пассивного открытия,
эти 2 блока (нижних) - это закрытие сокета. Базовый вариант закрытия.
Нормальное закрытие - одна сторона посывлает функцию fin, при этом вторая сторона может отправлять свои данные, после конца отправки и подтверждения ack сокет является закрытым, данное приложение олжно быть закрытым.
SYN - первые 2 пакета хендшейкинга
RST - reset 0 соединение нормально закрывается без доставки данных
PSH push - доставь данные как можно скорее
ECE, CWR, NS - выставляют маршрутизаторы. Единственная тема, которую не озвучиил - window size.
Каждый раз отправитель указывать кол-во свободных в буфере байт.
Если сделать receive number буфера = 0, тогда window = 0, и пир не может отправлить ни одного байта.