ФГБОУ «КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕН»

МИППС

КОНТРОЛЬНАЯ РАБОТА № 1 вариант Компьютерные сети. Сетевые протоколы и уровни

по_____Архитектура вычислительных систем_____
наименование дисциплины по учебному плану

Студент __3__ курса, шифр ___17-ЗКБс-003__, специальность _____09 03 04_____

Фамилия _____Бабич_____

Имя_____Михаил_____ Отчество _____Михайлович_____

Дата поступления работы _____

Оценка_____ Рецензент_____Бабенко Г. В._____

«____»_____2020 г. Подпись_____

Содержание

1 Определение компьютерной сети

Сеть — это совокупность устройств и систем, которые подключены друг к другу (логически или физически) и общающихся между собой. Сюда можно отнести сервера, компьютеры, телефоны, маршрутизаторы и так далее. Размер этой сети может достигать размера Интернета, а может состоять всего из двух устройств, соединенных между собой кабелем. Чтобы не было каши, разделим компоненты сети на группы:

1) Оконечные узлы: Устройства, которые передают и/или принимают какие-либо данные. Это могут быть компьютеры, телефоны, сервера, какие-то терминалы или тонкие клиенты, телевизоры.

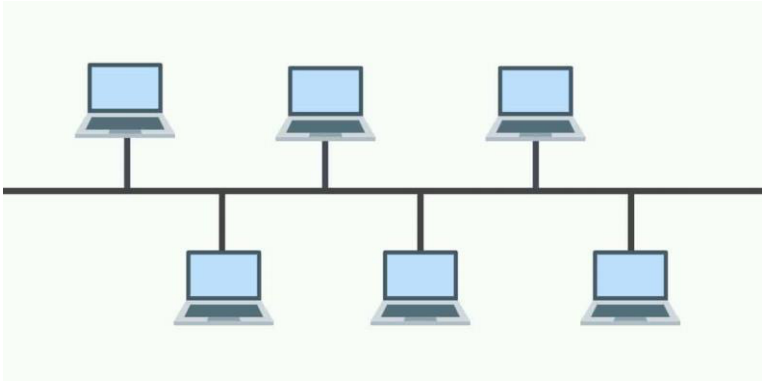
2) Промежуточные устройства: Это устройства, которые соединяют оконечные узлы между собой. Сюда можно отнести коммутаторы, концентраторы, модемы, маршрутизаторы, точки доступа Wi-Fi.

3) Сетевые среды: это те среды, в которых происходит непосредственная передача данных. Сюда относятся кабели, сетевые карточки, различного рода коннекторы, воздушная среда передачи. Если это медный кабель, то передача данных осуществляется при помощи электрических сигналов. У оптоволоконных кабелей, при помощи световых импульсов. Ну и у беспроводных устройств, при помощи радиоволн.

Теперь поговорим о такой важной вещи, как топология. Она делится на 2 большие категории: **физическая** и **логическая**. Очень важно понимать их разницу. Итак, **физическая** топология — это как наша сеть выглядит. Где находятся узлы, какие сетевые промежуточные устройства используются и где они стоят, какие сетевые кабели используются, как они протянуты и в какой порт воткнуты. **Логическая** топология — это каким путем будут идти пакеты в нашей физической топологии. То есть физическая — это как мы расположили устройства, а логическая — это через какие устройства будут проходить пакеты.

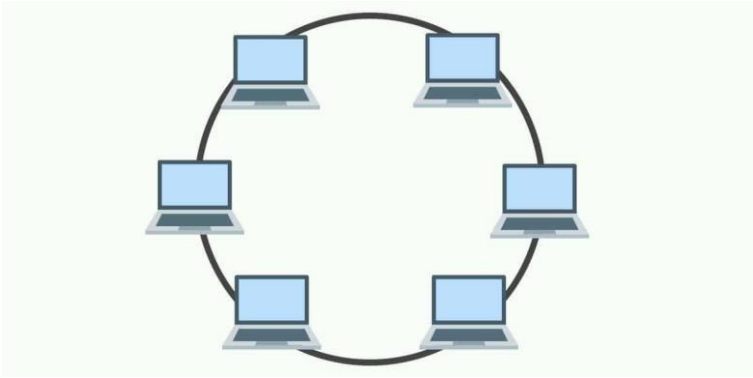
2. Виды сетевых топологий

1) Топология с общей шиной (англ. Bus Topology)



Одна из первых физических топологий. Суть состояла в том, что к одному длинному кабелю подсоединяли все устройства и организовывали локальную сеть. На концах кабеля требовались терминаторы. Как правило — это было сопротивление на 50 Ом, которое использовалось для того, чтобы сигнал не отражался в кабеле. Преимущество ее было только в простоте установки. С точки зрения работоспособности была крайне неустойчивой. Если где-то в кабеле происходил разрыв, то вся сеть оставалась парализованной, до замены кабеля.

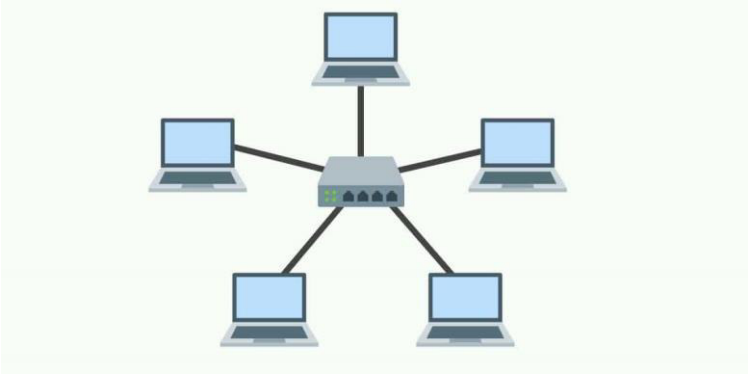
2) Кольцевая топология (англ. Ring Topology)



В данной топологии каждое устройство подключается к 2-ум соседним. Создавая, таким образом, кольцо. Здесь логика такова, что с одного конца компьютер только принимает, а с другого только отправляет. То есть, получается передача по кольцу и следующий компьютер играет роль ретранслятора сигнала. За счет этого нужда в терминаторах отпала. Соответственно, если где-то кабель повреждался, кольцо размыкалось и сеть

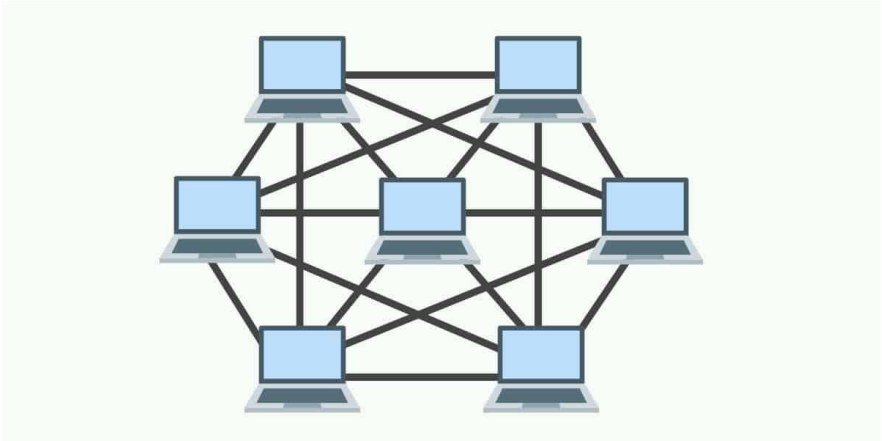
становилась не работоспособной. Для повышения отказоустойчивости, применяют двойное кольцо, то есть в каждое устройство приходит два кабеля, а не один. Соответственно, при отказе одного кабеля, остается работать резервный.

3) Топология звезда (англ. Star Topology)



Все устройства подключаются к центральному узлу, который уже является ретранслятором. В наше время данная модель используется в локальных сетях, когда к одному коммутатору подключаются несколько устройств, и он является посредником в передаче. Здесь отказоустойчивость значительно выше, чем в предыдущих двух. При обрыве, какого-либо кабеля, выпадает из сети только одно устройство. Все остальные продолжают спокойно работать. Однако если откажет центральное звено, сеть станет неработоспособной.

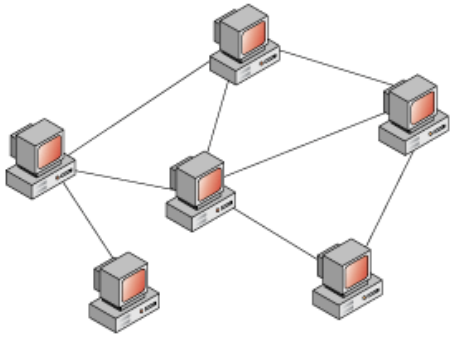
4) Полносвязная топология (англ. Full-Mesh Topology)



Все устройства связаны напрямую друг с другом. То есть с каждого на каждый. Данная модель является, пожалуй, самой отказоустойчивой, так как не зависит от других. Но строить сети на такой модели сложно и дорого. Так

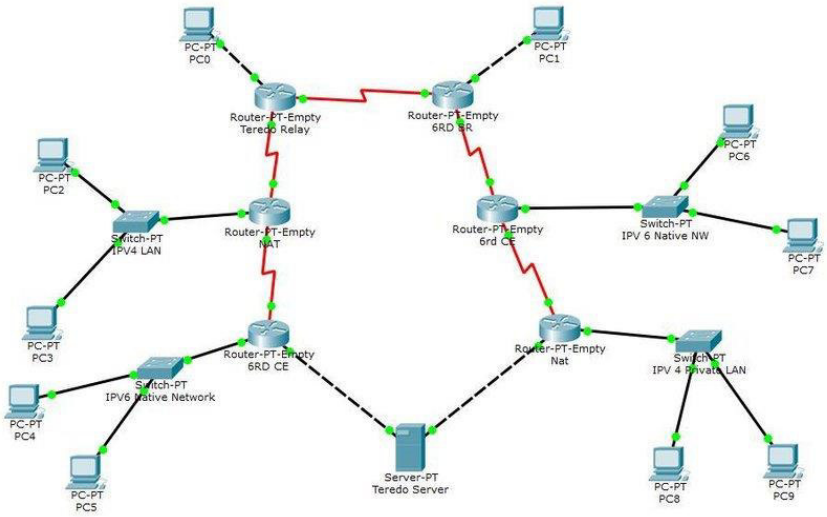
как в сети, в которой минимум 1000 компьютеров, придется подключать 1000 кабелей на каждый компьютер.

5)Неполносвязная топология (англ. Partial-Mesh Topology)



Как правило, вариантов ее несколько. Она похожа по строению на полносвязную топологию. Однако соединение построено не с каждого на каждый, а через дополнительные узлы. То есть узел А, связан напрямую только с узлом В, а узел В связан и с узлом А, и с узлом С. Так вот, чтобы узлу А отправить сообщение узлу С, ему надо отправить сначала узлу В, а узел В в свою очередь отправит это сообщение узлу С. В принципе по этой топологии работают маршрутизаторы. Приведу пример из домашней сети. Когда вы из дома выходите в Интернет, у вас нет прямого кабеля до всех узлов, и вы отправляете данные своему провайдеру, а он уже знает куда эти данные нужно отправить.

6) Смешанная топология (англ. Hybrid Topology)



Самая популярная топология, которая объединила все топологии выше в себя. Представляет собой древовидную структуру, которая объединяет все топологии. Одна из самых отказоустойчивых топологий, так как если у двух площадок произойдет обрыв, то парализована будет связь только между ними, а все остальные объединенные площадки будут работать безотказно. На сегодняшний день, данная топология используется во всех средних и крупных компаниях.

3 Модель OSI и сетевые уровни

И последнее, что осталось разобрать — это сетевые модели. На этапе зарождения компьютеров, у сетей не было единых стандартов. Каждый вендор использовал свои проприетарные решения, которые не работали с технологиями других вендоров. Конечно, оставлять так было нельзя и нужно было придумывать общее решение. Эту задачу взвалила на себя международная организация по стандартизации (ISO — International Organization for Standartization).

Они изучали многие, применяемые на то время, модели и в результате придумали **модель OSI**, релиз которой состоялся в 1984 году. Проблема ее была только в том, что ее разрабатывали около 7 лет. Пока специалисты спорили, как ее лучше сделать, другие модели модернизировались и набирали обороты. В настоящее время модель OSI не используют. Она применяется только в качестве обучения сетям. Е

Состоит она из 7 уровней и каждый уровень выполняет определенную ему роль и задачи. Разберем, что делает каждый уровень снизу вверх:

1) Физический уровень (Physical Layer): определяет метод передачи данных, какая среда используется (передача электрических сигналов, световых импульсов или радиоэфир), уровень напряжения, метод кодирования двоичных сигналов.

2) Канальный уровень (Data Link Layer): он берет на себя задачу адресации в пределах локальной сети, обнаруживает ошибки, проверяет целостность данных. Если слышали про MAC-адреса и протокол «Ethernet», то они располагаются на этом уровне.

3) Сетевой уровень (Network Layer): этот уровень берет на себя объединения участков сети и выбор оптимального пути (т.е. маршрутизация). Каждое сетевое устройство должно иметь уникальный сетевой адрес в сети. Думаю, многие слышали про протоколы IPv4 и IPv6. Эти протоколы работают на данном уровне.

4) Транспортный уровень (Transport Layer): Этот уровень берет на себя функцию транспорта. К примеру, когда вы скачиваете файл с Интернета, файл в виде сегментов отправляется на Ваш компьютер. Также здесь вводятся понятия портов, которые нужны для указания назначения к конкретной

службе. На этом уровне работают протоколы TCP (с установлением соединения) и UDP (без установления соединения).
5) Сеансовый уровень (Session Layer): Роль этого уровня в установлении, управлении и разрыве соединения между двумя хостами. К примеру, когда открываете страницу на веб-сервере, то Вы не единственный посетитель на нем. И вот для того, чтобы поддерживать сеансы со всеми пользователями, нужен сеансовый уровень.

6) Уровень представления (Presentation Layer): Он структурирует информацию в читабельный вид для прикладного уровня. Например, многие компьютеры используют таблицу кодировки ASCII для вывода текстовой информации или формат jpeg для вывода графического изображения.

7) Прикладной уровень (Application Layer): Наверное, это самый понятный для всех уровень. Как раз на этом уроне работают привычные для нас приложения — e-mail, браузеры по протоколу HTTP, FTP и остальное.

Самое главное помнить, что нельзя перескакивать с уровня на уровень (Например, с прикладного на канальный, или с физического на транспортный). Весь путь должен проходить строго с верхнего на нижний и с нижнего на верхний. Такие процессы получили название **инкапсуляция** (с верхнего на нижний) и **деинкапсуляция** (с нижнего на верхний). Также стоит упомянуть, что на каждом уровне передаваемая информация называется по-разному.

4 Сетевые протоколы

На прикладном, представления и сеансовым уровнях, передаваемая информация обозначается как PDU (Protocol Data Units). На русском еще называют блоки данных).

Информацию транспортного уровня называют сегментами. Хотя понятие сегменты, применимо только для протокола TCP. Для протокола UDP используется понятие — датаграмма. Но, как правило, на это различие закрывают глаза.

На сетевом уровне называют IP пакеты или просто пакеты.

И на канальном уровне — кадры. С одной стороны, это все терминология и она не играет важной роли в том, как вы будете называть передаваемые данные, но для экзамена эти понятия лучше знать.

Итак, протоколы прикладного уровня обеспечивают взаимодействие между человеком и сетью. Этих протоколов огромное количество, и выполняют они совершенно различные роли.

I) Протокол HTTP (англ. HyperText Transport Protocol). Протокол передачи данных, используемый обычно для получения информации с веб-сайтов. С каждым годом этот протокол становится все популярнее, и возможностей для его применения становится все больше. Использует он «клиент-серверную» модель. То есть существуют клиенты, которые формируют и отправляют запрос. И серверы, которые слушают запросы и, соответственно, на них отвечают.

В качестве клиентов выступают известные многим веб-браузеры: Internet Explorer, Mozilla Firefox, Google Chrome и т.д. А в качестве серверного ПО используют:Apache, IIS, nginx и т.д.

II) DNS (Domain Name System). Система доменных имен. Если говорить в целом, то она хранит информацию о доменах. Например, какому IP адресу соответствует определенное имя. Приведу пример: когда вы открываете свой любимый сайт, то обращаетесь к нему по имени. Но в поля Source Address и Destination Address, которые работают на сетевом уровне нельзя вставить имя. Там обязательно должен присутствовать именно IP адрес. Вот DNS как раз этим и занимается. Она сообщает, какой IP адрес у запрошенного имени. Вы, к примеру, обращаетесь на google.ru. Ваш компьютер понятия не имеет, кто и что это. Он спрашивает у DNS-сервера: Кто такой google.ru? И сервер отвечает, что google.ru — это 74.125.232.239 (это один из его адресов). И уже после этого, компьютер отправляет запрос на 74.125.232.239. Для пользователя все останется по-прежнему, и в адресной строке он также будет видеть google.ru.

III) DHCP (Dynamic Host Configuration Protocol). Протокол динамической настройки узла. Он позволяет узлам динамически получать IP адреса и другие параметры для корректной работы в сети (основной шлюз, маску подсети, адреса DNS-серверов). От себя скажу, что этот протокол спасает жизнь многим сисадминам по всему миру. Согласитесь, что ходить и вручную прописывать IP параметры каждому узлу, не самое приятное занятие.

При помощи DHCP можно обеспечить полный контроль над IP адресами: создавать отдельные пулы для каждой подсети, выдавать адреса в аренду, резервировать адреса и многое другое.

Работа его очень тяжела для нынешнего понимания. Слишком много пакетов, данных и кадров должно передаться, прежде чем запрошенный адрес будет присвоен компьютеру.

IV) POP3 (англ. Post Office Protocol Version 3). Протокол почтового отделения версии 3. Протокол, который используют клиенты для получения почтовых писем с сервера. Версии 1-ая и 2-ая устарели и в нынешнее время не используются. Работает он по принципу «загрузи и удали». Что это значит? Это значит, что клиент заходит на сервер и смотрит, есть ли для него письмо. И если оно присутствует, он загружает его к себе и ставит отметку об удалении на сервере. Хорошо это или плохо, вопрос спорный. протокол POP3 на практике после того, как узнаем про протокол SMTP.

Стоит упомянуть про аналог POP3. Это протокол **IMAP (англ. Internet Message Access Protocol).** Протокол доступа к электронной почте. Он более умный и посложнее, чем POP3. Но главное их различие в том, что клиент, заходя на сервер, не удаляет почту, а копирует ее. Таким образом, у клиента отображается копия почтового ящика, который хранится на почтовом сервере. И если клиент у себя удаляет какое-либо письмо, то оно удаляется только у него. На сервере оригинал остается целым. Слушает он 143 порт. Рассмотреть IMAP подробно в CPT не получится, так как полноценно он там не реализован.

V) SMTP (англ. Simple Mail Transfer Protocol). Простой протокол передачи почты. Используется он, как вы поняли, для передачи почты на

почтовый сервер. Вот почему мы изучаем POP3 и SMTP параллельно. Использует он 25 порт. Это тоже важно помнить.

Также важно помнить, что все почтовые протоколы работают по TCP-соединению. То есть с установлением соединения. Здесь важно получить каждый пакет в целости и сохранности.

VI) Telnet (от англ. terminal network). Если переводить дословно, то это сетевой терминал. Основы этого протокола были заложены давным давно, и до сих пор он не теряет своей актуальности. Применяется он для отображения текстового интерфейса, а также для управления ОС. Очень полезный протокол, и каждый сетевой инженер обязан уметь работать с ним. Объясню почему. Каждое сетевое устройство, интерфейс которого представляет собой командную строку, настраивается либо при помощи специального консольного кабеля, либо через виртуальные терминалы, в который и входит протокол Telnet. И, если консольный кабель требует нахождения специалиста рядом с настраиваемым оборудованием, то настройка при помощи виртуальных терминалов, а в данном случае Telnet, не ограничивает специалиста в расстоянии. Можно находиться в другой комнате, здании, городе и все равно иметь возможность доступа к оборудованию. Использует он 23 порт. А самые популярные дистрибутивы, которые работают с этим протоколом — это Putty, Kitty, XShell и т.д.

VII) SSH (англ. Secure Shell). В переводе с английского — безопасная оболочка. Как и Telnet позволяет управлять ОС. Отличие его в том, что он шифрует весь трафик и передаваемые пароли. Шифруется при помощи алгоритма Диффи-Хеллмана. Кому интересно почитайте. Практически все современные ОС системы умеют работать с этим протоколом. Если у вас стоит выбор, какой протокол применять, то используйте SSH. Сначала немного помучаетесь в настройке, и многое будет непонятно, но со временем в голове уляжется. Главное запомните сейчас, что самое главное отличие SSH от Telnet — это то, что SSH шифрует трафик, а Telnet нет.

VIII) FTP (англ. File Transfer Protocol). Протокол передачи файлов. Думаю из названия протокола ясно, что он передает файлы. Очень древний протокол, вышедший в начале 70-х годов. Появился он еще до HTTP и стека TCP/IP. Как работал раньше, так и сейчас работает по «клиент-сервер» модели. То есть, присутствует инициатор соединения и тот, кто его слушает. Есть несколько модификаций, которые поддерживают шифрование, туннелирование и так далее. Раньше с этим протоколом работали разные консольные утилиты, у которых не было графики и работали они, при помощи ввода определенных команд. В нынешнее время присутствуют и графические программы. Самой популярной и простой является Filezilla. В CPT реализован только консольный метод.

IX) TFTP (англ. Trivial File Transfer Protocol). Простой протокол передачи файлов. Придумали его в 80-х годах. Хоть FTP был достаточно популярным, не все его функции были нужны для решения простых задач. И был придуман его простой аналог. Он работает по UDP, то есть не требует установления соединения. Также он не требует аутентификации и авторизации. Достаточно знать его IP-адрес и самому его иметь. Это конечно не безопасно, так как адрес можно подделать. Но когда нужен простой протокол и не требуется авторизация, выбор падает на него. Очень плотно с ним работает цисковское оборудование, для копирования образа или скачивания на flash-память.

Самый популярный на сегодняшний момент протокол, используемый в локальных сетях — это **Ethernet**. IEEE описала его стандартом 802.3. Так что, все версии, которые начинаются с 802.3, относятся именно к нему. Например, 802.3z — это GigabitEthernet через волоконно-оптический кабель; 1 Гбит/с, а 802.3af — это электропитание через Ethernet (PoE — Power over Ethernet).

Ethernet-кадр					
8 байт	6 байт	6 байт	2 байта	46-1500 байт	4 байта
Преамбула	MAC-адрес полу-чателя	MAC-адрес источника	Тип(длина)	SNAP/LLC и данные	FCS(Frame Check Sequence)-контроль суммы

1) **Преамбула.** Поле, используемое для указания начала кадра. То есть, чтобы приемник смог понять, где начало нового кадра. Раньше, когда использовалась топология с общей шиной и были коллизии, преамбула помогала предотвращать коллизии.

2) **MAC-адрес получателя.** Поле, куда записывается адрес получателя.

3) **MAC-адрес отправителя.** Соответственно сюда записывается адрес отправителя.

4) **Тип (длина).** В этом поле указывается вышестоящий протокол. Для IPv4 — это 0x0800, для ARP — 0x0806, а для IPv6 — 0x86DD. В некоторых случаях сюда может записываться длина поля данных кадра (следующее поле в заголовке).

5) **Поле SNAP/LLC + данные.** В этом поле содержатся данные, полученные с высших уровней (или полезные данные).

6) **FCS (от англ. Frame Check Sequence — контрольная сумма кадра).** Поле в котором подсчитана чек-сумма. По ней получатель понимает, побился кадр или нет.

Переходим к сетевому уровню, и тут нас встречает нашумевший протокол IP. Раз мы говорим о сетевом уровне, то значит протокол, работающий на этом уровне, должен каким-то образом уметь передавать данные из одной канальной среды в другую. Но для начала посмотрим, что это за протокол и из чего он состоит.

IP (от англ. Internet Protocol). Протокол семейства TCP/IP, который был разработан в 80-х годах. Используется для объединения отдельных компьютерных сетей между собой. Также важной его особенностью является адресация, которую называют

IP-адрес. На текущий момент существуют 2 версии протокола: IPv4 и IPv6. Пару слов о них:

1) IPv4. Использует 32-битные адреса, которые записываются в формате четырёх десятичных чисел (от 0 до 255), разделённых точками. Например,

адрес 192.168.0.4. Каждое число, разделенное точками, называют октетом. Это самая популярная версия на сегодняшний день.

2) IPv6. Использует 128-битные адреса, которые записываются в формате восьми четырехзначных шестнадцатеричных чисел (от 0 до F). Например, адрес 2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d. Каждое число, разделенное точками, называют хекстетом. На заре всеобщей компьютеризации появилась проблема. Стали заканчиваться IP-адреса и нужен был новый протокол, который смог бы обеспечить больше адресов. Так и появился в 1996 году протокол IPv6. Но благодаря технологии NAT, которая будет рассмотрена позже, была частична решена проблема нехватки адресов, и, в связи с этим, внедрение IPv6 затянулось по сегодняшний день.

Итак, протокол IP работает с блоком информации, который принято называть IP-пакет. Рассмотрим его структуру.

1) Версия. Протокол IPv4 или IPv6.

2) IHL (от англ. Internet Header Length — размер заголовка). Так как многие из показанных на картинке полей не фиксированы, то это поле считает размер заголовка.

3) Тип обслуживания. Обслуживает размер очередей QoS (Quality of Service — качество обслуживания). Делает он это при помощи байта, который указывает на определенный набор критериев (требование ко времени задержки, пропускной способности, надежности и т.д.)

4) Длина пакета. Размер пакета. Если **IHL** отвечает только за размер полей в заголовке (заголовком являются все поля на картинке, кроме поля данных), то длина пакета отвечает за весь пакет в целом, включая пользовательские данные.

5) Время жизни (TTL- Time To Live). Поле, используемое для предотвращения циклического пути пакета. При прохождении через маршрутизатор, значение уменьшается на единицу, и когда достигает нуля, пакет отбрасывается.

6) Протокол. Для какого вышестоящего протокола предназначается данный пакет (TCP, UDP).

7) Контрольная сумма заголовка. Здесь считается целостность полей заголовка. Не данных! Данные проверяются соответствующим полем на канальном уровне.

8) Опции. Это поле используется для расширения стандартного заголовка IP. Используется в привычных сетях редко. Сюда записываются данные для какого-нибудь специфического оборудования, которое читает это поле. Например, система управления дверными замками (где идет общение с контроллером), технология умного дома, интернет-вещи и так далее. Привычные сетевые устройства, как роутеры и коммутаторы, будут игнорировать это поле.

9) Смещение. Указывает, какому месту принадлежит фрагмент в оригинальном IP. Это значение всегда кратно восьми байтам.

10) Данные. Здесь как раз содержатся данные, полученные с вышестоящих уровней. В Ethernet-кадре тоже есть поле данных. И в его поле данных будет включен данный IP-пакет. Важно помнить, что максимальный размер Ethernet-кадра равен 1500 байт, а вот размер IP пакета может быть 20 Кбайт. Соответственно весь пакет не вместится в поле данных Ethernet-кадра. Поэтому пакет делят и отправляют частями. И вот для этого используются 3 поля ниже.

11) Идентификатор. Это 4-х байтовое число, которое показывает, что все части разделенного пакета одно единое целое.

12) Флаги. Указывает, что это не единый, а фрагментированный пакет.

13) Смещение фрагмента. Сдвиг относительно первого фрагмента. То есть это нумерация, которая поможет собрать IP-пакет воедино.

14) IP-адрес отправителя и IP-адрес получателя. Соответственно эти 2 поля указывают от кого и для кого пакет.

Остался последний уровень из стека TCP/IP. Это **транспортный уровень**. Пару слов о нем. Он предназначен для доставки данных определенному приложению, которое он определяет по номеру порта. В зависимости от протокола, он выполняет разные задачи. Например, фрагментация файлов, контроль доставки, мультиплексирование потоками данных и управление ими. 2 самых известных протокола транспортного уровня — это UDP и TCP. Поговорим о каждом из них подробнее, и начну с UDP, в силу его простоты. Ну и по традиции показываю, из чего он состоит.

<table>
<tr><td>Порт источника (16 бит)</td><td>Порт назначения (16 бит)</td></tr>
<tr><td>Длина UDP (16 бит)</td><td>Контрольная сумма UDP</td></tr>
<tr><td colspan="2">Данные</td></tr>
</table>

1) Порт источника. Порт, используемый клиентом или сервером для идентификации службы. На этот порт, при необходимости, будет посылаться ответ.

2) Порт назначения. Здесь указывается порт, который будет являться адресатом. Например, если клиент запрашивает страницу сайта, то порт назначения, по умолчанию, будет 80-ый (протокол HTTP).

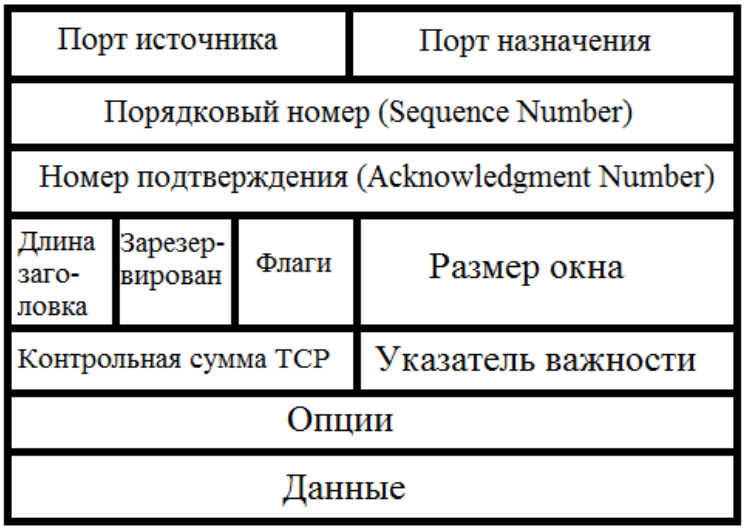
3) Длина UDP. Длина заголовка UDP. Размер варьируется от 8 до 65535 байт.

4) Контрольная сумма UDP. Проверка целостности. Если нарушена, то просто отбрасывает без запроса о повторной отправки.

5) Данные. Здесь упакованы данные с верхнего уровня. Например, когда веб-сервер отвечает на запрос клиента и отправляет веб-страницу, то она будет лежать в этом поле.

Как видите, у него не так много полей. Его задачи — это нумерация портов и проверять побился кадр или нет. Протокол простой и не требовательный к ресурсам. Однако он не может обеспечивать контроль доставки и повторно запрашивать побитые куски информации. Из известных сервисов, которые работают с этим протоколом — это DHCP, TFTP.

Переходим к более сложному протоколу. Встречаем протокол TCP. Смотрим, из чего состоит, и пробегаем по каждому полю.



1) Порт источника и порт назначения. Выполняют те же роли, что и в UDP, а именно нумерация портов.

2) Порядковый номер. Номер, который используется для того, чтобы на другой стороне было понятно какой этот сегмент по счету.

3) Номер подтверждения. Это поле используется, когда ожидается доставка или подтверждается доставка. Для этого используется параметр ACK.

4) Длина заголовка. Используется для того, чтобы понять какой размер у TCP-заголовка (это все поля, представленные на картинке выше, кроме поля данных), а какой у данных.

5) Зарезервированный флаг. Значение этого поля должно устанавливаться в ноль. Оно зарезервировано под специальные нужды. Например, чтобы сообщить о перегрузках в сети.

6) Флаги. В это поле устанавливаются специальные биты для установления или разрыва сессии.

7) Размер окна. Поле, указывающее, на сколько сегментов требовать подтверждения. Наверное, каждый из вас наблюдал такую картину. Вы скачиваете какой-то файл и видите скорость и время скачивания. И тут сначала он показывает, что осталось 30 минут, а через 2-3 секунды уже 20 минут. Еще спустя секунд 5, показывает 10 минут и так далее. Это и есть размер окна. Сначала размер окна устанавливается таким образом, чтобы получать больше подтверждений о каждом отправленном сегменте. Далее все идет хорошо и сеть не сбоит. Размер окна меняется и передается больше сегментов и, соответственно, требуя меньше отчетов о доставке. Таким образом, скачивание выполняется быстрее. Как только сеть даст краткий сбой, и какой то сегмент придет побитым, то размер опять изменится и потребуется больше отчетов о доставке. В этом суть данного поля.

8) Контрольная сумма TCP. Проверка целостности TCP-сегмента.

9) Указатель важности. Это смещение последнего октета важных данных относительно SEQ для пакетов с установленным флагом URG. В жизни применяется, когда необходимо осуществить контроль потока или состояния протокола верхнего уровня со стороны передающего агента (например, если принимающий агент может косвенно сигнализировать передающему, что не справляется с потоком данных).

10) Опции. Используется для каких ни будь расширенных или дополнительных параметров. Например, для параметра timestamp, который является своеобразной меткой, показывающей время произошедшего события.

11) Данные. Практически тоже самое, что и в протоколе UDP. Здесь инкапсулированы данные с вышестоящего уровня.