GeekBrains

Лекция 1. Основы компьютерных сетей

Введение в компьютерные сети

Построение телекоммуникаций в сфере IT - это комплекс довольно трудных и объемных задач, начиная от таких как прокладка кабеля по дну океана, заканчивая такими как разделять входящий трафик на сетевую карту между приложениями на отдельно взятом компьютере. Как и любая большая задача, она разбивается на несколько небольших независимых, каждая из которых решается отдельно. В сетях такие подзадачи называются уровнями. В стеке TCP/IP, который стал стандартом де-факто в мире сетей, таких уровней 4. На каждом уровне описаны свои правила работы в сетях - протоколы. И в начале каждый наш урок будет посвящён изучению одному или двум уровней.

Сначала мы рассмотрим вкратце первый – физический уровень компьютерных сетей. Мы не будем заострять на нём внимание, т.к. разработчики редко касаются его. Именно этот уровень отвечает за взаимодействие участников сети друг с другом на уровне проводов, соединений, сигналов, которые посылаются друг другу на основе физических законов природы.

Научить устройства принимать и передавать сигналы это хорошо, но на электрические сигналы, на сами электроны, нельзя повесить никакую логику. Мы не можем на них повесить какие-либо ярлыки и как-либо их пометить, поэтому нам нужен второй уровень - канальный, на котором решается задача маркировки трафика: от кого этот трафик идет, кому он должен идти, каким методом он должен передаваться, и так далее. Эта проблема решается на втором уровне, его мы и изучим следующим.

После изучения уровня L2, мы поймем, чем он ограничен и почему чисто на нем нельзя построить глобальные и большие сети. И тут мы

познакомимся с уровнем L3, он же сетевой уровень, на котором решается задача передачи трафика между большим количеством сетей. В стеке TCP/IP на L3 работает протокол IP, интернет-протокол.

После знакомства с L3 мы поговорим о маршрутизации на роутерах - устройствах, которые объединяют сети.

Далее мы перейдем к следующему уровню, L4. Он же транспортный уровень. И поймём зачем он нужен, поговорим о двух основных протоколах, это UDP и TCP, чем они отличаются, какую задачу решает каждый из них.

Зная об этих четырёх уровнях, мы поймем как ходит трафик от конкретного приложения к конкретному приложению в сетях, поэтому после этого мы начнем изучать дополнительные технологии в сетях. Например, мы познакомимся с технологией, которая позволяет нам экономить IP адреса, технологией NAT, а также поговорим про VPN.

Также мы рассмотрим основные проблемы информационной безопасности в сетях, поговорим про шифрование, про сертификаты, а также затронем основы HTTP и DNS.

Давайте немножко поговорим, почему так важно знать, как работают компьютерные сети любому IT-специалисту.

В фундаменте IT сферы лежат такие основные отрасли как разработка, администрирование систем, информационная безопасность и телекоммуникационная отрасль. Хороший специалист отличается тем, что знаком со всеми этими отраслями.

Например я, как сетевой инженер, помимо сетей, также разбираюсь:

- в разработке, и я могу на питоне написать простую многопоточную программу для более эффективной работы со своим сетевым оборудованием
- я разбираюсь в системе Git, чтобы реализовывать подход infrastructure-as-code
- я немного разбираюсь в базах данных чтобы хранить информацию о IP-адресах своих сетевых устройств и пользователей и могу делать так, чтобы скрипты или программы обращались к этой базе данных
- я также разбираюсь в системном администрировании для того чтобы разворачивать свои вспомогательные сервисы, такие как сервис мониторинга Zabbix, или сервис аутентификации на сетевое оборудование.

Занимаясь разработкой, с большой вероятностью вы будете писать приложения, которые работают в сети. Поэтому знание о том, как работают компьютерные сети может помочь вам в решении ваших задач, при отладке, при улучшении отказоустойчивости ваших приложений. Вы сможете не только заниматься кодингом, но и смотреть немного шире на свои проекты, понимать в какой среде они размещены, как организован к ним доступ.

Сегодня мы познакомимся:
- с базовой терминологией в компьютерных сетях,
- с моделью OSI,
- поймем какие задачи решает L1,
- начнем знакомится с уровнем L2.

Виды связи

Поговорим сначала про виды связи такие как: симплексная, Half-duplex и Full-Duplex.

Симплексная связь подразумевает, что мы имеем один источник данных, а остальные все только получают эти данные. Если приводить примеры, то симплексная связь реализована у нас на радиостанциях, спутниках GPS, в телевидении. Радиовышки, спутники постоянно передают сигнал на приемники, которые только слушают, обратно послать сигнал они не могут. В таких сетях связи не может идти и речи о интерактивности между узлами.

Развитием симплексной связи является полудуплексная связь или Half-duplex. Это когда у нас есть одна среда для передачи данных, и узлы сети конкурируют за эту среду и передают в ней информацию по очереди. В таких сетях связи в конкретный момент времени у нас только один кто-то вещает, а остальные все слушают. Отличным примером здесь является общение по рации. Если вы сами никогда по рации не общались, то вы, наверное, видели вживую или в фильмах, что когда кто-то хочет сказать что-либо по рации, то он нажимает специальную кнопку, в этот момент он превращается в радиовышку, а остальные в приемники, и так люди общаются по очереди в радиоэфире. Это и есть half-duplex’ная связь. Если же начинается одновременное вещание двух узлов сети, то происходит наложение сигналов и их искажение, которое называется коллизией. Чем больше узлов в сети с Half-Duplex, чем больше данных они передают, тем больше вероятность возникновения таких коллизий. Сейчас в современных сетях half-duplex вы редко где встретите.

Чтобы избегать коллизий, вместо half-duplex лучше использовать full-duplex. При организации такой связи у нас есть две независимые среды для передачи данных. В одной среде происходит передача данных в одну сторону от узла А к узлу Б, а во второй среде происходит передача уже от узла Б у узлу А. Таким образом данные сигналы помешать друг другу не могут, тут нет места для возникновения коллизий. Для лучшего понимания сравним Half-Duplex с узкой дорогой во дворе, где если встретятся две

машины, то это создаст пробку и замедлит трафик, тогда Full-Duplex - это дорога с двумя полосами, где две машины разъедутся спокойно, не помешав друг другу. Full-Duplex - самый эффективный способ связи между двумя узлами.

Теперь давайте поговорим о коммутации пакетов, мы чуть выше уже упоминали про пакеты пакеты данных. Так почему же связь происходит именно пакетами - порциями данных? Для того, чтобы понять почему это более эффективный способ передачи данных в интерактивных сетях, давайте сначала рассмотрим сеть на основе коммутации каналов.

Ярким примером таких сетей были телефонные городские сети. В таких сетях, когда вы, абонент А, звоните абоненту Б, между вами резервируются целый канал связи. Вы общаетесь только между собой, больше никто не может позвонить ни абоненту А ни абоненту Б. Ни о каком одновременном взаимодействии между абонентами речи не идёт. Предположим, что в интернете была бы связь на основе коммутации каналов, представьте что абонент Б это сайт mail.ru, а абонент А это я.

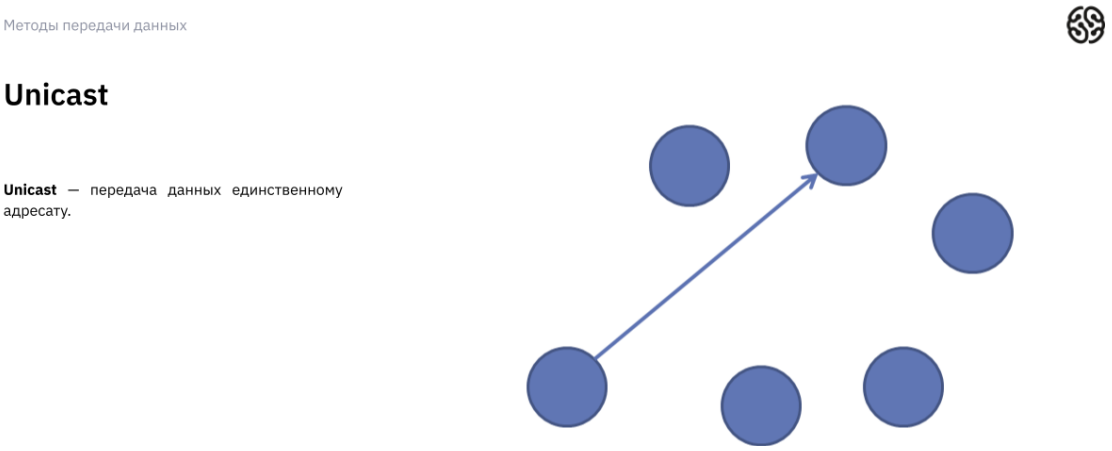
И вот я сижу на этом сайте и больше никто не может зайти на него. Удобно? Нет конечно же.

Поэтому с появлением производительных процессоров и с появлением возможности нарезать данные с высокой частотой, появилась концепция сетей с пакетной передачи данных. В таких сетях данные от абонента А к абоненту Б идут не непрерывным сигналом, а небольшими порциями - пакетами. Поэтому когда к абоненту Б обращается и абонент А, и абонент С, пакеты к нему перемешиваются, и как бы поступают одновременно. Как там дальше абонент будет различать эти пакеты это дело третье, главное мы обеспечили одновременное взаимодействие между абонентами.

Именно поэтому компьютерные сети - это сети с передачей пакетов. Именно эти пакеты мы будем тщательно разглядывать и изучать ближайшие 6 уроков. Решая такие вопросы, как их более эффективно передать, как их гарантированно доставить, как их доставить на максимально большой скорости, и строится огромная отрасль телекоммуникаций.

Методы передачи данных

Давайте еще познакомимся с методами передачи данных. Пока поговорим об этом абстрактно, не привязываясь ни к какой сети и ни к какому протоколу.

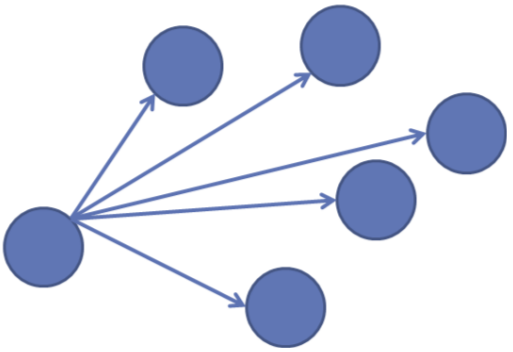


На примере данного слайда мы видим так называемую юникаст передачу, когда один участник сети, условно Маша, знает что ей надо передать данные условно Пете. И она передаёт пакеты точечно только ему, пакеты другим участникам сети не долетают, это называется юникаст вещание.

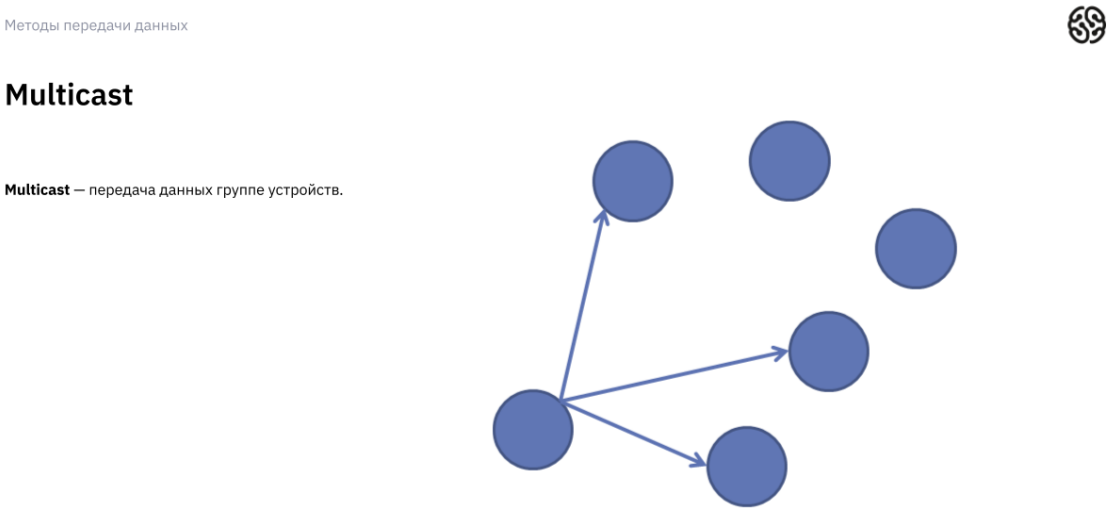
Методы передачи данных

Broadcast

Broadcast — широковещательная передача данных всем устройствам.



Противоположным примером является бродкаст вещание, или широковещательное вещание. При таком способе передачи данных пакеты доставляются всем участникам сети. Когда такое вещание необходимо? Предположим, что Маше надо передать сообщение Пете, но она не знает кто в сети Петя, тогда она спрашивает сразу всех: “Кто здесь петя?”. Получив ответ (ответ пойдет юникастом), она уже начинает с ним юникастово общаться. Важно понимать, что бродкаст вещание очень сильно нагружает сеть, так как пакеты множатся для всех её участников, поэтому броадкаст чаще всего необходим для работы служебных протоколов.

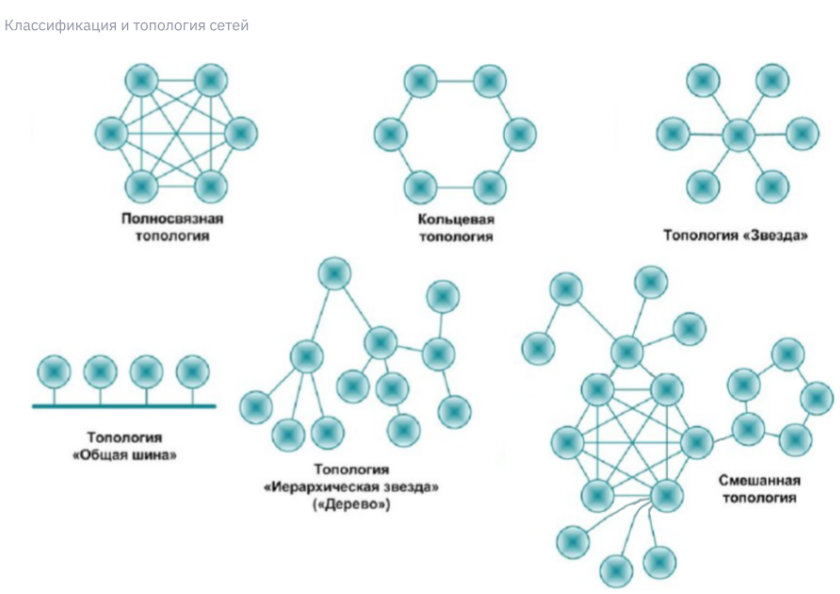


Существует также вариант усеченного броадкаст вещания, когда данные доставляются только определенной группе узлов - мультикаст. Ярким примером мультикаст вещания является IPTV, когда у некоторых абонентов есть подписка на какую-либо телепередачу, а у других нет.

Классификация сетей

Также давайте познакомимся с классификацией сетей по административно-территориальному признаку и введем понятие LAN и WAN.

LAN - это локальная вычислительная сеть Local Area Network. Такая сеть строится в ограниченной области, по единой политике, построенной администратором или командой администраторов. При этом, всё что для нас не является LAN - это WAN (Wide Area Network), окружающая нас сеть.



Топология сети

Топология - это схема сети.

Самой надежной топологией, но при этом и самой дорогой является полносвязная топология. На такой топологии каждый узел сети связан с каждым другим узлом сети. Отметим, что здесь при увеличении количества узлов, увеличение связей растет по экспоненте. Посмотрите на слайд, на полносвязной топологии мы видим 6 узлов, которые связаны между собой “каждый с каждым”, для того чтобы добавить ещё один узел, нам нужно уже 7 линков, чтобы потом добавить ещё один уже 8 линков, в итоге для добавления двух дополнительных узлов, нам нужно уже 15 линков. Но зато, если у нас выходит из строя несколько линков или узлов, связность между оставшимися узлами останется. Эта топология самая отказоустойчивая.

Еще одной топологией является топология кольцо. Такая топология минимально отказоустойчивая. При такой связности, при разрыве

одного линка, всегда остаётся один резервный путь для связи узла А а с узлом Б. И у нас есть время для того, чтобы восстановить отказавший линк.

Обратите внимание что в таких двух топологиях есть небольшой изъян. Если мы настроим сеть некорректно, то есть вероятность, что некоторые пакеты попадут в петлю и будут вечно пересылается между узлами, засоряя сеть. Чего не может быть например в топологии дерево. Посмотрите на неё внимательно, в дереве нет места для возникновения таких петель.

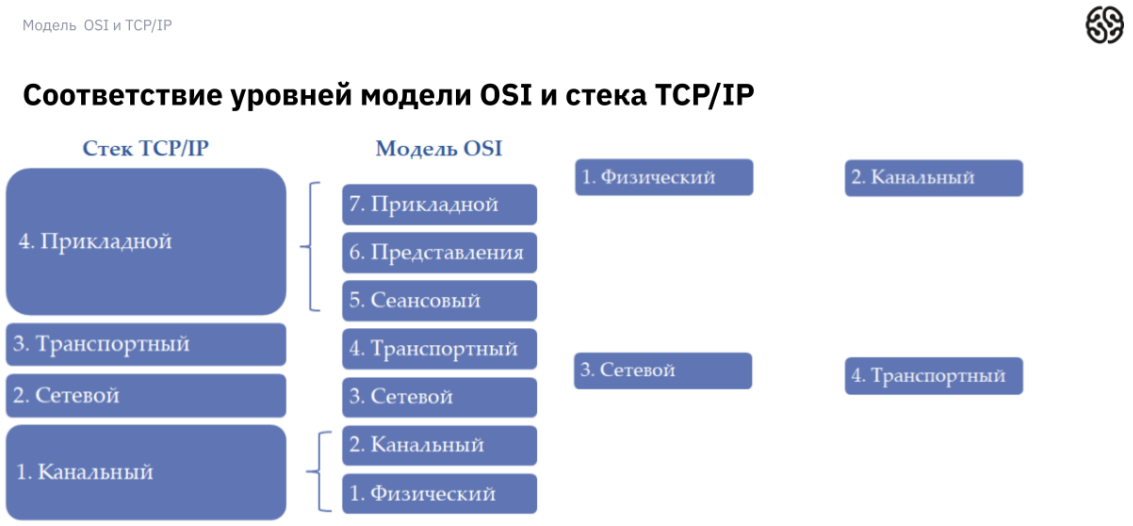
Также существует топология общая шина, где в одну среду передачи данных подключается сразу несколько участников сети. Как вы наверное уже поняли в такой топологии есть возможность для возникновения коллизий, поэтому в общей шине передача данных будет идти в халф-дуплексе.

Недостаток такой топологии также еще в том, что если у нас рвется линк в одном месте, то одна половина сети сразу ломается и становится недоступной. Чтобы решить такую проблему можно узлы сети соединить топологией звезда, тогда при разрыве кабеля, из строя выйдет только один узел.

Скорее всего, на предприятии, в ЦОДах, на проектах, вы встретите сочетание нескольких топологий. Обычно сети не строятся только согласно одной выбранной топологии. Часть сети может работать в виде кольца, другая часть сети в виде дерева, третья часть сети может выступать в виде ядра и соединять отдельные сети с помощью полносвязной топологии.

Модель с OSI и стеком TCP/IP

Чуть ближе познакомимся с моделью с OSI и стеком TCP/IP. Как я и говорил ранее компьютерные сети это довольно масштабная задача, особенно если мы строим такую сеть как “Интернет”. Как и любую другую большую задачу, мы разбиваем её на несколько маленьких, независимых друг от друга задач, а потом соединяем их какой-либо логикой.



Тоже самое было сделано и с компьютерными сетями, правила взаимодействия узлов в сетях были разбиты на уровни. Подхода построений таких уровней было два.

Первый подход - больше теоретический, его концепцию начала разрабатывать в 1982 году Международная Организация по Стандартизации ISO (International Standardization Organization) и называлась она OSI (Open System Interconnection - модель взаимодействия открытых систем), эта модель была больше теоретическая и большая часть её протоколов так и не вышла в свет или не прижилась в сетях. Модель содержала в себе 7 уровней.

Параллельно с этим в 70-х годах развивался стек более простых протоколов TCP/IP, который был популярен в молодой тогда сети Интернет.

На первом уровне OSI (и первой половине Канального уровня TCP/IP) решалась задача по стандартизации передачи данных на физических линиях связи, то есть задача передачи битов, ноликов и единичек, задача по их кодированию. Ведь если производитель А сделает устройство на котором будет приниматься сигнал по двум проводам с напряжением 5 вольт а производитель Б начнёт производить сетевое оборудование на котором будет приниматься сигнал по четырем проводам с разницей напряжения 3, 6 и 12 вольт, то к сожалению они не смогут друг с другом работать. Поэтому надо договориться о уровнях напряжений, а если сигнал передается по оптоволокну, то необходимо договориться о длине волны, сечении оптоволокна и прочих физических параметрах. На этом уровне работают законы физики, собственно поэтому уровень называется физическим.

На втором уровне модели OSI у нас уже формируется понятие пакета данных как некоторого куска данных, появляется логика, мы можем этот пакет абстрактно изобразить. Пакет разделяется на две части: сами данные и метаданные - данные о данных. Эти метаданные называются заголовком пакета, или header. Зачем он нужен? Если у нас участников сети становятся более чем 2, то как правило, у нас уже появляется некое сетевое оборудование, которое занимается пересылкой данных. И узлам сети необходимо различать пакеты друг от друга, от кого кому они идут, какие данные эти пакеты передают, и так далее, вот собственно эта информация и описана в заголовке пакета.

Далее нам нужен еще один уровень. Зачем? А представьте, что мы в нашу небольшую сеть подключаем компьютер Маши, ей надо передать информацию Пете, но она не знает кто в сети Петя. Что она

делает? Помните? Она рассылает броадкаст. Если все устройства в мире будут работать в одной сети, и слать друг другу броадкасты, то такая сеть будет заполнена только одним таким броадкастом. При этом мы теряем в гибкости и в масштабируемости такой сети, а еще и в безопасности, ведь весь мир будет знать, что Маша ищет Петю.

Поэтому на сцену выходит такое понятие как отдельная сеть, в которой ограничен броадкаст трафик. Именно множество таких независимых сетей, соединенных между собой и представляет из себя интернет. Ведь если мы посмотрим на само слово Интернет, то увидим что оно состоит из двух слов - “Интер” и “Нет”. Где “Интер” можно перевести как “между”, а “Нет”, как “сеть”, получается - “межсеть”, а Internet Protocol - “протокол межсетевого взаимодействия”.

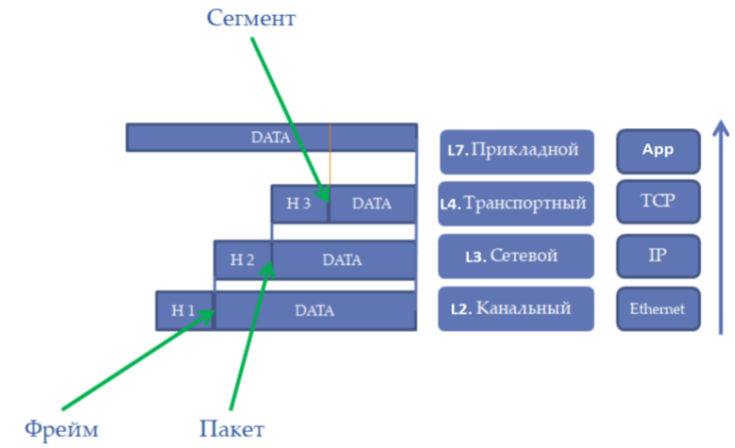
Именно эти три уровня позволяют осуществлять доставку трафика от любого хоста к любому другому хосту в сетях, если все настроено верно. Но на этом еще не всё. Как быть в следующей ситуации: пакет данных прилетает на компьютер, компьютер смотрит в заголовок этого пакета и понимает что он ему, комп отрезает заголовок у пакета, а дальше идут нолики и единички, биты Payload’a. Как быть с этими битами? Какому приложению передать их дальше? Может быть это для веб-браузера? Может быть это для мессенджера? может быть это данные протокола ssh? Который служит для администрирования удалёнными серверами?
Если процессор будет заниматься перебором и угадыванием какому приложению передать дальше эти нолики и единички, это будет совсем не эффективно. Данные на сетевой интерфейс могут прилетать со скоростью несколько десятков тысяч пакетов в секунду. Тут нас и выручает L4 - транспортный уровень, который как раз решает эту проблему и отвечает за доставку трафика от конкретного приложения до конкретному приложению в сети.

Далее у нас начинаются различия в моделях OSI и TCP/IP, в OSI представлены такие уровни как сеансовый и представления. В теории сеансовый уровень должен был отвечать за поддержание сеанса связи между различными приложениям, чтобы взаимодействие устанавливалось на длительное время. Уровень представления должен был отвечать за представление форматов данных, которые передаются по сети, то есть определять должен ли это быть ASCII код или UTF для текста, или JPEG или PNG для изображений. На практике пятый и шестой уровни оказались не нужны. Предыдущих 4 уровней хватает для того, чтобы трафик дошел от приложения к приложению, а как будут общаться между собой эти приложения договариваются уже разработчики этих приложений. Для этого было достаточно одного уровня - L7, он же прикладной уровень или application layer.

Исторически сложилось так, что TCP/IP стал практически повсеместно используемым протоколом. Но для обозначения уровней в технических характеристиках устройств, в литературе, при общении, используют семиуровневую модель. В итоге получаем, что:

- L1 -это физика,
- L2 - канальный уровень,
- L3 сетевой,
- L4 транспортный,
- и L7 прикладной.

Стек TCP/IP. Инкапсуляция



Важно понять, что на каждом из этих уровней есть свой заголовок внутри пакета и эти заголовки следуют друг за другом. Это называется инкапсуляция. Сетевое устройство которое пересылает пакеты, принимает решение куда дальше направить пакет, на основе заголовка определённого уровня.

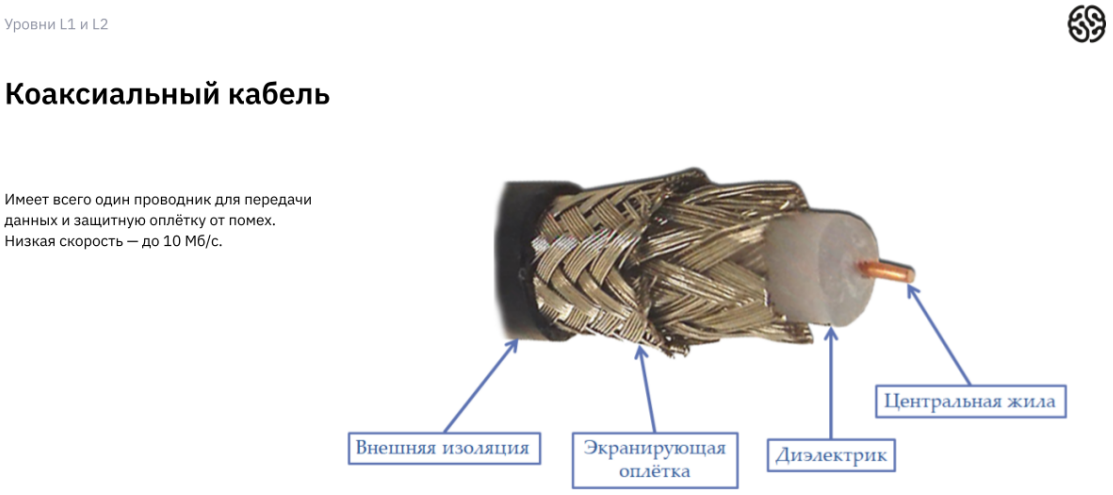
На каждом уровне пакет носит своё название. На L2 - это Фрейм, на L3 - это пакет, на L4 - это сегмент. Но такие названия больше используются в официальной литературе, на практике все обычно просто говорят пакет, и уже из контекста разговора понимаем о заголовке какого уровня идёт речь.

Также нужно учитывать, что эти уровни независимы друг от друга. Что имеется в виду? Например, физика у нас может быть как оптикой так и медными проводами, передаваться сигналы могут с помощью одного метода кодирования или другого, но это не влияет на то, что будет у нас на уровне L2, там формат заголовка пакета не поменяется от способа передачи бит на L1. Точно также выбор протокола L2 не влияет на заголовок L3 уровня. У нас пакеты могут идти из одной IP сети в другую, а под собой использовать канал, построенный например по протоколу Ethernet или по X.25 (это устаревший

протокол передачи данных канального уровня в основном через телефонные сети).

Уровни L1 и L2

Теперь давайте перейдём к разбору уровня L1. Данный уровень, как и говорилось ранее, предназначен для определения физических характеристик и величин, на которых должны работать устройства, чтобы понимать друг друга. Самые распространненые протоколы на данном уровни входят в стандарт под номером 802.3 - Ethernet. Данный Стандарт был описан в IEEE - в институте инженеров по электронике электротехнике. Этот стандарт сразу описывает работу двух уровней L1 и L2. Исторически самый первый стандарт описывал работу сетевых устройств, соединенных коаксиальным кабелем.



Вы могли его встречать при подключении телевизора, раньше он также использовался и для передачи данных. В таком кабеле есть одна медная жила, которая из себя представляла конкурентную среду передачи данных. Максимальная скорость передачи данных по такому кабелю была 10Мб/с в секунду, и это в идеальных условиях.

Уровни L1 и L2

Hub

Ранее сети строились на коаксиальных кабелях, которые замыкались с помощью хаба.

Был период, когда сети строились на коаксиальных кабелях, которые соединялись с помощью хаба - простого устройства повторяющего сигнал на все свои порты, что создавало возможность для наложения сигналов и возникновения коллизий. При этом сигнал, который шел от одного компьютера к другому доходил до всех других участников сети и все знали о чем эти два компьютера общаются. Обратите внимание, что если мы соединим через хаб 5 компьютеров, то это не будет полносвязной топологией между ними, т.к. при разрыве линка компьютер теряет связь сразу со всеми. Втыкание в хаб можно сравнить с топологией общая шина (которую мы рассмотрели ранее), по сути в хабе создается общая конкурентная среда, общение в которой возможно только по half-duplex - домен коллизий.

Уровни L1 и L2

Витая пара

В отличии от коаксиала, можем организовывать цифровую передачу данных, замыкая/размыкая цепь.
Скорость до 100 Мб/с, если использовать 8 жил — до 1 Гб/с, специальные сетевые карты — до 10 Гб/с.

Но стояла проблема коллизий и ограничений по скорости, поэтому развитием этого коаксиального кабеля стал многожильный кабель, который называется витая пара. Витая пара состоит из скрученных между собой попарно проводников. По каждой паре передается сигнал в одну сторону. Благодаря тому, что проводников в паре два, мы можем замыкая/размыкая цепь, организовывать цифровую передачу данных, в отличии от коаксиала. Замыкая/размыкает цепь на каждой паре, мы можем передавать единички и нолики. Передаются они конечно же не прям в таком чистом виде, на оборудовании используются свои методы кодирования, которые позволяют передавать эти биты более помехозащищенным способом.

В сетях построенных на основе витой пары с двумя парами, раньше были такие, можно было уже реализовать протокол Fast Ethernet, в котором скорость передачи данных могла достигать уже 100 Мб/с с Full-Duplex’ом. Обратите внимание, что тут одна пара служит для приёма а другая для передачи, что дает нам возможность избегать коллизий.

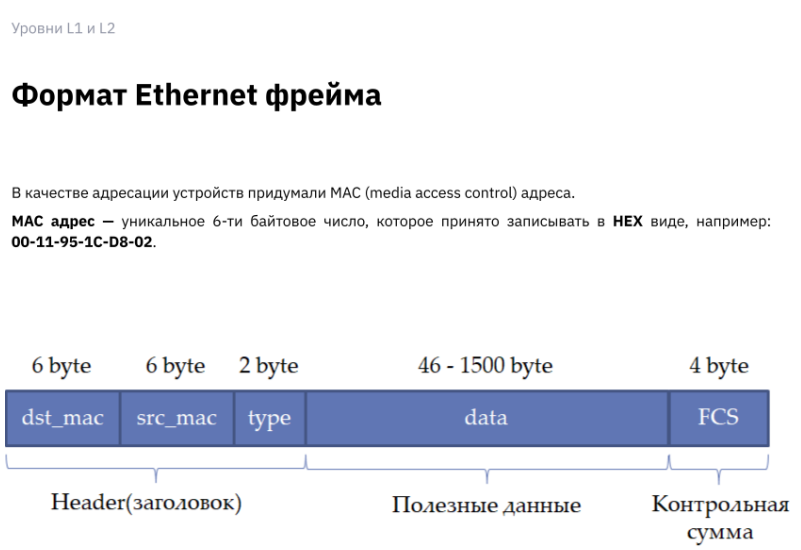
Электроника развивалась и в какое-то время мы научились создавать чипы, которые могли одновременно передавать и принимать данные

по всем 8 жилам витой пары без коллизий. Это позволило достигать скорости до 1Гб/с, и на смену Fast Ethernet, пришёл Gigabit Ethernet.

На сегодняшний день есть стандарт и существуют сетевые карты, которые уже позволяют разгоняться до 10 Гб/с. Реализуется это благодаря специальной витой паре, в которой каждая пара идет отдельно друг от друга в защитной металлической оплетке.

Стоит отметить что длина сегмента на витой паре в стандартах Ethernet ограничена ста метрами из-за затухания сигнала в медном проводнике. Когда необходимо передать сигнал на более длинное расстояние, необходимо использовать оптическую линию связи. В такой среде вместо электронов мы используем фотоны, которые двигаясь по тонкому стеклянному проводу, встречают гораздо меньше сопротивления, чем электроны в меди. В оптике также существуют многообразие стандартов и оборудования, которые имеют различные характеристики. Например:
- какой источник света используется? Светодиод или мощный лазер.
- по двум жилам передается/принимается информация, или по одной?
- свет какой длины волны используется?
- какое сечение оптоволокна?

Такой подход позволяет строить как небольшие корпоративные сети, так и глубоководные оптические линии связи и передавать сигналы на расстояние 50 км.

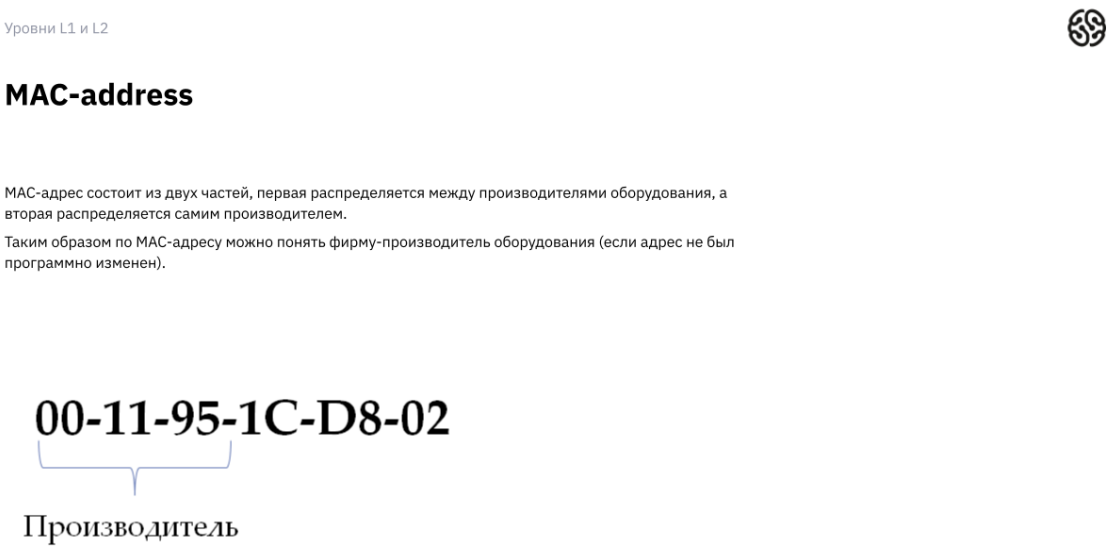


После того, как мы научились передавать сигналы от от узла к узлу, давайте разберёмся с тем вопросом, с той проблемой, как вообще узел может отличить ему эти электроны летят, или не ему? К сожалению на электрон мы не можем повесить ярлык и написать, что этот электрон должен лететь от Маши к Пете. Но зато, мы можем объединить пачку электронов, и в этой пачке часть отвести для того, чтобы закодировать информацию о том, от кого и кому летят эти электроны. Такая информация называется метаданными - данные о данных. Таким образом мы вносим человеческую логику в законы физики, и у нас появляется канальный уровень L2. Метаданные еще называются хедером или заголовком пакета.

Всё тот же стандарт Ethernet описывает какой должен быть заголовок на уровне L2. Считывая эти заголовки, сетевые устройства

принимают решение куда дальше направлять пакет и как обрабатывать его.

Давайте рассмотрим структуру L2 заголовка, она довольно-таки простая. Первые 6 байт отведены под Destination адрес, то есть адрес кому этот пакет предназначен, вторые 6 байт выделены под Source адрес, то есть адрес узла, который отправил этот пакет, эти адреса называются MAC адресами, они в основном назначаются на сетевые интерфейсы оборудования, серверов, узлов сети.



У каждой сетевой карты на узле в сети должен быть свой уникальный MAC адрес. Каким образом это достигается? Дело в том, что производитель сетевого оборудования и сетевых карт не может сам рандомно назначать MAC адреса на них, он должен выкупить первые 3 байта всё в том же институте IEEE. Институт за деньги выделяет такой блок и производитель по очереди назначает на устройства адреса из этого блока, пока не исчерпает его. Как только у него заканчиваются адреса в этом блоке, он идет покупать следующий блок. Таким образом достигается, что не должно существовать 2 устройств с одинаковым MAC адресом. Но я отмечу что это в теории,

на практике MAC адрес на сетевой карте можно поменять вручную, но не нужно этого делать, без лишней необходимости, чтобы не сломать работу сети.

Итак когда у нас устройство А хочет послать пакет устройству Б, он заполняет все данные в пакете, пакет с незаполненными полями не может быть послан сетевым интерфейсом. Запомните это правило. Пока вынесем за скобки то, как устройство узнаёт Destination MAC адрес, просто знает и всё. Source MAC адрес, то есть MAC адрес своей сетевой карты, он сам знает.

Далее идет поле type, которое описывает какой протокол следует у нас дальше. Зачем он нужен? Предположим, что этого поля не было бы. Тогда, принимающая сторона, как только получает такой пакет и понимает от кого он отправлен, понимает что он отправлен ему, отрезает заголовок L2. А дальше идут нолики и единички - какие-то данные. И что делать дальше с этими данными, какому процессу передать эти нолики и единички? На обработку какого протокола? Может быть ipv4? Может быть это ipv6? Может быть это протокол ARP? Если процессор на сетевой карте начнет тратить время на то, чтобы угадать как дальше обрабатывать пакет (а я уточню, что таких пакетов может прилетать десятки тысяч в секунду), то это будет совсем не эффективно. Поэтому отправитель должен заполнить поле type для того, чтобы принимающей стране было проще и быстрее пакет обработать. Запомним этот приём, так как указание инкапсулированного протокола будет сопровождать нас на каждом уровне.

Дальше идёт payload или полезные данные, которые несут в себе информацию для следующего уровня.

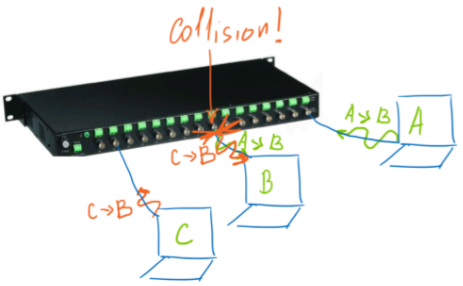
Нам также важно понять, пострадали ли данные например от помех, пока пакет летел по каналу связи. Для этого в конце у нас есть контрольная сумма, checksum. Чтобы понять, как этот механизм работает, представим что это просто побитовая сумма, и не четырех байтная, а четырех битовая для всего нашего пакета. Предположим что в пакете идут следующие данные 0110 1101 1010 1001. Нарежем их по 4 бита, и просто сложим в столбик побитно:
0111
1101
1010
1001
—----
1001

Это значение и будет нашей упрощенной контрольной суммой. Отправитель обязан вписать её в последние поле. Если такой пакет подвергнется помехам, и где-то поменяется 1 бит. Например здесь 0110 1001 1010 1001, то контрольная сумма которую посчитает получатель при приеме такого пакета не совпадёт с той, которую отправил отправитель. И данный пакет должен быть уничтожен, как искажённый. Это простой принцип описывающий механизм контрольной суммы. Но теперь усложним немного этот принцип. Допустим, что у нас также подвергся изменениям второй бит в третьем слагаемом 0110 1001 1110 1001. Тогда получается, что у нас контрольные суммы совпадут, хотя пакет по факту будет искаженным. Дабы такого избегать, вместо просто обычной побитовой суммы, используются более сложные математические функции, и вместо 4 бит берутся более длинные куски по 4 байта.

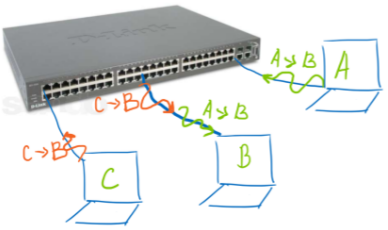
Уровни L1 и L2

Hub vs Switch

Hub физически повторяет сигнал на все порты одновременно.



Switch умеет передавать сигналы точечно получателю, при этом он выстраивает пакеты в очередь (проводит буферизацию).



Сформировать наши пакеты с заголовком L2 - это половина дела. Если мы будем посылать такие пакеты на самое простое устройство - на хаб (который, напомню, просто повторяет сигнал на все порты), то у нас будет оставаться возможность для возникновения коллизий. При коллизиях сеть не будет эффективно работать. Когда пакетов будет много и участников сети тоже будет много, то коллизии будет возникать все чаще и чаще. Поэтому нам нужно придумать устройство, которое умеет считывать этот заголовок L2 и принимать решение куда, на какой порт, отправить этот пакет, а не плодить его глупо во все порты. Такое устройство называется свитч, он же коммутатор. Его принципы работы мы и будем рассматривать на следующем уроке.

Подведем итог. Сегодня мы познакомились с основной терминологией в компьютерных сетях, которая пригодится нам в дальнейшем. Познакомились с моделью OSI. И если даже она пока у вас вызывает вопросы, то ничего страшного, к концу этого курса вы поймёте что она из себя представляет целиком и у вас сформируется картинка в голове^ для чего она нужна. Также мы вкратце познакомились с уровнем L1, на котором реализован стандарт Ethernet, который буквально говорит вендорам, как производить

сетевые карты Ethernet, сетевые устройства, работающие по протоколу Ethernet, какими физическими характеристиками они должны обладать и прочее. И еще начали знакомиться с уровнем L2, рассмотрев структуру заголовка пакета.

Программы для имитации работы сети

Теперь о практике. Для изучения сетей на начальном этапе нет необходимости приобретать само сетевое оборудование. Существуют программы для имитации работы сети.

Самая популярная это **Cisco Packet Trace**r, я и сам вначале в ней изучал сетевые технологии. В ней очень неплохо реализована симуляция основных сетевых технологий, что очень подходит для новичков.

Преимущества:

- легкая установка/настройка
- поддержка симуляции различного оборудования, начиная с сетевых устройств, заканчивая IoT
- симуляция Wi-Fi
- бесплатная
-

Недостатки:

- закрытый код
- имитация устройств (не запускает настоящие образы)
- изучать можно только с оборудование Cisco
- на устройствах доступен не весь функционал

Есть альтернатива от Huawei, называется **eNSP**, в отличии от CPT, она использует учебные образы сетевого оборудования, в том числе и популярного файервола USG, но остальные недостатки те же.

Также еще существуют проекты EVE-NG и GNS3, они представляют из-себя виртуализацию, а внутри позволяют использовать реальные образы оборудования разных производителей, что позволяет более глубоко изучить возможности сетевых устройств, но при этом они ограничены на уровнях L1 и L2. Стоит отметить что GNS3 - opensource проект, который довольно неплохо поддерживается. Также их несколько сложнее устанавливать чем CPT, но и возможностей у них больше. С какой программой мы будем работать и как её получить, смотри в описании к этому ролику.

Во время семинара:
- мы познакомимся с одной из таких программ, запустим симуляцию работы сети, в которой будем выполнять практику
- изучим что и как происходит на уровне L1, как менять такие параметры подключения как скорость и дуплекс
- посмотрим на состав пакета
- научимся смотреть настройки сетевой карты
- попробуем починить “поломанную” сеть.
- начнем знакомиться с операционной системой Cisco IOS на коммутаторе, которая нужна для управления сетевым оборудованием.

Глоссарий

Телекоммуникации – отрасль связанная с передачей информации на расстояние.
Протокол – набор правил, позволяющих стандартизировать.
ISO – International Organization for Standardization - международная организация по стандартизации.
Сеть – объединение хостов с помощью телекоммуникаций для передачи информации между ними.
Узел сети (хост) – компьютер, сервер или другое конечное оборудование, подключенное к сети.
Сетевое оборудование – промежуточное оборудование между хостами.
Модель OSI – набор стандартов и протоколов разработанный ISO для построения телекоммуникационных сетей. Включает в себя 7 уровней.
Физический уровень (L1, “физика”) – первый уровень модели OSI, отвечает за физическое взаимодействие узлов сети.
Канальный уровень (L2, логический уровень, “логика”) – второй уровень модели OSI, отвечает за организацию заголовка L2 в пакетах, а также за взаимодействие узлов в сети в одном L2 домене.
Сетевой уровень (L3, IP уровень) – третий уровень модели OSI, отвечает за организацию множества узлов в сети, а также за передачу трафика между сетями.
Транспортный уровень (L4, TCP, UDP) – четвертый уровень модели OSI, отвечает за доставку данных уже конкретному приложению на узле сети.
Сеансовый уровень (L5) – пятый уровень модели OSI, должен был отвечать за поддержание сеанса связи между приложениями. По факту не используется в стеке TCP/IP, задача решается на L7.
Уровень представления (L6) – шестой уровень модели OSI, отвечает за кодировку и представление данных между приложениями. По факту не используется в стеке TCP/IP, задача решается на L7.

Уровень приложений (L7, Application Layer) – седьмой уровень модели OSI, который отвечает уже за то как приложения будут между собой передавать информацию.
Simplex – вид связи с одним источником информации, когда другие являются только потребителями (пример: радио- и телевещание).
Half Duplex – вид интерактивной связи между участниками, но общение происходит в конкурентной среде, в один момент времени вещать может кто-то один.
Full Duplex – вид интерактивной связи между участниками, но общение происходит в не конкурентной среде, в один момент времени вещать могут все.
Коллизия – искажение сигналов в конкурентной среде передачи данных, возникает когда одновременно передают данные два и более участника сети с Half Duplex.
Домен коллизий – общая конкурентная среда, где есть возможность возникновения коллизий, общение в которой возможно только по half-duplex.
Коммутация каналов – метод коммутации, где резервируются ресурсы связи между участниками сети (пример - телефонная городская сеть)
Коммутация пакетов – метод коммутации, где данные передаются небольшими частями (пакетами), благодаря чему избегается резервирование ресурсов сети, они остаются общедоступными.
Пакет (на L2 - **фрейм** (кадр), на L3 - **пакет**, на L4 - **сегмент**) – часть данных, снабженных заголовками, которые передаются по сети как единое целое.
Заголовок пакета (Header, метаданные) – информация о данных в пакете, по которой сетевое оборудование и хосты понимают как обрабатывать пакет.
Payload (Полезные данные) – то, что идет после заголовка в пакете.
Инкапсуляция - строгая последовательность заголовков разных уровней, которая позволяет определенному сетевому оборудования

на определенном уровне сети абстрагироваться от остальной части пакета.
Юникаст – метод вещания в сети, когда данные предназначены одному участнику.
Броадкаст – метод вещания в сети, когда данные предназначены всем участникам сети.
Мультикаст – урезанный броадкаст, метод вещания в сети, когда данные предназначены определенной группе участников в сети.
LAN (Local Area Network) – локальная вычислительная сеть - сеть находящаяся под единой политикой управления.
WAN (Wide Area Network) – глобальная вычислительная сеть - обычно та сеть, которая находится вне LAN. Понятия часто используются вместе для обозначения назначения портов на оборудовании (LAN/WAN).
Топология – схема сети.
TCP/IP – стек протоколов, завоевавший популярность в телекоммуникациях, практически стал стандартом де-факто в мире компьютерных сетей. На L1,L2 здесь работает протокол Ethernet, на L3 - Internet Protocol (IP), на L4 - TCP (Transmission Control Protocol) или UDP (User Datagram Protocol).
IEEE – Institute of Electrical and Electronics Engineers - Институт инженеров по электротехнике и электронике - разработчик стандарта Ethernet.
Ethernet – стандарт, определяющий как должно работать сетевое оборудование на уровне L1 и L2.
Порт (сетевой интерфейс, “интерфейс”) – разъем, куда вставляется провод для передачи информации. Не путать с портом на уровне L4.
Витая пара (Twisted Pair, “медь”) – многожильный медный провод для передачи информации, завоевал популярность в корпоративных и домашний сетях.
Fast Ethernet (100M, “сотка”) – стандарт Ethernet, позволяющий работать на скоростях до 100 Мбит/с, м.б. реализован как на оптике, так и на витой паре, но чаще на витой паре.

Gigabit Ethernet (1G, “гигабит”) – стандарт Ethernet, позволяющий работать на скоростях до 1 Гбит/с, м.б. реализован как на оптике, так и на витой паре, но чаще на витой паре.
10-Gigabit Ethernet (10G, “десятка”) – стандарт Ethernet, позволяющий работать на скоростях до 10 Гбит/с, м.б. реализован как на оптике, так и на витой паре, но чаще на оптике.
MAC address (MAC) – 48-битный адрес, идентифицирующий хост в сети.
Хаб (Hub, повторитель, концентратор) – устройство повторяющее электрический сигнал с одного порта на все остальные.
Свитч (Switch, коммутатор) – устройство адресно передающее пакеты от одного участника сети до другого.