Лекция 2. Технология Ethernet. Протокол IP

Уважаемые студенты, всем привет. Мы с вами начинаем вторую лекцию по курсу “Основы компьютерных сетей”.

На прошлом уроке мы посмотрели как сети организовываются на физическом уровне и остановились на изучении уровня L2, поговорили о пакете, и рассмотрели заголовок Ethernet.

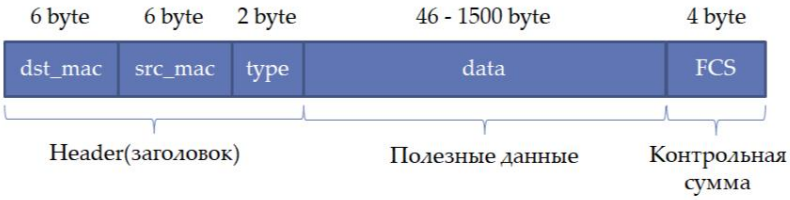
Также напомню, что у нас оставалась проблема с хабом - с простым устройством, которое повторяет пришедший сигнал во все порты. Если мы будем посылать пакеты на хаб, то у нас будет оставаться возможность для возникновения коллизий. При коллизиях сеть не будет эффективно работать. Когда пакетов будет много 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 сегодня.

Сегодня мы продолжим изучение L2 уровня и плавно перейдем к L3. Поговорим о коммутаторах, чем они лучше хабов и как с помощью них удаётся избежать коллизий. Также поймем почему невозможно на одних коммутаторах строить глобальные сети и для чего нам нужны еще и роутеры. Я расскажу вам зачем нам еще один уровень адресации, поговорим об IP адресе и IP сетях.

Уровень L2

MTU

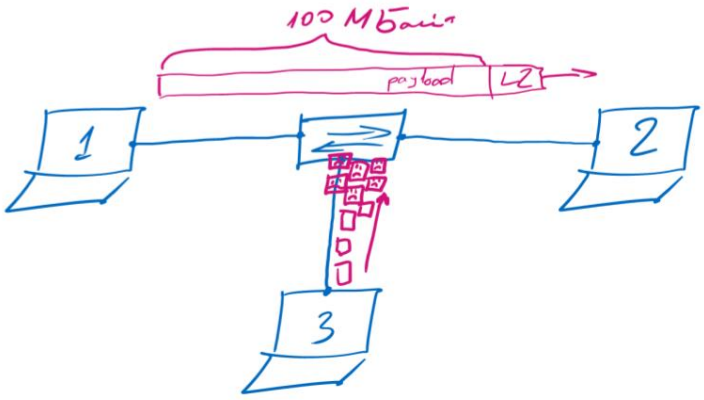
MTU (Maximum Transmission Unit; максимальная единица передачи) - максимальный размер пакета, который может быть передан по сети без фрагментации. Для Ethernet это значение составляет 1500 байт.



Итак, ещё одним важным параметром для пакетов является MTU или Maximum Transmit Unit - это максимальное количество байт, которое может нести в себе пакет после L2 заголовка, т.е. другими словами это максимальный размер payload. Откуда вообще выросло это верхнее ограничение?

Уровень L2

MTU



Давайте рассмотрим следующую схему: у нас есть три хоста и коммутатор, который их соединяет. Коммутатор - это устройство, которое занимается пересылкой пакетов между хостами. Предположим, что один хост начнет начнет передавать другому хосту пакет, payload которого

будет размером 100 МБайт. (Посмотрите на слайд). Такой пакет будет передаваться довольно долго, допустим 5 минут. Получается, что если у нас третий хост захочет передать данные тому же хосту, он пошлёт пакет, который повиснет в буфере на коммутаторе на целых 5 минут, пока он занят пересылкой пакета. На что это похоже? Это практически коммутация каналов, про которую мы говорили на прошлой лекции. Её минус мы знаем - коммутация каналов не дает нам эффекта одновременного общения между всеми. Мы наоборот пытаемся от нее уйти. Поэтому на заре становления протокола Ethernet инженеры договорились ограничивать MTU размером 1500 байт. Получается, что максимальная эффективность передаваемых полезных данных данных будет равна 1-(18/1500)=0,988, где:

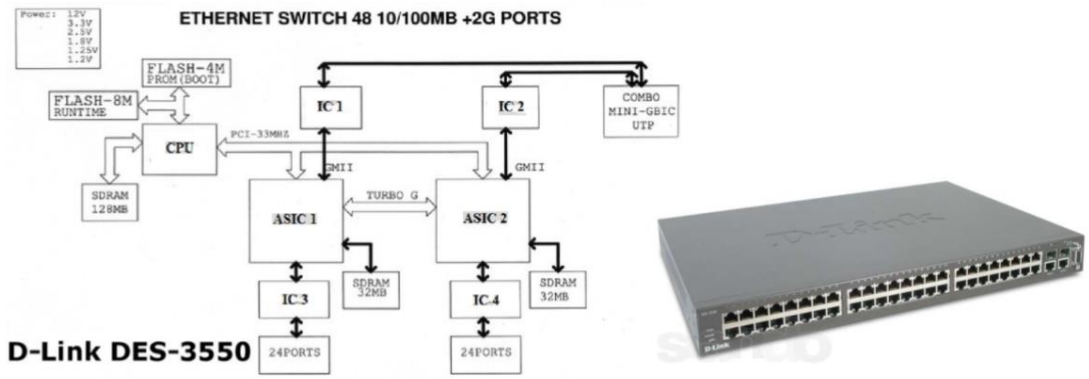
- 18 байт - это заголовок L2 (2*6 байт - MAC адреса, 2 байта - тип протокола, 4 байта - контрольная сумма)
- 1500 байт - MTU

Поделив 18/1500, эту формулу можно переписать в таком виде 1-0,012=0,988, где:

- 0,988 или 98,8% - доля полезной информации
- 0,012 или 1,2% - доля служебной информации, которую несет в себе заголовок L2.

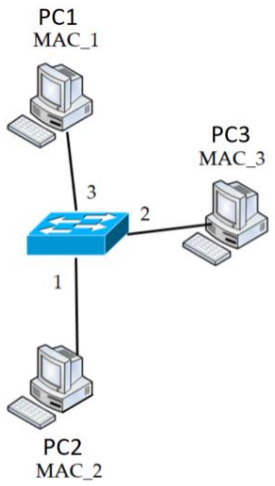
Это именно максимальная эффективность, такой расклад будет у нас только тогда, когда все пакеты в сети будут набиты под завязку по 1500 байт в payload’е, в реальности эффективность меньше.

Но если нам необходимо повысить эффективность, мы можем увеличить MTU на сетевой карте хоста, но учтите, что тогда нужно увеличить MTU и на сетевом оборудовании, куда этот хост воткнут. То есть если у нас компьютер имеет mtu 6000 байт, то порт оборудования, который в него вставлен тоже должен уметь обрабатывать пакеты с MTU 6000 байт, иначе это приведет к фрагментации пакетов - пакеты будут нарезаны на несколько частей, а заголовки специальным образом продублированы.



А теперь поговорим об устройствах которые не просто так пересылают сигналы во все порты, а по-умному, точечно передают пакет в определенному получателю - о коммутаторах (или свичах). Чтобы передать пакет получателю, свич уже должен уметь читать заголовок и уметь доставать из него destination MAC-адрес и понимать по этому MAC’у в какой порт отправить пакет дальше.Свич устроен уже посложнее чем Hub, т.к. ему нужны уже специальные чипы, которые достают MAC-адреса из пакета и память для хранения адресов, портов, настроек.

Таблица коммутации



MAC адрес	Порт свитча

Основная задача коммутатора изучить какие устройства с какими MAC-адресами находятся за каждым из портов. Для этого каждый коммутатор имеет специальную таблицу “MAC address table”. В одном столбце таблицы находится номер порта, а в другом MAC-адреса устройств, которые находятся за этим портом. Эта таблица заполняется динамически, когда трафик проходит через коммутатор.

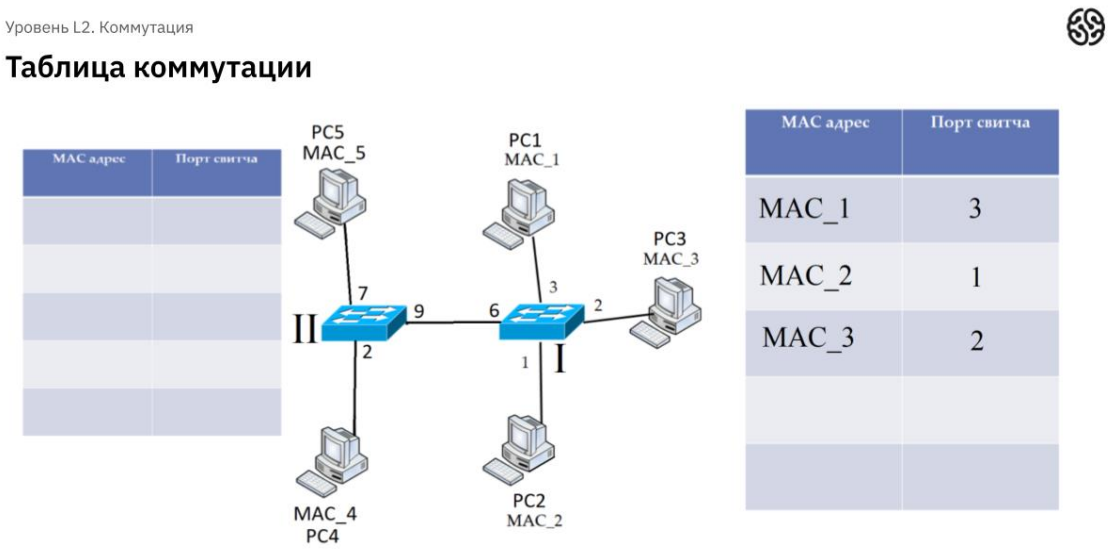
Давайте посмотрим, как заполняется таблица. Посмотрите на слайд. На слайде у нас изображено 3 узла, соединенных с коммутатором, в коммутаторе таблица “MAC address table” пока пустая. Рано или поздно какой-то один из узлов захочет передать данные другому узлу, например PC1 -> PC2.

1. PC1 составляет пакет в котором Source MAC-адрес это его собственный MAC-адрес, а Destination MAC-адрес - это MAC-адрес сетевой карты PC2.
2. Такой пакет отправляется в сеть и прилетает на коммутатор. Первое, что делает коммутатор при приеме пакета на порт, считывает Source MAC-адрес и записывает его в таблицу за портом 3, тем самым запоминая: “Ага у меня там сидит кто-то с MAC1, и этот кто-то шлёт пакеты. Дай-ка я его запишу, вдруг к нему когда-нибудь полетит пакет.”
3. Дальше он считывает Destination MAC-адрес, и ищет его в таблице “MAC address table”. Этого Destination MAC-адреса к сожалению пока нет в таблице.
4. Коммутатор, в надежде что пакет хоть куда нибудь дойдет, рассылает этот пакет во все остальные порты. Такая рассылка называется Unknown Destination Unicast.
5. Узел PC3 получив такой пакет по Destination MAC адресу поймёт что он предназначен не ему и уничтожит его.
6. Узел PC2 получит такой пакет, поймёт что он ему и обработает его. При этом он может отправить ответный пакет.
7. Когда ответный пакет прилетит на коммутатор, он уже запишет Source MAC-адрес узла PC2 за портом 1.
8. Если ли узел PC1 снова захочет отправить какие-либо данные узлу PC2, или, наоборот, PC2 захочет отправить что-то для PC1, коммутатор уже будет знать MAC адреса обоих узлов и пакеты

будут ходить между ними точечно и не будут отправляться в другие порты.

Если узел PC3 захочет переслать данные для узла PC1, то как только он отправит пакет на коммутатор, он выучит его MAC адрес за портом 2 и PC1 с PC3 будут общаться друг с другом точечно.

Таким образом, рано или поздно, коммутатор выучит все MAC адреса за всеми портами. Учитывая, что компьютеры у нас практически всегда что-то пытаются отправить в сеть (винда хочет узнать погоду, приложения хотят обновиться и т. д.), таблица “MAC address table” заполняется довольно-таки быстро.



Теперь усложним нашу схему, предположим что у нас теперь есть второй коммутатор и он соединен с первым. Во втором коммутаторе есть ещё два компьютера, PC4 и PC5.

1. Если PC4 захочет пообщаться с PC1, он сформирует к нему пакет, где Destination MAC адрес будет MAC_1.
2. Второй коммутатор, как получит этот пакет, разошлет Unknown Destination unicast.
3. PC5 отбросит этот пакет, а первый коммутатор точечно направит этот пакет к PC1, информация то у него есть.

4. При этом коммутатор 1 внесет запись, что MAC_4 находится за портом 6.
5. Теперь предположим что и PC5 отправляет пакет компьютеру PC1. Пакет опять достигает 1 коммутатора, и он заносит в таблицу MAC_5 за порт 6.
6. Таким образом за портом 6 изучаются MAC адреса компьютеров за другим коммутатором. Можно сделать вывод, что если мы видим несколько MAC адресов за одним портом, то скорее всего там стоит коммутатор и за ним находятся несколько компьютеров.

Поведение, когда несколько MAC адресов находится за одним портом, это нормально, но когда один мак адрес находится за несколькими портами, это уже ненормально, такого на коммутаторе быть не должно.

Если у нас PC1 переезжает за коммутатор 2, то как только порт падает в shutdown в таблице стираются все MAC адреса за этим портом.

Если PC4 переезжает на коммутатор 1 и втыкается в порт 11, то когда пакеты PC4 начинают приходить с порта 11, в таблице MAC адресов коммутатора 1 начальная запись про MAC_4 удалится и появится новая запись, что MAC_4 теперь находится за портом 11.

При этом, я напоминаю, что в сети 2 одинаковых MAC адреса быть не должно, и не должно быть именно из-за такого поведения наших коммутаторов. Иначе строчки в таблице постоянно будут перезаписываться и хосты не смогут нормально работать.

У каждой записи есть время жизни, то есть если пакеты с определенным MAC адресом перестают ходить, то такая запись через какое-то время удалится. (Практически на всех коммутаторах это 300 секунд, это время можно менять).

Итак, подытожим, у коммутатора есть таблица “MAC address table”, которая хранит информацию за каким портом какие устройства находятся. Согласно этой таблице коммутатор пересылает пакет адресно конкретному устройству. Благодаря такому подходу в сети не флудятся

лишние пакеты, как например делает это хаб. Следовательно, у нас не могут возникать коллизии, если на портах full-duplex.

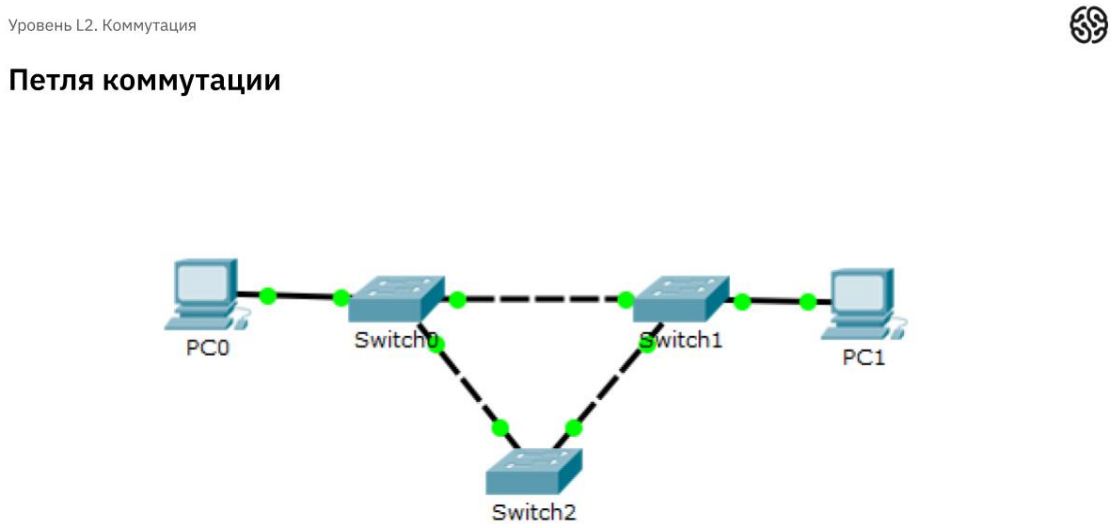
Теперь смотрите, до этого момента мы всегда исходили из того, что отправителю известен Destination MAC, который он вставлял в пакет. Но обычно изначально MAC адрес, к которому хочет обратиться узел, неизвестен. Чтобы его узнать, хост рассылает специальный широковещательный пакет, в котором он спрашивает какой MAC адрес у определенного компьютера, например PC1 ищет MAC адрес PC4. Это то самое бродкаст вещание, которые мы рассматривали на прошлом уроке. Задача коммутатора определить такой специальный пакет и размножить во все активные порты (кроме того порта, с которого он пришел).

В таком широковещательном пакете Destination MAC адрес состоит из всех битов равных единице, в шестнадцатеричном формате это 12 букв F. На нашей схеме, если коммутатор 1 получит такой пакет он его отправит во все порты, соответственно на второй коммутатор тоже прилетит этот пакет, и он тоже обязан отправить этот пакет дальше во все порты, и так далее.
Таким образом широковещательный пакет так или иначе разлетится до всех участников сети и собственно в этом его задача. Все компьютеры которые получат такой пакет, прочитают его и увидят, что кто-то ищет MAC адрес PC4 и проигнорируют его. Но как только сам компьютер PC4

поймет, что спрашивают его MAC адрес, он уже юникастом ответит пакетом, в котором будет содержаться его MAC адрес.

Только что я схематично обрисовал работу протокола ARP, address resolution protocol, но о нём более подробно мы будем говорить в конце. Пока важно понять что просто бывают такие вспомогательные служебные пакеты, которые являются широковещательными, и они выполняют определенную сетевую задачу. То есть они не несут какой-то полезной нагрузки. И коммутаторы обязаны их размножать во все активные порты.

Область сети, где распространяется такой широковещательный пакет, называется Broadcast Domain, или широковещательный домен, или L2 домен.



А теперь представьте, что мы имеем 3 коммутатора, и эти 3 коммутатора связаны между собой друг с другом топологией кольцо. Так обычно делают, чтобы у нас было минимальное резервирование на случай обрыва одной физической линии.

Что будет, если широковещательный пакет прилетит на один из портов коммутатора? Ну, первым делом он размножит его в остальные два порта, а затем обновит информацию в таблице MAC адресов, где запишет Source MAC за соответствующим портом. Те два коммутатора, которые получат

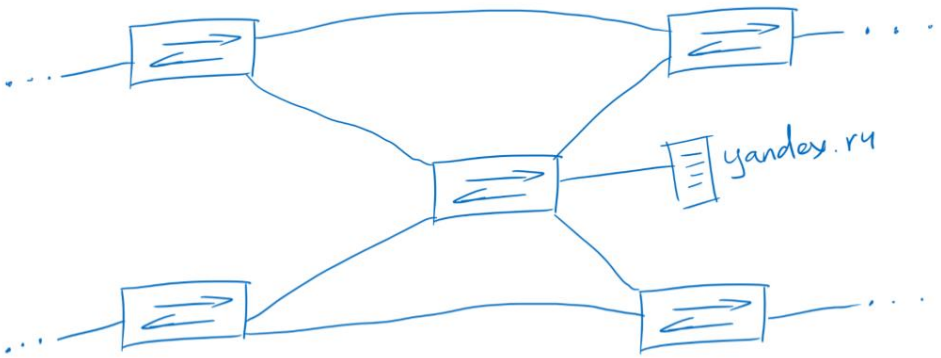
такие пакеты, тоже должны размножить его во все порты, и так далее до бесконечности. Количество таких пакетов в сети начнет увеличиваться и начнётся широковещательный шторм - страшный сон любого сетевого инженера. Обратите внимание, что каждый раз когда будет приходить такой пакет на порт, он не только будет просто множиться, из-за него будет переписываться MAC адрес в таблице “MAC address table”, так как у нас каждый раз один и тот же Source MAC адрес будет приходить с разных портов. Это тоже будет нагружать коммутатор, так как скорость чтения/записи в таблицу тоже не бесконечна.

В итоге мы придём к тому что все 3 коммутатора в какой-то момент времени будут заняты пересылкой одних таких широковещательных пакетов, и времени для обработки полезного трафика не останется совсем. А всё из-за того, что у нас есть петля коммутации.
Нам надо сделать так, чтобы петель не было. А где у нас нет петель? В какой топологии? Дерево! Чтобы топология была дерево, для этого есть специальный протокол Spanning Tree Protocol - STP. Не будем углубляться в подробности работы этого протокола, так как разработчики редко касаются деталей его настройки. Важно понять принцип, что этот протокол, блокируя определенные порты, превращает любую топологию в дерево.

Таким образом, если STP здесь, на нашей схеме, заблокирует один из линков, у нас броадкаст пакеты не смогут породить шторм, они просто разойдутся по сети и достигнут каждого узла в сети. При этом STP ещё носит задачу по резервированию каналов связи. Если один из зелёных линков сломается, STP сможет перестроить топологию в другое дерево, разблокировав один из линков.

Важно отметить, что это сторонний протокол, то есть защиты от петель в самом протоколе Ethernet не существует, и нам необходим сторонний протокол, который настраивается на каждом отдельном коммутаторе. Благо производители позаботились, и включают STP по дефолту. Запомните, что его не надо отключать без необходимости, иначе мы можем породить броадкаст шторм.

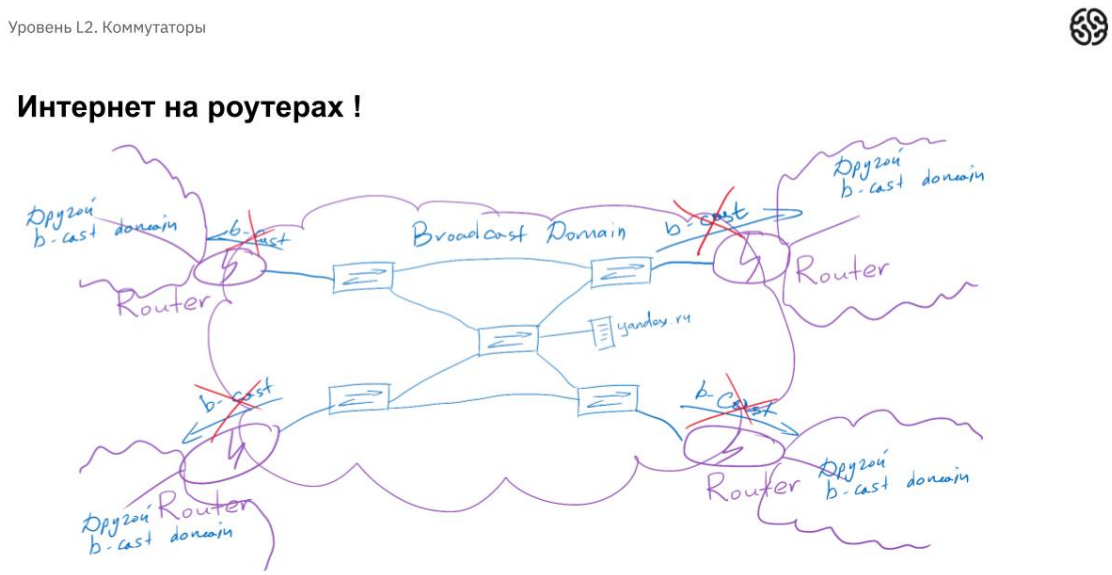
Интернет на коммутаторах ?



Итак, на данном этапе у нас есть Ethernet, с помощью которого мы можем строить большие сети. У нас есть средство идентификации оборудования в сети - MAC адреса. И мы можем строить сеть, в которой каждый участник может общаться с любым другим участником благодаря коммутатору, который изучает все MAC адреса.

Насколько большую такую сеть мы можем построить? Предположим, что перед нами стоит задача построить глобальную сеть Интернет. Можем ли мы её построить на чистом L2? Давайте пофантазируем, что вся сеть Интернет построена на коммутаторах. Тогда, если я начну искать yandex.ru, как Маша искала Петю, я разошлю свой бродкаст пакет на свой ближайший свич, этот свитч разошлет бродкаст и дальше, те ещё дальше, и так далее. В итоге этот бродкаст разошлется по всему миру. А таких как я - много, и все те, кто будет искать yandex.ru, будут рассылать свои броадкаст пакеты. А ещё все пользователи интернета будут искать другие сайты, и по итогу получится, что вся сеть Интернет будет наводнена такими бродкаст пакетами. Более того, все будут знать кто что ищет, и кто с кем общается. Я уже не говорю, каких размеров должны быть таблицы MAC адресов, и как быстро должен отрабатывать всемирный STP, чтобы не было всемирного броадкаст шторма…

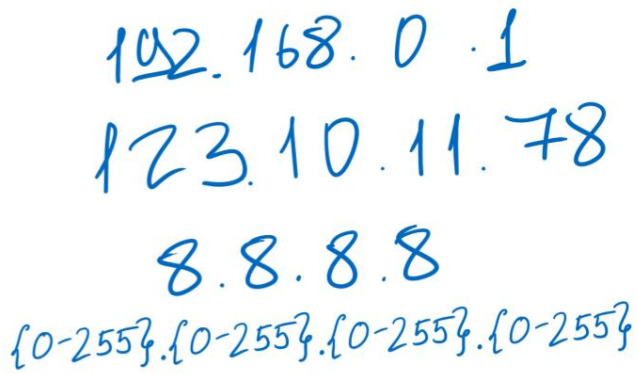
Помимо этой проблемы, у L2 нет гибкости. Все адреса по умолчанию выдаются производителем, а хотелось бы тоже как-то сгруппировать эти адреса, как-то ими управлять, разрешать одним группам ходить в одни места, другим группам ходить в другие места. К сожалению один L2 с этими задачами не справляется.



Тут нам нужен новый слой адресации - L3, который позволяет делать отдельные сети, как группы узлов, а также нужно отдельное устройство, которое режет бродкаст рассылки, и которое позволяет передавать пакеты из сети в сеть. Основным протоколом на L3 в стеке TCP/IP является IP - Internet Protocol. Адресация называется IP-адресация, а устройство называется роутер или маршрутизатор.

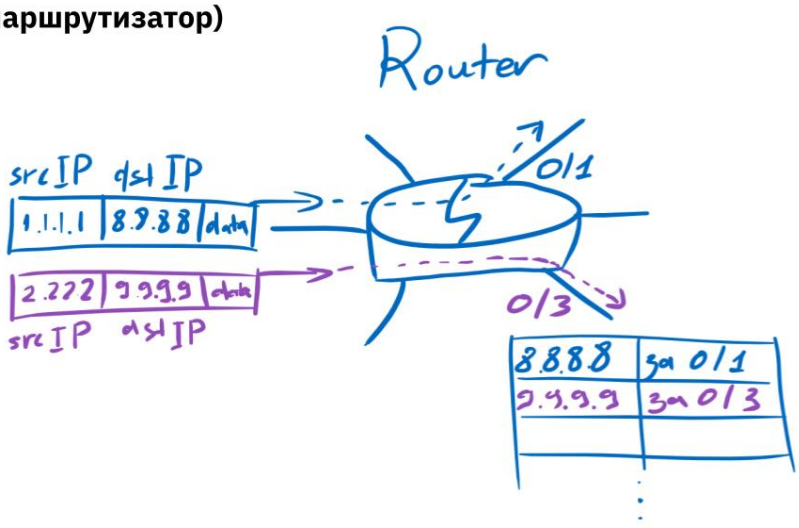
Давайте вкратце познакомимся с основными участниками L3 уровня. Сначала поговорим что такое IP адрес, затем про принцип работы маршрутизатора, а потом внимательно посмотрим что такое IP сеть.

IP адреса



IP адрес - это специальный адрес длиной 32 бита, который уникально идентифицирует компьютер в сети. Он задаётся уже не на заводе производителе сетевых карт, а администратором сети. IP адрес записывают в формате четырех чисел, разделенных точкой, где каждое число лежит в диапазоне от 0 до 255 (что занимает ровно 8 бит). Когда компьютер создает пакет для отправки по сети, на уровне L2 он заполняет src и dst MAC адреса, а на уровне L3 он заполняет src и dst IP адреса.

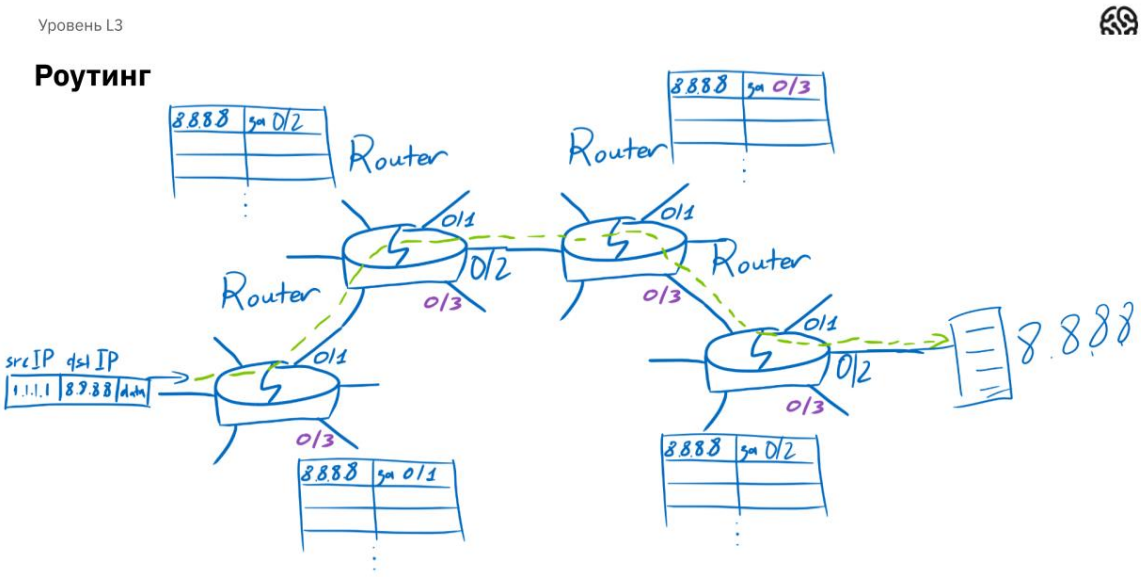
Роутер (маршрутизатор)



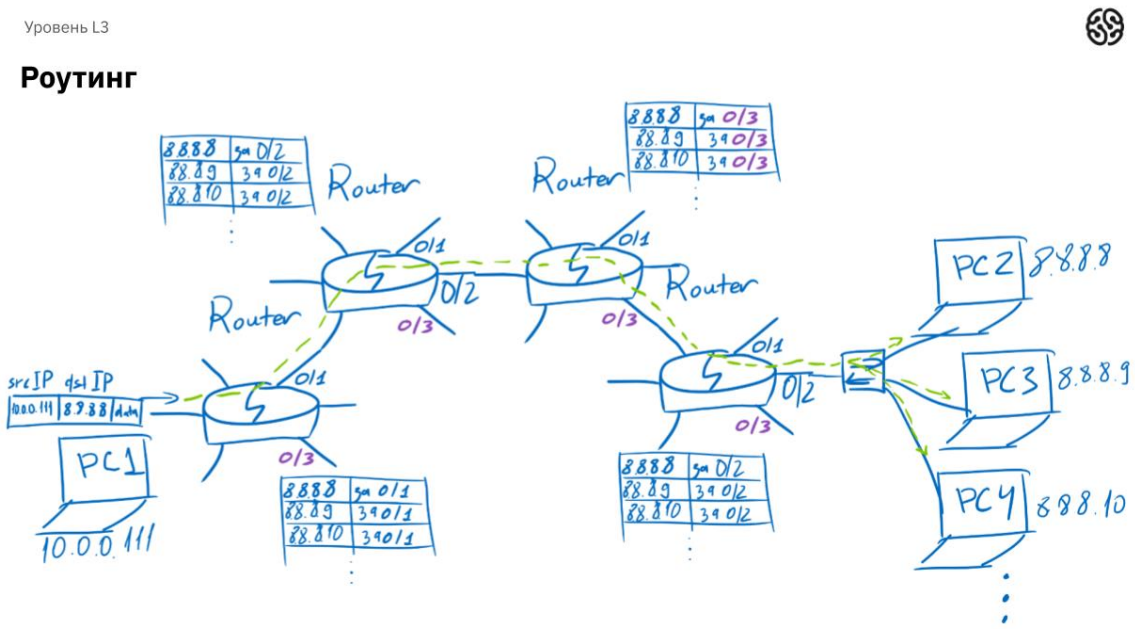
Роутер, он же маршрутизатор - это устройство, главная задача которого перенаправлять пакеты между сетевыми интерфейсами. Он смотрит на все входящие пакеты, в пакете считывает Destination IP адрес, ищет у себя в специальной таблице куда по этому Destination IP надо направить пакет, и отправляет дальше в сеть. На каждом интерфейсе роутера назначается свой IP адрес.

Таблица заполняется кем-то (или чем-то) заранее и называется таблица маршрутизации (или routing table). Таблица состоит из двух столбцов, в первом информация о сети куда летит пакет, во втором столбце интерфейс, куда этот пакет надо перенаправить. Пока сделаем допущение, что в таблице маршрутизации у нас в первом столбце прописывается конкретный IP адрес конкретного узла.

Предположим, что у нас компьютер отправляет пакет в котором Destination IP адрес 8.8.8.8. Когда такой пакет приходит на роутер, то он у себя в таблице маршрутизации ищет запись что 8.8.8.8 лежит через интерфейс 0/1, если пакет приходит в котором Destination адрес 9.9.9.9, то в таблице маршрутизации должна быть запись что 9.9.9.9 лежит через интерфейс 0/3. Если в пакете Destination адрес 1.1.1.1, а в таблице маршрутизации записи с таким IP нет, то такой пакет просто уничтожается. В этом одно из основных отличий роутера от коммутатора. Коммутатор старается разослать пакет с неизвестным MAC адресом во все порты, роутер же действует жестче: нет записи - нет пакета. Еще одно главное отличие, в том, что роутер приняв броадкаст пакет не распространяет его дальше во все порты, а обрабатывает его как обычный конечный хост.



Если роутеров между отправителем и получателем много, то в таблице маршрутизации каждого роутера должна быть соответствующая запись, только в таком случае пакет пройдет всю цепочку роутеров. Главное надо запомнить, что основная задача роутера перенаправить пакет, основываясь на Destination адресе в его IP заголовке.



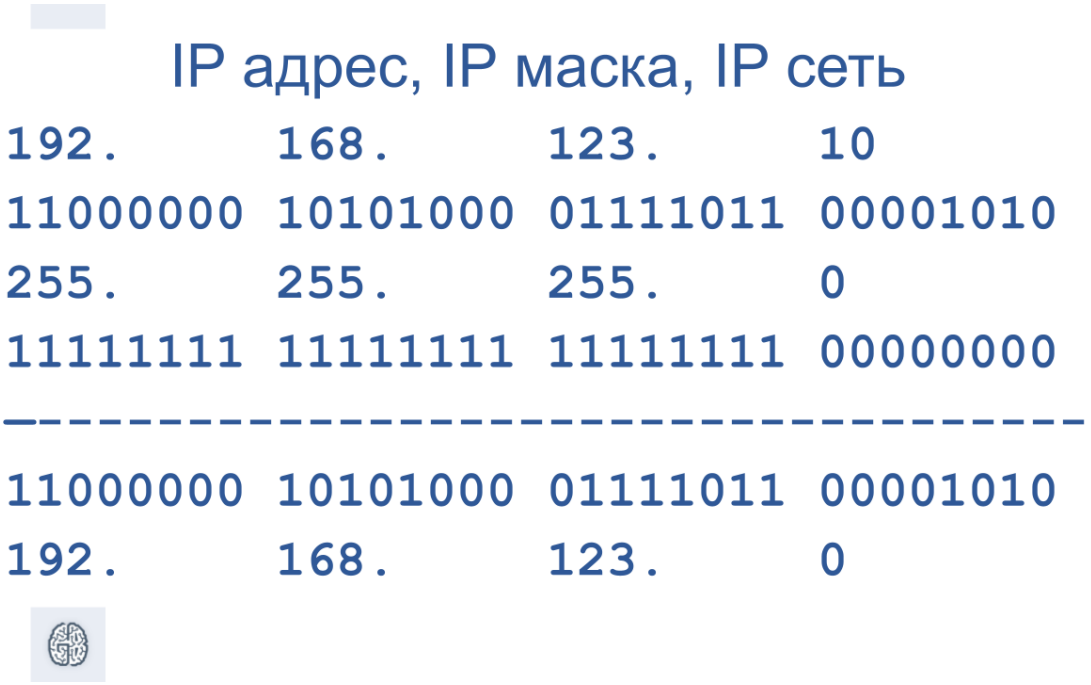
Теперь давайте посмотрим на следующую схему, у нас есть узлы PC1 и PC2-PC4. Узлы PC2-PC4 находятся за одним и тем же маршрутизатором. Компьютер PC1 с адресом 10.0.0.111 направляет пакет на PC2 с адресом 8.8.8.8. Для того, чтобы такой пакет дошел до получателя, как я уже

говорил ранее, в каждом маршрутизаторе должен быть маршрут на адрес 8.8.8.8. Если компьютер PC1 захочет направить пакеты на адрес 8.8.8.9 на PC3, то такой маршрут тоже должен быть на каждом маршрутизаторе. Если он потом захочет направить пакет на адрес 8.8.8.10, то опять таки на всех роутерах должна быть информация про адрес 8.8.8.10. И так далее. Хотя обратите внимание, что компьютеры 8.8.8.8, 8.8.8.9 и 8.8.8.10 лежат за одним и тем же роутером. Эффективно ли делать маршрут про каждый IP адрес, если маршруты ведут к одному и тому же роутеру? Учитывая, что IP адресов может быть много (максимальный предел это 255*255*255*255=4 228 250 625), то поиск по таблице с большим числом IP адресов будет не очень эффективным. Чтобы сделать поиск по таблице маршрутизации проще, были придуманы IP сети. Если по-простому, IP сеть - это просто некий диапазон IP адресов. В нашем случае, вместо трёх записей в таблице маршрутизации, мы можем сделать одну, что 8.8.8.8-8.8.8.10 лежат через IP адрес 1.1.1.1.

В таблице маршрутизации именно в столбце “Куда летит пакет” записана информация о IP сетях, а не о каждом конкретном IP адресе. Во-первых, это делает таблицу маршрутизации более простой, а следовательно и поиск по ней более быстрый, во-вторых, мы можем сгруппировать определенные IP-адреса под различные задачи. Например, одна IP сеть это - сеть бухгалтеров, другая IP сеть - это сеть администраторов. Далее, задав политики на таких сетевых устройствах, как файерволы, мы можем определить, что администраторам можно ходить куда угодно, а бухгалтерам можно ходить только на определенные сайты, например на consultant.ru и 1С, при этом описав целую группу, а не каждый отдельный IP адрес компьютера каждого сотрудника. Группировка IP адресов по IP сетям даёт нам большую гибкость и удобство при построении и планировании сетей.

Итак, давайте более подробно рассмотрим что такое IP сеть. Специфика маршрутизаторов в том, что они с огромной скоростью должны обрабатывать входящие пакеты, т.к. они тоже могут лететь с огромной скоростью, несколько десятков тысяч в секунду. Отметим, что производить поиск по рандомному диапазону IP адресов для процессора довольно не тривиальная задача. Поэтому инженеры пошли на некоторое

ухищрение, и придумали записывать IP сеть специальным образом в виде адреса самой IP сети и маски IP сети. Давайте разберемся как это работает.



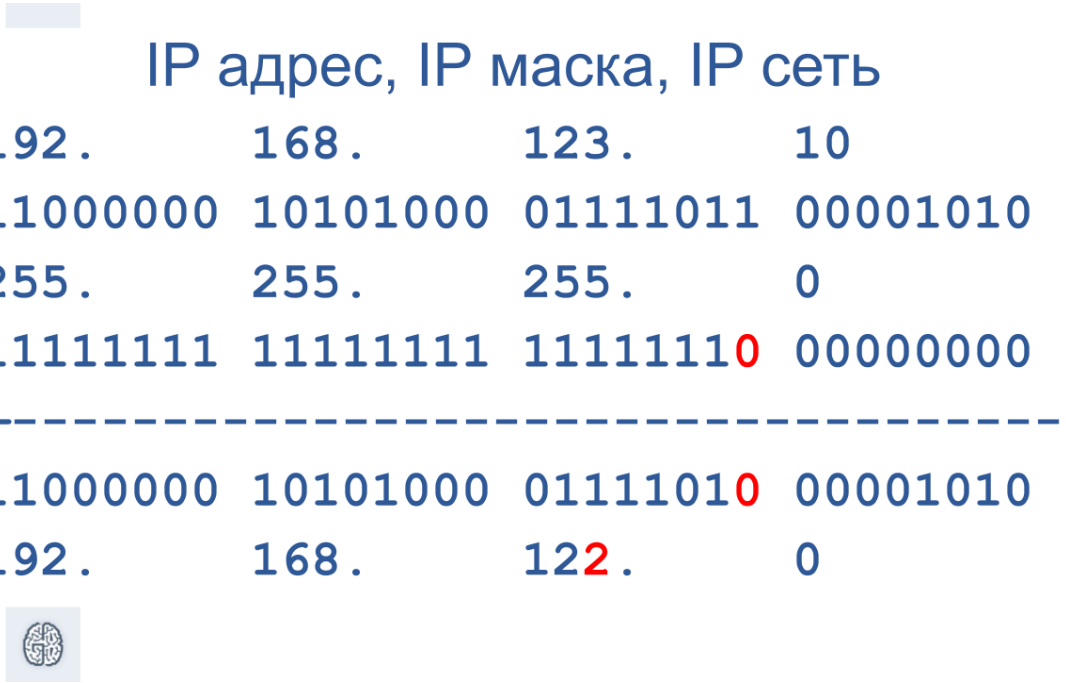
Посмотрите на слайд, предположим, что у нас есть адрес компьютера 192.168.123.10 и маска у этого компьютера 255.255.255.0. Вы наверное ранее могли встречать такую маску, так вот, что же она означает? Если мы переведем IP адрес из десятичного вида в двоичный, то мы получим следующее значение 11000000 10101000 01111011 00001010, в то время как маска представляет из себя набор единиц, а потом последующих за ними нулей, в нашем случае будет 11111111 11111111 11111111 00000000. Обратите внимание, что маска - это всегда сначала только единицы, а потом только нули, между единицами не может быть нулей, и между нулей не может быть единиц. Следовательно масок может быть всего 32. Маску в более коротком виде так и записывают /21 /23 /28 - по количеству единиц.

Так вот, если мы мы умножим побитно IP адрес на маску, то там где у маски единицы, мы будем получать те же самые значения что в IP адресе, а там где у Маски нули, всё будет обращаться в нули. Таким образом мы получаем адрес IP сети, которой принадлежит компьютер.

Если мы теперь возьмем второй компьютер с адресом 192.168.123.11 с такой же маской, то мы получим то же самое значение адреса IP сети. Следовательно эти два компьютера будут лежать в одной IP сети.

Вообще получается, что какой бы мы адрес в последнем октете не брали, то там всегда будет 0 и всегда будет один и тот же адрес IP сети. Таким образом у нас появляется диапазон значений в этой IP сети от 192.168.123.0 до 192.168.123.255.

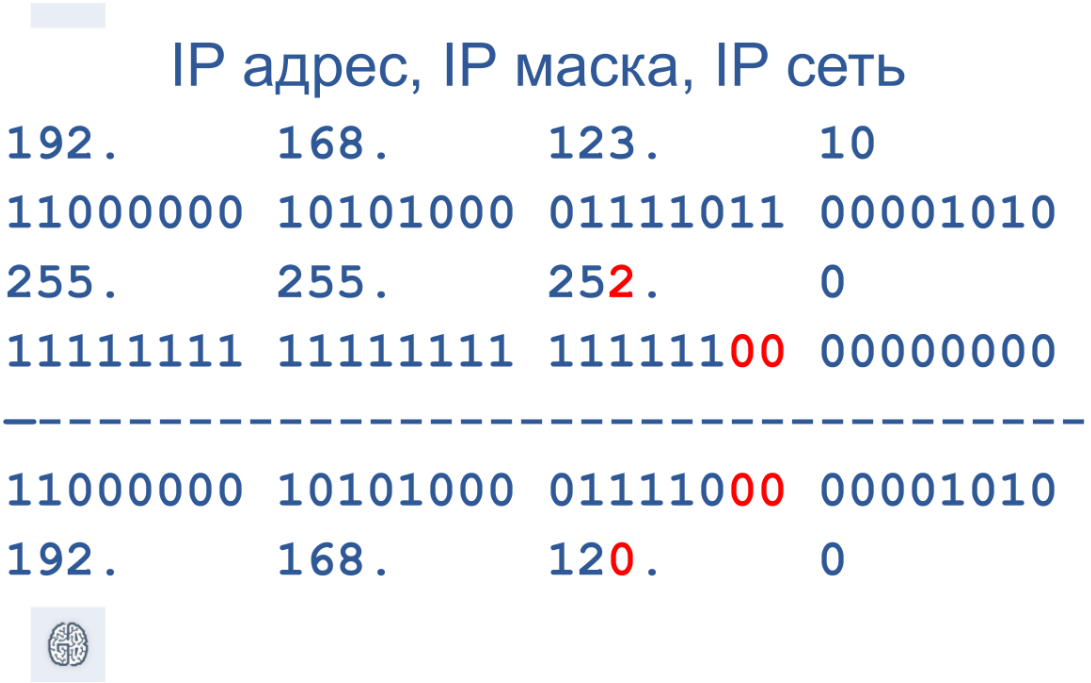
Вот в виде таких диапазонов на маршрутизаторах и задается маршрут в какую-либо сеть, в виде адреса IP сети и маски. Поиск по такой записи будет происходить практически мгновенно, потому что в роутерах есть особая память со специальным доступом к ней, но хранить информацию в ней можно только в таком виде. К сожалению это накладывает ограничение, что мы не можем задавать любые диапазоны IP адресов которые нам вздумаются, мы можем задавать только диапазоны кратные двум.



Давайте посмотрим как работает механизм увеличения и уменьшения IP сетей. В нашем случае в диапазоне всего 256 возможных значений адресов, где первый и последний адрес зарезервированы. Первый адрес - это адрес сети, его нельзя назначать на хосты, последний адрес это так

называемый IP бродкаст, его тоже нельзя назначать на хосты. Итого у нас остаётся всего 256-2=254 адреса для назначения их на хосты.

А теперь предположим, что у нас дата-центр, и у нас в одной IP сети должно быть 500 серверов. Для этого нам надо всего лишь украсть 1 бит из маски, таким образом под IP адреса хостов у нас будет целых 9 бит, а следовательно в два раза больше всевозможных значений, то есть 512. Опять-таки первый последний адрес зарезервированы поэтому у нас остается 510 возможных IP адресов под наши сервера. При этом обратите внимание, что адрес 192.168.123.0/24 мы можем использовать как адрес хоста на нашем сервере.



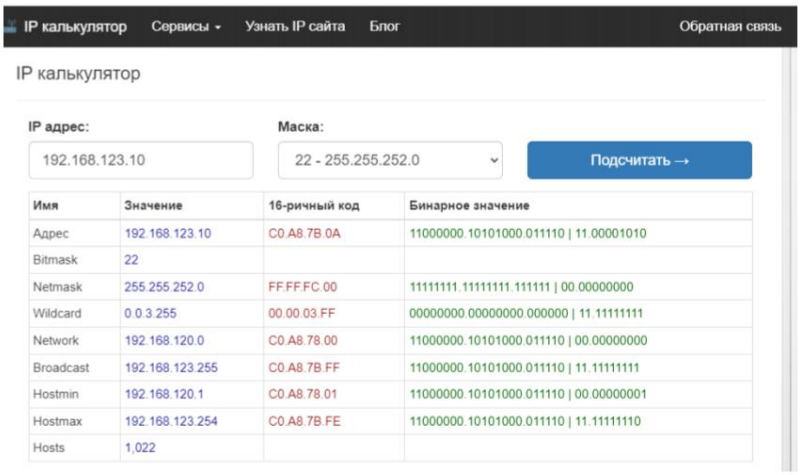
Если нам надо ещё больше, мы снова отнимаем у нашей маски ещё одну единичку, и следовательно увеличиваем сеть ещё в 2 раза больше. Таким образом у нас будет уже 1024 всевозможных значений, минус 2 адреса, и получаем 1022 адреса под наши сервера. Также работает и в обратную сторону, если нам не нужны такие слишком большие IP сети, мы можем увеличивать количество единичек в нашей маске и уменьшать сеть в два раза.

Важно отметить, что все IP адреса предопределены математически, то есть мы не можем взять рандомный IP адрес с какой-либо маской, например мы не можем взять 192.168.121.0/22 и объявить что это IP сеть, т.к. это будет

IP адрес хоста в сети 192.168.120.0/22. IP сеть надо обязательно всегда вычислять.

Уровень L3

IP адрес, IP маска, IP сеть

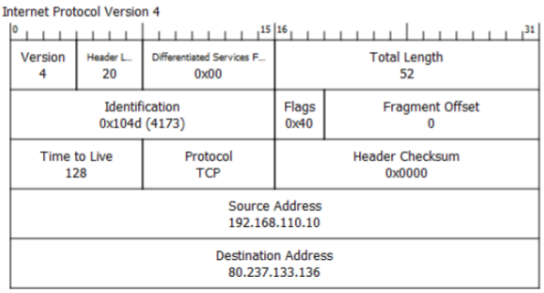


Вычислять каждый раз вручную таким образом IP адреса нет необходимости. Для этого уже придуманы специальные IP калькуляторы, в которые мы просто вбиваем свои исходные параметры IP адреса/маски и получаем необходимый расчет. Как ими пользоваться более подробно мы разберем на семинаре.

Уровень L3

Internet Protocol (IPv4)

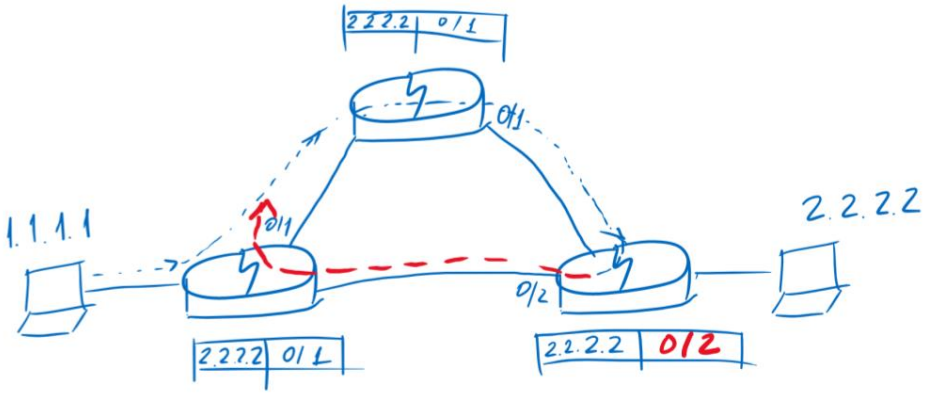
Internet Protocol (IP, Интернет протокол или межсетевой протокол) является маршрутизируемым протоколом сетевого уровня. На основе протокола IP работает большинство современных сетей.



Давайте посмотрим теперь на заголовок IP пакета.

- Первые 4 бита отводятся под версию протокола IP. В нашем случае это IP версии 4. Еще существует IP версии 6, в котором у нас IP адрес занимает 128 бит вместо 32-ух.
- Дальше у нас идёт header length - это длина заголовка IP пакета в 32-битных “словах”. То есть по сути это количество вот таких 32-битных строк. Обычно их 5.
- Следующие 8 бит служат для приоритезации трафика и уведомления о перегрузках на маршрутизаторе. Про них мы особо ничего говорить не будем, так как это отдельная тема в сетевом инжиниринге.
- После этого идёт длина идёт длина пакета в байтах, включая заголовок и сами данные.
- Следующие 3 поля служат для правильной сборки фрагментированных пакетов. Напомню, что фрагментация у нас получается, когда различаются MTU на входном и выходном интерфейсе. Здесь тоже особо заострять внимание не будем.
- Следующее поле “Time To Live”, и он нам помогает бороться с петлями на уровне L3. Поговорим об этом чуть попозже.
- Далее следует поле, в котором находится информация о протоколе, который находится дальше, в заголовке L4. Помните, мы рассматривали, что обязательно в каждом заголовке должен быть указан нижележащий протокол, для более эффективной и более быстрой обработки пакетов. Здесь поле протокол играет именно эту роль.
- После этого идёт контрольная сумма заголовка. По аналогии с контрольной суммой, которая делается в Ethernet, она также защищает заголовок IP от помех.
- После всего идёт IP адрес отправителя и IP адрес получателя. Зачем они, мы уже знаем.

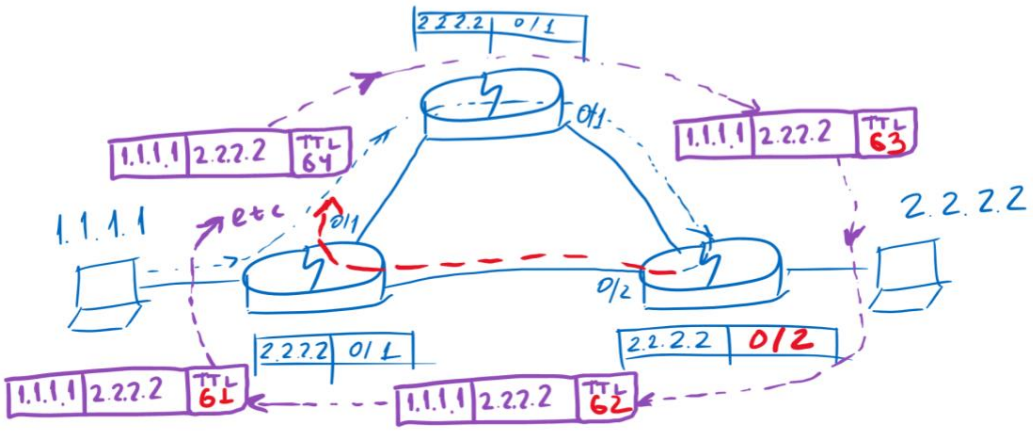
Петли. TTL



Давайте теперь поговорим про петли и механизм борьбы с ними. Представим, что у нас есть три роутера, соединённых между собой друг с другом. Если в каждом роутере есть маршрут в сеть X.X.X.X, который указывает на соседний роутер, то пакет, в котором IP destination принадлежит этой сети, вечно бы ходил между всеми этими тремя роутерами. Если таких пакетов будет много, они загрузят нашу сеть, а роутеры занимались бы бессмысленной работой по пересылке одних и тех же пакетов. Если в Ethernet у нас был вспомогательный протокол Spanning Tree Protocol, то здесь механизм борьбы с петлями уже встроен в сам протокол IP. Благодаря полю Time To Live у нас ограничено время жизни пакета. Но это время жизни измеряется не в секундах. Это время в жизни измеряется в “прыжках” пакета между роутерами - в хопах.

Уровень L3

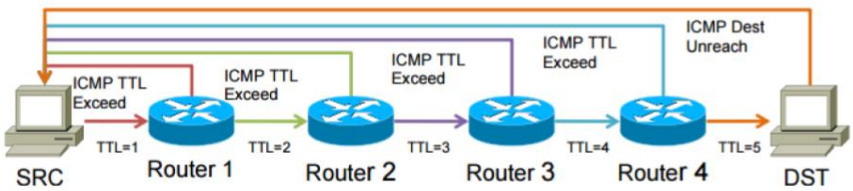
Петли. TTL



Операционная система компьютера, который посылает пакет, устанавливает это время жизни в определённое значение, например Linux, MacOS, Android устанавливает значение 64. Windows устанавливает 128. Каждый раз когда пакет “перепрыгивает” с роутера на роутер, у него вычитается единица из поля Time To Live. И как только значение там становится нулевым, пакет уничтожается, а отправителю пакета отправляется сообщение, что его пакет уничтожили в связи с тем что Time To Live истёк.

Уровень L3

Утилита tracert

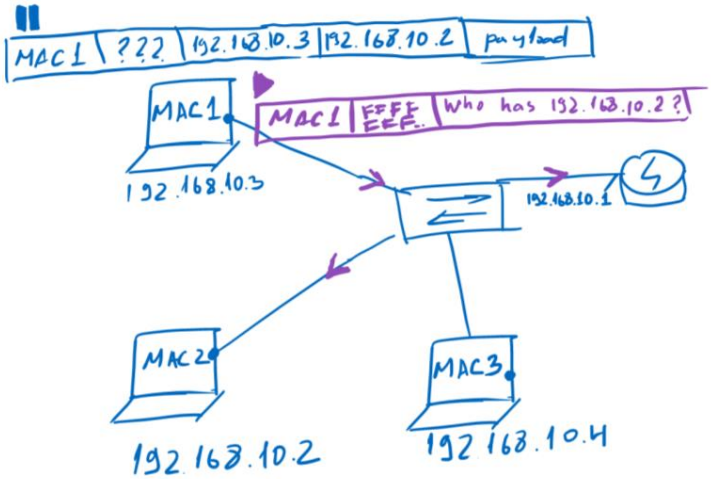


Благодаря такому механизму работает специальная программа tracert (в разных ОС она может называться по разному, но смысл не меняется). Она позволяет искать проблемы на сети. Её задача вывести нам список маршрутизаторов от одного узла, с которого мы запускаем эту программу, до другого узла. Как она работает? Очень просто, программа создает пакет, в котором заполняет Source IP и Destination IP. Но поле Time To Live сначала устанавливается в единичку. Такой пакет, дойдя до первого же роутера, уничтожается, и роутер 1 обратно отсылает сообщение что он убил наш пакет, при этом в этом сообщении Source IP будет IP адрес ближайшего к PC1 интерфейса. PC1 аккуратненько запишет этот IP и отправит еще раз этот же пакет, но с Time To Live равным 2. Такой пакет уже пройдёт первый хоп и уничтожается на втором хопе. 2 роутре сделает то же самое, что и первый, отправит сообщение что он уничтожил пакет, в этом сообщении Source IP адрес будет уже равен IP адресу интерфейса, ближайшего к PC1. Когда такое сообщение дойдёт PC1 он его аккуратненько запишет после первой записи. И так далее, пока пакет не дойдет до конечного получателя.

Если у нас где-то есть разрыв линии связи или неправильные настройки маршрутизации, то в какой-то момент при увеличении Time to Live мы не будем получать ответы и у нас пойдут вот такие звёздочки. Значит мы знаем что до последнего хопа, который нам ответил, все хорошо, а вот дальше надо чинить.

Теперь давайте поговорим про взаимодействие уровня L2 с L3. Как согласуются IP адреса и MAC адреса в отправляемых в пакетах? Во-первых учтем, что Destination IP адрес всегда должен быть нам известен, он всегда связан с каким-то ресурсом, к которому мы обращаемся. Например, если мы идём на yandex.ru, мы сначала спросим у специального DNS сервера какой IP адрес у yandex.ru. Уже после этого мы его вставляем в пакет и отправляем этот пакет на сам сервер yandex.ru. Про систему DNS мы будем говорить немного позже, сейчас важно понять что именно Destination IP адрес должен быть нам известен.

ARP



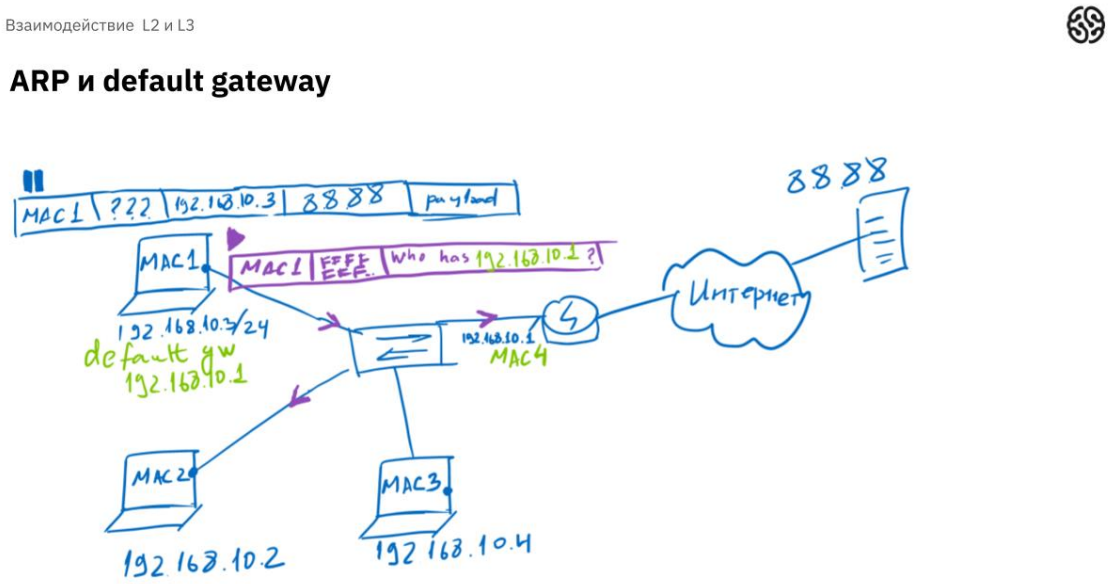
Итак, рассмотрим ситуацию, когда компьютер хочет отправить какое-то сообщение соседнему компьютеру в своей же сети:

- он формирует какой-либо payload,
- навешивает заголовок IP, в котором он вставляет свой IP адрес как Source IP,
- вставляет Destination IP адрес ресурса, которому payload предназначен
- далее в заголовок L2 вставляет в Source MAC адрес своей сетевой карты
- а в Destination Mac Address он пока не знает что вставлять. Как быть?

Он ставит отправку этого пакета на паузу, и формирует специальный широковещательный пакет, в котором на всю свою L2 сеть спрашивает: “Какой MAC адрес у хоста с IP адресом 192.168.10.2?”. Как мы помним широковещательный пакет распространится по всей по всему L2 бродкаст домену. Он прилетит на наш роутер, на котором настроен адрес 192.168.10.1, и роутер его проигнорирует, так как спрашивают не его. Пакет прилетит и на 192.168.10.4, который тоже его проигнорирует. И рано или поздно пакет прилетит на хост 192.168.10.2. Хост поймёт, что спрашивают его MAC адрес и уже юникастом ответит: “192.168.10.2 сидит за MAC адресом MAC2”. Получив такой ответ PC1 возобновит

формирование пакета, который он поставил на паузу. Вставит туда полученный MAC адрес и уже юникастом отправит его на PC2.

То, что я описал - это принцип работы протокола ARP - address resolution protocol - протокол сопоставления адресов MAC и IP. Протокол позволяет в своём L2 броадкаст домене узнавать MAC адреса своих соседей и общаться с ними.



Теперь предположим, что у нас компьютер хочет послать пакет на 8.8.8.8, который лежит за роутером где-то там в интернете. Сравнивая свой IP и свою маску с 8.8.8.8, он понимает что адрес 8.8.8.8 лежит не в своей IP сети. Искать MAC адрес 8.8.8.8 широковещательно смысла нет, надо сразу отправлять этот пакет на роутер, а он там разберется. Значит компьютеру надо знать IP адрес своего роутера, который называется еще шлюз по умолчанию или default gateway. Что будет делать компьютер? Он сформирует пакет, в котором:

- Destination IP address будет 8.8.8.8,
- Source IP адрес будет свой IP адрес,
- Source Mac Address MAC адрес своей сетевой карты,
- а Destination Mac Address опять надо узнавать через протокол ARP, и узнавать его надо для шлюза по умолчанию.

После того как MAC шлюза по умолчанию будет известен можно отправлять пакет на 8.8.8.8. Важно отметить, что когда такой пакет придёт на роутер, в Destination MAC адресе будет MAC адрес интерфейса роутера, роутер:

- удалит заголовок L2 у этого пакета,
- посмотрит таблицу маршрутизации,
- поймет в какой интерфейс дальше направлять пакет,
- в этот интерфейс также с помощью ARP узнает MAC адрес следующего роутера
- навесит новый L2 заголовок заголовок, с Source MAC своего интерфейса, в Destination MAC будет MAC адрес следующего роутера
- и так будет происходить по всей цепочке, пока наш пакет не дойдет до 8.8.8.8.

Итак, подведём итог. Сегодня мы:

- до конца поговорили про L2 уровень,
- познакомились с таким устройством как свитч, поняли как он устроен, как он работает, как решает проблему точечной доставки пакета адресату
- начали изучать L3 уровень и познакомились с таким устройством как роутер, поняли его основной принцип работы
- поняли чем отличается роутер от свича,
- также мы узнали, что L2 сети нельзя растягивать до бесконечности, эти сети должны быть ограничены как раз L3 устройствами, роутерами, которые уже соединяют такие вот L2 сети между собой, маршрутизируя трафик.
- для того, чтобы вместе работали эти два слоя адресации есть протокол согласования между ними - протокол ARP

На следующем уроке мы поговорим, как настраивать ту самую маршрутизацию вручную. Посмотрим на протоколы, которые позволяют заполнять таблицы маршрутизации автоматически. А также познакомимся с технологией VLAN, которая позволяет разделять L2 броадкаст домены на несколько.

А вот чем мы займемся на ближайшем семинаре:

- будем изучать настройки коммутатора,
- изучим как настраивать роутер,
- также мы будем считать IP сети и поработаем с IP калькулятором,
- также посмотрим на заполнение таблиц MAC адресов, таблиц маршрутизации, таблиц ARP.
- А также познакомимся с протоколом DHCP, который позволяет автоматически назначать IP адреса и не настраивать их вручную. Это очень полезный протокол, особенно когда нам надо настроить множество устройств в нашей сети.
- также поговорим про железную составляющую роутеров, посмотрим что такое модульность, и поймем, как можно наши роутеры Cisco кастомизировать
- посмотрим что такое sfp модули и тоже с ними поработаем.

Так что приходите, будет интересно, до встречи на семинаре!