Базовые термины

Сначала разберем термины, которые могут вам встречаться в документации,

**Link** (линк) – это просто соединение между двумя компьютерами. Чаще всего — обычный провод: витая пара (рис. 1) или оптоволокно.



Рис. 1. Витая пара

**Ethernet**. Произносится как 'И́*зэрнэт*' (ударение на первой букве "И").

Это технология, которая позволяет компьютерам соединяться в локальную сеть (LAN) и обмениваться данными с помощью проводов (витая пара или оптоволокна).

**LAN** (Local Area Network) — сеть, объединяющая устройства, расположенные на территории в масштабах от одной комнаты до города. Обычно LAN принадлежит одному физическому или юридическому лицу.

**Сетевой адаптер**(или **сетевой интерфейс**, или сетевая карта) – разъём в компьютере, куда втыкать кабель от интернета. Контролируется операционной системой (рис. 2).



Рис. 2. Сетевой интерфейс

**Топологии сетей** – схема соединения устройств сети. Как мы объединяем компьютеры в сеть с помощью проводов.

Пример самой простой топологии на рис. 3, когда мы просто соединили два компьютера одним проводом.

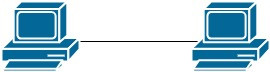


Рис. 3. Топология "точка-точка" (point-to-point)

А теперь вопрос: ***как соединить в одну сеть много компьютеров?*** Все они должны иметь связь (через кабель) друг с другом. На компьютерах в нашей схеме будет по одному сетевому интерфейсу (разъем, куда можно воткнуть кабель).

Не будем углубляться в другие виды топологии и перейдем сразу к популярной схеме "Звезда" (рис. 4.)

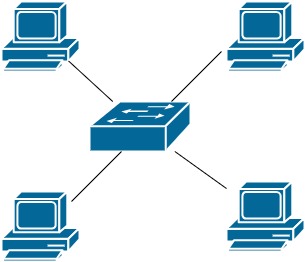


Рис. 4. Топология "звезда" (star)

Прекрасно, все компьютеры соединены. А что в центре рис. 4? Это **коммутатор**.

**Коммутатор** (switch – свитч) – получает сигнал от компьютера и пересылает сигнал другому компьютеру, которому этот сигнал был предназначен.

Можно представить, что коммутатор – это умный переходник. В нем есть много разъемов (рис. 5), в которые подключаем кабели от компьютеров. Современные коммутаторы работают сразу из коробки. Достаете, подключаете к ним провода – и все работает (почти все... дальше станет понятно, почему не все бывает так гладко изначально).

На рис. 6. показан простой домашний коммутатор.



Рис. 5. Коммутаторы.



Рис. 6. Домашний коммутатор

Коммутатор – это устройство **канального уровня**. Пока не надо акцентировать внимание, что это за уровни. Ближе к концу лекции мы их разберем. Сейчас надо просто сделать пометку в уме, что коммутатор понимает, к какому компьютеру перенаправить сигнал.

И нам никто не запретит строить сложные схемы из коммутаторов, как на рис. 7.

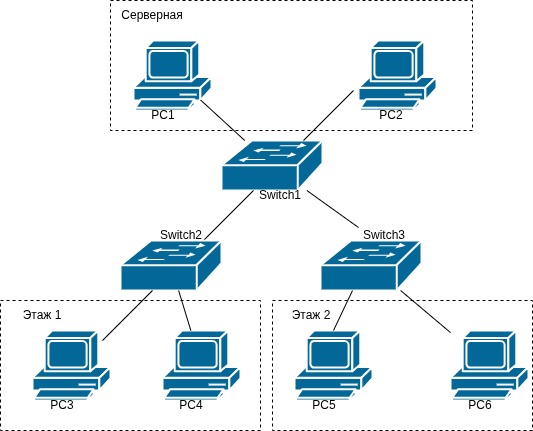


Рис. 7. Усложняем схему нашей сети

Тут появляется следующий вопрос: ***Как коммутатор понимает, кому пересылать сигнал?***

Мы хотим отправить сообщение с PC3 на PC1. Сигнал попадет на Switch2, с него на Switch1, а с него на PC1. Но почему не на PC2? Или не на PC5 через Switch3.

Перед ответом разберем еще один термин – MAC-адрес.

**MAC** (читается, как "мак") (Media Access Control)**–** уникальный "номер" сетевого устройства.

MAC-адреса уникальны для устройств и задаются производителем. Он уникален, как СНИЛС гражданина.

Пример: F8:63:3F:28:4E:CE

--

По мак адресу можно определить производителя. Например, начало на A8:BB:CF – это Apple, F8:63:3F – Intel.

--

**ВНИМАНИЕ!** MAC-адрес можно изменить вручную.

*Не то чтобы замена MAC-адреса была для нас важна. Но это надо понимать с точки зрения безопасности. Злоумышленник может подменить MAC-адрес на своем устройстве внутри сети и получить сообщение, которое предназначалось другому устройству.*

Закончили с MAC-адресом и вернемся к предыдущему вопросу: ***Как коммутатор понимает, кому пересылать сигнал?***

У коммутатора есть таблица коммутации.

**Таблица коммутации** – это "память" коммутатора, где он хранит список, кто к какому порту подключён.

|  |  |
| --- | --- |
| Порт | MAC-адрес |
| 1 | AA:BB:CC:11:22:33 |
| 2 | 11:22:33:44:55:66 |
| 3 | FF:EE:DD:00:11:22 |

Когда мы отсылаем запрос с PC1, то мы формируем на канальном уровне (опять упомянули этот уровень, но не будет пока говорить про него подробно) сообщение, которое называется **кадр**.

В кадре есть:

* MAC-адрес **получателя**
* MAC-адрес **отправителя**

Изначально эта таблица коммутации пустая. Когда мы делаем запрос через коммутатор, то он:

1. Читает из кадра мак адрес получателя.
2. Сопоставляет его с портом, по которому пришел запрос.  
   Так таблица заполняется.

Отлично. Мы сделали запрос с PC1. Switch 2 на рис.7 теперь знает MAC-адрес PC1. Но как он понимает, куда отправить запрос дальше?

Сейчас скажем, что дальше коммутатор делает broadcast (широковещательный) запрос по специальному MAC-адресу - FF:FF:FF:FF:FF:FF (разобрали его чуть выше). Мы:

Опросим все устройства связанные с этим коммутатором.

Получим от них ответы, в которых будет указан их MAC-адрес.

Заполним таблицу коммутации.

Прекрасно. Теперь мы связали все устройства внутри локальной сети.

А что дальше? **Что такое локальная сеть? Как общаться между несколькими локальными сетями?**

Перед ответом разберем очень важное понятие – IP.

**IP** (Internet Protocol) – это протокол сетевого уровня (опять говорим про уровни, но что это за уровни – разберем позже). У него 2 версии:

* IPv4
* IPv6

Пример IPv4: 192.168.1.100

Данные, которые пересылаем по этому протоколу называются – **пакет**.

IP-адрес устройства – это ЛОГИЧЕСКИЙ адрес.

MAC-адрес – физический. Его присваивает производитель и он живет с устройством всю жизнь. Если отключаем ноутбук от провода в офисе, приносим его домой, подключаем в домашнюю сеть, то коммутатор (дома обычно роутер – это тоже затронем чуть позже) увидит тот же самый MAC-адрес сетевого интерфейса (разъем ноутбука, куда подключили провод), который был и в офисе. Правда, помним, что можно и поменять MAC-адрес, если сильно приспичит.

А вот IP-адрес постоянно меняется. В офисе и дома он будет разным (рис. 8). То есть, это что-то абстрактное, не привязанное к устройству.

Пример:

Вы вошли в сеть кафе по Wi-Fi через телефон - вам дали IP-адрес

Вы отключились

Подключился следующий посетитель - ему вполне могут дать тот же IP-адрес в этой сети

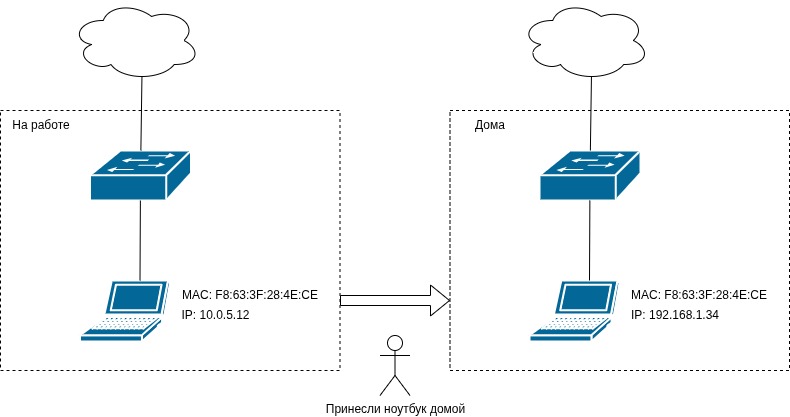


Рис. 8. MAC-адрес остается. IP-адрес меняется.

Более того, возможна такая ситуация:

В офисе "МегаСофт" есть локальная сеть

В офисе "НаноСофт" - другая локальная сеть

Компьютер в "МегаСофт" имеет IP 192.168.1.100

Компьютер в "НаноСофт" тоже имеет IP 192.168.1.100

Но MAC-адреса у них будут разные - уникальные

**Теперь вернемся к структуре IP.**

Возьмем пример: 172.16.128.17

IP состоит из 4 частей, которые называются – октеты. Каждая часть – это 8 бит (ноли и единицы).

Вот как выглядит 172.16.128.17 в виде бит:

10101100.00010000.10000000.00010001

Минимальный IP, когда у него все части состоят из нолей

00000000.00000000.00000000.00000000

Максимальный IP, когда у него все части состоят из единиц

11111111.11111111.11111111.11111111

Смотреть на IP в виде бит – очень увлекательно, но совершенно не воспринимается человеческим разумом. Не очень удобно диктовать свой IP в виде бит по телефону техподдержке. Для этого мы переводим его в привычную нам десятичную структуру.

00000000.00000000.00000000.00000000 будет 0.0.0.0

00000000.00000000.00000000.00000001 - 0.0.0.1

00000000.00000000.00000000.00000010 - 0.0.0.2

...

00000000.00000000.00000000.11111111 - 0.0.0.255

00000000.00000000.00000001.11111111 – 0.0.1.255

...

11111111.11111111.11111111.11111111 – 255.255.255.255

Таким образом, минимальный IP в десятичной системе (в которой нам удобно, а не машине) – это 0.0.0.0. Максимальный IP – 255.255.255.255.

Получается пример 4,3 миллиарда комбинаций.

На самом деле этого очень мало.

И не все комбинации доступны. Большое количество IP-адресов зарезервировано.

К примеру:

0.0.0.0 – это "любой адрес". Эту комбинацию пишут в настройках серверах. Означает, что сервер готов получать запрос с любого устройства.

255.255.255.255 — широковещательный запрос для всей сети. Когда хотим отправить запрос всем устройствам внутри этой сети.

Часть адресов вообще недоступны в интернете (только внутри локальных сетей):

10.0.0.0/8, 192.168.0.0/16, 172.16.0.0/12 – это примерно 18 млн комбинаций.

Тут появились новые обозначения: /8, /16, /12 – это маска подсети. Сейчас ее разберем.

**Маска подсети**

Чтобы разобрать, что такое маска подсети, надо углубиться в структуру IP. Мы говорили выше, что он состоит из 4 частей по 8 бит (в десятичной системе от 0 до 255). Еще IP разделяется логически на 2 части:

адрес подсети (сетевая часть адреса) - одинаковая для устройств в одной сети

адрес хоста (узловая часть адреса) - уникальная для устройств в одной сети

Аналогия. Вы почтальон. Видите на письме индекс. Этот индекс будет состоять из двух частей (рис. 9):

адрес подсети – номер дома. В нем много квартир – это локальная сеть, в которую мы принесли письмо.

адрес хоста – номер квартиры. Это конкретный получатель письма в доме (в локальной сети)

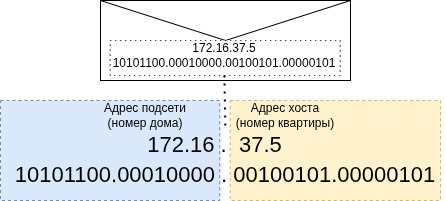


Рис. 9. IP состоит из двух частей.

*А теперь****ВНИМАНИЕ****! Сейчас будет сложно.*

Мы не знаем, какие значения из IP будут в первой и второй части:

Может быть 172.16 и 37.5 или 172.16.37 и 5. А могут быть и другие комбинации (чуть позже поймем, что за другие комбинации).

**Маска подсети** показывает, где первая часть и вторая часть IP

Напомним, что IP состоит из 4 кусочков (октеты) по 8 бит (ноли и единицы). Получается, что в IP ровно 32 бита (8х4).

Маска подсети показывает, сколько бит с левой части принадлежит адресу подсети. Оставшиеся биты принадлежат адресу хоста.

Раньше мы указывали маску подсети, как /8, /12, /16 – это как раз количество бит слева.

**Пример 1**: 172.16.37.5/16

IP 172.16.37.5

Маска подсети /16

--

Представим IP в виде бит:

10101100.00010000.00100101.00000101

Отсчитаем слева 16 бит:

10101100.00010000

Дополним нолями вторую часть:

10101100.00010000.00000000.00000000 – это адрес подсети: 172.16.0.0/16

--

Оставшиеся биты из правой части:

00100101.00000101

Дополним нолями первую часть:

00000000.00000000.00100101.00000101 – это адрес хоста в этой подсети: 0.0.37.5

--

А сколько всего хостов может быть в этой подсети 172.16.0.0/16?

Если первая часть отвечает за номер дома, то вторая часть – это номера квартир:

Первый хост (первая квартира в доме):

10101100.00010000 . 00000000.00000001 – 172.16.0.1

|  |  |  |
| --- | --- | --- |
| Адрес подсети IP (не меняется) | Адрес хоста IP (меняется) | IP |
| 10101100.00010000 | 00000000.00000001 | 172.16.0.1 |
| 10101100.00010000 | 00000000.00000011 | 172.16.0.2 |
| 10101100.00010000 | 00000000.00000100 | 172.16.0.3 |
| ... | … | ... |
| 10101100.00010000 | 00000000.11111111 | 172.16.0.255 |
| 10101100.00010000 | 00000001.00000000 | 172.16.1.0 |
| ... | … | ... |
| 10101100.00010000 | 11111111.11111111 | 172.16.255.255 |

**Последний хост в подсети всегда зарезервирован**. Он используется, когда вы хотите отправить сообщение всем компьютерам в этой сети.

Пример: Если почтальон увидит адрес на конверте 172.16.255.255/16, то это значит, что он должен уведомить каждую квартиру.

Получается, что диапазон хостов в этой подсети:  
172.16.0.1 – 172.16.255.254

Можем разместить 65 534 устройства внутри этой сети.

**Пример 2**: 172.16.37.5/24

IP 172.16.37.5

Маска подсети /24

--

Представим IP в виде бит

10101100.00010000.00100101.00000101

--

Отсчитаем слева 24 бита:

10101100.00010000.00100101

Дополним нолями вторую часть:

10101100.00010000.00100101.00000000 – это адрес подсети: 172.16.37.0/24

--

Оставшиеся биты:

00000101

Дополним нолями первую часть:

00000000.00000000.00000000.00000101 – это адрес хоста в этой подсети: 0.0.0.5

--

А сколько всего хостов может быть в этой подсети 172.16.37.0/24?

|  |  |  |
| --- | --- | --- |
| Адрес подсети IP (не меняется) | Адрес хоста IP (меняется) | IP |
| 10101100.00010000.00100101 | 00000001 | 172.16.37.1 |
| 10101100.00010000.00100101 | 00000011 | 172.16.37.2 |
| 10101100.00010000.00100101 | 00000100 | 172.16.37.3 |
| ... | … | ... |
| 10101100.00010000.00100101 | 00000000.11111111 | 172.16.37.255 |

**Последний хост в подсети зарезервирован**. Он используется, когда вы хотите отправить сообщение всем компьютерам в этой сети.

172.16.37.255/24 – это широковещательный адрес для отправки сообщений всем хостам в подсети 172.16.37.0/24.

Диапазон хостов в этой подсети 172.16.37.0/24:  
172.16.37.1 - 172.16.37.254

Можем разместить 254 устройства внутри этой сети.

Это все прекрасно. В примерах выше маска подсети очень удачно накладывается на октеты:

/24 – первые три октета (3 части по 8 бит)

/16 – первые 2 октета (2 части по 8 бит)

**А что делать с маской подсети /12 ?**

Сейчас будет опять сложно. Надо собраться с силами.

**Пример 3**: 172.16.37.5/12

IP 172.16.37.5

Маска подсети /12

--

Представим IP в виде бит

10101100.00010000.00100101.00000101

--

Отсчитаем слева 12 бит:

10101100.0001

Дополним нолями вторую часть:

10101100.00010000.00000000.00000000 – это адрес подсети: 172.16.0.0/12

10101100 – это 172

00010000 – это 16

00000000 – 0

00000000 – 0

--

Оставшиеся биты:

0000.00100101.00000101 – это адрес хоста в этой подсети

*А вот как это представить в десятичном виде?*

*Тут уже начинаются сложности для понимания в нашей системе исчисления.*

Дополним нолями первую часть:

00000000.00000000.00100101.00000101 – это адрес хоста в этой подсети: 0.0.37.5

А сколько всего хостов может быть в этой подсети 172.16.0.0/12?

|  |  |  |
| --- | --- | --- |
| Адрес подсети IP (не меняется) | Адрес хоста IP (меняется) | IP |
| 10101100.0001 | 0000.00000000.00000001 | 172.16.0.1 |
| 10101100.0001 | 0000.00000000.00000011 | 172.16.0.2 |
| 10101100.0001 | 0000.00000000.00000100 | 172.16.0.3 |
| ... | … | ... |
| 10101100.0001 | 0000.00000000.11111111 | 172.16.0.255 |
| 10101100.0001 | 0000.00000001.0000000 | 172.16.1.0 |
| ... | ... | ... |
| 10101100.0001 | 0000.11111111.11111111 | 172.16.255.255 |
| 10101100.0001 | 0001.0000000.0000000 | 172.**17**.0.0 **ВНИМАНИЕ** |
| ... | … | ... |
| 10101100.0001 | 1111.11111111.11111111 | 172.**31**.255.255 |

172.31.255.255/12 – это широковещательный адрес для рассылки сообщений всем хостам в подсети 172.16.0.0/12.

Диапазон хостов в этой подсети 172.16.0.0/12:

От 172.16.0.1 до 172.31.255.254

Можем разместить 1 048 574 устройства внутри этой сети.

Закончили с IP.

Если пример 3 был сложным, то остановитесь пока на примере 1 и 2. Если понять их, то этого будет достаточно для минимальных знаний.

**Как подсети общаются между собой?**

Выше мы разобрались с IP, а конкретнее - с IPv4. Узнали, что с помощью маски подсети можно узнать адрес подсети (номер дома) и адрес хоста (номер квартиры в этом доме).

И, зная маску подсети, мы можем определить, какой диапазон хостов входит в нашу подсеть (какие номера квартир могут быть в нашем доме).

--

*Разберем пример:*

На нашем компьютере программа делает запрос на какой-то IP. В настройках нашего компьютера есть информация о подсети, в которой мы находимся и маске этой подсети. Мы можем понять: IP, на который мы делаем запрос, находится внутри нашей подсети?

Если да, то мы отсылаем запрос напрямую (рис. 10).

Если нет, то мы должны передать запрос из нашей подсети в другую подсеть, где этот компьютер находится (рис. 11).

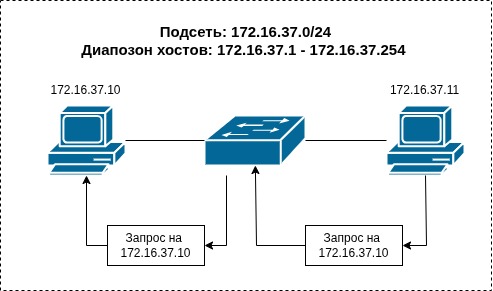


Рис. 10. Отправляем сообщение на IP, который входит в подсеть

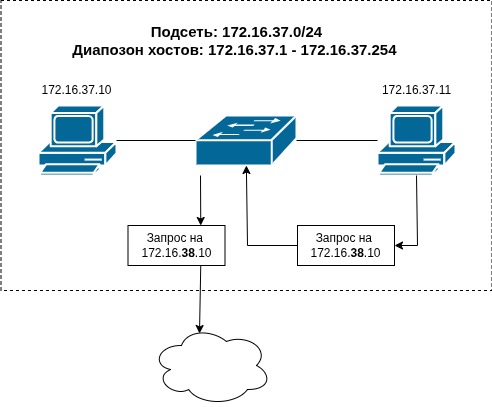


Рис. 11. Отправляем сообщение на IP, который НЕ входит в подсеть

Для того, чтобы передать сообщение в другую подсеть нужно специальное устройство **маршрутизатор** (рис. 12)

*На самом деле любой компьютер можно настроить как маршрутизатор.*



Рис. 12. Домашний маршрутизатор (роутер)

**Маршрутизатор** (еще называют **роутер**) - устройство, которое пересылает данные между разными сетями, выбирая оптимальный путь для их доставки (рис. 13).

Маршрутизатор – это устройство для сетевого уровня (опять говорим про уровни, но раскроем их понятие позже).

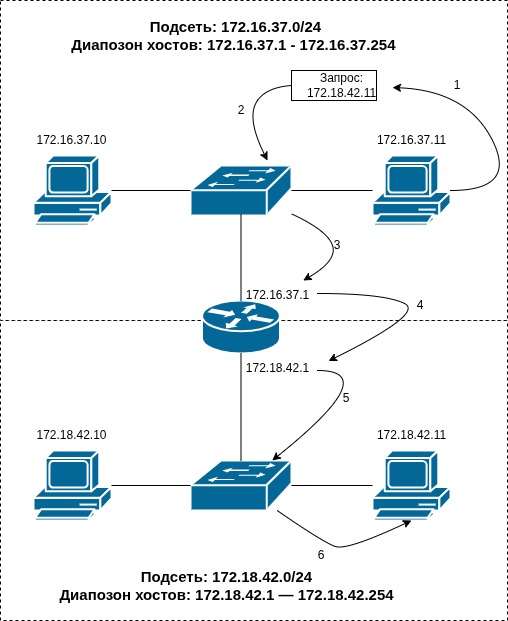


Рис. 13. Передача запроса из одной подсети в другую подсеть через маршрутизатор.

На рис. 13 показано, как маршрутизатор соединяет подсети. Он находится на их границе. Один сетевой интерфейс (напоминаю, что это разъем, куда вставляем провод), имеет IP из подсети 172.16.37.0/24, другой сетевой интерфей этого маршрутизатора имеет IP из подсети 172.18.42.0/24.

Проследим, как идет запрос с компьютера (IP 172.16.37.11) на IP в другой подсети (172.18.42.11):

Делаем запрос 172.18.42.11

Этот IP не принадлежит подсети 172.16.37.0/24

Отправляем его маршрутизатору на сетевой интерфейс, который входит в нашу подсеть (IP 172.16.37.1)

Маршрутизатор перенаправляет запрос на другой свой сетевой интерфейс, который соответствует подсети запроса (IP 172.18.42.1). Как видим, маршрутизатор стоит на границе подсетей

Перенаправляем запрос коммутатору

Коммутатор передает запрос нужному устройству в этой подсети

Прежде, чем идти дальше, вспомним предыдущие темы из лекции.

Коммутатор (работает, как переходник) передает сообщения, которые называются **кадр**. Кадр содержит в себе MAC-адрес (физический адрес устройства) получателя и отправителя. В таблице коммутации хранится информация, через какой порт (и провод, который в него воткнут) какому устройству передавать кадр.

Посылая сообщения по сетевому уровню (по IP-протоколу через маршрутизатор), мы указываем **IP-**адрес получателя и отправителя.

Посылая кадр по канальному уровню (через коммутатор), мы указываем **MAC-**адрес получателя и отправителя.

--

Как все происходит (рис. 14):

Представьте конверт – пакет. Его мы сформировали на сетевом уровне. Положили в него данные. Написали на нем информацию об IP

Теперь представьте конверт – кадр. Его сформировали на канальном уровне. Положили в него пакет (конверт с IP). Написали на нем информацию о MAC

Отправляем конверт в конверте.

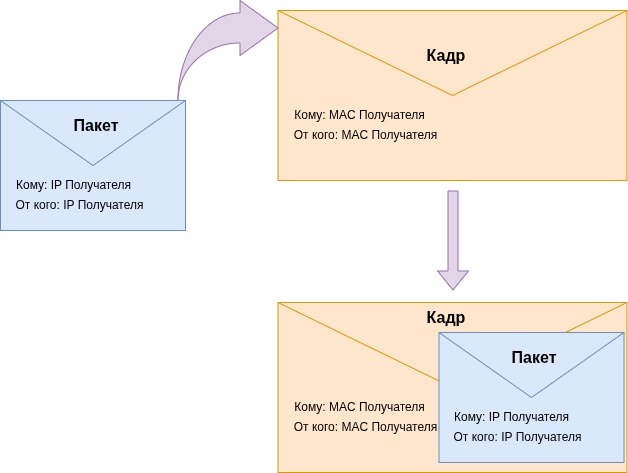


Рис. 14. Формируем пакет на сетевом уровне и кадр на канальном уровне

А теперь разберем подробнее, как сообщение проходит через маршрутизатор (рис. 15):

Отправляем сообщение с PC1

IP-получателя находится в другой подсети

Формируем пакет, в котором указали IP PC1(отправитель) и IP PC2 (получатель).

Формируем кадр. В нем указали MAC PC1 - отправитель. Но мы не знаем MAC-адреса в другой подсети. Укажем MAC получателя – шлюз подсети (сетевой интерфейс маршрутизатора, который находится в нашей подсети)

Запаковали пакет (в нем IP) в кадр (в нем MAC)

Маршрутизатор пересылает сообщение на свой сетевой интерфейс, который находится в подсети получателя

Теперь новая проблема. В конверте с кадром указаны MAC-адрес получателя и отправителя из другой подсети.

Маршрутизатор распаковал конверт с кадром. Достал из него конверт с пакетом и вставил пакет в новый кадр.

В новом кадре указан MAC-адрес получателя – PC2 и MAC-адрес отправителя сетевой интерфейс маршрутизатора из второй подсети.

Сообщение доходит до PC2

Ответ от PC2 идет до PC1 аналогичным образом.

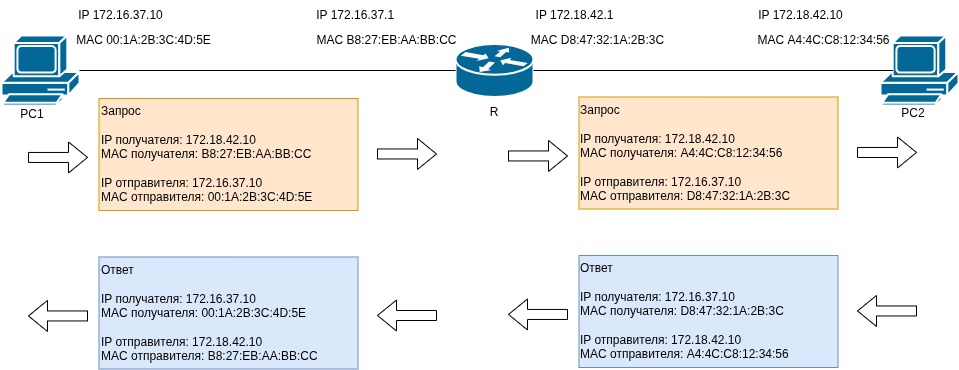


Рис. 15. IP и MAC-адреса при отправке сообщения через маршрутизатор

Если вы запутались, как сообщение идет через маршрутизатор из одной подсети в другую, то ничего страшного. Главное запомните:

Сообщения внутри одной подсети идут напрямую через коммутатор (типа разветвителя)

Сообщения в другую подсеть идут через специальное устройство - маршрутизатор

Мы знаем только логический адрес устройства в другой сети – IP

Мы НЕ знаем физический адрес устройства в другой сети – MAC

Все это замечательно. Но наш маршрутизатор стоит на границе двух подсетей, в которых мы пересылаем сообщения.

**А что если сообщение предназначается вообще другой подсети? С которой мы не связаны через маршрутизатор.**

Никаких проблем. Маршрутизаторы связаны со другими маршрутизаторами (рис. 16). У маршрутизаторов есть таблица маршрутизации. И ваш запрос проходит через цепочку маршрутизаторов до нужной подсети оптимальным образом.

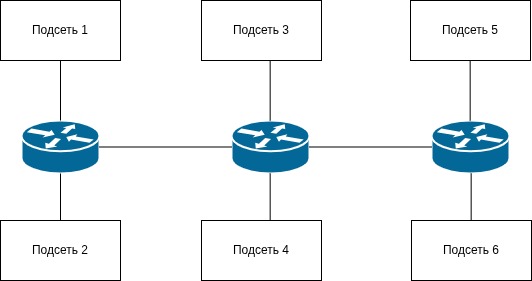


Рис. 16. Связь маршрутизаторов.

Можно настраивать маршрутизаторы статически. Т.е. явно указывать в настройках каждого, что если хочешь попасть в такую подсеть, то постучать через такой маршрутизатор. А есть уже готовые настройки для составления маршрутов. Например, протокол RIP (старый для малых сетей) или протокол OSPF. Не будем разбирать их подробно. Это уже относится к теме администрирования сетей. Но эти протоколы прекрасно поддерживаются современными маршрутизаторами из коробки. Включаем настройки на устройстве и он автоматически начинает обмениваться данными с соседними маршрутизаторами о доступных подсетях и строить таблицы маршрутизации.

Остается еще один важный вопрос. MAC-адрес прописывается на устройстве производителем (хоть мы и можем его поменять вручную). **А как устанавливается IP-адрес?**

Можно прописывать IP-адрес вручную (назначать статически). Способ подойдет, если устройств в вашей подсети мало, а у администратора много свободного времени.

А что если устройств много? И они постоянно меняются?

--

Например посетители в кафе. Вы подключаетесь по Wi-Fi.

*Это тоже подключение к подсети, как и через провод. Только используются радиоволны.*

*Одно устройство раздает Wi-Fi внутри этой подсети , а на вашем телефоне есть сетевой интерфейс, который подключается к этому устройству. Для упрощения можете представить, что вы хватаете провод и втыкаете его в свой телефон, когда подключаетесь к сети Wi-Fi.*

Чтобы телефон мог отправлять запросы, ему надо сформировать пакет (конверт с IP) на сетевом уровне. В пакете прописывается IP-получателя и IP отправителя. Значит, ваш телефон должен иметь IP. И не любой, а подсети из этого кафе.

В этом нам помогает еще одно устройство – **DHCP сервер**.

--

**DHCP сервер** - автоматический раздатчик IP-адресов.

Он знает все об этой подсети, какой диапазон адресов хостов доступен. И он решает, какой IP-адрес получит каждое устройство и на какое время (рис. 17).

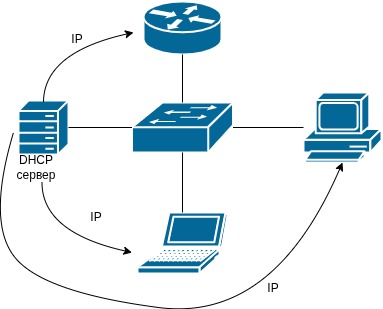


Рис. 17. DHСP назначает IP-адреса устройствам внутри подсети

Теперь можем подумать, как работает ваша домашняя подсеть. Возьмем обычную квартиру. Что там может быть (рис. 18):

стационарный компьютер: подключен через витую пару (провод)

легкий ноутбук: подключен через Wi-Fi

несколько телефонов: подключены через Wi-Fi

роутер (маршрутизатор), который:

будет совмещать в себе функции коммутатора (разветвитель)

иметь точку доступа Wi-Fi (к ней будут подключаться телефоны и ноутбук)

назначать IP (выступает в роли DHCP сервера)

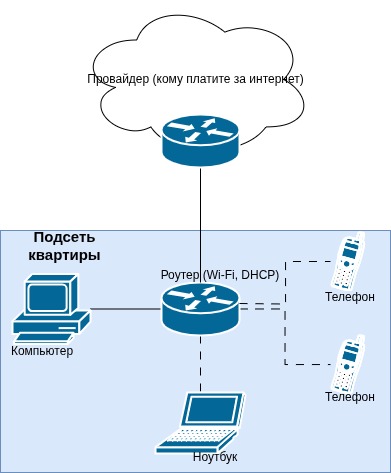


Рис.18. Подсеть квартиры

Модель OSI

Теперь мы знаем уже достаточно много, чтобы разобрать модель OSI и проекцию на нее TCP/IP (рис. 19).

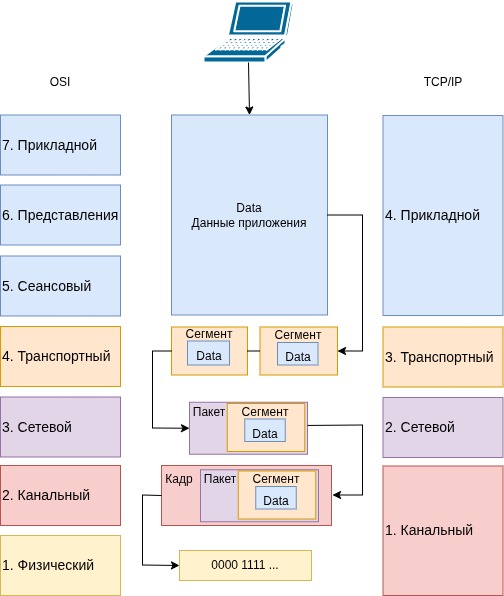


Рис. 19. Модель OSI и TCP/IP

Рисунок 19 может показаться страшным и непонятным, но не надо пугаться. На самом деле мы уже почти все разобрали, осталось свести наши знания в целостную модель.

Сразу скажем, что OSI – академическая модель. Разработана Международной организацией по стандартизации (ISO). Ее учат в вузах, приводят во всех учебниках и зачем-то спрашивают на собеседованиях :)

TCP/IP – модель, которую используют в реальной жизни, и она накладывается на модель OSI.

**Будем сразу разбирать TCP/IP** (и ее проекцию на OSI):

***4. Прикладной уровень:***

Вы работаете на ноутбуке. Открыт браузер – программа. Посылаем запрос из браузера к серверу (сайту). На этом уровне программа должна подготовить данные.

Пример:

отправляем JPEG-картинку.

Проекция на OSI:

подготовить данные (саму картинку) - 7. прикладной уровень

делаем перекодировку и шифрование данных - 6. представление уровень

если созваниваемся с коллегой, то нужно установить сеанс - 5. сеансовый уровень

--

***3. Транспортный уровень:***

Тут делают сегментацию данных — это процедура разбиения исходных данных уровня приложений (независимо от исходного формата, HTML-файл или JPEG-изображение и т.д.) на небольшие части размером около полутора килобайт. Эти части называются – **сегмент**.

Мы берем наши данные, раскладываем их по конвертам – **сегменты**. На каждый конверте пишем подробную инструкцию, как их распаковать и собрать данные заново.

Принимающая сторона по информации на конверте сможет понять, что лежит в этом сегменте и восстановить ее в нужной программе.

Проекция на OSI:

в OSI такой же уровень – транспортный.

*--*

***2. Сетевой уровень:***

Тут данные передаются логически между устройствами по IP (логический адрес устройства, который может меняться). IP помогает понять, в какую подсеть отправить данные.

Берем конверт (тут он называется **пакет**) и пишем на нем IP-адрес получателя и отправителя.

Помещаем в конверт (пакет) предыдущий конверт (сегмент).  
  
  
Проекция на OSI: в OSI такой же уровень – сетевой.

Устройства:

Маршрутизатор (роутер) – с его помощью по IP перенаправляем пакеты из нашей подсети в нужную подсеть.

*--*

***1. Канальный уровень:***

Он отвечает за физическую адресацию. Тут мы указываем уникальный код устройства – MAC-адрес.

Опять представим, что берем конверт (тут он называется **кадр**). На нем пишем MAC-адрес получателя и отправителя. Кладем в него конверт с предыдущего уровня (пакет).

Проекция на OSI:

упаковка в кадр – в OSI такой же уровень – канальный

пересылаем эти данные физически с помощью электрических, световых импульсов или радиоволн – 1. физический уровень

Устройства:

Коммутатор – с его помощью по MAC перенаправляем кадр нужному устройству в этой подсети.

Прекрасно, мы разобрали модели OSI и TCP/IP!!!

Теперь для полноты картины нужно осветить еще одну тему.

Протоколы транспортного уровня: TCP и UDP

Чуть выше мы узнали, что на транспортном уровне мы берем данные (JPEG-картинку) и разбиваем эти данные на сегменты (примерно по 1.5 килобайта).

Здесь существует 2 основных протокола, которые поддерживают все устройства:

TCP

UDP

Собственно, задачи этих протоколов:

Разбить данные на сегменты

Сопоставить эти сегменты с конкретной программе на компьютере

Проконтролировать порядок передачи сегментов

Отличие TCP от UDP – гарантия передачи данных.

TCP гарантирует, что данные будут переданы. Если что-то пойдет не так, то все пересылается заново. Применяется везде, где нам важно получить каждый байт.

К примеру, будет печально, если вы получили инструкцию, в которой будет пропущена часть символов. Скорее всего вы ее даже не сможете правильно раскодировать на своем компьютере.

UDP – отправил и забыл. Применяется там, где нам не критична потеря части данных. Пример: созвоны с коллегами. Если перестали слышать коллегу, то ничего страшного. В крайнем случае, он может повторить, что хотел донести до вас.

А теперь представим следующую ситуацию:

Есть сервер. На него идет большое количество одновременных запросов с разных - устройств. Кто-то запрашивает HTML-страницы, кто-то данные из базы данных, кто-то присылает email.

Все это приходит в виде одного большого потока данных в виде нолей и единиц по проводу на физический сетевой интерфейс.

Наш сервер умеет разбирать этот поток нолей и единиц на кадры.

Открываем кадр и достаем из него пакет

Открываем пакет и достаем из него сегмент

--

А что делать дальше с этим сегментом? Его передать программе, которая вернет HTML? Или программе, которая поищет данные в базе данных? Или передать почтовому менеджеру?

Тут разберется протокол транспортного уровня. В сегменте указан **порт** программы, к которой он относится.

**Порт** — число от 0 до 65 535. По нему сопоставляют сегмент с нужной программой (рис. 20).

У программ есть стандартные порты, которыми они пытаются завладеть. Часто программы могут конфликтовать, т.к. слушают одинаковые порты. Эти ситуации приходится разруливать вручную.

К примеру web-server на http по умолчанию слушает порт 80.

Популярная база данных PostgreSQL по умолчанию слушает порт 5432.

Почтовые серверы слушают по умолчанию порт 25.

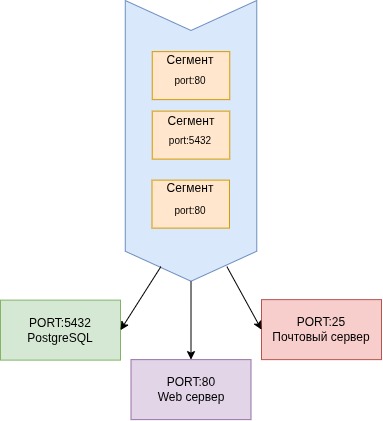


Рис. 20. Сопоставление сегмента по порту с нужной программой.

Вопрос: **как сопоставить сегмент с нужным портом?**

Это делается с помощью сокета.

**Сокет** – это тоже интерфейс, только программный. До этого мы говорили про сетевой интерфейс (разъем для провода в ноутбуке или модуль для связи по Wi-Fi). А это интерфейс на уровне программы компьютера. Он привязывает соединение по IP к определенному порту (рис. 21).

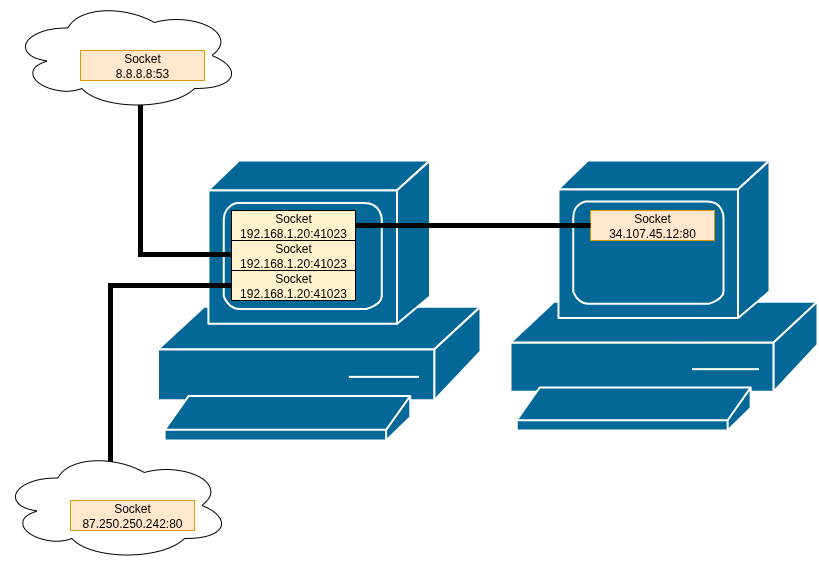


Рис. 21. Сокеты – программные интерфейсы. IP:ПОРТ

Каждое открытое соединение имеет 2 сокета:

На стороне клиента (кто открыл соединение) IP\_клиента:ПОРТ\_в\_системе\_клиента

На стороне сервера (кро принял соединение) IP\_сервера:ПОРТ\_в\_системе\_сервера

Сейчас узнаем, для чего вообще мы заморачивались с этими сокетами (IP:ПОРТ).

Вспомним, что у нас не так уж много комбинаций IP-адресов (пример 4,3 миллиарда комбинаций) и часть из них служебные.

Если у нас закрытая локальная сеть, то никаких проблем. Мы используем любые (не служебные) IP.

Наш сосед в своей закрытой локальной сети также использует любые IP. Они могут совпадать с нашими.

Если мы соединимся с соседом через маршрутизатор (устройство сетевого уровня для объединения сетей), то наши адреса уже не должны совпадать. У нас должна быть хотя бы разная подсеть (отличаться по маске подсети).

А что если нам надо соединить ОЧЕНЬ много локальных сетей? Так много, что получится Интернет :)

У нас у всех должны быть уникальные подсети и IP. Проблема, что на всех не хватит IP в такой системе.

Можно решить эту проблему, если давать 1 публичный IP на всю сеть (рис. 22).

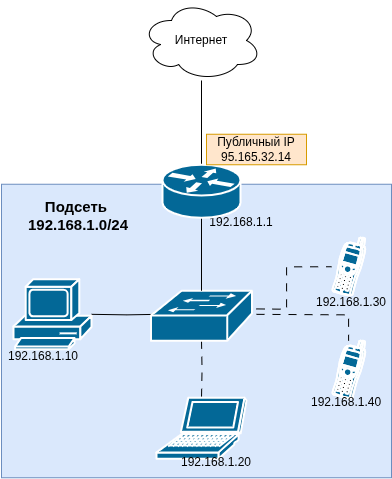


Рис. 22. Выделяем 1 публичный IP на нашу подсеть.

На рис. 22 у нас есть закрытая подсеть 192.168.1.0/24. Внутри нее выделяем IP каждому устройству. Но с интернетом мы общаемся только через 1 публичный IP (его называют WAN-адрес роутера) – 95.165.32.14.

95.165.32.14 – закреплен за нами во всей публичной сети интернет.

Когда наши устройства хотят постучаться к другому публичному IP (сервер Яндекса), то мы обращаемся к нему по нашему публичному IP 95.165.32.14. Откроете страницу с компьютера, ноутбука или телефона, то Яндекс увидит вас, как 95.165.32.14.

Круто! На все наши устройства теперь хватило 1 публичного IP. Мы решили проблему нехватки IPv4 в мире (до поры до времени).

Но какому устройству ответит Яндекс. Он видит 1 IP для всей нашей подсети.

Пример:

Вы лежите с женой в постели и почему-то оба шлете с телефонов (по Wi-Fi) запросы в Яндекс вместо других супружеских дел.

Вы отправили запрос "Найди анекдот". Он пришел на Яндекс с IP 95.165.32.14 (IP вашего телефона в подсети 192.168.1.30).

Ваша жена отправила запрос "Найди рецепт борща". Он пришел на Яндекс с IP 95.165.32.14 (IP вашего телефона в подсети 192.168.1.40).

Яндекс отправляет ответ со свежим анекдотом на IP 95.165.32.14.

Как ваш роутер поймет, на какой телефон в подсети переслать анекдот?

Тут на помощь приходят сокеты, которые мы разобрали раньше.

Ваш запрос и запрос вашей супруги имеет разный сокет подключения к Яндексу.

Сокет – это IP:ПОРТ.

--

Продолжаем пример:

Ваш запрос к Яндексу по сокету 95.165.32.14:41000

Запрос вашей супруги к Яндексу по сокету 95.165.32.14:41001

Ответ от Яндекса с анекдотом на сокет 95.165.32.14:41000

Ответ от Яндекса с рецептом борща на сокет 95.165.32.14:41001

Вашему роутеру осталось только запомнить: к какому устройству внутри сети привязан открытый сокет. Это называется – технология NAT.

**NAT** (Network Address Translation) – преобразует частный IP в публичный и наоборот.

На рис. 23 показана таблица трансляции вашего телефона и телефона вашей супруги.

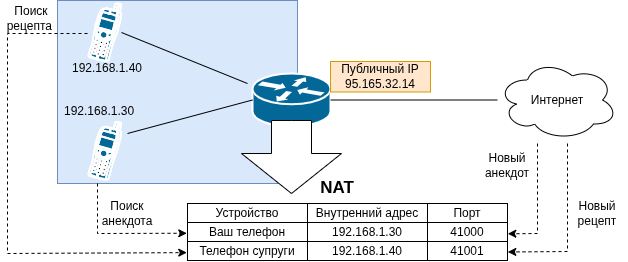


Рис. 23. NAT

Как мы уже разбирали, IPv4-адресов не хватает, и NAT помогает решить эту проблему.

СУПЕР! Мы разобрали минимальные основы, как работает сеть!

**Разберем напоследок IPv6**

До этого говорили, что протокол IP имеет 2 реализации:

IPv4

IPv6

Главная проблема с IPv4 – нехватка адресов.

IPv6 решает эту проблему.

Он состоит из 8 частей (IPv4 из 4 частей). Каждая часть содержит 16 бит (нолей и единиц). В IPv4 было 8 бит в каждой части.

В IPv6 128 бит (8 частей по 16 бит). Это дает 2 ^ 128 степени. Поверьте, это ОЧЕНЬ много. Хватит всем с запасом.

--

Пример IPv6 в виде бит:

0010000000000001 0000110110111000 1000010110100011 0000000000000000

0000000000000000 1000101000101110 0000001101110000 0111001100110100

Прочитать и запомнить человеку – нереально. Для чтения надо привести к другой системе счисления.

Решили привести к шестнадцатеричной системе счисления. В ней одно число может иметь значения:

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

В этом системе исчисления запись будет иметь вид:

2001:0db8:85a3:0000:0000:8a2e:0370:7334

--

В IPv6 можно сокращать и дальше. Правила сокращения:

Пропуск ведущих нулей в группах: 0db8 → db8, 0370 → 370.

Замена последовательности нулей на :: (можно сделать только один раз в адресе).

Сокращенный вариант того же адреса:

2001:db8:85a3::8a2e:370:7334

Сокращенный вариант уже начинает восприниматься человеческим сознанием.

В IPv6 есть и другие плюсы помимо большего числа комбинаций. В нем можно использовать встроенное шифрование, не надо выделять отдельный широковещательный адрес. Но мы уже не будем останавливаться на этом.

Почему же все не перешли на протокол IPv6?

несовместимость «из коробки» с IPv4. Дополнительные усилия на настройку

старые маршрутизаторы не поддерживают IPv6. А маршрутизатор не самое дешевое устройство, чтобы массово закупать новое оборудование

И так все работает. NAT позволяет работать всей подсети через 1 публичный IP