Что такое модель OSI и зачем она нужна: препарируем слоёный пирог

Что происходит, когда вы отправляете сообщение, скажем, в Telegram? Понятно, что Telegram отправляет это сообщение. Но что в этот момент происходит в компьютере и в сети? Куда летят файлы и как они понимают, куда им лететь?

Что такое модель OSI

Модель OSI (Open System Interconnection), или эталонная модель взаимодействия открытых систем описывает, как устройства в локальных и глобальных сетях обмениваются данными и что происходит с этими данными. Её предложили в 1984 году инженеры из Международной организации по стандартизации (ISO), которая работала над единым стандартом передачи данных по интернету.

При этом сама по себе эталонная модель — не стандарт интернета, как, например, TCP/IP; её можно сравнить с фреймворками в мире языков программирования: в OSI «из коробки» доступны разные веб-стандарты — UDP, HTTP, FTP, Telnet и другие. Всего таких протоколов — более 100 штук.

Модель OSI включает семь слоёв, или уровней, — причём каждый из них выполняет определённую функцию: например, передать данные или представить их в понятном для человека виде на компьютере. Кстати, у каждого слоя — свой набор протоколов.

Слои ничего не знают о том, как устроены другие слои. Это называется абстракцией.
Самый нижний слой отвечает за физическое представление данных, то есть за то, как данные передаются по проводам или с помощью радиоволн, а самый верхний — за то, как приложения взаимодействуют с сетью.

Нижний слой оперирует такими понятиями, как «тип кабеля» или «тип коннектора», а верхний — такими, как HTTP или API.

Описание сетевой технологии и алгоритма функционирования компьютерной сети связано с описанием соответствующих интерфейсов и протоколов.

Интерфейс – соглашение о взаимодействии (границе) между уровнями одной системы, определяющее структуру данных и способ (алгоритм) обмена данными *между соседними уровнями* OSI-модели.

Интерфейсы подразделяются на:

1) *схемные* – совокупность интерфейсных шин;

2) *программные* – совокупность процедур реализующих порядок взаимодействия между уровнями.

Протокол – совокупность правил, регламентирующих формат и процедуры взаимодействия процессов *одноименных уровней* на основе обмена сообщениями.

Описание протокола предполагает задание:

1) **логической характеристики протокола**, определяющей *структуру* (формат) и *содержание* (семантику) *сообщений* путём перечисления типов сообщений и их смысла;

2) **процедурной характеристики протокола**, представляющей собой *правила выполнения действий*, предписанных протоколом взаимодействия и задаваемых в форме: операторных схем алгоритмов. автоматных моделей, сетей Петри и др.

Рассмотрим каждый слой подробнее.

1-й уровень OSI — физический (L1, physical layer)

На самом нижнем уровне модели OSI данные представляют собой физические объекты — ток, свет или радиоволны. Они передаются по проводам или с помощью беспроводных сигналов.

Этот слой работает с кабелями, контактами в разъёмах, модуляцией сигнала, кодированием единиц и нулей и другими низкоуровневыми штуками. По сути, первый уровень — это уровень проводов и физических способов передачи сигнала. Минимальная абстракция. Данные в виде сигналов передаются между устройствами

Самый известный протокол на физическом уровне — Ethernet. Он описывает, как сигналы кодируются и передаются по проводам. Кроме него есть Bluetooth, Wi-Fi и ИК-порт, которые также содержат инструкции для передачи данных.

Устройства физического уровня — концентраторы и репитеры. Они работают с физическим сигналом «втупую» и не вникают в его логику: получили данные — передали их дальше по проводу.

2-й уровень OSI — канальный (L2, data link layer)

Над физическим уровнем располагается канальный. Его задача — проверить целостность полученных данных и исправить ошибки. Этот уровень «поумнее» предыдущего: он уже понимает, что разные амплитуды напряжений отвечают разным битам — нулям и единицам. А ещё канальный уровень умеет кодировать сигналы в биты и передавать их дальше.

Полученные с нижнего уровня данные делятся на фреймы, или кадры. Каждый фрейм состоит из служебной информации — например, адреса отправителя и адреса получателя, — а также самих данных.

Получается что-то вроде почтового конверта. На лицевой стороне у него написано, от кого пришло письмо, а внутри находится само письмо (в нашем случае данные).

Лицевая сторона конверта — это MAC-адрес устройства, которое отправило нам информацию. Он нужен, чтобы идентифицировать устройства в локальной сети, состоит из 48 или 64 бит и выглядит примерно так:

```
1d:85:25:11:b2:0a
```

Ещё один важный факт о MAC-адресах: когда на заводе собирают ноутбук или смартфон, ему сразу же присваивают определённый MAC-адрес, который потом уже никак нельзя поменять. MAC-адрес настольных ПК зашит в сетевую карту, поэтому его можно изменить, только заменив эту самую карту.

Канальный уровень не так прост — он делится ещё на два подуровня:

- уровень управления логическим каналом — LLC (logical link control);
- уровень управления доступом к среде — тот самый MAC (media access control).

Первый подуровень нужен для взаимодействия с верхним уровнем, сетевым, а второй — для взаимодействия с нижним, физическим.

Устройства канального уровня — коммутаторы и мосты. Они нужны, чтобы передавать фреймы нужному адресату. Протоколы канального уровня — PPP, CDP.

3-й уровень OSI — сетевой (L3, network layer)

Этот уровень отвечает за маршрутизацию данных внутри сети между компьютерами. Здесь уже появляются такие термины, как «маршрутизаторы» и «IP-адреса».

Маршрутизаторы позволяют разным сетям общаться друг с другом: они используют MAC-адреса, чтобы построить путь от одного устройства к другому.

Данные на сетевом уровне представляются в виде пакетов. Такие пакеты похожи на фреймы из канального уровня, но используют другие адреса получателя и отправителя — IP-адреса.

Чтобы получить IP-адрес обоих устройств (отправителя и получателя), существует протокол ARP (address resolution protocol). Он умеет конвертировать MAC- в IP-адрес и наоборот.

4-й уровень OSI — транспортный (L4, transport layer)

Из названия понятно, что на этом уровне происходит передача данных по сети. Так и есть. Два главных протокола здесь — TCP и UDP. Они как раз и отвечают за то, как именно будут передаваться данные.

TCP (Transmission Control Protocol) — это протокол, который гарантирует доставку данных в корректном виде. Он жёстко следит за каждым битом информации, но работает гораздо медленнее UDP.

Например, когда вы вводите логин и пароль при входе в социальную сеть, очень важно, чтобы все символы отправились в определённой последовательности. Если какие-то потеряются или изменятся, вы просто не сможете авторизоваться. Поэтому протокол TCP использует разные методы проверок — например, контрольные суммы.

А вот в видео или аудио небольшие потери некритичны, зато важна скорость передачи данных. Для таких задач как раз и придумали протокол UDP (user datagram protocol). Он уже не проверяет цельность битов, его задача — как можно быстрее передать данные с одного устройства на другое.

В протоколе TCP данные делятся на сегменты. Каждый сегмент — часть пакета. Сегменты нужны, чтобы передавать информацию по сети, учитывая её пропускную способность.

Например, если вы передаёте данные с компьютера, у которого пропускная способность 100 Мб/с, на смартфон с пропускной способность 10 Мб/с, то данные разделятся так, чтобы не застревать в самом медленном устройстве.

Ещё сегментация важна для надёжности. Один большой пакет может потеряться или направиться не тому адресату. А маленькие пакеты снижают риск подобных ошибок и даже позволяют проверять их количество. Если какой-то сегмент не получилось доставить, протокол TCP может запросить его у отправителя снова. Так обеспечивается надёжность.

В UDP данные делятся на датаграммы — это примерно то же, что и пакет, только датаграммы автономны. Каждая датаграмма имеет всё необходимое, чтобы дойти до получателя. Поэтому они не зависят от сети и могут доставляться по разным маршрутам и в произвольном порядке.

5-й уровень OSI — сеансовый (L5, session layer)

Начиная с этого уровня и выше, данные имеют уже нормальный вид — например, привычных нам JPEG- или MP3-файлов. Задача сети на этих уровнях — представить информацию в понятном для человека виде и сделать так, чтобы пользователь мог её как-то «потрогать».

Сеансовый уровень управляет соединениями, или сессиями. Типичный пример — звонок по Skype или Zoom. Когда вы звоните другому человеку, между вашими компьютерами устанавливается соединение, по которому передаются аудио и видео. Если такое соединение разорвать, то и ваш звонок прервётся.

На сеансовом уровне очень важно, чтобы соединение правильно установилось и поддерживалось. То есть механизмы протоколов должны проверить, что у обоих собеседников есть нужные кодеки и сигнал между устройствами присутствует.

6-й уровень OSI — уровень представления данных (L6, presentation layer)

На этом уровне происходит преобразование форматов данных — их кодирование и сжатие. Например, полученные данные могут превратиться в GIF- или MP4-файл. То же самое происходит и в обратном порядке: когда пользователь отправляет файл другому человеку, данные сначала конвертируются в биты и сжимаются, а потом уже передаются на транспортный уровень.

Помимо кодировки и сжатия на уровне представления, данные могут шифроваться — если, конечно, это необходимо.

7-й уровень OSI — прикладной (L7, application layer)

Последний уровень модели OSI — прикладной. На нём находятся сетевые службы, которые помогают без проблем сёрфить в интернете.

Прикладной уровень похож на некий графический интерфейс для всей модели OSI — с его помощью пользователь взаимодействует с другими уровнями, даже не подозревая об этом. Этот интерфейс называется сетевым.

Самые популярные из сетевых интерфейсов — это HTTP, HTTPS, FTP и SMTP. А «устройства» здесь — это уже программы: Zoom, Telegram, браузеры.

Как на практике работает сетевая модель OSI

В начале статьи мы задались вопросом: а как передаются сообщения в Telegram? Настало время на него ответить — и показать весь процесс передачи данных по модели OSI.

Мы хотим отправить сообщение нашему другу. Печатаем текст и нажимает кнопку «Отправить», а дальше перемещаемся внутрь компьютера.

Прикладной уровень. Приложение Telegram работает на прикладном уровне модели OSI. Когда мы печатаем текст сообщения и нажимаем кнопку «Отправить», эти данные передаются на сервер мессенджера, а оттуда — нашему другу.

Весь процесс проходит через API разных библиотек — например, для HTTP-запросов. Интерфейсы позволяют без лишних проблем обмениваться данными и не погружаться в то, как они представлены на низком уровне. Всё, что нужно знать, — это какую функцию вызвать и какие переменные туда передать.

Уровень представления. Здесь данные должны преобразоваться в унифицированный формат, чтобы их можно было передавать на разные устройства и операционные системы. Например, если мы отправляем сообщение с Windows на macOS, данные должны быть в читаемом для компьютеров Apple виде. Такая же ситуация и с другими устройствами.

Раз мы собираемся передать данные на другой компьютер, их нужно перевести в бинарный формат. После этого начнётся сам процесс передачи по сети.

Сеансовый уровень. Чтобы данные успешно передались сначала на сервер Telegram, а затем к нашему другу, приложению нужно установить соединение, или сеанс. Он обеспечивает синхронизацию между устройствами и восстанавливает связь, если она прервалась.

Благодаря сеансам вы можете видеть, что собеседник что-то печатает или отправляет вам картинки или видео. Но главная задача этого соединения — обеспечить стабильное соединение для передачи данных.

Транспортный уровень. Когда соединение установлено и данные унифицированы, пора передавать их. Этим занимается транспортный уровень.

Здесь данные разбиваются на сегменты и к ним добавляется дополнительная информация — например, номер порта и контрольные суммы. Всё это нужно, чтобы данные дошли до пользователя в целостности.

Сетевой уровень. Теперь данным нужно найти маршрут к устройству нашего друга, а затем отправить их по нему. Поэтому данные упаковываются в пакеты и к ним добавляются IP-адреса.

Чтобы получить IP-адрес устройств, которым нужно отправить пакеты, маршрутизаторы (устройства сетевого уровня) обращаются к ARP. Этот протокол быстро найдёт адрес получателя и отдаст его нам.

Канальный уровень. Здесь данные передаются от одного MAC-адреса к другому. Изначальный текст делится на фреймы — с заголовками и контрольными суммами для проверки целостности данных.

Физический уровень. И на самом нижнем уровне данные в виде электрических сигналов передаются по проводам, кабелям или по радиоволнам. Тут только одна задача — как можно быстрее откликаться на сигналы свыше.

После прохождения всех уровней модели OSI сообщение успешно доставляется на устройство нашего друга. Правда, в реальности это занимает всего миллисекунды.

Что такое IP-адрес и маска подсети и зачем они нужны

Рассказываем, что такое IP-адрес и маска подсети, зачем они нужны и как используются.

Компьютерам, серверам и роутерам в интернете нужно понимать, куда отправлять данные, чтобы они не потерялись в паутине проводов и прочих вайфаев по пути с какого-нибудь американского хранилища «Ютуба» в браузер дяди Васи в Череповце. Один из помощников в этом деле — IP-адрес. Он представляет собой что-то вроде дорожного указателя, маяка, который содержит данные о месте конкретного устройства в структуре Глобальной сети.

Чтобы узнать IP-адрес вашего устройства, можно открыть терминал и ввести ipconfig в Windows или ifconfig в macOS и Linux:

Что такое IP-адрес и как он устроен

Чаще всего это четыре числа, которые разделены между собой точками (такой формат поддерживается в протоколе IPv4). Например, вот один из самых популярных IP-адресов — вы могли вводить его, чтобы зайти на свой роутер:

Каждое из чисел в адресе — это восьмизначное двоичное число, или октет. Оно может принимать значения от 0000 0000 до 1111 1111. Или же от 0 до 255 в десятичной системе счисления — то есть 256 разных значений.

Получается, диапазон IP-адресов стартует с 0.0.0.0 и заканчивается 255.255.255.255. Если посчитать количество всех адресов в этом диапазоне, получится 4 294 967 296.

Формат адресов IPv4 — не единственный, хоть и один из самых популярных в интернете. Есть ещё стандарт IPv6 — его адреса состоят уже из 128 битов (в IPv4 — 32 бита). Таким образом, IPv6 позволяет пронумеровать 2^{128} устройств (по 300 миллионов на каждого жителя Земли).

Ниже мы будем говорить только об IPv4, однако эти принципы хорошо ложатся и на IPv6.

Из чего состоит IP-адрес

На самом деле IP-адрес — это чуть больше, чем просто набор чисел. Он всегда состоит из двух частей: номера хоста (устройства) и номера сети.

Например, IPv4-адрес 192.168.1.34 состоит из таких смысловых частей:

В нём первые три числа означают номер сети, а четвёртое — номер хоста (то есть вашего устройства). Все устройства, идентификаторы которых начинаются с 192.168.1, находятся в одной сети.

Устройство, идентификатор которого начинается, например, с 192.168.2, будет принадлежать к другой сети и не сможет связываться с устройствами из сети 192.168.1. Чтобы это сделать, понадобится роутер, который соединит две сети между собой.

Он будет мостом, по которому данные переходят из одной сети в другую. Если же говорить техническим языком, то роутер — это сеть более высокого уровня, которая объединяет несколько подсетей. Со стороны это будет выглядеть так, будто у роутера есть устройства, которым он передаёт данные и которые могут связываться между собой.

Какими бывают IP-адреса

Номер сети может храниться не только в первых трёх октетах, но и в первых двух или даже в одном. Остальные числа — это номера устройств в сети.

Чтобы компьютер понимал, какие октеты обозначают сеть, а какие — компьютеры и роутеры, используют несложный механизм. Первые несколько битов в двоичном представлении IP-адреса фиксируются, считываются компьютером и автоматически распознаются — это похоже на конструкцию switch в языках программирования:

- Если первый бит — это 0, значит, компьютер имеет дело с большой сетью, на которую указывает только одно, самое первое число.

При этом первый бит у нас уже зарезервирован под такой «свитч», поэтому всего таких сетей может быть 128 (от нуля до 127), а устройств в них — более 16 миллионов.

- Если первые два бита — это 10 (то есть 2 в десятичной системе счисления), значит, IP-адрес принадлежит к средней сети и использует два числа как указатель на неё.

У такого адреса уже зарезервировано два первых бита, а значит, для номера сети остаётся только 14 битов — это более 16 тысяч сетей и более 65 тысяч устройств.

- Если первые три бита — это 110, значит, компьютеру попался IP-адрес из маленькой сети, в качестве указателей на которую используются только три первых числа.

Всего таких сетей существует более двух миллионов, а подключаемых устройств в каждой — 256. Диапазон значений — от 192.0.0.0 и до 223.255.255.0 (223 — потому что у нас зарезервировано три бита).

Все эти виды IP-адресов имеют свои названия: класс A, B и C. Класс A — это большие сети, B и C — средние и маленькие. Кроме них существуют ещё сети класса D и E. В них входят зарезервированные адреса — например, 127.0.0.0 или 192.168.X.X. Первый указывает сам на себя — когда он отправляет данные

по этому адресу, они тут же приходят обратно (его ещё называют localhost). А второй — это стандартный идентификатор интернет-модемов и Wi-Fi-роутеров.

Бывает, что хостов в сети больше, чем доступных IP-адресов, — в современном интернете дела обстоят именно так. В этом случае интернет-провайдеры выдают устройствам адреса формата IPv6. При этом адрес IPv4 можно легко переделать в формат IPv6, а вот в обратную сторону это уже не работает.

Однако не все интернет-провайдеры перешли на новую версию IP-адресов, и это создало новую проблему: невозможно напрямую отправлять данные с устройств, поддерживающих IPv4, на устройства с IPv6. Проблему решили с помощью туннелирования — создали специальный канал между двумя устройствами, по которому обмениваются информацией между сетями с разными версиями протокола.

Маска подсети

Маска подсети — это более удобный способ разделить IP-адрес на номер сети и номер хоста. Она пришла на смену алгоритму, который мы описали выше. Маска подсети состоит из тех же четырёх чисел и похожа на IP-адрес:

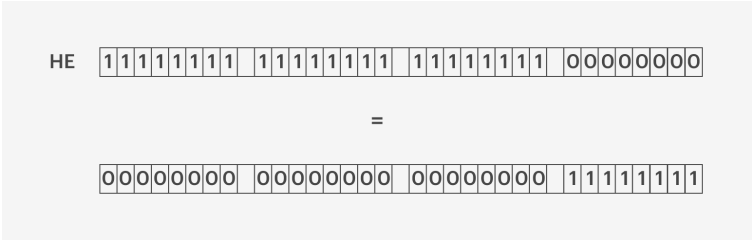
В двоичном представлении такая маска выглядит как 11111111 11111111 00000000 00000000. Нули показывают, где находится номер хоста, а единицы — номер сети.

Чтобы применить маску, нужно воспользоваться логическими операторами «И» и «НЕ».

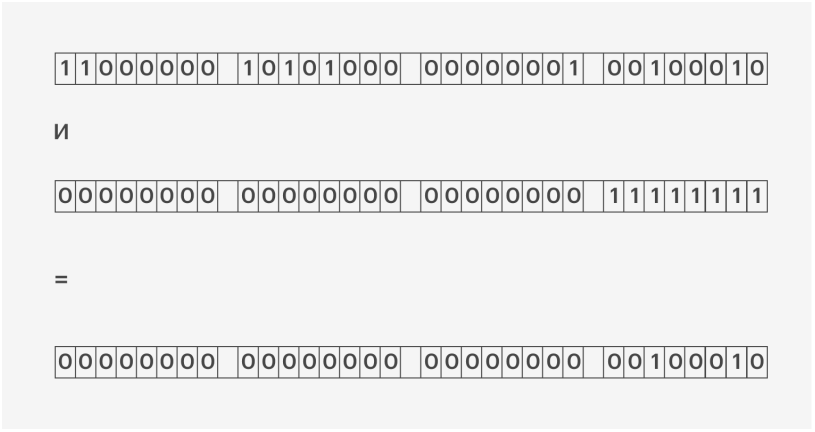
Давайте применим к IP-адресу 192.168.1.34 маску подсети 255.255.255.0:

На картинке показано, как мы сначала перевели IP-адрес и маску подсети в двоичную систему счисления. А затем побитово справа налево применили операцию логического «И». Маска помогла удалить ненужную часть адреса, и мы выделили номер сети — 192.168.1.0.

Чтобы выделить номер хоста, нужно сначала применить операцию логического «НЕ» к маске подсети, а затем — операцию логического «И» к IP-адресу и полученной маске:



Так мы получили маску для выделения номера устройства. А теперь применим операцию логического «И»:



У нас получился адрес 0.0.0.34. Это и есть номер хоста.

Как выбрать маску подсети

Обычно маска задаётся программистами в настройках серверов или пользователями в настройках системы.

Маска показывает, сколько битов включает в себя номер сети. Например, у большой сети номером будет только первое число (8 битов), а маска будет состоять из восьми единиц и 24 нулей: 255.0.0.0.

Если IP-адрес принадлежит к маленькой сети, то первые три числа в нём будут представлять номер сети. Значит, маска будет выглядеть так: 255.255.255.0.

Есть и слегка необычные маски подсетей — например, 255.255.254.0. Они тоже означают, сколько битов используется в номере сети. Только в данном случае их будет 23 — по 8 в первых двух числах и 7 в третьем. Остальные биты будут принадлежать номеру хоста.

Как ещё используют маски подсети

Выделять номера хостов и сетей удобно, но это не самая интересная часть использования масок. Их главная суперсила — умение разделять большие сети на несколько маленьких.

Допустим, у нас есть номер сети 185.12.0.0 с маской 255.255.0.0. В такой сети может быть более 65 тысяч устройств, чего вполне хватит, чтобы вместить все компьютеры в одном большом офисе.

Но что если у нас есть несколько маленьких офисов в одном здании, и мы хотим их все подключить к сети? Создавать новую сеть с 65 тысячами IP-адресов для

каждого офиса нерационально. Поэтому мы можем разбить сеть 185.12.0.0 на подсети.

Для этого вместо маски 255.255.0.0 мы возьмём маску 255.255.255.0. Так у нас появится 256 новых подсетей внутри одной большой. При этом в каждой подсети будет по 256 устройств.

Если в офисе понадобится больше устройств, мы можем взять другую маску — например, 255.255.254.0. И теперь нам будет доступно 512 устройств, а количество подсетей сократится до 128.

В компьютерных сетях **порт** или **номер порта** - это номер, присвоенный для уникальной идентификации конечной точки соединения и направления данных в определенную службу. На программном уровне, в операционной системе, порт - это логическая конструкция, которая идентифицирует конкретный процесс или тип сетевой службы. Порт на программном уровне идентифицируется для каждого транспортного протокола и комбинации адресов по присвоенному ему номеру порта. Наиболее распространенными транспортными протоколами, использующими номера портов, являются протокол управления передачей (TCP) и протокол пользовательских дейтаграмм (UDP); эти номера портов представляют собой 16-разрядные числа без знака.

Номер порта всегда связан с сетевым адресом хоста, таким как IP-адрес, и типом транспортного протокола, используемого для связи. Он завершает адрес назначения или источника сообщения. Определенные номера портов зарезервированы для идентификации конкретных служб, чтобы поступающий пакет можно было легко перенаправить запущенному приложению. Для этой цели номера портов меньше 1024 идентифицируют исторически наиболее часто используемые службы и называются хорошо известными номерами портов. Порты с большим номером доступны для общего использования приложениями и известны как эфемерные порты.

Порты обеспечивают мультиплексирование для нескольких служб или сеансов связи по одному сетевому адресу. В клиент-серверной модели архитектуры приложений для одной и той же службы может быть инициировано несколько одновременных сеансов связи.

Номер порта

Для TCP и UDP номер порта представляет собой 16-битное целое число без знака, которое может принимать значения от 0 до 65535. Для TCP номер порта 0 зарезервирован и не может быть использован, в то время как для UDP номер порта источника не является обязательным, а нулевое значение означает *отсутствие порта*. Процесс связывает свои входные или выходные каналы с помощью интернет-сокета, который представляет собой тип файлового дескриптора, связанного с транспортным протоколом, сетевым адресом, таким как IP-адрес, и номером порта. Это называется *привязкой*. Сокет используется процессом для отправки и получения данных по сети. Сетевому программному обеспечению операционной системы поручено передавать исходящие данные со всех портов приложений в сеть и перенаправлять поступающие сетевые пакеты процессам, сопоставляя IP-адрес и номер порта пакета с сокетом. В случае TCP только один процесс может быть привязан к определенной комбинации IP-адреса и номера порта. Распространенные сбои приложений, иногда называемые *конфликтами портов*, возникают, когда несколько программ пытаются использовать один и тот же номер порта на одном и том же IP-адресе с одним и тем же протоколом.

Приложения, реализующие общие сервисы, часто используют специально выделенные известные номера портов для получения запросов на обслуживание от клиентов. Этот процесс известен как *прослушивание* и включает в себя получение запроса на доступ к известному порту, что потенциально может привести к установлению диалога между сервером и клиентом один на один с использованием этого прослушивающего порта. Другие клиенты могут одновременно подключаться к одному и тому же прослушивающему порту; это работает, потому что TCP-соединение идентифицируется с помощью кортежа, состоящего из локального адреса, локального порта, удаленного адреса и удаленного порта.[1] Известные порты определяются по соглашению, контролируемому Управлением по присвоению номеров в Интернете (IANA). Во многих операционных системах для привязки приложений к этим портам требуются специальные привилегии, поскольку они часто считаются критически важными для работы IP-сетей. И наоборот, клиентский конец соединения обычно использует высокий номер порта, выделенный для краткосрочного использования, поэтому его называют временным портом.

Со́кет (англ. *socket* — разъём) — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой только сетью. Сокет — абстрактный объект, представляющий конечную точку соединения.

Следует различать **клиентские** и **серверные сокеты**. Клиентские сокеты грубо можно сравнить с конечными аппаратами телефонной сети, а серверные — с коммутаторами. Клиентское приложение (например, браузер) использует только клиентские сокеты, а серверное (например, веб-сервер, которому браузер посылает запросы) — как клиентские, так и серверные сокеты.

Интерфейс сокетов впервые появился в BSD Unix. Программный интерфейс сокетов описан в стандарте POSIX.1 и в той или иной мере поддерживается *всеми* современными операционными системами.

Принципы сокетов

Для взаимодействия между машинами с помощью стека протоколов TCP/IP используются адреса и порты. Адрес представляет собой 32-битную структуру для протокола IPv4, 128-битную для IPv6. Номер порта — целое число в диапазоне от 0 до 65535 (для протокола TCP).

Эта пара определяет сокет («гнездо», соответствующее адресу и порту).

В процессе обмена, как правило, используется два сокета — сокет отправителя и сокет получателя. Например, при обращении к серверу на HTTP-порт сокет будет выглядеть так: 194.106.118.30:80, а ответ будет поступать на mmm.nnn.ppp.qqq:xxxxx.

Каждый процесс может создать «слушающий» сокет (серверный сокет) и *привязать* его к какому-нибудь порту операционной системы (в UNIX непривилегированные процессы не могут использовать порты меньше 1024).

Слушающий процесс обычно находится в цикле ожидания, то есть просыпается при появлении нового соединения. При этом сохраняется возможность проверить наличие соединений на данный момент, установить тайм-аут для операции и т. д.

Каждый сокет имеет свой адрес. ОС семейства UNIX могут поддерживать много типов адресов, но обязательными являются INET-адрес и UNIX-адрес. Если привязать сокет к UNIX-адресу, то будет создан специальный файл (*файл сокета*) по заданному пути, через который смогут сообщаться любые локальные процессы путём чтения/записи из него (см. сокет домена Unix). Сокеты типа INET доступны из сети и требуют выделения номера порта.

Обычно клиент явно «подсоединяется» к слушателю, после чего любое чтение или запись через его файловый дескриптор будут передавать данные между ним и сервером.

Маршрутизация (англ. *Routing*) — процесс определения оптимального маршрута данных в сетях связи.

Маршруты могут задаваться административно (статические маршруты), либо вычисляться с помощью алгоритмов маршрутизации, базируясь на информации о топологии и состоянии сети, полученной с помощью протоколов маршрутизации (динамические маршруты). Маршрутизация в компьютерных сетях выполняется специальными программно-аппаратными средствами — маршрутизаторами; в простых конфигурациях может выполняться и компьютерами общего назначения, соответственно настроенными.

Статическими маршрутами могут быть:

- маршруты, не изменяющиеся во времени;
- маршруты, изменяющиеся по расписанию;

Для начала разберемся с понятием “динамическая маршрутизация”. До сего момента мы использовали так называемую статическую маршрутизацию, то есть прописывали руками таблицу маршрутизации на каждом роутере. Использование протоколов маршрутизации позволяет нам избежать этого нудного однообразного процесса и ошибок, связанных с человеческим фактором. Как понятно из названия, эти протоколы призваны строить таблицы маршрутизации сами, **автоматически**, исходя из текущей конфигурации сети. В общем, вещь нужная, особенно когда ваша сеть это не 3 роутера, а 30, например.
Помимо удобства есть и другие аспекты. Например, **отказоустойчивость**. Имея сеть со статической маршрутизацией, вам крайне сложно будет организовать резервные каналы — некому отслеживать доступность того или иного сегмента.
Протоколы динамической маршрутизации в течение нескольких секунд (а то и миллисекунд) узнают о проблемах на сети и перестраивают свои таблицы маршрутизации и в вышеописанном случае пакеты будут отправляться уже по актуальному маршруту
Ещё один важный момент — **балансировка трафика**. Протоколы динамической маршрутизации практически из коробки поддерживают эту фичу и вам не нужно добавлять избыточные маршруты вручную, высчитывая их.

Ну и внедрение динамической маршрутизации сильно облегчает **масштабирование сети**. Когда вы добавляете новый элемент в сеть или подсеть на существующем маршрутизаторе, вам нужно выполнить всего несколько действий, чтобы всё заработало и вероятность ошибки минимальна, при этом информация об изменениях мгновенно расходится по всем устройствам. Ровно то же самое можно сказать и о глобальных изменениях топологии.

Все протоколы маршрутизации можно разделить на две большие группы: внешние (**EGP** — Exterior Gateway Protocol) и внутренние (**IGP** — Interior Gateway Protocol). Чтобы объяснить различия между ними, нам потребуется термин “автономная система”. В общем смысле, автономной системой (доменом маршрутизации) называется группа роутеров, находящихся под общим управлением.
В случае нашей обновлённой сети AS будет такой:

Так вот, протоколы внутренней маршрутизации используются внутри автономной

системы, а внешние — для соединения автономных систем между собой. В свою очередь, внутренние протоколы маршрутизации подразделяются на **Distance-Vector** (RIP, EIGRP) и **Link State** (OSPF, IS-IS). Коренные различия между этими двумя видами состоят в следующем:
1) типе информации, которой обмениваются роутеры: таблицы маршрутизации у Distance-Vector и таблицы топологии у Link State,
2) процессе выбора лучшего маршрута,
3) количестве информации о сети, которое “держит в голове” каждый роутер: Distance-Vector знает только своих соседей, Link State имеет представление обо всей сети.