Как устроен интернет? Немного лжи, немного правды.

Основная единица сети — router/switch. Для успешной навигации по ней каждому устройству присваивается некоторый IP-адрес. Поговорим про него.

Протокол IP

Есть две версии: IPv4, IPv6. Отличия заключаются в различной длине адреса: 4 байта и 8, соответственно. Для простоты будем говорить про первый вариант.

IPv4

- Адрес — 4 байта (*.*.*.*, где * — число от 0 до 255)
- Так как адресов на всех уже давно не хватает, выделена группа “серых” адресов — таких, что по ним нельзя выйти в интернет:
  1. `192.168.*.*`
  2. `127.*.*.*`
  3. `255.255.255.255`
  4. `10.*.*.*`
- Зачем все это нужно? Устройство, непосредственно связанное с интернетом, при передаче запроса подменяет адреса на свой личный “белый” адрес и при получении ответа отправляет обратно по нужному адресу. То есть мы просто теперь умеем использовать больше устройств по достаточно малому числу адресов (~4 млрд. возможных адресов против ~16 млрд. устройств)
- Понятно, что за “белыми” адресами кто-то следит. Это компания InterNIC. Она же и получает деньги за пользование этими выделенными адресами. Это объясняет то, что провайдеры готовы предоставлять личные “белые” IP-адреса лишь за дополнительную плату.

Чуть подробнее про трансляцию

В локальной сети есть сетевая маска (NM). Зачем? Например, чтобы определять принадлежность одной сети: Пусть есть два устройства с адресами IP1, IP2. Тогда они принадлежат одной сети при IP1 & NM == IP2 & NM. То есть сетевая маска задаёт доступные IP локальной сети. Посмотрим, как мы будем устраивать соединение между этими адресами. Хотим переслать пакет от IP1 к IP2. Он проходит через ближайший роутер. Он смотрит, лежат ли они в одной подсети (указанным выше способом), и, если да, то соединяет их условно напрямую. Так как он не знает, какая именно из связанных с ним нод (будем называть их там) может передать этот запрос, то он пытается переслать его через всех (впоследствии он запоминает ноду, с которой пришёл ответ и шлёт его дальнейшие запросы уже через неё). Чтобы пакет не множился бесконечно, ему изначально присваиватся некоторый `TTL` (time to live) — число нод, через которое он может пройти (если не дошёл до точки назначения, а `TTL = 0`, то нода его дальше не передаёт). Теперь вторая ситуация: если надо слать не в локальной сети, то нужно подняться на уровень выше. На этот случай нода знает некоторый шлюз (gateway), через который это можно сделать. Дальше шлюз делает аналогичные действия, пока мы в итоге не окажемся в одной локальной сети, дальше похожим образом спускаемся. Ответ создаётся совершенно так же.

Пример

Что есть: - ноутбук, подключённый к Wi-Fi сети, с адресом 192.168.0.5 Что хочу: - отправить запрос на 8.8.8.8. Что происходит: - Точка перебирает все свои интерфейсы (тройка IP, NM, Gateway, кодирующая какую-то локальную

сеть) и смотрит, попадает ли желаемый адрес хотя бы в одну локальную сеть. Иначе точка использует Default Gateway - Для ноды, которой мы делегировали наш запрос, рекурсивно выполняется то же самое

Если всё адреса, почему мы пользуемся какими-то названиями (`vk.com`)?

Помимо прочего точка знает IP-адрес некоторого DNS-сервера, который знает, какому URL какой IP соответствует. Поэтому мы сначала посылаем DNS-серверу соответствующий запрос, а потом уже по полученному адресу. Если же наш DNS-сервер чего-то не знает, то он спрашивает то же самое у известного ему вышестоящего. Всего есть 12 корневых DNS-серверов, но они знают не всё, а только то, у какого DNS-сервера спросить про конкретный домен верхнего уровня (`.ru`, `.com`, …). DNS-запрос бывает двух типов: рекурсивный и нет (в первом случае мы кешируем на сервере запрашиваемый адрес для более быстрого доступа в дальнейшем).

Слои протоколов

- Application Layer – протоколы общения непосредственно сетевых приложений
  - POP3
  - SMTP
  - SSH
  - HTTP
  - FTP
  - …
- Transport Layer
  - TCP
  - UDP
- Network Layer
  - ICMP
  - IP
  - IGMP
  - ARP
  - RARP
- Link Layer – протокол передачи данных
  - PPP
  - Ethernet
  - Wi-Fi
  - LTE

Идея всего этого: максимальная абстракция. В самом деле, зачем нам знать, как передаются данные по сети, если мы пишем сетевое приложение?

При передаче из пакетов складывается некоторая “матрешка” (пакет оборачивается в пакет более низкого уровня): [MAC1 | MAC2 [IP1 | IP2 | … [Port1 | Port2 [APL] TL] Checksum]]

Порядок получения пакета:

- смотрим на MAC-адрес получателя: если не нам, обычно игнорируем, иначе распаковываем и передаём на уровень выше
- если нам, то смотрим на IP-адрес

Что тут интересно?

- MAC-адрес — 6-байтный относительно уникальный номер сетевой карты, записанный традиционно в 16-ричной форме.
  - Зачем нужен? Потому что у каждого интерфейса может быть свой IP-адрес. А также к одной сетевой карте можно привязать много IP-адресов. Да и на физическом уровне тоже хотелось бы иметь какую-то адресацию.
- MAC2 изначально не совсем известен – он определяется отдельным ARP-запросом по всей локальной сети:
  - [MAC1 | FF:FF:FF:FF:FF:FF | IP]
  - FF:FF:FF:FF:FF:FF — “широковещательный адрес” — отправка всем
  - обратно от правильного устройства будет послан запрос, уже содержащий нужный MAC-адрес
- А как работает определение MAC через шлюз?
  - MAC2 — MAC шлюза
  - IP2 — IP-получателя
  - Шлюз немного по-другому работает с запросами — он не выкидывает пакет с “не своим” IP, а просто поступает аналогично запросившему
- А что если в сети есть два устройства с одним MAC-адресом? Оба будут получать все пакеты друг друга.

Ещё абстракция!

По одному IP может быть несколько приложений. Поэтому вводится понятие порта — дополнительное число до 65535. Определяется протоколом. Использование портов контролируется операционной системой (не даёт слушать нескольким приложениям один и тот же порт).

Немного про TCP|UDP:

- первый пытается гарантировать передачу пакета (перепосылка пакетов, фоновое общение для “поддержания” соединения) — слишком много допонительной нагрузки
- второй же передаёт “как можно быстрее” — но тут уже может не всё приходить (зато вовремя)

Хорошо, но мы ничего не сказали про отправку из локальной сети!

Передавать “наружу” с “серым” IP нельзя - Подменять? Ни в коем случае! — непонятно, как обратно передавать ответ - Технология называется NAT: разворачиваем ещё на один слой и передаём с изменённым портом, который мы “присвоили” тому, чей запрос передаём. — Обратно тоже будем разворачивать и менять Всё очень плохо: это дорого по времени, особенно, если к нам много всего подключено. А ведь мы ещё можем захотеть разворачивать пакет до верхнего уровня (проксирование, фильтры).

Маршрутизаторы делятся на разные классы в зависимости от того, до какого уровня они могут разворачивать пакеты: Switch Layer {1, 2, 3}. Стоимость скалируется совсем нелинейно, так как разворачивать дорого, а пакетов много. Вплоть до того, что Switch Layer 1 нужно чуть ли не по процессору на соединение.

О всяких интересных командах

- ping — ну вы поняли
- traceroute (посылаем пакеты к заданному адресу с разными TTL и понимаем, на каком расстоянии от нас какой маршрутизатор получил наш запрос и отослал обратно соответствующий ICMP-запрос) — следует понимать, что никто не гарантирует один и тот же маршрут, но некоторое понимание происходящего мы всё же получим

- route (Unix) | route print (Windows) — просто таблица маршрутизации.