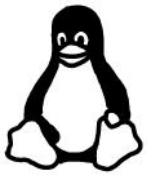

NETWORKING!
ACK!
Журнал о компьютерных сетях
Юля Эванс
О компьютерных сетях нужно знать ОЧЕНЬ много! О МЭ ГЭ!
Спокуха! Давай разберемся во всем по порядку.
В работе с сетями нет магии, но я реально чувствую себя волшебницей!

Действующие лица

У тебя дома

Твой ноутбучек (все картинки котиков там)

Твоя программа



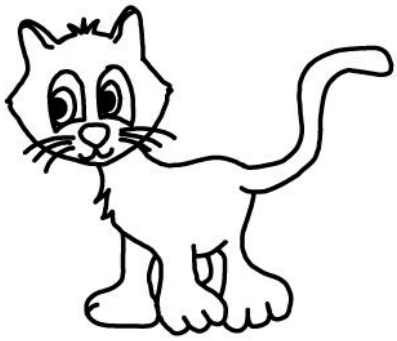
Операционная система (знает, как работать с сетью)

Твой домашний маршрутизатор

С этими компьютерами мы будем общаться

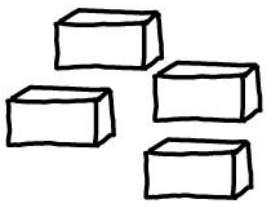
Сервер (тут лежит картинка котика)

DNS-сервер (знает на каком сервере хостится jvns.ca)

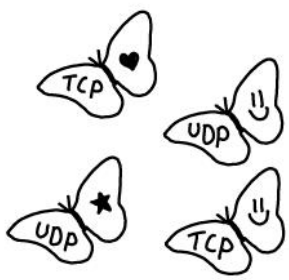


Та самая картика котика, которую мы хотим скачать

Между всем этим



Промежуточные маршрутизаторы в интернете



Пакеты!

Што здесь?!

Привет! Я – Джулия

Твиттер: @bOrk

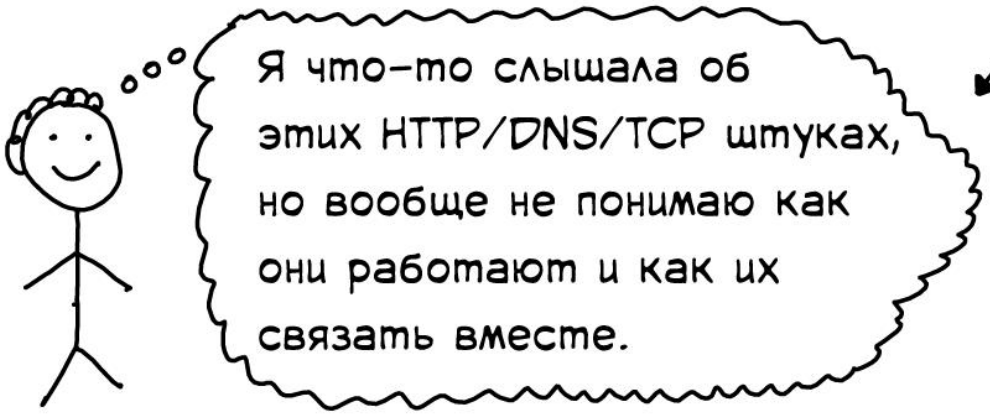
Бложик: http://jvns.ca

Я выложила картинку котика в интернете по этому адресу:

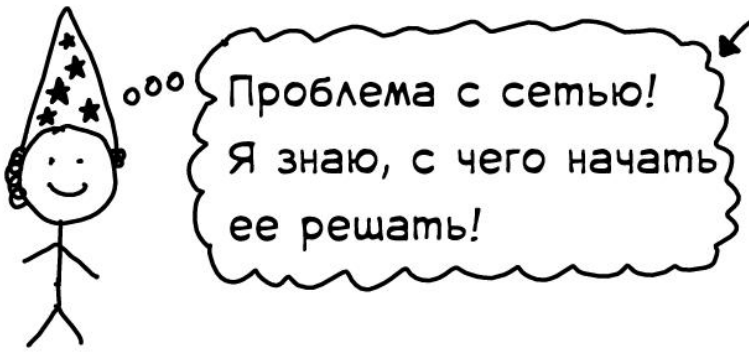
★ jvns.ca/cat.png ★ (можно смотреть!)

Из этого журнала ты узнаешь ВСЕ (но это не точно), что должно произойти, чтобы эта картинка попала с моего сервера к тебе на компьютер.

Моя задача–помочь тебе перестать думать так...



И начать думать вот так...



Наш главный герой: ПАКЕТ

Все данные в интернете передаются в ПАКЕТАХ. Пакет – это последовательность бит (0100101110011....), разделенная на секции (или "заголовки").

Вот как выглядит UDP-пакет, в котором написано "mangotea". В нем всего 50 байт! (400 бит)

84 бита

Входящий MAC	Исходящий MAC	Тип

Заголовок кадра локальной сети (14 байт)

4 байта (32 бита)

ver	hlen	TOS	Длина пакета
идентификация		flg	Смещение фрагмента
TTL	протокол	Контрольная сумма заголовка	
Исходящий IP-адрес			
Входящий IP-адрес			

160 бит
Заголовок IP 20 байт

Тут вся информация о том, на какой IP-адрес маршрутизатору нужно отправить пакет.

64 бита

Исходящий порт	Входящий порт
длина	Контрольная сумма UDP

Заголовок UDP 8 байт

(у TCP-пакета тут был бы TCP-заголовок)

m	a	n	g
o	t	e	a

Сюда заезжает все содержимое пакета. Каждый символ ASCII занимает 1 байт, поэтому "mangotea" = 8 байт
64 бита

Необходимые действия, чтобы скачать картинку котика

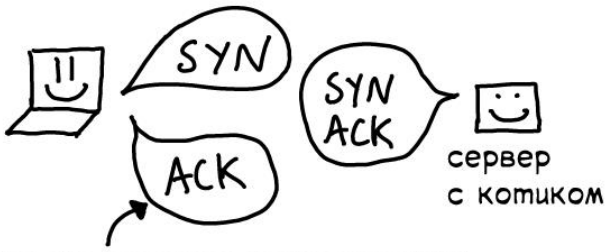
Конечно же с jvns.ca/cat.png

Когда ты загружаешь картинку, в сети выполняется ОЧЕНЬ много операций. Вот основные действия, в которых мы разберемся на следующих страницах.

1 Определяем IP-адрес jvns.ca

2 Открываем СОКЕТ

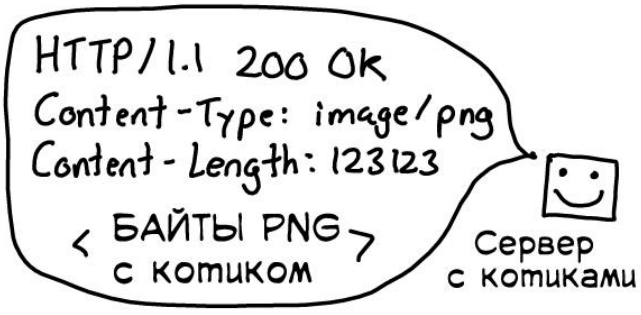
3 Открываем TCP-соединение к 104.28.7.94, порт 80



Это называется "рукопожатие"
Разберемся с этим на странице про TCP.

4 Запрашиваем котика

5 Получаем котика

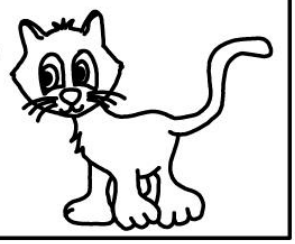


6 Прибираемся

-> Закрываем соединение, но это не точно,

-> складываем байты от PNG в файл, но это не точно,

-> смотрим на котика, инфа 100%.

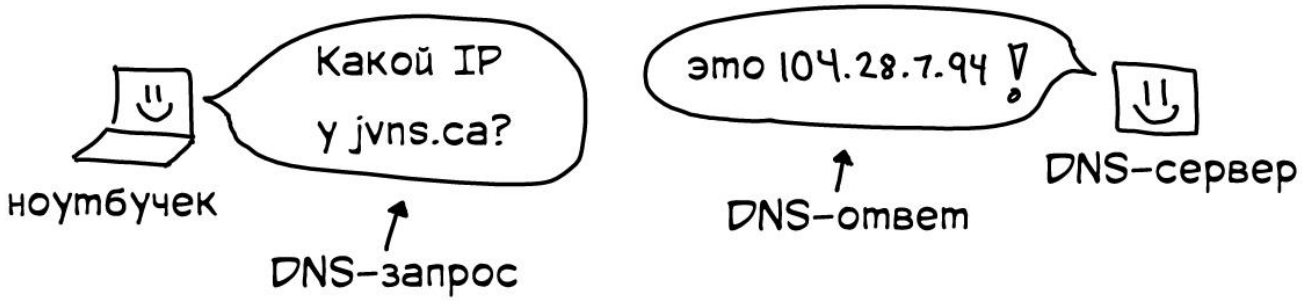


DNS

Шаг ① : определяем IP-адрес jvns.ca.

Вся сетевая активность происходит через передачу пакетов. Чтобы отправить пакет на сервер в интернете, тебе нужно знать его {IP-АДРЕС}, например 104.27.7.94.

Jvns.ca и firstvds.ru – это доменные имена. DNS ("Domain Name System", "Система Доменных Имен") – протокол, которым мы воспользуемся для определения IP-адреса доменного имени.



Обычно и DNS-запрос, и DNS-ответ, это UDP-пакеты.

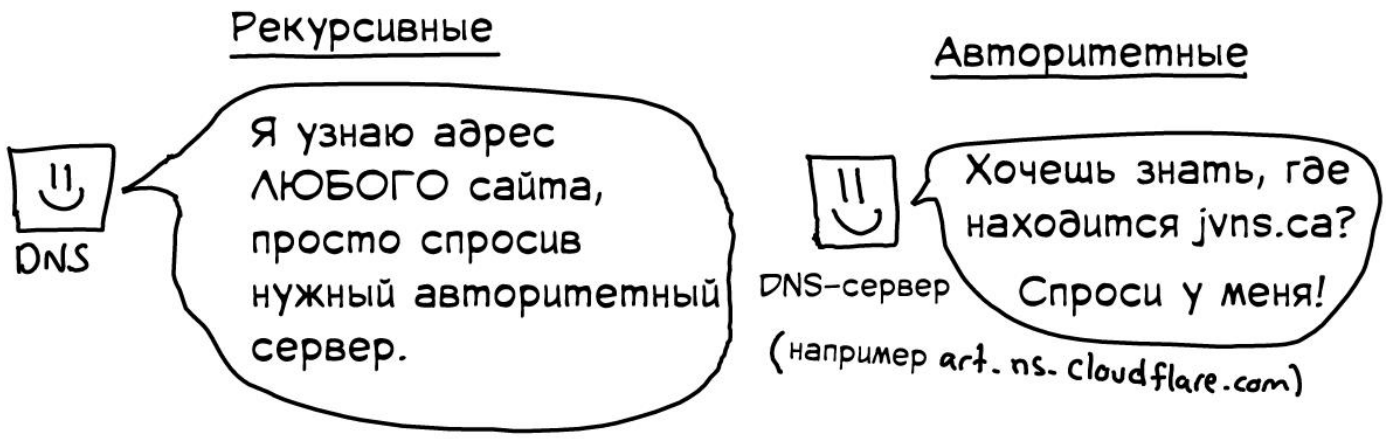
Когда ты запускаешь `$ curl jvns.ca/cat.png` :

Curl вызывает функцию "getaddrinfo" со значением "jvns.ca".	Getaddrinfo определяет системный DNS-сервер (например 8.8.8.8).	Getaddrinfo отправляет этому серверу запрос.	IP-адрес получен! 104.28.7.94

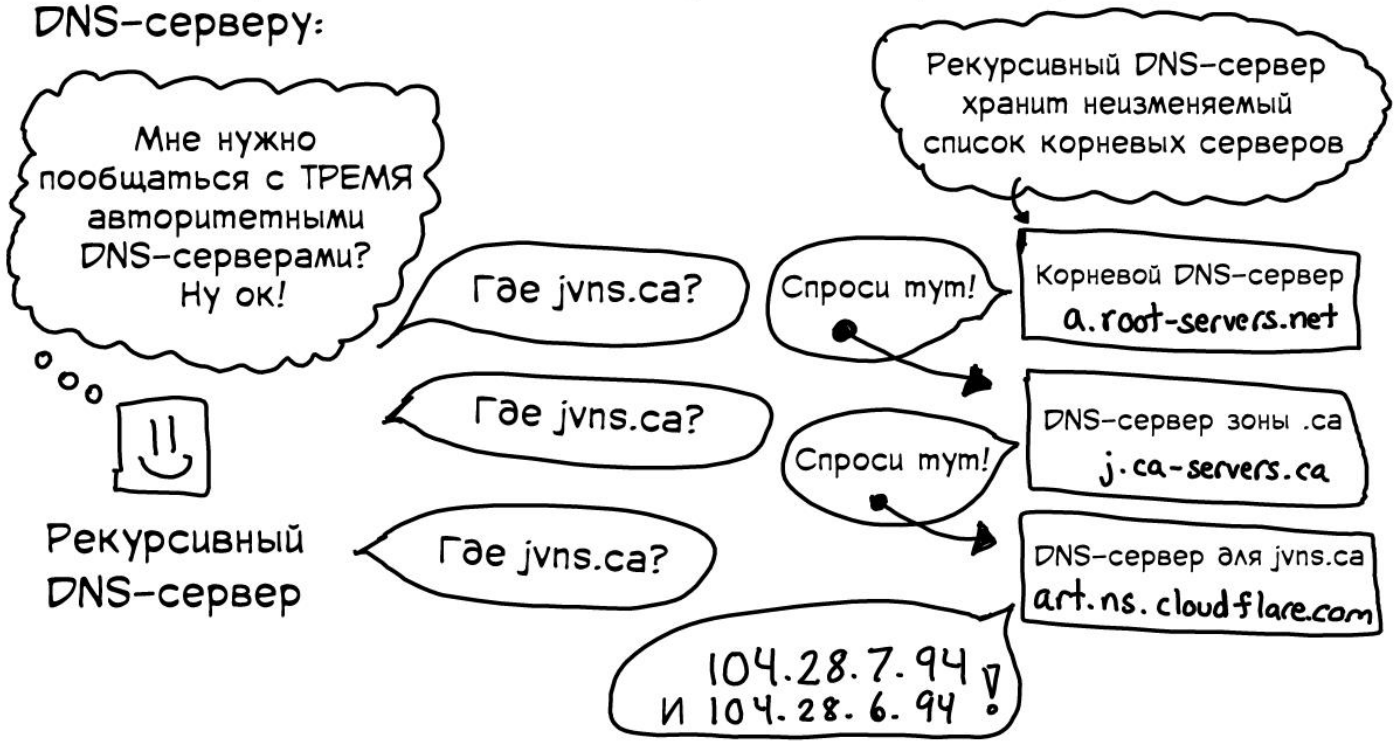
DNS-сервер по умолчанию обычно прописан в /etc/resolv.conf

8.8.8.8 – это DNS-сервер Гугла, и им пользуется куча народа. Отличный выбор для большинства задач!

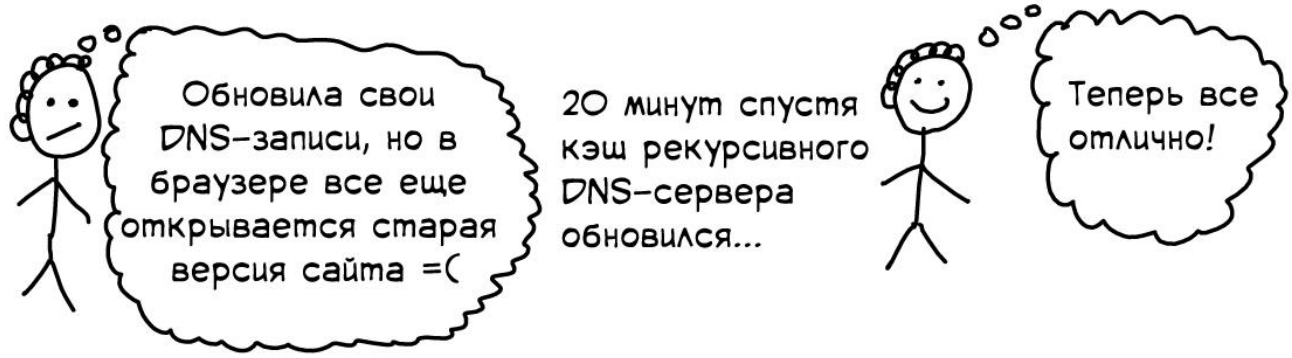
DNS-серверы бывают 2-х видов:



Вот что произойдет, если отправить запрос к рекурсивному DNS-серверу:



Рекурсивные DNS-серверы обычно кэшируют записи. У каждой DNS-записи есть TTL ("Time to Live", "время жизни"), который определяет время ее хранения в кэше. Обычно их нельзя заставить обновить свой кэш, придется просто ждать:



Давай делать ♡ DNS-запросы ♡

Когда настраиваешь DNS для нового домена, частенько случается следующее:

Чтобы понять, что происходит, можно отправлять DNS-запросы из командной строки:

```
$ dig jvns.ca

;; ANSWER SECTION
jvns.ca   268 IN A 104.28.6.94
jvns.ca   268 IN A 104.28.7.94

;; SERVER 127.0.1.1#53
```

У одного домена может быть МНОГО IP-адресов.

Эта запись истекает через 268 секунд.

"A" запись-это IP-адрес.

DNS-сервер, который я использую.

```
$ dig @8.8.8.8 jvns.ca
```

8.8.8.8-это рекурсивный DNS-сервер Google. @8.8.8.8 отправляет запрос этому серверу, вместо дефолтного.

```
$ dig +trace jvns.ca

.          502441 IN NS   h.root-servers.net
ca.        172800 IN NS   c.ca-servers.net
jvns.ca.   86400  IN NS   art.ns.cloudflare.com
jvns.ca.   300    IN A    104.28.6.94
```

корневой DNS-сервер!

Dig +trace делает практически то же самое, что и рекурсивный DNS-сервер, когда нужно определить IP твоего домена.

Это те 3 авторитетные сервера, с которыми должен связаться рекурсивный сервер, чтобы определить IP для jvns.ca.

СОКЕТЫ

Шаг (2): теперь, когда у нас есть IP-адрес, нам нужно открыть сокет! Давай узнаем, что это такое.

Твоя программа ничего не знает о TCP.

ХБЗ что такое TCP, мне просто нужно открыть страницу.

code.py
программа

Не боись, я могу помочь!

ОС

Чтобы использовать сокет мы:

Шаг 1: просим сокет у ОС.

Шаг 2: подключаем сокет к IP-адресу и порту.

Шаг 3: пишем в сокет, чтобы отправить данные.

4 стандартных вида сокетов

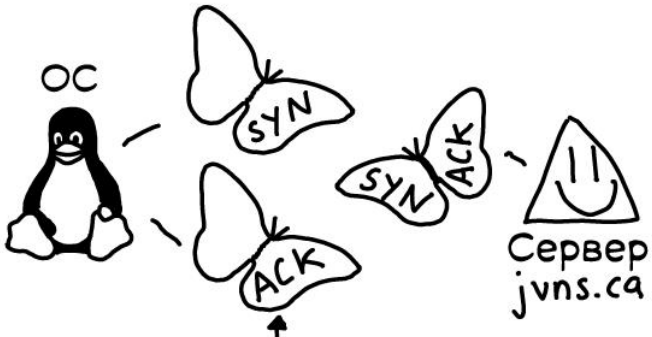
TCP
для TCP

UDP
для UDP

raw
для ЗАПРЕДЕЛЬНОЙ ВЛАСТИ. Ping пользует такой сокет, чтобы слать ICMP пакеты.

unix
чтобы общаться с программами на одном компьютере.

Когда ты соединяешься с TCP-сокетом.

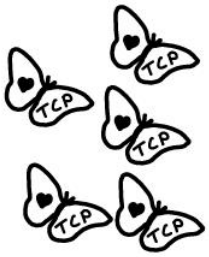


(разберемся в том, что такое SYN ACK, чуть позже)

Когда ты записываешь данные в сокет

code.py
программа
-> записывает гору данных ♥♥♥♥♥

делит данные на пакеты, чтобы отправить.



Интерфейс сокетов просто офигенен! Операционная система делает для меня очень много.

TCP: как гарантированно получить котика

Шаг ③ в нашем плане это "открываем TCP-соединение!". Выясним, что это же за штука такая "TCP".

Иногда пакеты теряются в интернете.

TCP поможет надежно передать поток данных, даже если пакеты потеряются или придут в случайном порядке.

Так ты хочешь знать, как работает TCP? Ну вот так!

Как узнать, в каком порядке должны приходить пакеты:

В каждом пакете есть информация о диапазоне находящихся в нем байт.

Вот такая:

once upon a ti ← байты 0-13
agical oyster ← байты 30-42
me there was a m ← байты 14-29

Получатель сможет собрать это в единое целое:

"once upon a time there was a magical oyster"

Положение первого байта (в нашем примере это 0,14,30) называется «номером последовательности».

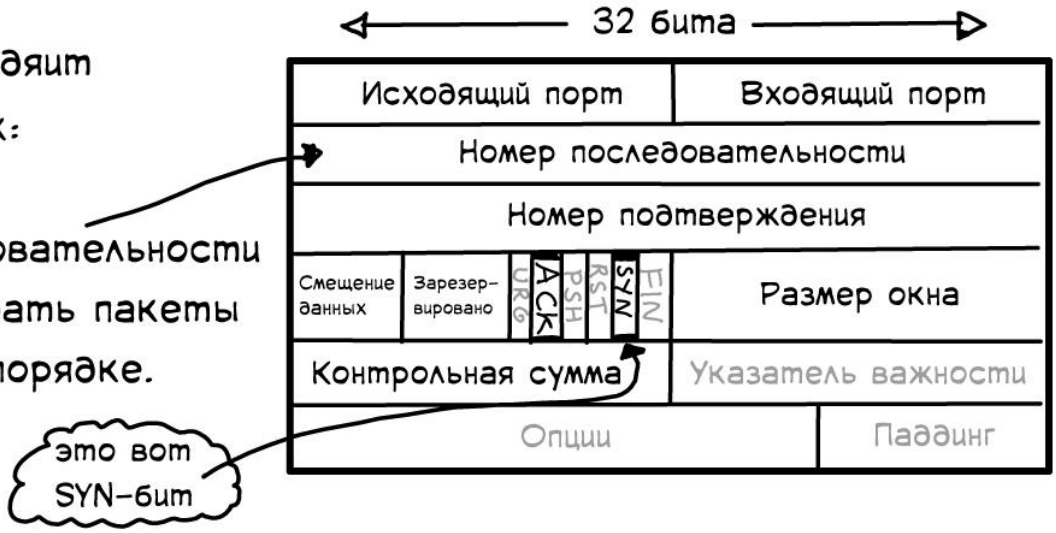
Как справиться с потерянными пакетами:

Принимая данные TCP, вам необходимо ACKтвердить (подтвердить) их: (ACK)

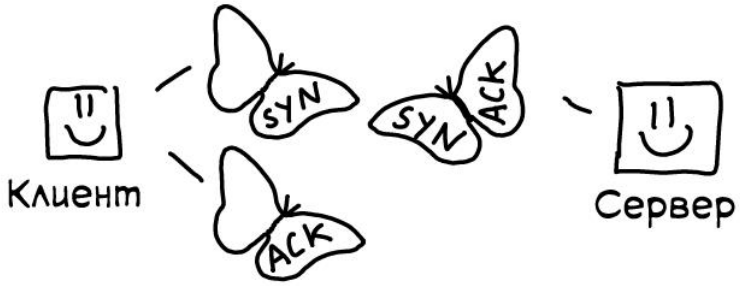
Если сервер не получит ACKтверждения, он отправит данные заново.

Рукопожатие TCP

Вот так выглядит TCP-заголовок:



Любое TCP-соединение начинается с "рукопожатия". Это нужно, чтобы обе стороны соединения могли общаться друг с другом.



Но что такое "SYN" и "ACK"? В TCP-заголовках можно установить 6 битовых флагов (SYN, ACK, RST, FIN, PSH, URG), которые видно на диаграмме выше. SYN-пакет – это пакет, в котором SYN флаг равен 1.

Если у тебя возникает ошибка типа "в соединении отказано" и "таймаут соединения", значит рукопожатие не завершилось!

Я выполнила `sudo tcpdump host jvns.ca` в одном терминале

и `curl jvns.ca` в другом.

Вот кусочек вывода:

```
localhost:51104 > 104.28.6.94:80   Flags [S]
104.28.6.94:80  > localhost:51104  Flags [S.]
localhost:51104 > 104.28.6.94:80   Flags [.]
```

TCP-рукопожатие

IP-адрес jvns.ca

S это SYN

. это ACK

HTTP

Шаг ④ : Наконец-то мы можем отправить запрос на cat.png!

Каждый раз, когда ты открываешь веб-страницу или картинку онлайн, используется HTTP.

HTTP очень простой текстовый протокол. Даже больше, он настолько простой, что ты можешь руками состряпать HTTP-запрос прямо сейчас. Давай попробуем!!!

Сначала давай создадим файл request.txt.

```
GET / HTTP/1.1
Host: ask.metafilter.com
User-Agent: zine
(в конце добавьте два пустых переноса строки)
```

Разберемся в Host части совсем скоро

После выполним:

```
cat request.txt | nc metafilter.com 80
```

Команда `nc` ("netcat") устанавливает TCP-соединение с metafilter.co и отправляет HTTP-запрос, который ты только что создал! В ответ придет вот такое:

```
200 OK
Content-Length: 120321
...заголовки...

Кучка всякого HTML
```

HTTP/2 – это следующая версия HTTP. Он очень сильно отличается, но у нас закончилось место.

Важность HTTP-заголовков

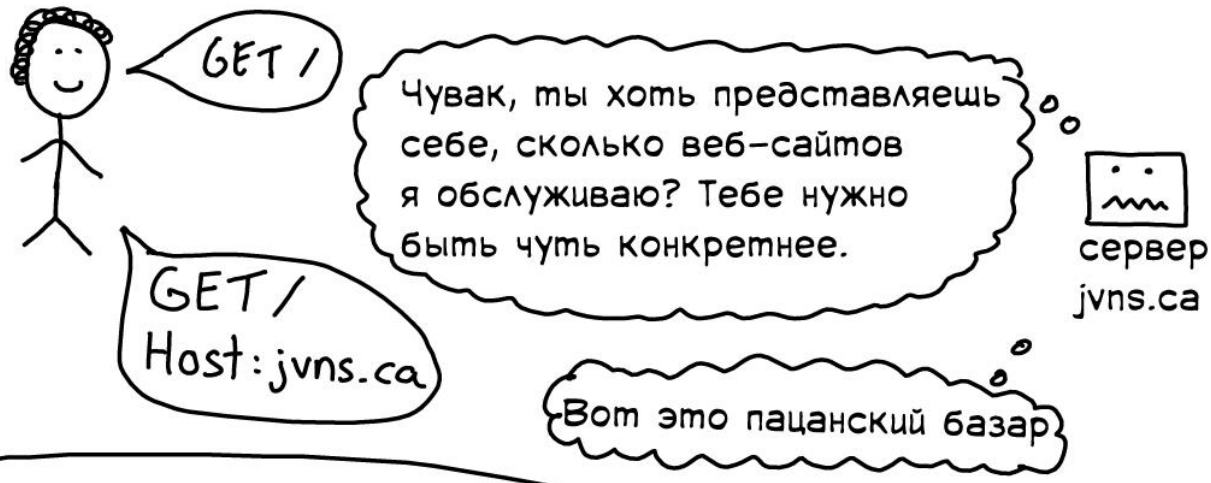
Вот так выглядит HTTP-запрос:

```
GET /cat.png HTTP/1.1
Host: jvns.ca
User-Agent: zine
```

Строки User-Agent и Host: называются "заголовками".

В них есть дополнительная информация для веб-сервера о том, какая страница тебе нужна.

Заголовок HOST ← мой любимый!



Большинство серверов обслуживает множество разных сайтов. Заголовок HOST позволяет выбрать нужный!

Серверы также отправляют заголовки ответа с дополнительной информацией о нем.

Еще полезные заголовки:

User-Agent

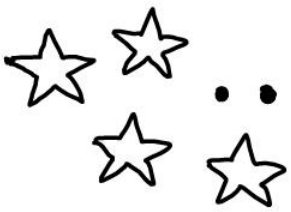
Куча серверов проверяют его, чтобы узнать, старый ли у тебя браузер или ты вообще бот.

Accept-Encoding

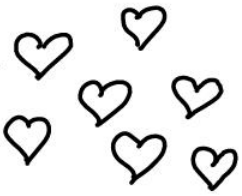
Хочешь сэкономить трафик? Установи тут "gzip" и сервер, возможно, сожмет свой ответ.

Cookie

Когда ты авторизовался на каком-то сайте, то твой браузер отправляет данные в этом заголовке! Так сервер узнает, что ты залогинен.



. . . а теперь у меня есть ЕЩЕ



Мы познакомились с основами того, как скачать картинку с котиком! Но вообще-то нужно знать много больше! Давай поговорим на еще несколько тем.

Познакомимся немного больше с сетевыми протоколами:

- Что же такое **порт**.
- Как же **пакет** собирается по частям.
- Безопасность: как работает **SSL**.
- Разные **сетевые слои**.
- **UDP** и почему он офигенен.

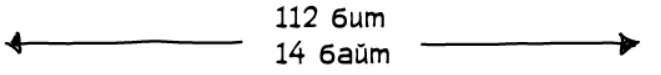
А также с тем, как пакеты перемещаются с места на место:

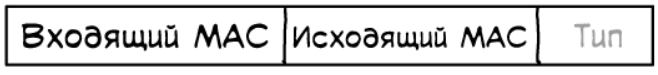
- Как отправляются пакеты в **локальной сети**.
- Как пакеты добираются **от твоего дома до jvns.ca**.
- Сетевая **нотация**.

Сетевые уровни

Считаю, что это не всегда полезно, но все-равно стоит знать, что такое "уровень 4".

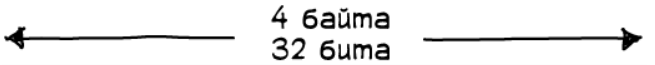
Сетевые уровни в основном относятся к разным секциям в пакете.





Уровень 1: Провода и радиоволны.

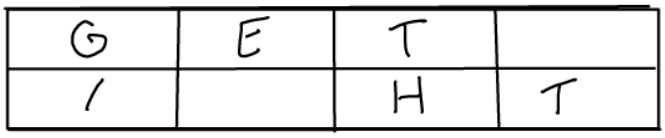
Уровень 2: Локальная сеть/wifi. Его понимает твоя сетевая карта.



Уровень 3: IP-адреса. Маршрутизаторы смотрят сюда, чтобы решить, куда отправлять пакет.

Уровень 4: TCP или UDP. Здесь определены порты.

Уровни 5+6: в целом их тут нет (хотя люди называют SSL "уровень 5").



Уровень 7: HTTP и всякое такое. Маршрутизаторы в основном игнорируют этот уровень. DNS-запросы, почта и другое находятся здесь.

Твой домашний маршрутизатор работает с уровнями 2+3+4

Сетевая утилита 3 уровня

↑

игнорирует уровень 4 и дальше

Я знаю только про IP-адреса! Не имею понятия, что такое порт и уж тем более ничего не знаю о содержимом пакета.

Приложения в основном переживают за 7 уровень, но они говорят ОС, какой IP и порт использовать.

Прикол в том, что уровни в основном не зависят друг от друга. Можно изменить IP-адрес (3 уровень) и не переживать за уровни 4+7.

Сетевая карта в твоем компьютере обращает внимание только на 1+2 уровни.

Что такое порт?

Порты – это часть протоколов TCP и UDP
(порт TCP 999 и порт UDP 999 не одно и то же).

Отправляя TCP-сообщение, ты хочешь передать его определенному типу программы.
Вот так будет не очень:

Мы хотим запускать разные типы программ на одном сервере:

Поэтому у каждого TCP-пакета есть номер порта в промежутке от 1 до 65535, которому он предназначается:

Netstat и lsof расскажут тебе, какие порты на компьютере используются.

Некоторые стандартные порты:

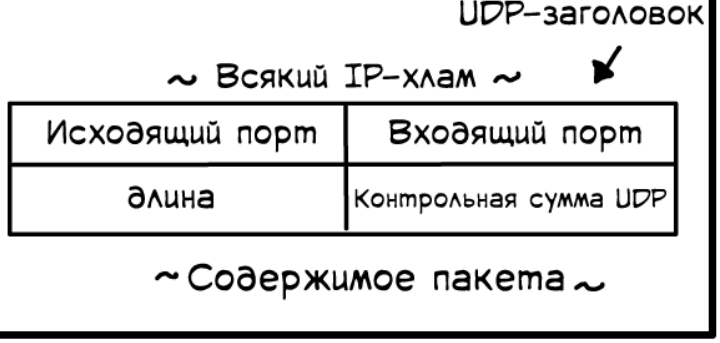
DNS: UDP порт 53
HTTP: TCP порт 80
HTTPS: TCP порт 443
SMTP (почта): TCP порт 25
Minecraft: TCP + UDP 25565

UDP

user datagram protocol

(протокол пользовательских датаграмм)

DNS-запросы отправляются через UDP. UDP очень простой протокол. Пакеты выглядят вот так:



"Протокол потерянных данных"

Отправляя UDP-пакеты они могут придти:

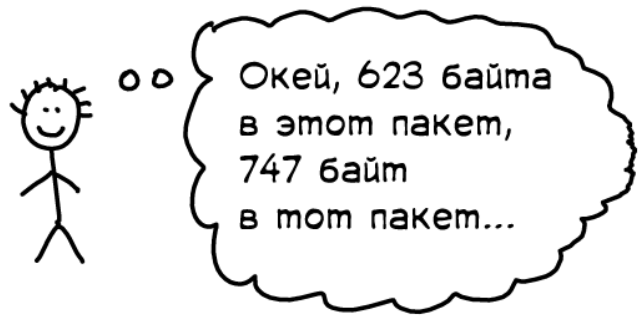
- Беспорядочно.
- Никогда.

На самом деле потеряться может любой пакет, но UDP не сделает ничего, чтобы тебе помочь.

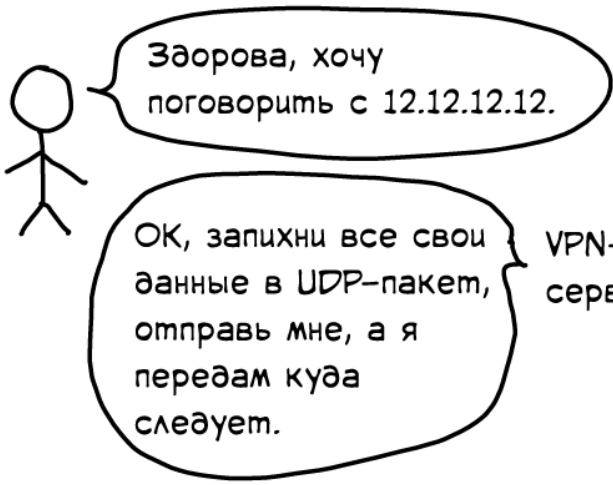
Размеры пакетов ограничены

*Размеры пакетов вообще суперски интересная тема. Гугли "MTU".

Придется решать, как организовывать данные в пакетах вручную:



VPNки пользуют UDP



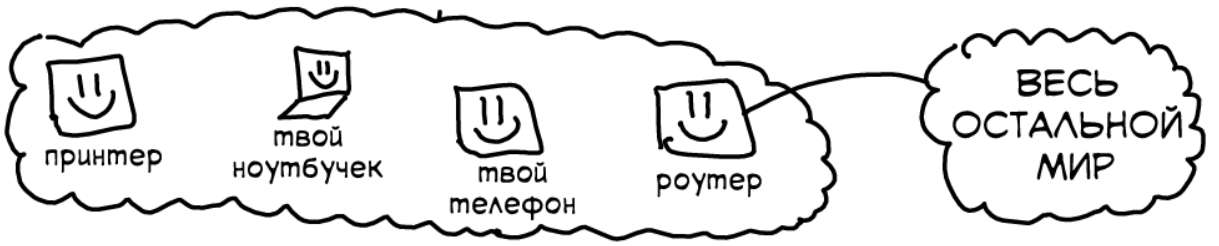
Потоковое видео часто передают по UDP.

Читай http://hpbn.co/webrtc (или просто по WebRTC), там ОФИГЕННОЕ обсуждение использования UDP как протокола в реальном времени.

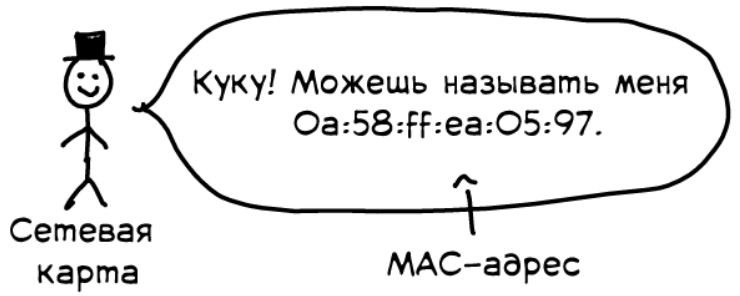
Локальные сети

Как поговорить с компьютером в этой же комнате

Каждый компьютер находится в подсети. Твоя подсеть–это все компьютеры, с которыми можно общаться напрямую.

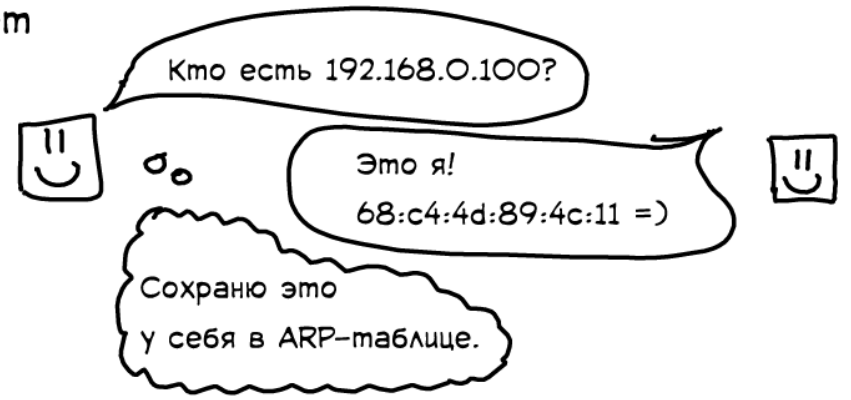


Что значит "разговаривать с компьютером напрямую"? Ну, у каждого компьютера в интернете есть сетевая карта с MAC–адресом.



У твоего ноутбучека поменяется IP–адрес, если подключиться к другой сети, но MAC останется прежним.

Отправляя пакет компьютеру в своей подсети, ты указываешь в нем его MAC–адрес. Чтобы определить правильный MAC, твой компьютер использует ARP–протокол (Address Resolution Protocol, протокол определения адресов).



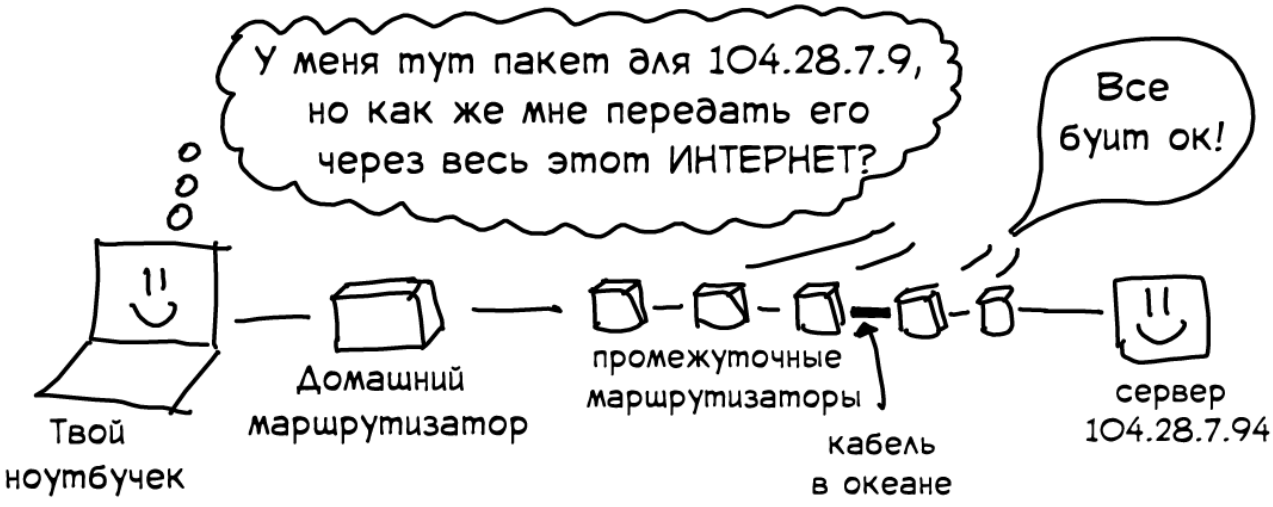
Выполни `arp -na` чтобы узнать содержимое ARP–таблицы на своем компьютере. Должно выглядеть как–то так:

```
$ arp -na
? (192.168.1.120) at 94:53:30:91:98:c8 [ether] on wlp3s0
```

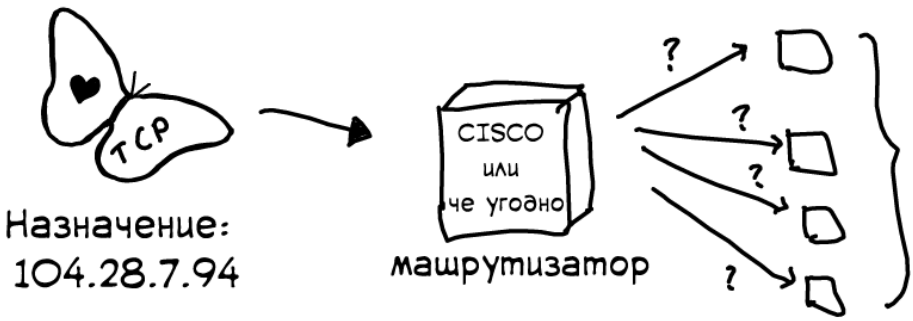
MAC для 192.168.1.120 (мой принтер)

моя wifi карта

Как пакеты отправляются за океан

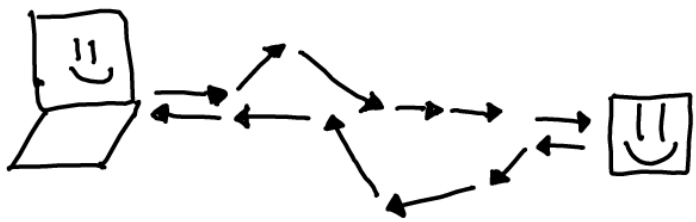


Когда пакет доезжает до маршрутизатора



Маршрутизаторы пользуются протоколом BGP, чтобы решить, на какой маршрутизатор передать пакет дальше.

Пакет может ездить по КУЧЕ разных путей, добираясь до одного и того же места назначения.



Путь пройденный им от A -> B, может отличаться от пути B -> A

УПРАЖНЕНИЕ: Запусти `traceroute firstvds.ru` и посмотри шаги, которые пройдет пакет до firstvds.ru.

Время нотаций!

10.0.0.0/8 132.5.23.0/24

Для обозначения групп IP-адресов человеки пользуются CIDR-нотацией.

Примеры CIDRов

CIDR	Диапазон IP
10.0.0.0/8	10.*.*.*
10.9.0.0/16	10.9.*.*
10.9.8.0/24	10.9.8.*

Важные примеры

10.0.0.0/8 и 192.168.0.0/16 и 172.16.0.0/12 зарезервированы для локальных сетей.

В CIDR-нотации "/n" обозначает 2^{32-n} IP-адресов.
Так что /24 будет означать $2^8 = 256$ IP.

Очень важно представлять группы IP-адресов как можно более эффективно, ведь у маршрутизаторов ОЧЕНЬ МНОГО РАБОТЫ.

Маршрутизатор

Принадлежит ли 192.168.3.2 подсети 192.168.0.0/16? Я могу провести очень быстрые битовые вычисления, чтобы это узнать!

10.9.0.0 в двоичном коде выглядит так:

00001010 00001001 00000000 00000000

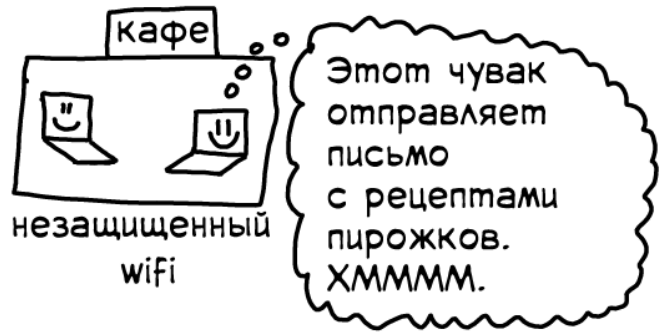
(первые 24 бита)

10.9.0.0/24 – это все IP-адреса с одинаковыми первыми 24 битами, что и 10.9.0.0!

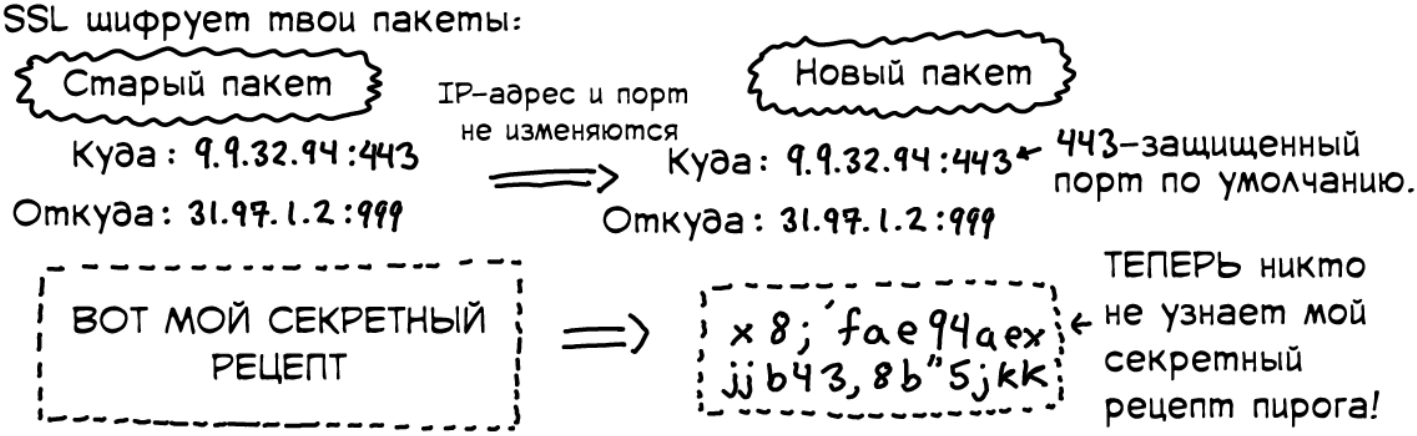
SSL / TLS

(TLS – новая версия SSL)

Когда ты отправляешь пакет через интернет, потенциально его могут прочитать ОЧЕНЬ МНОГИЕ.



SSL шифрует твои пакеты:



Вот что случится, если зайти на https://jvns.ca:

(очень упрощенно)

Когда клиент и сервер договорились о сессионном ключе, они могут шифровать любые данные, которые им только вздумается.

Чтобы посмотреть сертификат у jvns.ca выполните:

```
$openssl s_client -connect jvns.ca:443 -servername jvns.ca
```

TSL вообще очень непрост. Можно воспользоваться утилитой типа SSL Labs, чтобы проверить безопасность на твоем сайте.

wireshark

Wireshark – это ОФИГЕННАЯ тулза для анализа пакетов. Вот упражнения для ее изучения! Запусти:

```
sudo tcpdump port 80 -w http.pcap
```

Пока процесс выполняется, открой firstvds.ru в браузере. Нажми ctrl+c, чтобы остановить tcpdump. Теперь у нас есть pcap-файл!

Открой http.pcap в Wireshark.

Попытайся ответить на несколько вопросов:

① Какие HTTP-заголовки твой браузер отправил на firstvds.ru?

(подсказка: поищи `frame contains "GET"`)

② Сколькими пакетами ты обменялся с сервером firstvds.ru?

(подсказка: поищи `ip.dst == 212.109.222.30` тут напиши IP из "ping firstvds.ru")

С Wireshark можно легко разобраться:

- В IP-адресах и портах.
- SYNах и ACKах для TCP-трафика.
- Что вообще происходит с DNS-запросами.
- Да и вообще с кучей всего. Отличный способ разнюхать все и научиться чему-то.

♡ СПАСИБО ♡
ЧТО ПРОЧИТАЛИ

Если хотите узнать больше о работе сети:

→ Делайте сетевые запросы! Играйтесь с

dig traceroute tcpdump ifconfig
netcat wireshark netstat

→ Мануал от beej о сетевом программировании – это очень полезный и смешной гайд про API сокетов на UNIX-системах.

→ beej.us/guide/bgnet ←

→ High Performance Browser Networking

Просто ★ опупенный ★ и практичный гайд ко всему, что тебе потребуется знать о сетях, чтобы делать быстрые веб-сайты. Можно читать бесплатно здесь:

→ hpbn.co ←

Спасибо Камалу Маруби, Крису Каничу и Аде Мунро за проверку всего этого.

Обложку нарисовала офигительная Лиз Бэйли.

Перевела Команда FirstVDS.ru