

# NETWORKING!

A COMPUTER NETWORKING ZINE!

# ACK!

BY JULIA EVANS!



HEY, IT'S NOT SO BAD! LET'S LEARN THE IDEAS ONE AT A TIME!

NETWORKING ISN'T MAGIC, BUT I SURE FEEL LIKE A WIZARD NOW!



# cast of characters

in your house



your laptop  
(that you use  
to look at cats)

cool.py

your  
program



operating  
system  
(knows how to  
do networking)



your home  
router

computers you'll talk to

jvns.ca

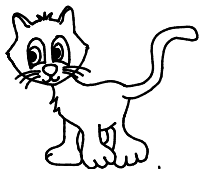


server  
(has cat  
picture)

DNS



DNS server  
(knows which  
server hosts  
jvns.ca)

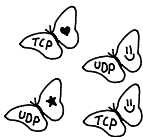


the cat picture  
we're downloading

in the middle



intermediate routers  
on the internet



packets!

# What's this?!

hi! I'm Julia



twitter: @b0rk

blog: <http://jvns.ca>

I put a picture of a cat on the internet here:

★ [jvns.ca/cat.png](http://jvns.ca/cat.png) ★ (go look!)

In this zine we'll learn everything (mostly) that needs to happen to get that cat picture from my server to your laptop.

My goal is to help get you from



I've heard about some of these HTTP/DNS/TCP things but I don't understand how they work exactly or how they all fit together

me after I'd been working as a web developer for a year

to...



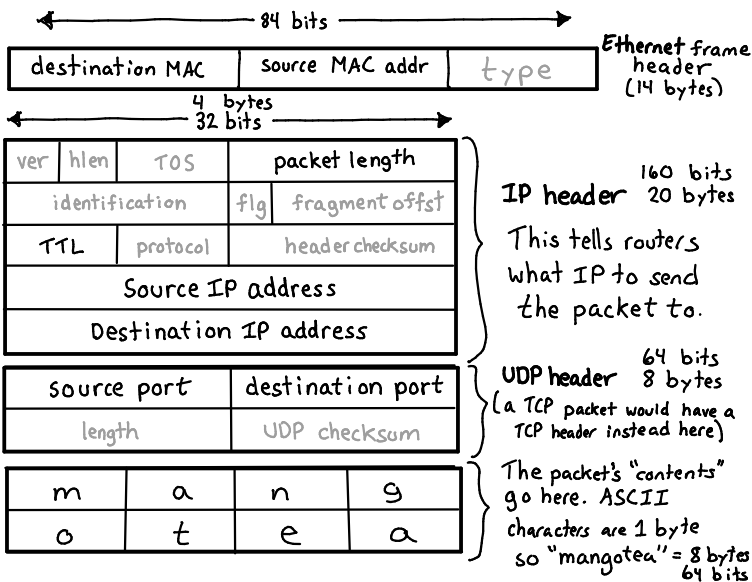
there's a networking problem! I totally know where to start!

me now

# our star: the packet

All data is sent over the internet in **packets**. A packet is a series of bits (010010111011....) and it's split into sections (or "headers")

Here's what a UDP packet that says "mangotea" looks like. It's 50 bytes in all!  
(400 bits)

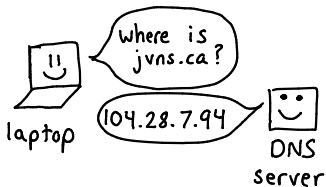


# steps to get a cat picture

from [jvns.ca/cat.png](http://jvns.ca/cat.png)

When you download an image, there are a LOT of networking moving pieces. Here are the basic steps we'll explain in the next few pages.

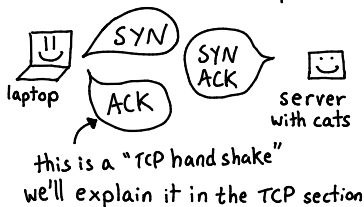
① get the IP address for `jvns.ca`



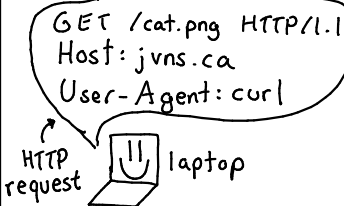
② open a socket



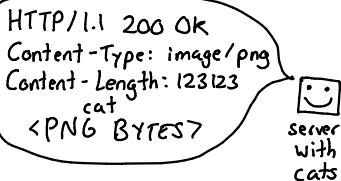
③ open a TCP connection to 104.28.7.94 port 80



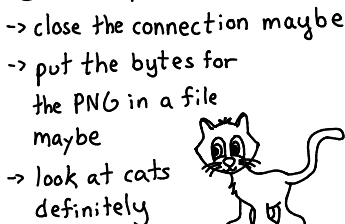
④ request a cat



⑤ get a cat back



⑥ clean up



# DNS

★ ★ Step ①: get the IP address for jvns.ca ★ ★  
★

All networking happens by sending packets. To send a packet to a server on the internet, you need an IP address like 104.28.7.94

jvns.ca and google.com are domain names. DNS (the "Domain Name System") is the protocol we use to get the IP address for a domain name.



The DNS request + response are both usually UDP packets.

When you run `$ curl jvns.ca/cat.png`:

curl calls the "getaddrinfo" function with "jvns.ca"	getaddrinfo finds the system DNS server (like 8.8.8.8)	getaddrinfo makes a request to that server	★ IP address: obtained! ★ 104.28.7.94
--	--	--	--

Your system's default DNS server is often configured in `/etc/resolv.conf`.

8.8.8.8 is Google's DNS server, and lots of people use it. It's a great choice!

There are 2 kinds of DNS servers:

recursive

authoritative



I can get you an IP address for ANY website by asking the right authoritative server



DNS server

(like art.ns.cloudflare.com)

wanna know where jvns.ca is?

Talk to ME!

When you query a recursive DNS server, here's what happens:

I have to talk to THREE authoritative DNS servers? Okay!



recursive DNS server

Where's jvns.ca?

ask there!

the recursive DNS server keeps a permanent list of root servers

root DNS server  
a.root-servers.net

Where's jvns.ca?

ask there!

.ca DNS server  
j.ca-servers.ca

Where's jvns.ca?

ask there!

jvns.ca DNS server  
art.ns.cloudflare.com

104.28.7.94  
and 104.28.6.94

Recursive DNS servers usually cache DNS records.

Every DNS record has a TTL ("time to live") that says how long to cache it for. You often can't force them to update their cache. You just have to wait:



I updated my DNS records, but when I visit the site in my browser I see the old version !!

20 minutes later after the recursive DNS server cache updates...



everything is great now

# let's make ♥ DNS requests ♥

When you're setting up DNS for a new domain, often this happens



I want  
jvns.ca



I don't know  
what that is yet  
(NXDOMAIN)



recursive  
DNS server

Here's how you can make DNS queries  
from the command line to understand  
what's going on:

```
$ dig jvns.ca
```

```
;; ANSWER SECTION
```

```
jvns.ca 268 IN A 104.28.6.94
```

```
jvns.ca 268 IN A 104.28.7.94
```

this record expires  
after 268 seconds

an "A" record is  
an IP address

there can  
be lots of  
IP addresses  
for one domain

```
;; SERVER 127.0.1.1#53
```

the DNS server I'm using

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

8.8.8.8 is Google's recursive  
DNS server. @8.8.8.8 queries that  
instead of the default.

```
$ 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
```

root DNS  
server!

dig +trace basically does the  
same thing a recursive DNS  
server would do to find your  
domain's IP

these are the 3 authoritative  
servers a recursive server has  
to query to get an IP for  
jvns.ca



# Sockets

Step ②: now that we have an IP address, the next step is to open a socket!  
Let's learn what that is.

your program doesn't know how to do TCP

idk what "TCP" is I just want to get a webpage

code.py  
program

OS  
don't worry!  
I can help!



what using sockets is like

step 1: ask the OS for a socket

step 2: connect the socket to an IP address and port

step 3: write to the socket to send data

4 common socket types

TCP

to use TCP

UDP

to use UDP

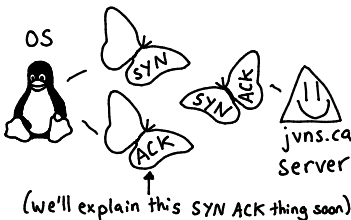
raw

for ULTIMATE POWER.  
ping uses this to send  
ICMP packets

unix

to talk to  
programs on the  
same computer

when you connect with a TCP socket



when you write to a socket

code.py  
program → writes lots of data ♥♥♥♥♥



→ splits it up into packets to send it



this socket interface is great! the operating system does so much for me!

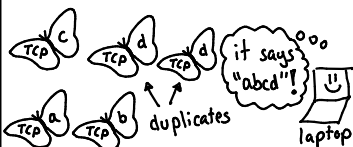
# TCP: how to reliably get a cat

Step ③ in our plan is "open a TCP connection!"  
Let's learn what this "TCP" thing even is ☺

When you send a packet on the internet sometimes it gets lost.



TCP lets you send a stream of data reliably even if packets get lost or sent in the wrong order.



how does TCP work, you ask? WELL!

how to know what order the packets should go in:

Every packet says what range of bytes it has

Like this:

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

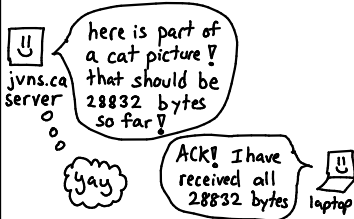
Then the client can assemble all the pieces into:

"once upon a time there was a magical oyster"

The position of the first byte (0, 14, 30 in our example) is called the "sequence number"

how to deal with lost packets:

When you get TCP data, you have to acknowledge it: (ACK)



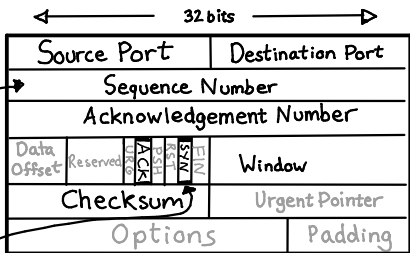
If the server doesn't get an ACKnowledgement, it will retry sending the data.

# The TCP Handshake

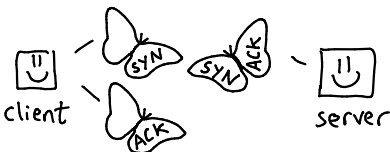
This is what a TCP header looks like:

the "sequence number" lets you assemble packets in the right order ☺

this is the SYN bit



Every TCP connection starts with a "handshake". This makes sure both sides of the connection can communicate with each other.



But what do "SYN" and "ACK" mean? Well! TCP headers have 6 bit flags (SYN, ACK, RST, FIN, PSH, URG) that you can set (you can see them in the diagram.) A SYN packet is a packet with the SYN flag set to 1.

When you see "connection refused" or "connection timeout" errors, that means the TCP handshake didn't finish!

I ran `!sudo tcpdump host jvns.ca!` in one terminal and `!curl jvns.ca!` in another. This is some of the output:

```
localhost:51104 > 104.28.6.94:80
104.28.6.94:80 > localhost:51104
localhost:51104 > 104.28.6.94:80
```

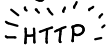
```
Flags [S]
Flags [S.]
Flags [.]
```

} TCP handshake!  
S is for SYN  
. is for ACK

jvns.ca IP address

# HTTP

Step ④: Finally, we can request cat.png!

Every time you get a webpage or see an image online, you're using 

HTTP is a pretty simple plaintext protocol. In fact, it's so simple that you can make a HTTP request by hand right now. Let's do it! ❗❗❗

First, let's make a file called request.txt

```
GET / HTTP/1.1
Host: ask.metafilter.com
User-Agent: zine
(put 2 newlines at the end)
```

we'll explain this Host: bit later

Then:

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

the `nc` command ("netcat") sets up a TCP connection to metafilter.com and sends the HTTP request you wrote! The response we get back looks like:

```
200 OK
Content-Length: 120321
... headers ...
a bunch of
HTML
```

HTTP/2 is the next version of HTTP. It's very different but we're out of space.

# important HTTP headers

This is a HTTP request:

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

The User-Agent: and Host: lines are called "headers".

They give the webserver extra information about what webpage you want!

the Host header ← my favorite!



GET /

GET /  
Host: jvns.ca

dude, do you even know how many websites I serve? You gotta be more specific.



NOW we're talking

Most servers serve lots of different websites. The Host header lets you pick the one you want!

Servers also send response headers with extra information about the response.

## More useful headers:

### User-Agent

Lots of servers use this to check if you're using an old browser or if you're a bot.

### Accept-Encoding

Want to save bandwidth? Set this to "gzip" and the server might compress your response.

### Cookie

When you're logged into a website, your browser sends data in this header! This is how the server knows you're logged in.

☆ ☆ ... and now for even MORE ☆ ☆ ☆ ☆ ☆ ☆

We've covered the basics of how to download a cat picture now! But there's a lot more to know! Let's talk about a few more topics.

We'll explain a little more about networking protocols:

- what a port actually is
- how a packet is put together
- security: how SSL works
- the different networking layers
- UDP and why it's amazing

and how packets get sent from place to place:

- how packets get sent in a local network
- and how packets get from your house to [jvns.ca](http://jvns.ca)
- networking notation



# networking layers



I don't always find this useful but it's good to know what "layer 4" means

Networking layers mostly correspond to different sections of a packet.

Layer 1: wires + radio waves

Layer 2: Ethernet/wifi protocol.  
Your network card understands it.

112 bits  
14 bytes

destination MAC	source MAC addr	type
-----------------	-----------------	------

4 bytes  
32 bits

ver	hlen	TOS	packet length	
identification		flg	fragment offset	
TTL	protocol	header checksum		
Source IP address				
Destination IP address				

source port		destination port	
length		UDP checksum	

G	E	T	
/		H	T

Layer 3: IP addresses

routers look at this a lot to decide where to send the packet next.

Layer 4: TCP or UDP

Where you get your ports!

Layer 5+6: don't really exist here (though people call SSL "layer 5")

Layer 7: HTTP and friends

Routers ignore this layer mostly. DNS queries, emails, etc. go here.

Your home router looks at layers 2+3+4

Your applications mostly worry about layer 7 but they get to tell the operating system what IP and port to use.

The network card in your computer only cares about layers 1+2.

layer 3  
networking  
tool

↑  
ignores layer  
4 and above

I only know about IP addresses!  
I don't even know what a port is let alone what the packet says

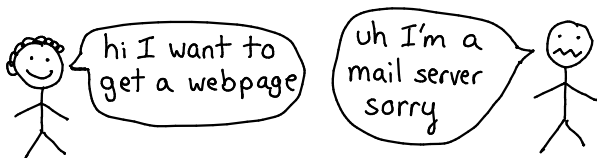
The cool thing is that the layers are mostly independent of each other - you can change the IP address (layer 3) and not worry about layers 4+7

# What's a port?

ports are part of the TCP and UDP protocols  
(TCP port 999 and UDP port 999 are different)

When you send a TCP message, you want to talk to a specific kind of program

This would be bad:



We want to have different kinds of programs on the same server:

minecraft DNS email

So every TCP packet has a port number between 1 and 65535 on it:



netstat and lsof can tell you which ports are in use on your computer

Some common ports:

DNS: UDP port 53  
HTTP: TCP port 80  
HTTPS: TCP port 443  
SMTP: TCP port 25  
(mail)  
Mine craft: TCP+UDP 25565



# UDP

User datagram protocol

DNS sends requests using UDP. UDP is a really simple protocol. The packets look like this:

UDP header

~ IP stuff ~

source port	destination port
length	UDP checksum

~ packet contents ~

"unreliable data protocol"  
(not what it really stands for)

When you send UDP packets, they might arrive

- out of order
- never

any packet can actually get lost, but UDP won't do anything to help you.

Packet sizes are limited



I'm gonna put 3000 characters in this packet

nope that won't fit. 1500 bytes is probably a better size.\*



\* packet sizes are actually a super interesting topic. Search "MTU"

you need to decide how to organize your data into packets manually



oo ok, 623 bytes in this packet, 747 bytes in that one...

VPNs use UDP



hi I want to talk to 12.12.12.12

OK stuff all your data into a UDP packet, send it to me, I'll pass it along.

VPN server

Streaming video often uses UDP

Read <http://hpbn.co/webbrtc> for a GREAT discussion of using UDP in a real time protocol.

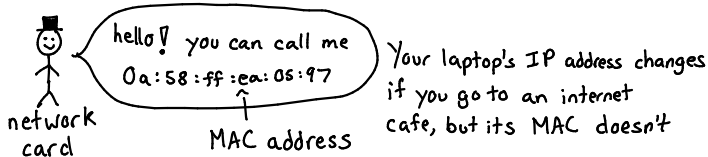
# Local networking

how to talk to a computer in the same room

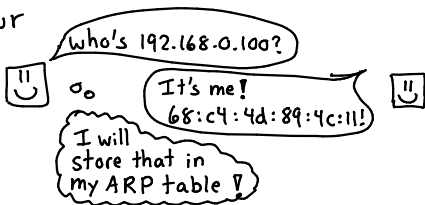
Every computer is in a subnet. Your subnet is the list of computers you can talk to directly.



What does it mean to talk "directly" to another computer? Well, every computer on the internet has a network card with a MAC address.



When you send a packet to a computer in your subnet, you put the computer's MAC address on it. To get the right MAC, your computer uses a protocol called ARP: (Address Resolution Protocol)

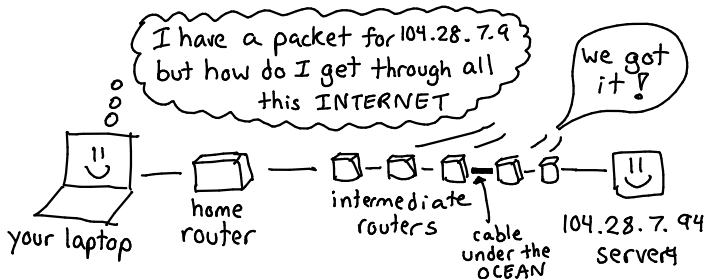


You can run `arp -na` to see the contents of the ARP table on your computer. It should look like this:

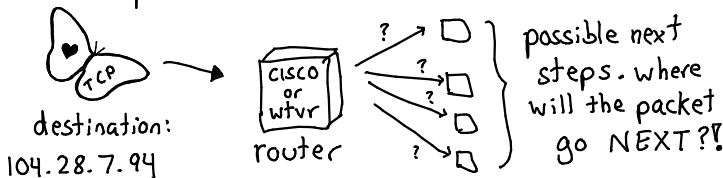
```
$ arp -na
? (192.168.1.120) at 94:53:30:91:98:c8 [ether] on wlp3s0
```

MAC for 192.168.1.120 (my printer) my wifi card

# How packets get sent across the ocean

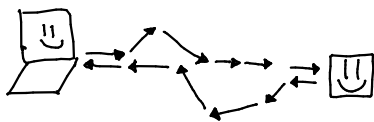


When a packet arrives at a router



Routers use a protocol called **BGP** to decide what router the packet should go to next:

A packet can take a lot of different routes to get to the same destination!



The route it takes to get from A → B might be different from B → A.

**Exercise:** Run `tracert google.com` to see what steps your packet takes to get to google.com.

# Notation time!

10.0.0.0/8

132.5.23.0/24

People describe groups of IP addresses using CIDR notation.

## example CIDRs

CIDR	range of IPs
10.0.0.0/8	10.*.*.*
10.9.0.0/16	10.9.*.*
10.9.8.0/24	10.9.8.*

## important examples

10.0.0.0/8 and 192.168.0.0/16 and 172.16.0.0/12 are reserved for local networking.

In CIDR notation, a /n gives you  $2^{32-n}$  IP addresses. So a /24 is  $2^8 = 256$  IPs.

It's important to represent groups of IP addresses efficiently because routers have LOTS TO DO.



000 is 192.168.3.2 in the subnet 192.168.0.0/16? I can do some really fast bit arithmetic and find out!

10.9.0.0 is this in binary:

00001010 00001001 00000000 00000000

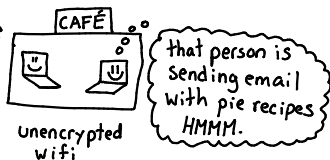
first 24 bits

10.9.0.0/24 is all the IP addresses which have the same first 24 bits as 10.9.0.0!

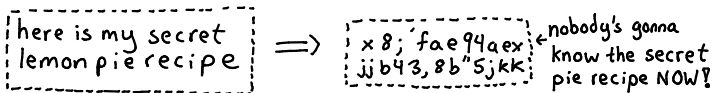
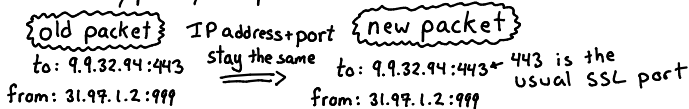
# SSL/TLS

(TLS: newer version of SSL)

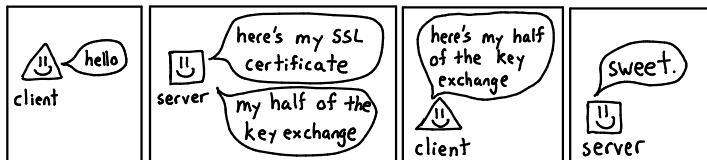
When you send a packet on the internet, LOTS of people can potentially read it.



SSL encrypts your packets:



What happens when you go to `https://jvns.ca:`



(very simplified)

Once the client and server agree on a key for the session, they can encrypt all the communication they want.

To see the certificate for `jvns.ca`, run:

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

TLS is really complicated. You can use a tool like SSL Labs to check the security of your site.

# wireshark

Wireshark is an amazing tool for packet analysis. Here's an exercise to learn it! Run this:

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

While that's running, open [metafilter.com](http://metafilter.com) in your browser. Then press Ctrl+C to stop tcpdump. Now we have a pcap!

Open `http.pcap` with Wireshark.

Some questions you can try to answer:

① What HTTP headers did your browser send to `metafilter.com`?

(hint: search `frame contains "GET"`)

② How many packets were exchanged with `metafilter.com`'s server?

(hint: search `ip.dst == 54.1.2.3`) put the IP from ping `metafilter.com` here

Wireshark makes it easy to look at:

- IP addresses and ports
- SYNs and ACKs for TCP traffic
- exactly what's happening with DNS requests
- and so much more. It's a great way to poke around and learn.

# ♡ thanks ♡ for reading

If you want to know more about networking:

→ make network requests! play with

dig traceroute tcpdump ifconfig

netcat wireshark netstat

→ beej's guide to network programming is a useful + funny guide to the socket API on Unix systems.

→ [beej.us/guide/bgnet](http://beej.us/guide/bgnet) ←

→ High Performance Browser Networking is a ★fantastic★ and practical guide to what you need to know about networking to make fast websites.

You can read it for free at:

→ [hpbrowser.com](http://hpbrowser.com) ←

Thanks to Kamal Marhubi, Chris Kanich, and Ada Munroe for reviewing this!

Cover art by the amazing Liz Baillie



like this ?

you can print more !  
for free !



<http://jvns.ca/zines>



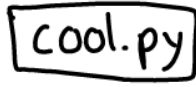


# Действующие лица

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

javns.ca



Сервер  
(тут лежит  
картинка котика)

DNS



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

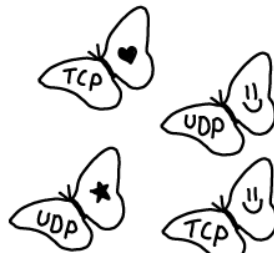


Та самая картинка котика,  
которую мы хотим скачать

Между всем этим



Промежуточные  
маршрутизаторы  
в интернете



Пакеты!

# Што здесь?!

Привет! Я – Джулия



Твиттер: @bOrk

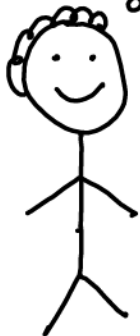
Бложик: <http://jvns.ca>

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...



Я что-то слышала об этих HTTP/DNS/TCP штуках, но вообще не понимаю как они работают и как их связать вместе.

Я после того, как проработала веб-разработчиком в течение года

И начать думать вот так...



Проблема с сетью!  
Я знаю, с чего начать ее решать!

А это я теперь

# ★ ★ Наш главный герой: ★ ★

## ПАКЕТ



Все данные в интернете передаются в **ПАКЕТАХ**. Пакет – это последовательность бит (010010111011....), 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.

Исходящий порт	Входящий порт
длина	Контрольная сумма UDP

64 бита  
Заголовок UDP 8 байт  
(у TCP-пакета тут был бы TCP-заголовок)

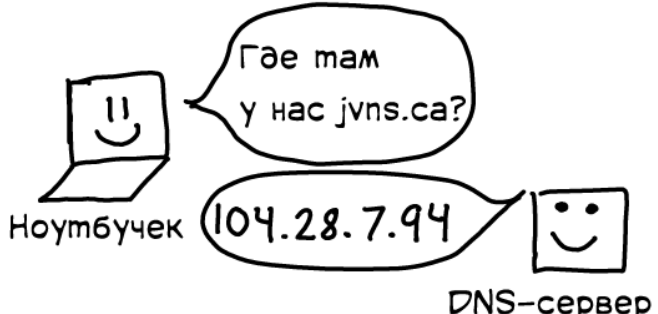
m	a	n	g
o	t	e	a

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 `javns.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 `javns.ca`



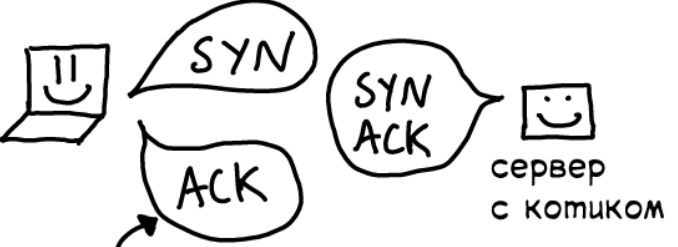
Ноутбучек: Где там у нас `javns.ca`?  
DNS-сервер: 104.28.7.94

② Открываем **СОКЕТ**



cool.py твоя программа: Хочу поговорить на TCP-шном с 104.28.7.94, порт 80.  
ось: Сейчас запилим.

③ Открываем TCP-соединение к 104.28.7.94, порт 80



сервер с котиком: SYN ACK  
ACK  
SYN


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

④ Запрашиваем котика



Ноутбучек: HTTP-запрос  
GET /cat.png HTTP/1.1  
Host: javns.ca  
User-Agent: curl

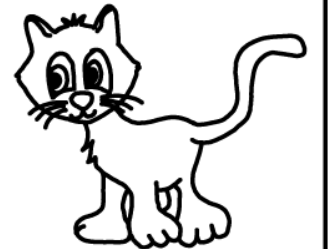
⑤ Получаем котика



Сервер с котиками: HTTP/1.1 200 OK  
Content-Type: image/png  
Content-Length: 123123  
< БАЙТЫ PNG >

⑥ Прибираемся

- > Закрываем соединение, но это не точно,
- > складываем байты от PNG в файл, но это не точно,
- > смотрим на котика, инфа 100%.

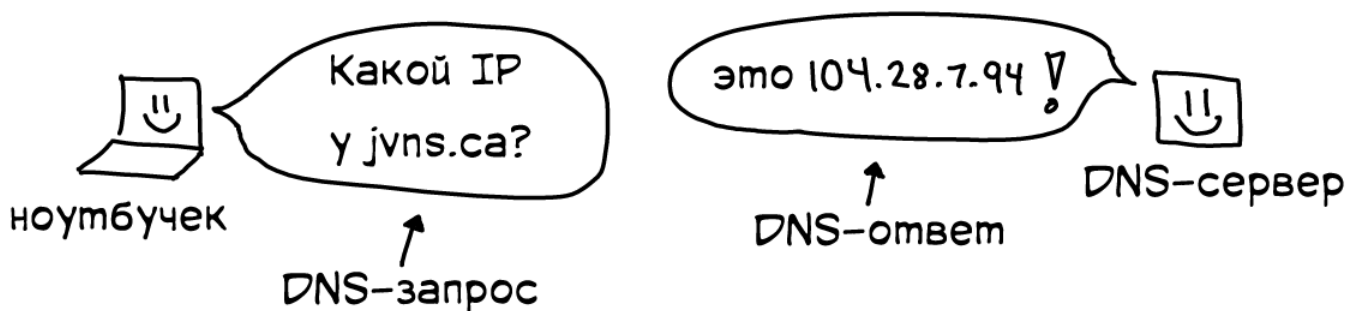


# DNS

☆ ☆ 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 мы 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:

## Рекурсивные



DNS

Я узнаю адрес ЛЮБОГО сайта, просто спросив нужный авторитетный сервер.

## Авторитетные

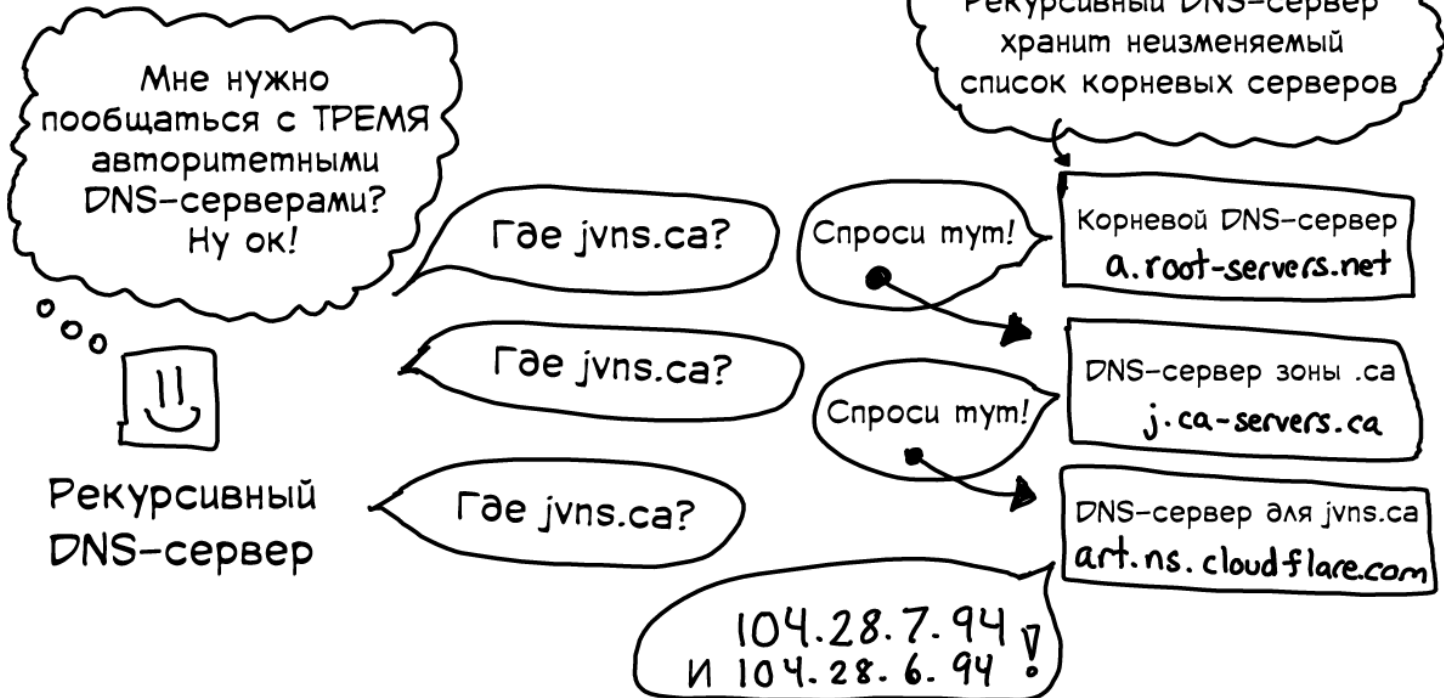


DNS-сервер

(например `art.ns.cloudflare.com`)

Хочешь знать, где находится `jvns.ca`?  
Спроси у меня!

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:



Обновила свои DNS-записи, но в браузере все еще открывается старая версия сайта =(

20 минут спустя кэш рекурсивного DNS-сервера обновился...



Теперь все отлично!

# Давай делать ♥ DNS-запросы ♥

Когда настраиваешь DNS для нового домена, частенько случается следующее:



Хочу увидеть  
jvns.ca.



Я еще не знаю,  
что это за домен  
(NXDOMAIN)



Рекурсивный  
DNS-сервер

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
```

Эта запись истекает  
через 268 секунд.

"A" запись-это  
IP-адрес.

У одного домена  
может быть  
МНОГО  
IP-адресов.

```
;; SERVER 127.0.1.1#53
```

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
```

```
. 50241 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 бойсь,  
я могу помочь!



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 использует  
такой сокет, чтобы  
слать ICMP пакеты.

zunix

чтобы общаться  
с программами  
на одном компьютере.

Когда ты соединяешься  
с TCP-сокетом.



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

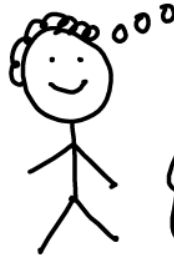
Когда ты записываешь данные  
в сокет

code.py  
программа

→ записывает гору  
данных ♥♥♥♥♥



делит  
данные  
на пакеты,  
чтобы  
отправить.



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



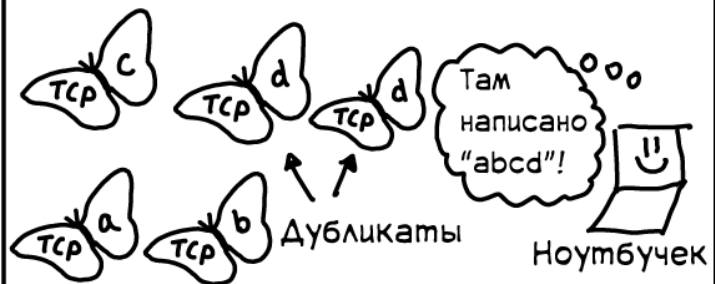
# TCP: как гарантированно получить котика

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

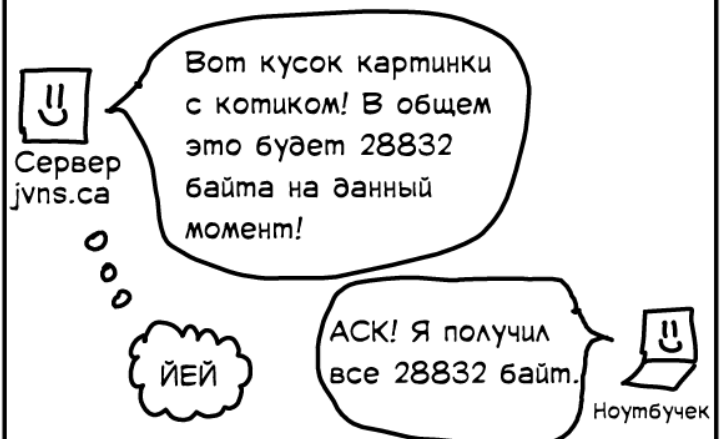
Получатель сможет собрать это  
в единое целое:

"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  
АСКтвердить (подтвердить) их: (ACK)



Если сервер не получит АСКтверждения,  
он отправит данные заново.

# Рукопожатие TCP

Вот так выглядят

TCP-заголовок:

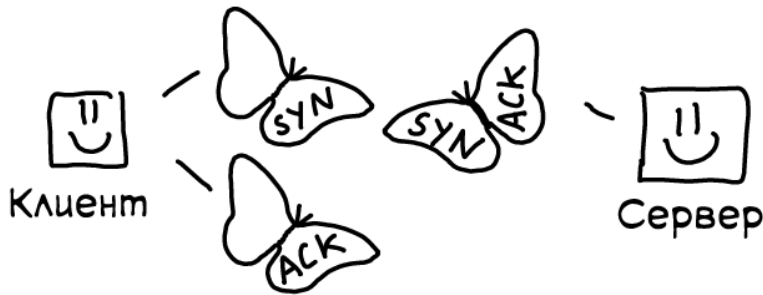
Номер последовательности позволит собрать пакеты в правильном порядке.

это вот SYN-бит

← 32 бита →



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
104.28.6.94:80 > localhost:51104
localhost:51104 > 104.28.6.94:80
```

```
Flags [S]
Flags [S.]
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 мы только что создали! 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!



GET /

GET /  
Host: jvns.ca

Чувак, ты хоть представляешь себе, сколько веб-сайтов я обслуживаю? Тебе нужно быть чуть конкретнее.



сервер  
jvns.ca

Вот это пацанский базар

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](http://jvns.ca).
- Сетевая нотация.

