CSCI-1680
DNS + (CC WRAPUP)

Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, Scott Shenker and John Jannotti

Administrivia

- TCP milestone I: this week, sign up for a meeting if you haven't
  - If you're stuck: bring what you have, it does not need to be perfect
  - DO NOT just hack stuff together to make it look good in Wireshark
- TCP Gearup II: TONIGHT, 10/31 6-8pm in CIT 368
  - Prep for milestone II
- HW3 (short!): Due next Thurs

↳ PREP FOR MILESTONE II

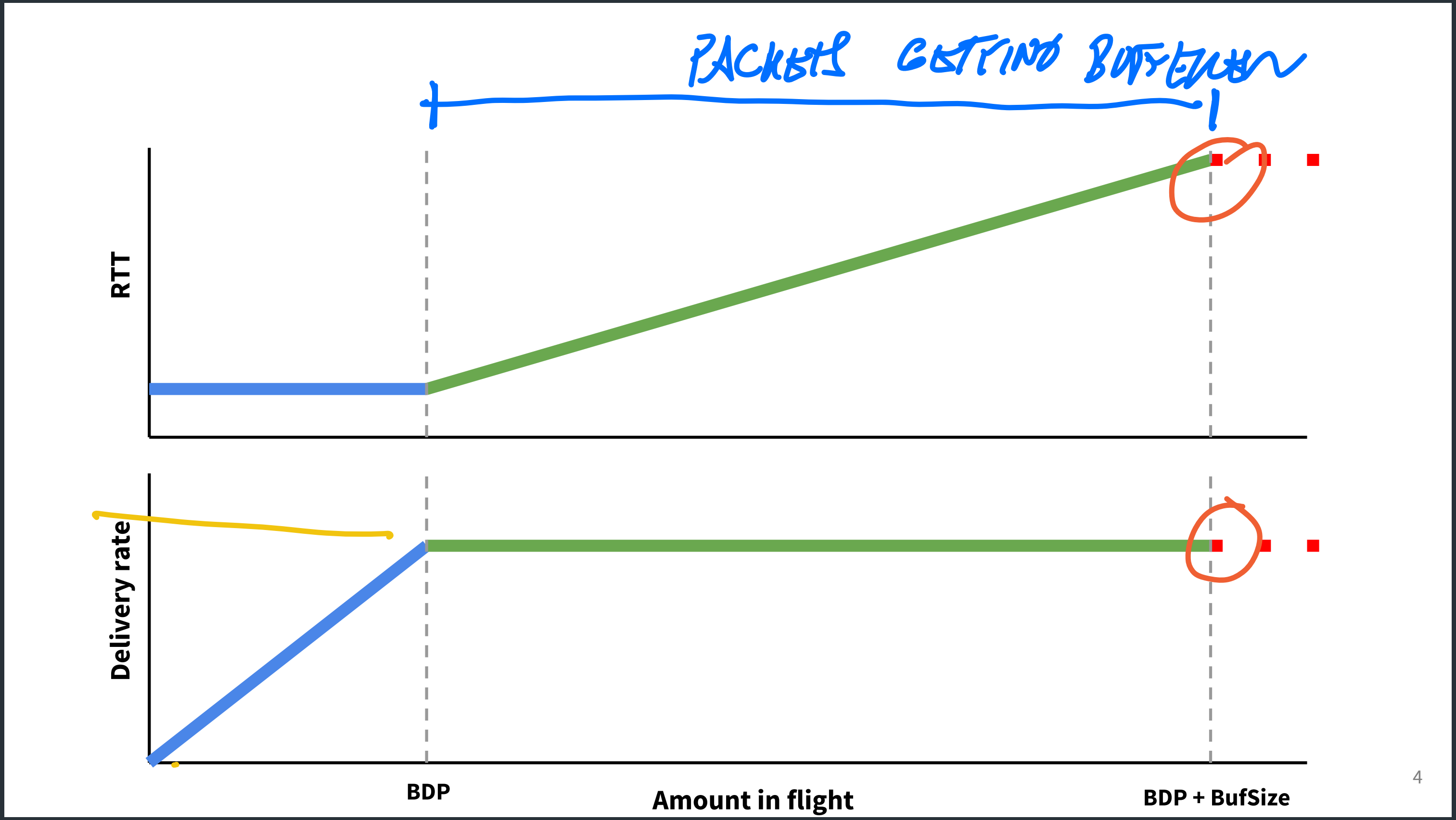
Warmup

WITHIN NETWORK

Which of the following contribute to congestion?

a. Packets queueing up at switches ✓
b. High CPU usage on the receiver ⇒ FLOW CONTROL
c. Many TCP connections sending on the same link ✓
d. Many UDP connections sending on the same link ✓
e. An unreliable Wifi link ✗

Thinking about congestion



"BBR congestion control"

The basic principle

Signals from the network
(ACKs, other TCP packet info, more...)

↓

Congestion Control (CC) algorithm

↓

Congestion window: cwnd

↓

Sender can send: min(advertised window, cwnd)
(Advertised window: flow control window from receiver)

⇒ Different CC algorithms use different signals, different techniques for adapting cwnd, but most fit this format

Lots of CC variants designed with different strategies and goals

Network Signals

- Packet loss (“loss-based”)
- Delay/RTT (“delay-based”)
- ”Marks” added on packets by routers

Goals

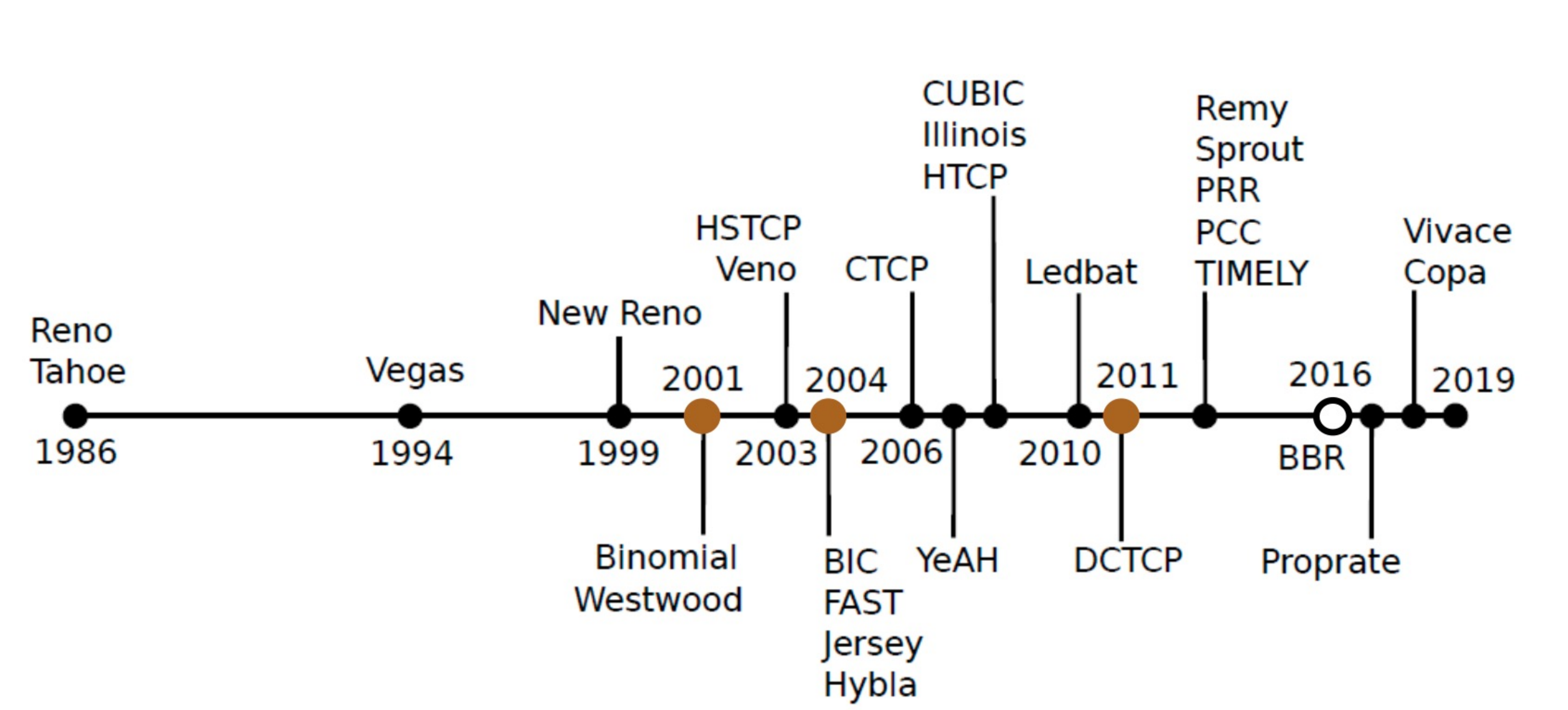
- Maximize throughput
- Recover from packet loss or high RTT
- Short-long “flows”
- Datacenter-specific (low-latency)

$\Rightarrow$ This is a big research area!

This is just the beginning…

Lots of congestion control schemes, with different strategies/goals:

- Tahoe (1988)
- Reno (1990)
- Vegas (1994): Detect based on RTT
- New Reno: Better recovery multiple losses
- Cubic (2006): Linux default, window size scales by cubic function
- BBR (2016): Used by Google, measures bandwidth/RTT

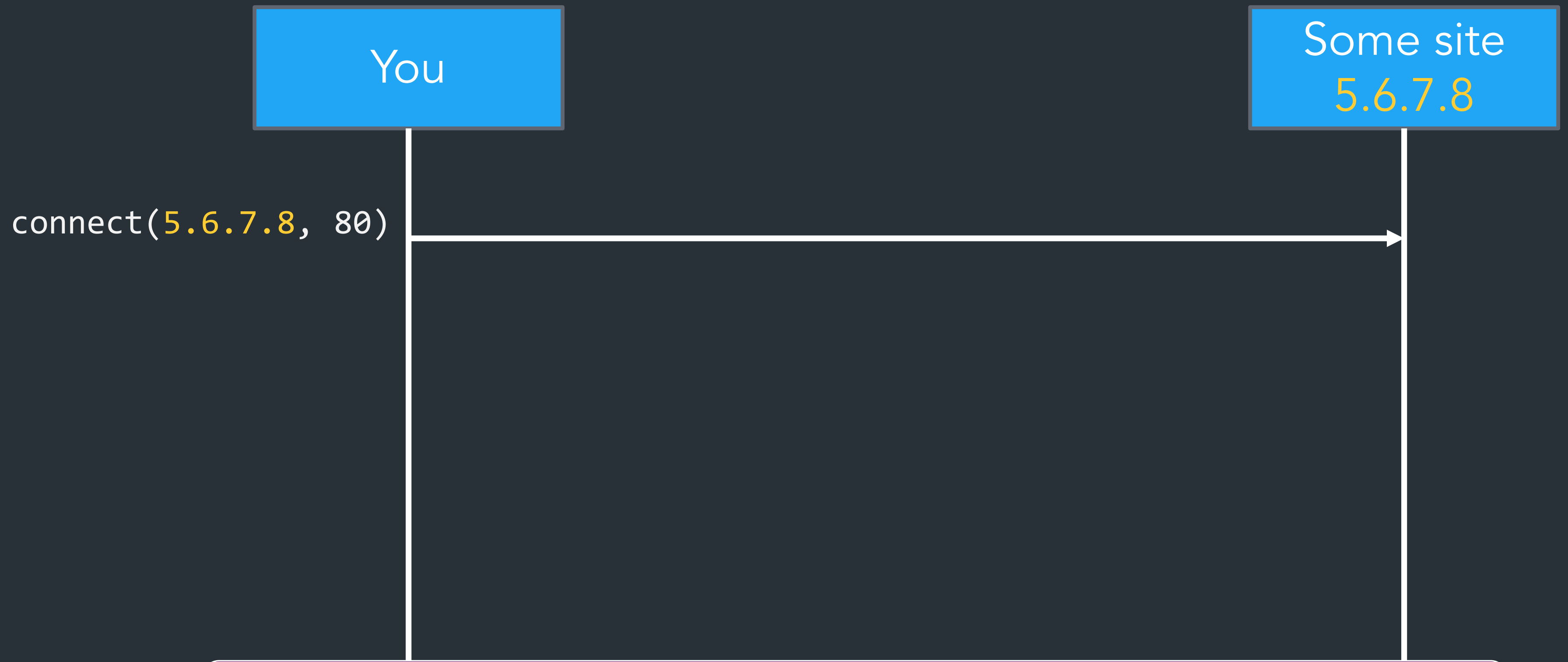


CC is a big (and active!) research area! For more on this and other network performance research, I recommend checking out CSCI 2680.

DNS

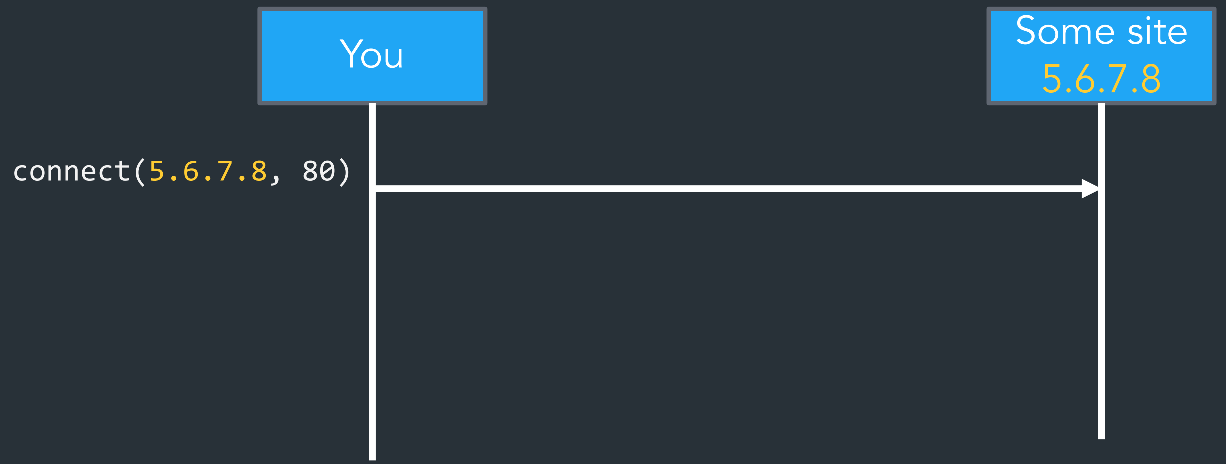
Connecting to a server: the story so far

POV: You want to connect to some website



Is this how users interact with the network? No!

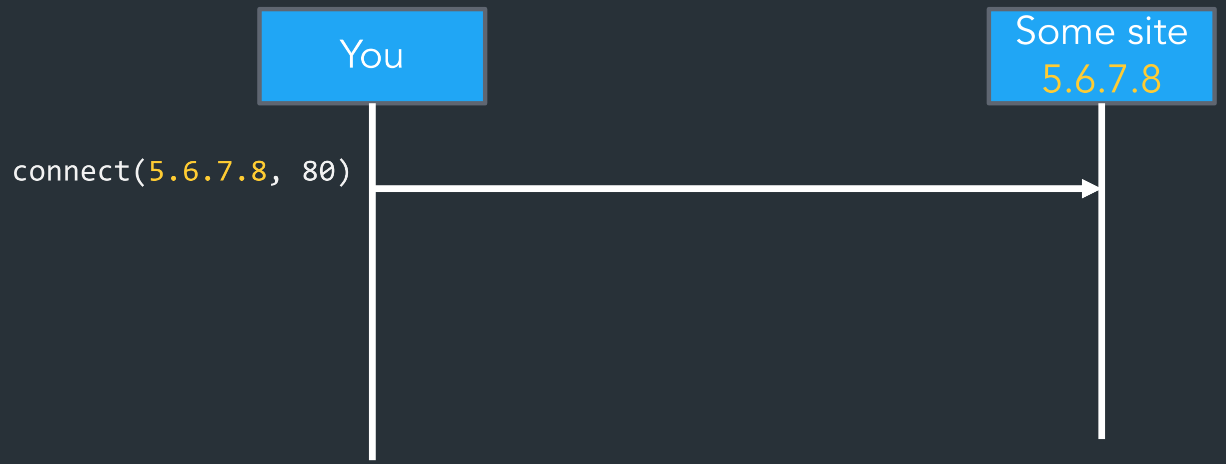
Why not? Why is this bad?



Why can't we just use IP addresses?

Typing them is annoying. What else?
Easy to make mistakes
Would like to have names for "services" => multiple IPs
    => IPs usually depend on where you are located on the network
Client applications don't know IPs of server

Why not? Why is this bad?



- Need to know IP addresses!
  - Users won't know
  - Hosts don't know—can't remember every single one!
- Some host ?= its IP address?   No!
  - A large website may be run by many servers
  - Devices may move between networks

What we have

IP addresses

- Used by routers to forward packets
- Fixed length, binary numbers
- Assigned based on where host is on the network
- Usually refers to one host

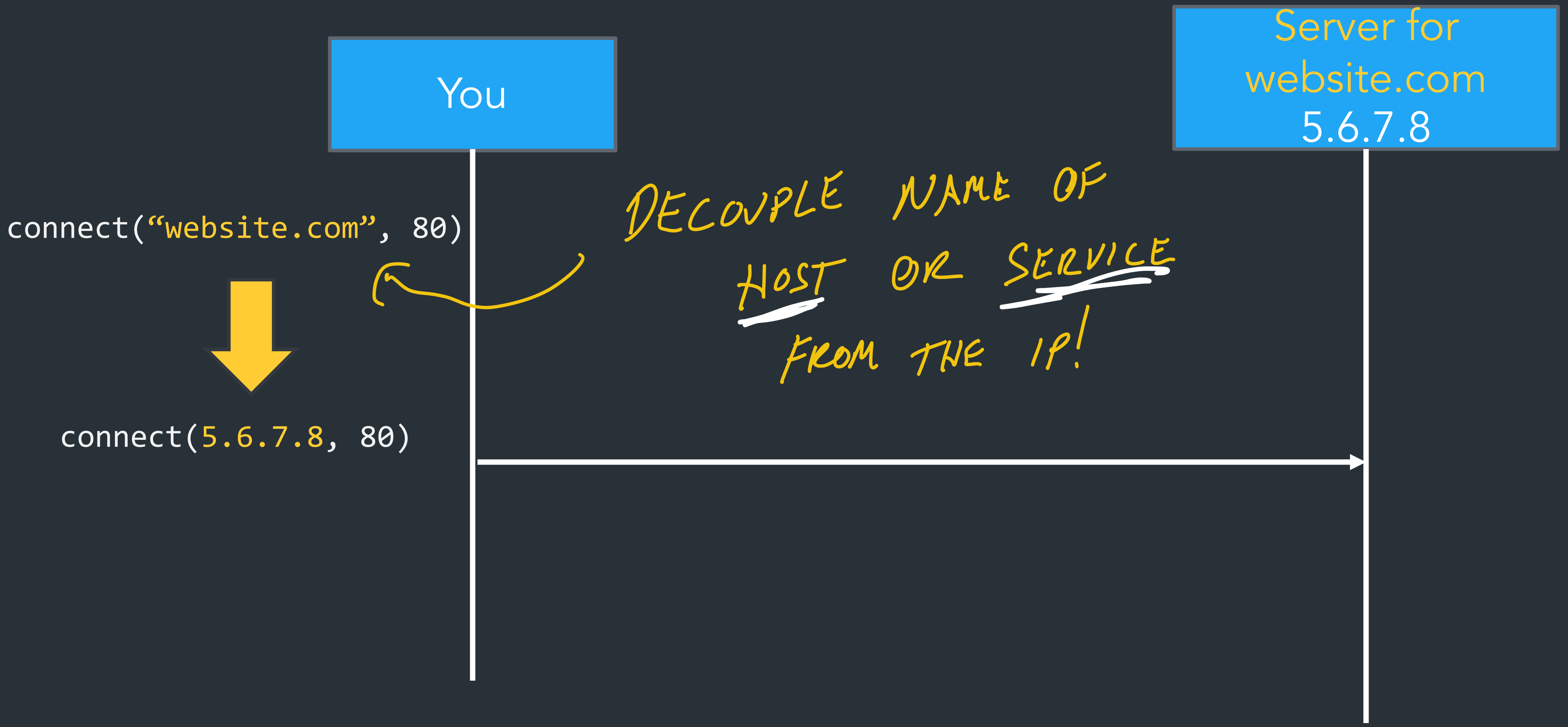
Examples

- 5.6.7.8
- 212.58.224.138
- 2620:6e:6000:900:c1d:c9f7:8a1c:2f48

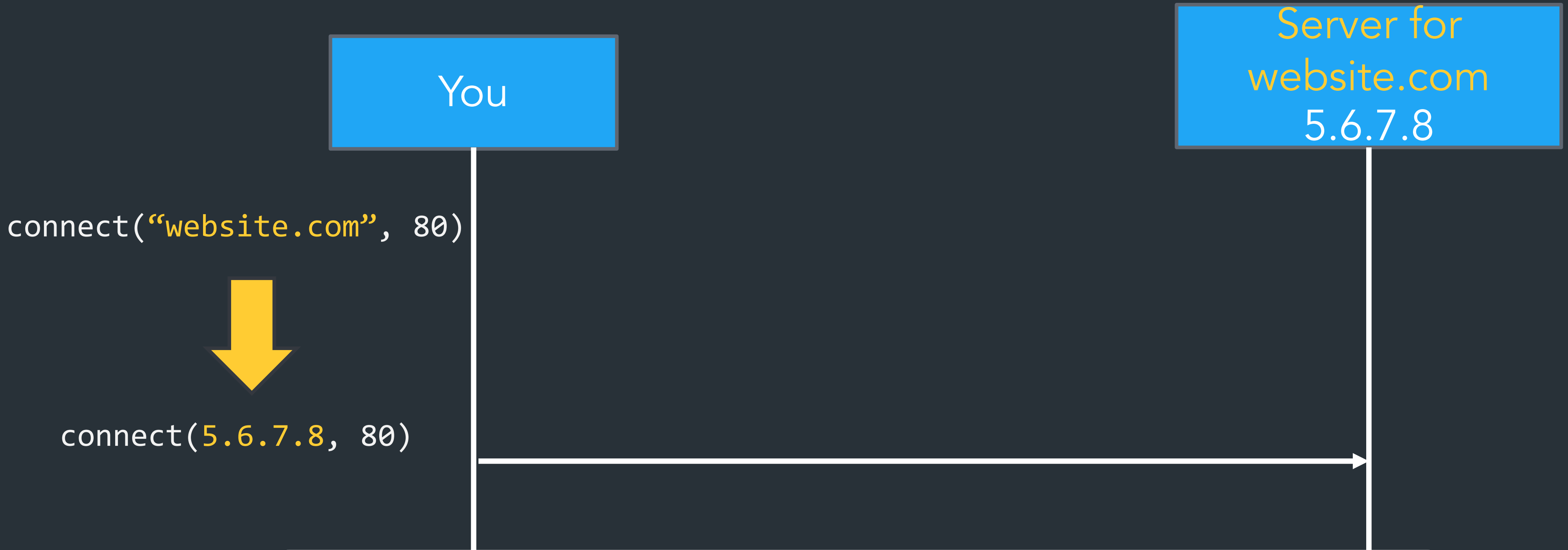
Efficient forwarding: ✅
Human readable: ❌
Scalable for distributed services: ❌

=> Need a new abstraction for "stuff" we are trying to access

What we want: a new abstraction for names



What we want: a new abstraction for names



Want: names
- Human-readable
- Variable length
- Don't need to care about where destination is/what server it is
  => Can refer to a service, not just a host

What does this mean?

"SERVICE"/"THING"

```
cs.brown.edu => 128.148.32.110
```

DNS

Why?

-

What does this mean?

```
cs.brown.edu => 128.148.32.110
```

Why?

- Names are easier to remember
- Addresses can change underneath
  - e.g, renumbering when changing providers
- Useful Multiplexing/sharing
  - One name -> multiple addresses
  - Multiple names -> one address

Remember ARP?

IP address => Link-layer address

L3 L2

Now: DNS

Names useful to users/applications => IP addresses

"WHO HAS GOOGLE.COM?" → 1.2.3.4

"QUESTION" "ANSWER"

Another change in layers => which enables so much more….

The original way: one file: `hosts.txt`

- Flat namespace
- Central administrator kept master copy (for the Internet)
- To add a host, emailed admin
- Downloaded file regularly

```
320 -- *************************
10-Jun-82 17:48:41-PDT,114828;000000000000
Mail-from: ARPANET host SRI-NIC rcvd at 10-Jun-82 1747-PDT
Date: 10 Jun 1982 1742-PDT
From: Dyer
Subject: Hostname table, 10-June-82
To:    dcacode252 at USC-ISI
cc:    nic

       ARPANET HOST NAMES AND LIAISON                10-Jun-82

HOST NAME      HOST ADDRESS       SPONSOR          LIAISON

ACC               10.2.0.54    VDH   ARPA    Lockwood, Gregory (LOCKWOOD@BBNC)
                                              Associated Computer Consultants
                                              414 East Cota Street
                                              Santa Barbara, California 93101
                                              (805) 965-1023

    CPUtype: PDP-11/70(UNIX)
ACCAT-TIP         10.2.0.35          ARPA    McBride, William T.
                                              (MCBRIDE@USC-ISIC)
                                              Naval Ocean Systems Center
                                              Code 8321
                                              271 Catalina Boulevard
                                              San Diego, California 92152
                                              (714) 225-2083 (AV) 933-2083

    CPUtype: H-316
AEROSPACE         10.2.0.65          AFSC    Nelson, Louis C. (LOU@AEROSPACE)
                                              Aerospace Corporation
                                              A2/1013
                                              P.O. Box 92957
                                              Los Angeles, California 90009
                                              (213) 615-4424

    CPUtype: VAX-11/780(UNIX)
AFGL              10.1.0.66          AFSC    Cosentino, Antonio
                                              (COSENTINO@AFSC-HQ)
                                              Air Force Geophysics Laboratory
                                              SUNA
                                              Mail Stop 30
                                              Hanscom Air Force Base,
```

The original way: one file: `hosts.txt`

- Flat namespace
- Central administrator kept master copy (for the Internet)
- To add a host, emailed admin
- Downloaded file regularly

Does it scale?

Lol no.

Domain Name System (DNS)

Originally proposed by RFC882, RFC883 (1983)

Distributed protocol to translate hostnames -> IP addresses

- Human-readable names
- Delegated control
- Load-balancing/content delivery
- So much more…

=> Distributed key-value store, before it was cool…

High level DNS goals

Scalability: need to be able to have a huge number of "records" (mappings from names => addrs)
- Lots of queries to look up names
- Lots of updates (#updates << #queries)

Distributed control: need to let people/organizations control their own names

Redundancy/fault tolerance:
Redundant way to do lookups, provide records

Some properties about the system that make this possible:
- Loose consistency: when changing records, not a huge deal if it takes a while to propagate (several minutes)

- Read-mostly database: writes generally infrequent, we can use lots and lots and lots of caching

The good news

Compared to other distributed systems, some properties that make these goals easier to achieve…

1. Read-mostly database

   Lookups MUCH more frequent than updates

2. Loose consistency

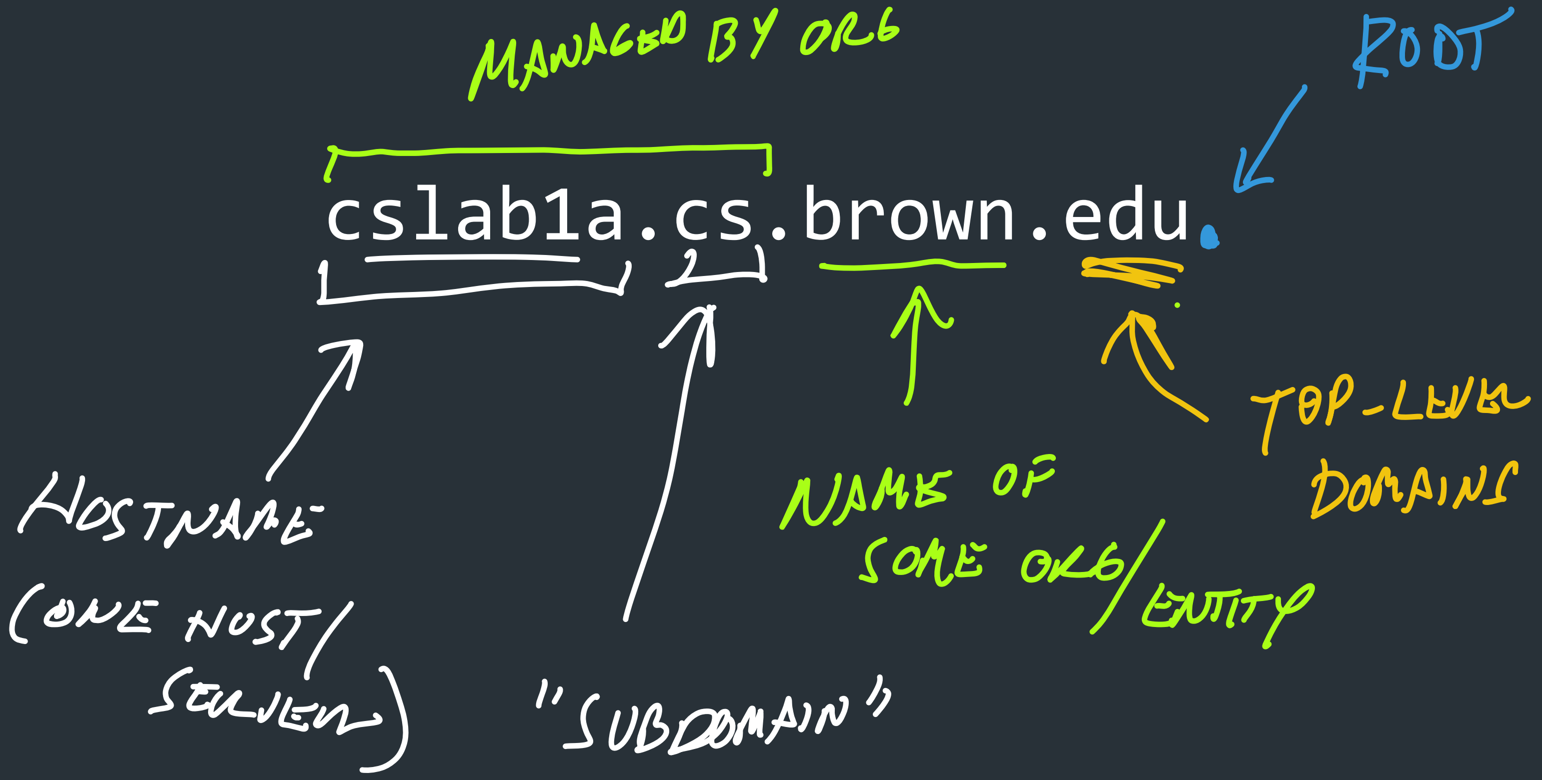
   When adding a machine, not end of the world if it takes minutes or hours to propagate

Can use lots and lots of caching

- Once you've lookup up a hostname, remember
- Don't have to look again in the near future

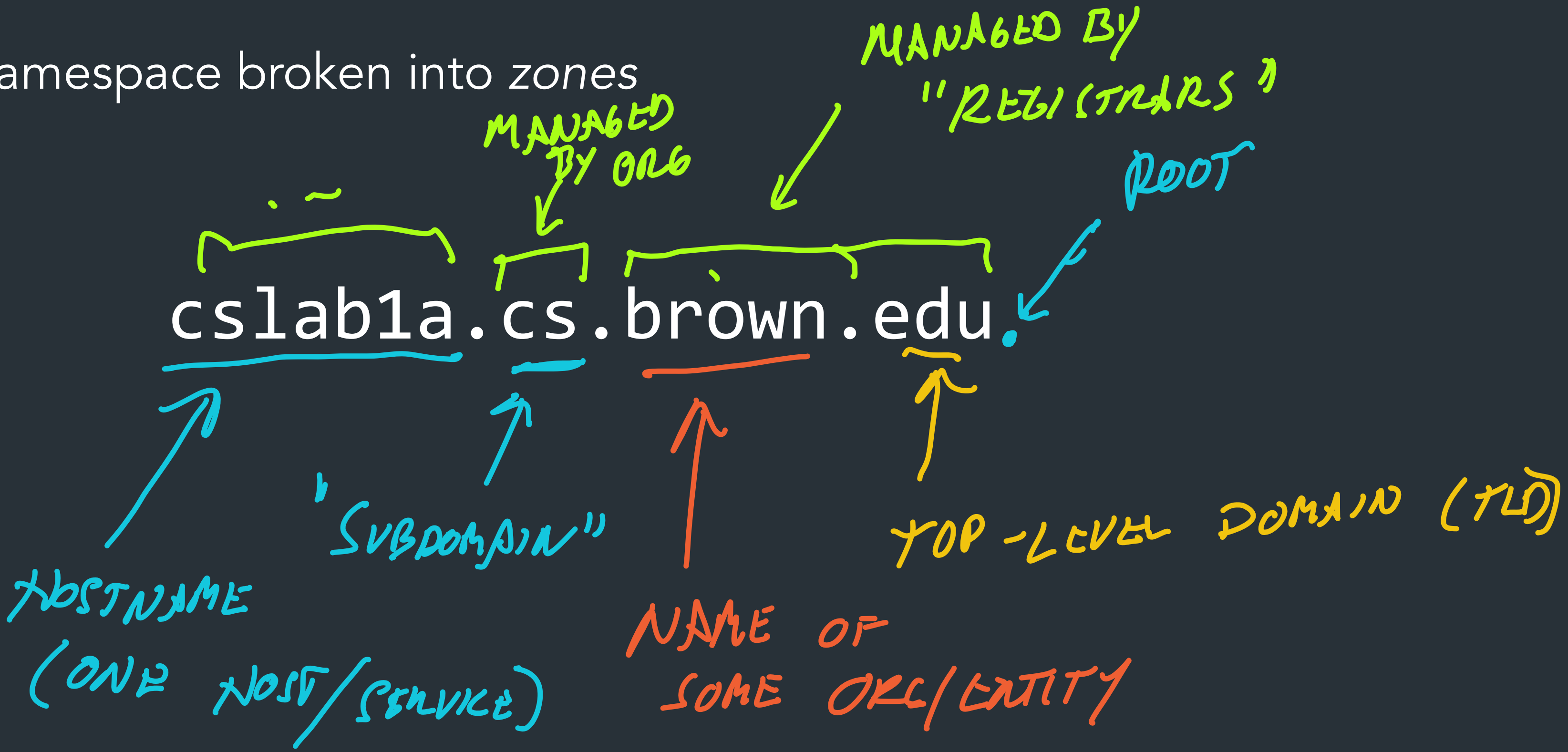
How it works

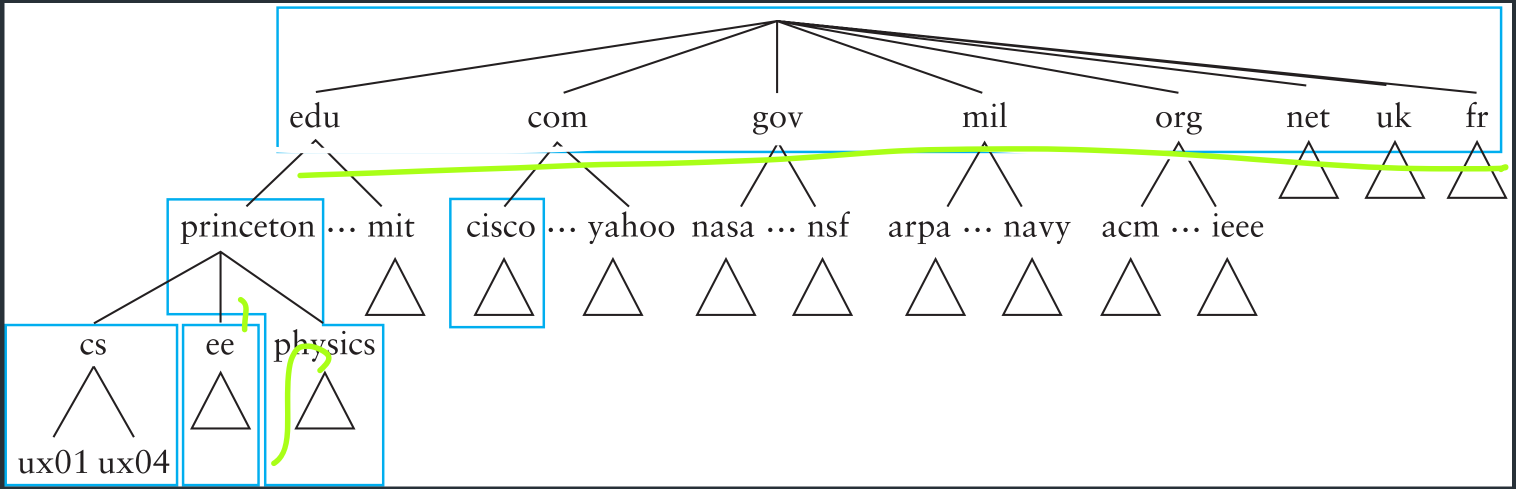
Hierarchical namespace broken into *zones*



How it works

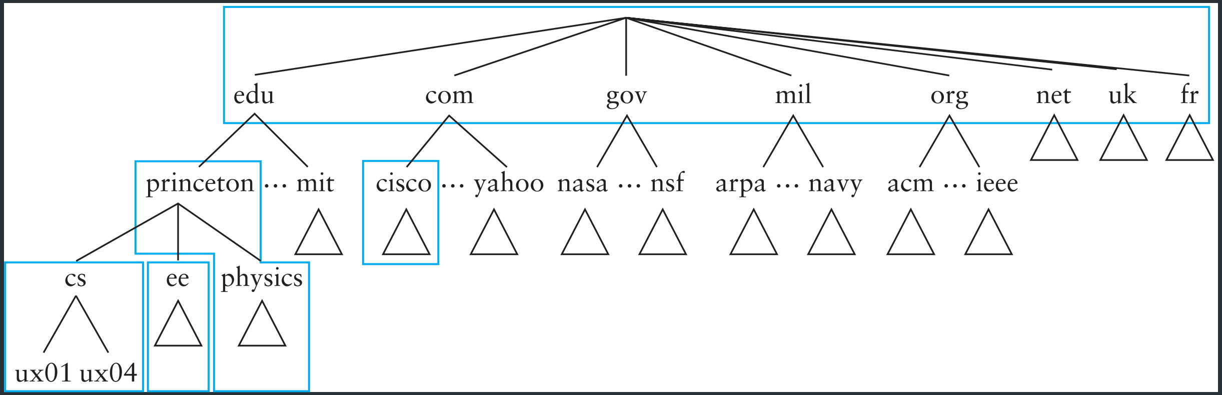
Hierarchical namespace broken into *zones*




edu
com
gov
mil
org
net
uk
fr
princeton
mit
cisco
yahoo
nasa
nsf
arpa
navy
acm
ieee
cs
ee
physics
ux01
ux04

How it works

- Hierarchical namespace broken into *zones*
  - root (.), edu., brown.edu., cs.brown.edu.,
  - Zones separately administered => delegation
  - Parent zone tells you how to find servers for subdomains
- Each zone served from multiple replicated servers
- Lots and lots of caching



DNS Example

(This is a typical, recursive-style query (more on what this means later))

QUESTION

NAMESERVER TO ASK

```
$ dig cs.brown.edu @10.1.1.10
; <<>> DiG 9.10.6 <<>> cs.brown.edu @10.1.1.10
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8536
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1


;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1220
;; QUESTION SECTION:
;cs.brown.edu. IN A


;; ANSWER SECTION:
cs.brown.edu.            1800      IN      A        128.148.32.12


;; Query time: 69 msec
;; SERVER: 10.1.1.10#53(10.1.1.10)
;; WHEN: Tue Apr 19 09:03:39 EDT 2022
;; MSG SIZE  rcvd: 57
```

TTL (SECONDS) – HOW LONG TO CACHE RECORD

RESULT TYPE

ANSWER (CAN HAVE MULTIPLE)

HOW LONG QUERY TOOK

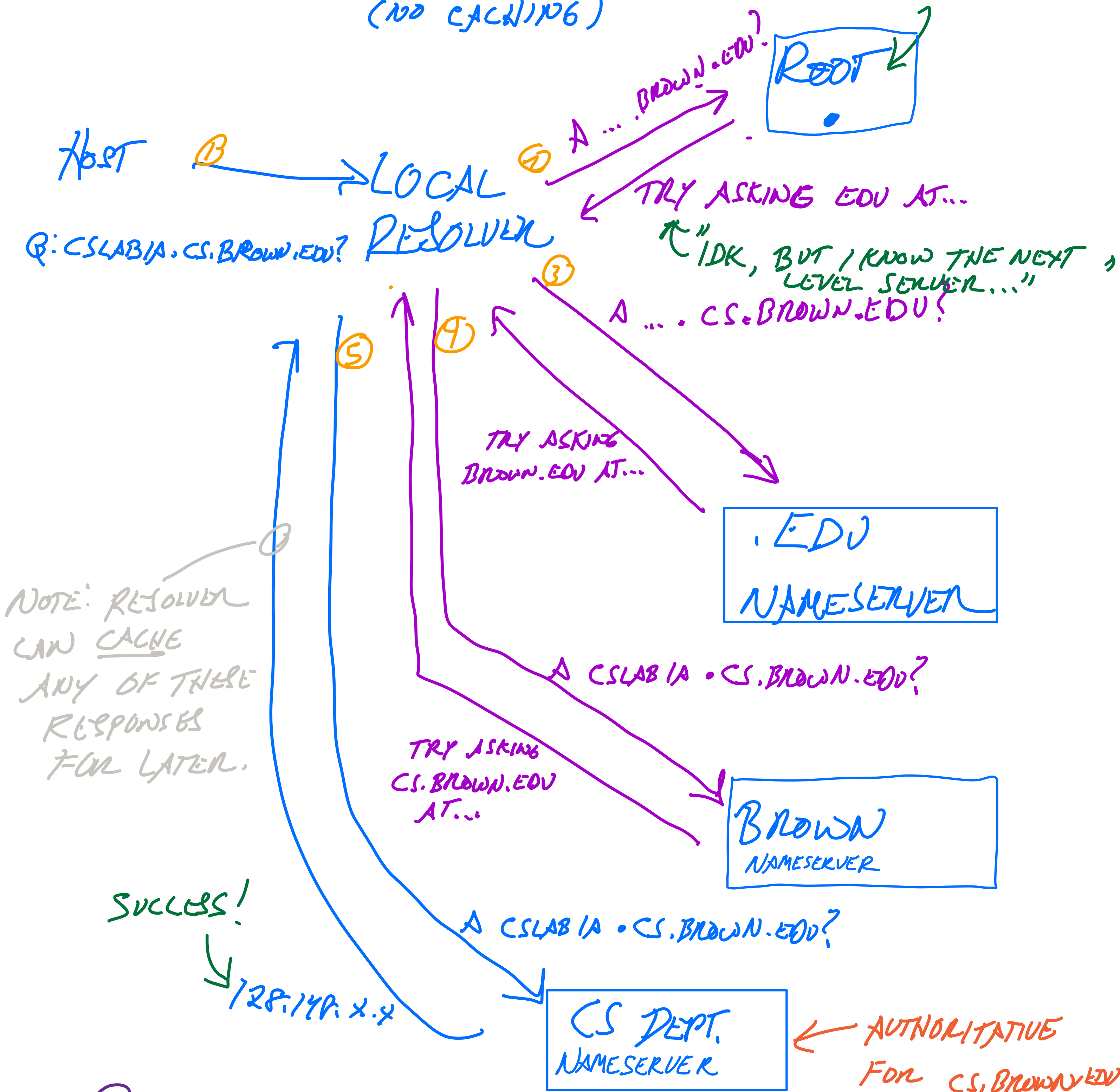
Types of DNS servers

- "authoritative servers" : servers that "own" records for some domain (e.g. cs.brown.edu)

- Resolvers: you (or another server" queries this to look up names, tries to get closer to authoritative server
  - => in most cases, you interact with a resolver, it contacts an authoritative server if it doesn't know the answer
  - => These are basically caches

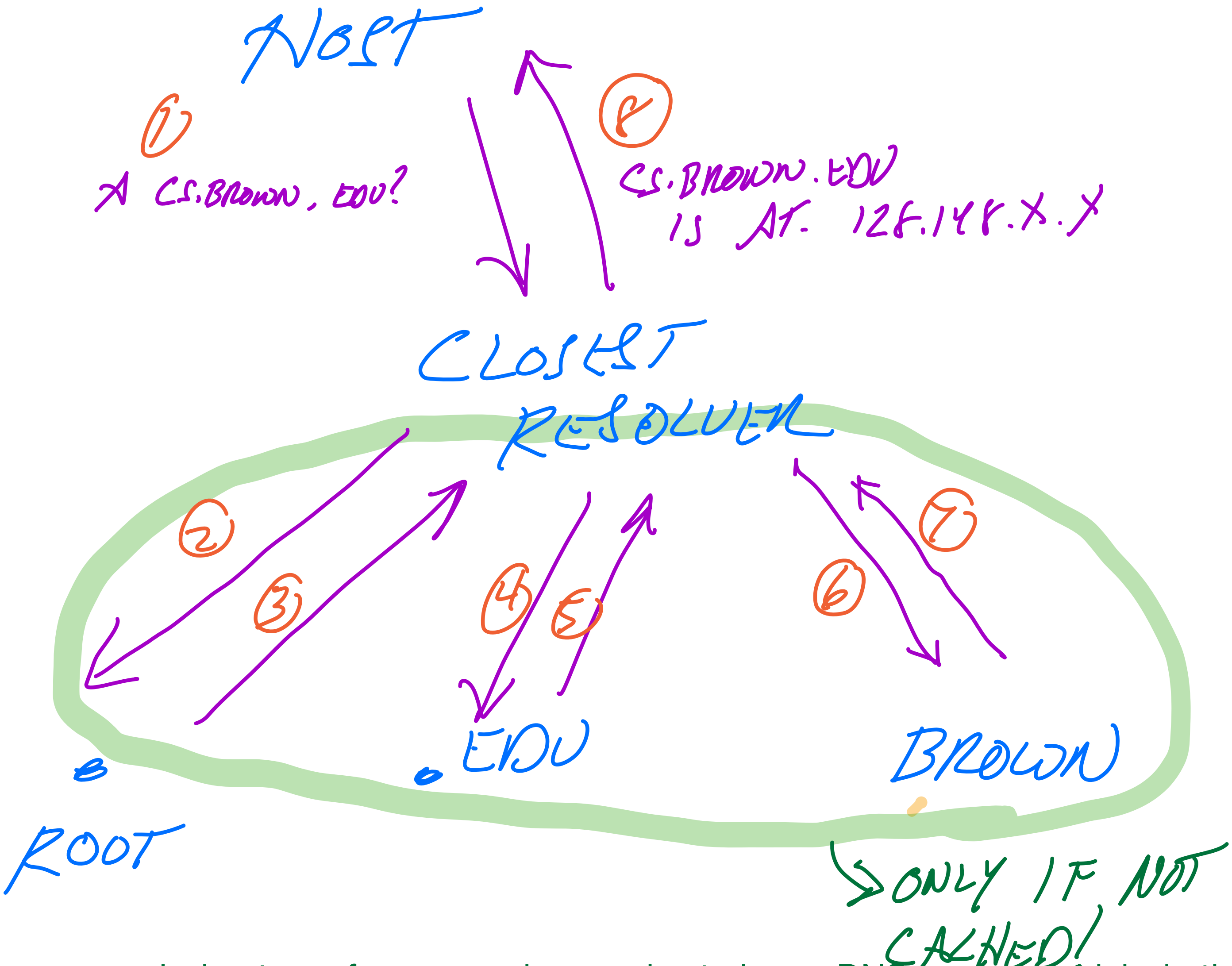
HOW A DNS QUERY WORKS: ITERATIVE VERSION

(NO CACHING)



1. HOST ASKS LOCAL RESOLVER
2. RESOLVER STARTS RECURSIVE QUERY FROM ROOT
   ⇒ 2 3 4 INTERMEDIATE NAMESERVERS DON'T HAVE ANSWER, BUT RESPOND W/ NEXT SERVER THAT KNOWS MORE
5. FOUND SERVER W/ AUTHORITATIVE ANSWER!

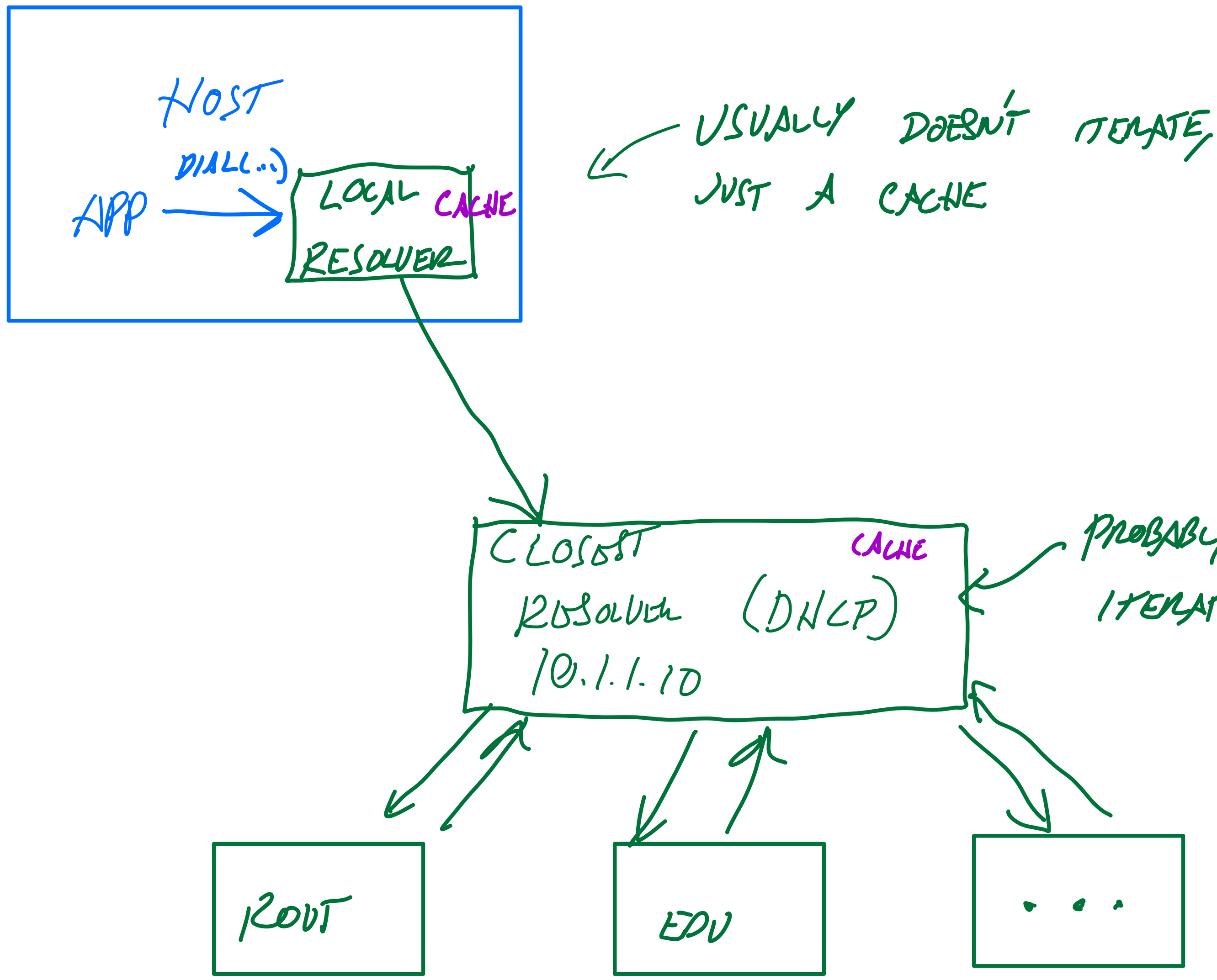
RECURSIVE DNS QUERIES (MORE COMMON)

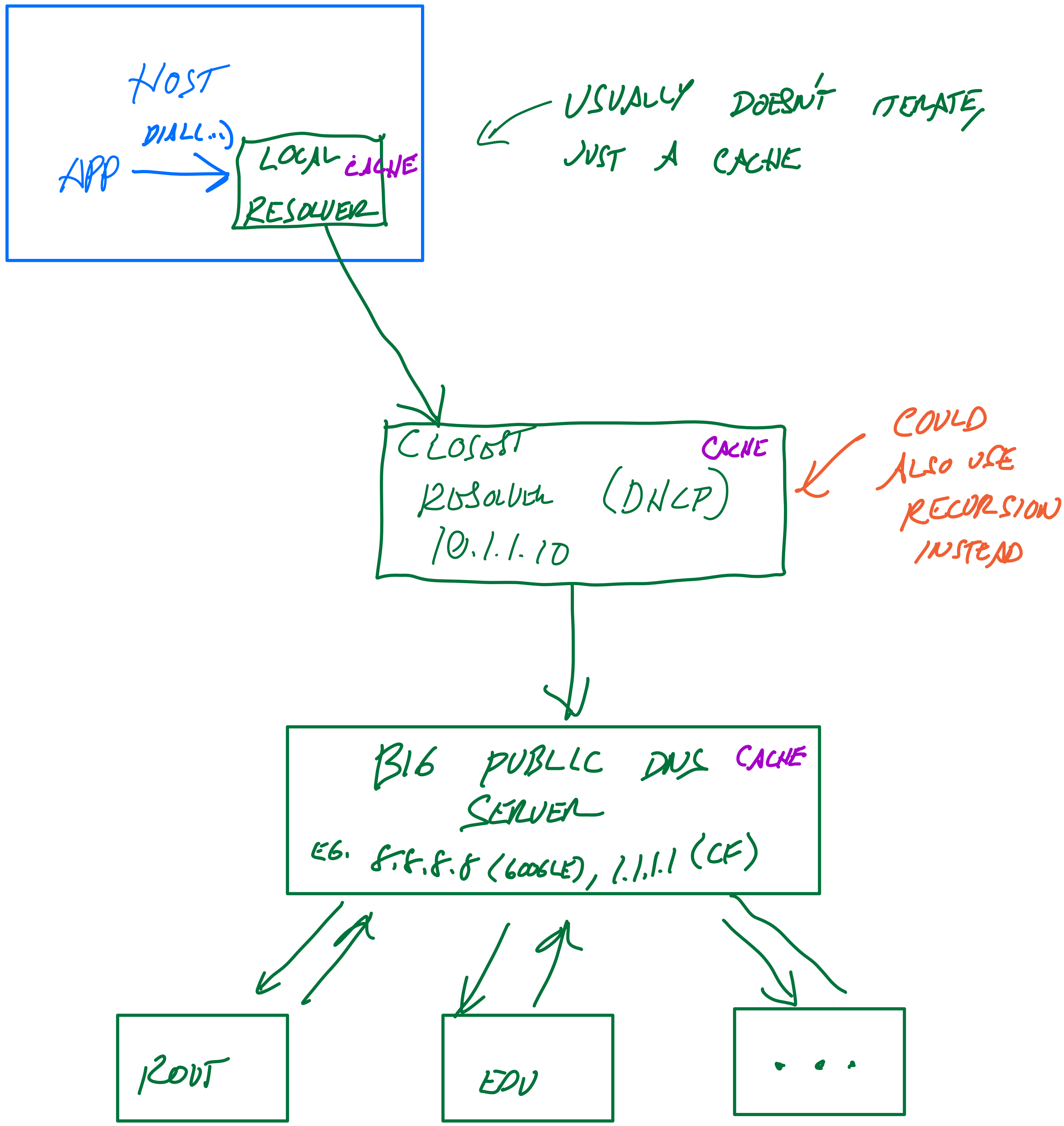


More commonly, hosts perform recursive queries to larger DNS servers, which do the typical iteration process (from the previous page) on the client's behalf.

Why? All resolvers cache responses—a larger resolver is more likely to have these entries in its cache. If the resolver has a valid answer for any of the steps, it can skip it! (For example, if the nameserver for .edu is cached but cs.brown.edu is not, the local resolver can skip skeps 2-3.

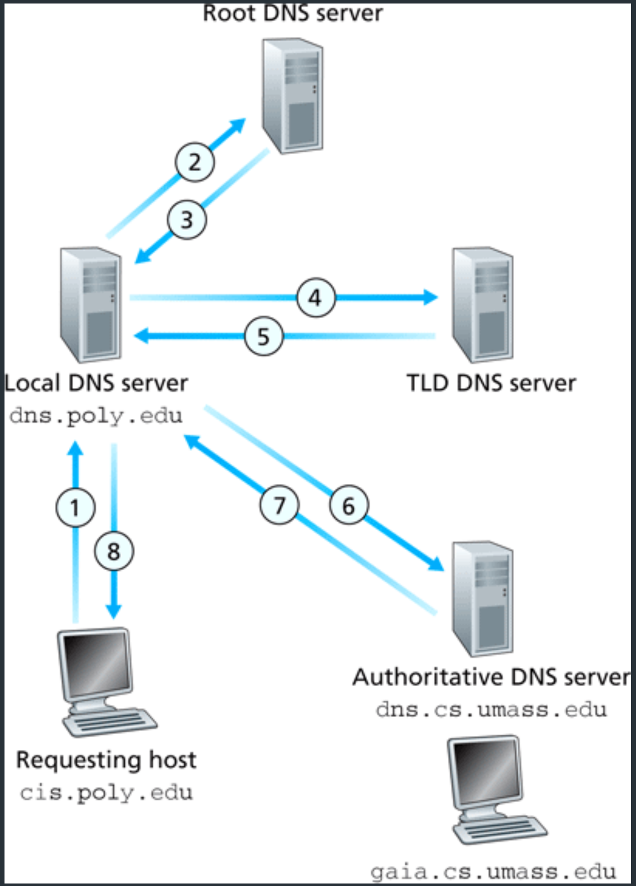
Who provides the closest resolver?
- Many OSes have a resolver on the local system, which acts as a local cache
- Usually, every local network has its own resolver (Brown, your home router, etc)
- These local resolvers MIGHT do iterative queries, but often do another recursive step to a big public DNS server (like Google's 8.8.8.8, or Cloudflare's 1.1.1.1)
  => Multiple levels of caching!


HOST
APP
DIAL(...)
LOCAL RESOLVER
CACHE
USUALLY DOESN'T ITERATE, JUST A CACHE
CLOSEST RESOLVER (DHCP)
10.1.1.10
CACHE
PROBABLY ITERATES
ROOT
EDU
. . .


HOST
APP
DIAL(...)
LOCAL RESOLVER
CACHE
USUALLY DOESN'T ITERATE, JUST A CACHE
CLOSEST RESOLVER (DHCP)
10.1.1.10
CACHE
COULD ALSO USE RECURSION INSTEAD
BIG PUBLIC DNS SERVER
CACHE
EG. 8.8.8.8 (GOOGLE), 1.1.1.1 (CF)
ROOT
EDU
. . .

Resolver operation

- Apps make recursive queries to local DNS server (1)
  - Ask server to get answer for you
- Server makes iterative queries to remote servers (2,4,6)
  - Ask servers who to ask next
  - Cache results aggressively



Iterative query: step 1

```
$ dig cs.brown.edu @e.root-servers.net

; <<>> DiG 9.10.6 <<>> cs.brown.edu @e.root-servers.net
[ . . .]
;; QUESTION SECTION:
;cs.brown.edu. IN A

;; AUTHORITY SECTION:
edu. 172800 IN NS b.edu-servers.net.
edu. 172800 IN NS i.edu-servers.net.
edu. 172800 IN NS g.edu-servers.net.
[ . . .]

;; ADDITIONAL SECTION:
[ . . .]
i.edu-servers.net. 172800 IN A 192.43.172.30
g.edu-servers.net. 172800 IN A 192.42.93.30
b.edu-servers.net. 172800 IN A 192.33.14.30

;; Query time: 123 msec
;; SERVER: 2001:500:a8::e#53(2001:500:a8::e)
;; WHEN: Thu Oct 31 08:29:45 EDT 2024
;; MSG SIZE  rcvd: 839
```

← ASK ROOT NAMESERVER

← QUERY

No answer, but try these authoritative servers (for .edu)

Additional records: "BTW, here are the IPs for those other nameservers to try"

=> These are called "glue records" (needed because resolving b.edu-servers.net would otherwise require another DNS query, and possibly have a circular dependency)

Iterative query: step 2

```
$dig cs.brown.edu @192.33.14.30.   [192.33.14.30 was IP returned for b.edu-servers.net]

; <<>> DiG 9.10.6 <<>> cs.brown.edu @192.33.14.30
[ . . . ]
;; QUESTION SECTION:
;cs.brown.edu. IN A


;; AUTHORITY SECTION:
brown.edu. 172800 IN NS ns1.ucsb.edu.
brown.edu. 172800 IN NS bru-ns1.brown.edu.
brown.edu. 172800 IN NS bru-ns2.brown.edu.
brown.edu. 172800 IN NS bru-ns3.brown.edu.


;; ADDITIONAL SECTION:
ns1.ucsb.edu. 172800 IN A 128.111.1.1
ns1.ucsb.edu. 172800 IN AAAA 2607:f378::1
bru-ns1.brown.edu. 172800 IN A 128.148.248.11
bru-ns2.brown.edu. 172800 IN A 128.148.248.12
bru-ns3.brown.edu. 172800 IN A 128.148.2.13
```

TRY BROWN.EDU NAMESERVERS

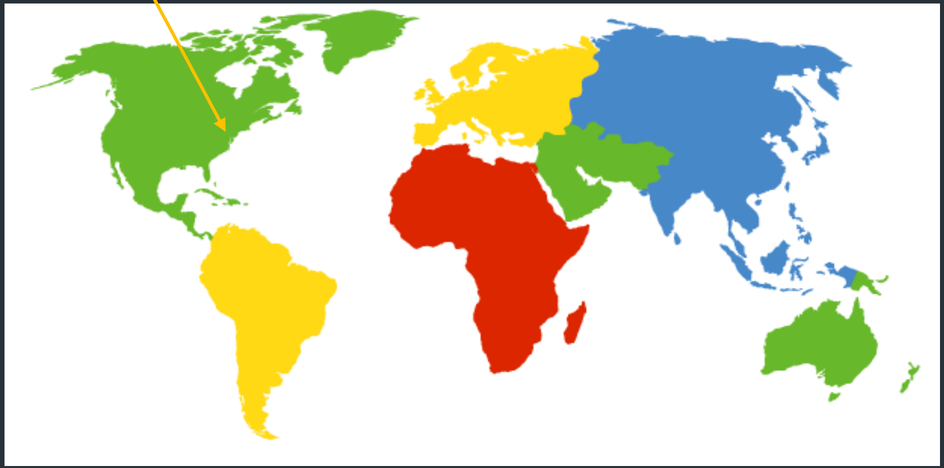
```
$ dig cs.brown.edu @128.111.1.1  [128.111.1.1 was IP returned for ns1.ucsb.edu]
; <<>> DiG 9.10.6 <<>> cs.brown.edu @128.111.1.1
[ . . . ]

;; QUESTION SECTION:
;cs.brown.edu. IN A

;; ANSWER SECTION:
cs.brown.edu. 1800 IN A 128.148.32.12


;; Query time: 77 msec
;; SERVER: 128.111.1.1#53(128.111.1.1)
;; WHEN: Thu Oct 31 08:35:11 EDT 2024
;; MSG SIZE  rcvd: 57
```

Answer!

Where is the root server?

- Located in New York
- How do we make the root scale?

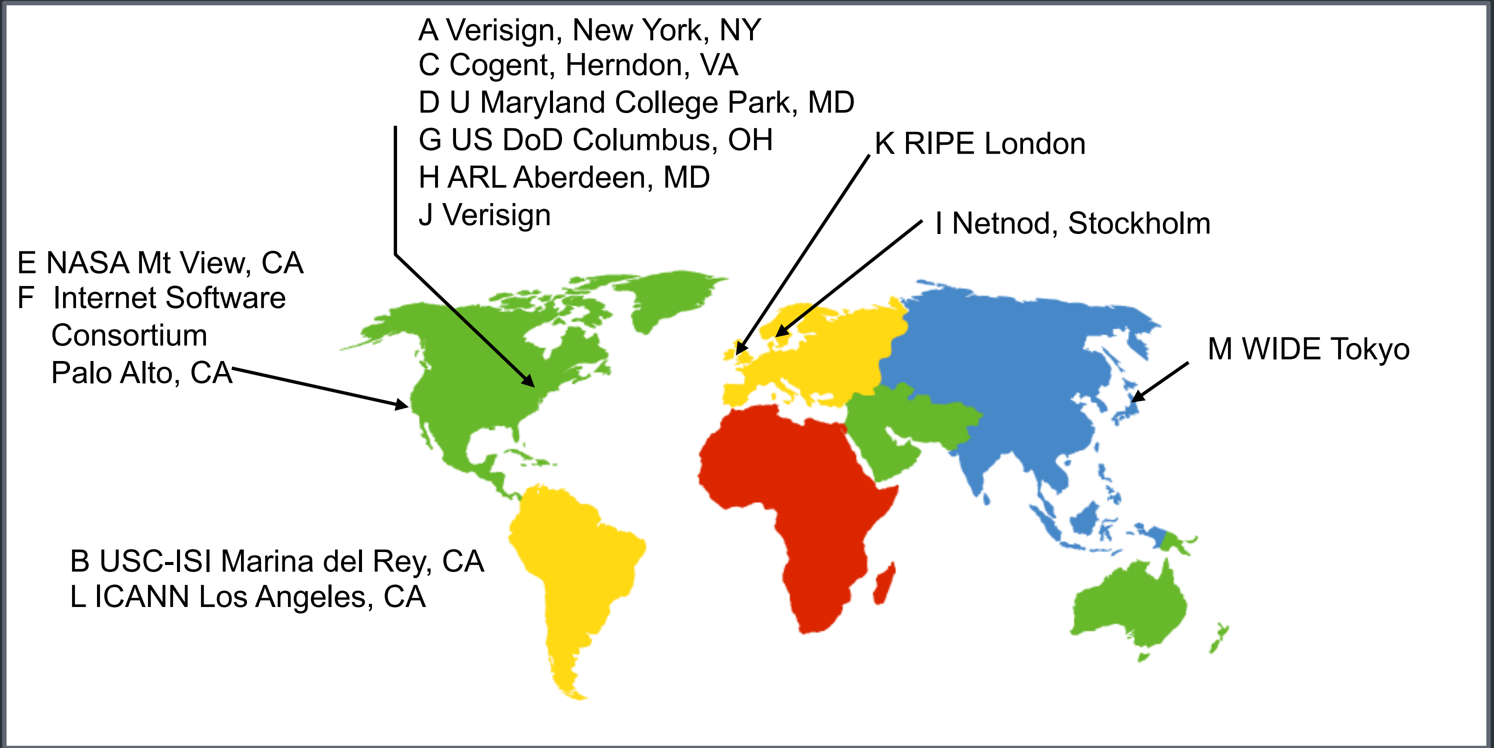
Verisign, New York, NY



DNS Root Servers

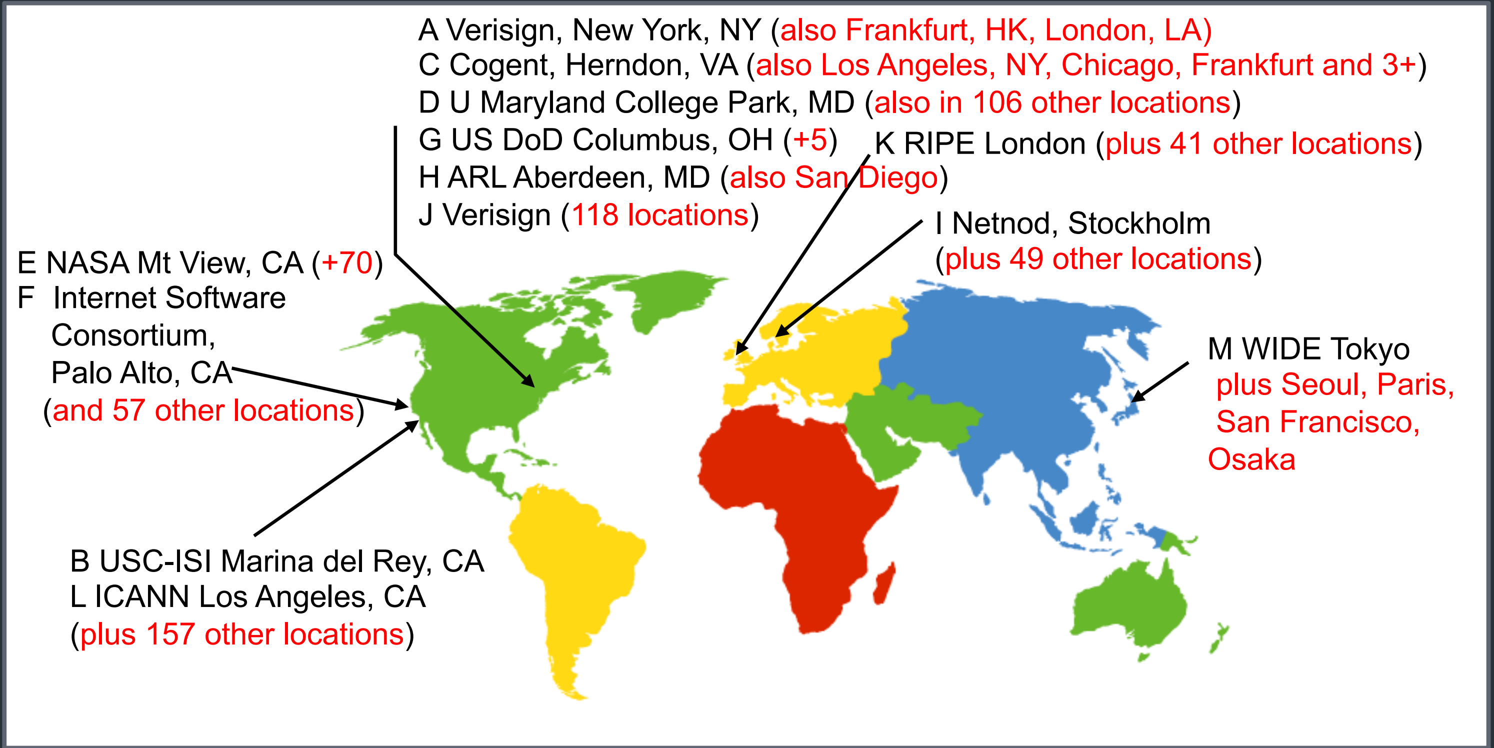
ANYCAST! USING BGP, ADVERTISE IP FROM MULTIPLE LOCATIONS

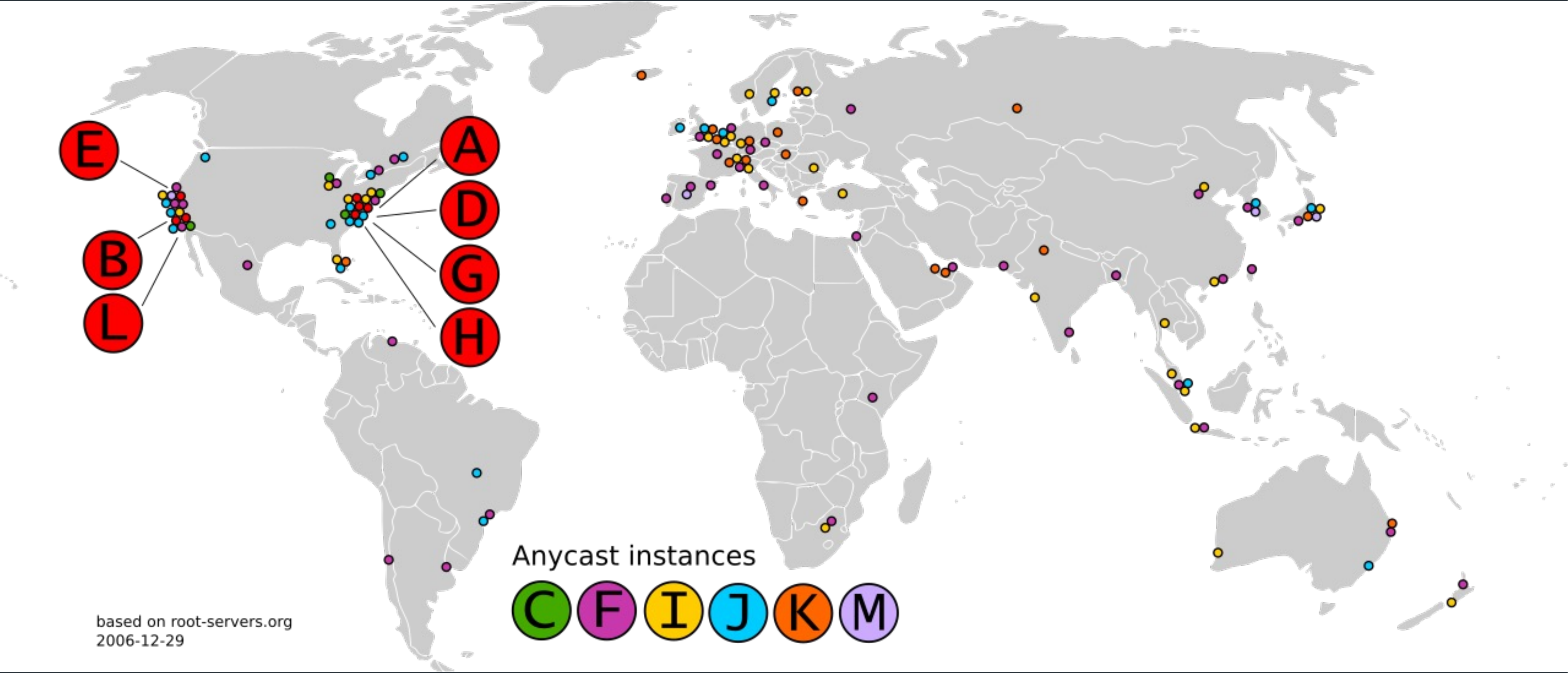
- 13 Root Servers (www.root-servers.org)
  - Labeled A through M (e.g, A.ROOT-SERVERS.NET)
- Does this scale?



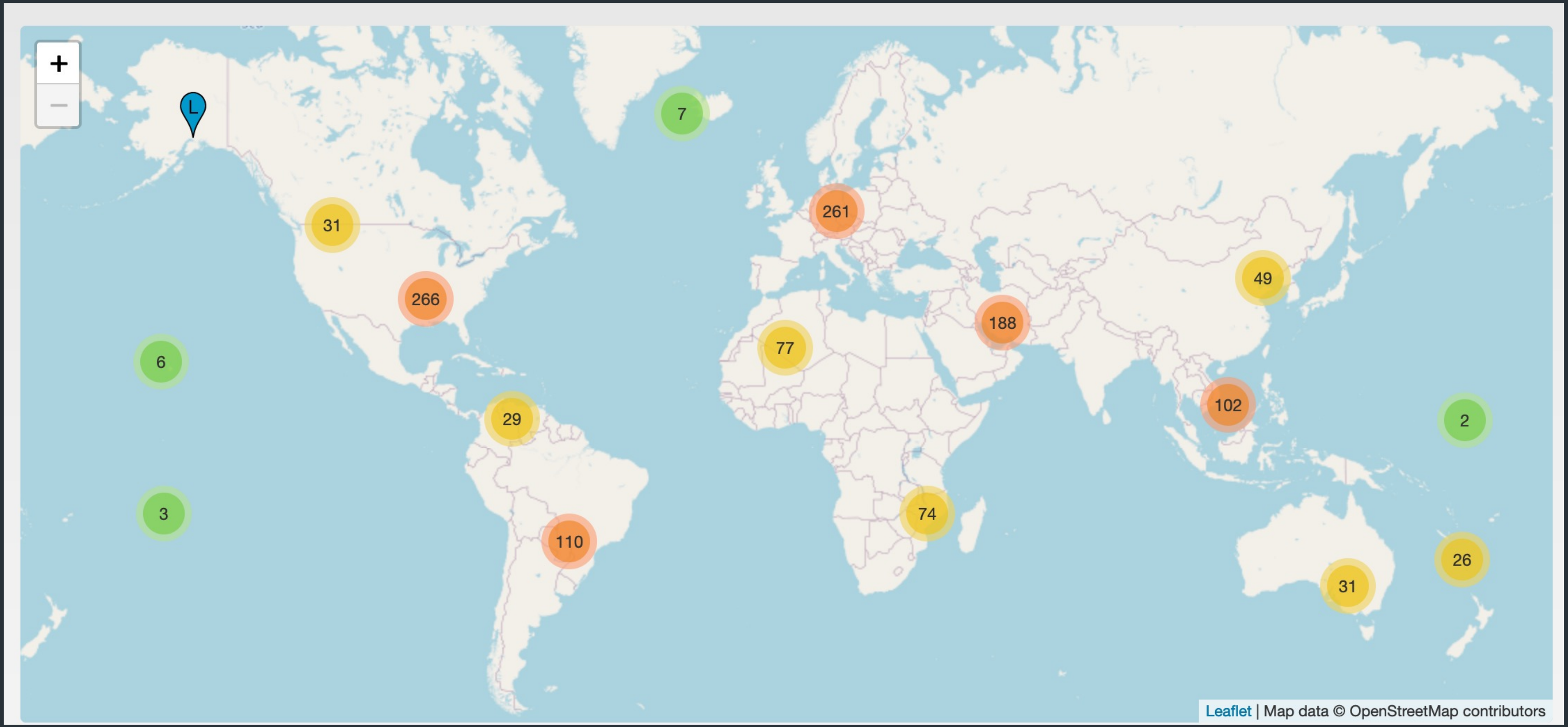
DNS Root Servers

- 13 Root Servers (www.root-servers.org)
  - Labeled A through M (e.g, A.ROOT-SERVERS.NET)
- Remember anycast?




E
B
L
A
D
G
H
Anycast instances
C
F
I
J
K
M
based on root-servers.org
2006-12-29

DNS Root Servers: Today



From: www.root-servers.org

How it scales: caching

Resolvers cache responses to avoid doing recursive/iterative queries

- Many messages => extra computation, extra latency

HOW LONG RESULT SHOULD BE CACHED => DELETE WHEN EXPIRES

```
$ dig cs.brown.edu @10.1.1.10
;; ANSWER SECTION:
cs.brown.edu.          1800     IN     A        128.148.32.12
```

Related: redundant services via DNS

Can return multiple answers for one record
=> If a client can't connect to first result, can try next one

```
$ dig nytimes.com

;; ANSWER SECTION:
nytimes.com. 111 IN A 151.101.65.164
nytimes.com. 111 IN A 151.101.1.164
nytimes.com. 111 IN A 151.101.129.164
nytimes.com. 111 IN A 151.101.193.164


;; Query time: 40 msec
;; SERVER: 10.1.1.10#53(10.1.1.10)
;; WHEN: Thu Nov 09 08:42:41 EST 2023
;; MSG SIZE  rcvd: 104
```

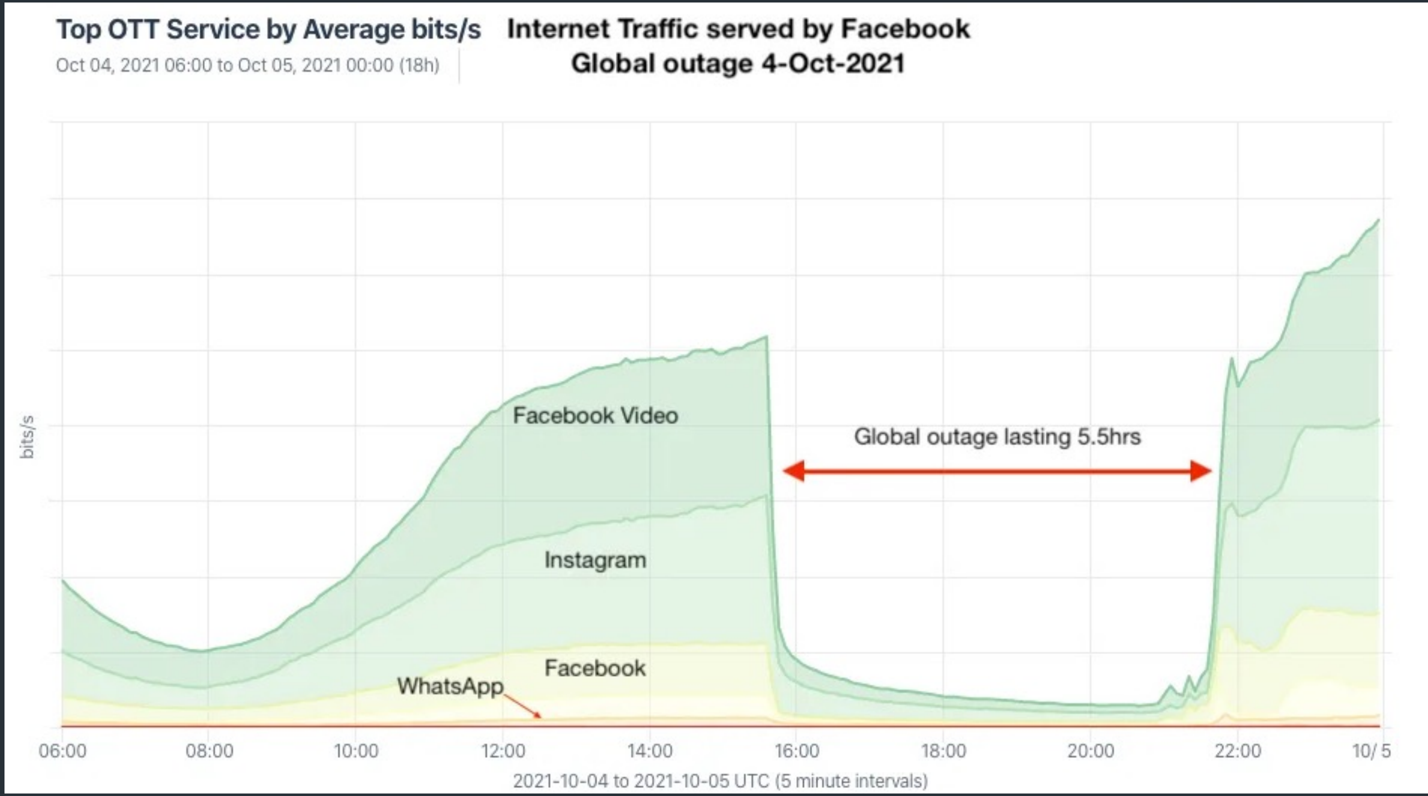
LOAD BALANCING
+
REDUNDANCY

DNS server usually shuffles answers on each response—why?

Facebook DNS outage (2021)

BGP configuration bug: Facebook withdraws all routes for its DNS servers to the Internet

=> Facebook DNS unreachable—not even Facebook could access their systems!



Traffic graph

Many writeups here

```
user@host$ dig @1.1.1.1 facebook.com # CloudFlare
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 5153
;facebook.com.                    IN      A
user@host$ dig @8.8.8.8 facebook.com # Google Public DNS
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 43224
;facebook.com.                    IN      A
user@host$ dig @208.67.222.222 facebook.com # OpenDNS
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 7643
;facebook.com.                    IN      A
user@host$ dig @176.103.130.130 facebook.com # AdGuard
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 5434
;facebook.com.                    IN      A
```

ERROR CODE:
AUTHORITATIVE SERVER
NOT FOUND! !!

Source

Reverse DNS

128.148.32.12

What if we want to map IP address => domain name?

Leverages hierarchy in IP addresses, but in reverse

=> How? reverse the numbers: 12.32.148.128, then look that up

Reverse DNS

How do we get the other direction, IP address to name?

- Addresses have a natural hierarchy:
  - 128.148.32.12
- Idea: reverse the numbers: 12.32.148.128 …
  - and look that up in DNS
- Under what TLD?
  - Convention: in-addr.arpa
  - Lookup 12.32.148.128.in-addr.arpa
  - in6.arpa for IPv6

DNS record types

RR Type	Purpose	Example
A	IPv4 Address	128.148.56.2
AAAA	IPv6 Address	2001:470:8956:20::1
CNAME	Specifies an alias ("Canonical name")	systems.cs.brown.edu. 86400 IN CNAME systems-v3.cs.brown.edu. systems-v3.cs.brown.edu. 86400 IN A 128.148.36.51
NS	DNS servers for a domain	cs.brown.edu. 86400 IN NS br1.brown.edu
MX	Mail servers	MX <priority> <ip> eg. MX 10 1.2.3.4
SOA	Start of authority	Information about who owns a zone
PTR	Reverse IP lookup	7.34.148.128.in-addr.arpa. 86400 IN PTR quanto.cs.brown.edu.
SRV	How to reach specific services (eg. host, port)	_minecraft._tcp.example.net 3600 SRV <priority> <weight> <port> <server IP>

More: https://en.wikipedia.org/wiki/List_of_DNS_record_types

Next time:
- What can go wrong?
- How can DNS help applications scale?