CSCI-1680
DNS

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
- TCP Gearup II: TONIGHT, 10/31 6-8pm in CIT 368
  - Prep for milestone II
- HW3 (short!): Due next Thurs

Administrivia

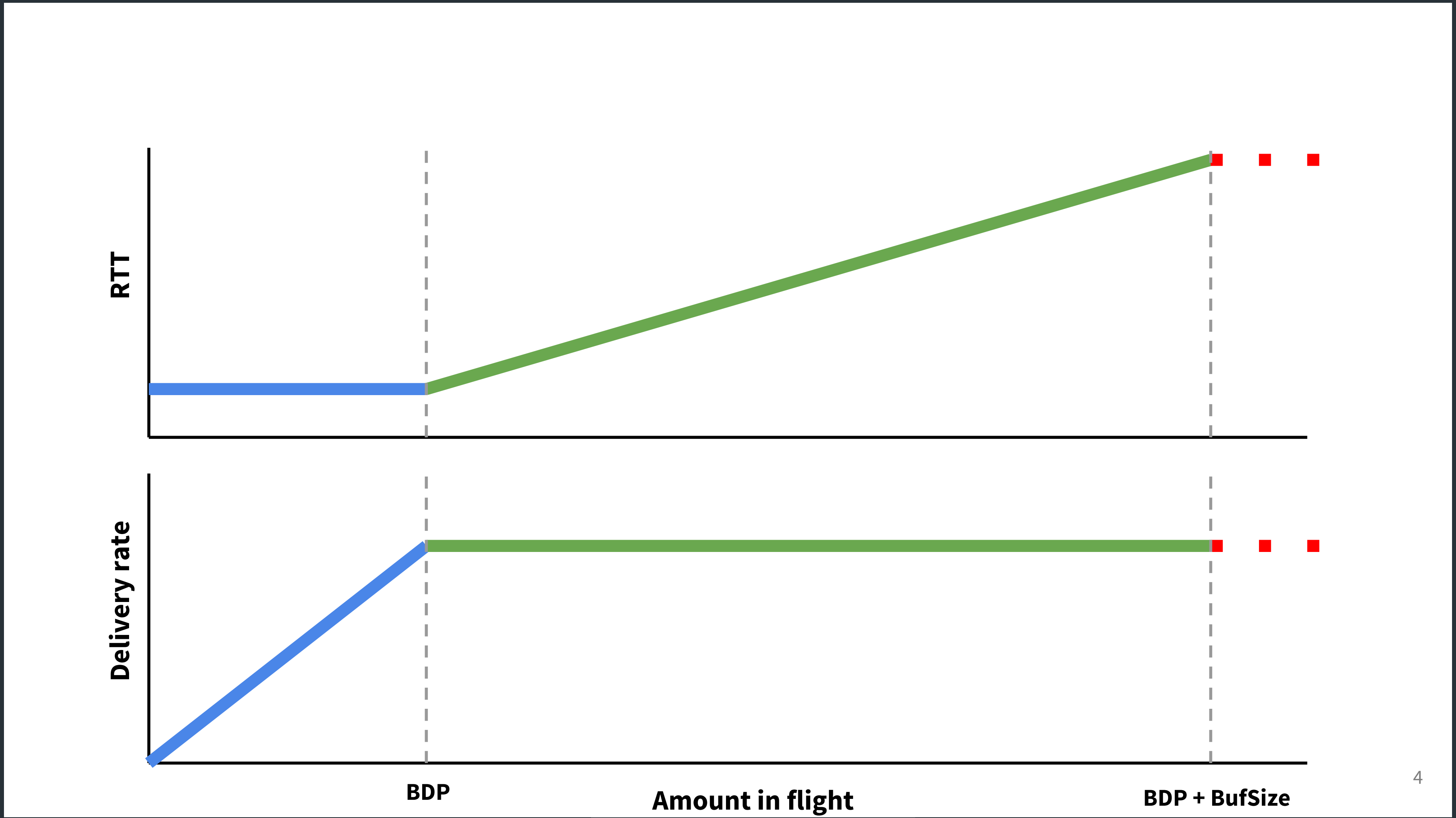
- TCP milestone I: this week, sign up for a meeting if you haven't
  - If you're stuck: bring what you have, it does not need to be perfect
  - DO NOT just hack stuff together to make it look good in Wireshark
- TCP Gearup II: TONIGHT, 10/31 6-8pm in CIT 368
  - Prep for milestone II
- HW3 (short!):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

Warmup

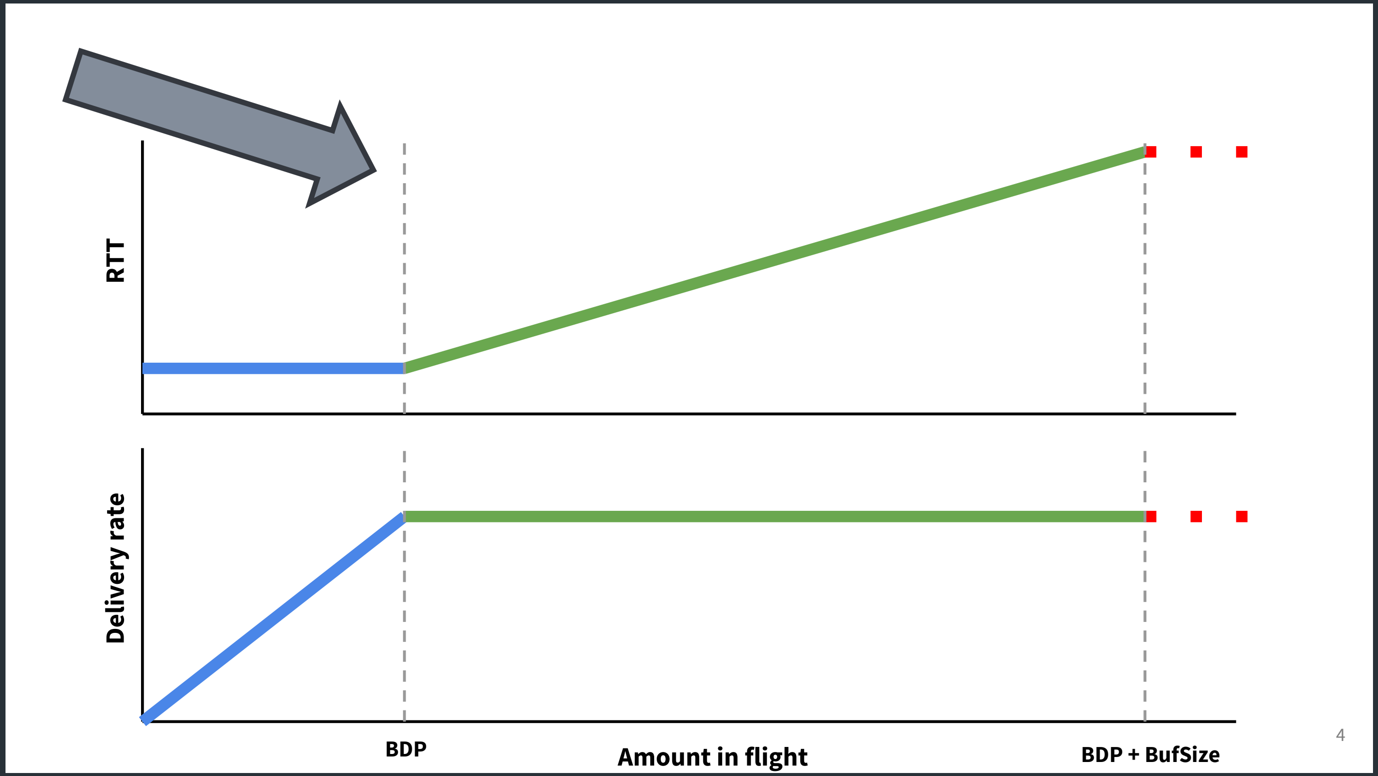
Which of the following contribute to congestion?

a. Packets queueing up at switches
b. High CPU usage on the receiver
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



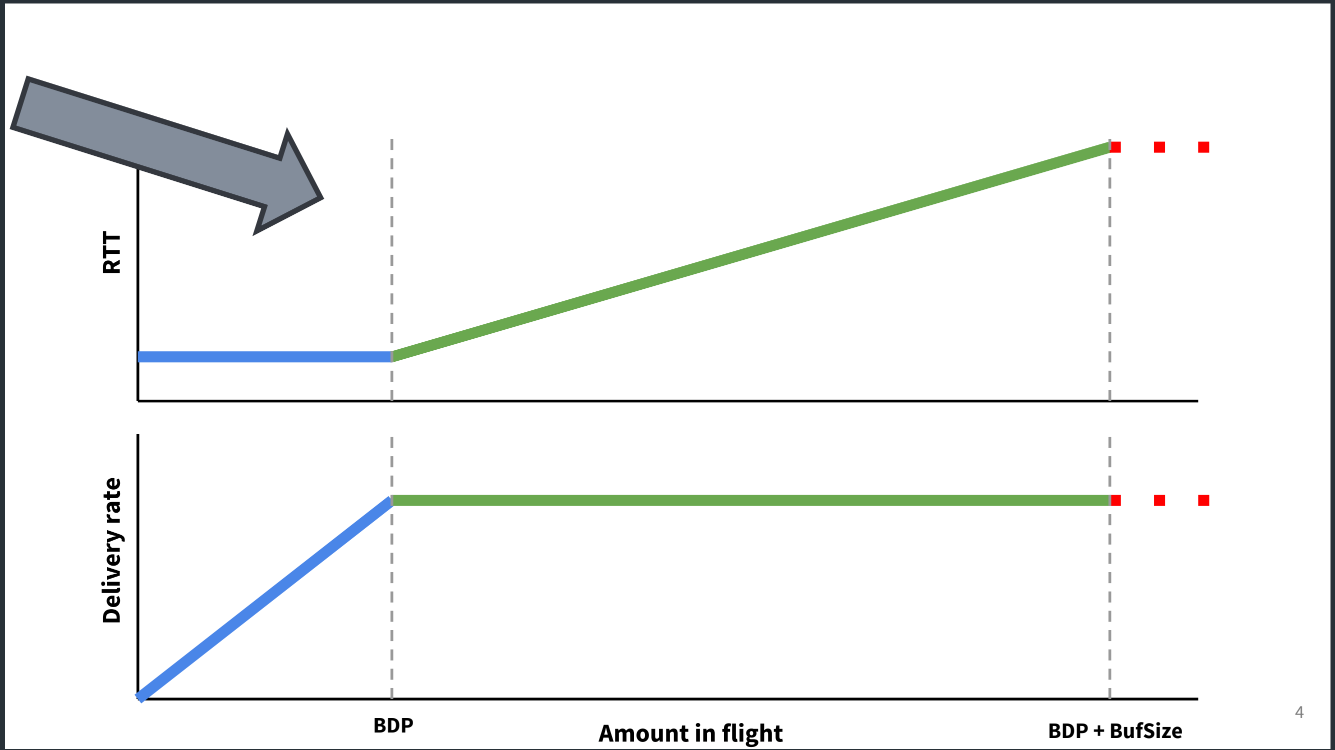
"BBR congestion control"



"BBR congestion control"

Bandwidth-delay product (BDP): maximum amount of data that can be in-transit on a network link at any given time

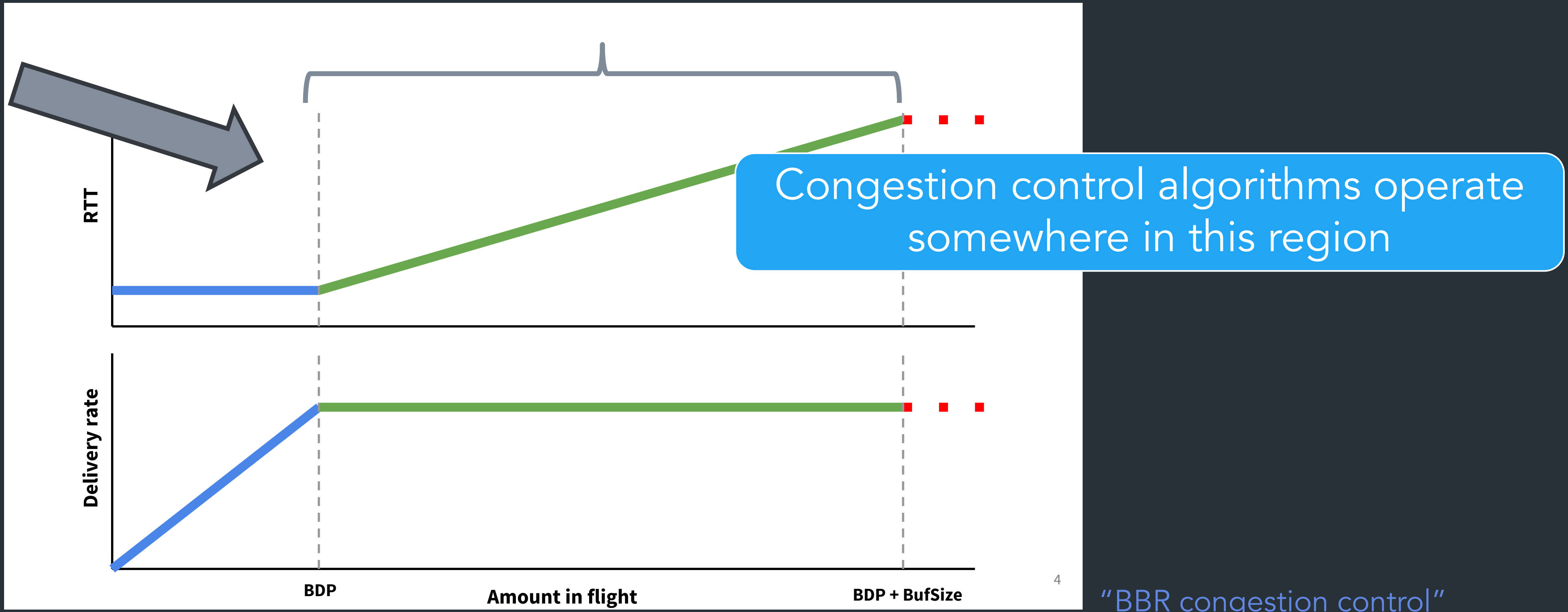
(Link capacity (bits/sec)) * (RTT (sec))
= (bytes)



"BBR congestion control"

Bandwidth-delay product (BDP): maximum amount of data that can be in-transit on a network link at any given time

(Link capacity (bits/sec)) * (RTT (sec))
= (bytes)
Eg. 1Gbps link * 1ms RTT = 125KiB BDP



"BBR congestion control"

Bandwidth-delay product (BDP): maximum amount of data that can be in-transit on a network link at any given time

(Link capacity (bits/sec)) * (RTT (sec))
= (bytes)

Eg. 1Gbps link * 1ms RTT = 125KiB BDP

=> After exceeding BDP, network is queueing packets. After queues are full, packets getting dropped due to congestion.

The basic principle

Congestion Control (CC) algorithm

The basic principle

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

↓

Congestion Control (CC) algorithm

↓

Congestion window: cwnd

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)

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

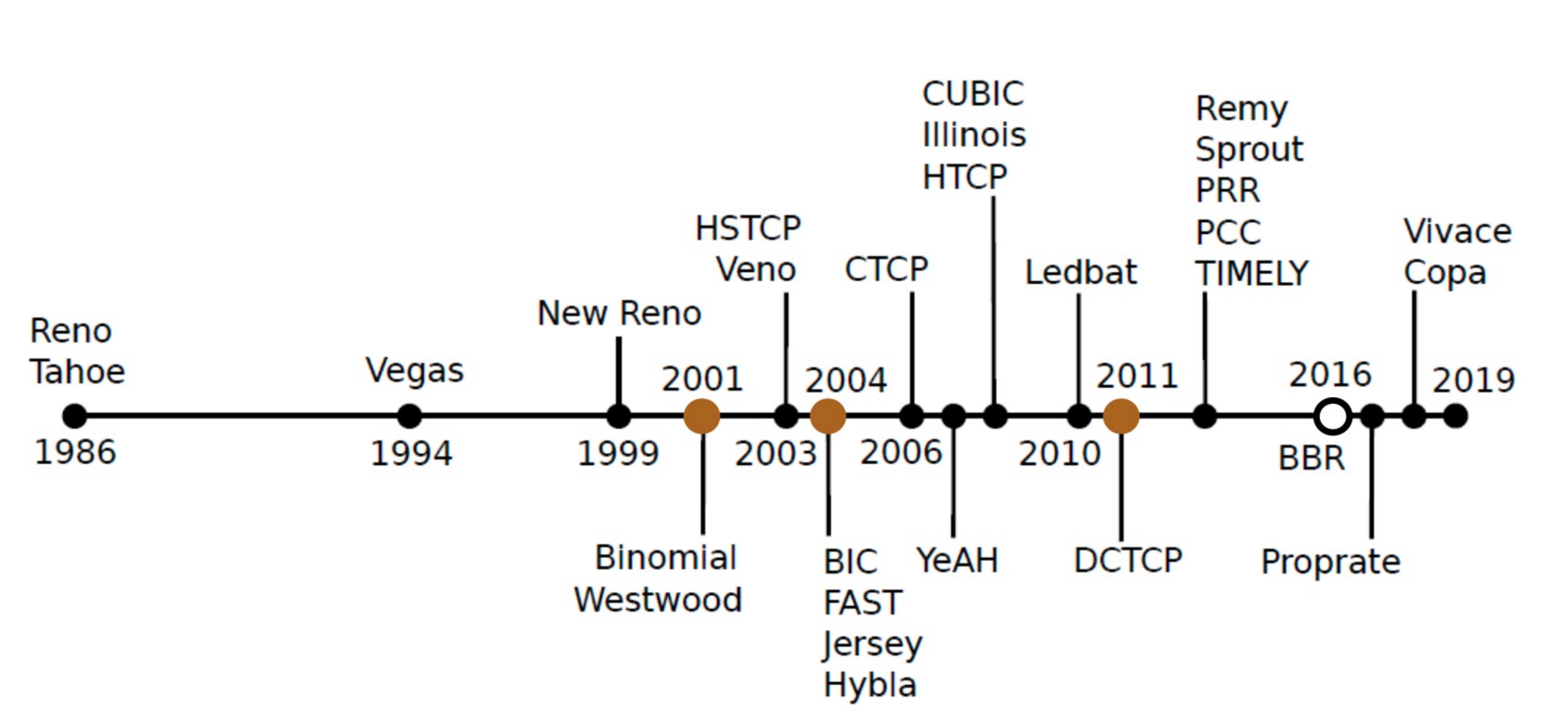
- Maximize throughput
- Recover from packet loss or high RTT
- Short-long "flows"
- Datacenter-specific (low-latency)

⇒ This is a big research area!

This is just the beginning…

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

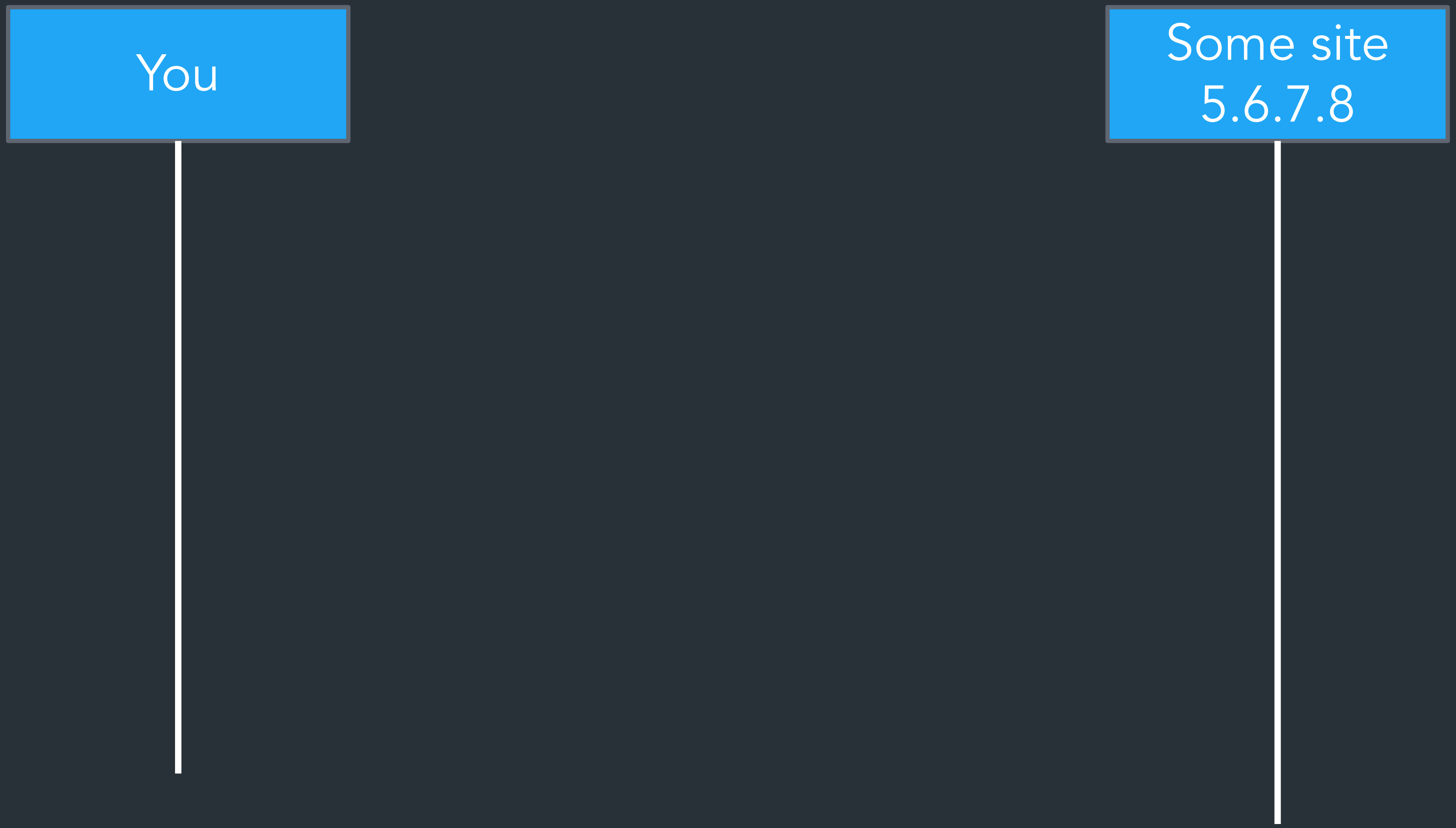
- Tahoe (1988)
- Reno (1990)
- Vegas (1994): Detect based on RTT
- New Reno: Better recovery multiple losses
- Cubic (2006): Linux default, window size scales by cubic function
- BBR (2016): Used by Google, measures bandwidth/RTT


Reno
Tahoe
1986
Vegas
1994
New Reno
1999
2001
Binomial
Westwood
HSTCP
Veno
2003
2004
BIC
FAST
Jersey
Hybla
CTCP
2006
YeAH
CUBIC
Illinois
HTCP
Ledbat
2010
2011
DCTCP
Remy
Sprout
PRR
PCC
TIMELY
2016
BBR
Proprate
Vivace
Copa
2019

DNS

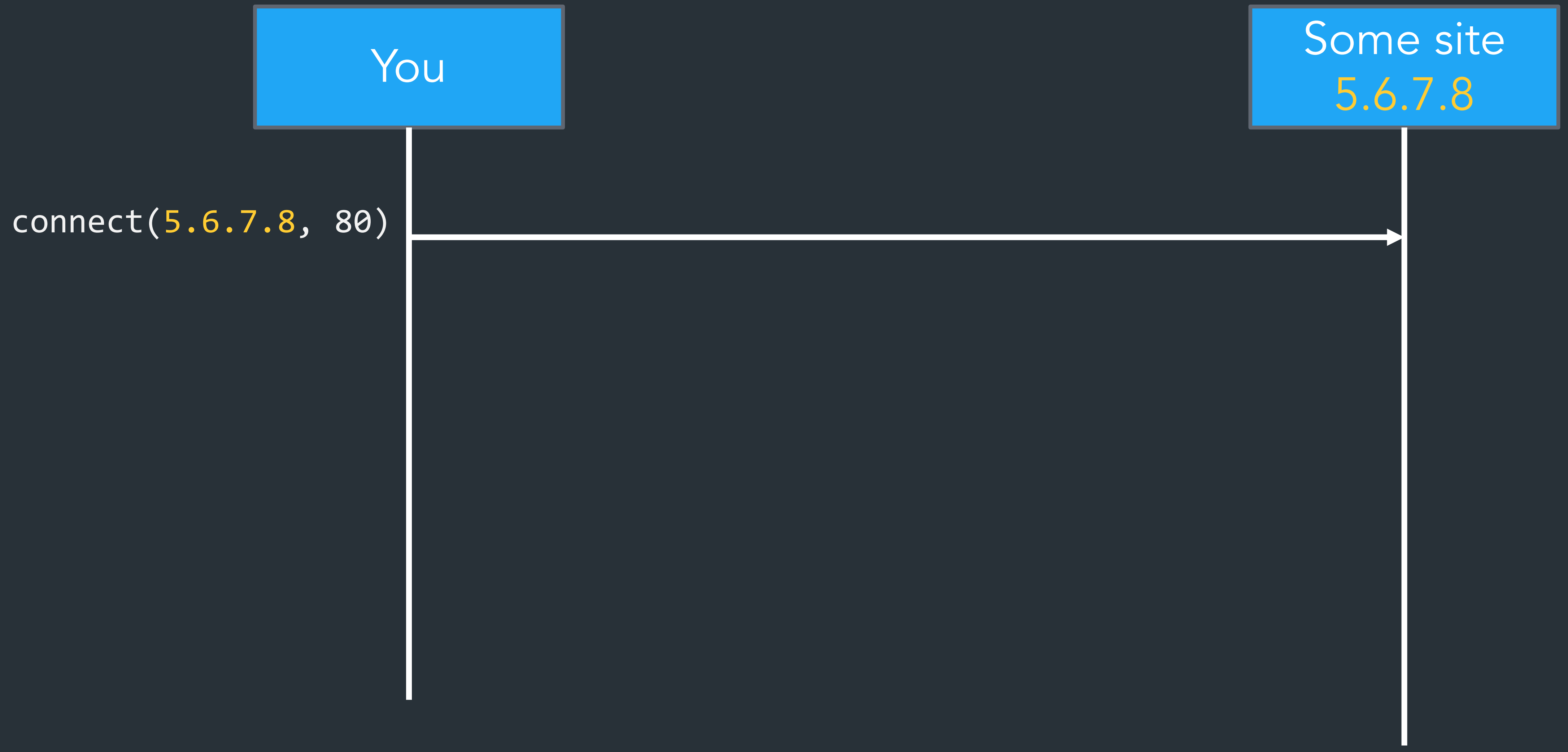
Connecting to a server: the story so far

POV: You want to connect to some website



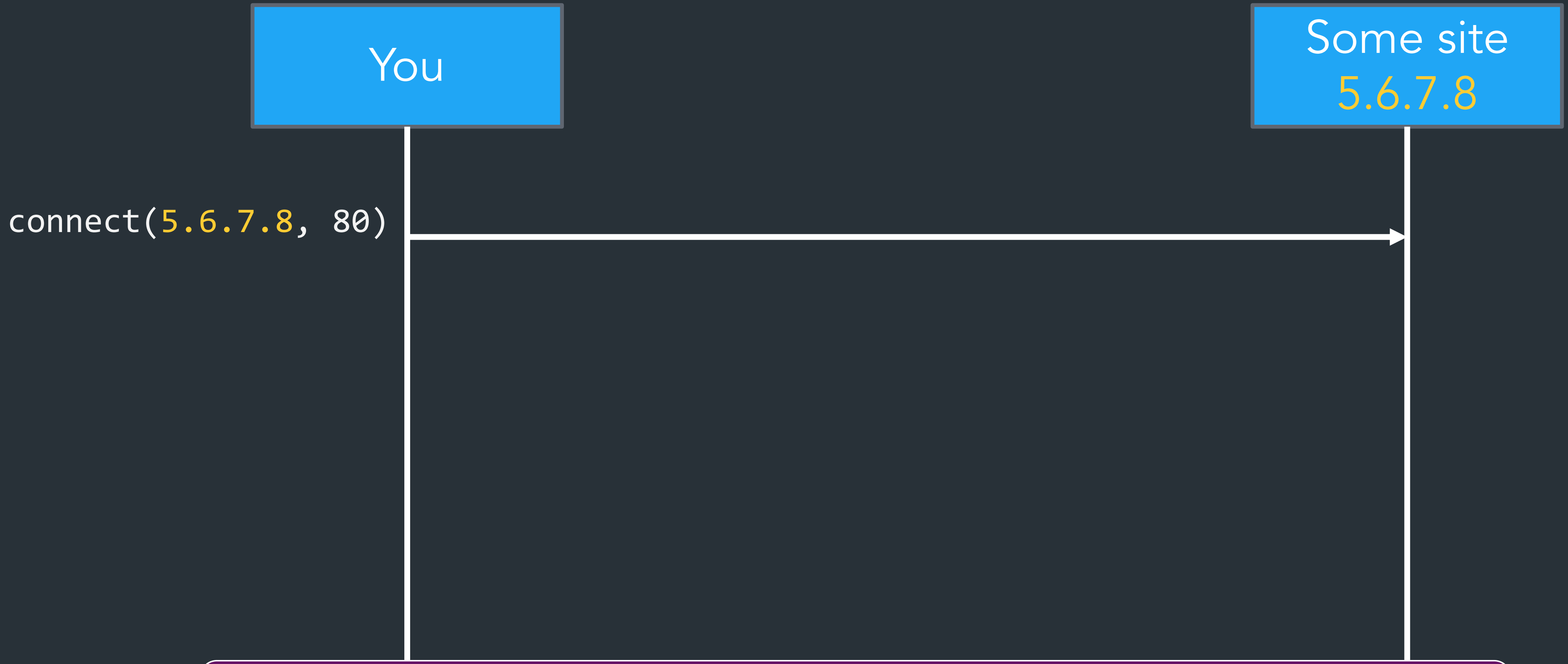
Connecting to a server: the story so far

POV: You want to connect to some website



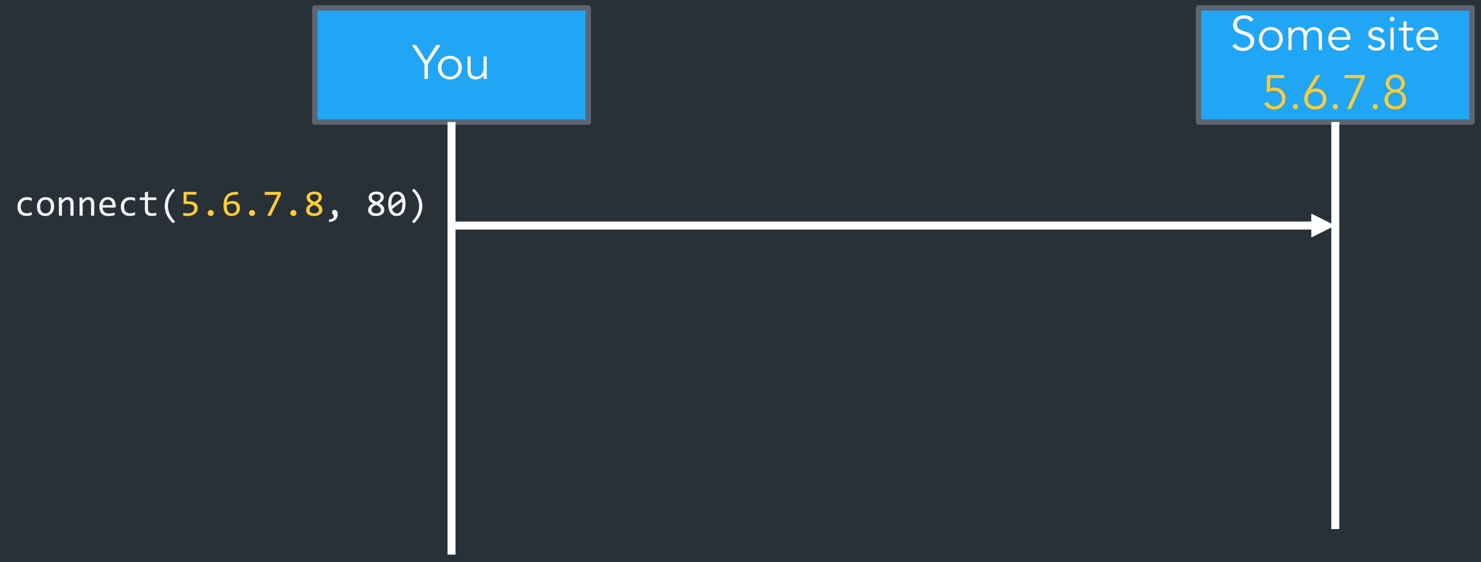
Connecting to a server: the story so far

POV: You want to connect to some website

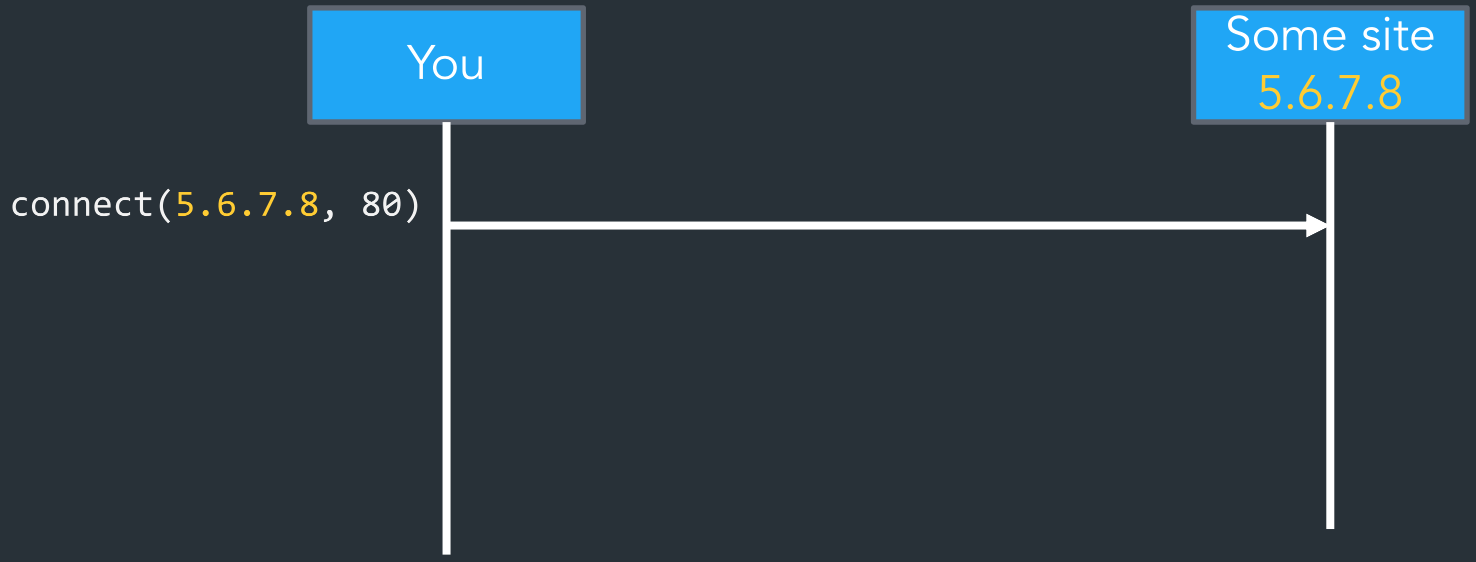


Is this how users interact with the network? No!

Why not? Why is this bad?



<u>Why not? Why is this bad?</u>



- Need to know IP addresses!
  - Users won't know
  - <u>Hosts</u> 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 so far

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

What we have so far

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

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: ❌

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

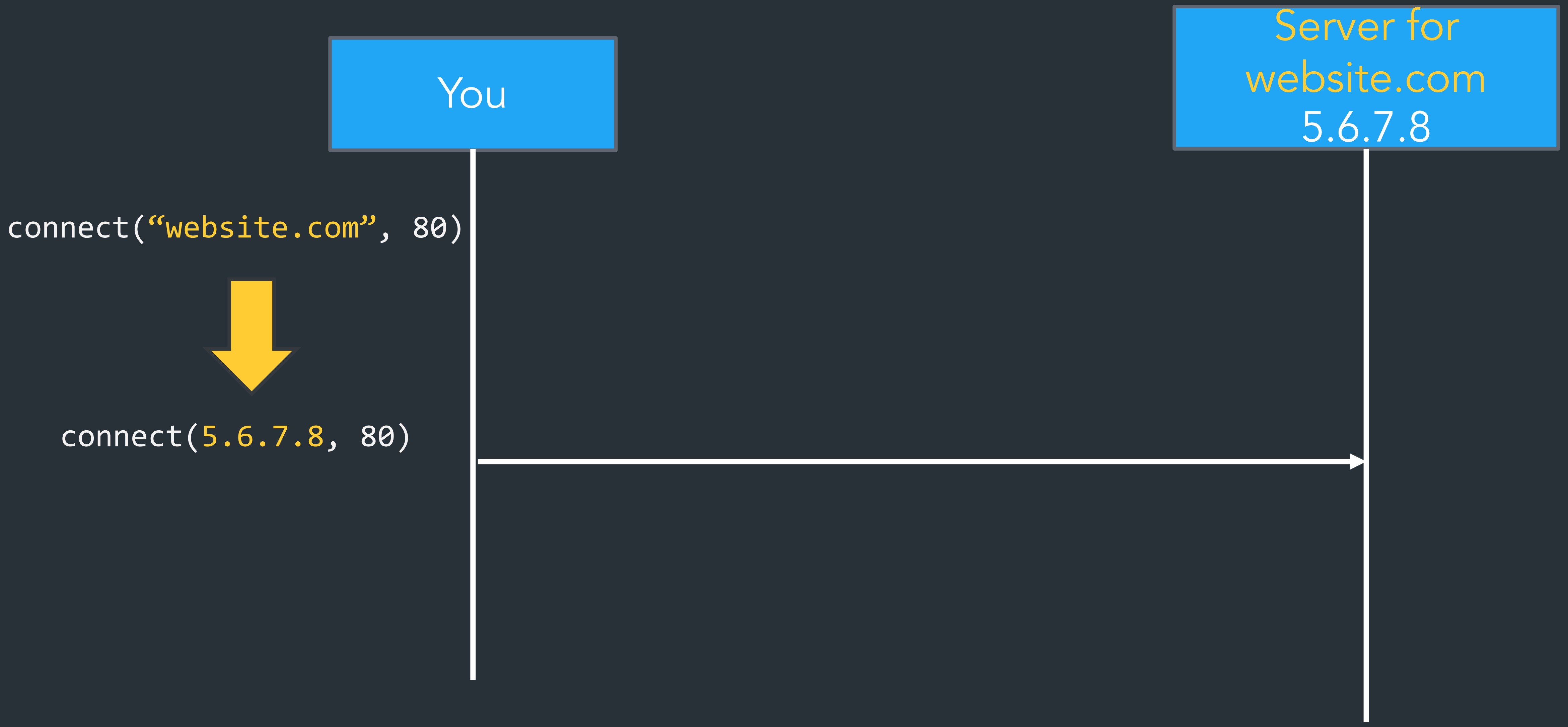
Examples

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

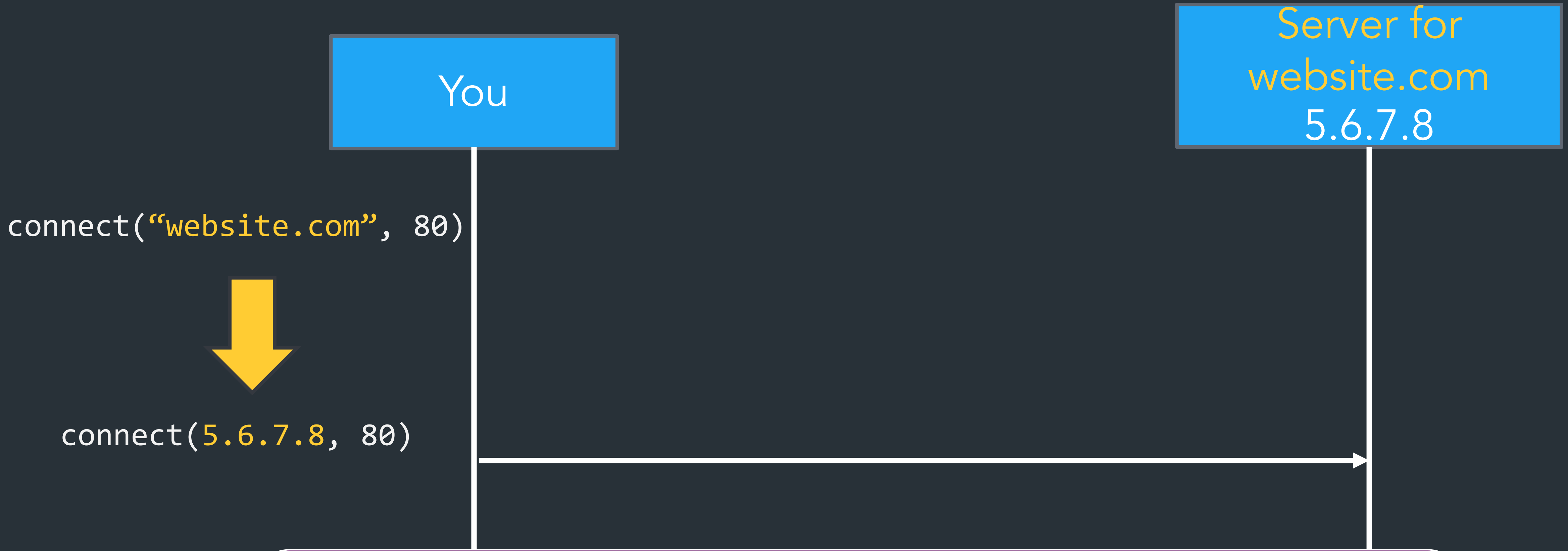
Efficient forwarding: ✅
Human readable: ❌
Scalable for distributed services: ❌

=> 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?

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

What does this mean?

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

Why?

- Names are easier to remember

- Addresses can change underneath
- Useful Multiplexing/sharing

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

Remember ARP?

IP address => Link-layer address

Now: DNS

Names useful to users/applications => IP addresses

Remember ARP?

IP address => Link-layer address

Now: DNS

Names useful to users/applications => IP addresses

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?

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…

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…

Goals for DNS

- Scalability
- Distributed Control
- Fault Tolerance

Goals for DNS

- Scalability
  - Must handle a huge number of records
    - With some software synthesizing names on the fly
  - Must sustain update and lookup load
- Distributed Control
  - Let people control their own names
- Fault Tolerance
  - Minimize lookup failures in face of other network problems

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

The good news

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

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

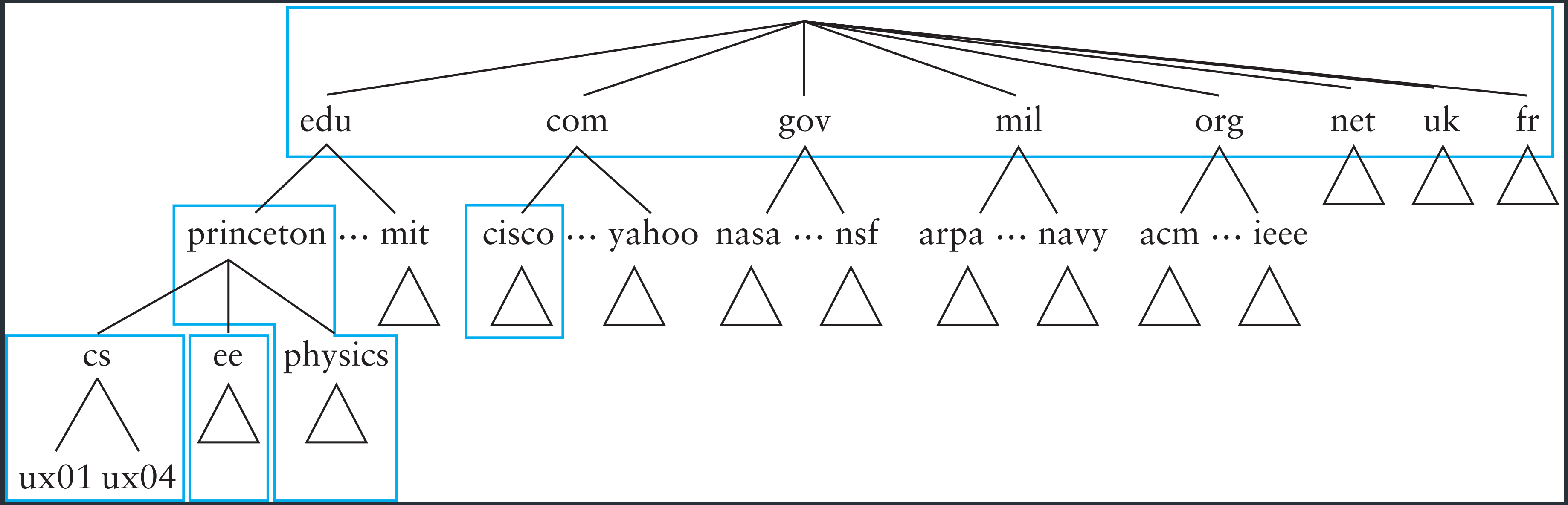
2. Loose consistency

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

How it works

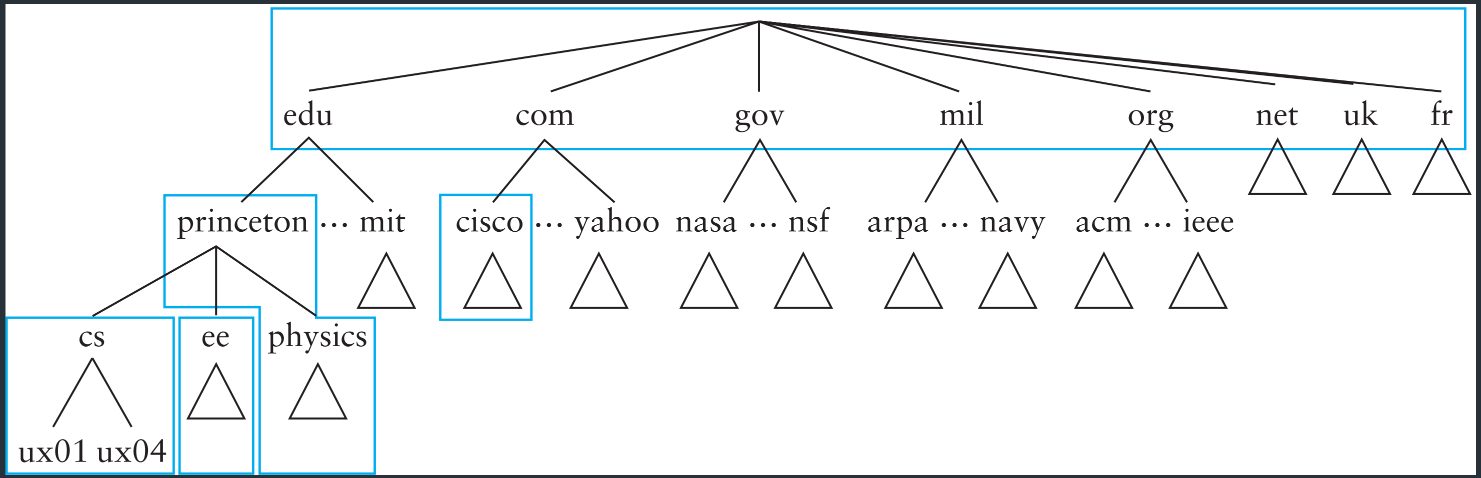
Hierarchical namespace broken into *zones*

`cslab1a.cs.brown.edu`


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



Types of DNS servers

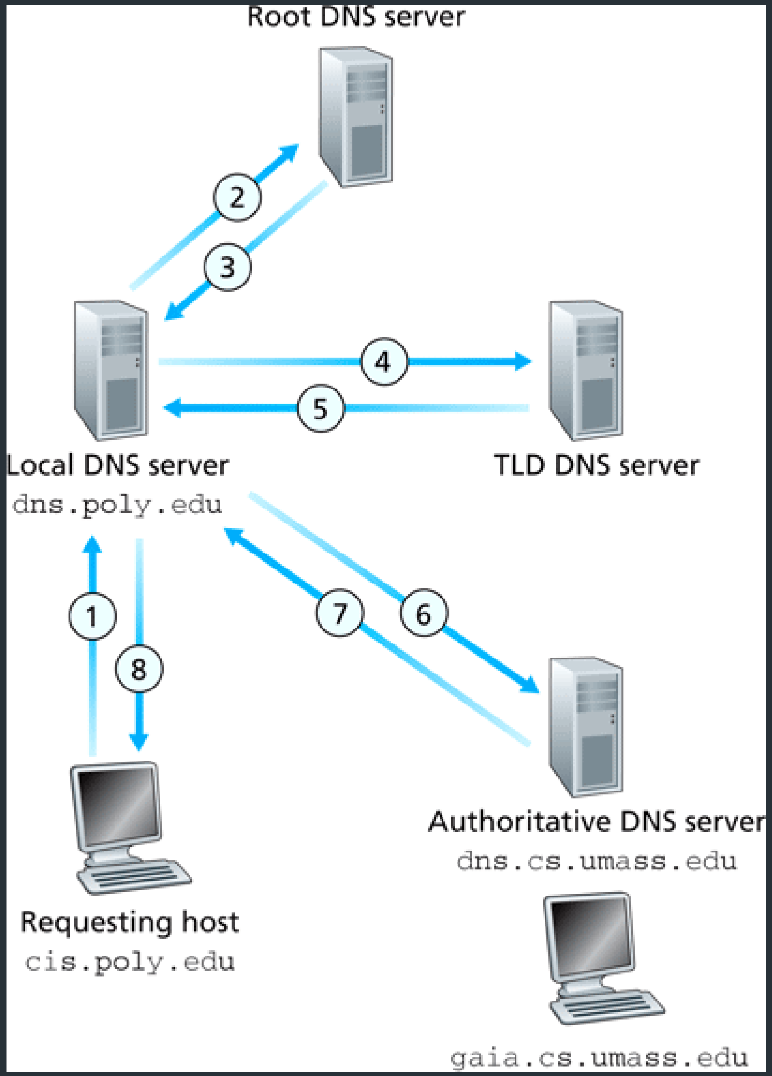
“Types” of DNS servers

- Top Level Domain (TLD) servers
  - Generic domains (e.g., com, org, edu)
  - Country domains (e.g., uk, br, tv, in, ly)
  - Special domains (e.g., arpa)
  - Corporate domains (…)
- Authoritative DNS servers
  - Provides public records for hosts at an organization
  - Can be maintained locally or by a service provider
- Recursive resolvers
  - Big public servers, or local to a network
  - Lots of caching

How a resolver works

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



```
$ 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
```

```
$ 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
```

```
$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
```

```
$ 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
```

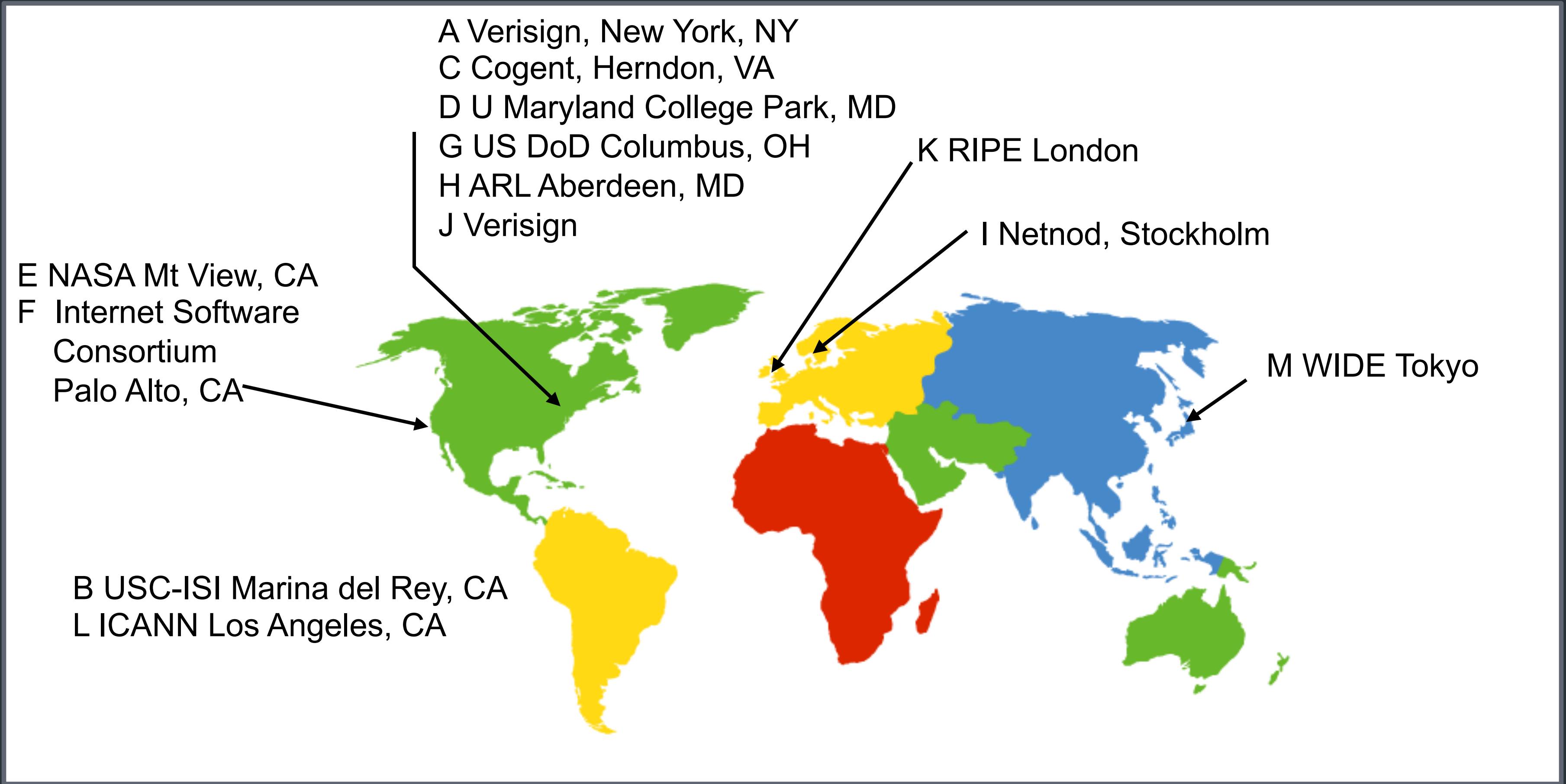
dig: DNS query/debugging tool

Where is the root server?

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

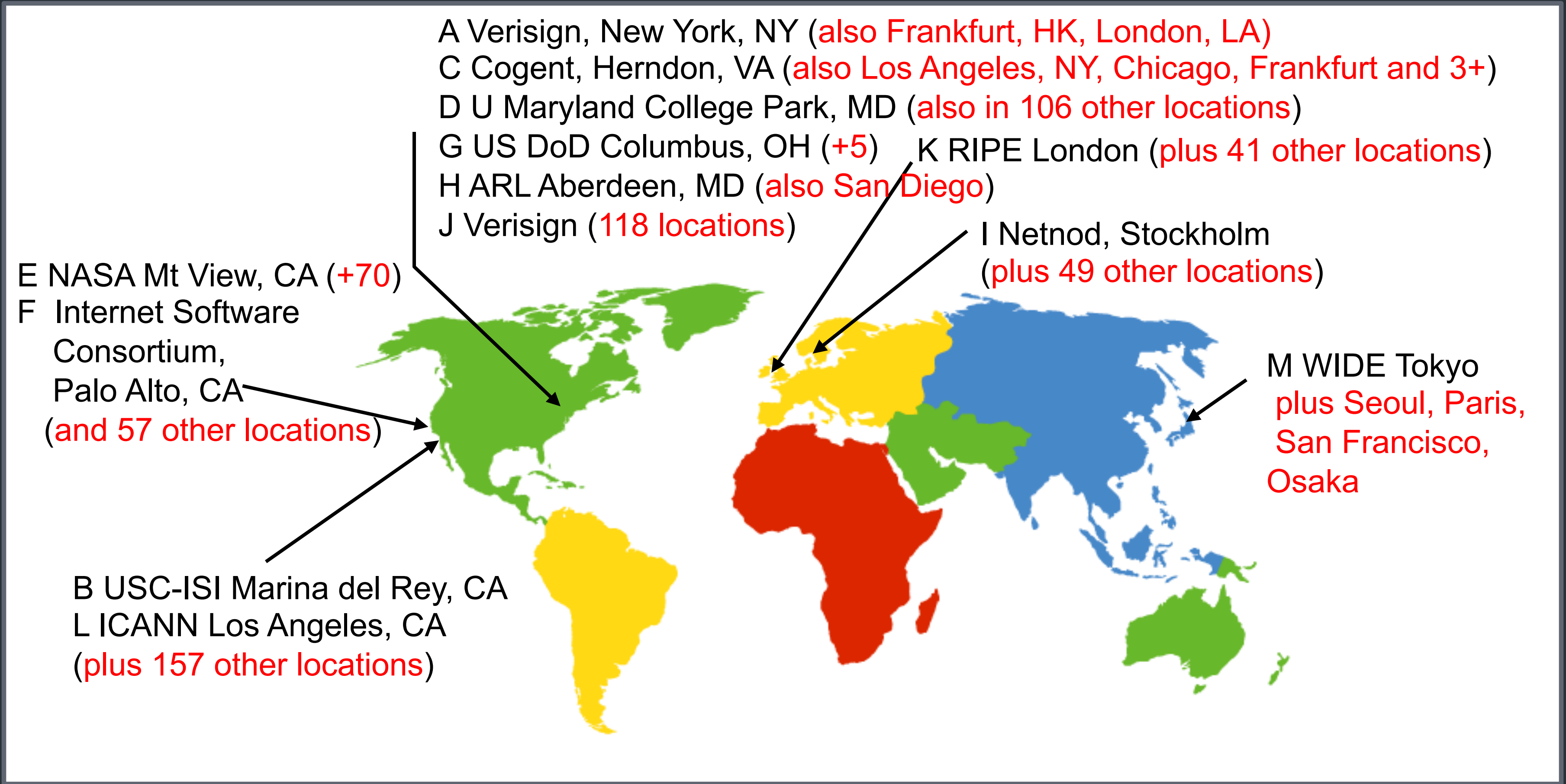
DNS Root Servers

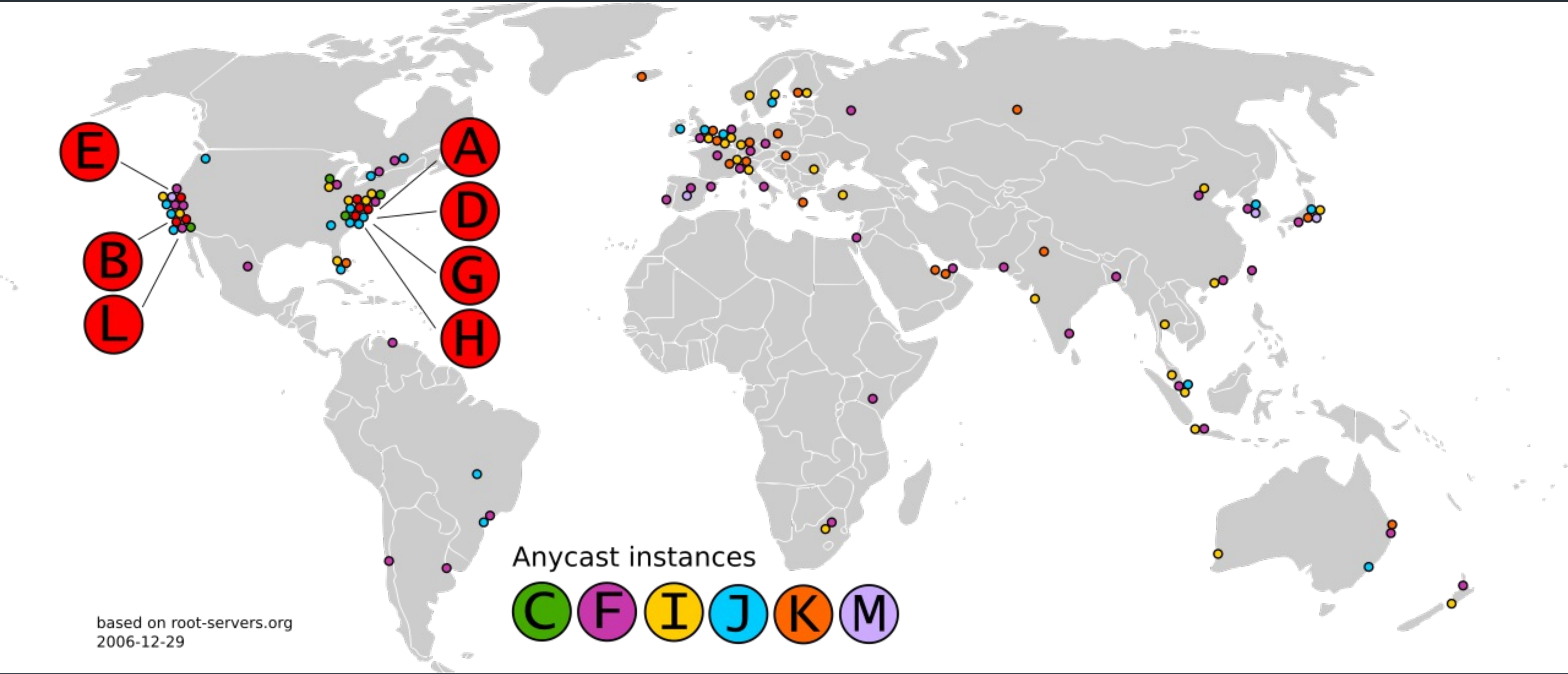
- 13 Root Servers (www.root-servers.org)
  - Labeled A through M (e.g, A.ROOT-SERVERS.NET)
- Does this scale?



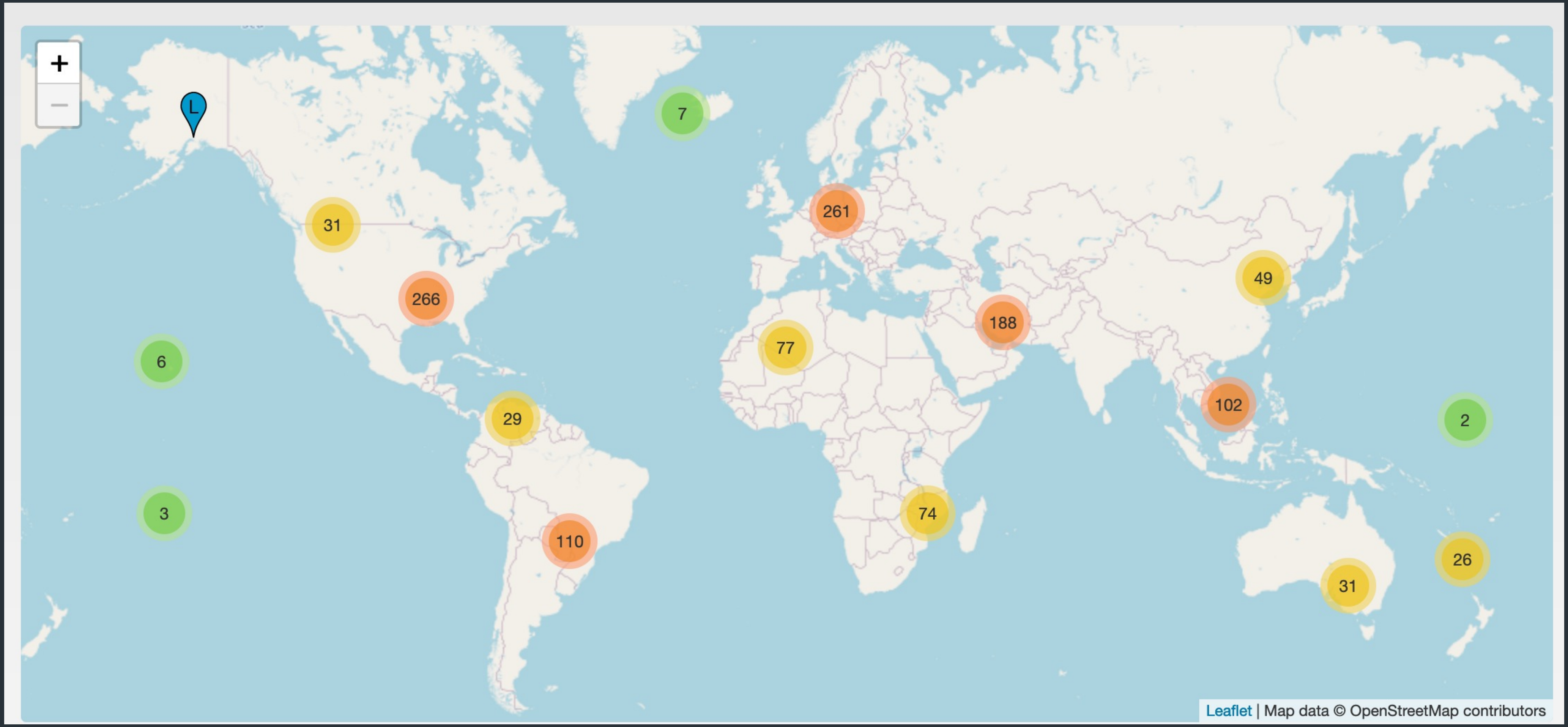
DNS Root Servers

- 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

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

How it scales: caching

Resolvers cache responses to avoid extra recursive/iterative queries

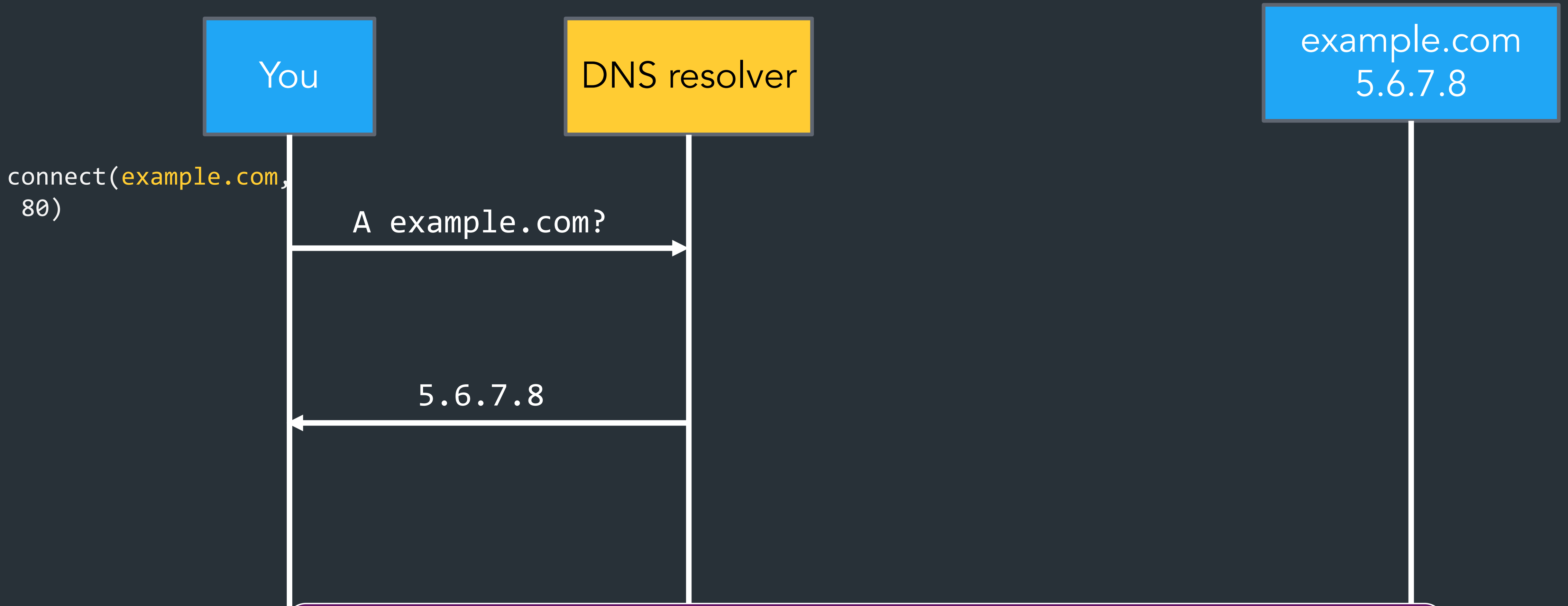
- Many messages => extra computation, extra latency

How long to cache?

=> Every record has a TTL (in seconds), delete when it expires

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

Today



How does this work in practice? What can go wrong?

How it scales: caching

DNS Resolvers cache responses to avoid doing recursive/iterative queries

- Many messages => extra computation, extra latency

How long to cache?

=> Every record has a TTL (in seconds), delete when it 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

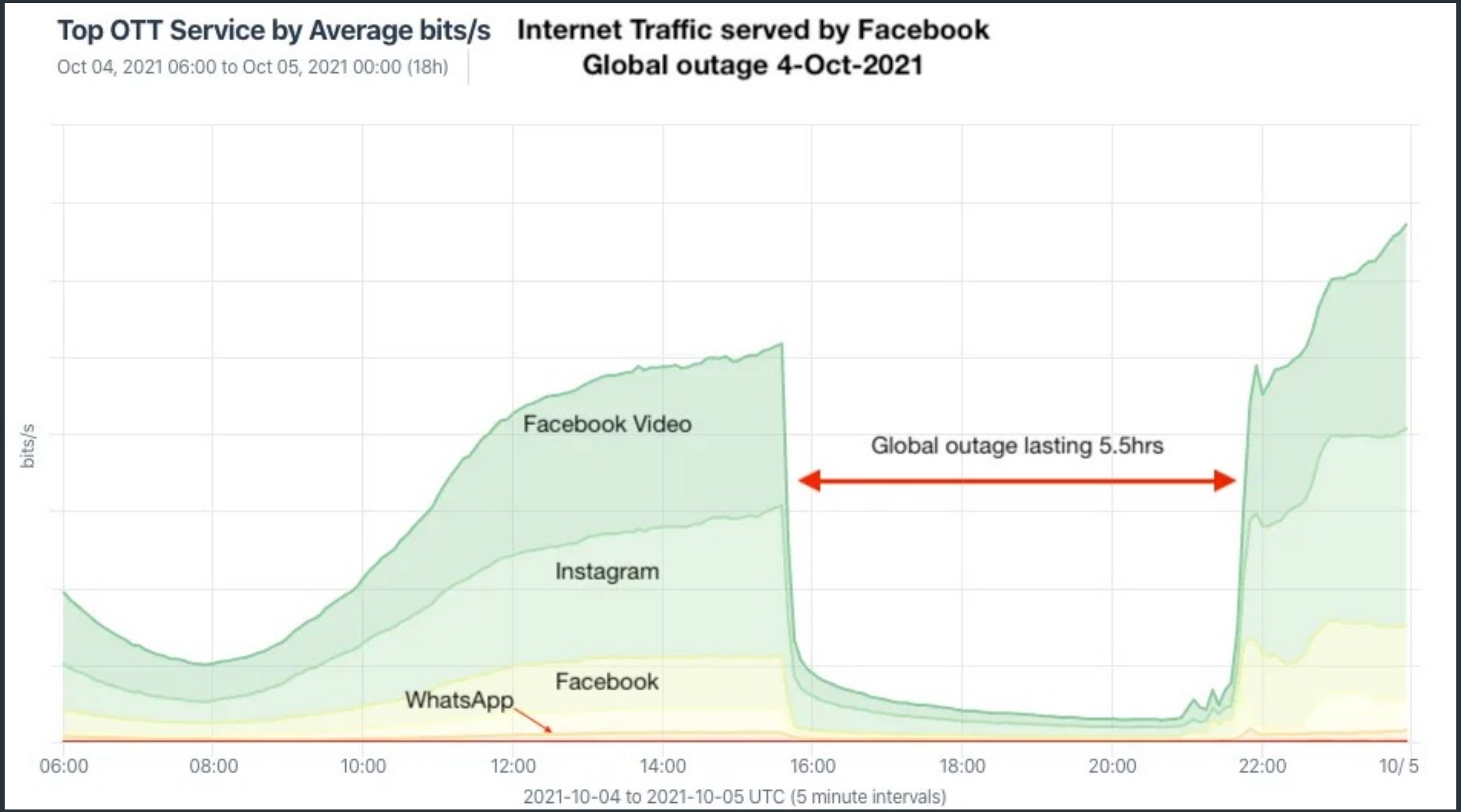
```
$ dig nytimes.com

;; 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
```

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
```

Source

DNS record types

RR Type	Purpose	Example
A	IPv4 Address	`128.148.56.2`
AAAA	IPv6 Address	`2001:470:8956:20::1`

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

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

Reverse DNS

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

Reverse DNS

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

DNS Caching

- Recursive queries are expensive
- Caching greatly reduces overhead
  - Top level servers very rarely change
  - Popular sites visited often
  - Local DNS server caches information from many users
- How long do you store a cached response?
  - Original server tells you: TTL entry
  - Server deletes entry after TTL expires

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 Protocol

- TCP/UDP port 53
- Most traffic uses UDP
  - Lightweight protocol has 512 byte message limit
  - Retry using TCP if UDP fails (e.g., reply truncated)
- Bit in query determines if query is recursive

DNS Example

```
$ 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
```

When server doesn’t know all info…

```
 % dig +norec cs.brown.edu @j.root-servers.net


; <<>> DiG 9.10.6 <<>> +norec cs.brown.edu @j.root-servers.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61618
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27


;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;cs.brown.edu. IN A


;; AUTHORITY SECTION:
edu. 172800 IN NS a.edu-servers.net.
edu. 172800 IN NS b.edu-servers.net.
edu. 172800 IN NS l.edu-servers.net.
edu. 172800 IN NS m.edu-servers.net.


;; ADDITIONAL SECTION:
a.edu-servers.net. 172800 IN A 192.5.6.30
b.edu-servers.net. 172800 IN A 192.33.14.30
c.edu-servers.net. 172800 IN A 192.26.92.30
d.edu-servers.net. 172800 IN A 192.31.80.30
e.edu-servers.net. 172800 IN A 192.12.94.30
```

Example

```
dig . ns

dig +norec www.cs.brown.edu @a.root-servers.net

dig +norec www.cs.brown.edu @a.edu-servers.net

dig +norec www.cs.brown.edu @bru-ns1.brown.edu

    www.cs.brown.edu.   86400 IN    A     128.148.32.110
```

Resource Records

All DNS info represented as resource records (RR)

name [ttl] [class] type rdata

- name: domain name
- TTL: time to live in seconds
- class: for extensibility, normally IN (1) “Internet”
- type: type of the record
- rdata: resource data dependent on the type

- Example RRs

```
www.cs.brown.edu.     86400  IN    A     128.148.32.110
cs.brown.edu.         86400  IN    NS    dns.cs.brown.edu.
cs.brown.edu.         86400  IN    NS    ns1.ucsb.edu.
```

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`

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

Inserting a Record in DNS

Your new startup helpme.com

Some important details

- How do local servers find root servers?
  - DNS lookup on a.root-servers.net ?
  - Servers configured with *root cache* file
  - Contains root name servers and their addresses

```
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
...
```

- How do you get addresses of other name servers?
  - To obtain the address of www.cs.brown.edu, ask a.edu-servers.net, says a.root-servers.net
  - How do you find a.edu-servers.net?
  - Glue records: A records in parent zone

Other uses of DNS

- Local multicast DNS
  - Used for service discovery
  - Made popular by Apple
  - This is how you learn of different Apple TVs in the building
- Load balancing
- CDNs (more on this later)

Reliability

- Answers may contain several alternate servers
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Use same identifier for all queries
  - Don't care which server responds, take first answer

Inserting a Record in DNS

- Your new startup helpme.com
- Get a block of addresses from ISP
  - Say 212.44.9.0/24
- Register helpme.com at namecheap.com (for ex.)
  - Provide name and address of your authoritative name server (primary and secondary)
  - Registrar inserts RR pair into the **.com** TLD server:
    - helpme.com NS dns1.helpme.com
    - dns1.helpme.com A 212.44.9.120
- Configure your authoritative server (dns1.helpme.com)
  - Type A record for www.helpme.com
  - Type MX record for helpme.com

Inserting a Record in DNS, cont

- Need to provide reverse PTR bindings
  - E.g., 212.44.9.120 -> dns1.helpme.com
- Configure your dns server to serve the 9.44.212.in-addr.arpa zone
  - Need to add a record of this NS into the parent zone (44.212.in-addr.arpa)
- Insert the bindings into the 9.44.212.in-addr.arpa zone

DNS Security

- You go to starbucks, how does your browser find www.google.com?
  - Ask local name server, obtained from DHCP

```
> Option: (15) Domain Name
v Option: (6) Domain Name Server
    Length: 12
    Domain Name Server: 1.1.1.1
    Domain Name Server: 4.2.2.1
    Domain Name Server: 8.8.8.8
```

- Can you trust this DNS server?

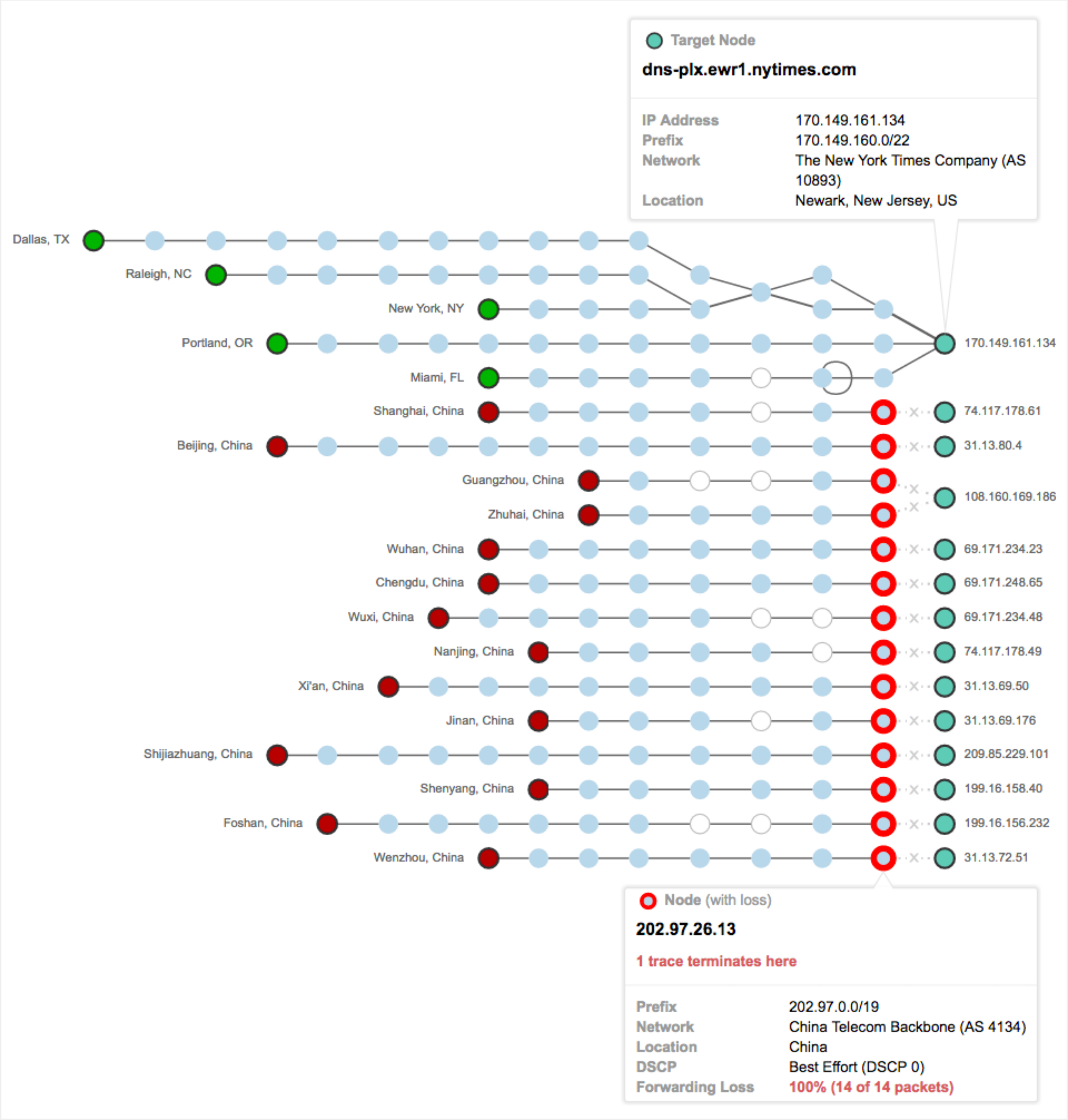
Great Firewall of CIT

If attacker is on the path (say, it is the ISP, or a malicious version of TStaff), what could they do?

- Can sniff all DNS queries
- Send fake responses back first
- Could do this selectively, to direct facebook.com to cs.brown.edu, for example…

Great Firewall of CIT

If attacker is on the path (say, it is the ISP, or a malicious version of TStaff), what could they do?



https://blog.thousandeyes.com/monitoring-dns-in-china/

Public DNS

Public DNS resolvers provided by cloud companies and ISPs

- 8.8.8.8 (Google)
- 1.1.1.1 (Cloudflare)
- … and others

Why do this?


DNS: 8.8.8.8 kuşun Ötsün !
Alternatif: 8.8.4.4
@KADIKOYBASKA

“Helpful” ISPs

- Many ISPs hijack NXDOMAIN responses to “help” by offering search and advertisement related to the domain
- E.g., www.bicycleisntadomain.com doesn’t (currently) exist
  - Could return a page with search and ads on bicycles (or domain registrations?)

What can be done?

Some defenses against DNS spoofing/hijacking

What can be done?

Some defenses against DNS spoofing/hijacking

- DNSSEC: protocol to sign/verify hierarchy of DNS lookups
  - Expensive to deploy, hierarchy must support at all levels
  - APNIC DNSSEC monitor: https://stats.labs.apnic.net/dnssec
  - https://www.internetsociety.org/resources/deploy360/2012/nist-ipv6-and-dnssec-statistics-6/
- Tunneling DNS: client uses DNS via more secure protocol
  - DNS over HTTPS
  - DNS over TLS

More on DNS

Structure of a DNS Message

- Same format for queries and replies
  - Query has 0 RRs in Answer/Authority/Additional
  - Reply includes question, plus has RRs
- Authority allows for delegation
- Additional for glue, other RRs client might need

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

Header format

- Id: match response to query; QR: 0 query/1 response
- RCODE: error code.
- AA: authoritative answer, TC: truncated,
- RD: recursion desired, RA: recursion availab

```
                                 1  1  1  1  1  1
   0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |                      ID                       |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |                    QDCOUNT                    |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |                    ANCOUNT                    |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |                    NSCOUNT                    |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |                    ARCOUNT                    |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Other RR Types

- CNAME (canonical name): specifies an alias

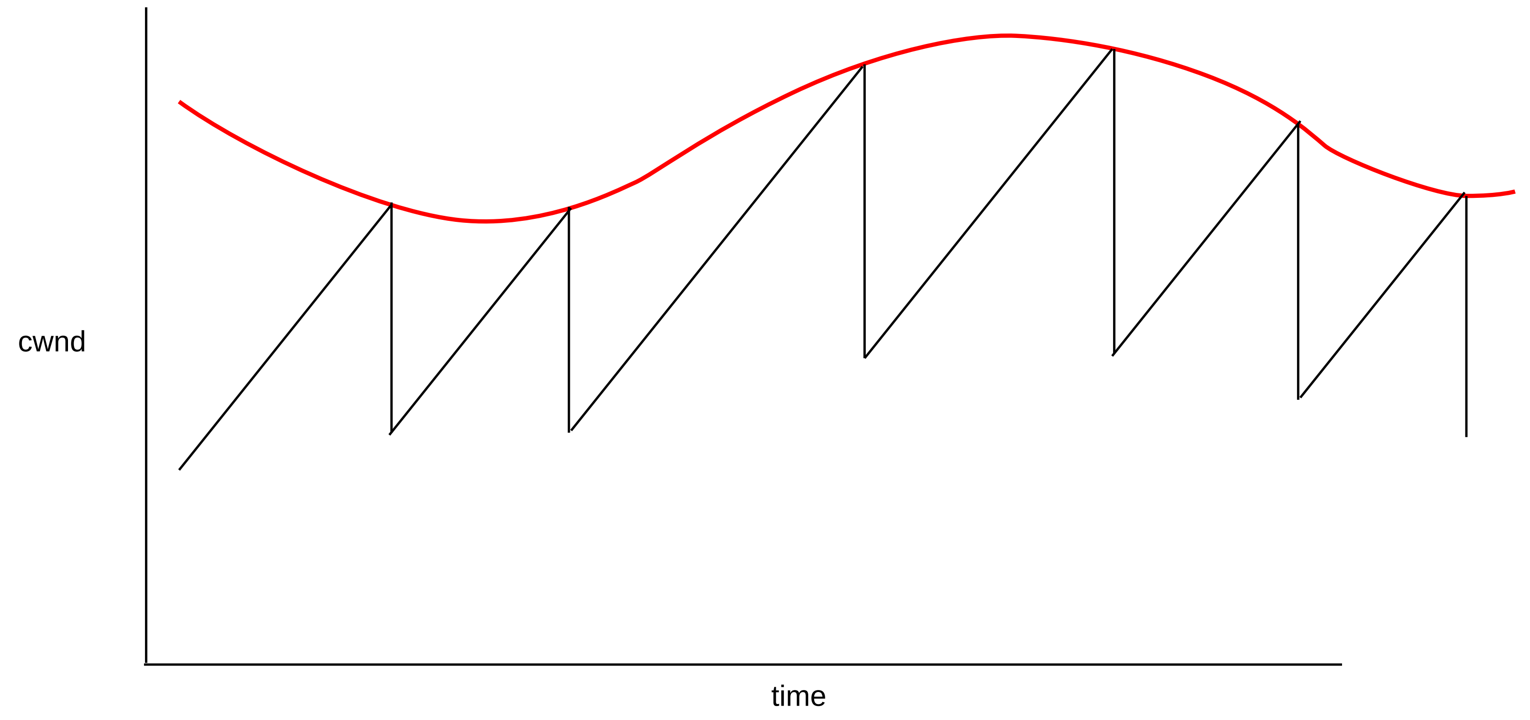
```
www.google.com.              446199  IN      CNAME   www.l.google.com.
www.l.google.com.    300     IN      A       72.14.204.147
```

- MX record: specifies servers to handle mail for a domain (the part after the @ in email addr)
  - Different for historical reasons
- SOA (start of authority)
  - Information about a DNS zone and the server responsible for the zone
- PTR (reverse lookup)

```
7.34.148.128.in-addr.arpa. 86400 IN   PTR     quanto.cs.brown.edu.
```

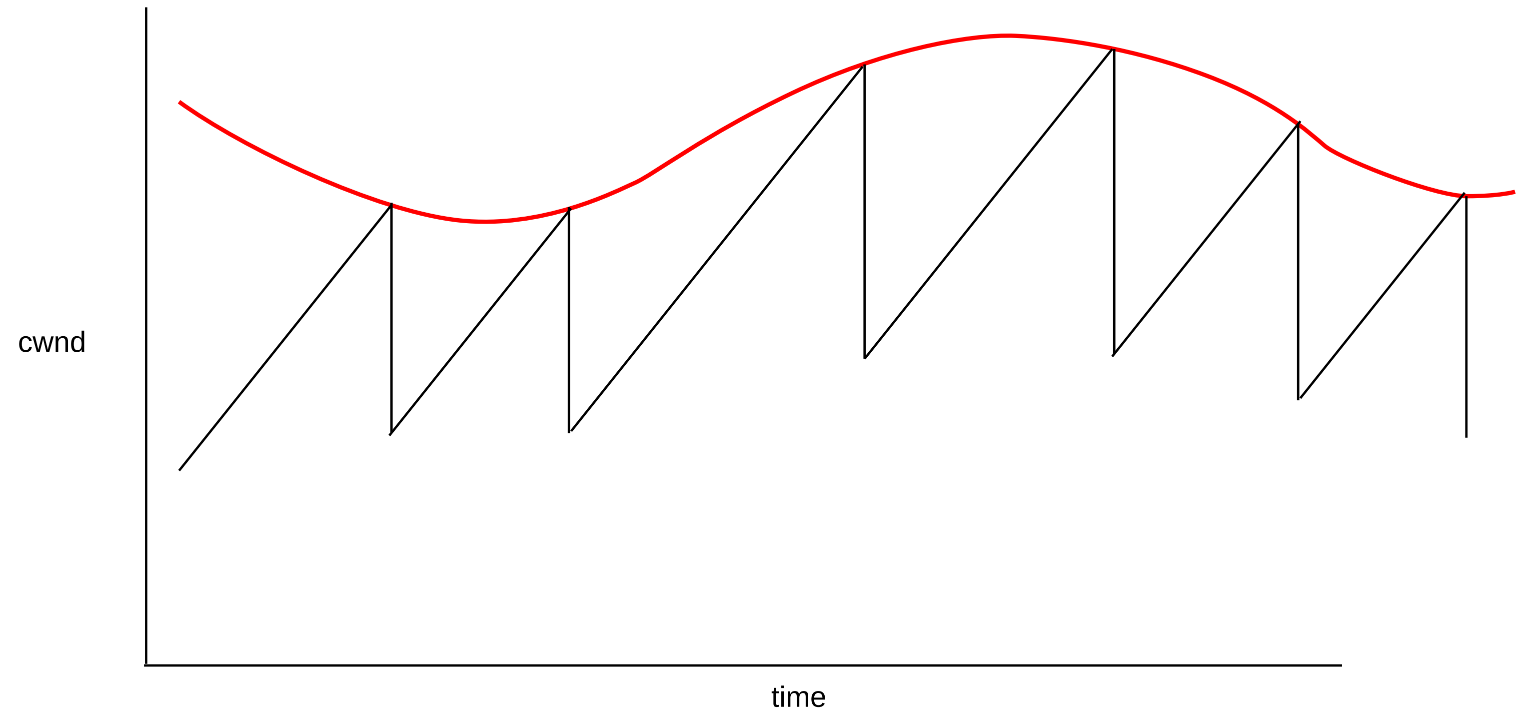
More on CC

Traditional, Loss-based CC



TCP Sawtooth, red curve represents the network capacity

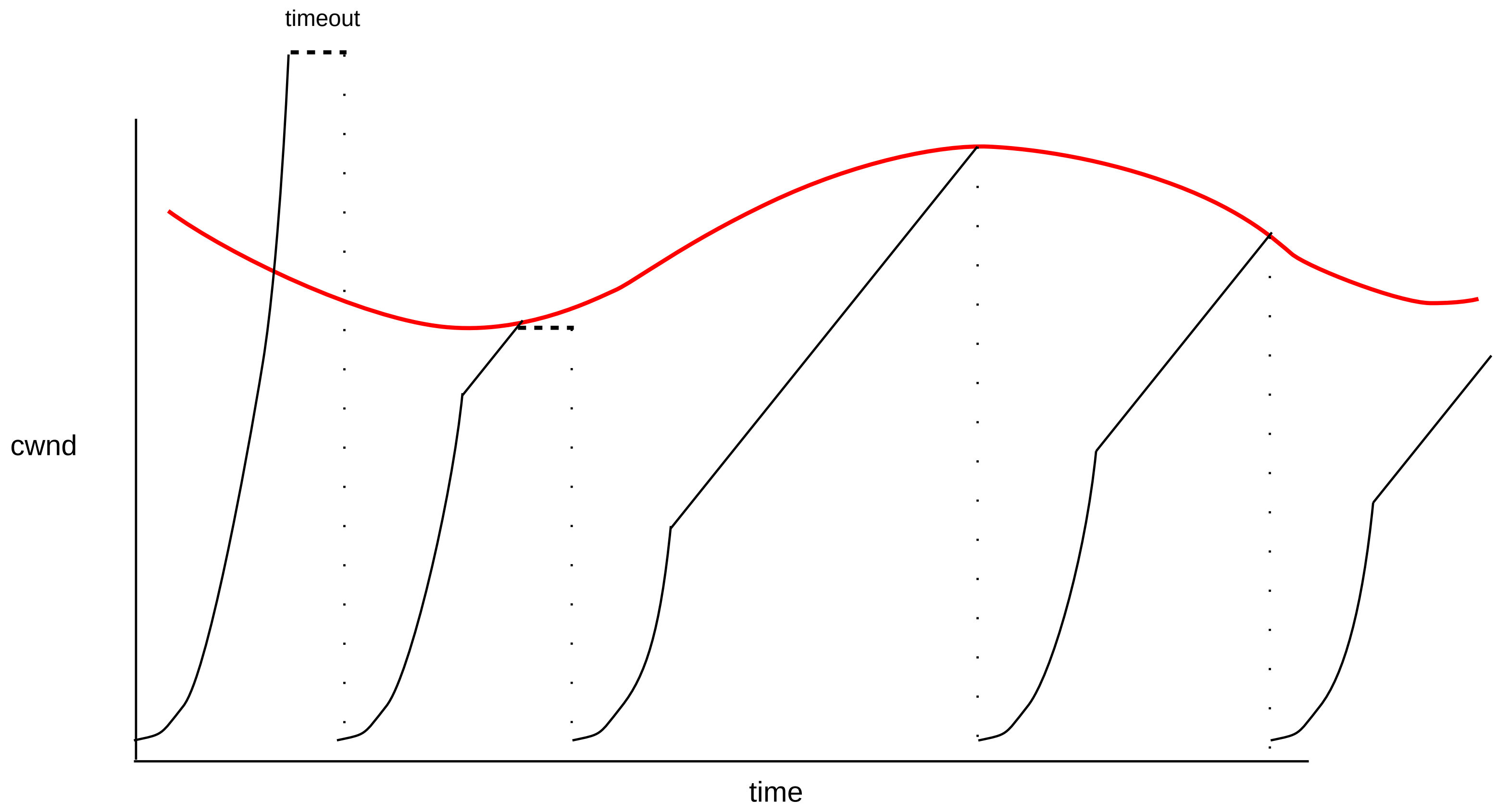
Traditional, Loss-based CC



TCP Sawtooth, red curve represents the network capacity

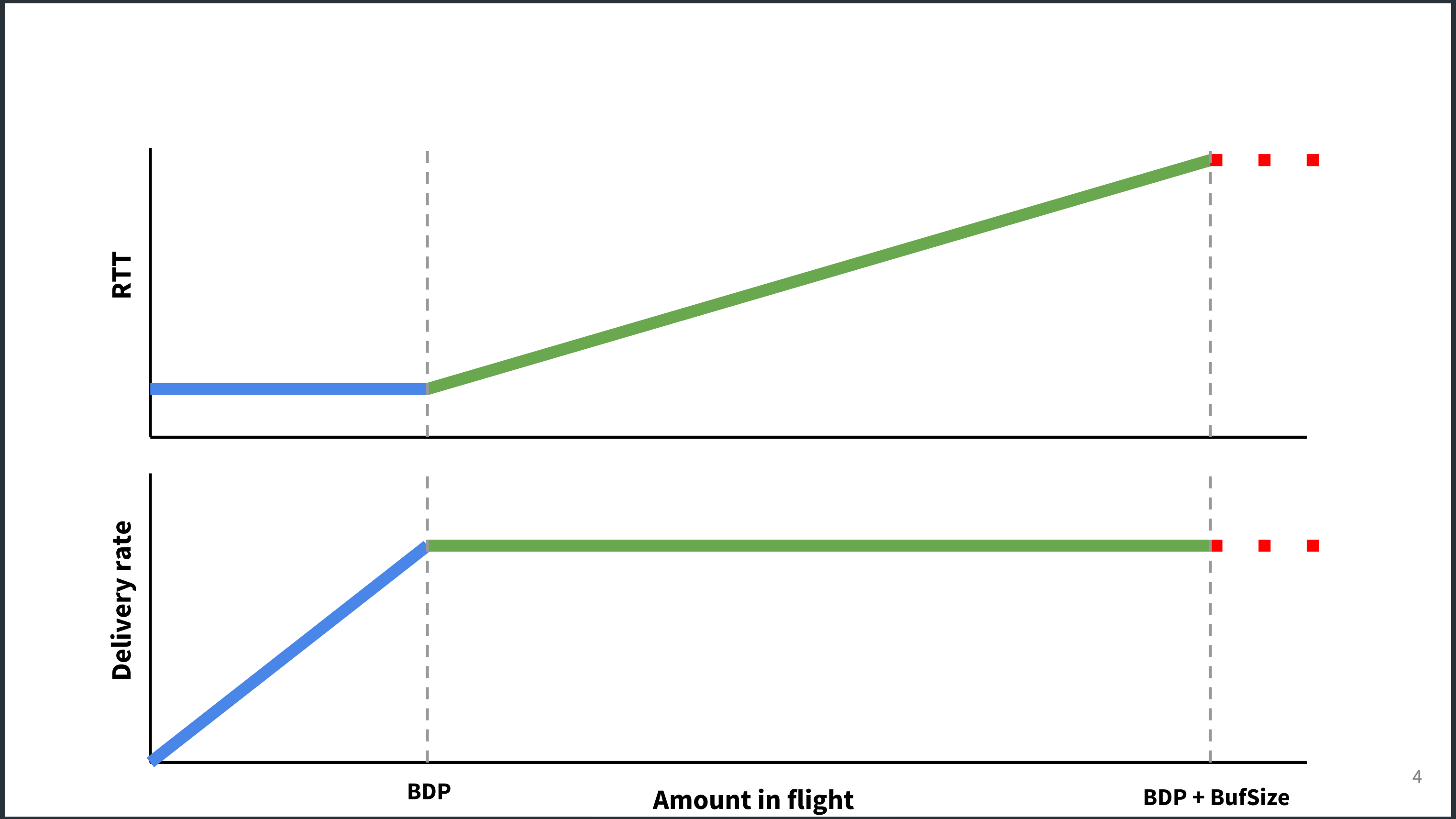
=> Additive Increase, Multiplicative Decrease (AIMD)

In practice: AIMD + Slow Start (SS)



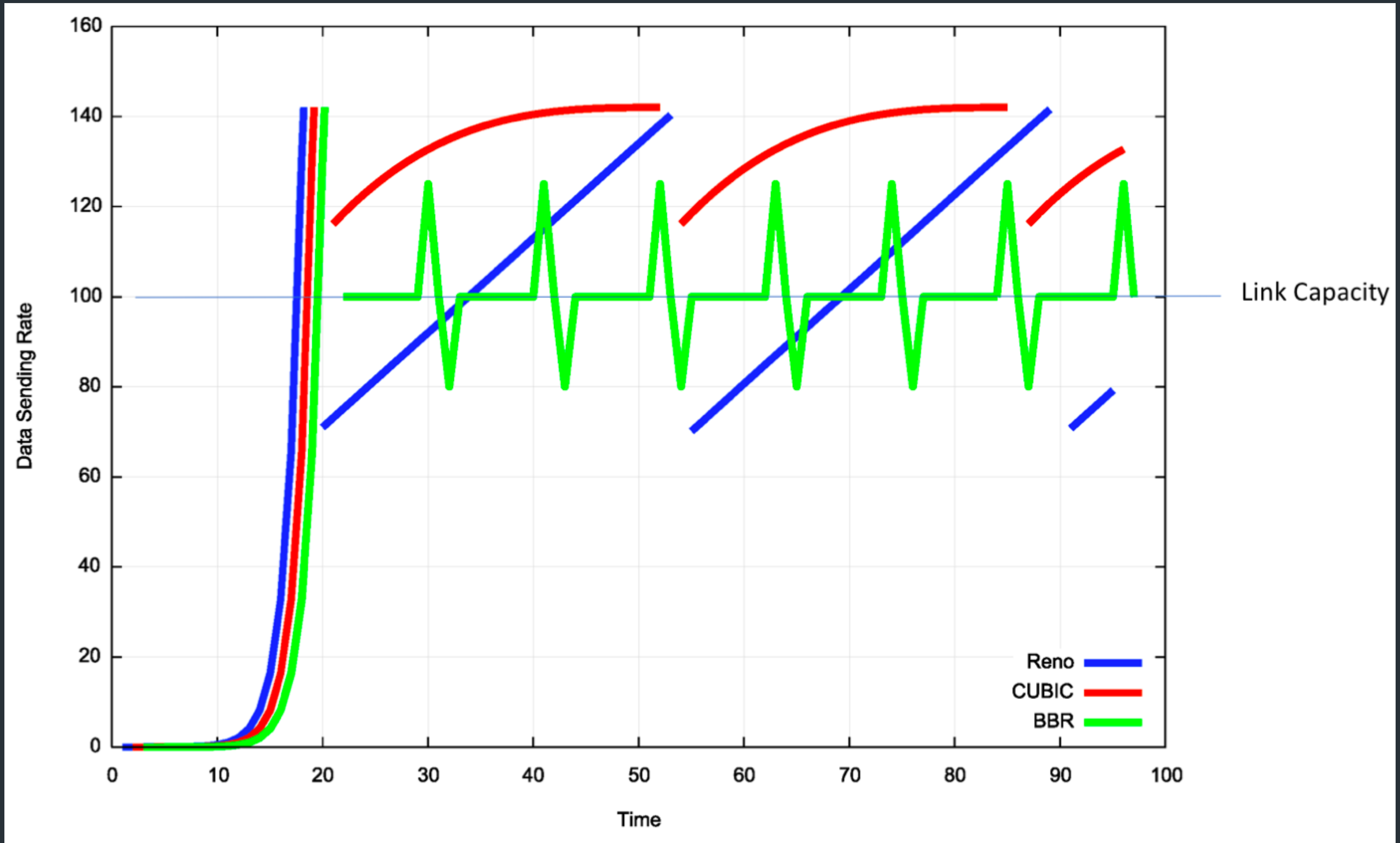
TCP Tahoe Sawtooth, red curve represents the network capacity
Slow Start is used after each packet loss until ssthresh is reached

BBR: what's different



"BBR congestion control"

BBR



From: https://labs.ripe.net/Members/gih/bbr-tcp

Another way: ECN

What if routers/switches could help?

- Routers/switches set bit in packet when experiencing congestion
- When sender sees congestion bit, scales back cwnd

In theory, no dropped packets!

Another way: ECN

What if routers/switches could help?

- Routers/switches set bit in packet when experiencing congestion
- When sender sees congestion bit, scales back cwnd

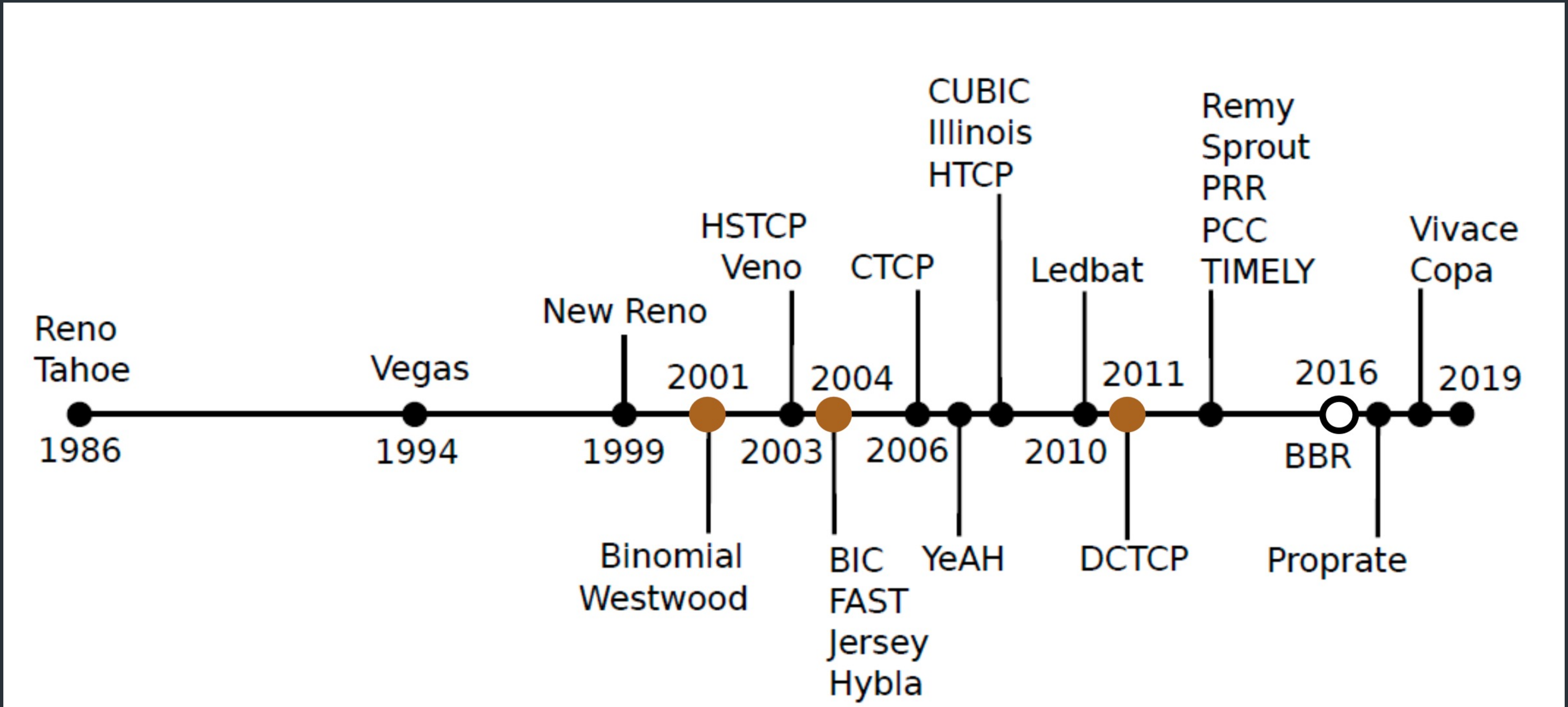
In theory, no dropped packets!

Special purpose example: DCTCP (2010)

Designed for datacenter usage only

- Want to avoid queuing as much as possible
- Routers/switches mark packets with ECN bit in header
- When this happens, senders scale back dramatically

Timeline of (some!) congestion control implementations



["The great Internet congestion control census" (2019)]()