Lecture 18

Today
- IPoAC
- DNS and what can go wrong

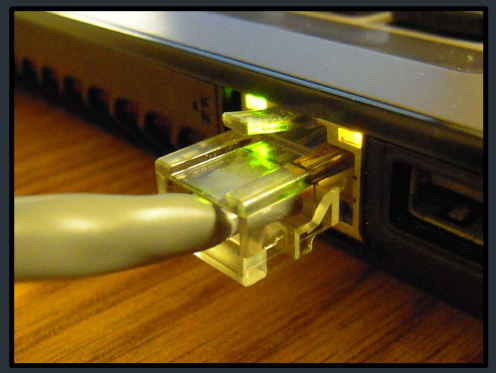
Administrivia
- HW3 due Friday
- TCP SRC: technically due Friday, but soft deadline
- Look for announcement about TCP milestone II signups
- HW4 out Friday

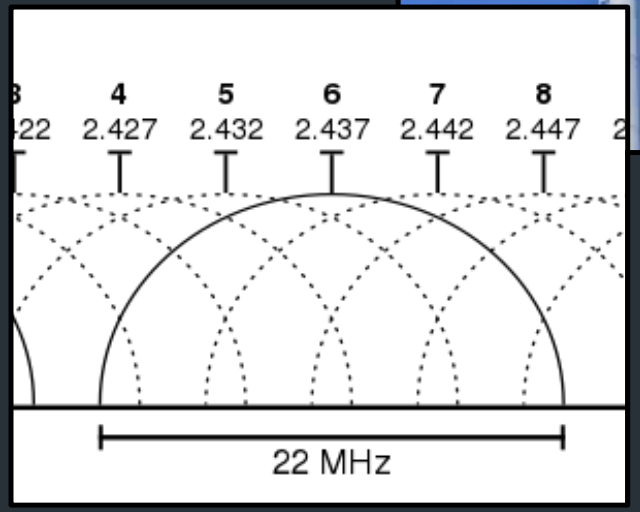
IPoAC

How can we improve the physical layer?

Traditional links have fixed bandwidth

- Media limits what frequencies can be used for signal
- Places upper bound on channel capacity





What if we weren’t constrained by the EM spectrum?

How else can we transmit data?

Network Working Group
Request for Comments: 1149

D. Waitzman
BBN STC
1 April 1990

A Standard for the Transmission of IP Datagrams on Avian Carriers

Status of this Memo

This memo describes an experimental method for the encapsulation of IP datagrams in avian carriers. This specification is primarily useful in Metropolitan Area Networks. This is an experimental, not recommended standard. Distribution of this memo is unlimited.

Overview and Rational

Avian carriers can provide high delay, low throughput, and low altitude service. The connection topology is limited to a single point-to-point path for each carrier, used with standard carriers, but many carriers can be used without significant interference with each other, outside of early spring. This is because of the 3D ether space available to the carriers, in contrast to the 1D ether used by

RFC1149: IPoAC

IP over Avian Carriers (1 April 1990)

- High delay, low throughput, low altitude datagram service
- Nearly unlimited movement in 3D etherspace
- Intrinsic collision avoidance
- Typical MTU: 256 milligrams

Network Working Group
Request for Comments: 1149

D. Waitzman
BBN STC
1 April 1990

A Standard for the Transmission of IP Datagrams on Avian Carriers

Status of this Memo

This memo describes an experimental method for the encapsulation of IP datagrams in avian carriers. This specification is primarily

IPoAC: Design

IPoAC: Implementation

Proof of concept: 28 April 2001
Bergen, Norway
https://web.archive.org/web/20140215072548/http://www.blug.linux.no/rfc1149/

IPoAC in practice

```
$ ping -c 9 -i 900 10.0.3.1
PING 10.0.3.1 (10.0.3.1): 56 data bytes
64 bytes from 10.0.3.1: icmp_seq=0 ttl=255 time=6165731.1 ms
64 bytes from 10.0.3.1: icmp_seq=4 ttl=255 time=3211900.8 ms
64 bytes from 10.0.3.1: icmp_seq=2 ttl=255 time=5124922.8 ms
64 bytes from 10.0.3.1: icmp_seq=1 ttl=255 time=6388671.9 ms

--- 10.0.3.1 ping statistics ---
9 packets transmitted, 4 packets received, 55% packet loss round-trip
min/avg/max = 3211900.8/5222806.6/6388671.9 ms
```

IPoAC: (more) Modern implementations

Pigeon-powered Internet takes flight

One of the Internet's
to life: transmitting n

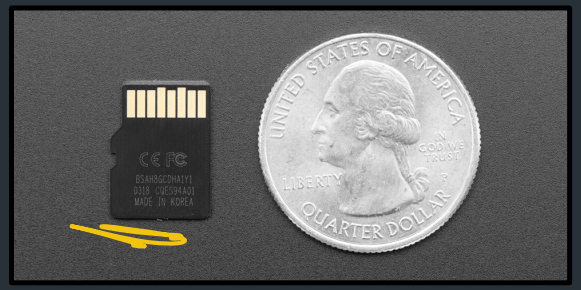
Stephen Shankland
Jan. 2, 2002 4:43 p.m. PT

BUSINESS

Pigeon carries data bundles faster than Telkom

Staff Reporter 10 Sep 2009

Today: microSD card: ~250mg, 1TB



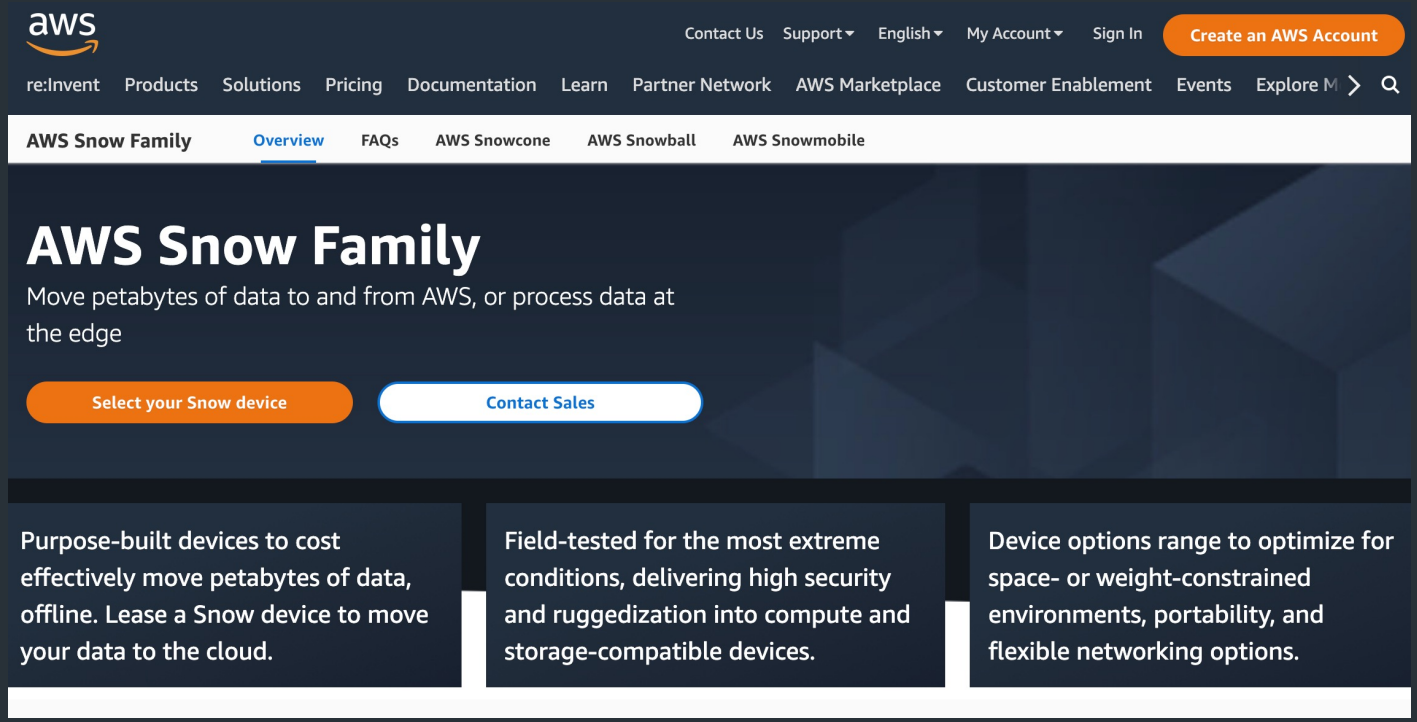
+

= ???

But actually

What happens if you have a LOT of data to move into the cloud?

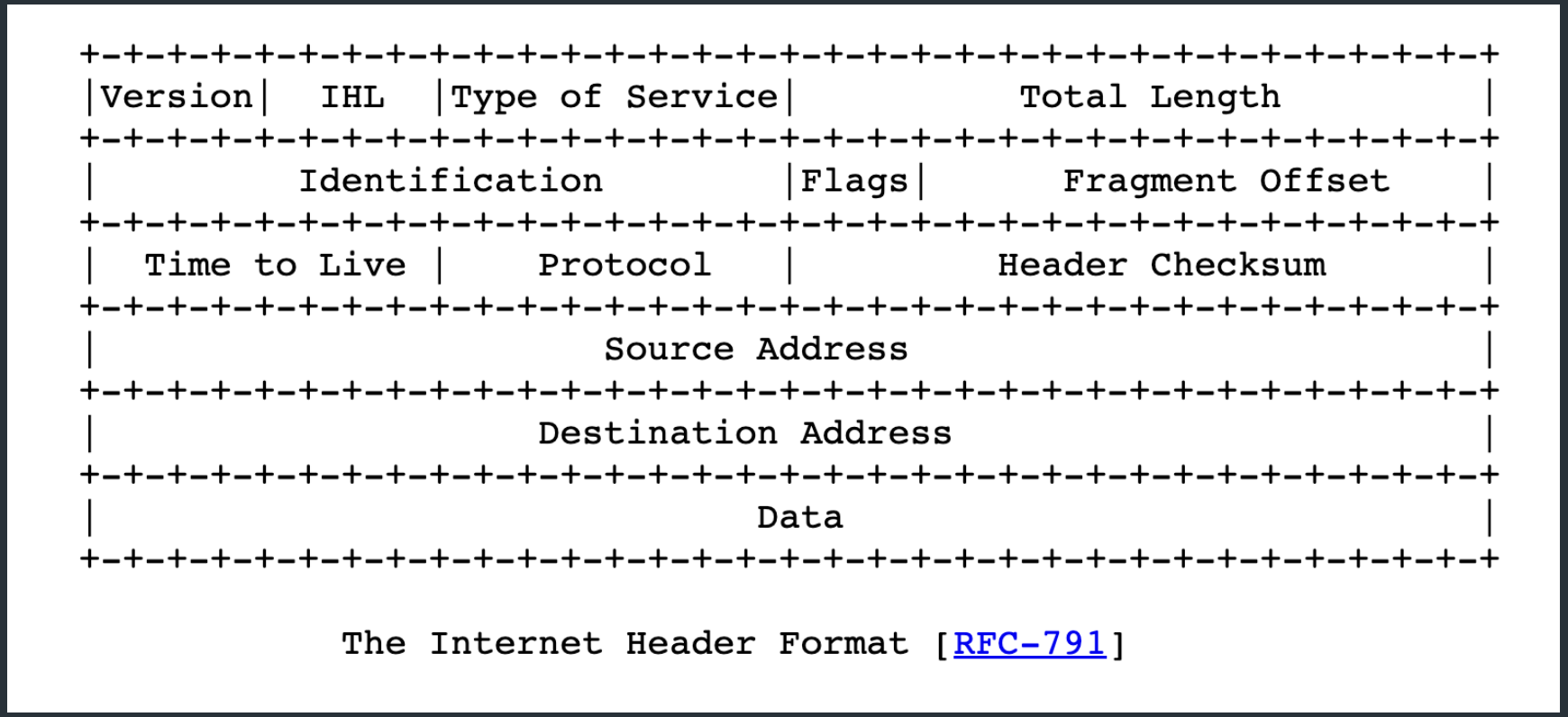
Example: AWS



Feature comparison matrix

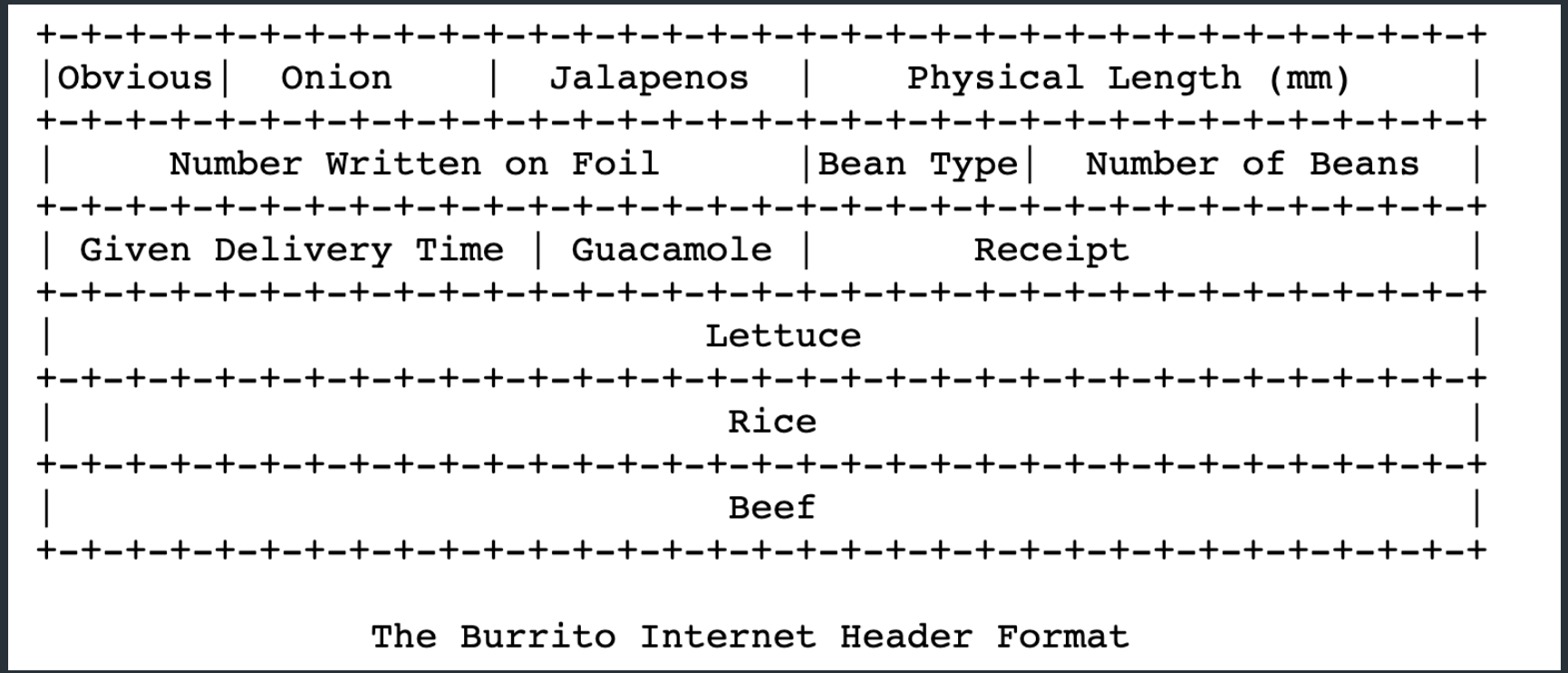
	AWS SNOWCONE	AWS SNOWBALL EDGE STORAGE OPTIMIZED	AWS SNOWBALL EDGE COMPUTE OPTIMIZED	AWS SNOWMOBILE
Usable HDD Storage	8 TB	80 TB	N/A	100 PB
Usable SSD Storage	14 TB	1 TB	28 TB	No
Usable vCPUs	4 vCPUs	40 vCPUs	104 vCPUs	N/A
Usable Memory	4 GB	80 GB	416 GB	N/A
Device Size	9in x 6in x 3in 227 mm x 148.6 mm x 82.65 mm	548 mm x 320 mm x 501 mm	548 mm x 320 mm x 501 mm	45 ft. shipping container
Device Weight	4.5 lbs. (2.1 kg)	49.7 lbs. (22.3 kg)	49.7 lbs. (22.3 kg)	N/A
Storage Clustering	No	Yes, 5-10 nodes	Yes, 5-10 nodes	N/A
256-bit Encryption	Yes	Yes	Yes	Yes
HIPAA Compliant	No	Yes, eligible	Yes, eligible	Yes, eligible

RFC791: IPv4 Header



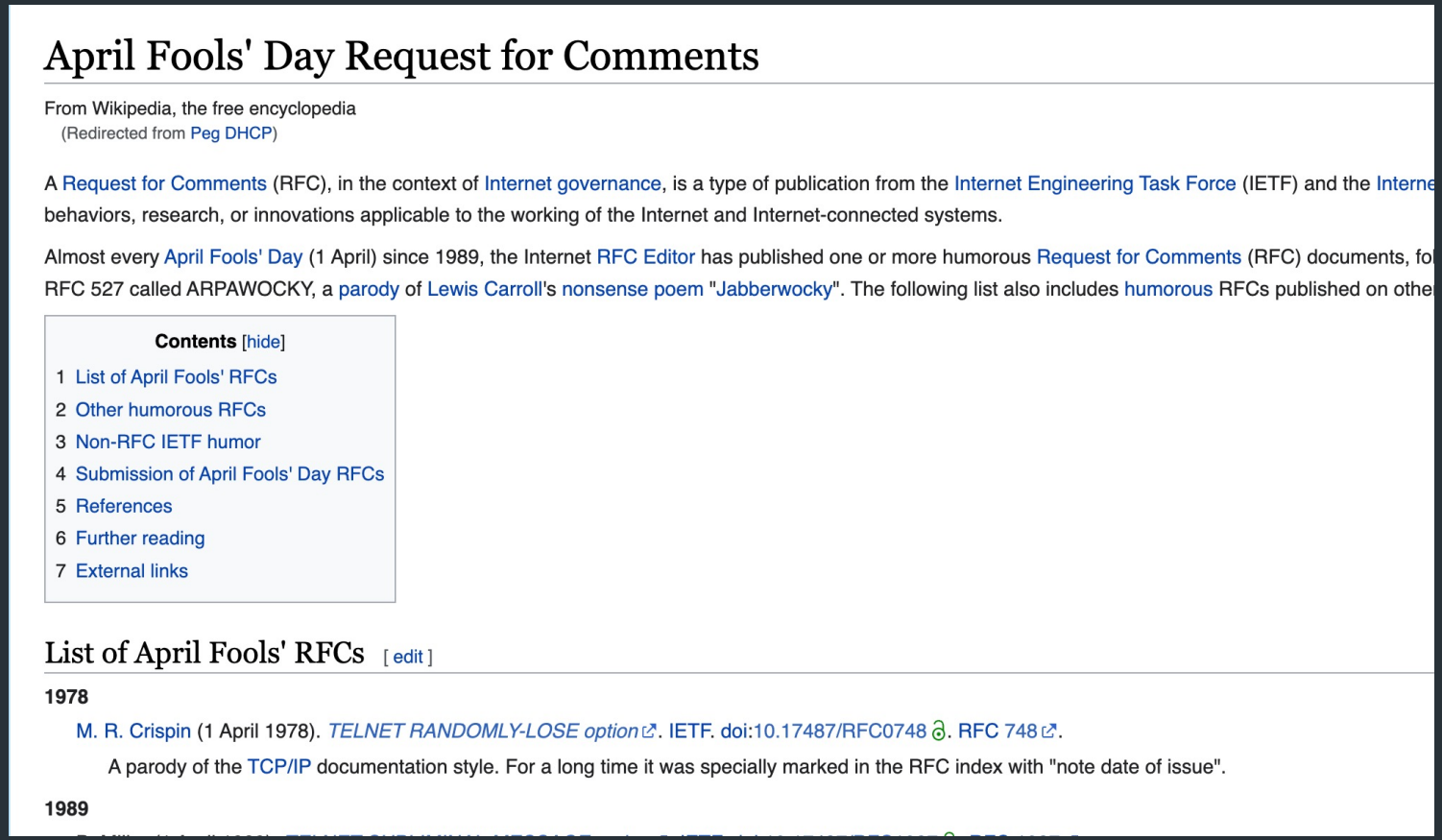
The Internet Header Format [RFC-791]

IP over Burrito Carriers



The Burrito Internet Header Format

April Fool's Day RFCs



April Fools' Day Request for Comments

From Wikipedia, the free encyclopedia
(Redirected from Peg DHCP)

A Request for Comments (RFC), in the context of Internet governance, is a type of publication from the Internet Engineering Task Force (IETF) and the Interne
behaviors, research, or innovations applicable to the working of the Internet and Internet-connected systems.

Almost every April Fools' Day (1 April) since 1989, the Internet RFC Editor has published one or more humorous Request for Comments (RFC) documents, fo
RFC 527 called ARPAWOCKY, a parody of Lewis Carroll's nonsense poem "Jabberwocky". The following list also includes humorous RFCs published on othe

Contents [hide]

1 List of April Fools' RFCs
2 Other humorous RFCs
3 Non-RFC IETF humor
4 Submission of April Fools' Day RFCs
5 References
6 Further reading
7 External links

List of April Fools' RFCs [edit]

1978

M. R. Crispin (1 April 1978). *TELNET RANDOMLY-LOSE option*. IETF. doi:10.17487/RFC0748. RFC 748.

A parody of the TCP/IP documentation style. For a long time it was specially marked in the RFC index with "note date of issue".

1989

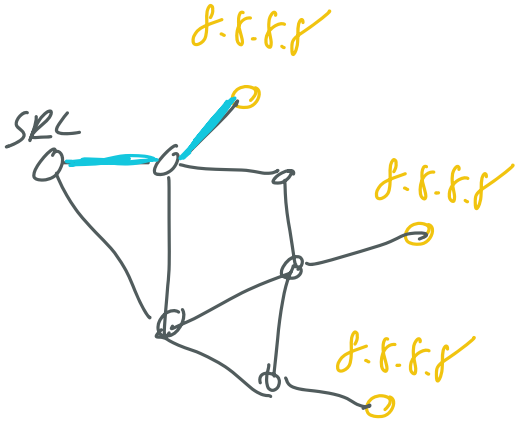
https://en.wikipedia.org/wiki/April_Fools%27_Day_Request_for_Comments Enjoy!

Warmup: About those root resolvers....

Anycast: advertise the same IP from multiple places around the globe

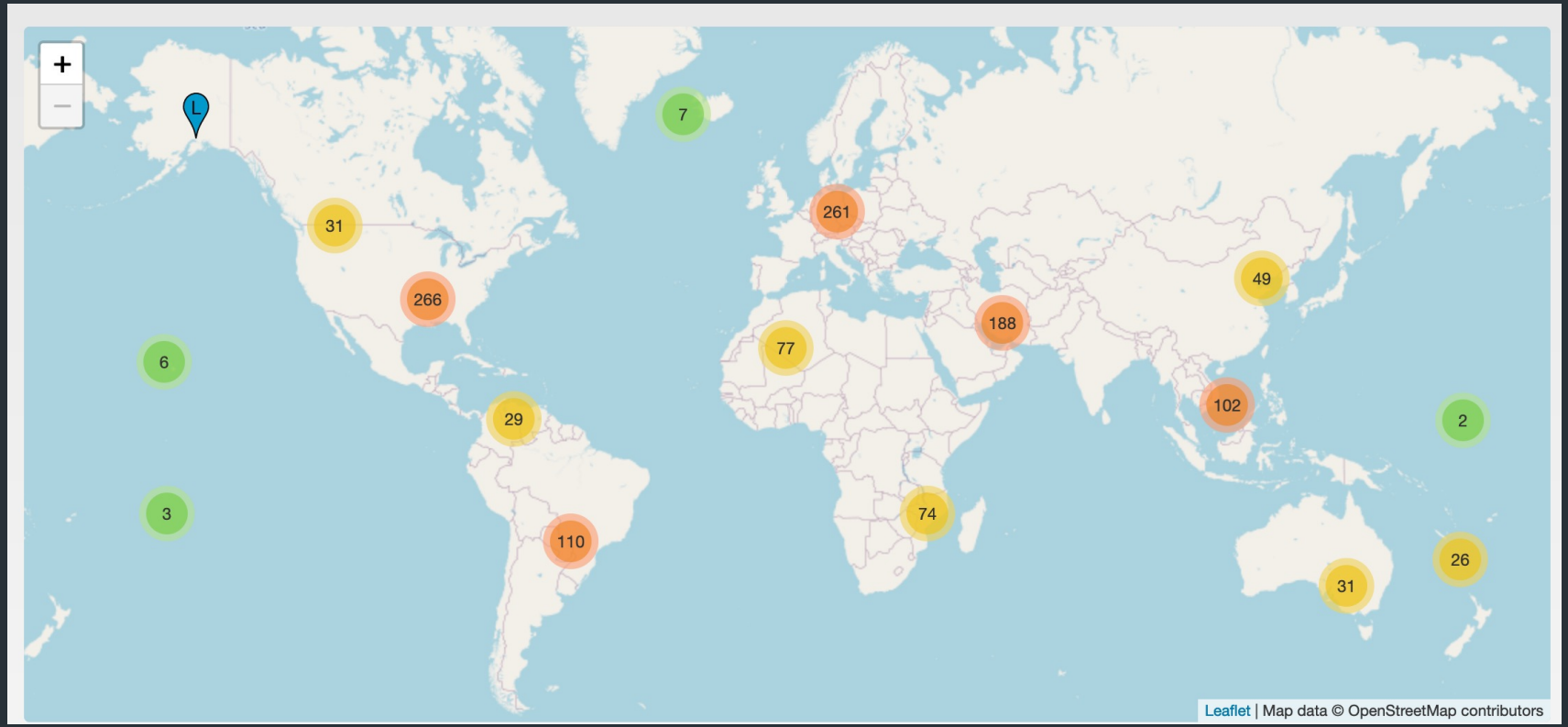
=> Multiple devices have the same IP!!

=> All routes are valid, so routers that receive the advertisments will install the closest route => reach the closest server

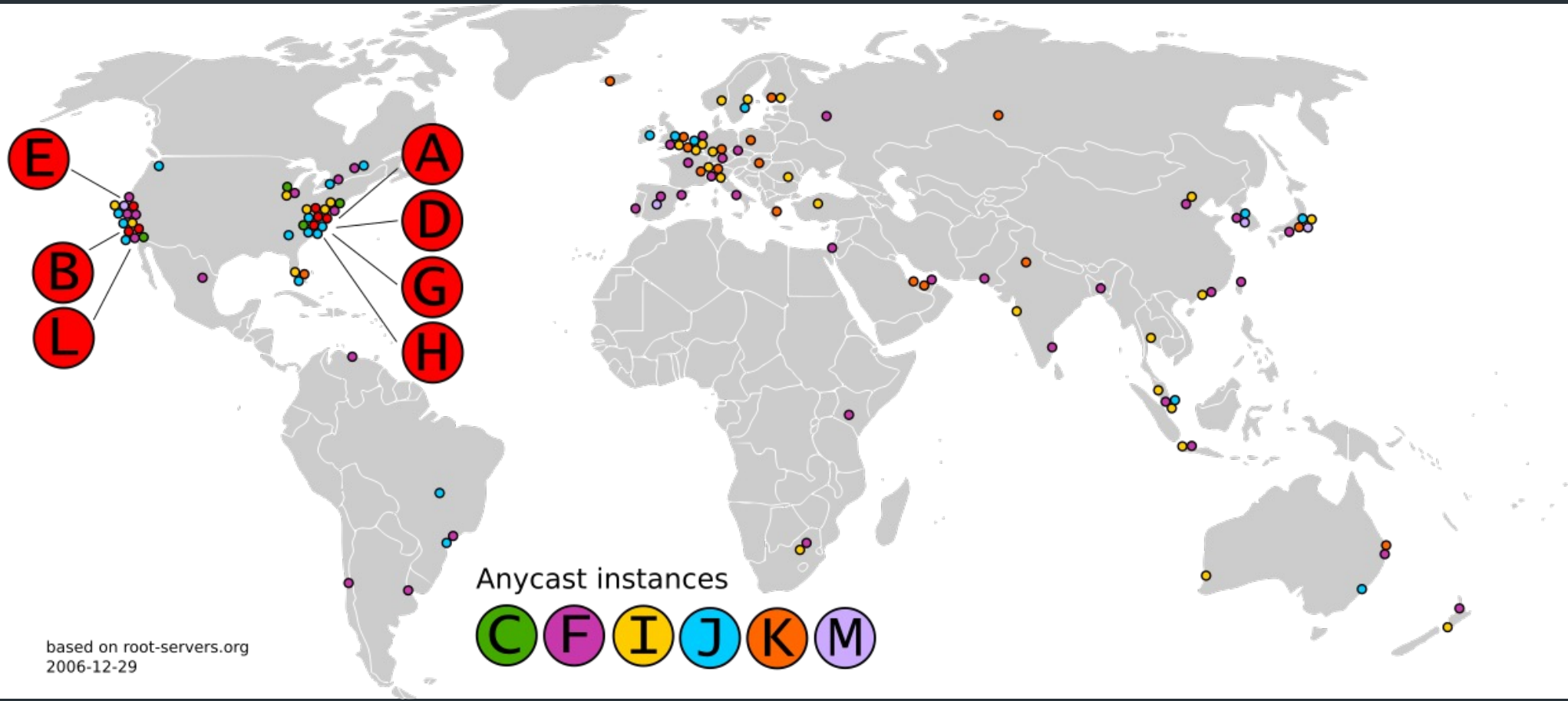


- Used to make certain IPs highly available (root nameservers, public DNS)

DNS Root Servers: Today

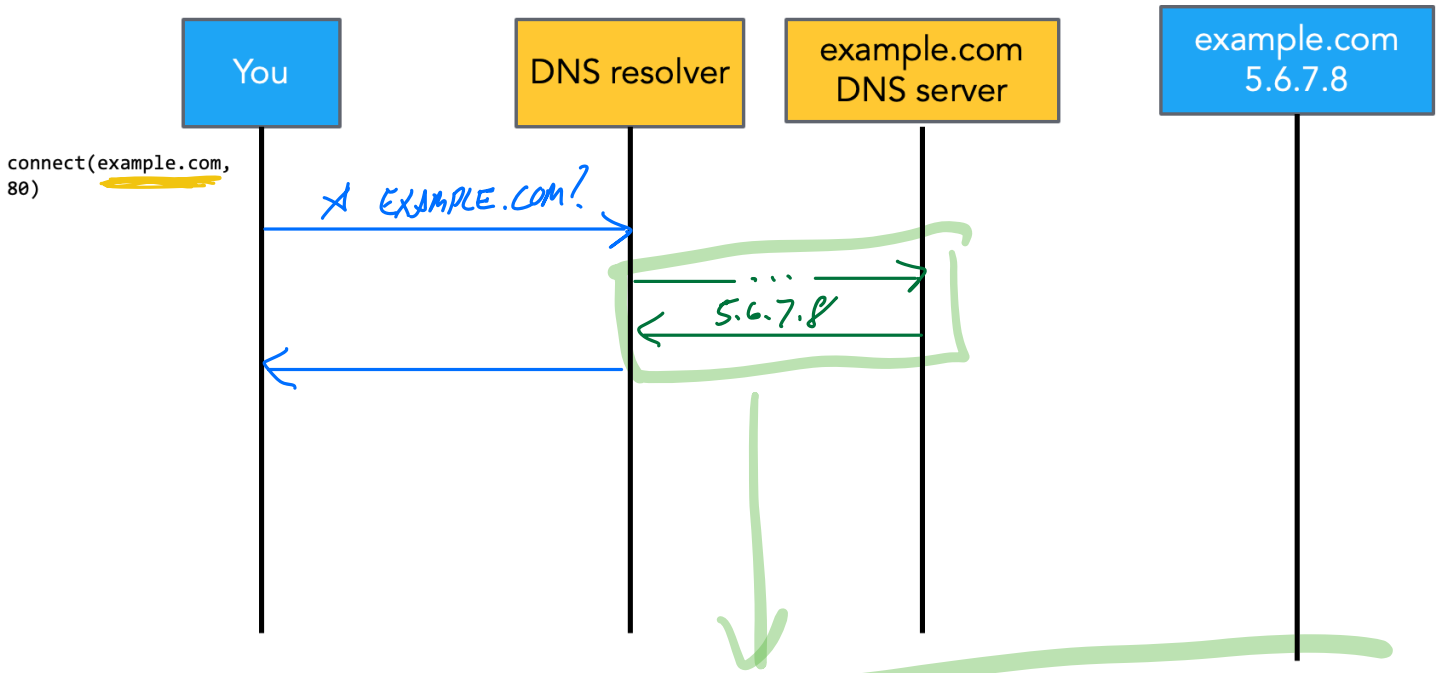


From: www.root-servers.org

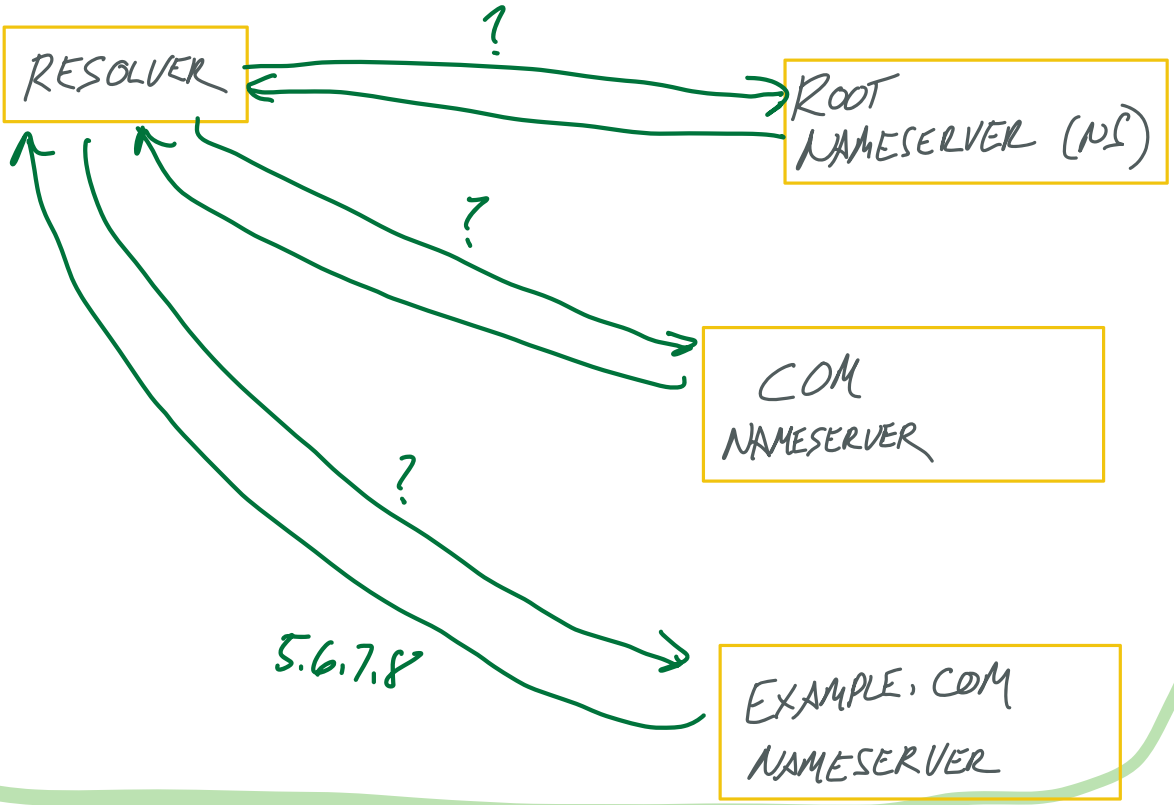

E
A
B
D
L
G
H
Anycast instances
C
F
I
J
K
M
based on root-servers.org
2006-12-29

The story so far

POV: You want to connect to some website



The query process as we've seen it so far (no caching): the resolver makes queries starting with the root nameserver. (See last lecture for full picture). This is called an *iterative* query.



Two "forms" of DNS lookups

- Iterative

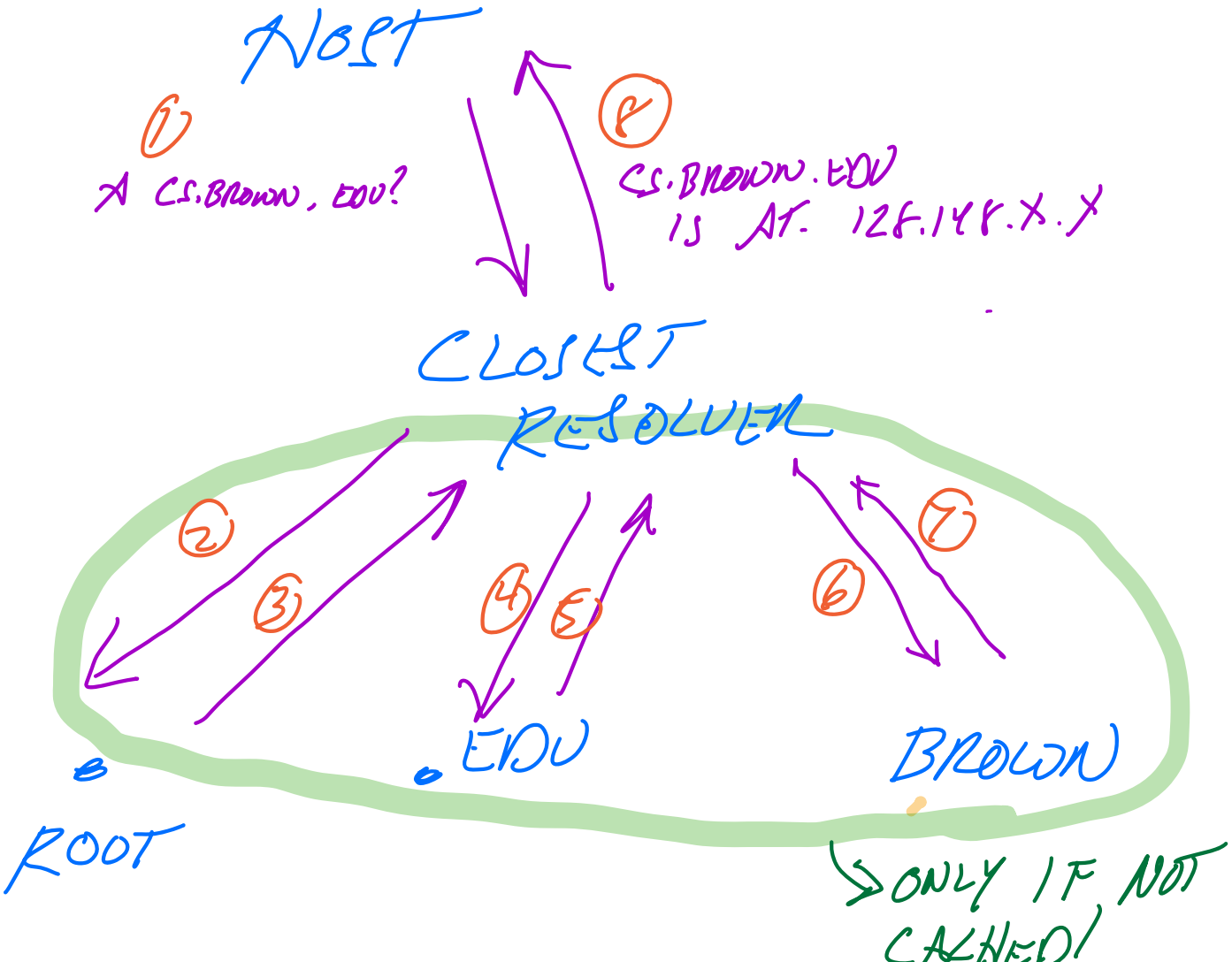
  resolver starts with root, queries each successive level of domain

- Recursive

  query another resolver, which does iterative query process for you, then returns result to you

How a recursive query works

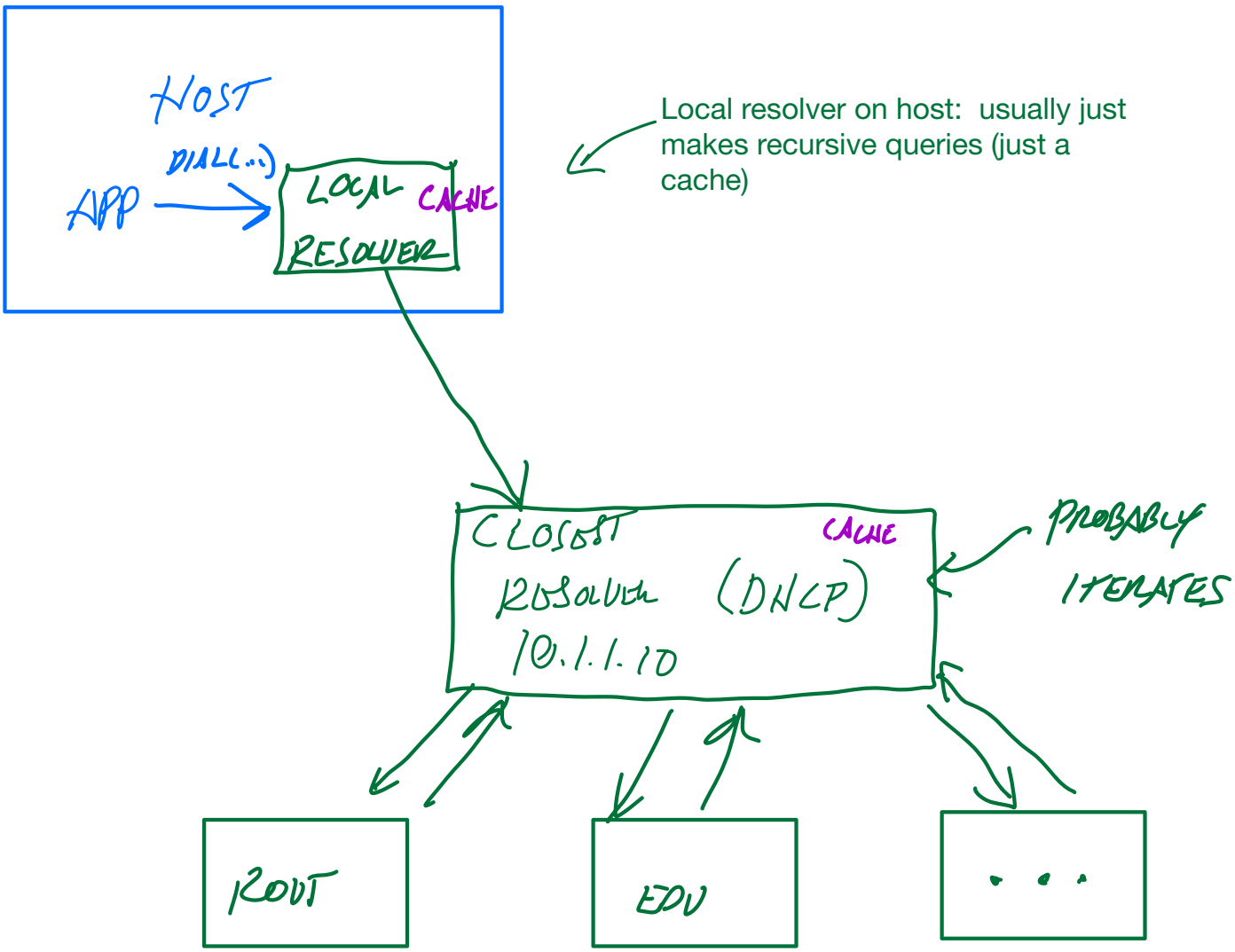
How a recursive query works (more common)

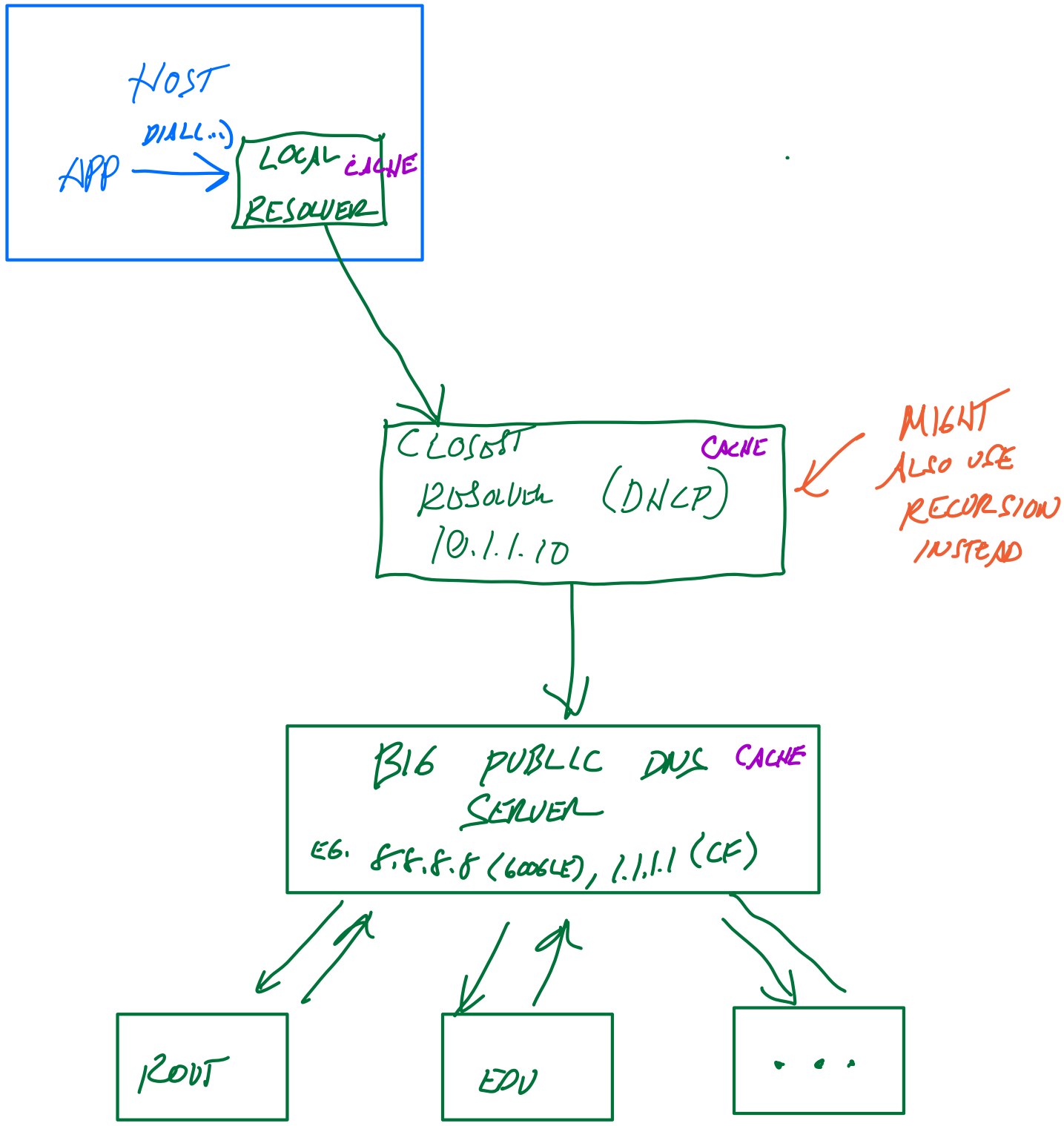


More commonly, hosts perform recursive queries to larger DNS servers, which do the typical iteration process (from the previous page) on the client's behalf.

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
DIAL(..)
APP
LOCAL
CACHE
RESOLVER
Local resolver on host: usually just makes recursive queries (just a cache)
CLOSEST
CACHE
RESOLVER
(DHCP)
10.1.1.10
PROBABLY ITERATES
ROOT
EDU
. . .

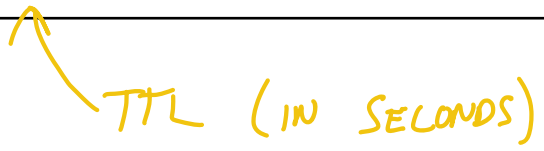

HOST
APP
DIALL..)
LOCAL
CACHE
RESOLVER
CLOSEST
RESOLVER
(DHCP)
10.1.1.10
CACHE
MIGHT
ALSO USE
RECURSION
INSTEAD
BIG PUBLIC DNS
SERVER
CACHE
EG. 8.8.8.8 (GOOGLE), 1.1.1.1 (CF)
ROOT
EDU
. . .

How it scales: caching

Resolvers cache responses to avoid extra iterative queries

Caching resolvers delete records when TTL expires
=> need to query again after that

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

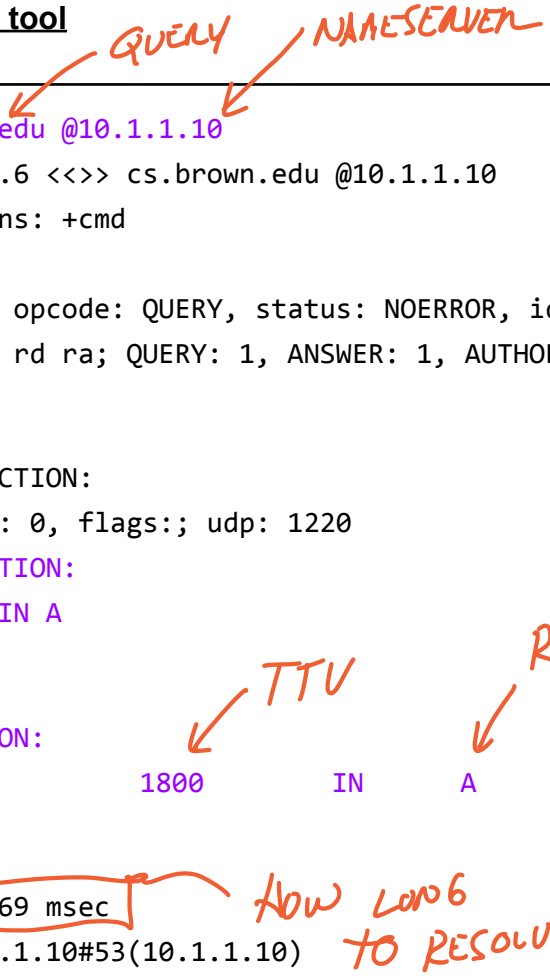


Q: do new queries refresh the TTL? No. The TTL just tells us how long we can wait before we must look up new data (until the server has updated it).

Here's how to make DNS queries yourself, like we're doing in class...

dig: DNS lookup tool

```
$ 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
```




```
Usage:  dig [+option -option] DOMAIN TYPE @nameserver
```

Useful options:

- +short: Don't print lots of stuff
- +rec: No recursion
- +nodnssec ([redacted]) DISABLE DNSSEC
- -x: Reverse DNS

DOMAIN: Domain to look up
TYPE: Record type
@nameserver: server to query

Reverse DNS

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

Special DNS zones that leverage hierarchy of IP addresses

Example: want to look up 128.148.32.19

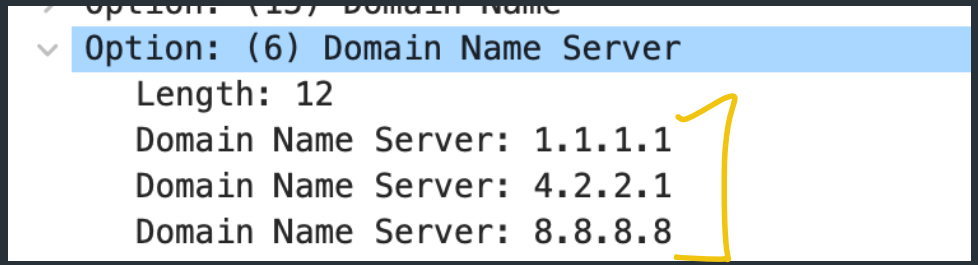
`dig -x 128.148.32.19`

```
; <<>> DiG 9.10.6 <<>> -x 128.148.32.19
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43289
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1220
;; QUESTION SECTION:
;19.32.148.128.in-addr.arpa.    IN    PTR

;; ANSWER SECTION:
19.32.148.128.in-addr.arpa. 1800 IN    PTR    rt.cs.brown.edu.

;; Query time: 87 msec
;; SERVER: 10.1.1.10#53(10.1.1.10)
;; WHEN: Thu Apr 02 08:56:05 EDT 2026
;; MSG SIZE  rcvd: 84
```

Who is your resolver?

You go to starbucks, how does your browser find www.google.com?

Typical, classic way: use DNS server from DHCP



Can you trust this DNS server?

DNS resolution: what can go wrong

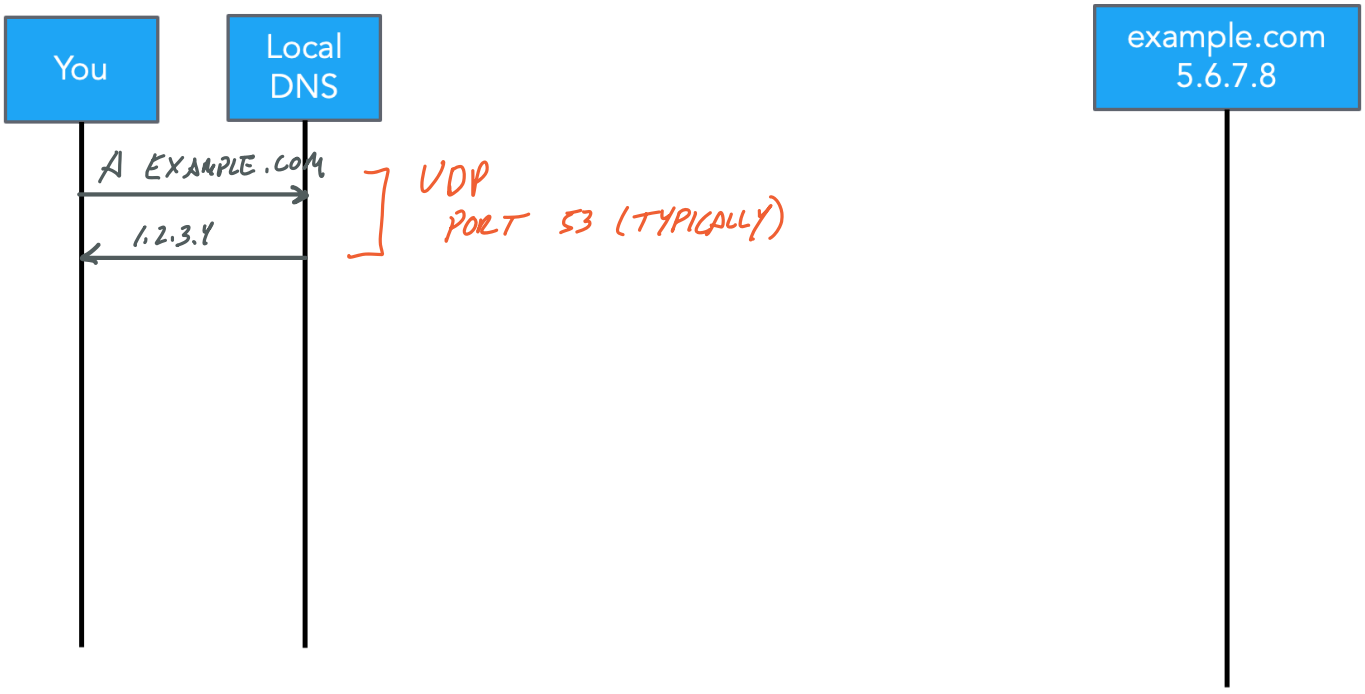
Context: the protocol

- TCP/UDP port 53
- Canonically uses UDP (but more on this later...)
- Usually: your host gets a resolver via DHCP when joining the network

```
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
```

These are public DNS servers, but DNS servers from DCHP are often local to the network you are on (eg. Brown internal DNS => 10.1.1.10)

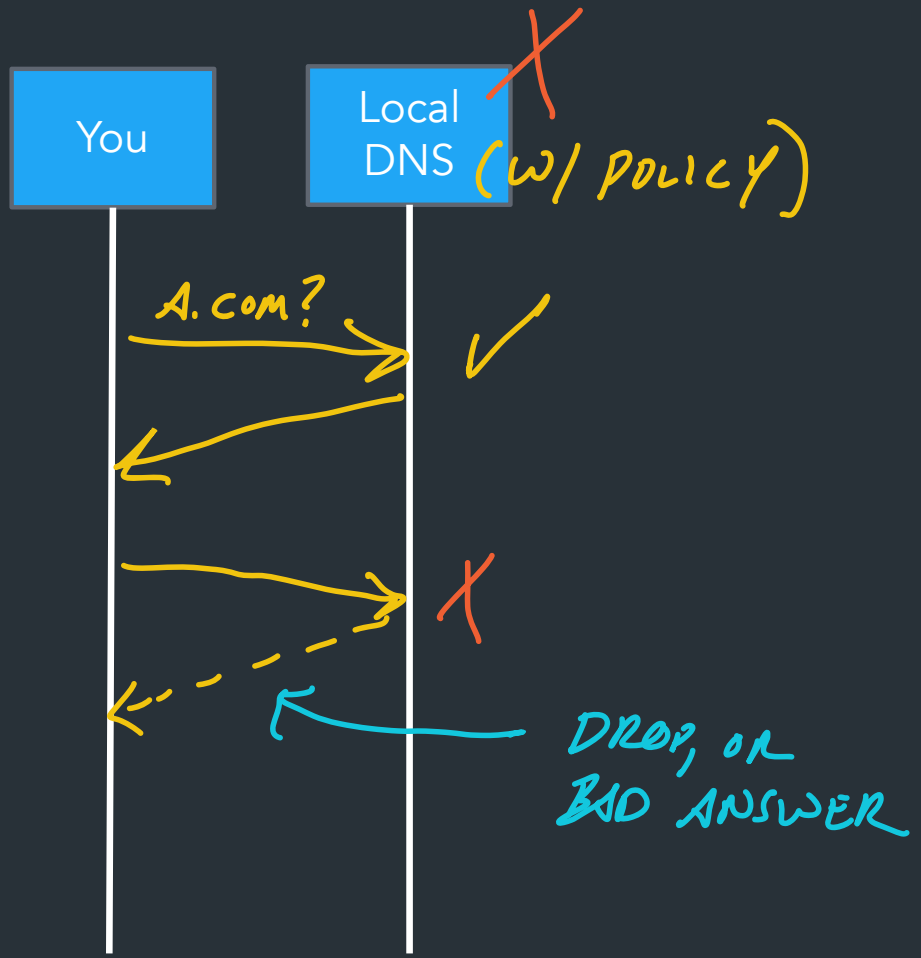
Can you trust this DNS server?? What could go wrong if you didn't?

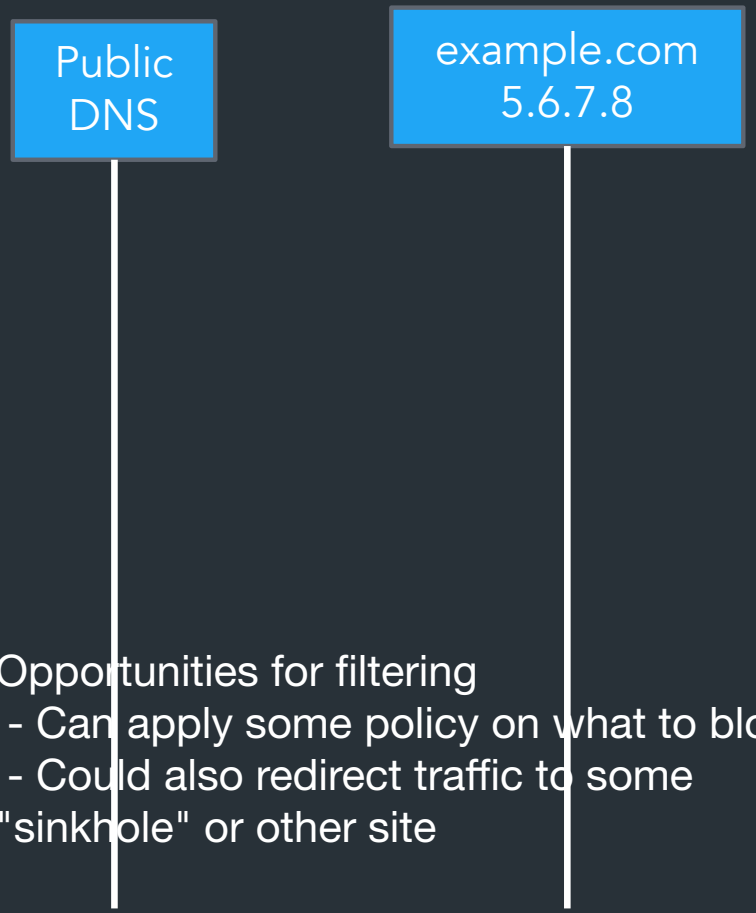


What are all the things a DNS server could do that are sus?

DNS server could lie => send bad result
Could block other queries
Could drop results
Could delay results and make users annoyed
Could redirect to something that does logging, traffic manipulation, etc.
DNS server itself can log data

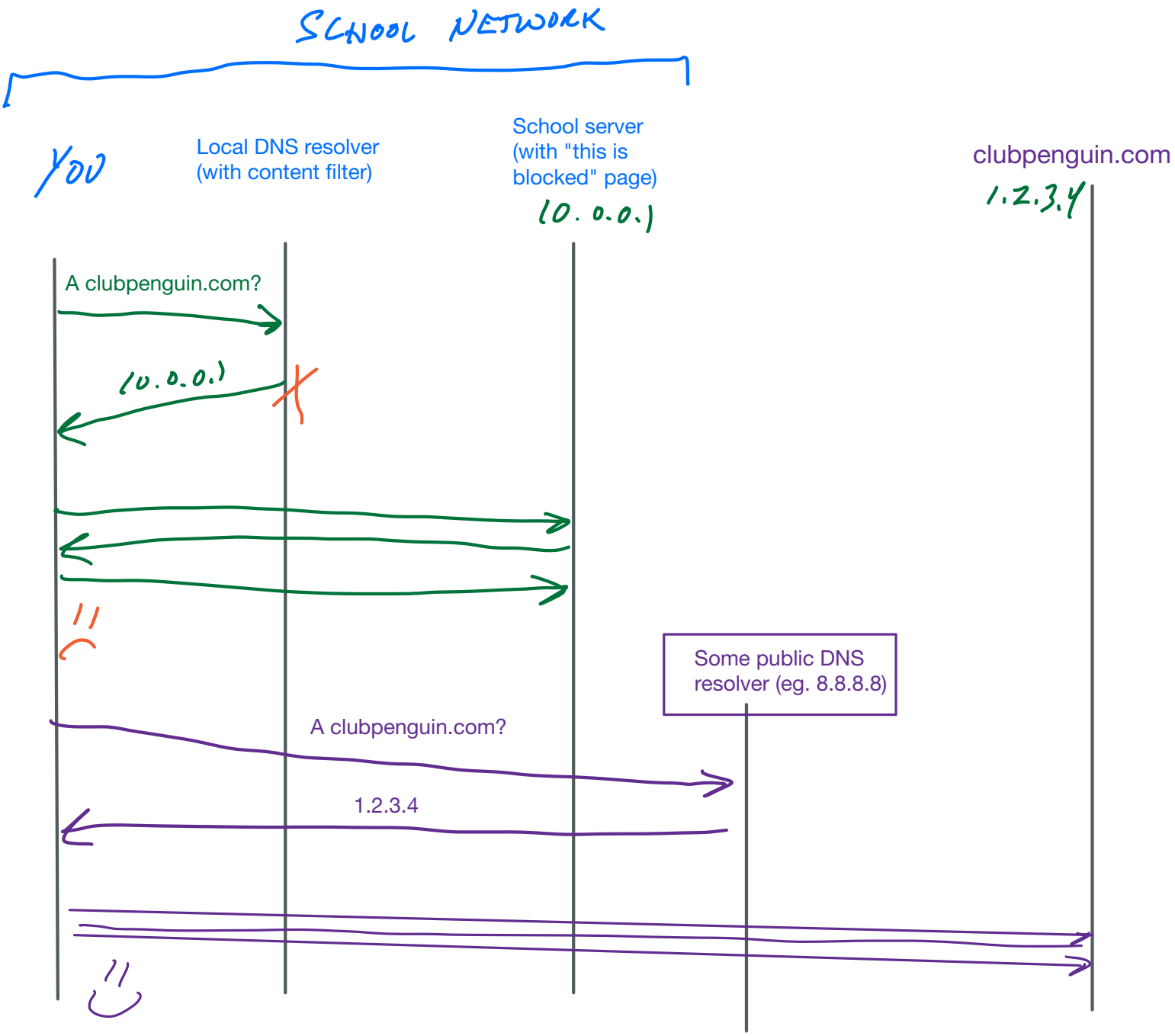
What if the resolver doesn't want you accessing a site?



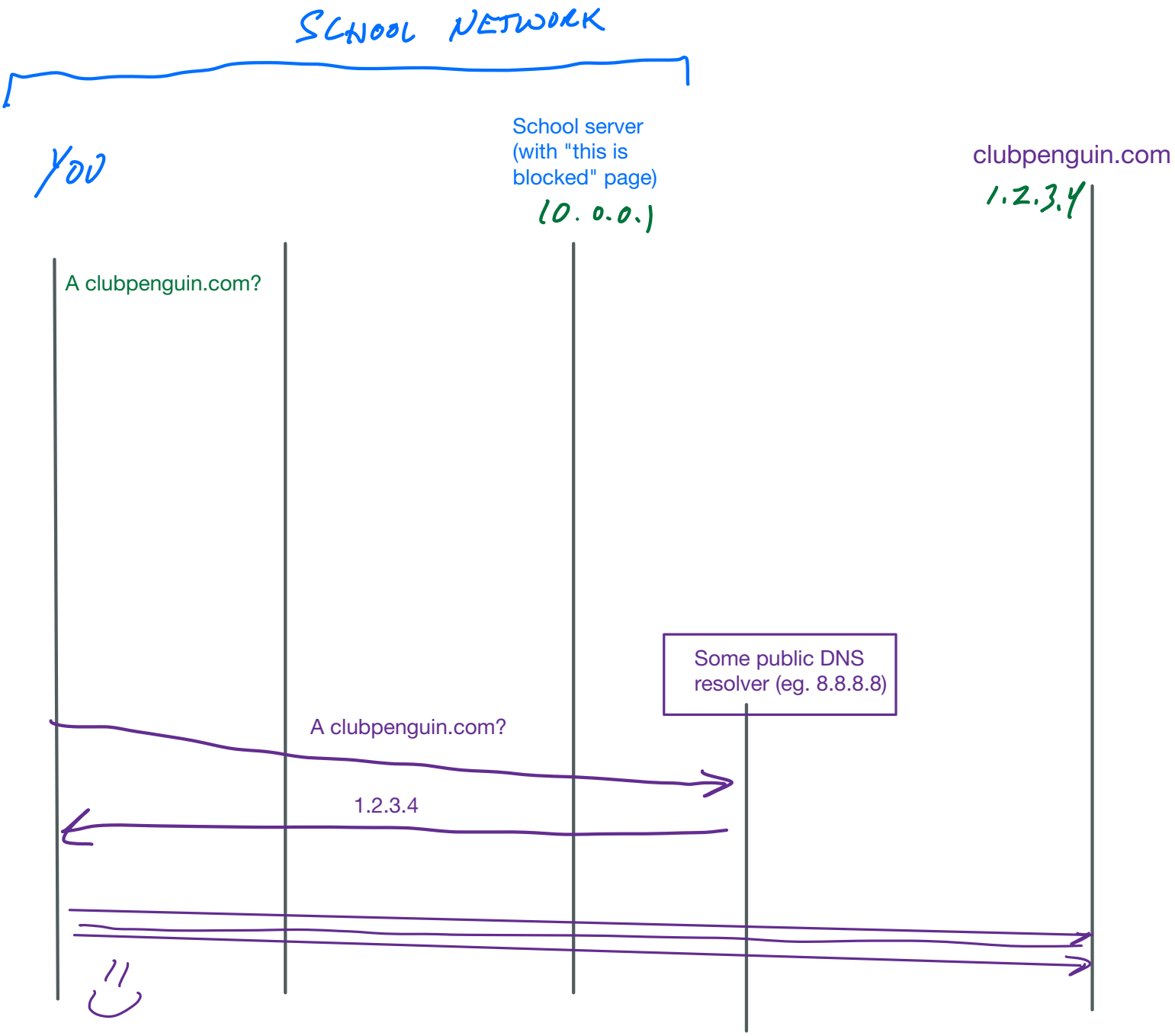


Opportunities for filtering
- Can apply some policy on what to block
- Could also redirect traffic to some "sinkhole" or other site

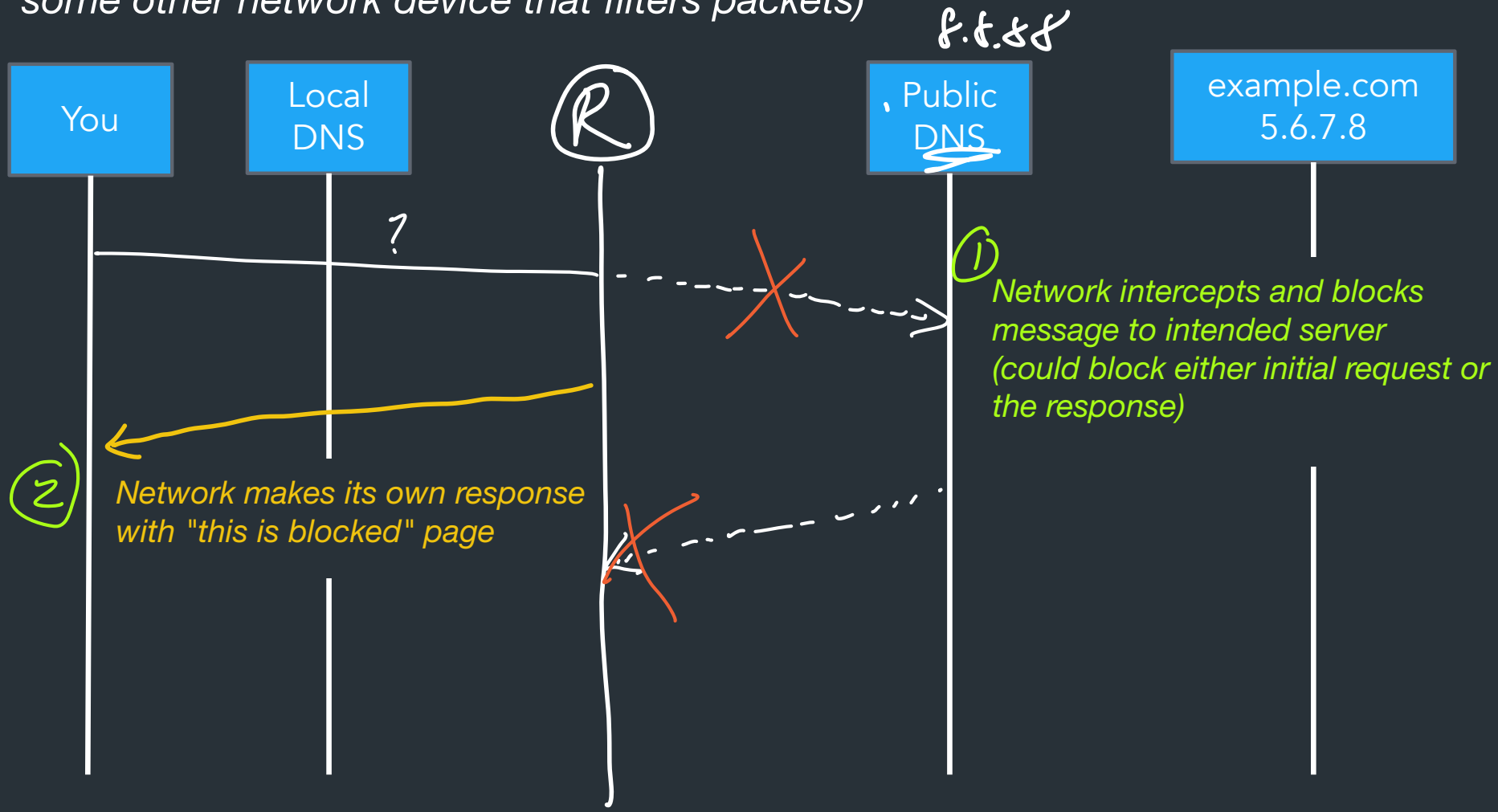
Example: DNS content filtering (which can be circumvented by changing DNS servers)



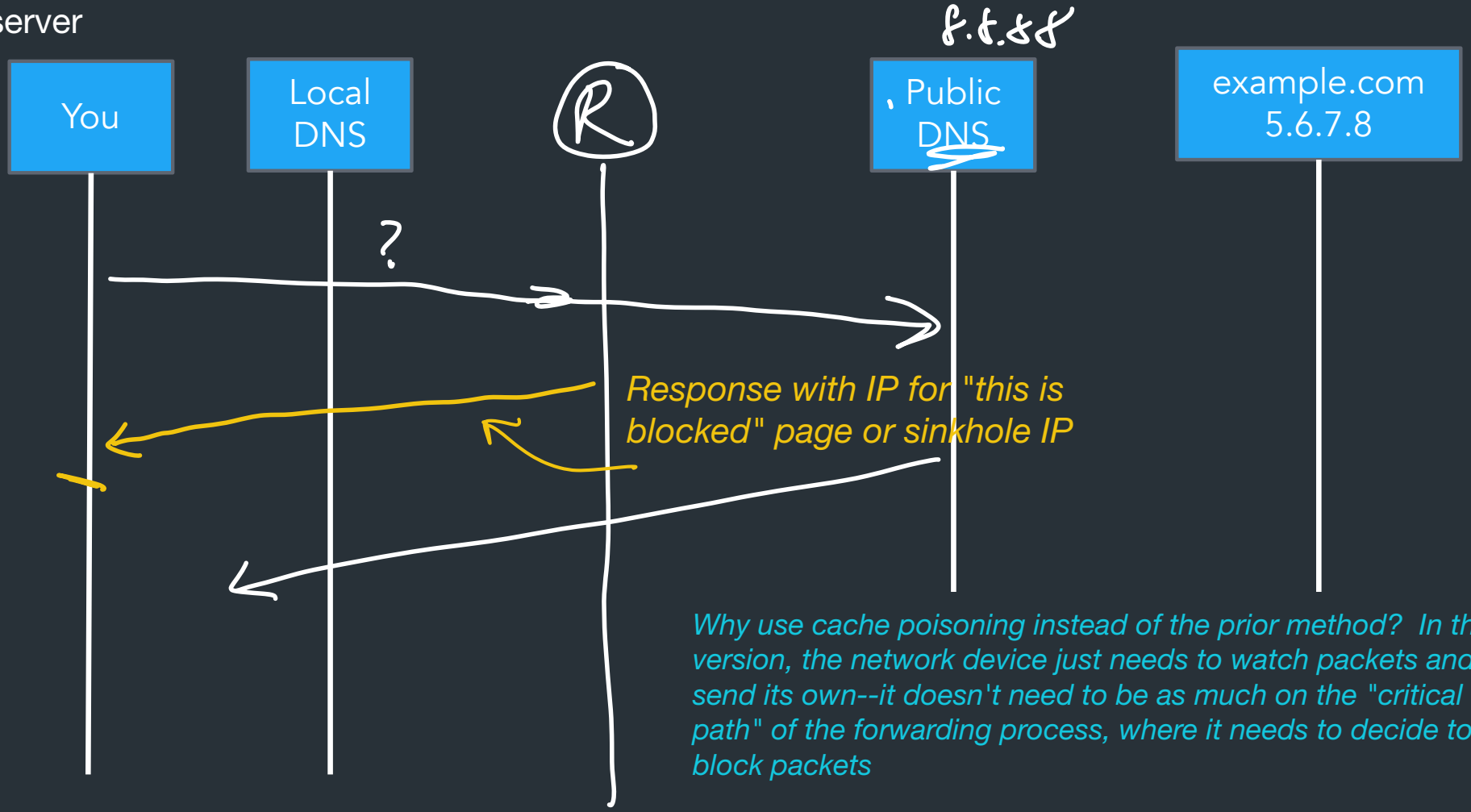
Example: DNS blocking, with help from a router or other network device



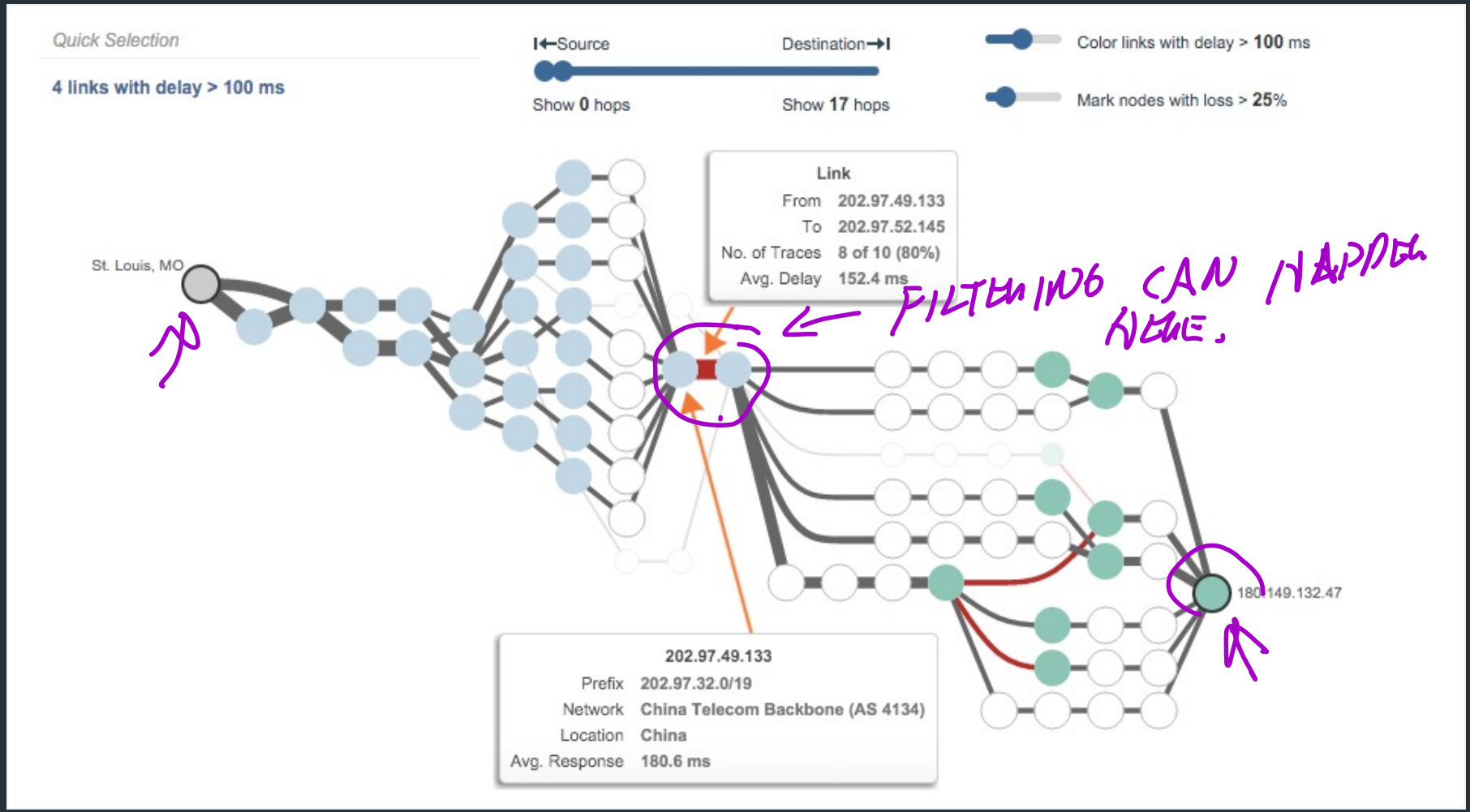
Example from class: DNS blocking, but with some help from a router *(or some other network device that filters packets)*



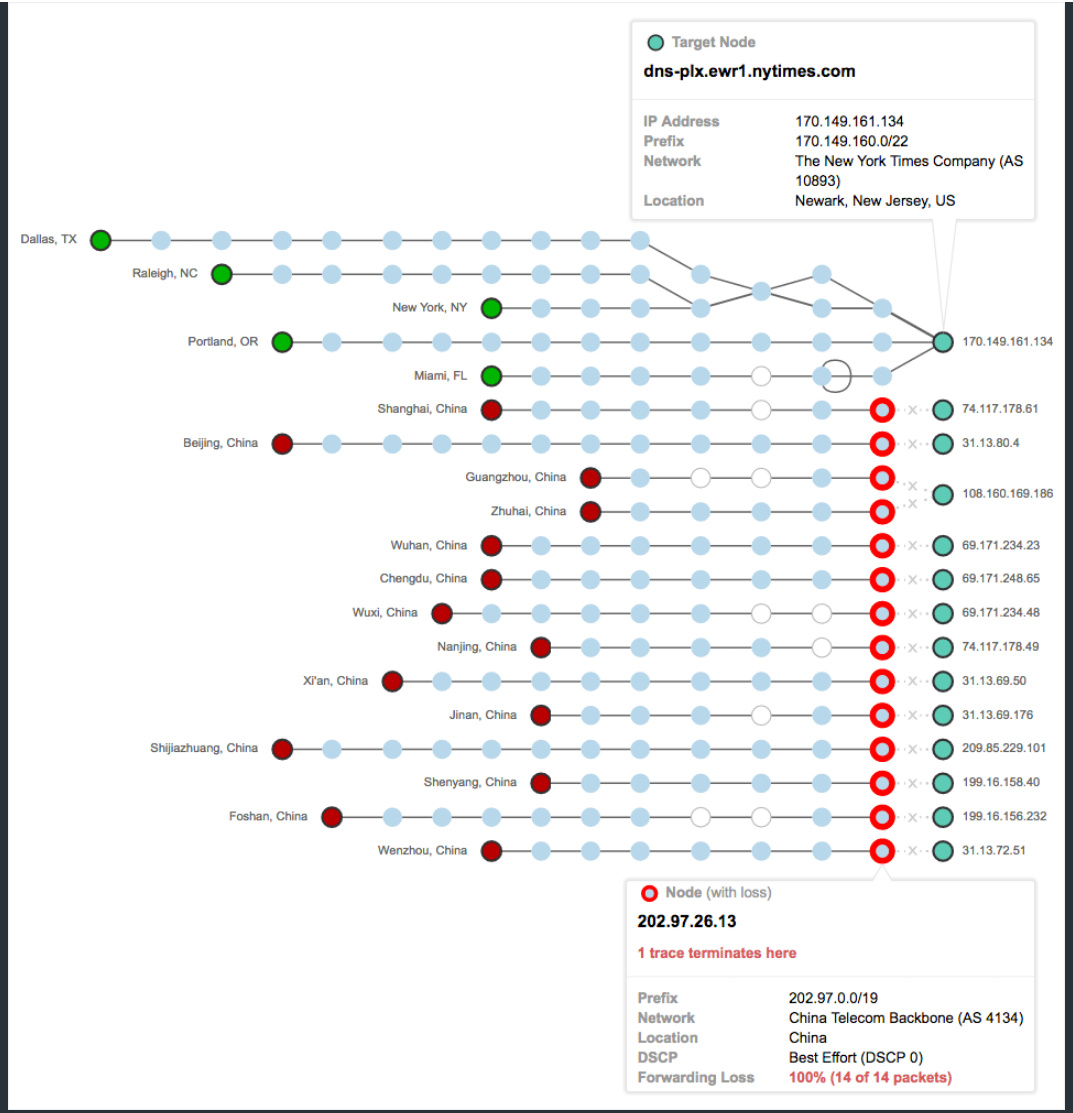
Another way: DNS cache poisoning: network responds with fake answer, can win race from legitimate server



Why use cache poisoning instead of the prior method? In this version, the network device just needs to watch packets and send its own--it doesn't need to be as much on the "critical path" of the forwarding process, where it needs to decide to block packets

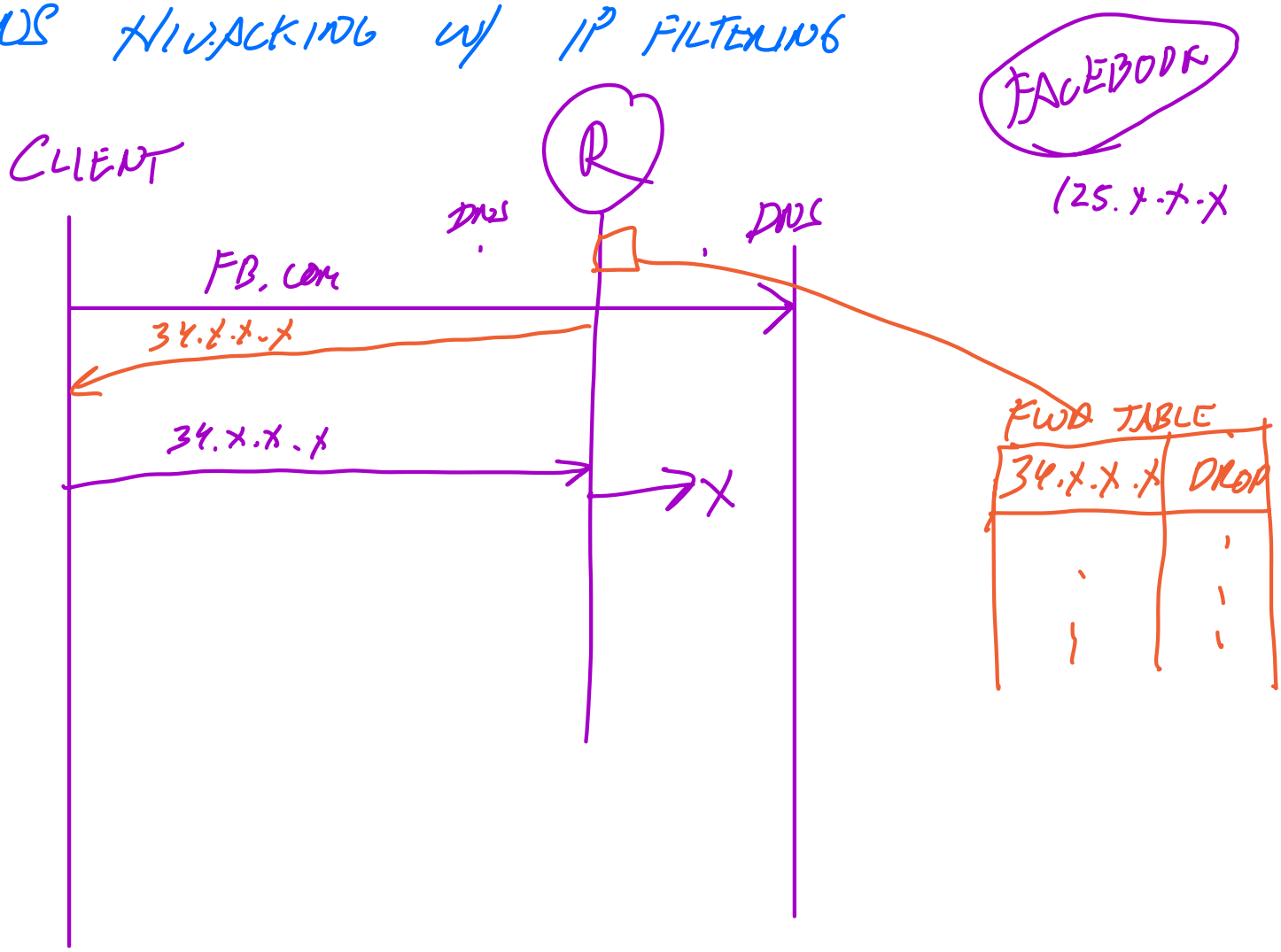


https://www.thousandeyes.com/blog/internet-censorship-around-the-world



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

DNS HIJACKING w/ IP FILTERING

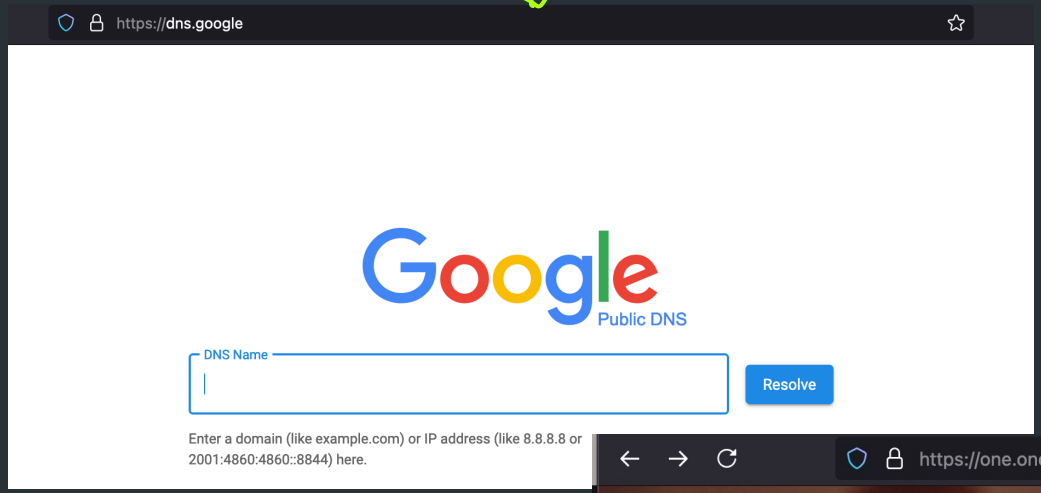


=> Malicious router (or DNS server) responds with IP it knows is blocked by its routers

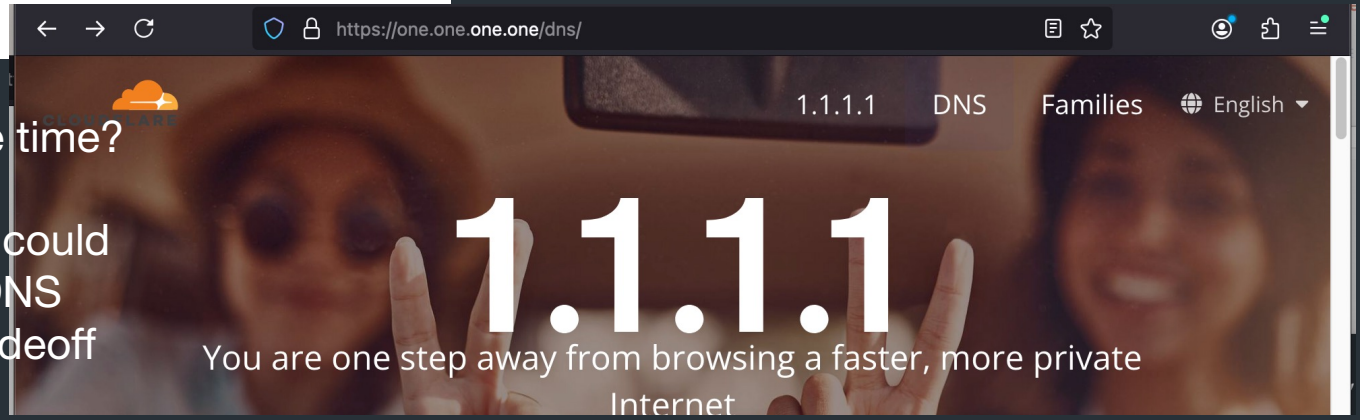
Public DNS: resolvers provided by cloud companies and ISPs

Highly-available public DNS servers
- Low latency, lots of caching
- Served by anycast

8.8.8.8 (GOOGLE)



CLOUDFLARE

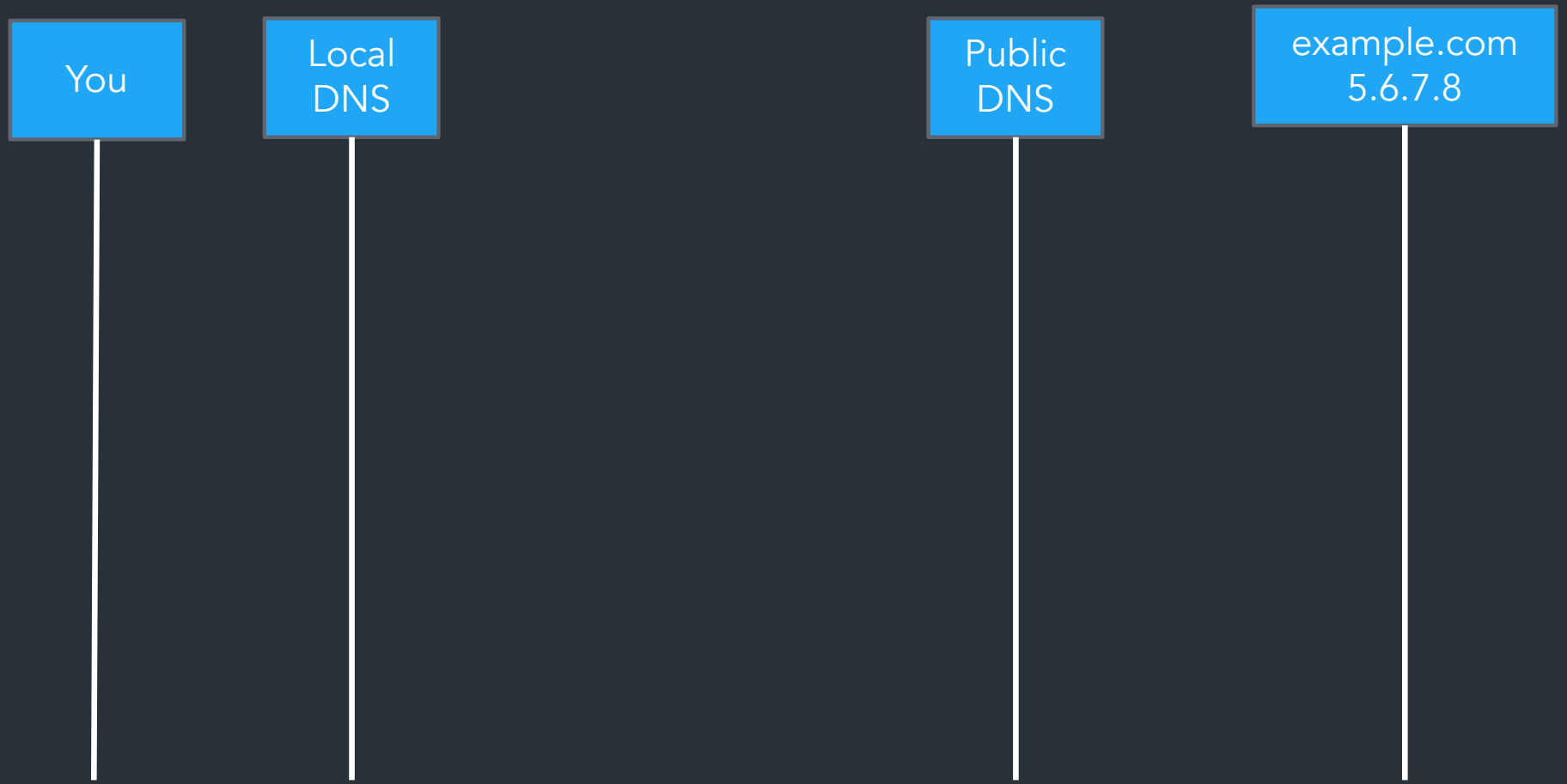


Why not use one of these all the time?
- Potential privacy risk
- Performance? Local resolver could have lower latency (but public DNS has much larger cache...) => tradeoff

Changing DNS servers in response to blocking of Twitter in Turkey (2014)

Writeup, with more links: https://www.thousandeyes.com/blog/internet-censorship-around-the-world

What else can a less-than-honest resolver do?



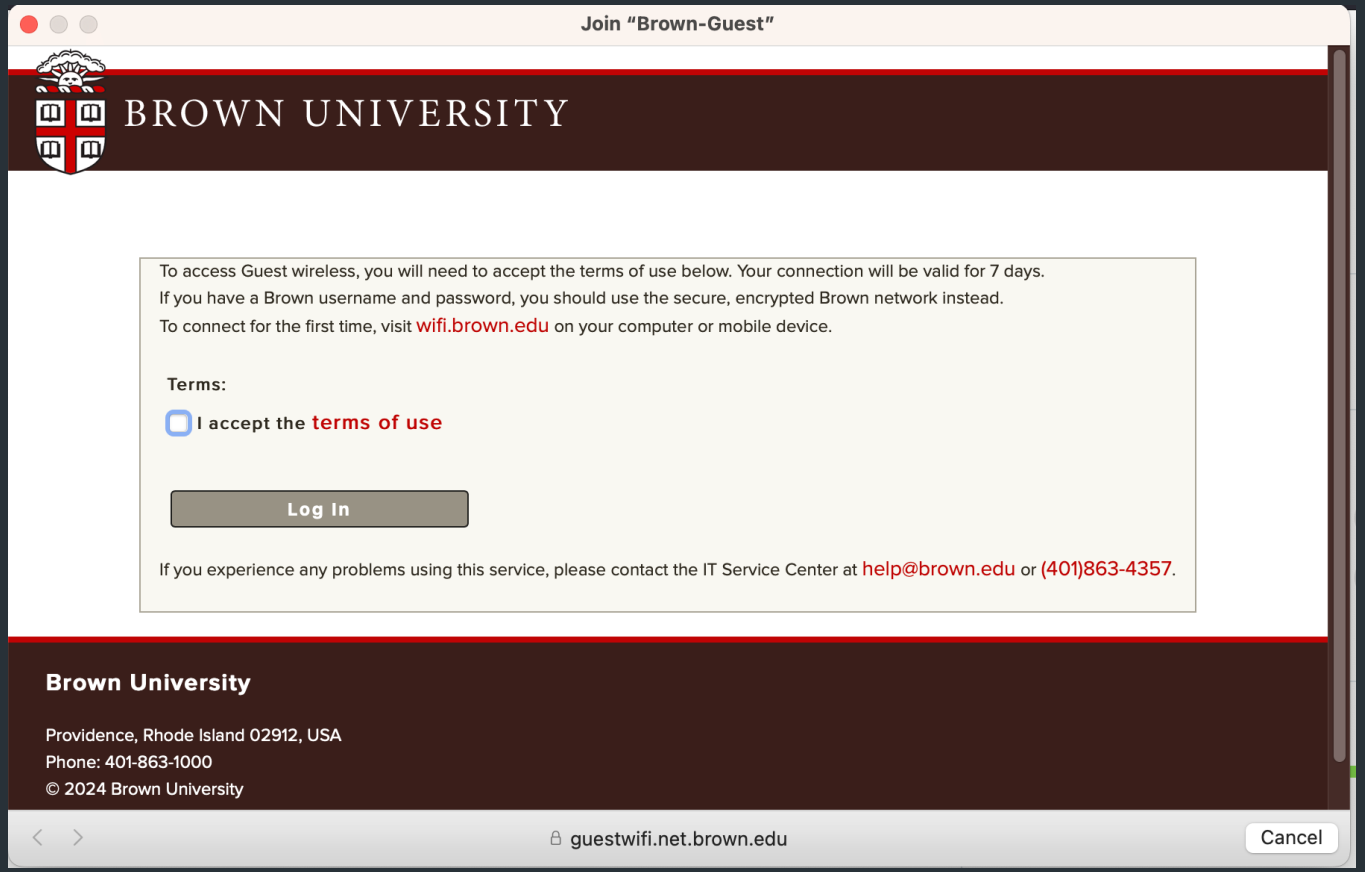
"Helpful" ISPs (OLD)

- 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?)

=> Less common nowadays: modern browsers will detect when DNS servers do this on startup => send bad queries to browser's preferred search engine

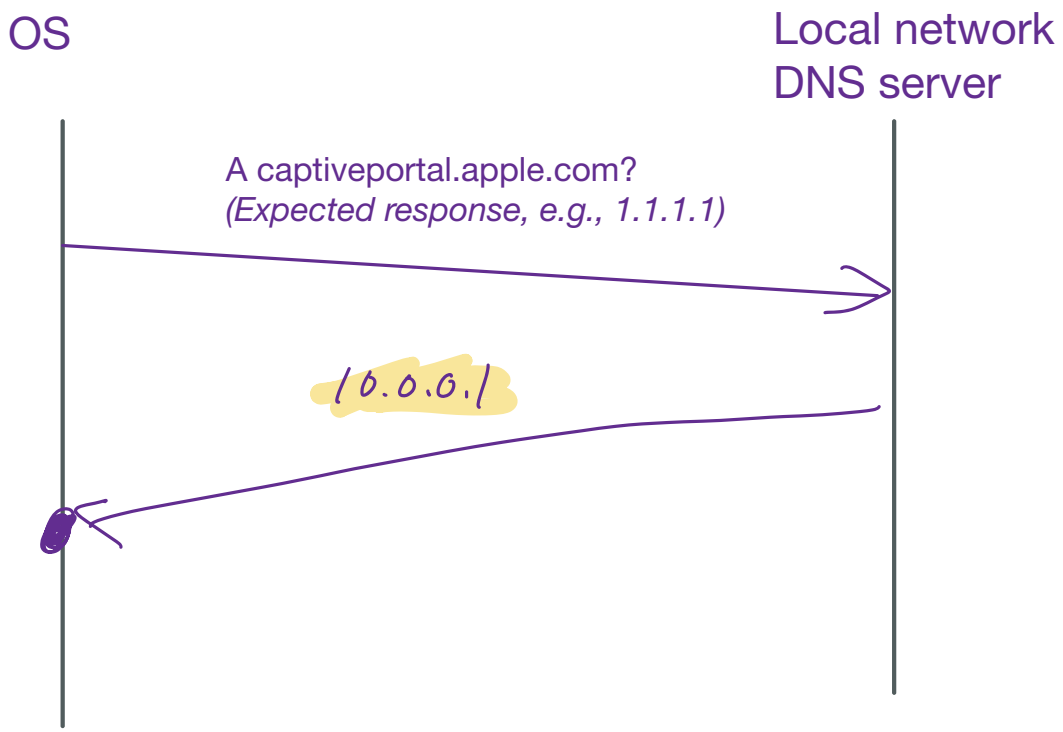
(not as easy anymore due to HTTPS, modern browsers having an interest in correct DNS results)

Captive portals Another "misuse" of DNS



Captive portal: local network ALWAYS responds with address of login page until you log in

=> Often breaks modern (secure) browsers. In last 5-10 years, OSes have figured out a strategy to make this better



OS queries a known domain => if answer doesn't match expected result, likely using a captive portal => display login page

What can be done?

Overall problem: DNS does not validate or verify responses by default

=> No way to know response actually came from authoritative server!

- DNSSEC: protocol to sign/verify hierarchy of DNS lookups
  - Results must be cryptographically signed, can be traced back to authoritative server and DNS root
  - Expensive to deploy, need to get hierarchy of signatures

- Tunneling DNS over a more secure protocol
  => DNS over HTTPS (DoH)

Home > Products > Make the Web Faster > Public DNS > Guides

Was this helpful?

DNS-over-HTTPS (DoH)

Google Public DNS provides two distinct DoH APIs at these endpoints:

- https://dns.google/dns-query – RFC 8484 (GET and POST)
- https://dns.google/resolve? – JSON API (GET)

Note: There is also a human-friendly web interface at https://dns.google/. This web app displays JSON results in a browser but does not implement an API; do not confuse its https://dns.google/query? URLs with the two API URLs.

FIREFOX

Firefox continues push to bring DNS over HTTPS by default for US users

FEBRUARY 25, 2020 SELENA DECKELMANN

Enable DNS over HTTPS using:

Default Protection

Firefox decides when to use secure DNS to protect your privacy.

- Use secure DNS in regions where it's available
- Use your default DNS resolver if there is a problem with the secure DNS provider
- Use a local provider, if possible Learn more
- Turn off when VPN, parental control, or enterprise policies are active
- Turn off when a network tells Firefox it shouldn't use secure DNS Learn more

Increased Protection

You control when to use secure DNS and choose your provider.

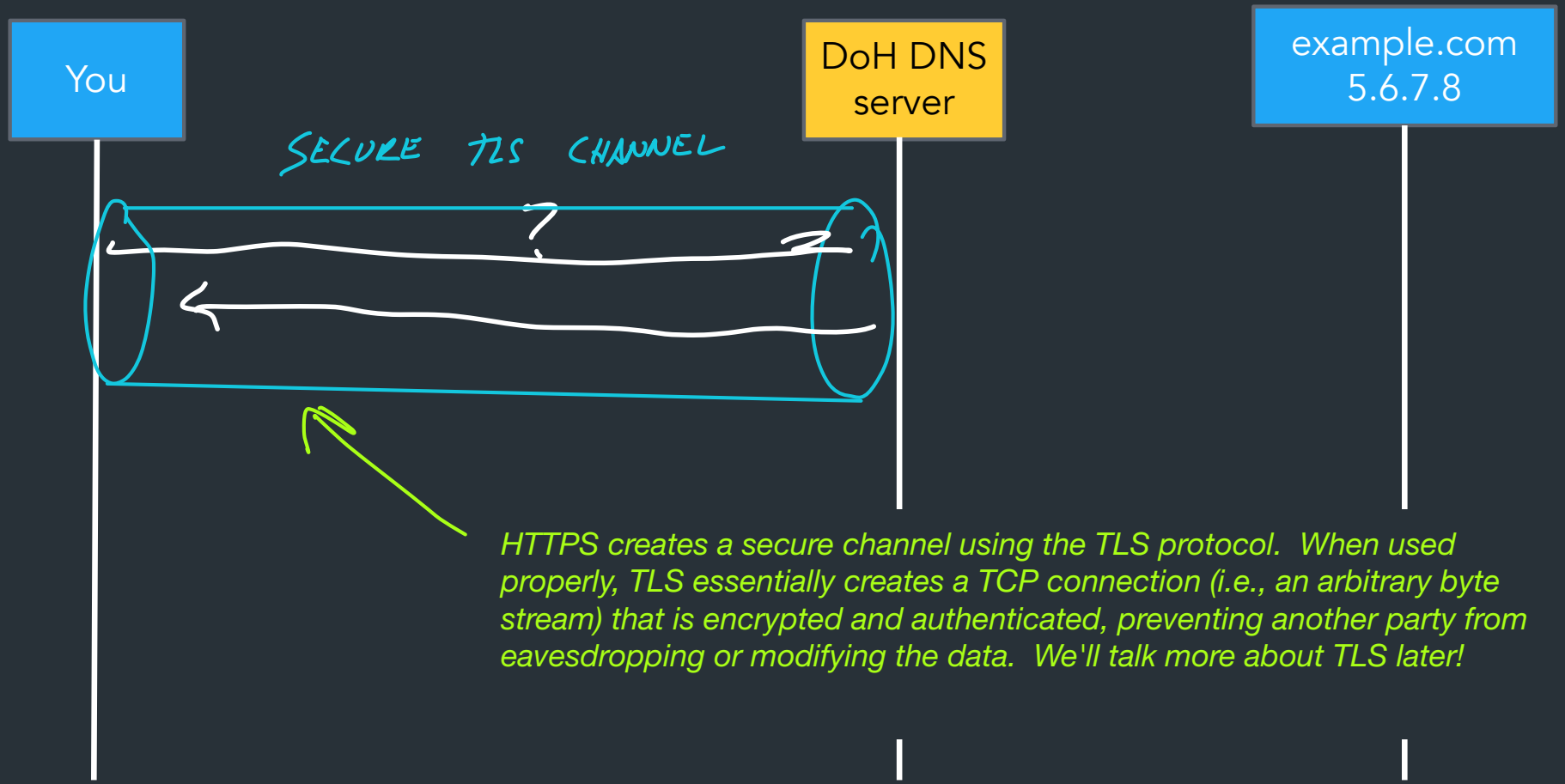
Max Protection

Firefox will always use secure DNS. You'll see a security risk warning before we use your system DNS.

Off

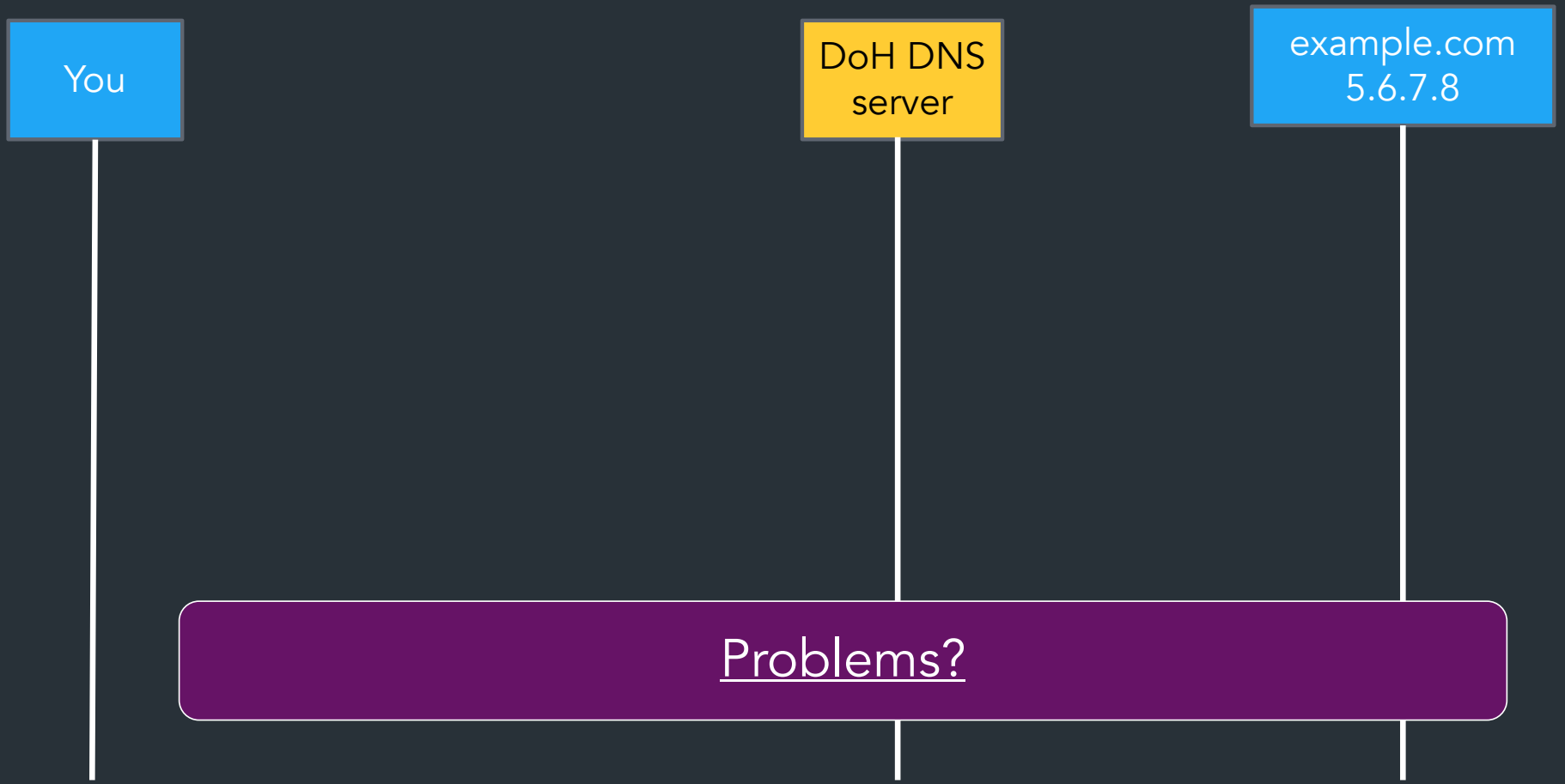
Use your default DNS resolver

DNS over HTTPs



HTTPS creates a secure channel using the TLS protocol. When used properly, TLS essentially creates a TCP connection (i.e., an arbitrary byte stream) that is encrypted and authenticated, preventing another party from eavesdropping or modifying the data. We'll talk more about TLS later!

DNS over HTTPs



Mozilla Policy Requirements for DNS over HTTPs Partners

This document describes the minimum set of policy requirements that a party must satisfy to be considered as a potential partner for Mozilla's Trusted Recursive Resolver (TRR) program. It specifically describes data collection and retention, transparency, and blocking policies and is in addition to any contractual, technical or operational requirements necessary to operate the resolver service.

Privacy Requirements

Mozilla's TRR is intended to provide better, minimum privacy guarantees to Firefox users than current, ad hoc provisioning of DNS services. As such, resolvers must strictly limit data collection and sharing from the resolver. More specifically:

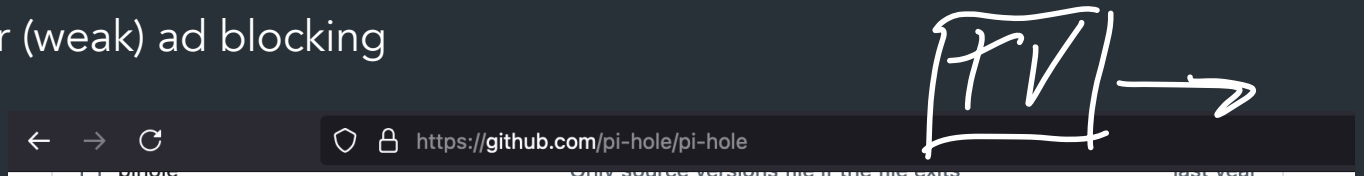
1. The resolver may retain user data (including identifiable data, data associated with user IP addresses, and any non-aggregate anonymized data) but should do so only for the purpose of operating the service and must not retain that data for longer than 24 hours.
    - Only aggregate data that does not identify individual users or requests may be retained beyond 24 hours.
2. The resolver must not retain, sell, or transfer to any third party (except as may be required by law) any personal information, IP addresses or other user identifiers, or user query patterns from the DNS queries sent from the Firefox browser.
3. The resolver must not combine the data that it collects from queries with any other data in any way that can be used to identify individual end users.
4. The resolver must not sell, license, sublicense, or grant any rights to user data to any other person or entity.
5. The resolver must support DNS Query Name Minimisation as defined in RFC 7816.
6. The resolver must not propagate unnecessary information about queries to authoritative name servers. In particular, the client subnet DNS extension in RFC 7871 must not be sent to servers unless the connection to the authoritative server is encrypted and only to authoritative name servers operated by the domain owner directly or by a DNS provider pursuant to its contract with the domain owner.

Mozilla privacy requirements for a Trusted DoH Resolver

Is a less-than-honest resolver always bad?

Examples:
- Captive portals (login pages on public wifi)
- Blocking fraudulent websites
- Schools blocking clubpenguin, etc.
- Company-internal domains (add extra info vs. "public" DNS)

DNS filtering for (weak) ad blocking



Extra content with more detail after this point...

Multiple answers??

Can return multiple answers for the same record. Why??

```
$ 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
```

=> If client can't connect to first result, try the next one

DNS server usually shuffles answers on each response--why?
=> Load balancing + redundancy!

In practice: DNS record types

"ANSWERS"

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>`
TXT	Arbitrary string	. . .

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

```
% dig brown.edu TXT @1.1.1.1

;; ANSWER SECTION:
brown.edu. 3600 IN TXT "google-site-verification=9NxSC7hDW1F0dOB7Hp5sj1b4Zh3YuOLsWALzm0W5e3M"
brown.edu. 3600 IN TXT "docusign=084c221f-92f9-4237-beb8-c25279422ee0"
brown.edu. 3600 IN TXT "atlassian-domain-verification=Hs1rCYbV9DaUuZH78ks2h7V68IDvaPFkmvb6vrxSU/3
brown.edu. 3600 IN TXT "cisco-ci-domain-verification=122c4cac85430ffd529f4c1f7dbc9a90b3d511cd9486
brown.edu. 3600 IN TXT "airtable-verification=5535dab3fd5abf3d31fb73a24ce236a4"
brown.edu. 3600 IN TXT "miro-verification=72ea693488717c97da4e2a48c89fdfee7e673b54"
brown.edu. 3600 IN TXT "v=spf1 ip4:128.148.0.0/16 ip4:209.235.66.235/32 ip4:74.122.104.0/22"
        " ip4:35.163.186.146 ip4:208.117.49.214 ip4:198.2.128.0/18 ip4:148.105.8.0/21
          ip4:198.105.13.0/24" " ip4:206.107.42.249 ip4:206.107.42.254 ip4:206.107.42.9
          ip4:35.169.11.239 ip4:198.187.196.100 include:_spf.google.com include:sendgrid.net includ
brown.edu. 3600 IN TXT "github-verification=ekYLSgdWns5XA28K32JpNpD53dWpQEHQAmg5y7BY"

[ . . . ]
```

TXT records: just arbitrary strings, usually used to validate something about the owner of the domain

Example: SPF: list of valid senders of email for this domain

What happens when you register a new domain?

What happens when you buy a domain?

You get control of yoursite.com
Need an authoritative DNS server for yoursite.com

Two choices:
1. (Most common) Can have external company manage DNS servers for you (Google DNS, amazon route53, name.com, godaddy)

2. Alternatively, you can run the authoritative server yourself

When you buy yoursite.com, an entry gets added to .com that says, “Nameservers for yoursite.com are …”

After this, you can configure actual records for your domain, eg.
yoursite.com => 1.2.3.4
something.yoursite.com => x.x.x.x

…