CSCI-1680
The End (of lectures)
Tor, Wrapup

Nick DeMarinis

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

Administrivia

- HW4: Due Friday 12/8
- Final project: Due 12/14
- Office hours: see the calendar

- Course feedback
  - University feedback
  - Critical Review
  - I will send you a form

My (major) TODOs

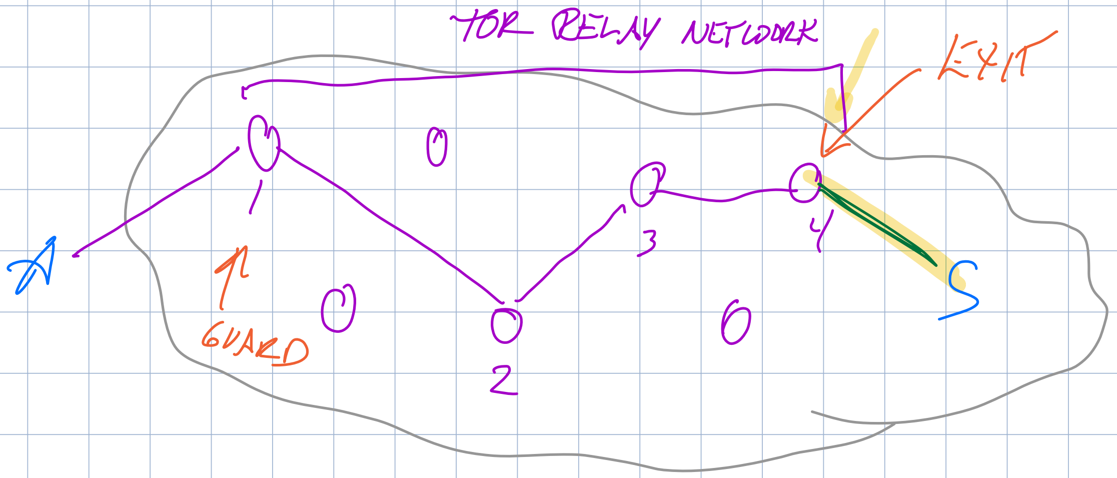
1. I owe you grades on HW2, Snowcast, TCP
2. Will send grade report next week ⇒ LOOK FOR ED ANNOUNCEMENT
3. I will be watching Ed for final project questions

Today’s Lecture

- More about Tor
- Wrapup

More on Tor

HOW TOR WORKS: RECAP



- ONLY GUARD NODE KNOWS A
- ONLY EXIT NODE KNOWS S
-

- IDEALLY, RELAYS OWNED BY MANY DIFFERENT PARTIES

Last hop => traffic is leaving tor network to reach destination server => not protected!

- If not using TLS or other protocol-level security, data is in the clear

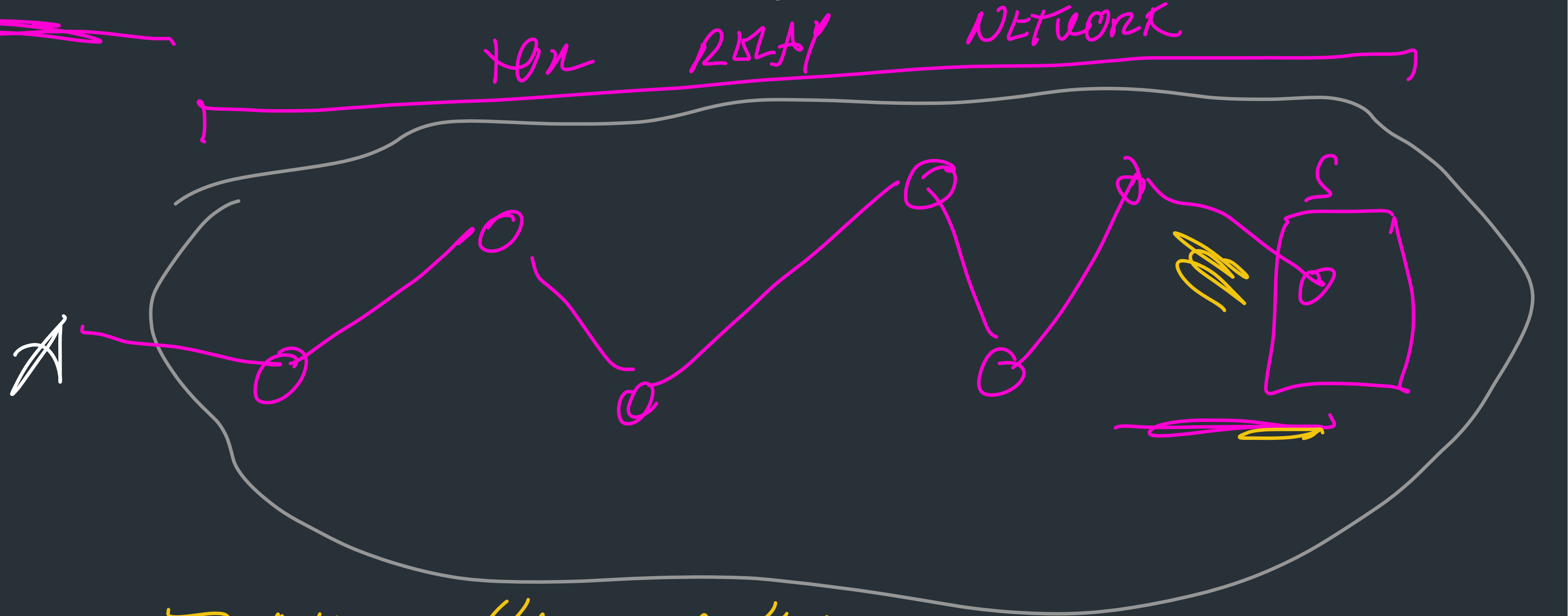
- Depending on the protocol/messages, may leak information that identifies you (eg. cookies, protocol info that contains your IP address)

Q: Why does tor require its own browser? (other than because it's easy)

=> If you used your normal browser, your existing browser state (cookies, etc) can be sent when you visit pages => more likely to identify you

What if the server wants to help?

Onion services: server connects to tor directly => no need for an exit node!



⇒ NEED "ADDRESS" FOR SERVICE

What if the server wants to help?

Onion services: server connects to tor directly => no need for an exit node!

- Accessible via .onion domain: special DNS TLD not in root zone
- Site addresses based on public key of server, client looks up using distributed hash table (DHT)

What if the server wants to help?

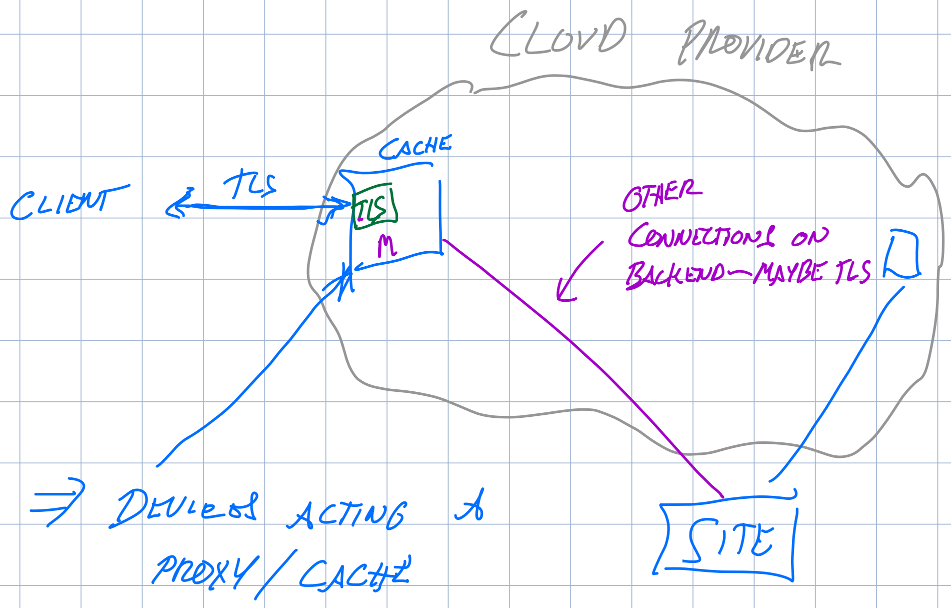
Onion services: server connects to tor directly => no need for an exit node!

- Accessible via .onion domain: special DNS TLD not in root zone
- Site addresses based on public key of server, client looks up using distributed hash table (DHT)

<u>Examples</u>

- *New York Times:*
  `https://www.nytimesn7cgmftshazwhfgzm37qxb44r64ytbb2dj3x62d2lljsciiyd.onion`
- Facebook
  `https://facebookwkhpilnemxj7asaniu7vnjjbiltxjqhye3mhbshg7kx5tfyd.onion`
- Cloudflare public DNS
  `dns4torpnlfs2ifuz2s2yf3fc7rdmsbhm6rw75euj35pac6ap25zgqad.onion`

ADDITIONAL STUFF: CACHING + TLS


CLOUD PROVIDER
CACHE
CLIENT
TLS
TLS
M
OTHER CONNECTIONS ON BACKEND—MAYBE TLS
SITE
⇒ DEVICES ACTING AS PROXY / CACHE

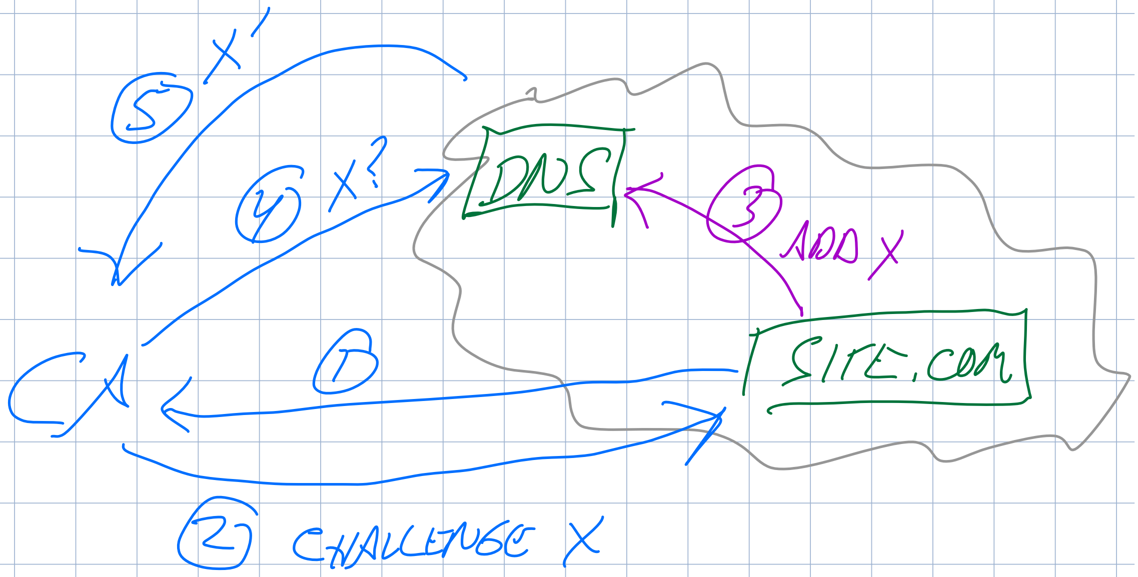
How does caching work with TLS?

- Client makes a TLS connection to some endpoint at cloud provider (cache, etc), not the backend server
- From there, the cache can see the client's request, then respond with cached data or query backend server
  - => Cache needs to have certificate
  - => Traffic is decrypted in the cloud provider (may or may not be what you want)

HOW DOES A CA VALIDATE A CERTIFICATE REQUEST?

Before signing a certificate, a CA should check the requestor's identity in some way. Two ways to do this:

- Organization validation (less common): manually verify contact info, in-person, etc.
- Domain validation (most common): verify that the requestor is in control of the domain name where they are requesting the certificate



How domain validation works:

1. Admin of some site site.com asks CA for certificate
2. CA issues challenge with random value X, asks requestor (admin, etc) to make it viewable on their site. Examples:
    A. eg. Add a DNS record on site.com containing challenge value (TXT record)
    B. Make challenge available on website (ACME protocol)
3. The CA checks for challenge value (DNS lookup for site, etc.) => finds challenge X'
4. If X == X', it means that the requestor can prove control of the site

Eg. Let's Encrypt (2014): Free CA that issues certificates using this method => now extremely common, issues >1M certificates per day

Problem: what if attacker can hijack DNS? Could spoof validation process with spoofed responses, BGP hijacking, …
One solution: need to verify challenge from multiple vantage points (ASes) to avoid querying from one bad server/path

Larger problem: how do we trust that CAs are issuing certificates properly?

Certificate Transparency (RFC9162, 2021): Recent effort to provide open standard to monitor how certificates are issued

- Verifiable, append-only logs of all certificates issued (built using Merkle trees)
- Browsers, CAs, other interested parties can maintain logs

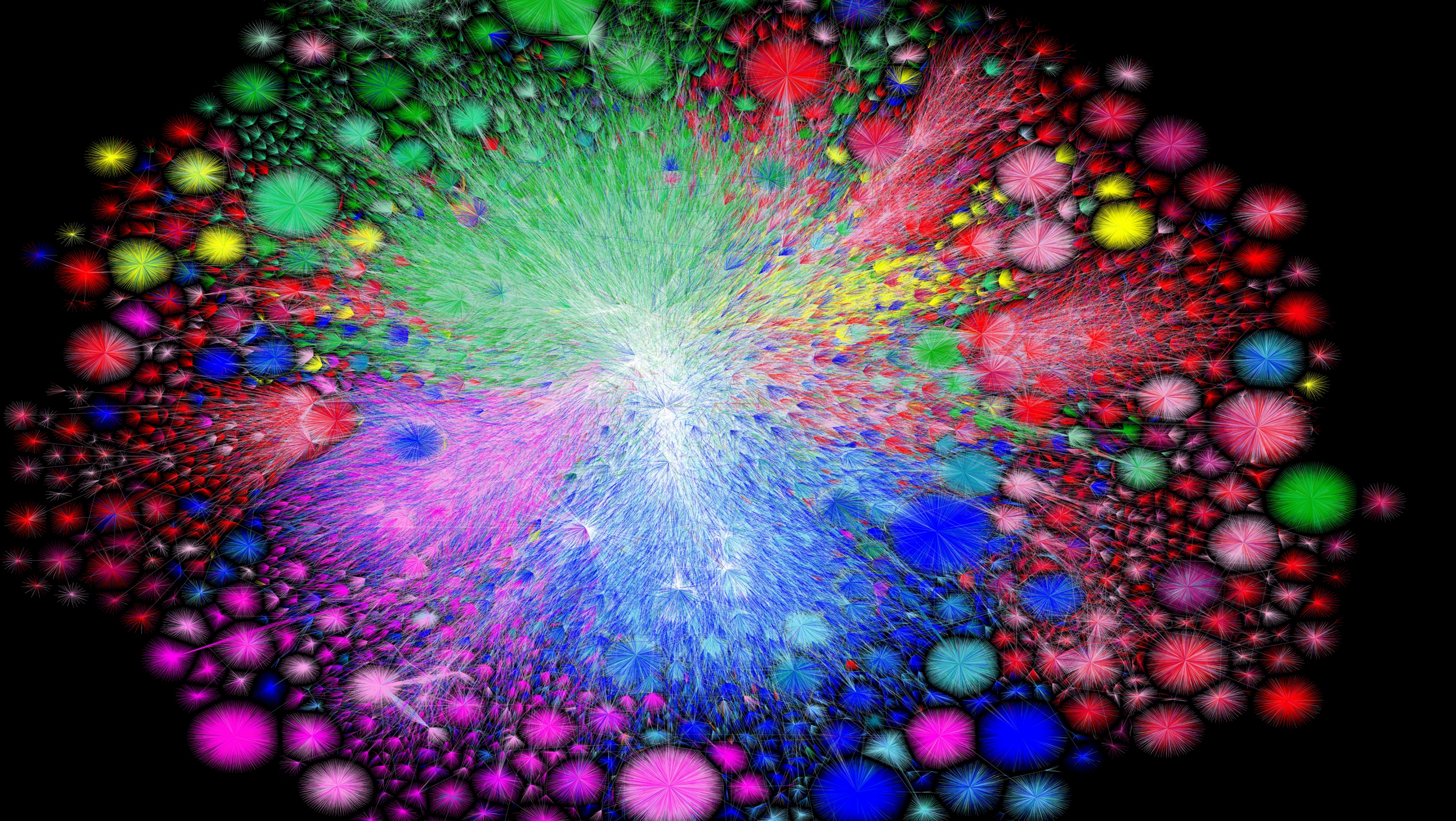
Modern browser vendors are starting to require that CAs use Certificate Transparency in order to be included as a trusted CA

Example CT monitor: https://crt.sh

Wrapping up

- This is our last formal lecture
- From here: work on final project

What I hope you have learned



We can’t cover (or remember) everything

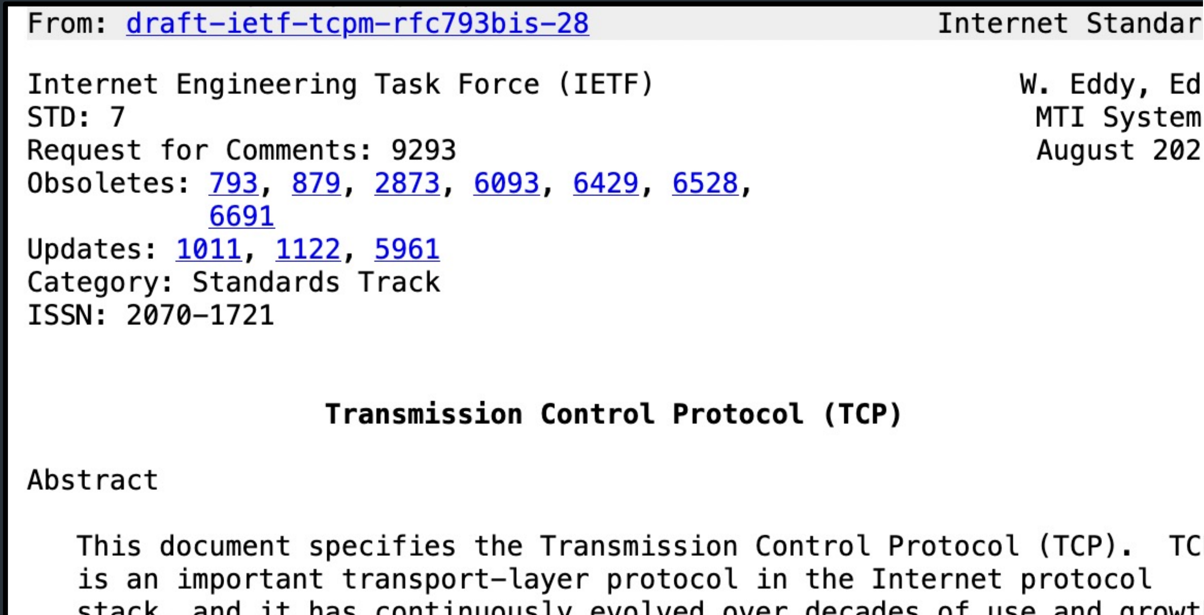
Hope you learn important tools/principles to understand networking challenges you encounter

Protocols Ways to communicate between *heterogeneous* systems

Network programming

```
conn, err :=  net.Dial("tcp", "10.0.0.1:80")
. . .

someBuf := make([]byte, . . .)
conn.Write(someBuf)
```



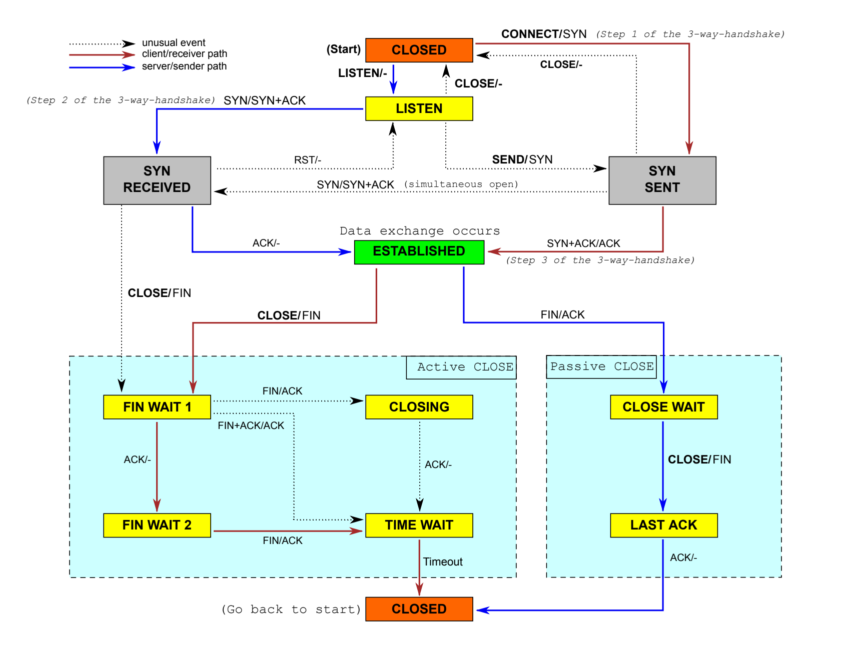
From: draft-ietf-tcpm-rfc793bis-28 Internet Standar

Internet Engineering Task Force (IETF) W. Eddy, Ed
STD: 7 MTI System
Request for Comments: 9293 August 202
Obsoletes: 793, 879, 2873, 6093, 6429, 6528,
6691
Updates: 1011, 1122, 5961
Category: Standards Track
ISSN: 2070-1721

Transmission Control Protocol (TCP)

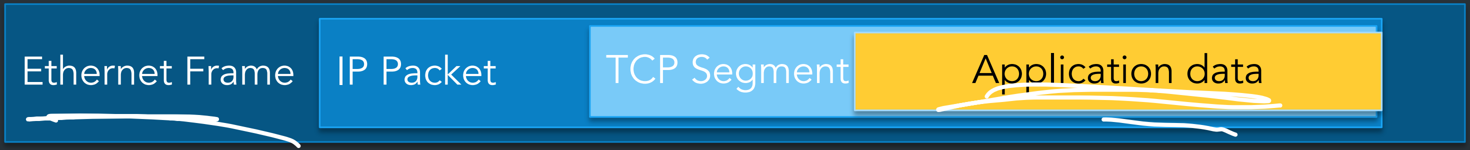
Abstract

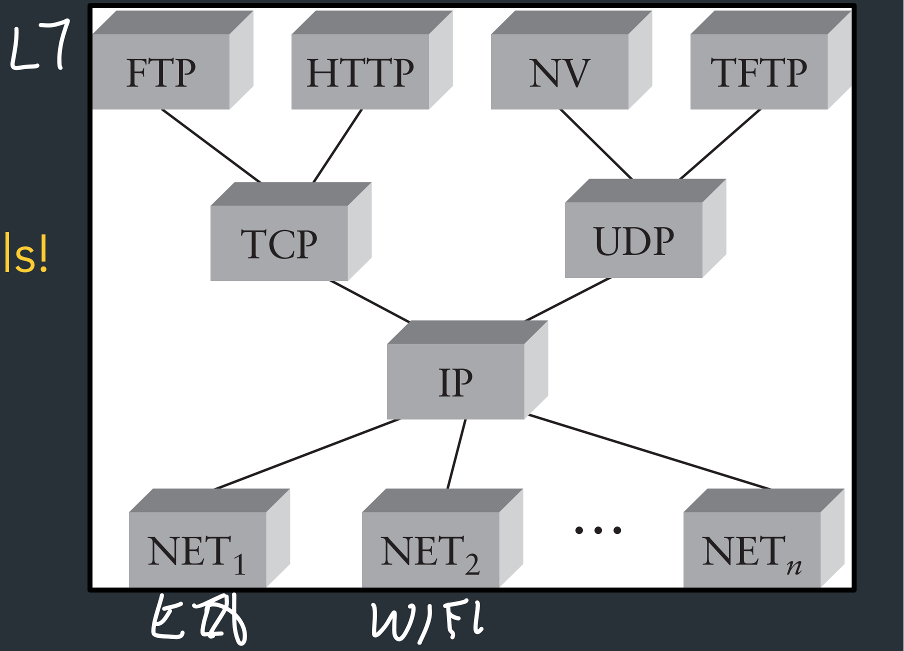
This document specifies the Transmission Control Protocol (TCP). TCP is an important transport-layer protocol in the Internet protocol stack, and it has continuously evolved over decades of use and growth



Layering / Encapsulation

Building abstractions and interfaces to hide lower-level details from "higher" layers





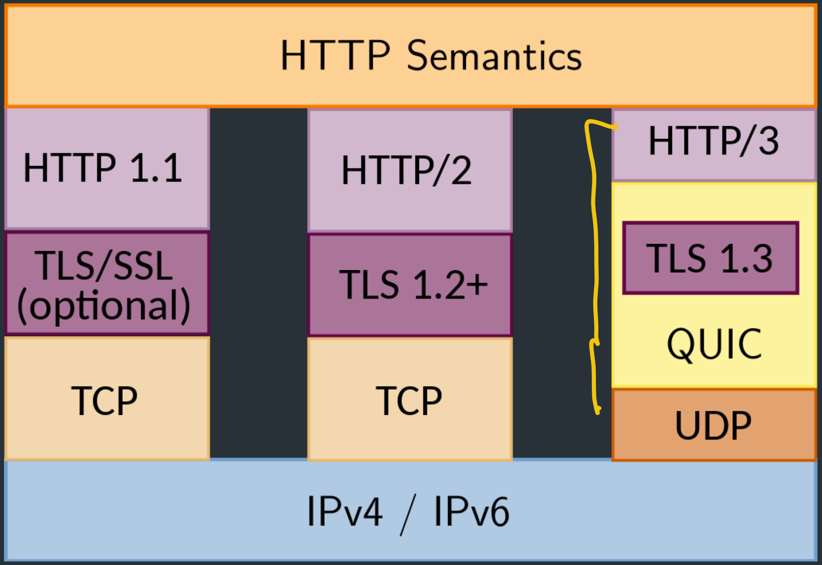
Abstractions are great!

- Can support huge variety of devices, protocols
- Allows independent evolution => new protocols!

… until they aren't

Sometimes, need to break them

(USUALLY PERFORMANCE)



HTTP Semantics

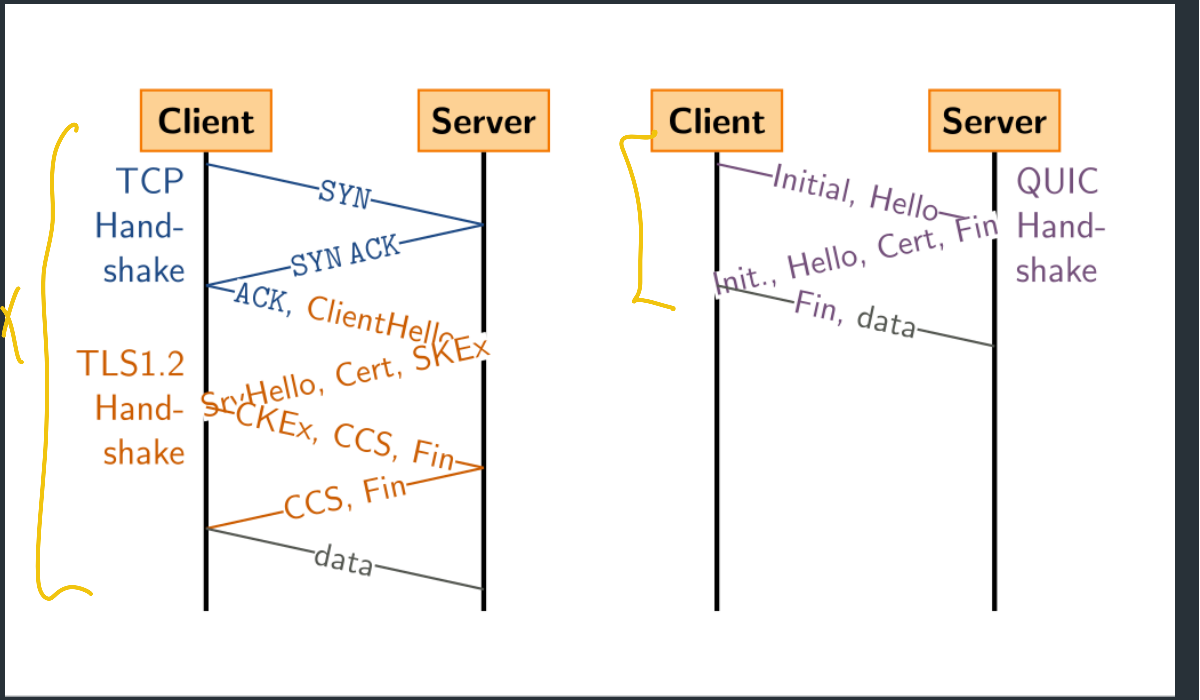
HTTP 1.1 | HTTP/2 | HTTP/3

TLS/SSL (optional) | TLS 1.2+ | TLS 1.3

QUIC

TCP | TCP | UDP

IPv4 / IPv6



Naming

Indirection: abstract low-level info with a higher-level name

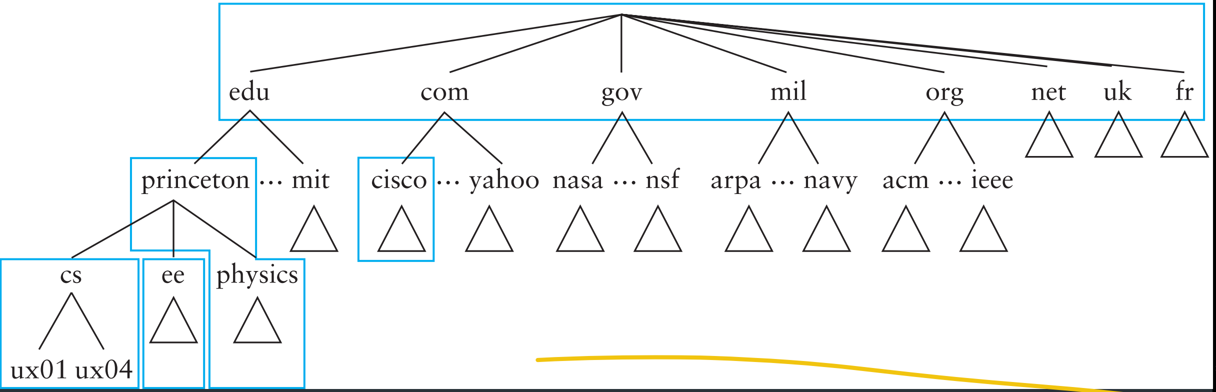
=> Human-readable DNS names

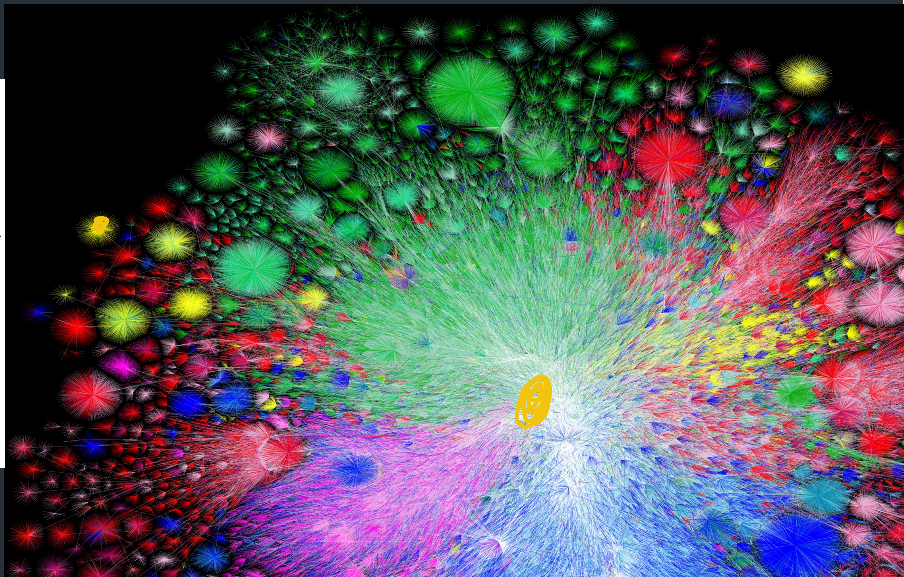
=> Scalability: redundancy, proxies, load balancing

[illegible]

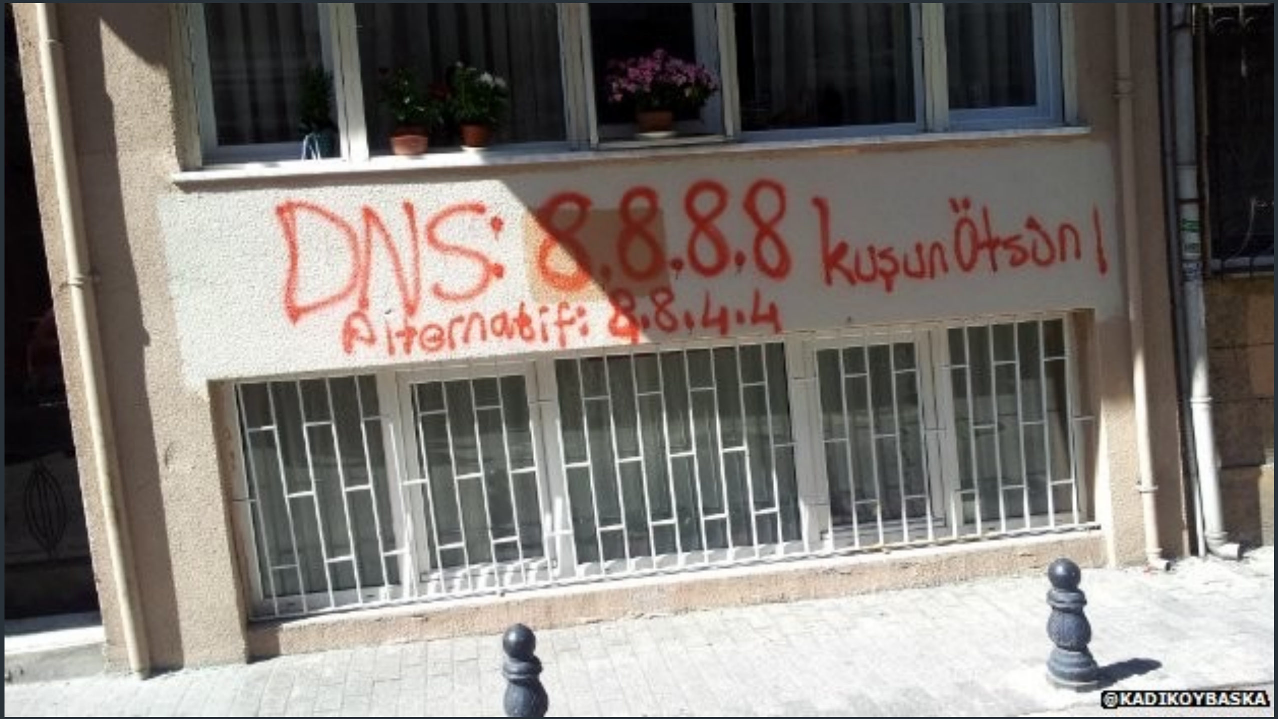
CDNs/Caching

Can leverage hierarchy of naming => scalability (IP, DNS, …)





How naming, etc. can be controlled…



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

Lots of challenges out there

Our Internet architecture was designed in the 1980s, where modern scale and complexity was unimaginable

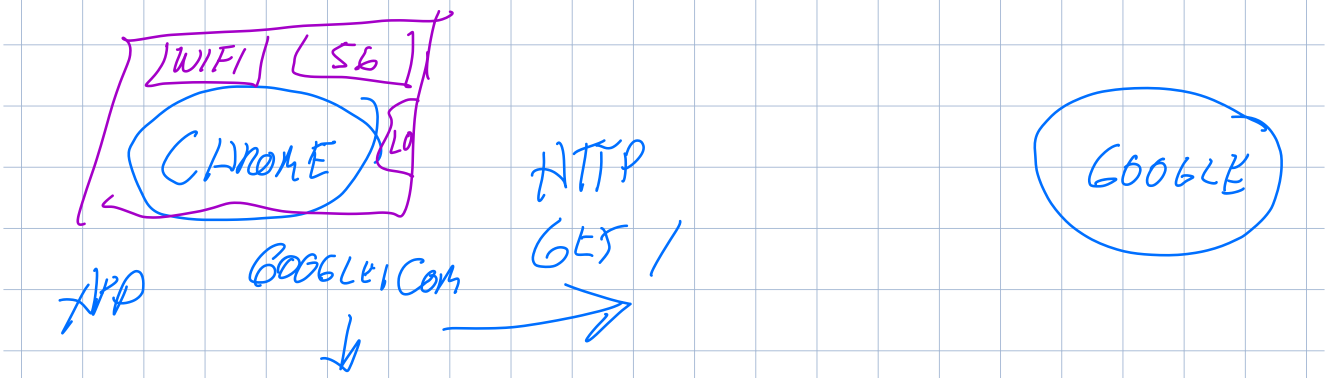
Lots of challenges out there

Our Internet architecture was designed in the 1980s, where modern scale and complexity was unimaginable

Now…

- No one knows how big the Internet is
- No one is in charge
- Anyone can add any application
- Packets traverse many paths, countries, regulatory domains

Thank you!
Please stay in touch!



```
net.Dial("tcp", google.com:80")
```

OS

```
Is this cached in OS DNS resolver

Otherwise: DNS server on current network
(Your OS has a default DNS server)
DNS: 1.1.1.1
```

2

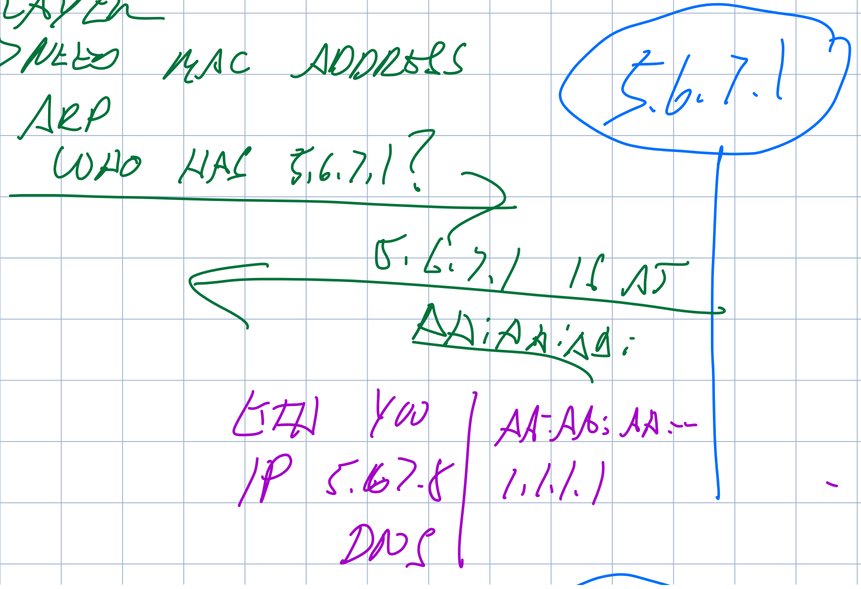
IP

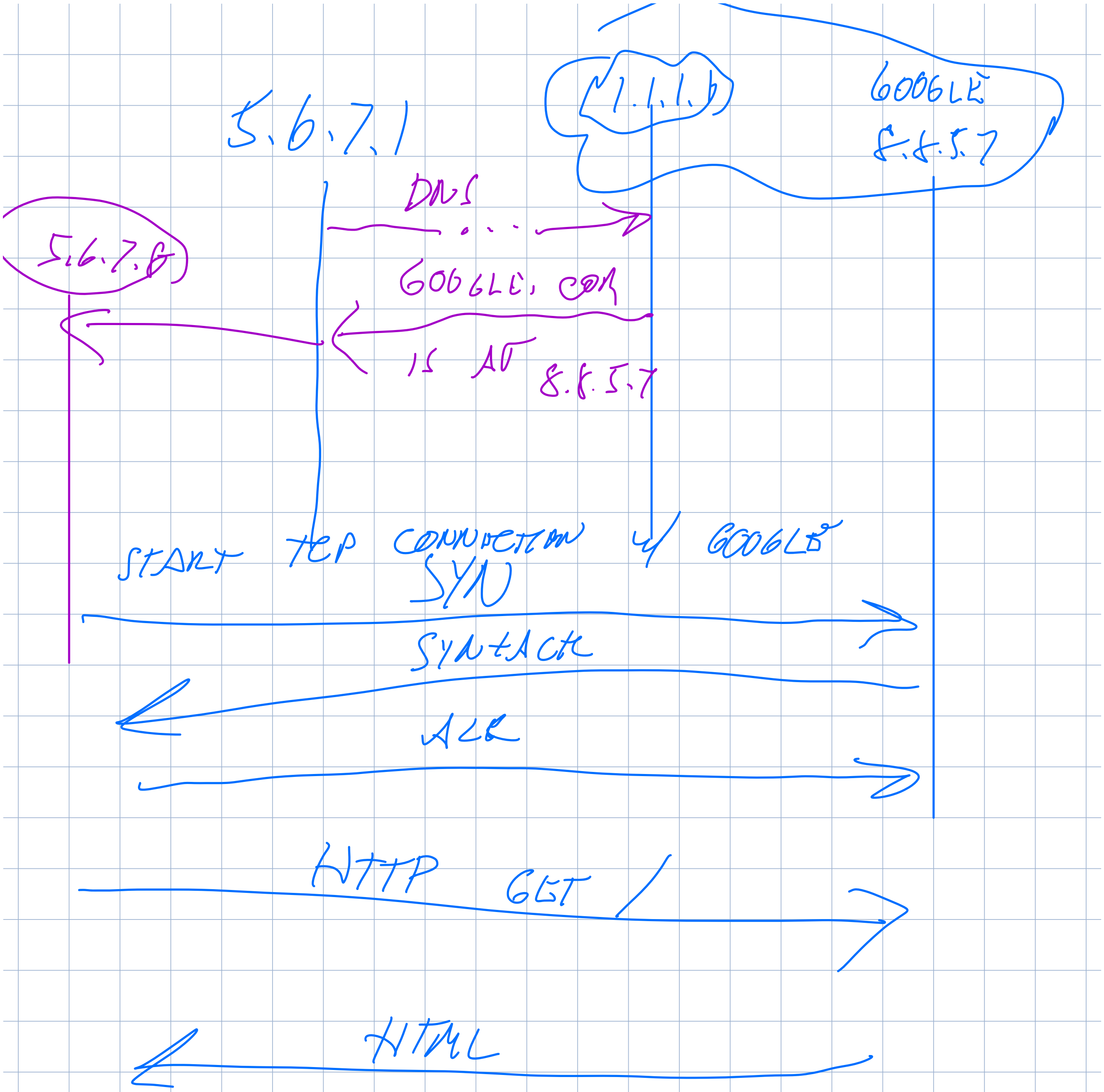
SRC: 5.6.7.8
DST: 1.1.1.1
DNS GOOGLE.COM

```
Consult forwarding table, find outgoing interface
for 1.1.1.1

=> Default gateway is 5.6.7.1, this is next hop
```

LINK LAYER
=> NEED MAC ADDRESS
ARP
WHO HAS 5.6.7.1?




5.6.7.1
1.1.1.1
GOOGLE
8.8.5.7
5.6.7.8
DNS
GOOGLE.COM
IS AT 8.8.5.7
START TCP CONNECTION W/ GOOGLE
SYN
SYN+ACK
ACK
HTTP GET /
HTML