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
3. I will be watching Ed for final project questions

Today’s Lecture

- More about Tor
- Wrapup

More on Tor

What if the server wants to help?

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

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)

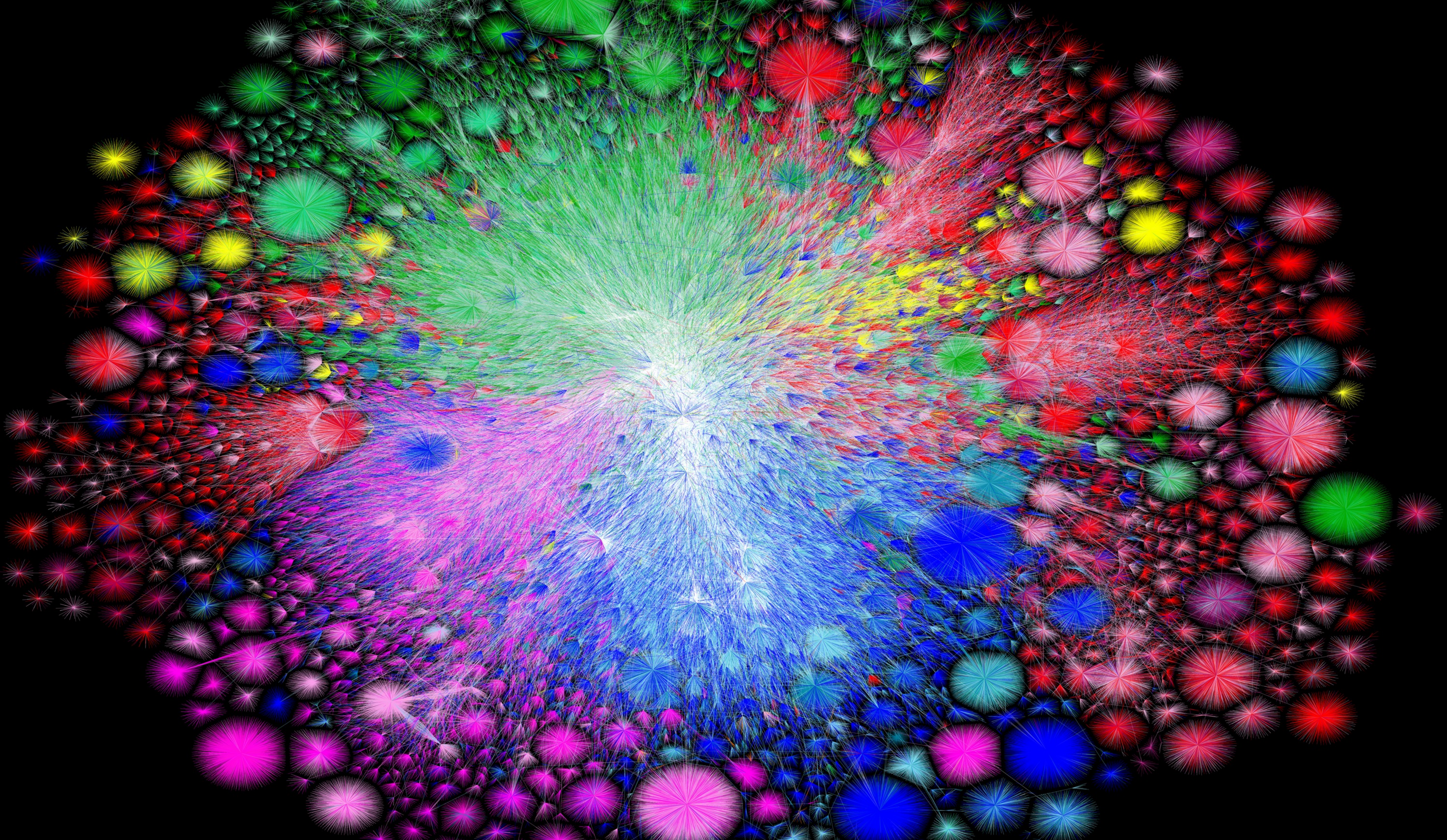
Examples

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

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

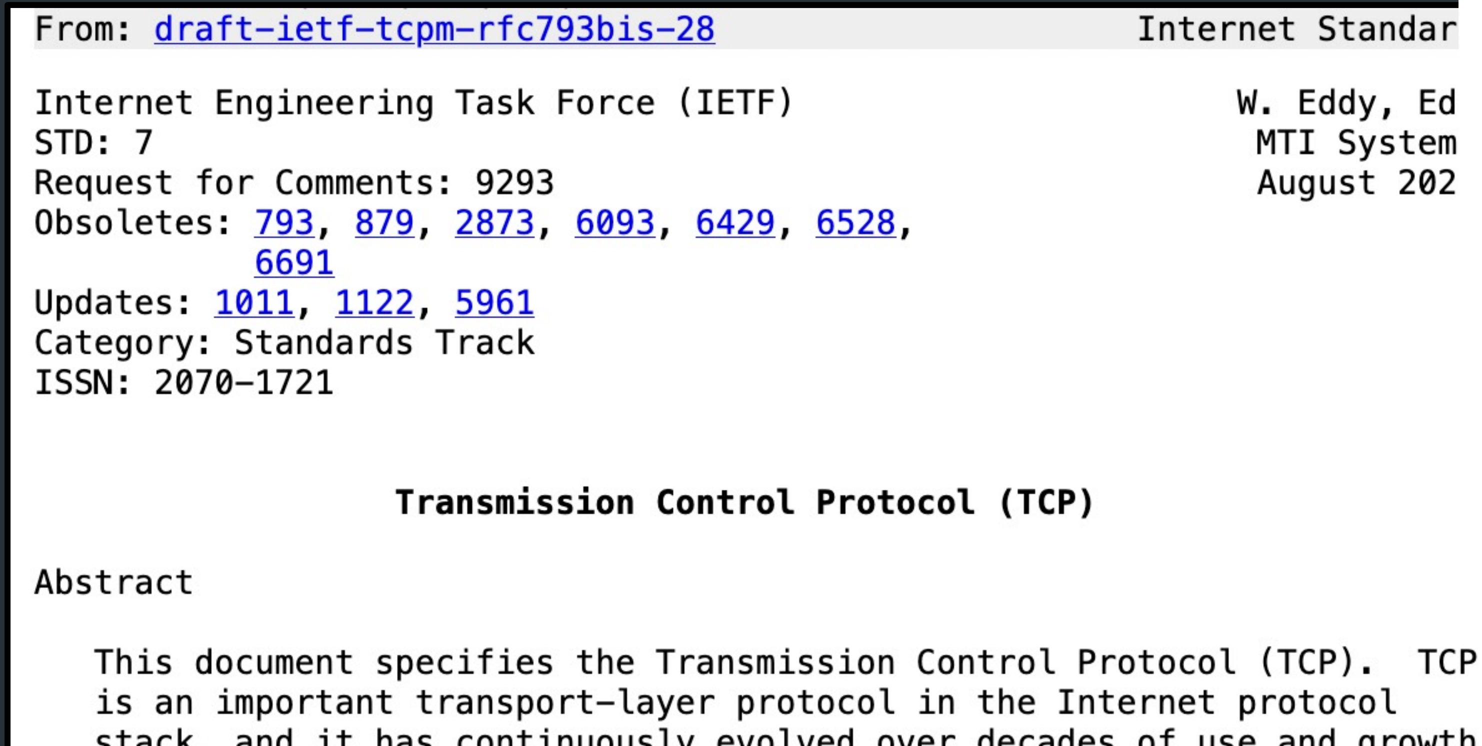
Hope you learn important tools/principles to understand networking challenges you encounter

Protocols

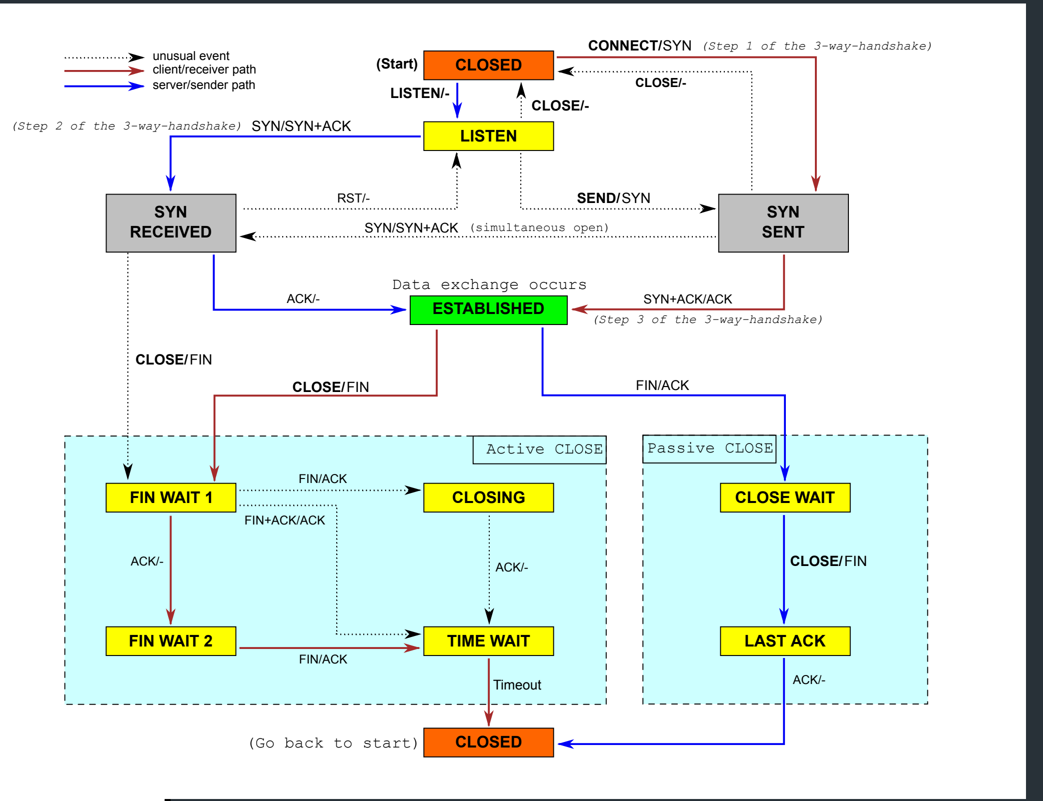
Ways to communicate between *heterogeneous* systems

Network programming

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

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

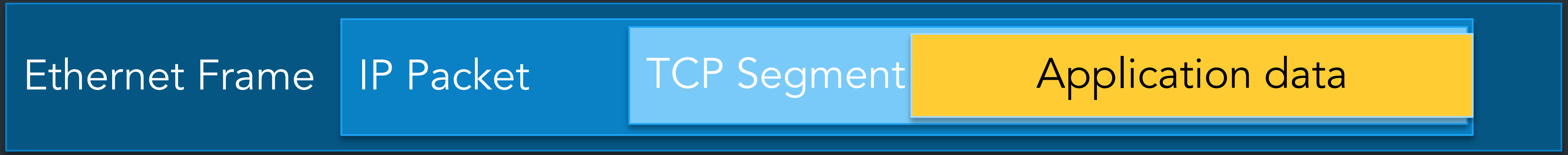


```
From: draft-ietf-tcpm-rfc793bis-28                  Internet Standar

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
```



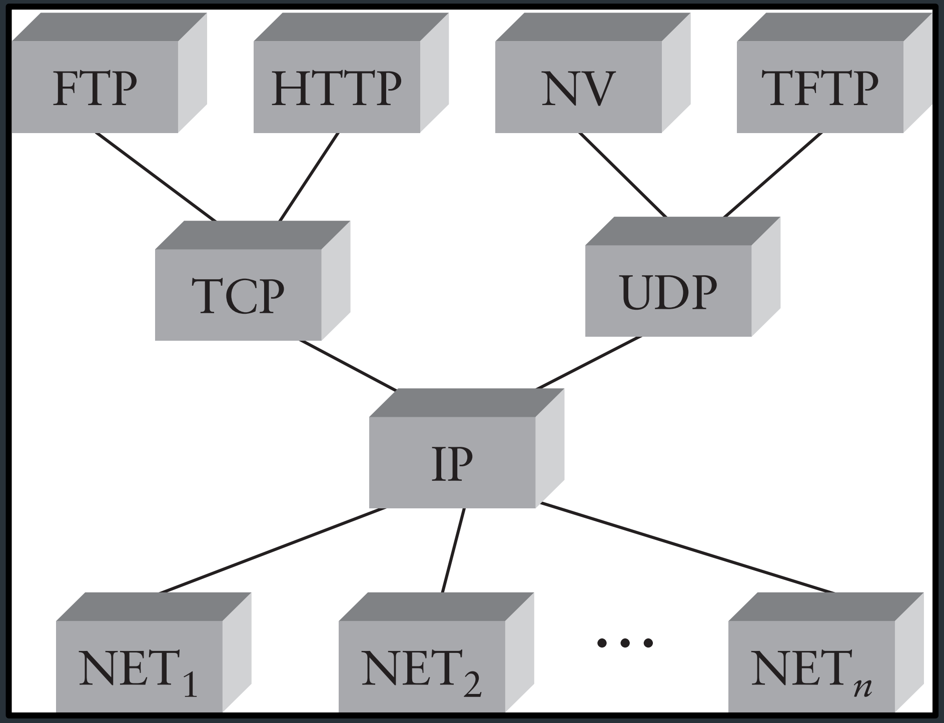
Layering / Encapsulation

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



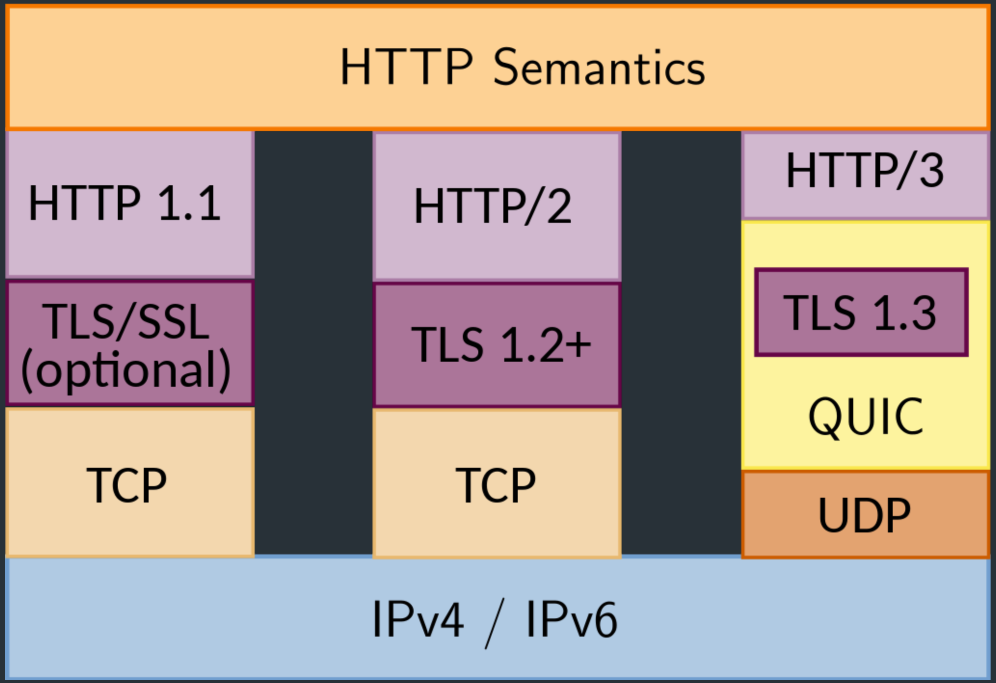
Abstractions are great!

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



… until they aren't

Sometimes, need to break them



HTTP Semantics

HTTP 1.1

TLS/SSL (optional)

TCP

HTTP/2

TLS 1.2+

TCP

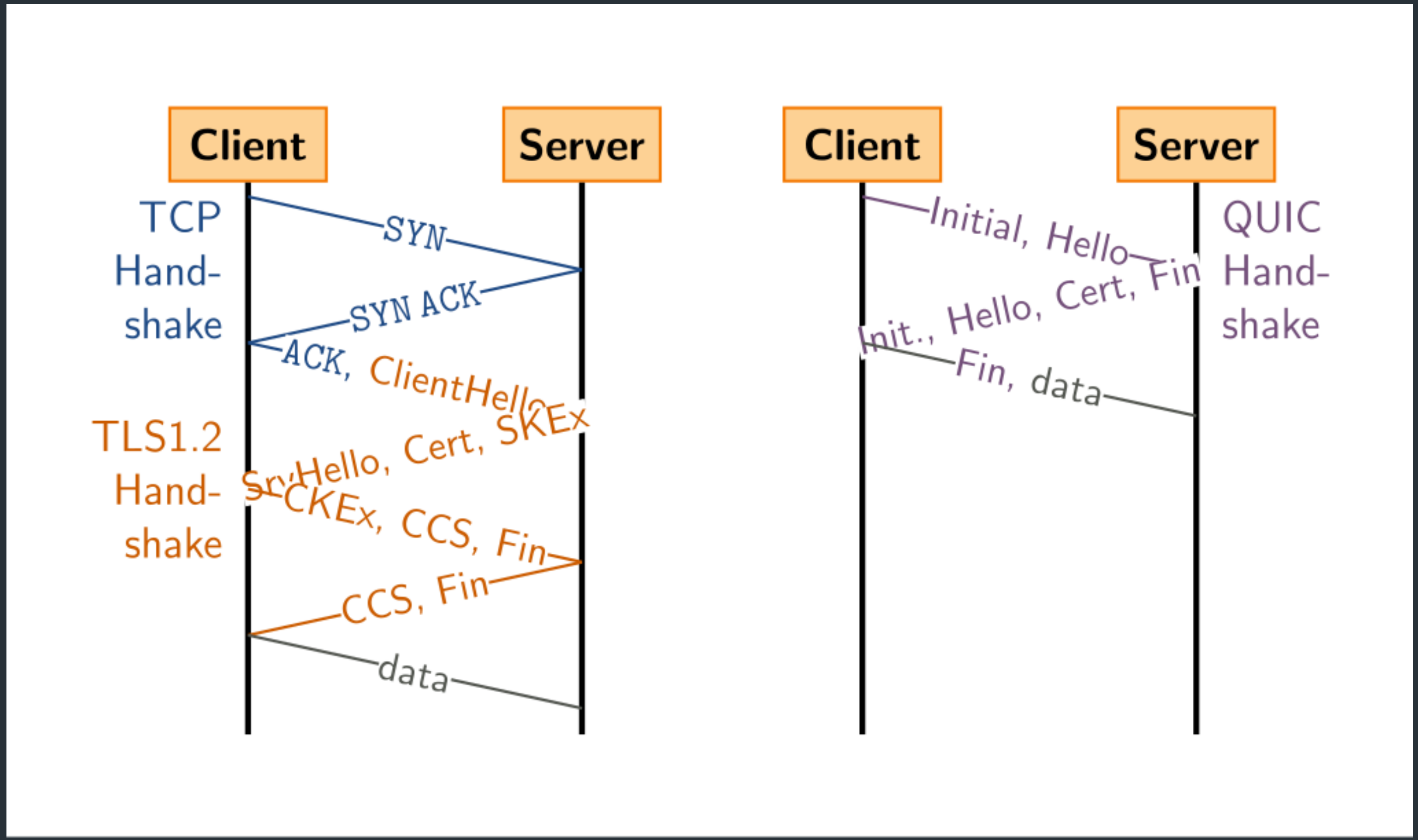
HTTP/3

TLS 1.3

QUIC

UDP

IPv4 / IPv6



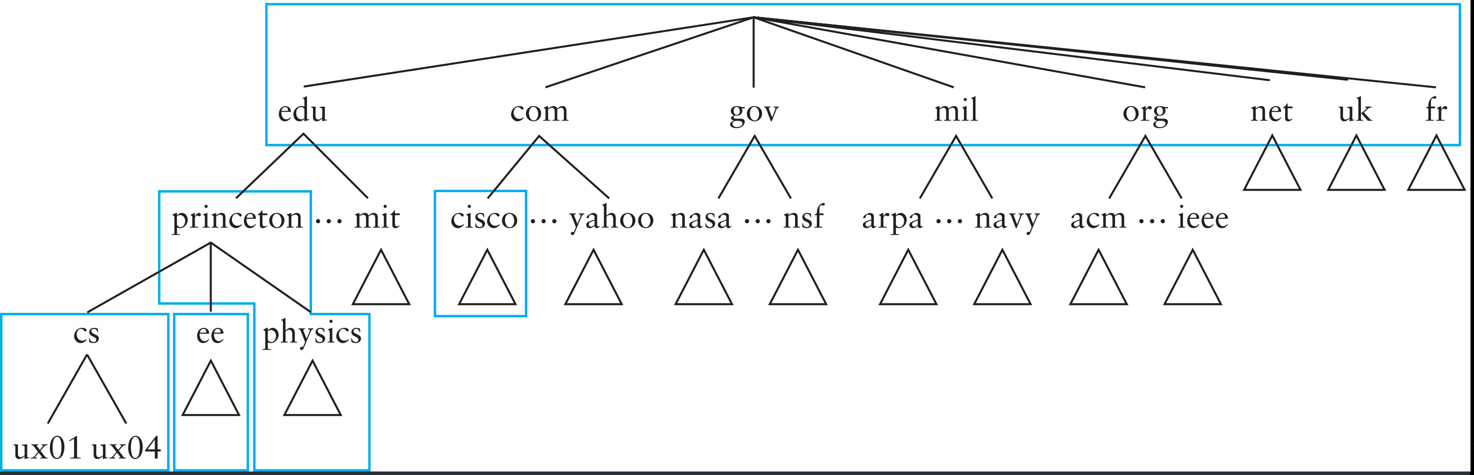
Naming

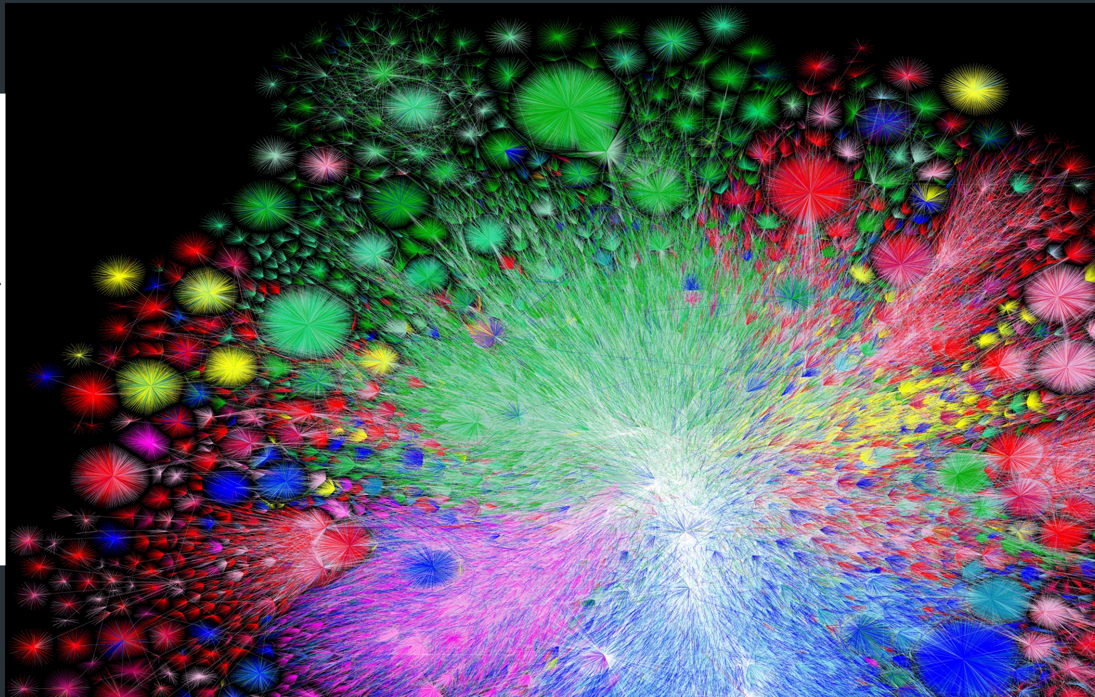
Indirection: abstract low-level info with a higher-level name

=> Human-readable DNS names

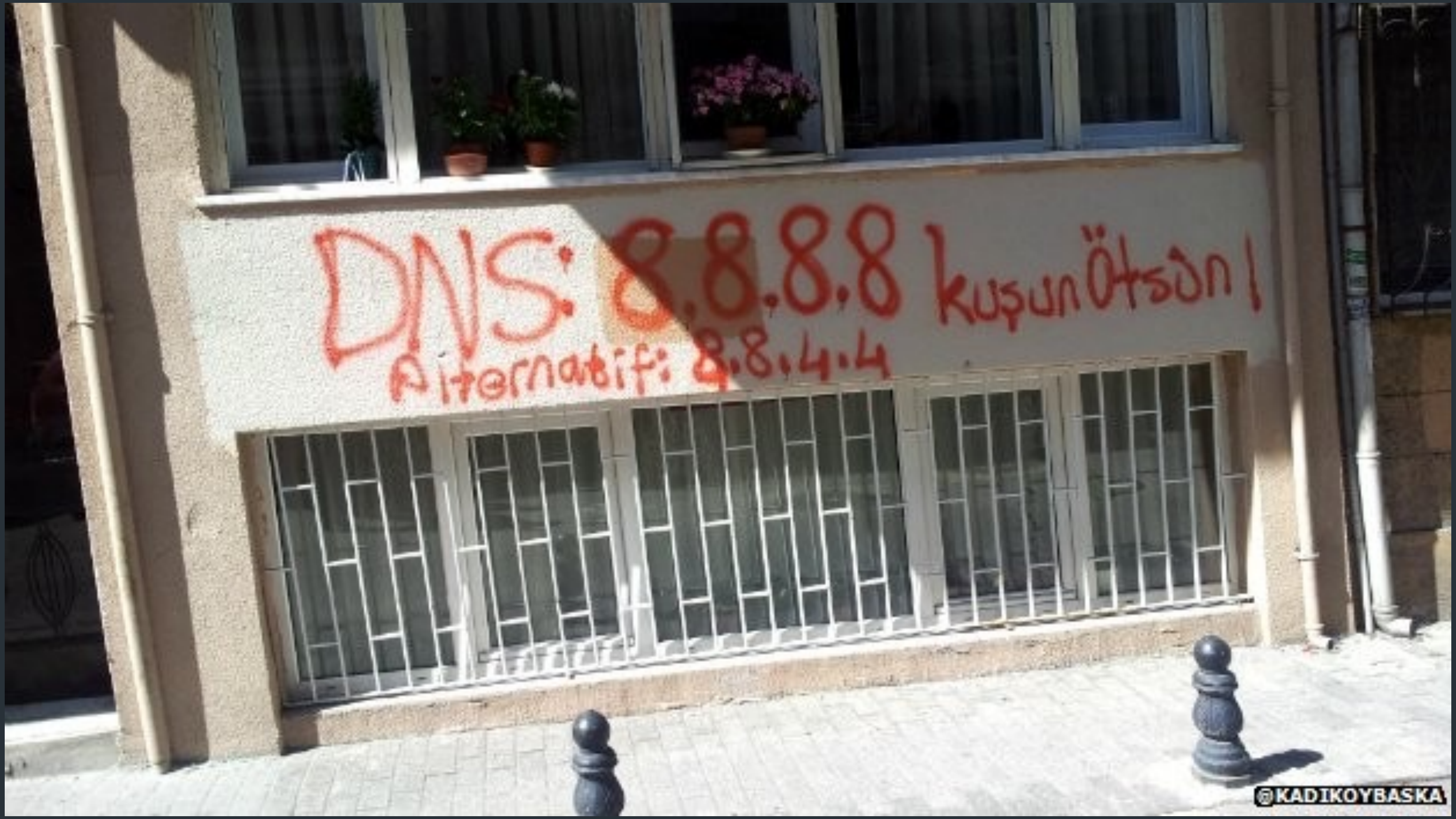
=> Scalability: redundancy, proxies, load balancing

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

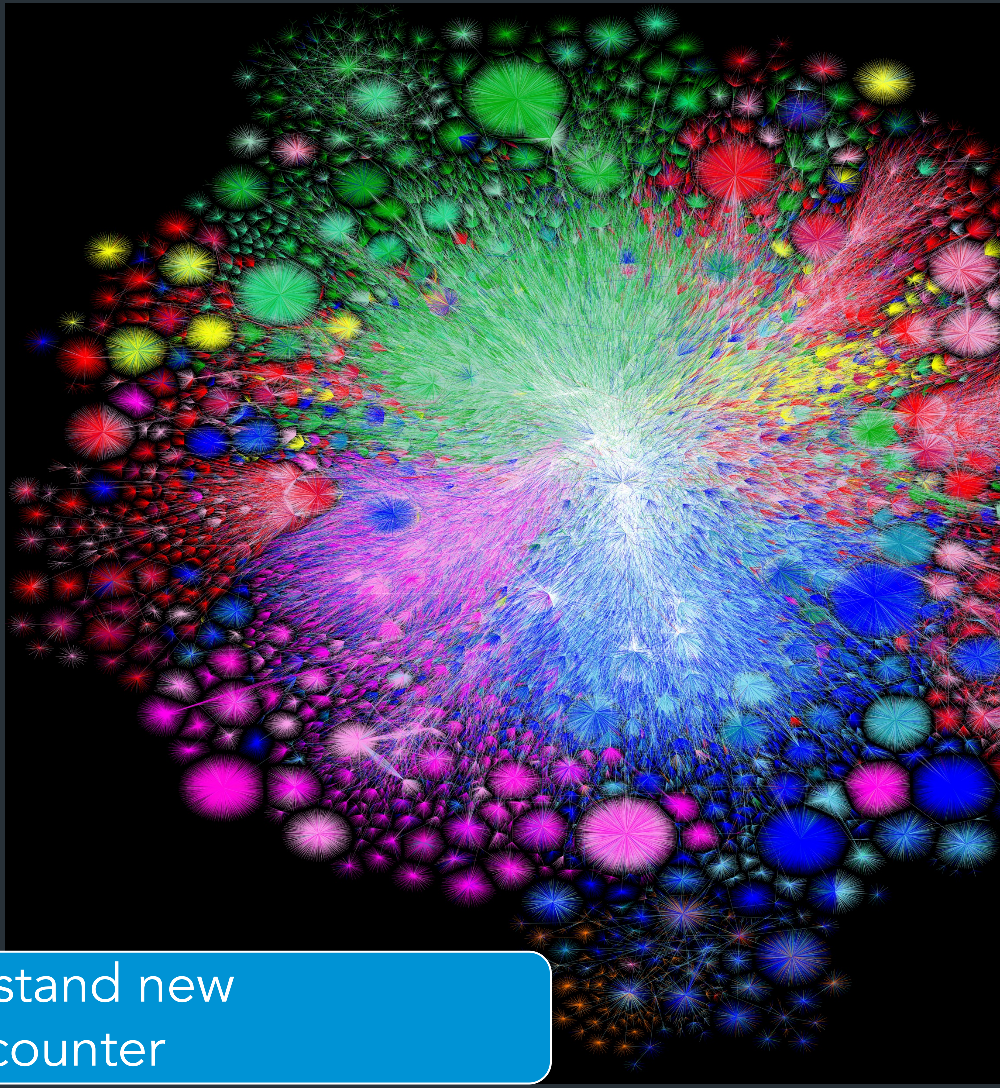
Now…

- 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!

What (I hope) you have learned

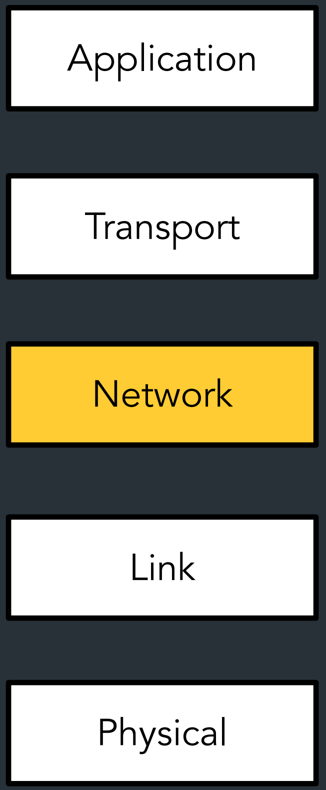
- Skill: network programming (and soft. eng) Socket programming
  - Server programming/robustness
  - Implementing protocols
- Knowledge: How the Internet Works
  - Internet architecture and design
  - Key Internet protocols
  - Some applications (Web, DNS, …)



My goal: give you tools to understand new networking challenges you encounter

Networking principles

- Some general CS concepts
  - Hierarchy (IP addressing, DNS, PKI, …)
  - Indirection (ARP, DNS, …)
  - Caching
- Some concepts (a bit) networking-specific
  - Layering
  - Multiplexing
  - End-to-end argument
  - Robustness principles



Service: user-facing application.
Application-defined messages

Service: multiplexing applications
Reliable byte stream to other node (TC
Unreliable datagram (UDP)

Service: move packets to any other no
Internet Protocol (IP)

Service: move frames to other node ac
May add reliability, medium access cor

Service: move bits to other node acros

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