CSCI-1680
How to (try) to be anonymous
Wrapup

Nick DeMarinis

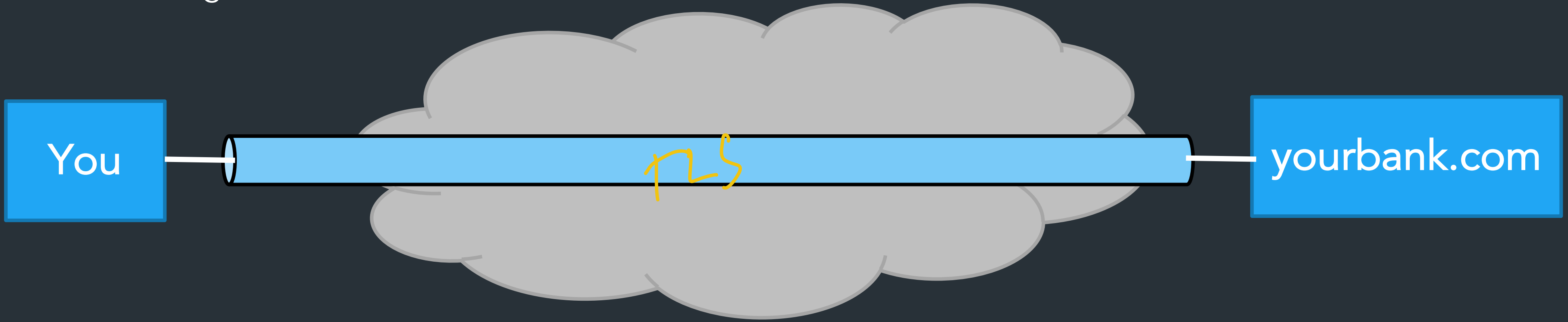
Based partly on lecture notes by Rodrigo Fonseca, Scott Shenker and John Jannotti

My (major) TODOs

1. I owe you grades on Snowcast, TCP
2. I owe you a bunch of lecture notes
3. I will be watching Ed for final project questions

Warmup

With TLS we get this:



Are we good? Have we solved web security?

Header info for src/dst: who you are, who you're talking to
- Can see protocol (by port numbers)
- Not all applications support TLS/ encryption, so maybe want something more hollistic
=> Would like to do security at a lower llayer than the transport layer

So, are we good?

If we use TLS, is it enough?

Overall, depends on your threat model…

- Server still knows who you are, even if connection is encrypted

- Even encrypted traffic leaks information!

Overall, depends on your threat model…

- Server still knows who you are, even if connection is encrypted

=> IPs can be traced to location (to varying levels of precision) ← (GeoIP)

=> Your browser may leak info (cookies, mouse usage, etc.)

- Even encrypted traffic leaks information!

=> Name of server: DNS, Server Name Indicator (SNI)

=> Traffic patterns (timing of packets, protocols, …)

Securing the transport layer not enough => info leaks based on other layers

Why?

- Avoiding censorship
- Avoiding surveillance (by person, or an organization)
- Anonymous reporting (journalists, whistleblowers)

Room 641A: alleged wiretapping room in a datacenter for an Internet backbone…
https://en.wikipedia.org/wiki/Room_641A

How can we deal with this?

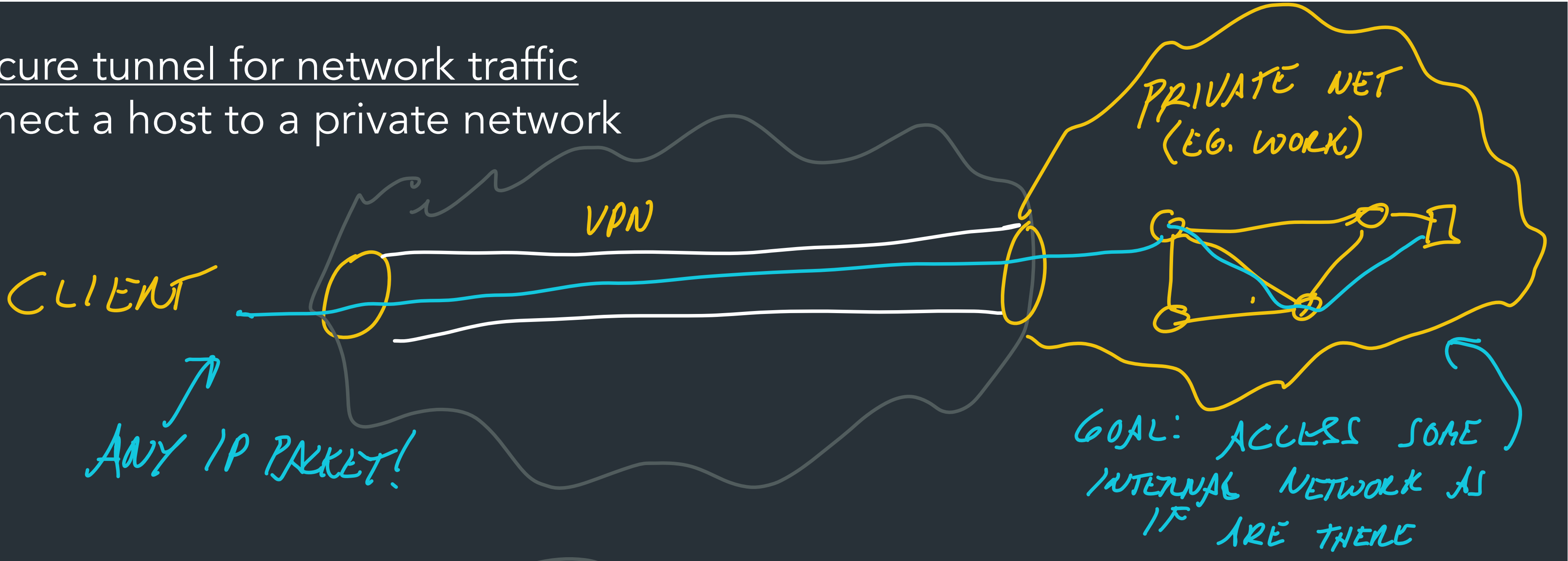
Mechanisms to provide more security at the network layer

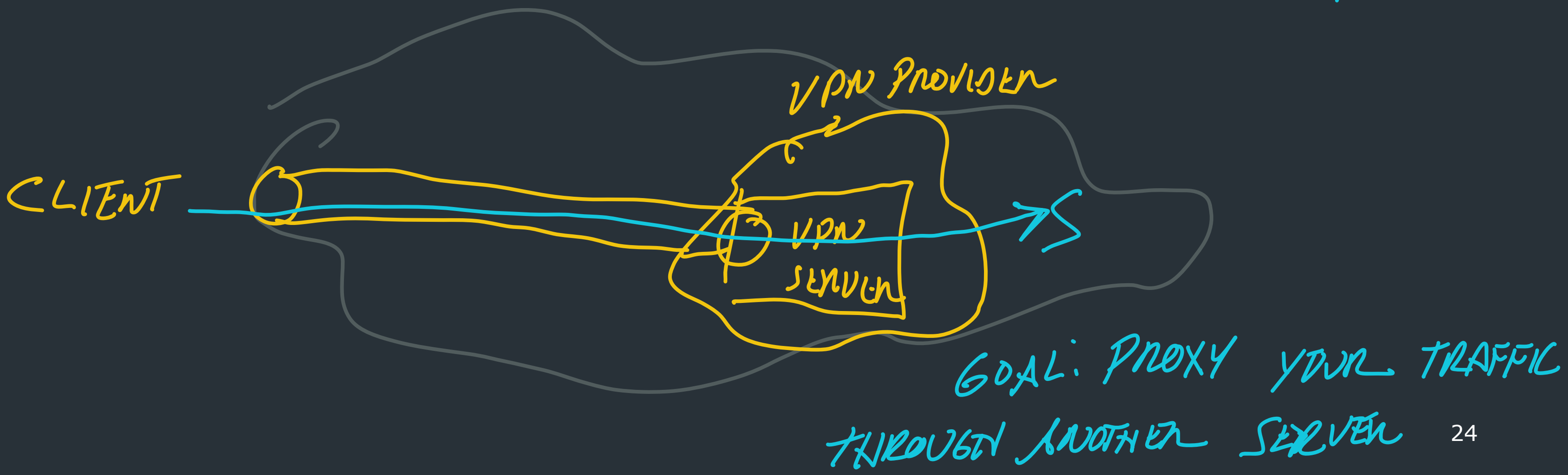
⇒ Security for all your network traffic => not just one 5-tuple

⇒ Can (try to) provide more anonymity

VPN: secure tunnel for network traffic

=> Connect a host to a private network





Virtual Private Network (VPN)

Secure tunnel for arbitrary network traffic (any IP packets)

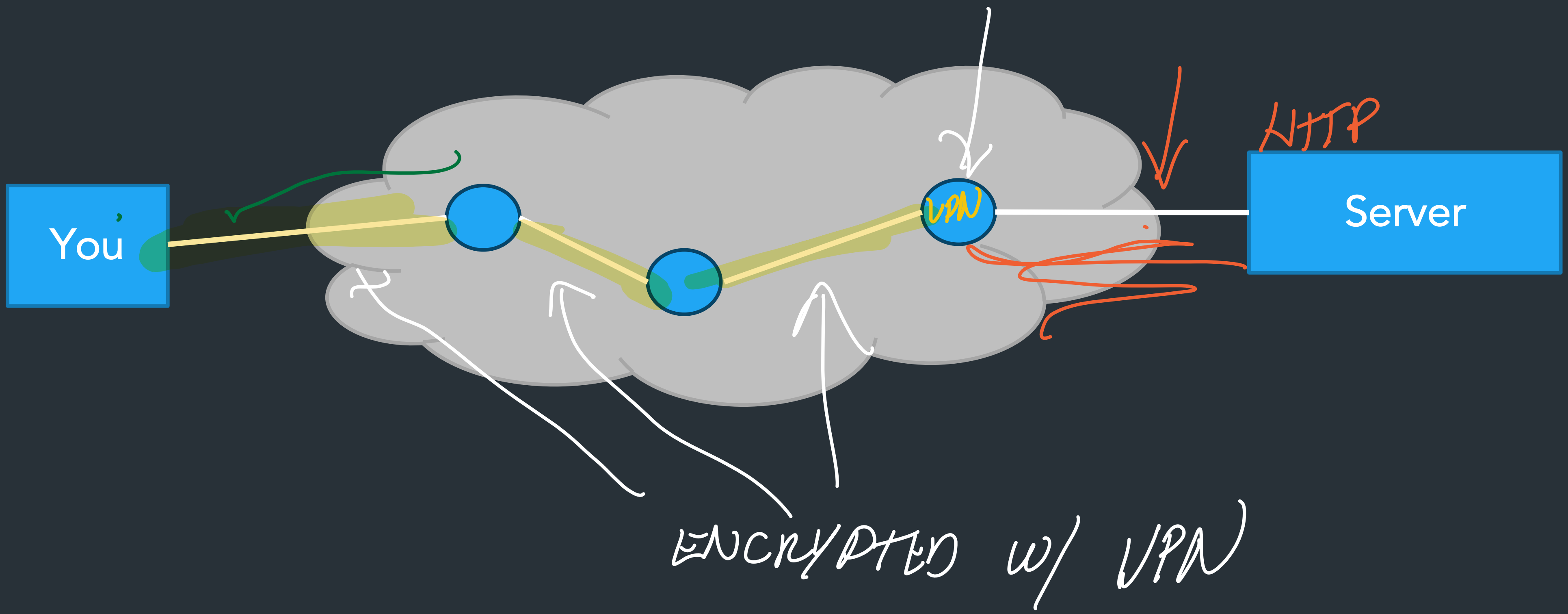
Use for

=> Accessing a private network (remote access internal network)

=> Secure proxy for your traffic: traffic appears to originate from VPN server

VPN: secure tunnel for network traffic

=> Connect a host to a private network



Q: How does a VPN client get all traffic to go to the VPN?

A VPN client will create a special network interface (on Linux, called a TUN interface) and then update the routing table to redirect (usually) all traffic to the VPN

Example: Normal routing table

```
default via 138.16.161.1   # Brown's router
138.16.161.0/24 wifi0
```

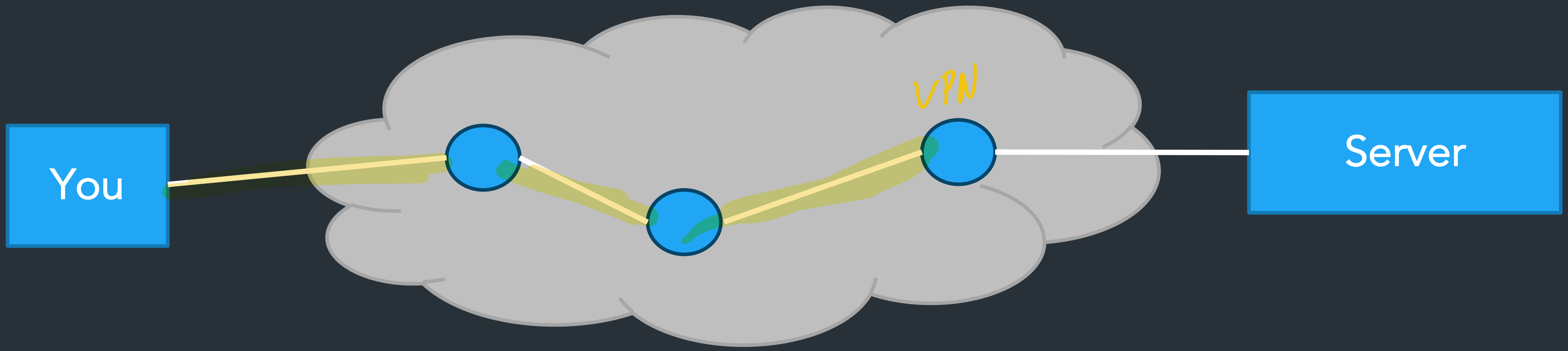
=> When you start a VPN client and make a connection, it adds an interface to this list

```
default via 10.2.3.4 priority 1
default via 138.16.161.1 priority 2  # Brown's router
10.2.3.4 via tun0
138.16.161.0/24 via wifi0
```

A bit more complicated than this in practice:

- Some VPN clients just send some traffic to the VPN (eg. company network)
- Need to make sure that traffic for the VPN client itself can still access the Internet
- If connection drops, need to adjust rules carefully so that 1) traffic you want to be encrypted isn't leaked, and 2) VPN client can still reconnect

Problems?



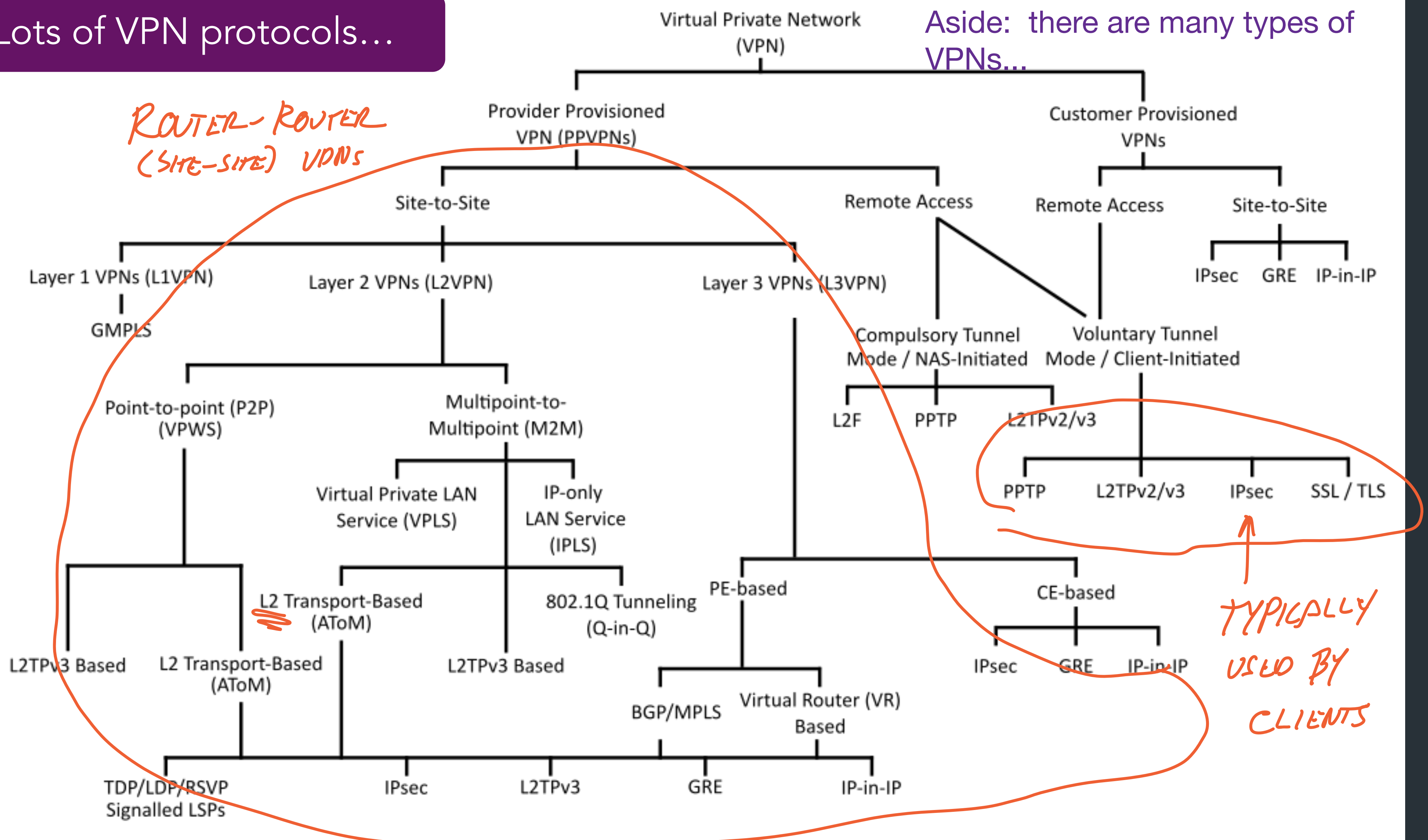
Security implications: VPNs

- Traffic still might be in the clear (vulnerable to eavesdropping/ analysis) once it leaves VPN
- Your traffic may blend in with other users (other VPN clients on same IP), but maybe not
  - => Possible to de-anonymize user with timing/correlation info

- VPN provider still knows who you are

Lots of VPN protocols…

Aside: there are many types of VPNs...

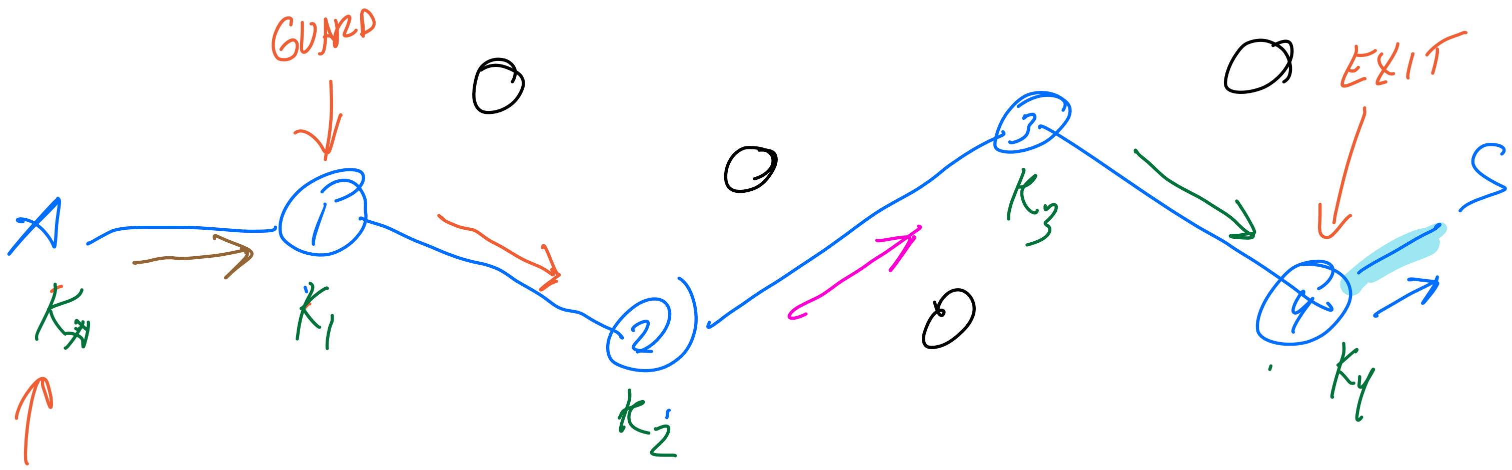


Can we do better?

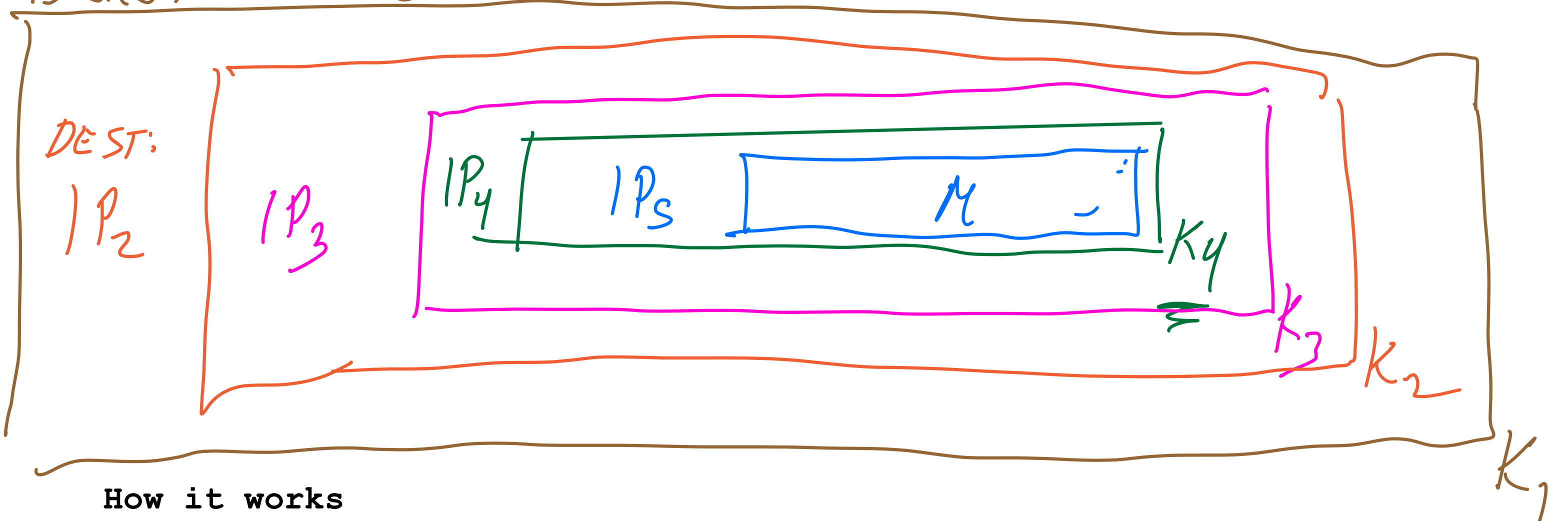
Tor

- Onion routing service: build encrypted circuit on tor relay network
- Network of relays, mainly operated by volunteers
- Started in 1990s from Naval Research Lab, now maintained by The Tor Project (a non-profit)

ONION ROUTING



PACKET RECEIVED BY NODE 1:

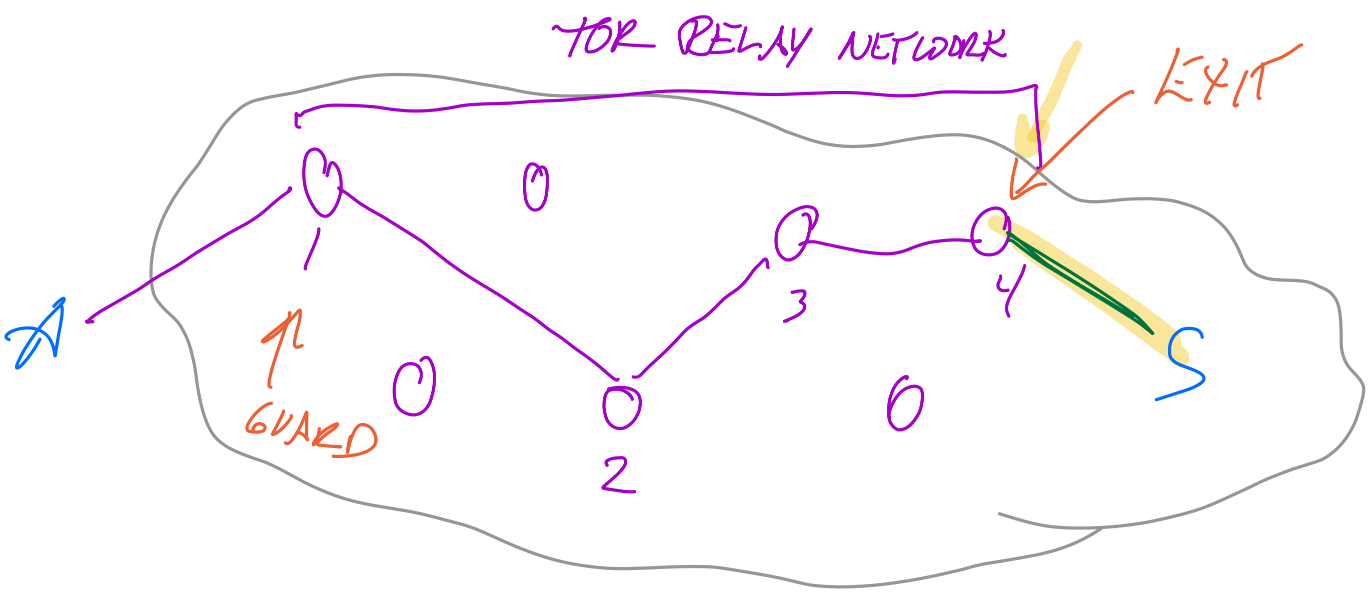


How it works
- Directory service knows about all relay nodes, which are run by many volunteers
- At connection start, establish a "circuit" or path from a subset of relays => know a symmetric key for each relay
  - First node is "guard node"
  - Last relay node is "exit node"

Send packets with layers of encryption => at each hop, relay can decrypt its layer of the encryption with its key
  - Unless it's the last node, it can only see the <u>destination of the next node</u> and the encrypted packet
    (At exit node, the packet is in cleartext => goes to destination)

Thus:
  - Only the guard node knows about the host
  - Only exit node knows the packet's true destination
  - Relays in middle only know about next hop

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

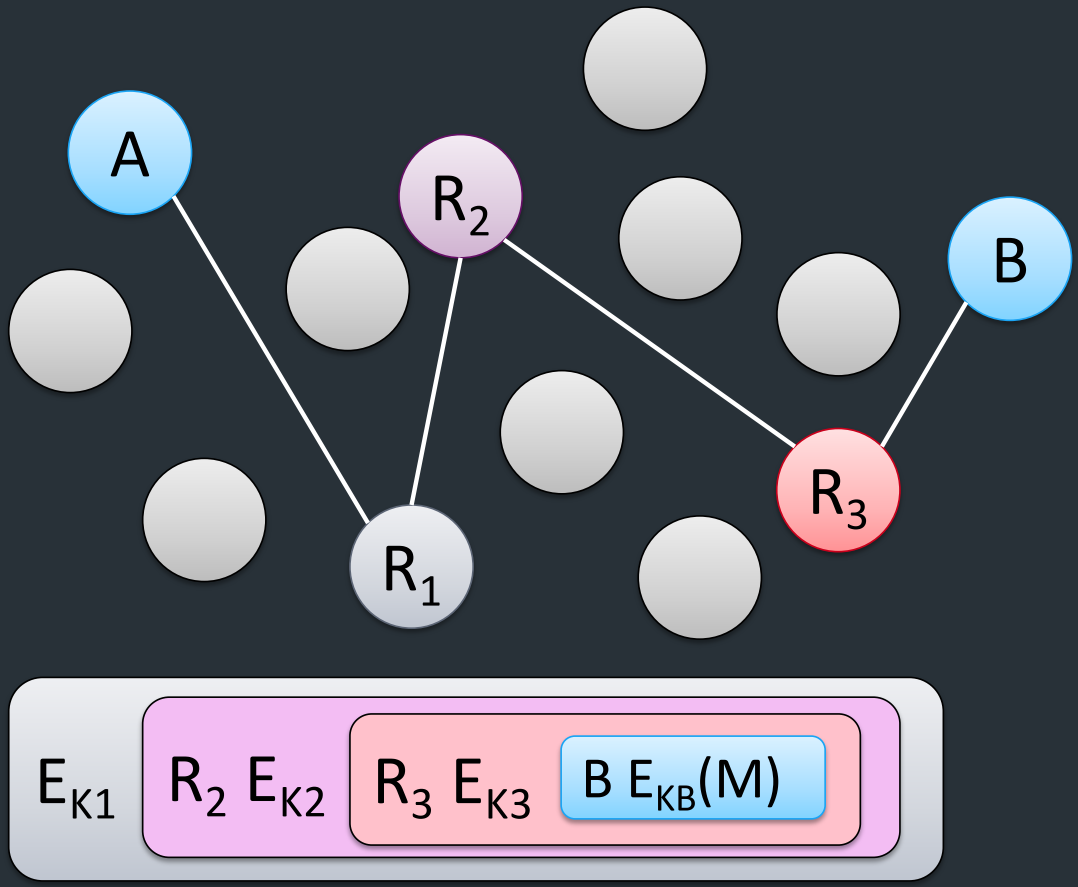
- Depending on the protocol/messages, may leak information that identifies you (eg. cookies, protocol info that contains your IP address)

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

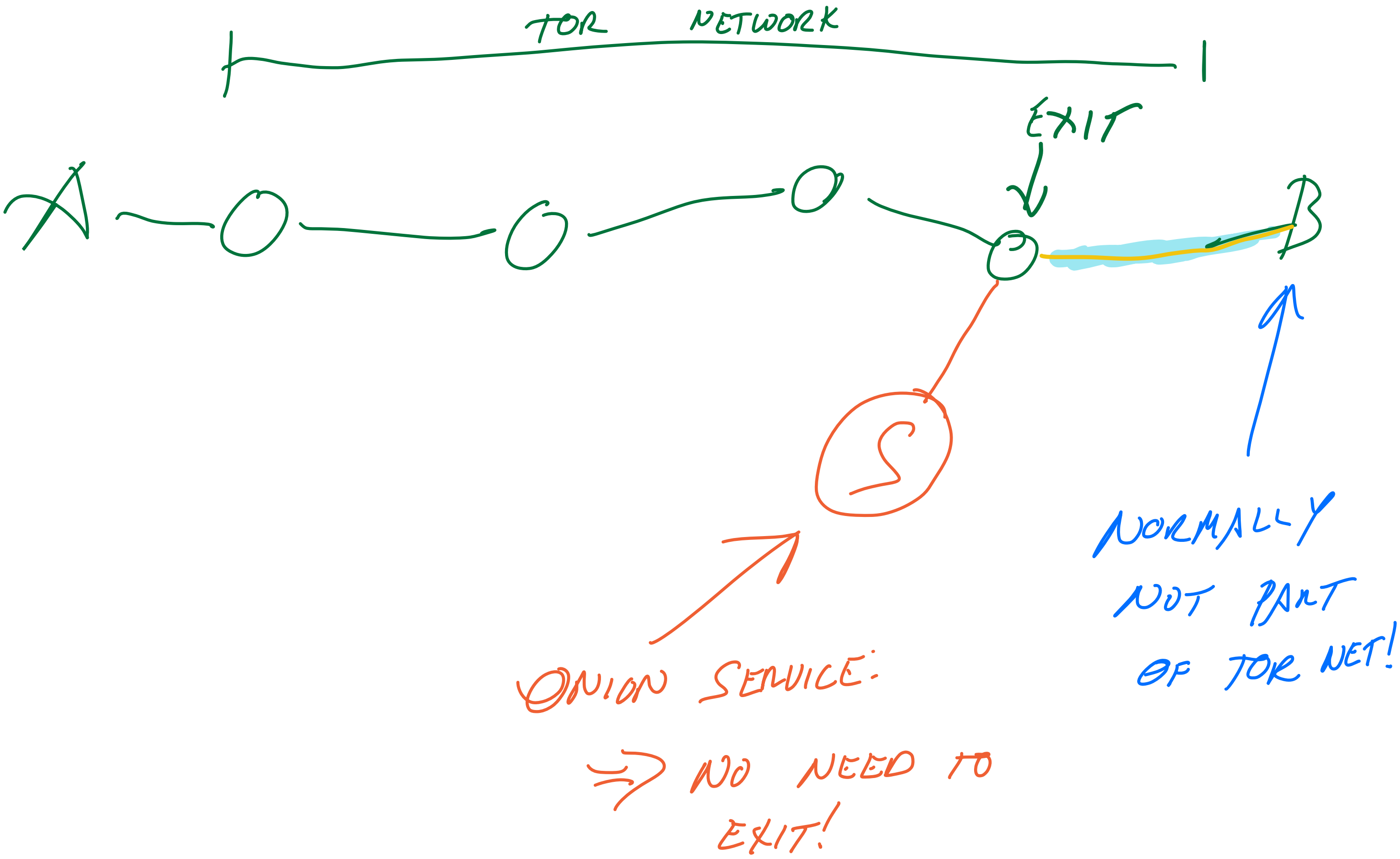
Onion Routing

- Layered encryption
  - Build onion inside out
- Routing
  - Peel onion outside in
- Each router knows only previous and next



Normally: final destination of user's connection (eg. B) is a regular server on the Internet and not aware of tor
 => Traffic is decrypted when it reaches the exit node, send to the public Internet

But what if the server also joins the tor network?
=> Onion services
 => No need for an exit node! Need a "name" for reaching them...



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)

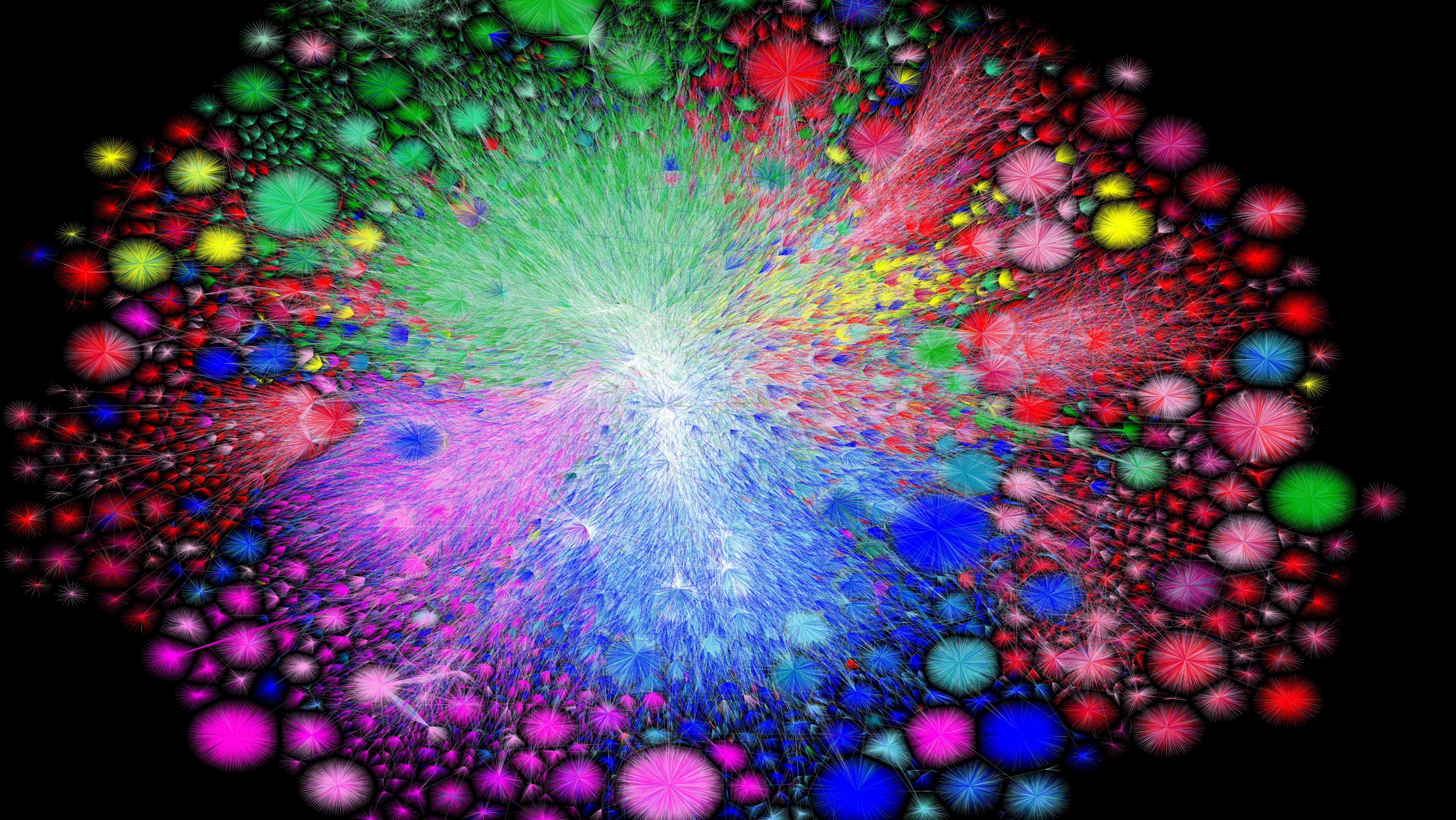
Examples

- *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)
```

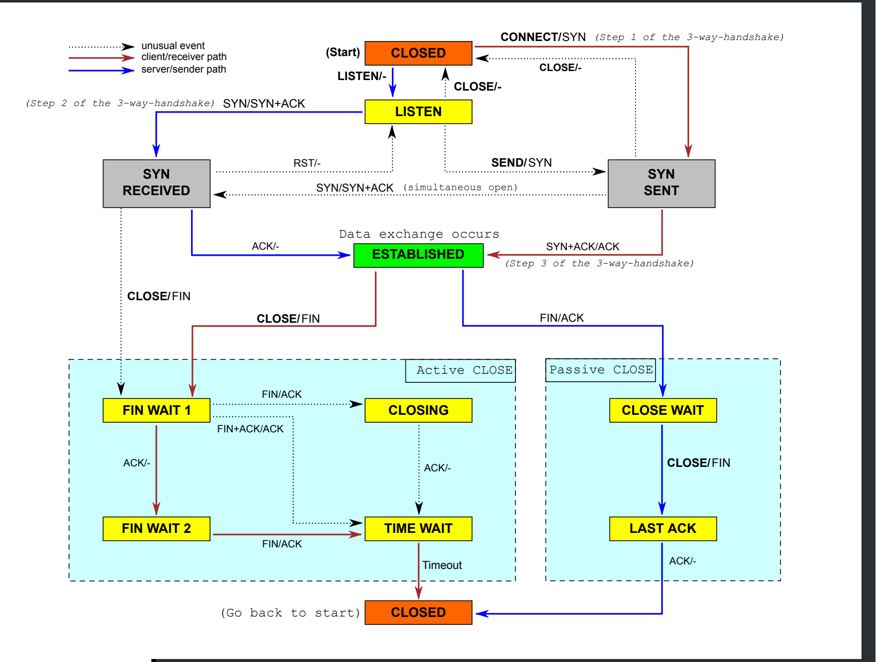
From: draft-ietf-tcpm-rfc793bis-28   Internet Standar

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)

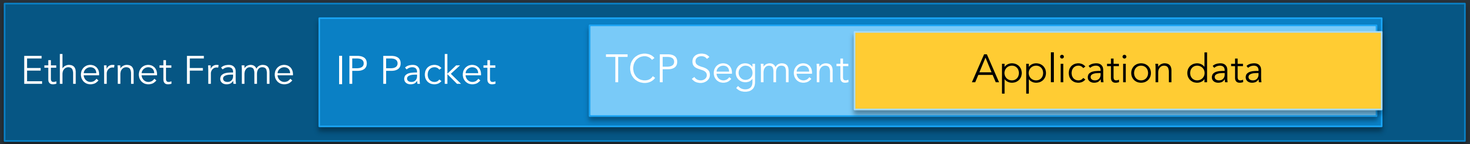
Abstract

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



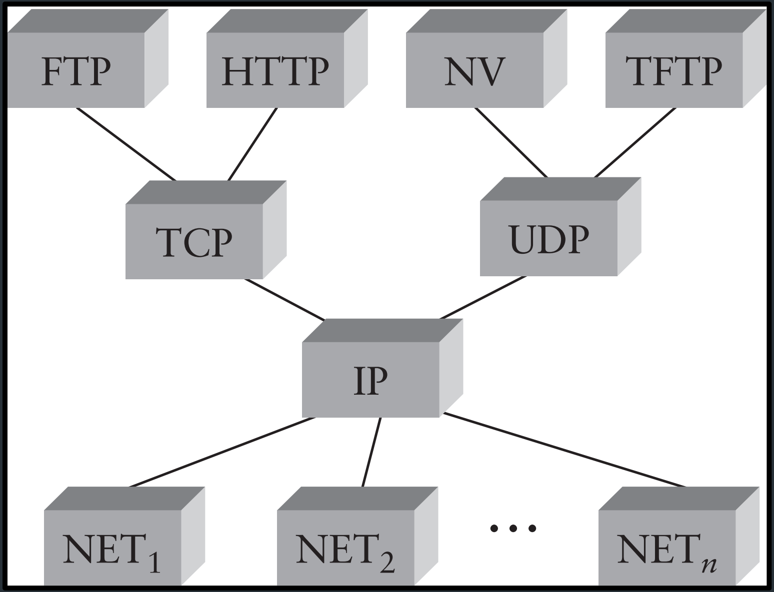
Layering / Encapsulation

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



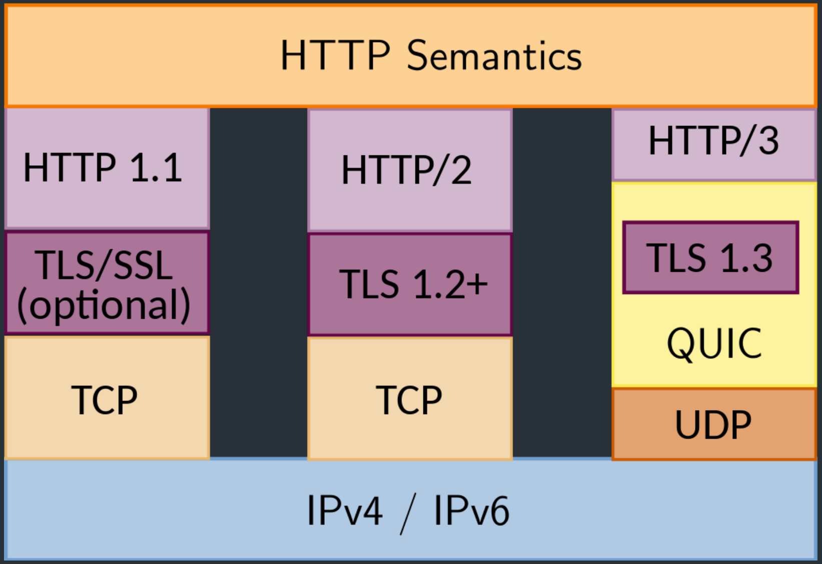
Abstractions are great!

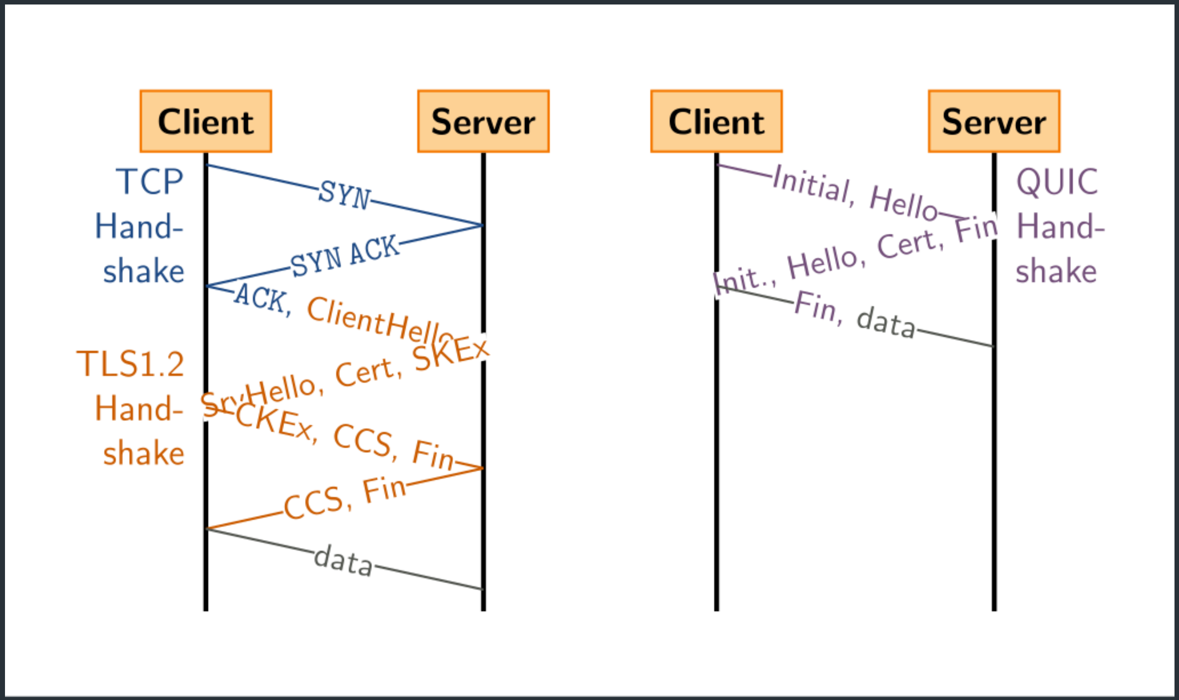
- Can support huge variety of devices, protocols
- Allows independent evolution => new protocols!



… until they aren't

Sometimes, need to break them





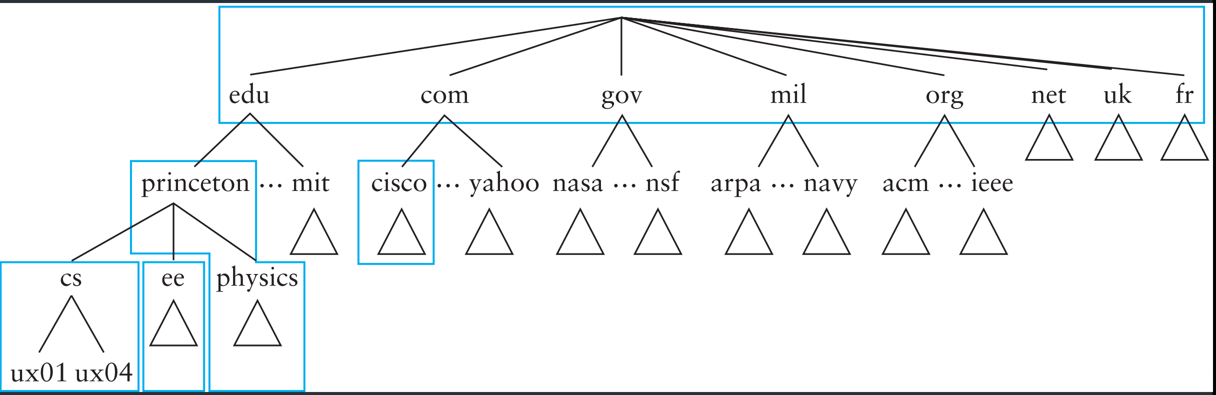
Naming

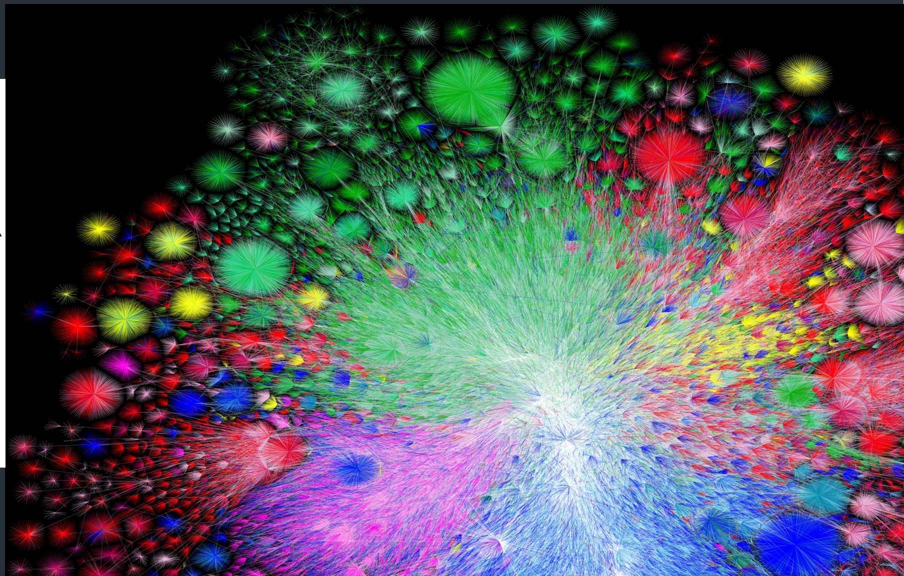
<u>Indirection</u>: abstract low-level info with a higher-level name

=> Human-readable DNS names

=> Scalability: redundancy, proxies, load balancing

Can leverage <u>hierarchy of naming</u> => 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

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

Other CS courses that may interest you if you liked 1680:

CS 2680: Grad seminar on networking (Akshay Narayan)

CS 2690: Datacenter and cloud operating systems (Deepti Raghavan)

CS 1675: Designing high-performance network systems

CS 2390: Privacy-Conscious Computer Systems

Thank you!
Please stay in touch!