- Thursday: VP of Engineering @ GrammaTech
- Moodle Grades: will happen by Thursday
- Project 2G: Released **this afternoon**

Digital certificates

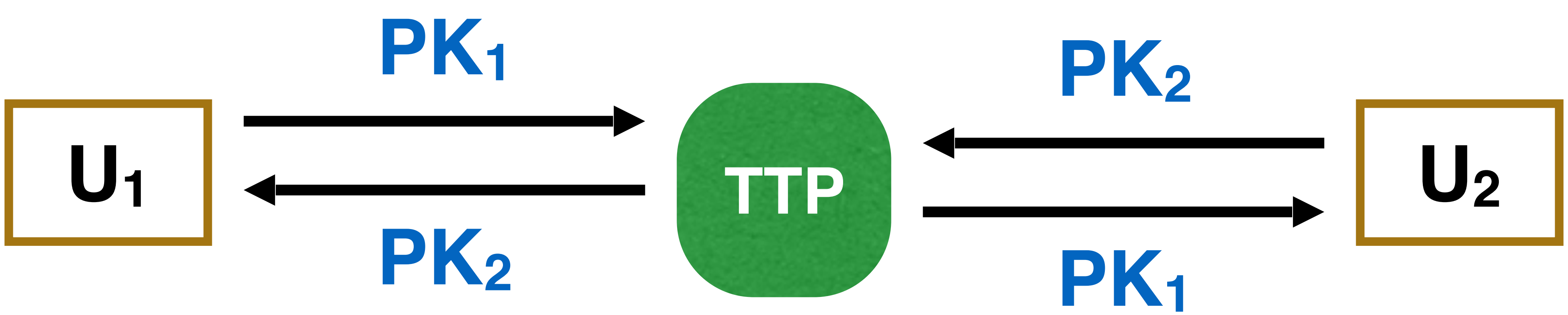
TLS, HTTPS, Revocation

https://s-media-cache-ak0.pinimg.com/originals/ed/af/55/edaf5554d92824ef3555d8b9fbff60c5.jpg

- For convenience, we will use PK_A and SK_A to denote public and secret keys for Alice

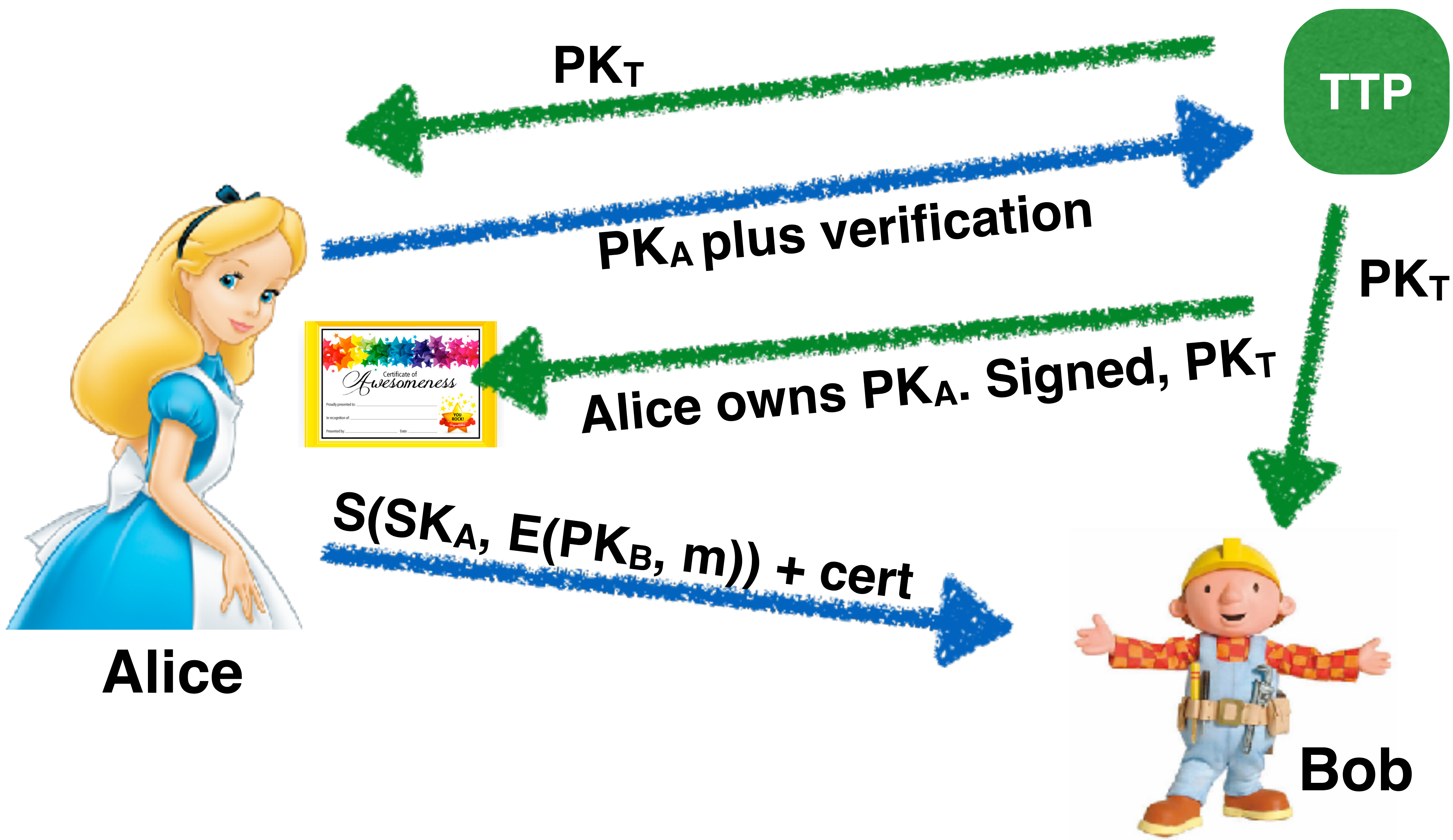
Trusted third party, revisited (1)

Trusted directory service



- TTP is a bottleneck for every conversation
- TTP must be online to start a new conversation
- ~~TTP can read every message~~
- TTP must be trusted to tell the truth!
- Does not solve bootstrapping problem

Trusted 3rd party, revisited (2)



Bob: Verify cert with PKT, verify message with PKA

Some Observations

- Doesn't that mean you're **always** talking to CA?
  - **That's a ton of communication with them!!**
- No! Turns out your browser ***caches*** public keys from CAs

Some Observations

- Let's say foo.com has a valid cert, why should I trust that they're really foo.com, and not some random person with a valid cert?
- Certificate **specifies the domain**
- *Careful*: Worry about things like DNS attacks!
  - We'll cover some, e.g., **cache poisoning**

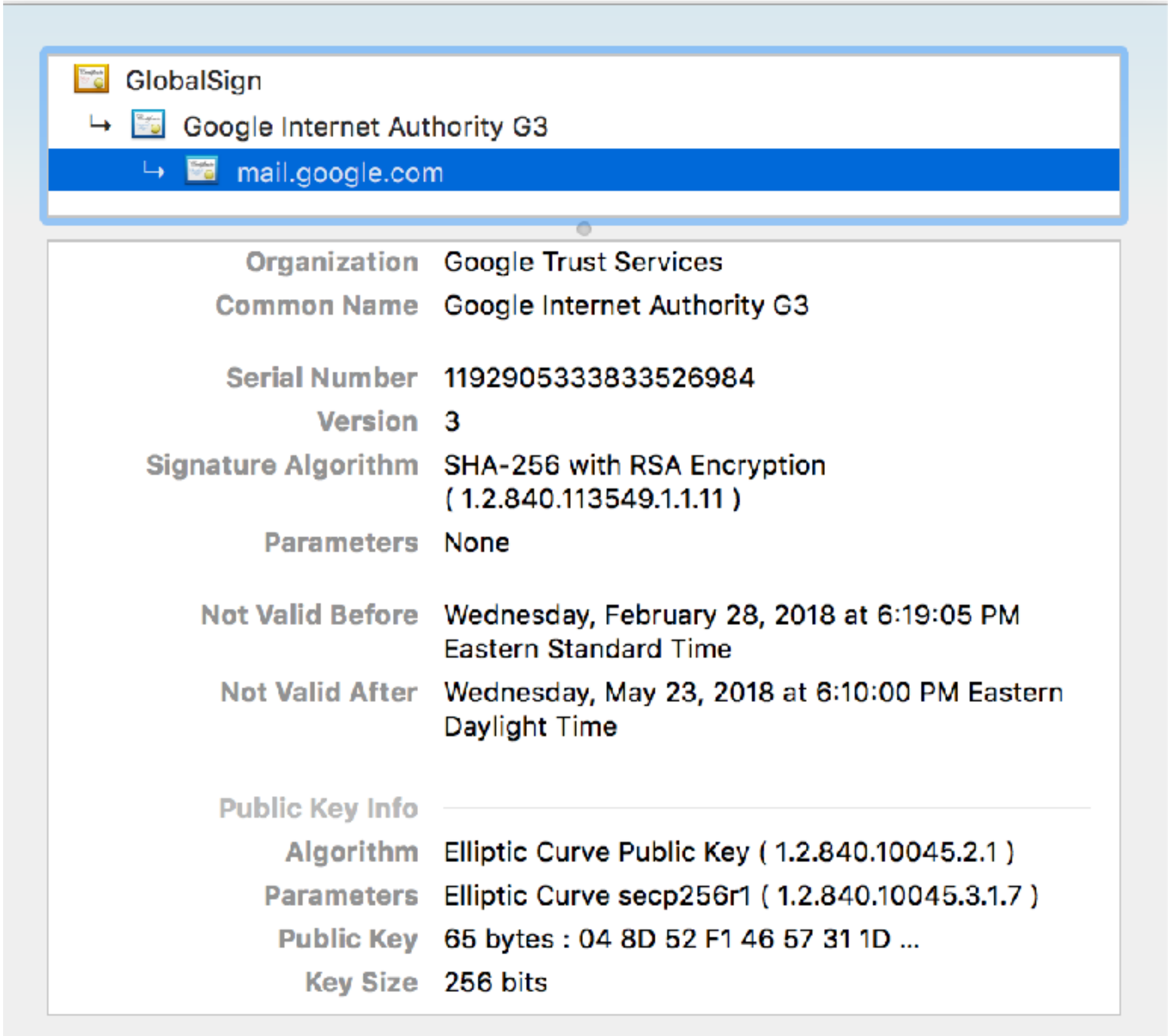
With certificates

- ~~TTP is a bottleneck for every conversation~~
- ~~TTP must be online to start a new conversation~~
- ~~TTP can read every message~~
- **TTP must be trusted to tell the truth!**
- Does not solve bootstrapping problem

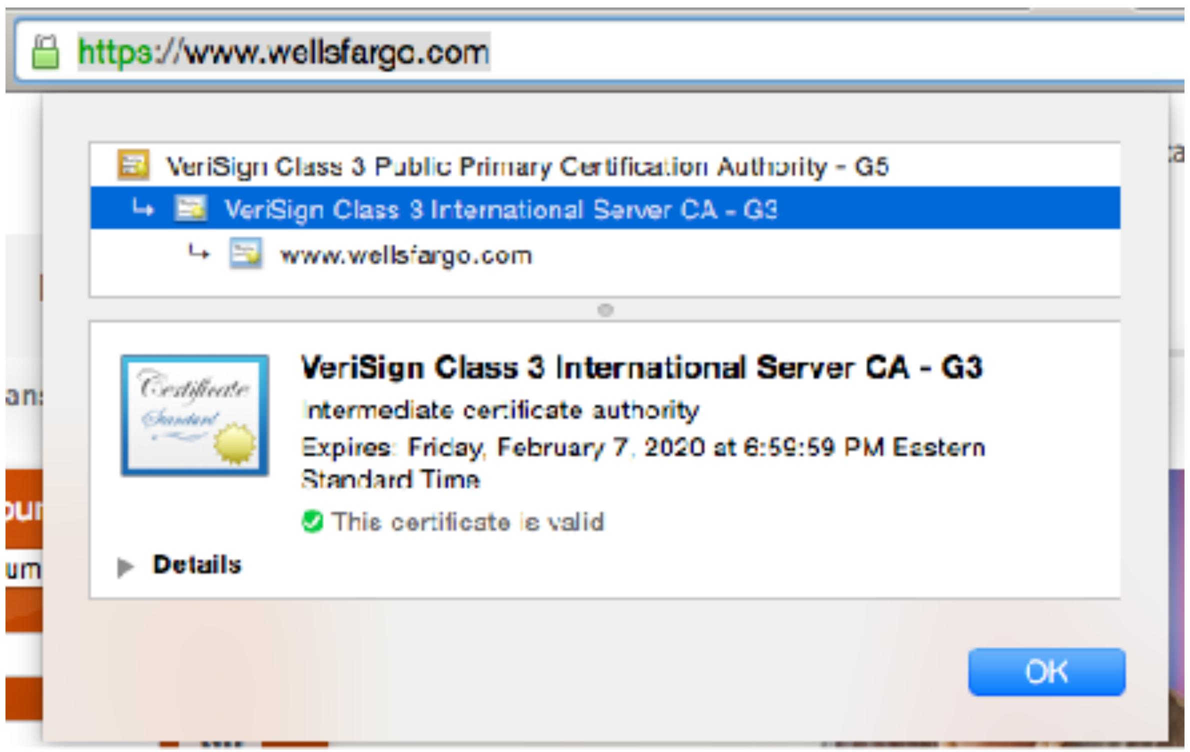
Security is **moot** if Symantec gives a cert for foo.com to evil.com

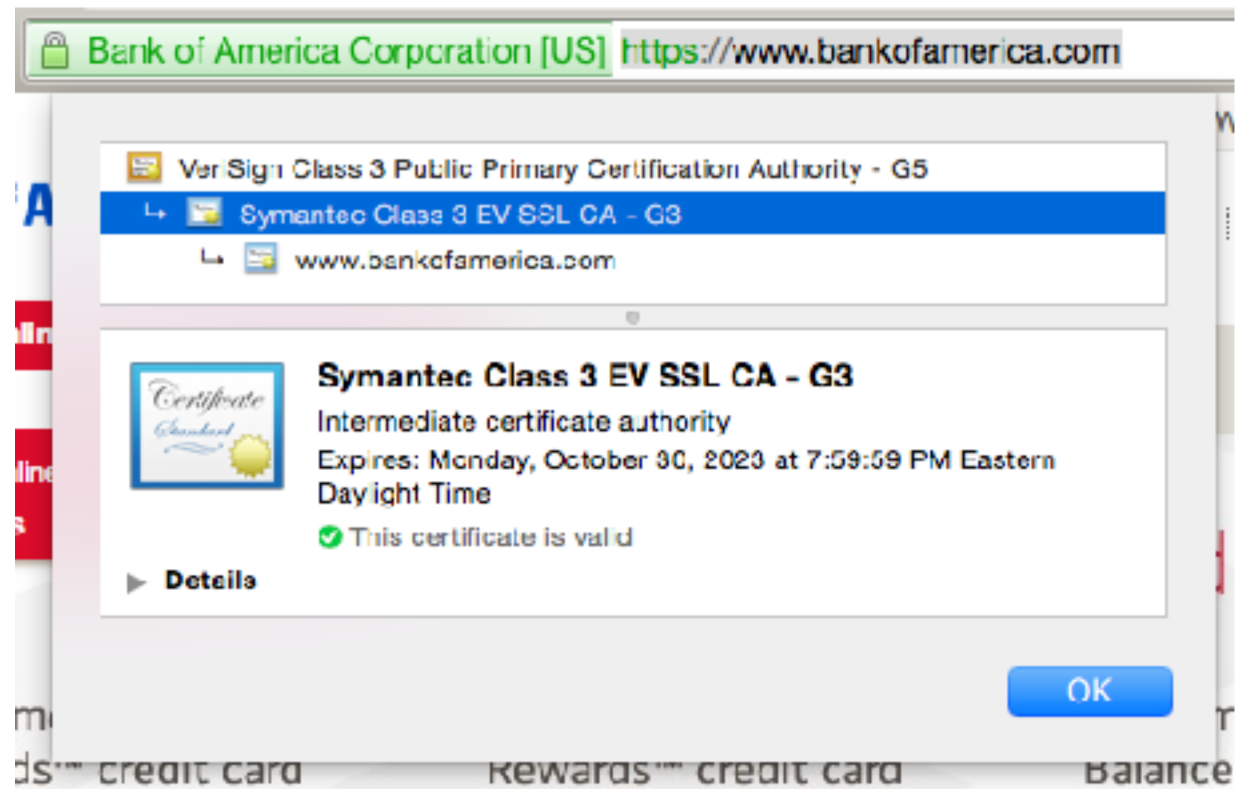
Certificates in practice

- TTP = Certificate Authority
  - Verisign, Comodo, Thawt, etc.
- Alice = web server
- Bob = user who visits alice.com
  - Validate talking to the real alice.com
  - Set up encrypted session for HTTPS
- This is a ***hierarchical*** public key infrastructure (PKI)


GlobalSign
Google Internet Authority G3
mail.google.com
Organization Google Trust Services
Common Name Google Internet Authority G3
Serial Number 1192905333833526984
Version 3
Signature Algorithm SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
Parameters None
Not Valid Before Wednesday, February 28, 2018 at 6:19:05 PM Eastern Standard Time
Not Valid After Wednesday, May 23, 2018 at 6:10:00 PM Eastern Daylight Time
Public Key Info
Algorithm Elliptic Curve Public Key (1.2.840.10045.2.1)
Parameters Elliptic Curve secp256r1 (1.2.840.10045.3.1.7)
Public Key 65 bytes : 04 8D 52 F1 46 57 31 1D ...
Key Size 256 bits

Certificate types





This is an EV (extended validation) certificate; browsers show the full name for these kinds of certs

EV cert = legal vetting process

Where do CAs come from?

- CA public keys shipped with browsers, OS
  - iOS9 ships with >50 that start with A-C
    - see here for full list

Networking Intro

(Most slides generously borrowed from Dave Levin)

This time

Starting with

Networking Basics

- A whirlwind tour of networking
- What is a protocol?
- What are the abstractions / mental models?
- Network stack

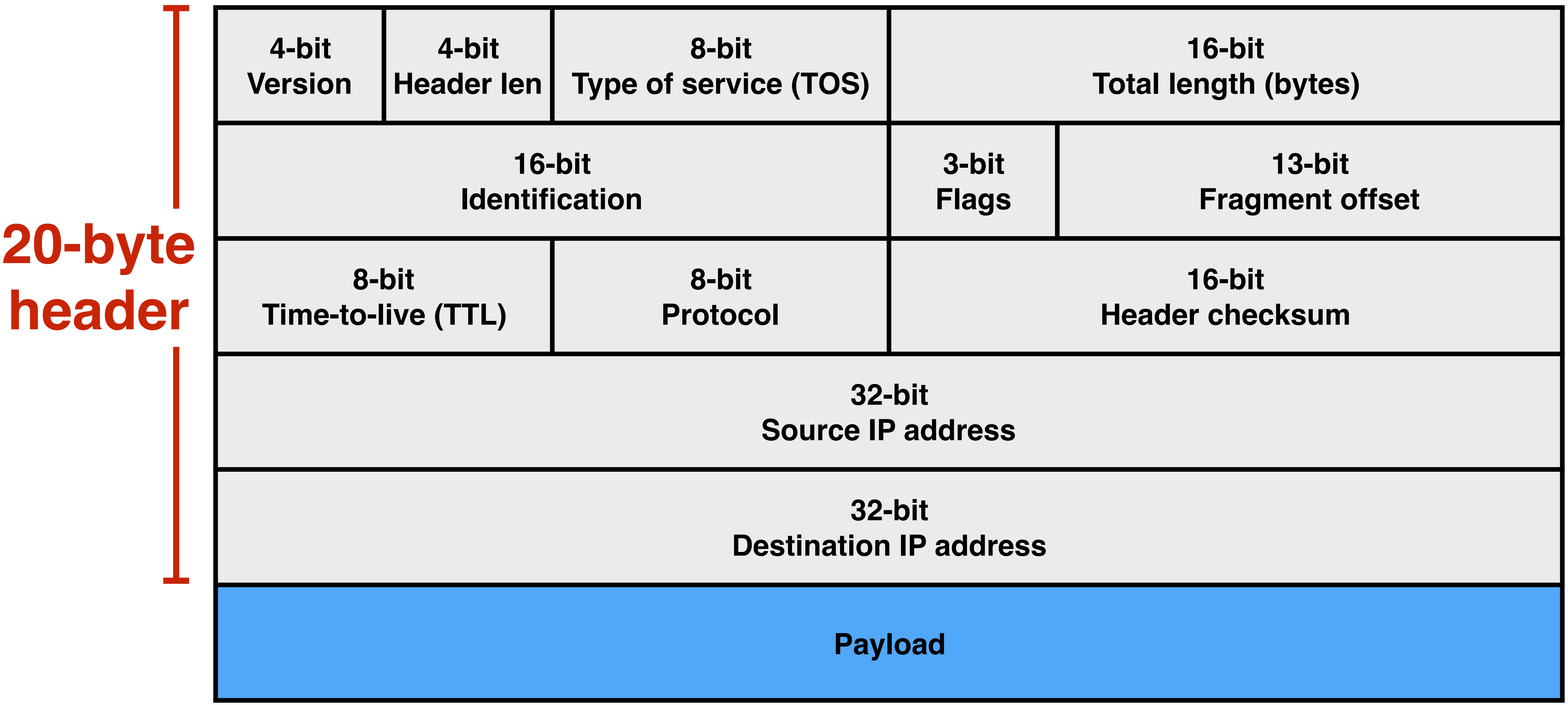
(1) Protocols

Agreement on how to communicate

- Syntax:
  - How the communication is specified and structured
  - Format, order of messages
- Semantics:
  - What the communication means
  - Actions that should be taken when transmitting, receiving, or when a timer expires.

An algorithm for communicating.
***And* a "language" to speak.**

IP packet “header”



The payload is the “data” that IP is delivering:

May contain another protocol’s header & payload, and so on

(2) The network is “dumb”

- End-hosts are on the periphery of the network
  - They can *connect* to one another, even though they are *not physically connected* to one another

- Routers are the interior nodes that
  - “Route”: *determine how* to get to B
  - “Forward”: actually forward traffic from A to B

- Principle: the routers have no knowledge of ongoing connections through them
  - They do “destination-based” routing and forwarding
    - Given the destination in the packet, send it to the “next hop” that is best suited to help ultimately get the packet there

Mental model: The postal system

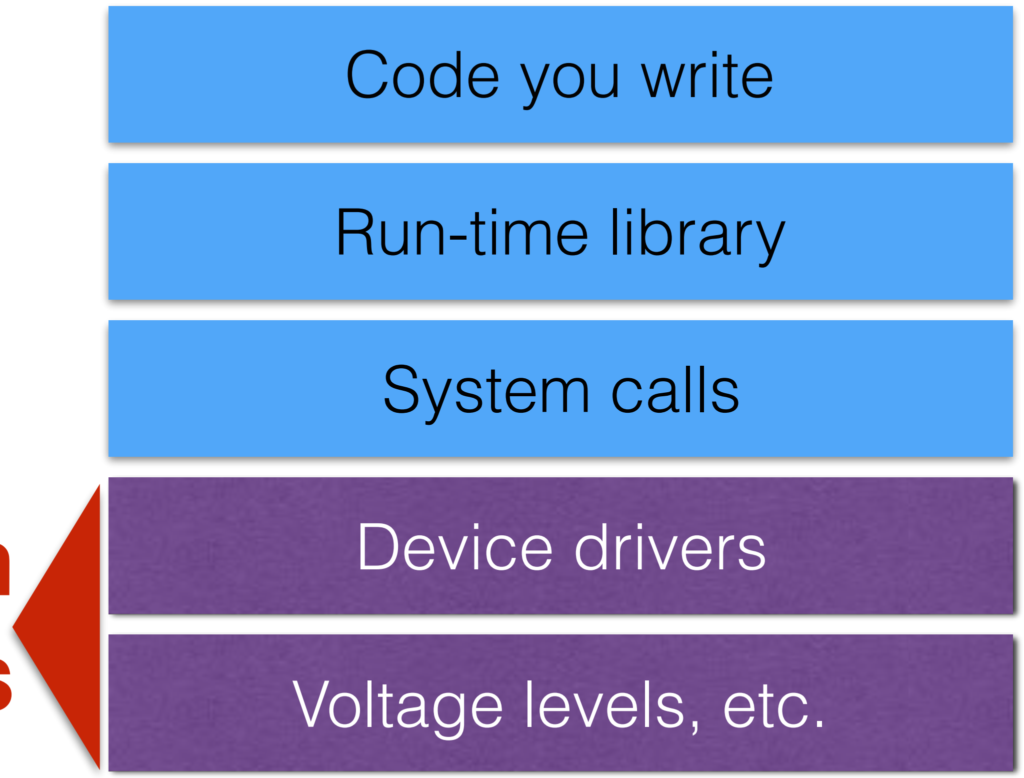
Postal system analogy

- Messages are self-contained
  - Post: a message in an envelope
  - Internet: data in a packet
- Interior routers forward based on destination address
  - Post: zip code, then street, then building, then apartment number (then the right individual)
  - Internet: progressively smaller blocks of IP addresses, then your computer (then the right application)
- Simple, robust.
  - More sophisticated things go at the *ends of the network*

(3) Layers

- The design of the Internet is strongly partitioned into layers
  - Each layer relies on the services provided by the layer immediately below it…
  - … and provides service to the layer immediately above it

Analogy:

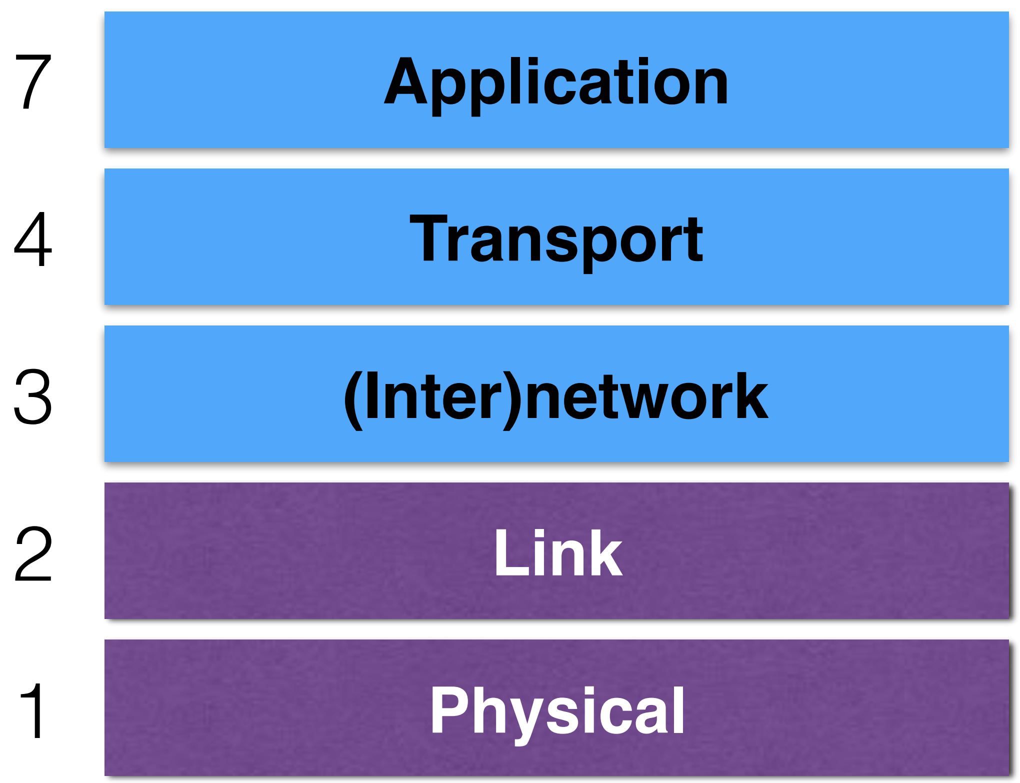


Isolated from user programs

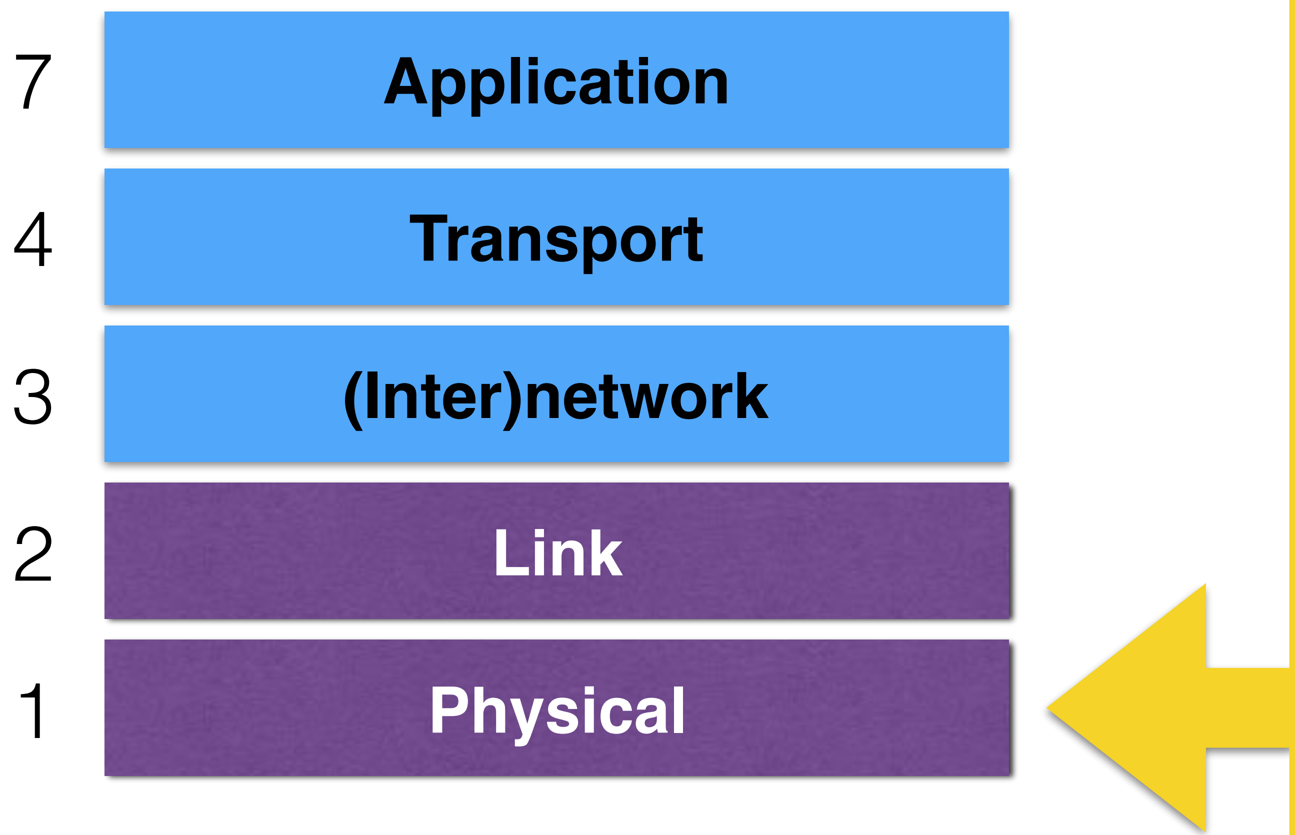
Each layer has a well-defined *role* that builds off of the layer below it

Between each layer is a well-defined *interface*

Internet layering = “Protocol stack”



Layer 1: Physical layer

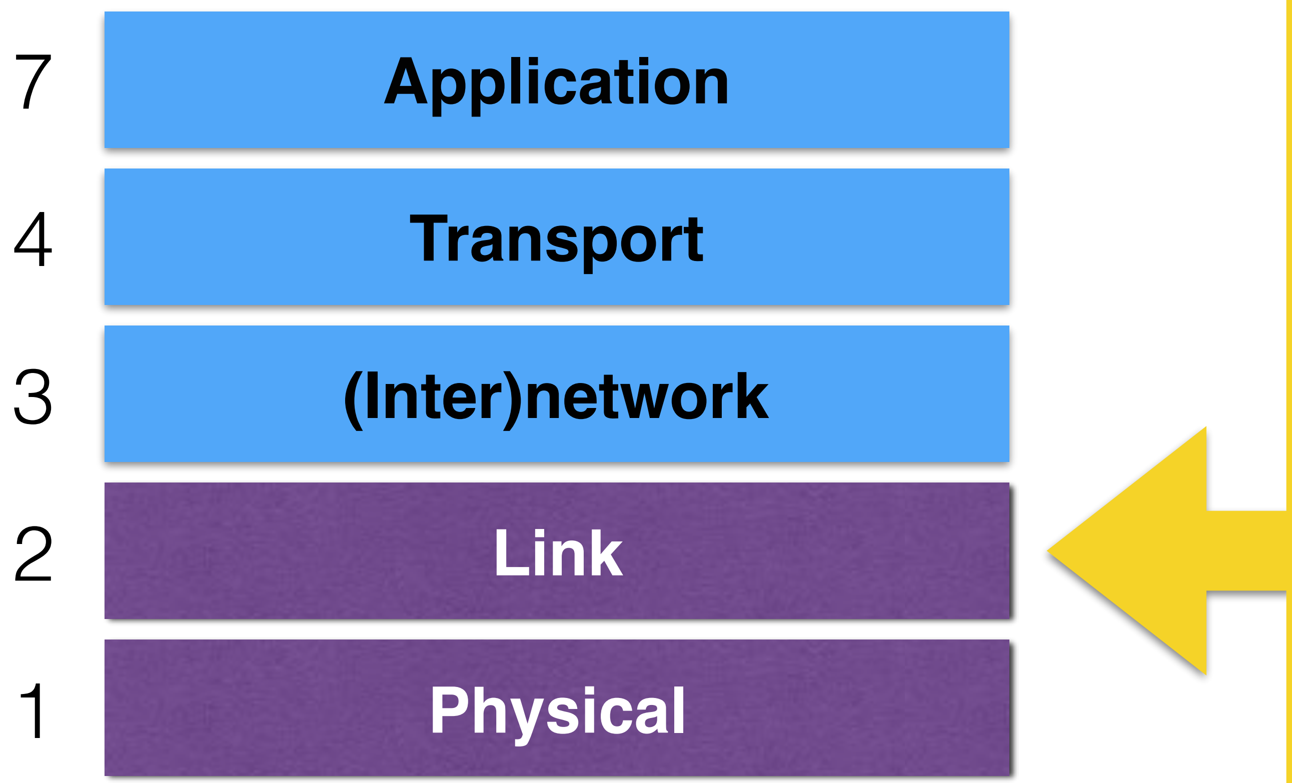


- Encoding of bits to send over a ***single*** physical link

- Examples:
  - Voltage levels
  - RF modulation
  - Photon intensities

Physical layer:
transmitting a single bit
over a physical link
(though not necessarily *wired* link)

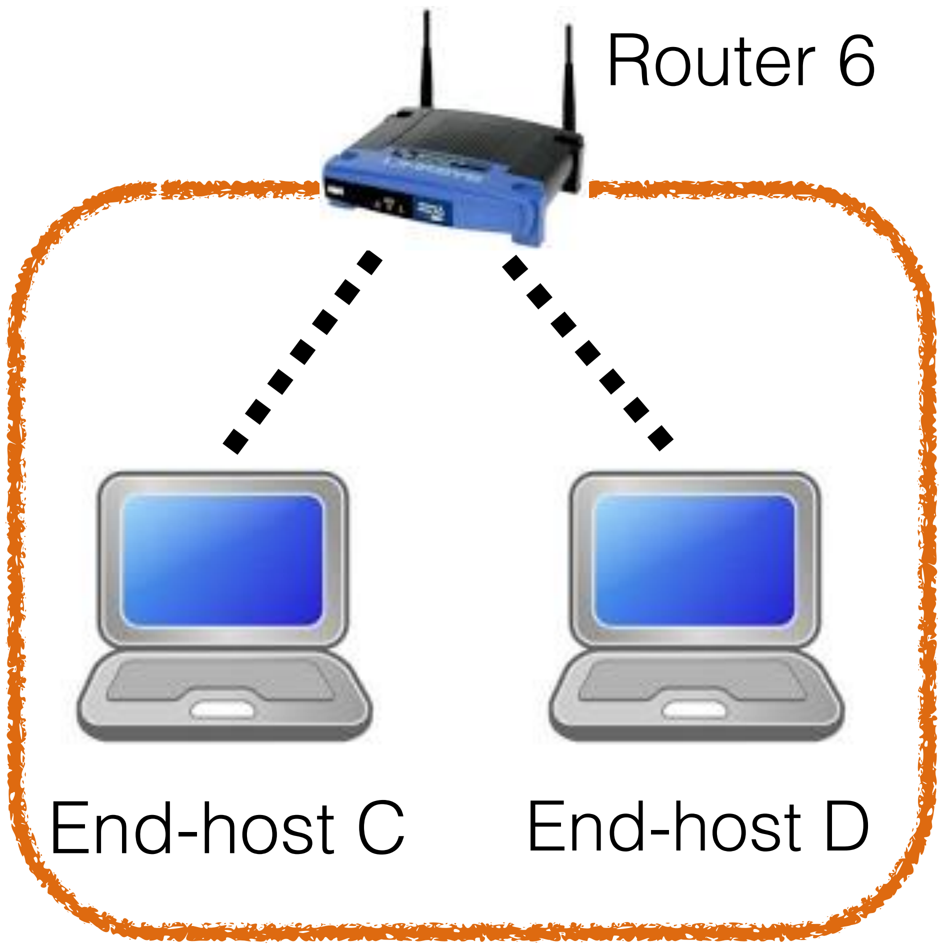
Layer 2: Link layer



- Framing and transmission of a collection of bits into individual **messages** sent across a single subnetwork (one physical topology)
- Provides **local** addressing (MAC)
- May involve multiple *physical links*
- Often the technology supports **broadcast**: every "node" connected to the subnet receives
- Examples:
  - Modern Ethernet
  - WiFi (802.11a/b/g/n/etc)

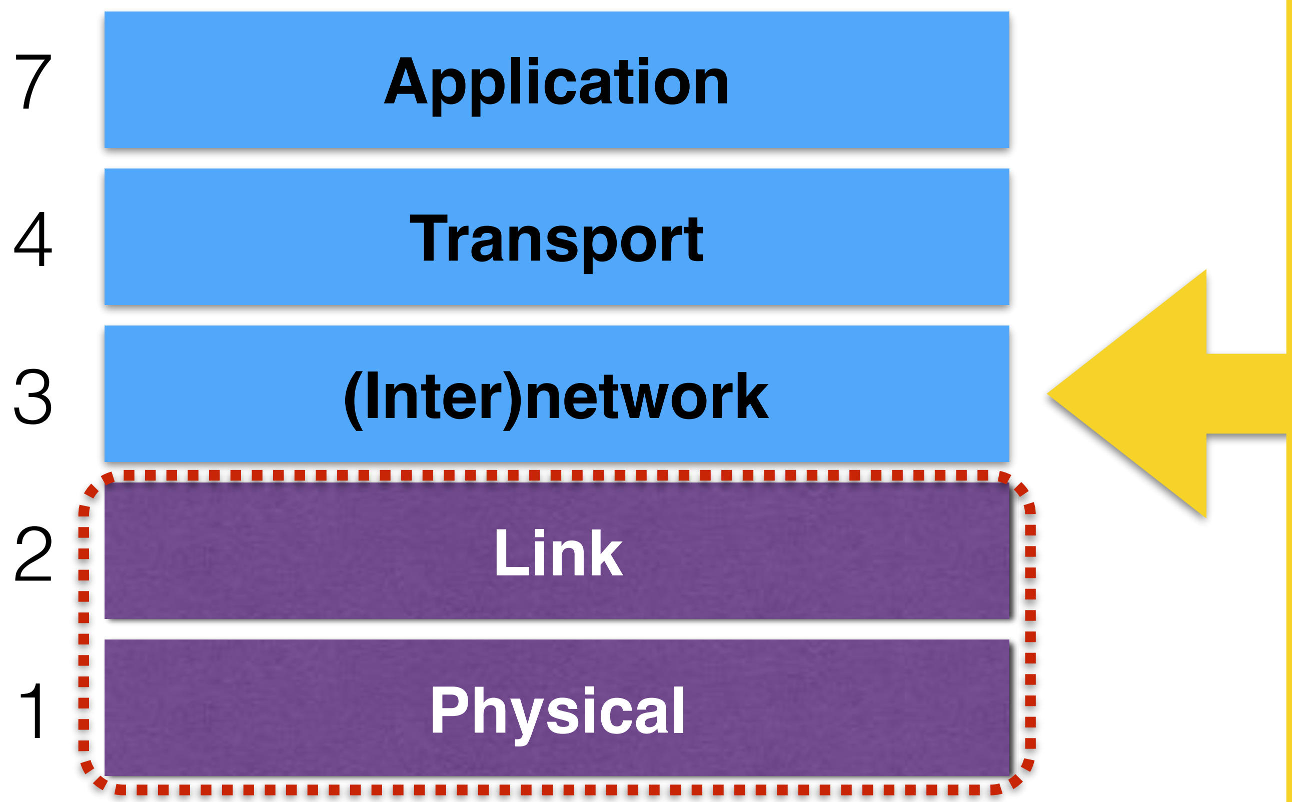
Link layer

- transmitting messages
- over a *subnet*
- src/dst identified by globally unique **MAC addrs**



Because you need to be able to join any subnet and be uniquely distinguishable

Layer 3: (Inter)network layer



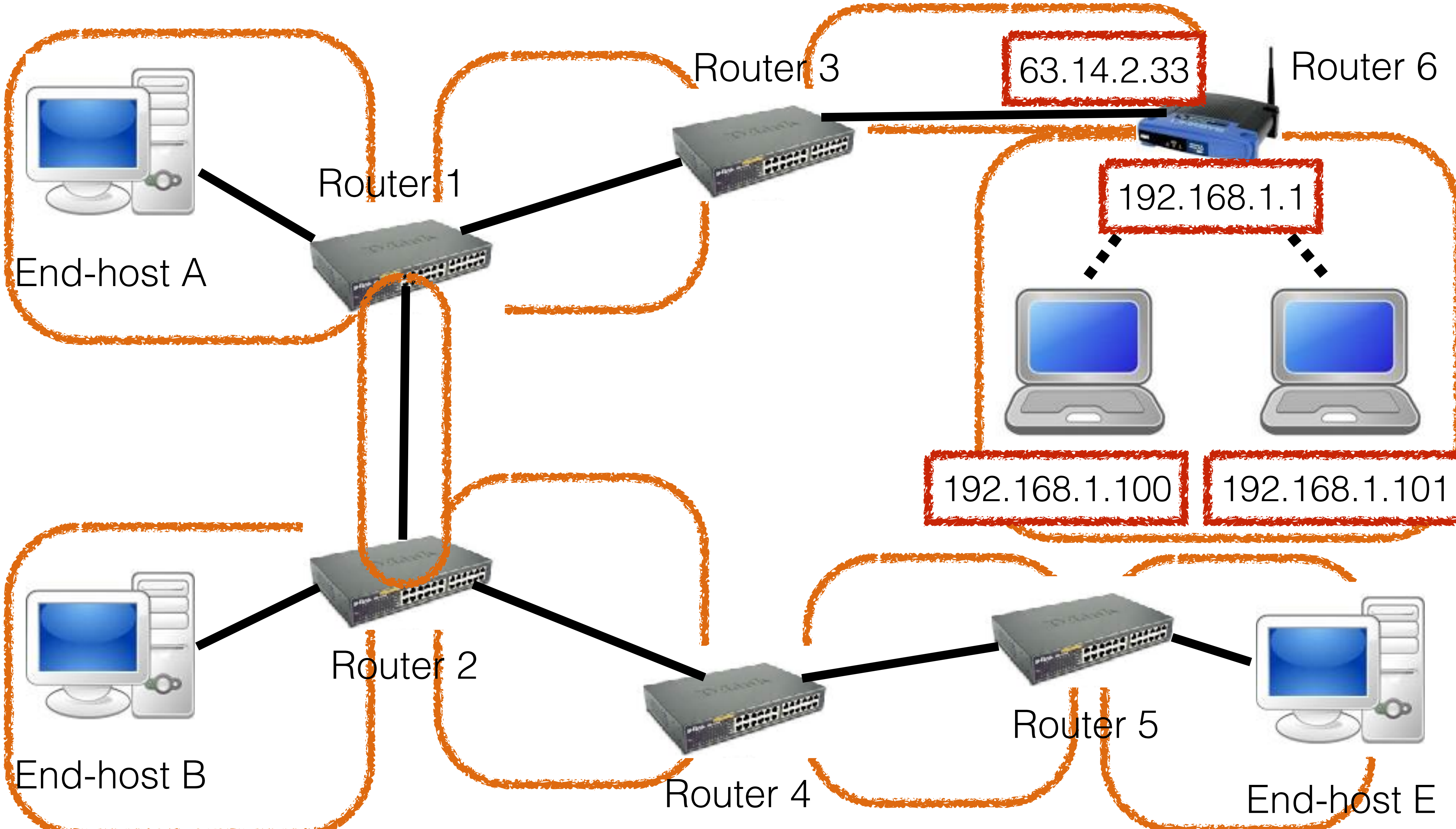
Different for each Internet "hop"

- Bridges multiple "subnets" to provide *end-to-end* internet connectivity between nodes
- Provides **global** addressing (IP addresses)
- Only provides **best-effort** delivery of data (i.e., no retransmissions, etc.)
- Works across different link technologies

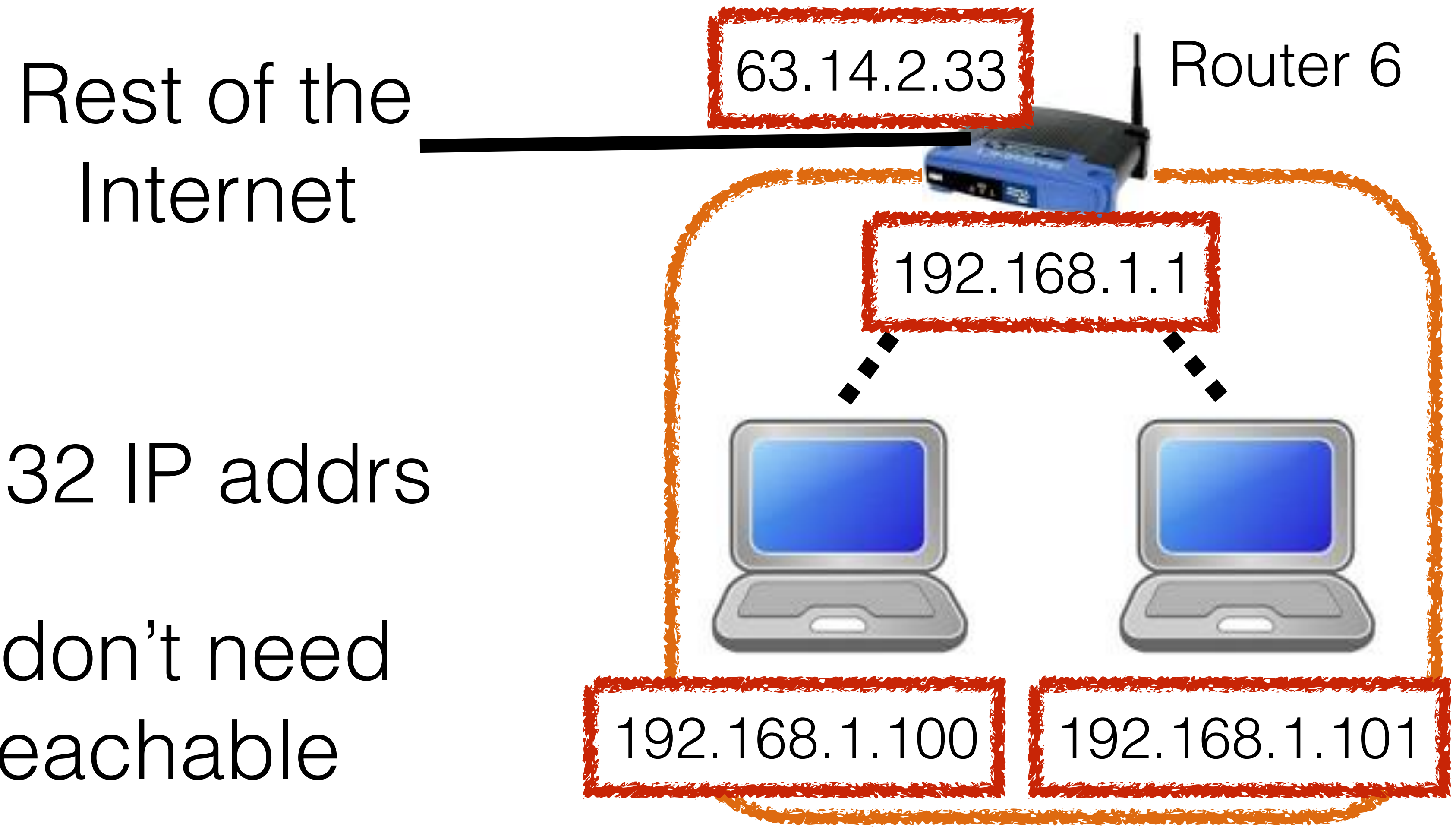
Lowercase-i "internet" = network of networks.
Uppercase-i Internet = "*the* Internet"

Network layer
- transmitting packets
- within or across subnets
- src/dst identified by locally unique **IP addrs**

Routers connect multiple subnets



Local uniqueness is often enough



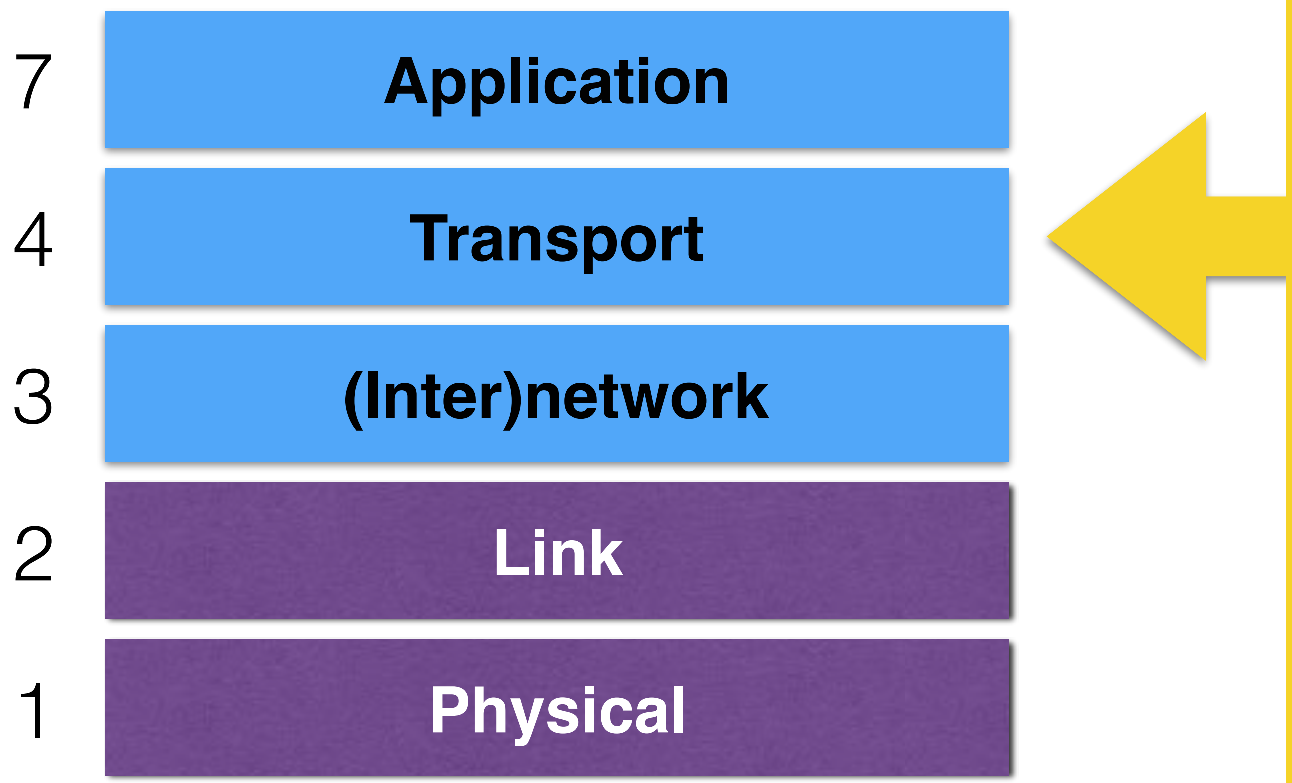
There are only 2^32 IP addrs

Many machines don't need to be publicly reachable

Some addresses are "private" addresses

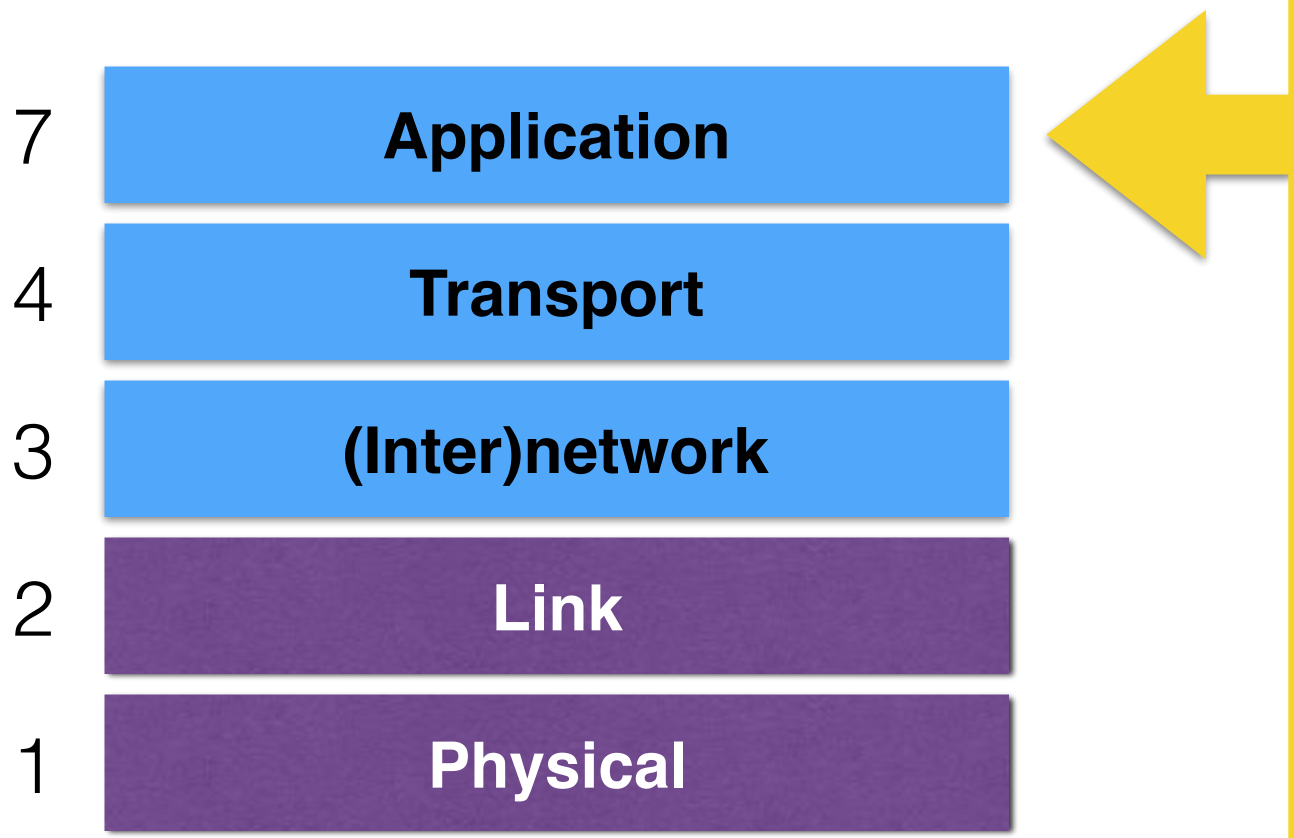
The router performs "**Network Address Translation**": changes outgoing packets' src from 192.168.1.100 to 63.14.2.33, and vice versa for incoming packets

Layer 4: Transport layer



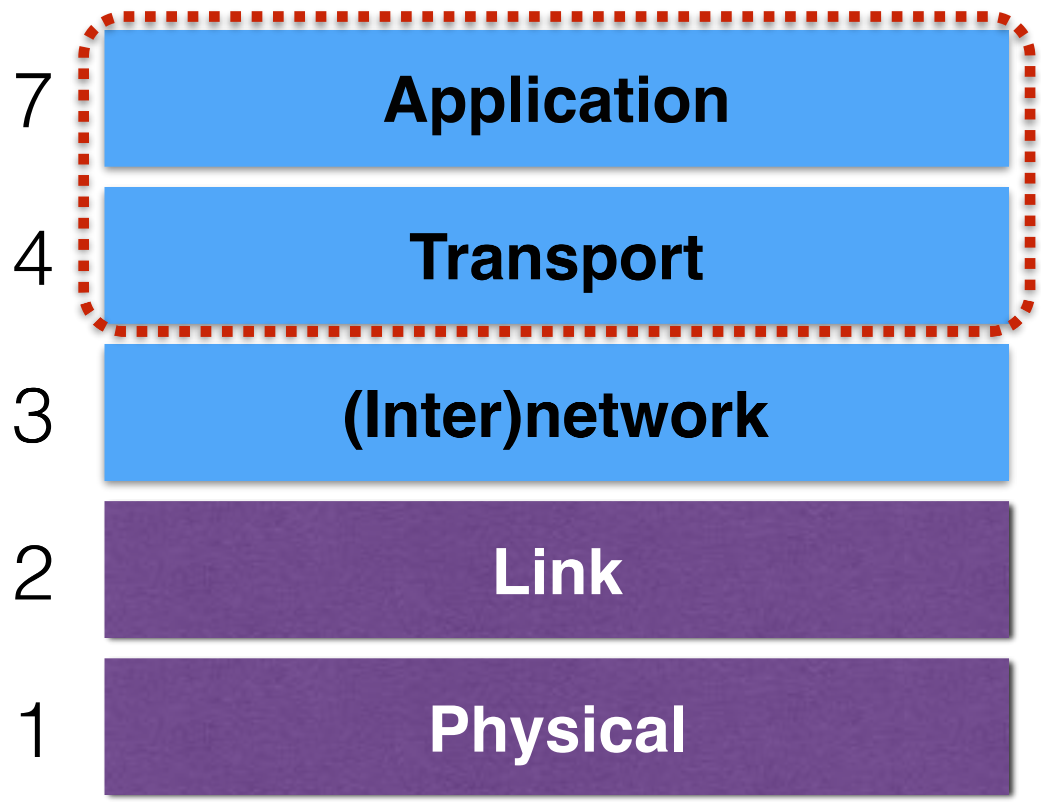
- End-to-end communication between **processes**
- Different types of services provided:
  - UDP: unreliable *datagrams*
  - TCP: *reliable* byte stream
- “Reliable” = keeps track of what data were received properly and retransmits as necessary

Layer 7: Application layer



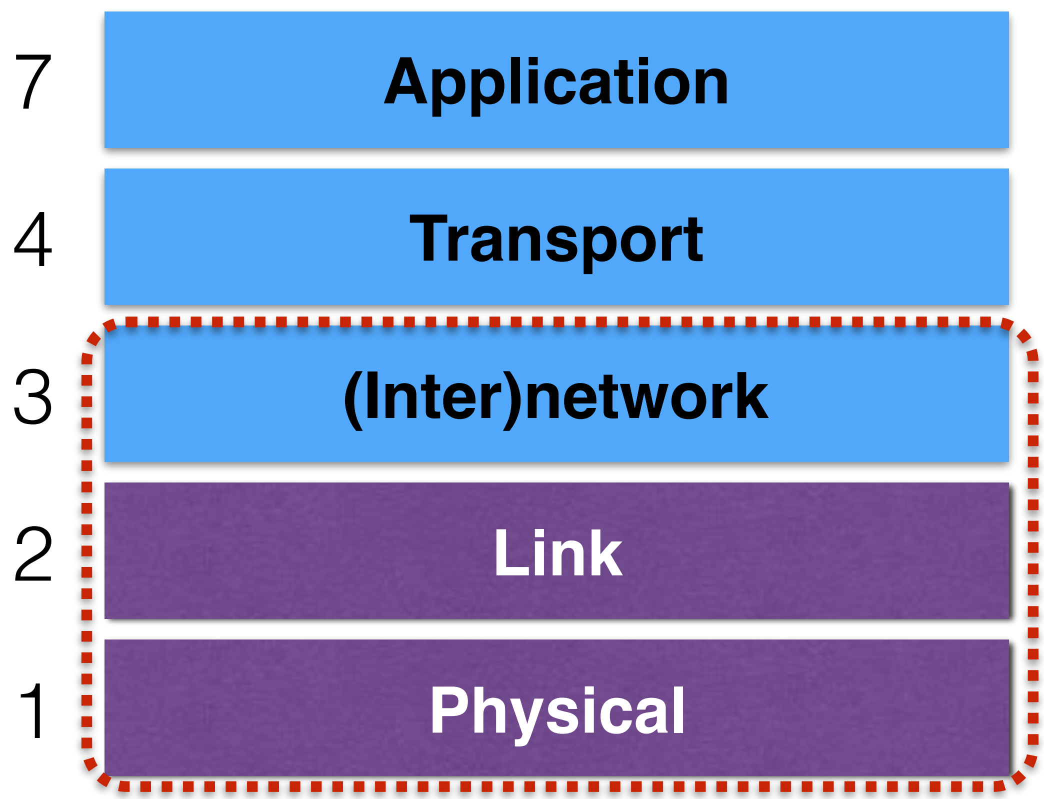
- Communication of whatever you want
- Can use whatever transport(s) is(are) convenient/appropriate
- Freely structured
- Examples:
  - Skype (UDP)
  - SMTP = email (TCP)
  - HTTP = web (TCP)
  - Online games (TCP and/or UDP)

Internet layering = "Protocol stack"



Implemented only at end hosts, not at interior routers (this is our "dumb network")

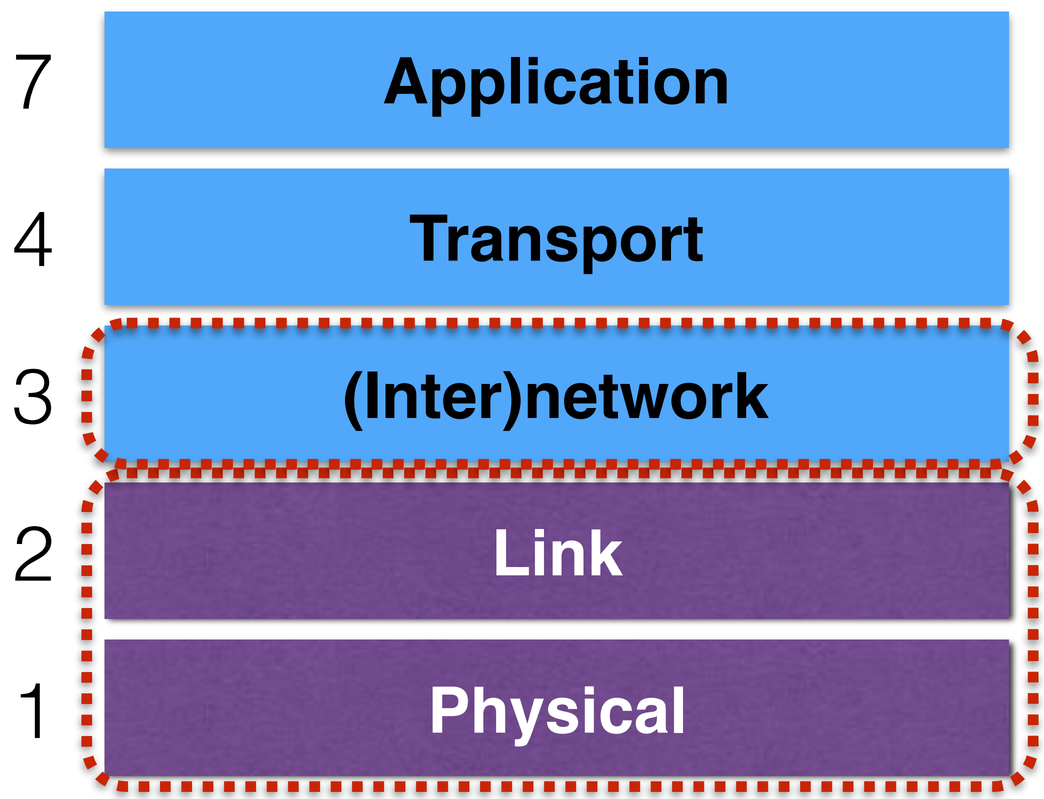
Internet layering = “Protocol stack”



Implemented *everywhere*

The network is “dumb” but it needs to know precisely this much to do its job.

Internet layering = “Protocol stack”

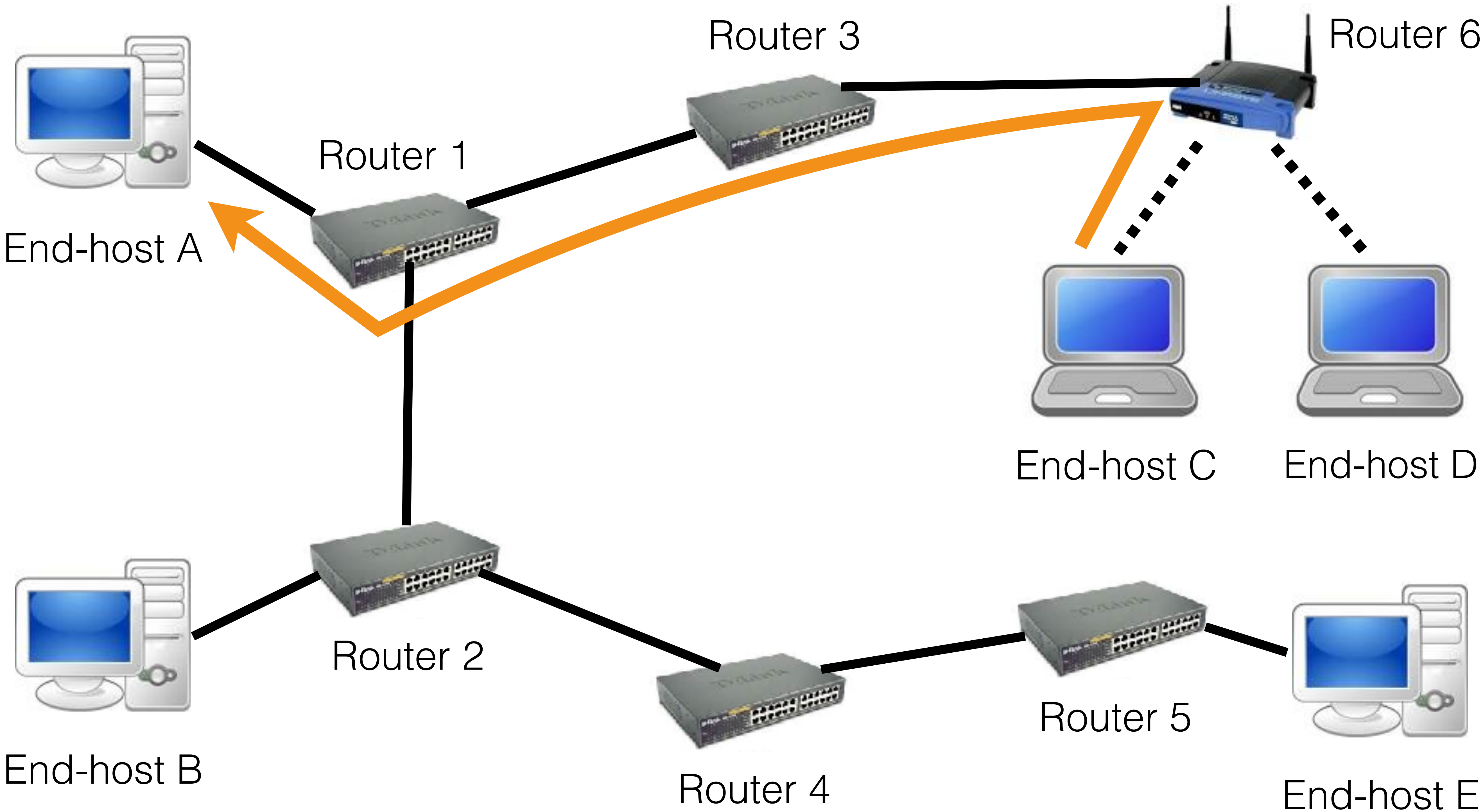


~Same for each Internet “hop”

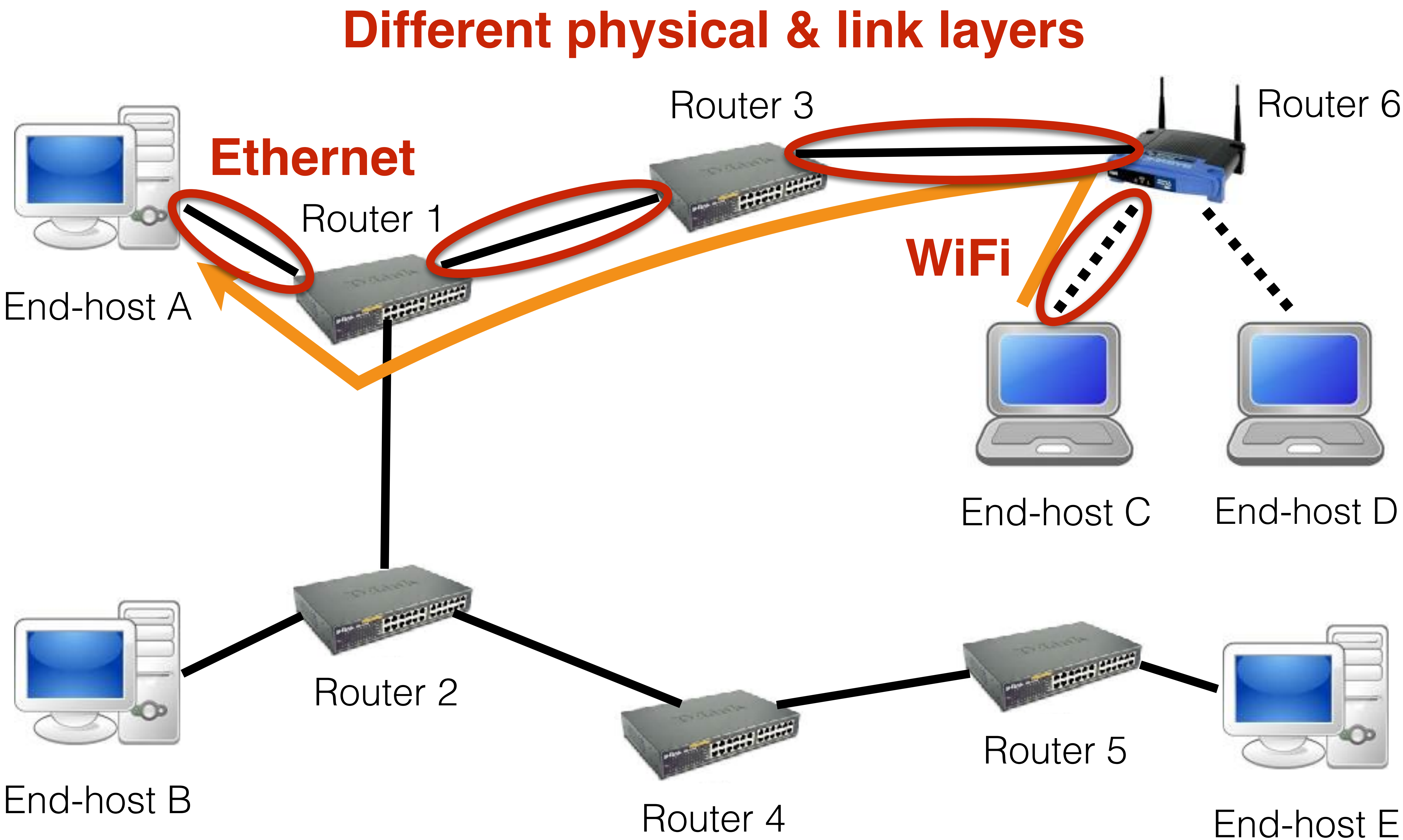
Can be different for each Internet “hop”

Hop-by-hop vs. end-to-end layers

Host C communicates with host A

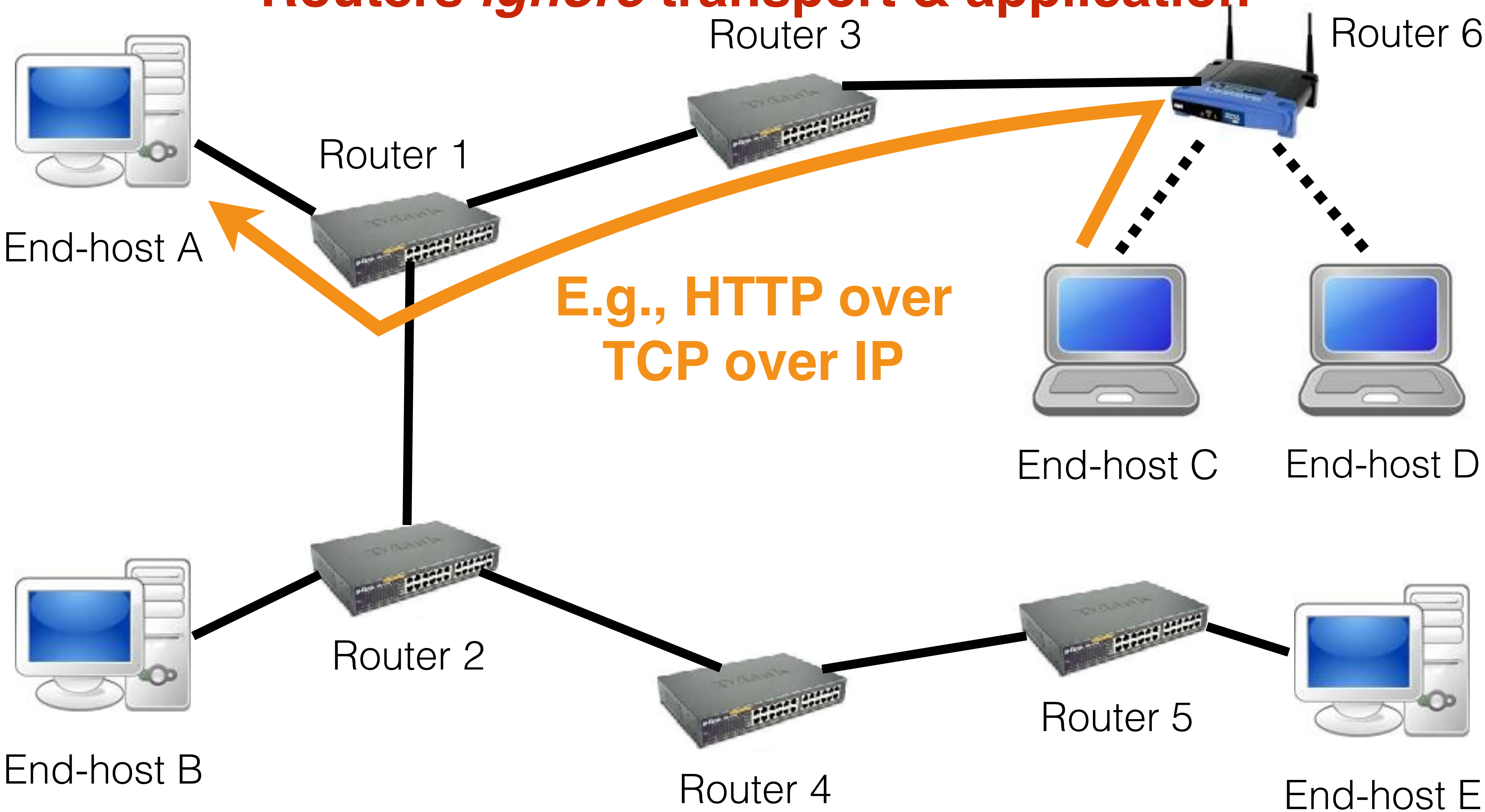


Hop-by-hop vs. end-to-end layers

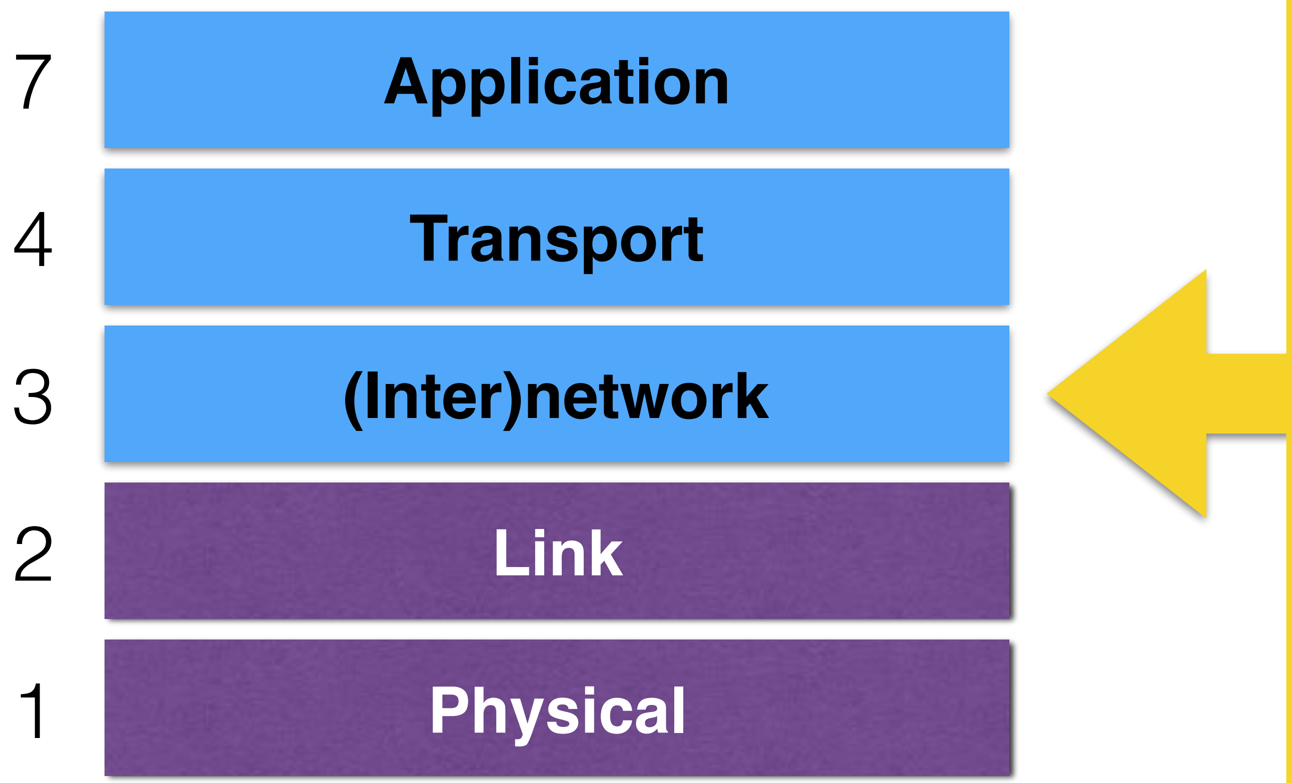


Hop-by-hop vs. end-to-end layers

Same network, transport, and application layers (3/4/7)
Routers *ignore* transport & application

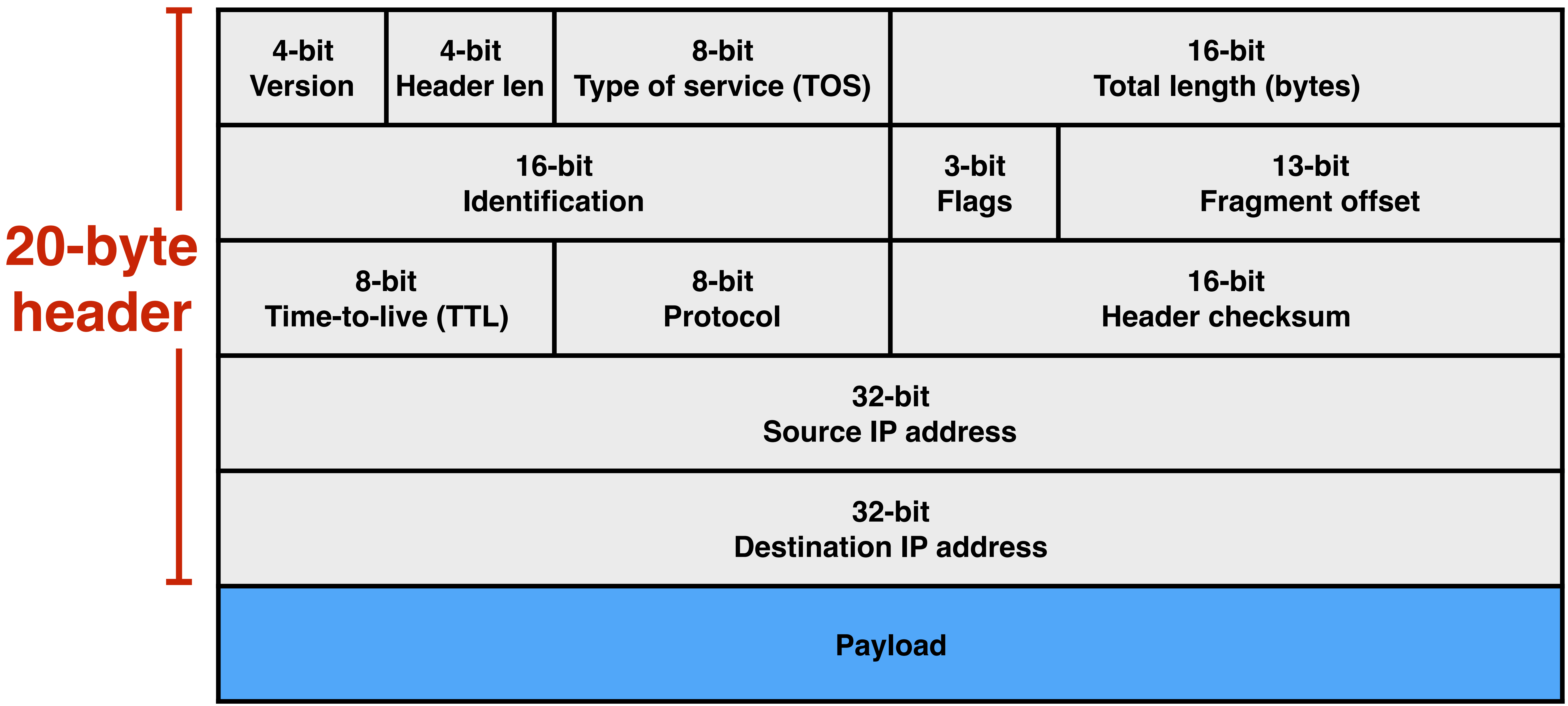


Layer 3: (Inter)network layer



- Bridges multiple “subnets” to provide *end-to-end* internet connectivity between nodes
- Provides **global** addressing (IP addresses)
- Only provides **best-effort** delivery of data (i.e., no retransmissions, etc.)
- Works across different link technologies

IP packet “header”



IP Packet Header Fields (1)

- Version number (4 bits)
  - Indicates the version of the IP protocol
  - Necessary for knowing what fields follow
  - “4” (for IPv4) or “6” (for IPv6)
- Header length (4 bits)
  - How many 32-bit words (rows) in the header
  - Typically 5
  - Can provide IP options, too
- Type-of-service (8 bits)
  - Allow packets to be treated differently based on different needs
  - Low delay for audio, high bandwidth for bulk transfer, etc.

IP Packet Header Fields (2)

- Two IP addresses
  - Source (32 bits)
  - Destination (32 bits)

- Destination address
  - *Unique* identifier/locator for the receiving host
  - Allows each node (end-host and router) to make forwarding decisions

- Source address
  - Unique identifier/locator for the sending host
  - Recipient can decide whether to accept the packet
  - Allows destination to *reply* to the source

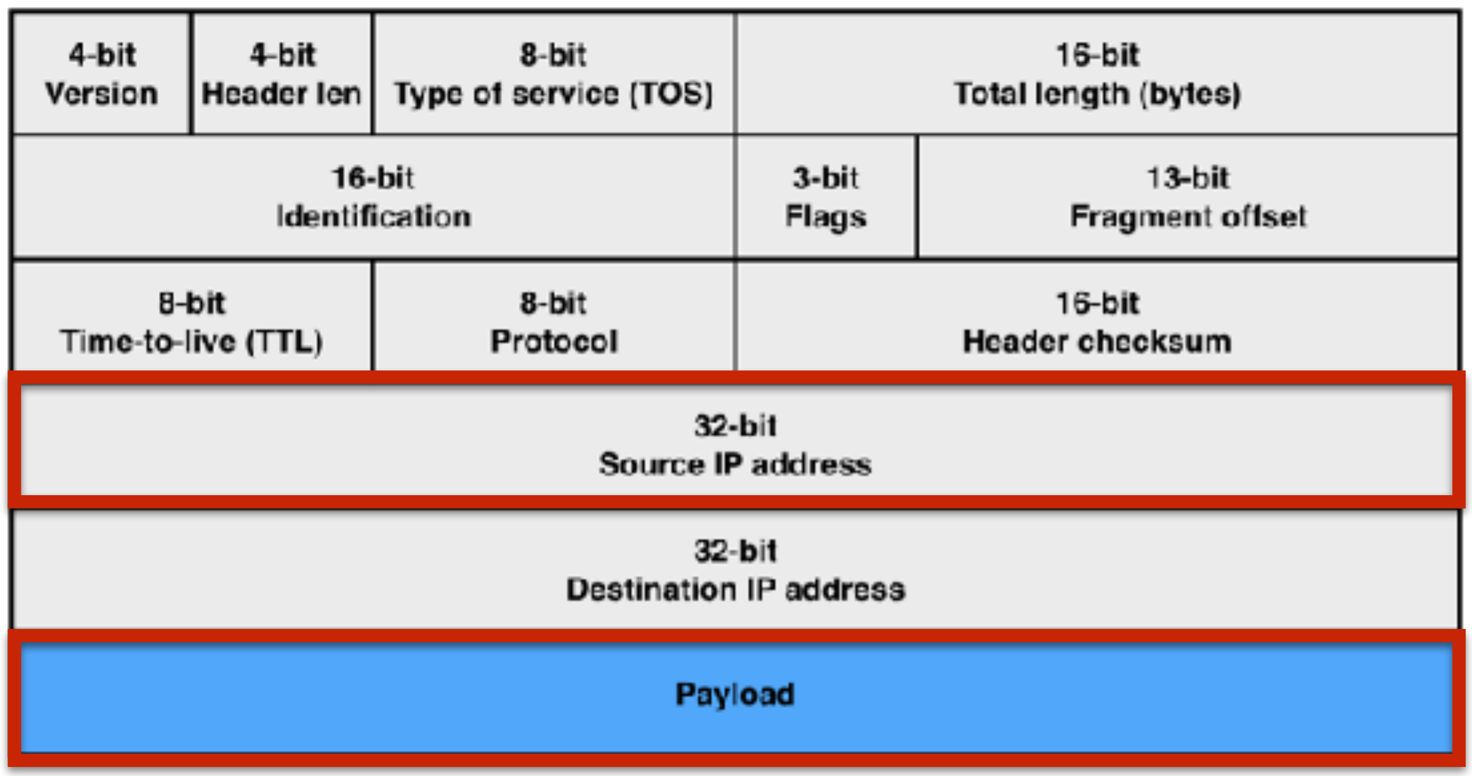
IP: “Best effort” packet delivery

- Routers inspect destination address, determine “next hop” in the forwarding table

- Best effort = “I’ll give it a try”
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order

Fixing these is the job of the transport layer!

Attacks on IP



Source-spoof

There is nothing in IP that enforces that your source IP address is really "yours"

Eavesdrop / Tamper

IP provides no protection of the *payload* or *header*

Source-spoofing

- Why source-spoof?
  - Consider spam: send many emails from one computer
  - Easy defense: block many emails from a given (source) IP address
  - Easy countermeasure: spoof the source IP address
  - Counter-countermeasure?

- How do you know if a packet you receive has a spoofed source?

Salient network features

- Recall: The Internet operates via *destination-based routing*
- attacker: pkt (spoofed source) -> destination
  destination: pkt -> spoofed source
- In other words, the response goes to the spoofed source, *not* the attacker

Defending against source-spoofing

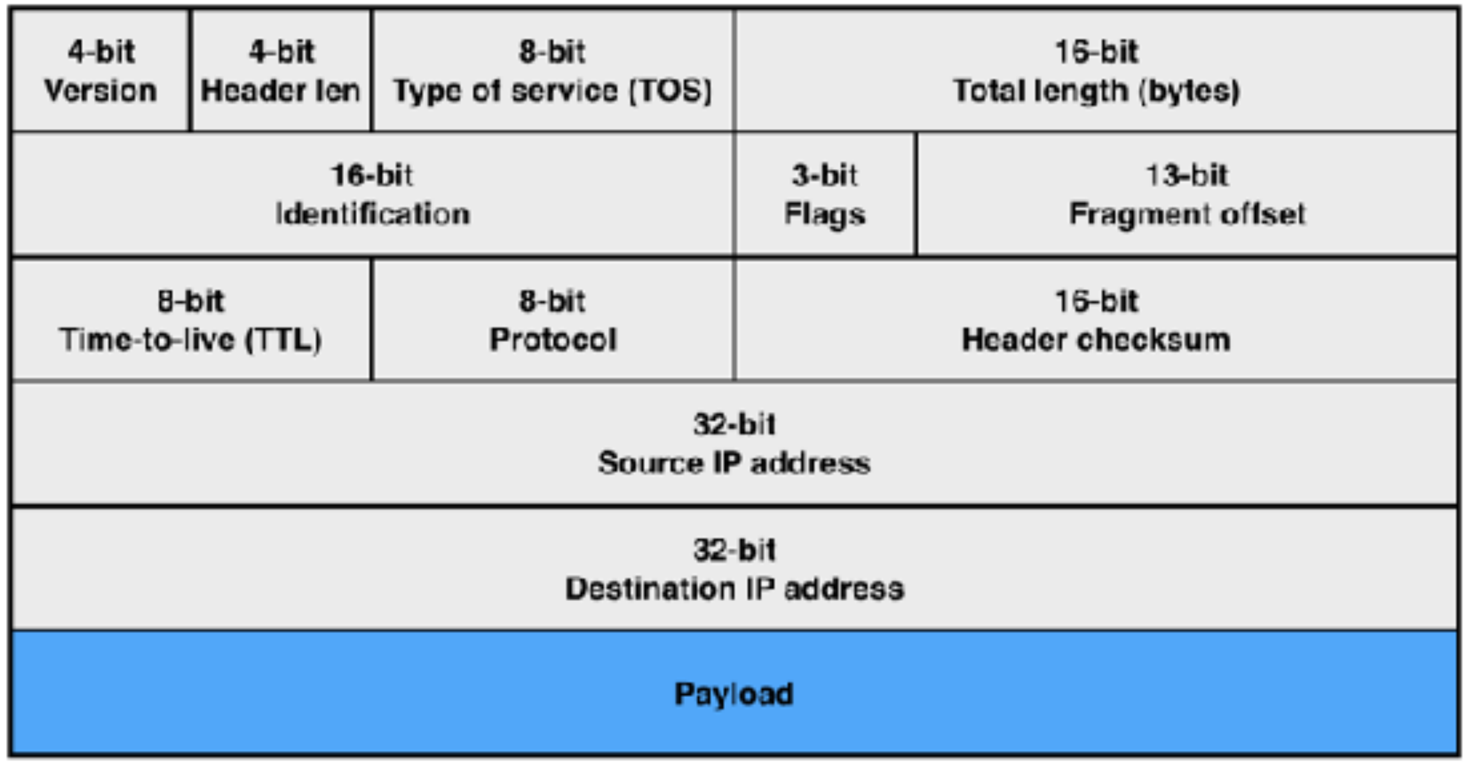
- How do you know if a packet you receive has a spoofed source?
  - Send a challenge packet to the (possibly spoofed) source (e.g., a difficult to guess, random nonce)
  - If the recipient can answer the challenge, then likely that the source was not spoofed

- So do you have to do this with every packet??
  - Every packet should have something that's difficult to guess
  - Recall the query ID in the DNS queries! Easy to predict => Kaminsky attack

Source spoofing

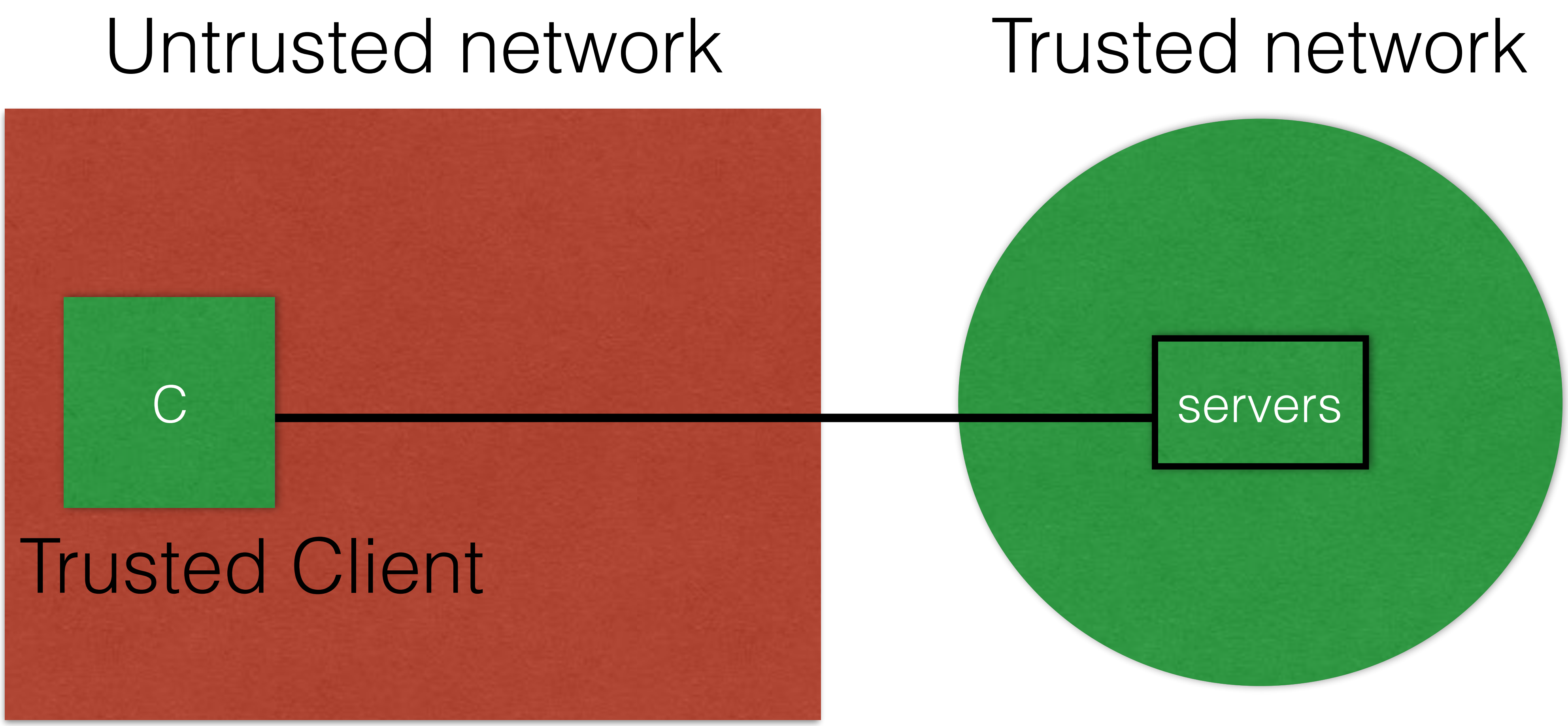
- Why source-spoof?
  - Consider DoS attacks: generate as much traffic as possible to congest the victim's network
  - Easy defense: block all traffic from a given source near the edge of your network
  - Easy countermeasure: spoof the source address
- Challenges won't help here; the damage has been done by the time the packets reach the core of our network
- Ideally, detect such spoofing *near the source*

Eavesdropping / Tampering



- No security built into IP
- => Deploy secure IP *over IP*

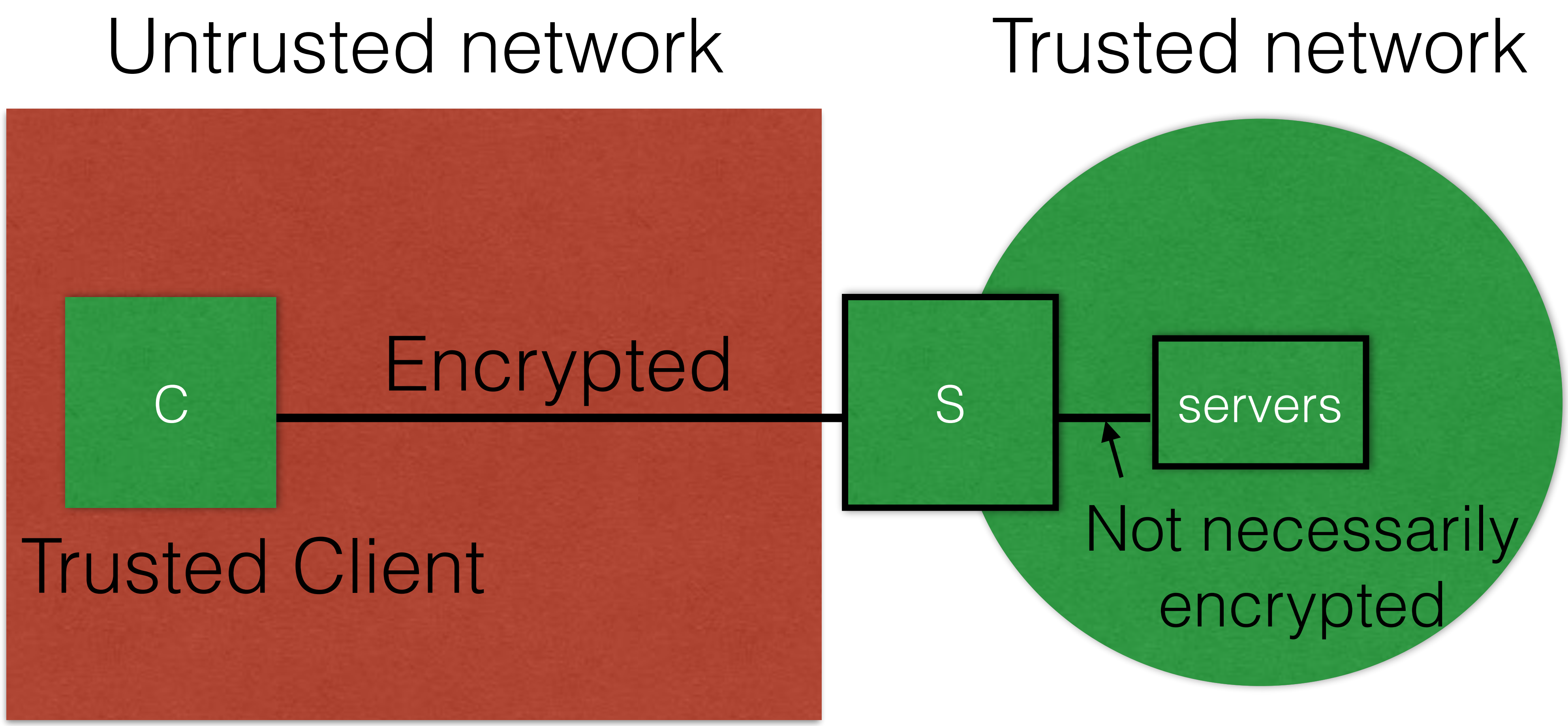
Virtual Private Networks (VPNs)



Goal: Allow the client to connect to the trusted network from within an untrusted network

Example: Connect to your company's network (for payroll, file access, etc.) while visiting a competitor's office

Virtual Private Networks (VPNs)



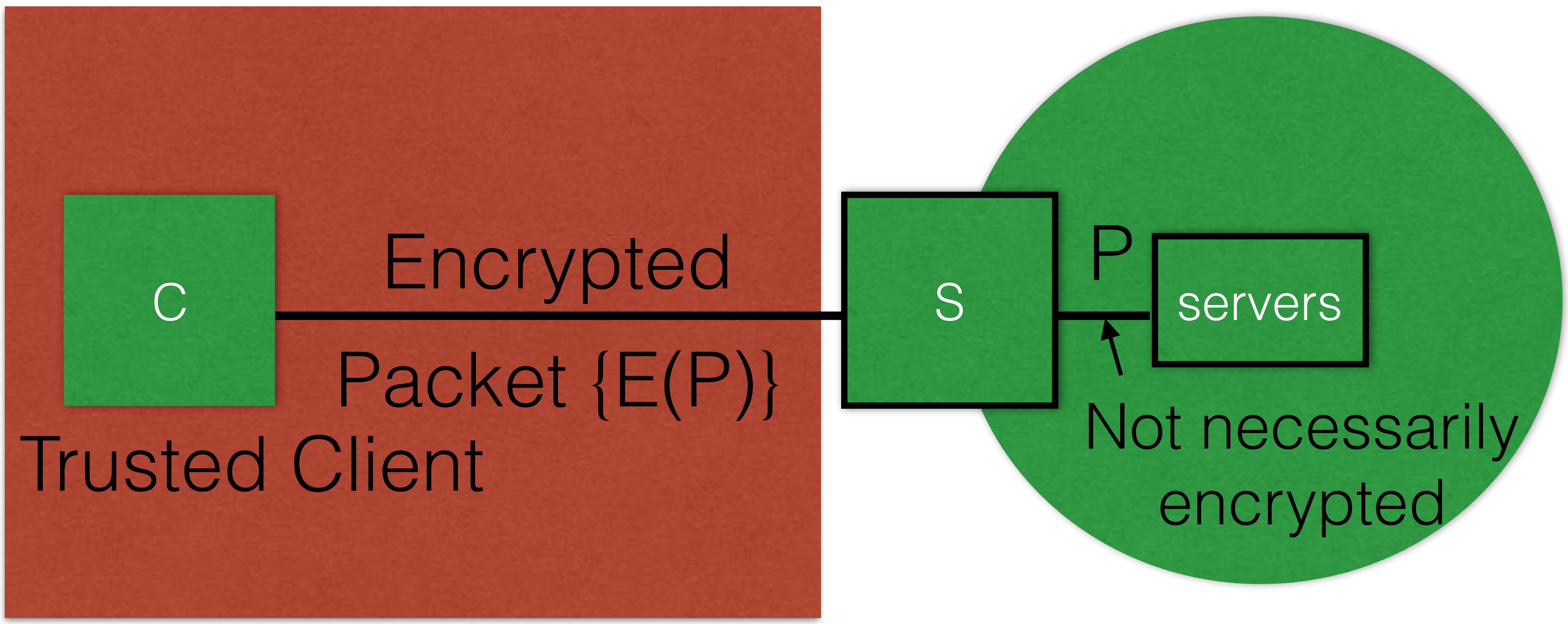
Idea: A VPN “client” and “server” together create end-to-end encryption/authentication

Predominate way of doing this: IPSec

IPSec

- Operates in a few different modes
  - Transport mode: Simply encrypt the payload but not the headers
  - Tunnel mode: Encrypt the payload *and* the headers
- But how do you encrypt the headers? How does routing work?
  - Encrypt the entire IP packet and make that the payload of another IP packet

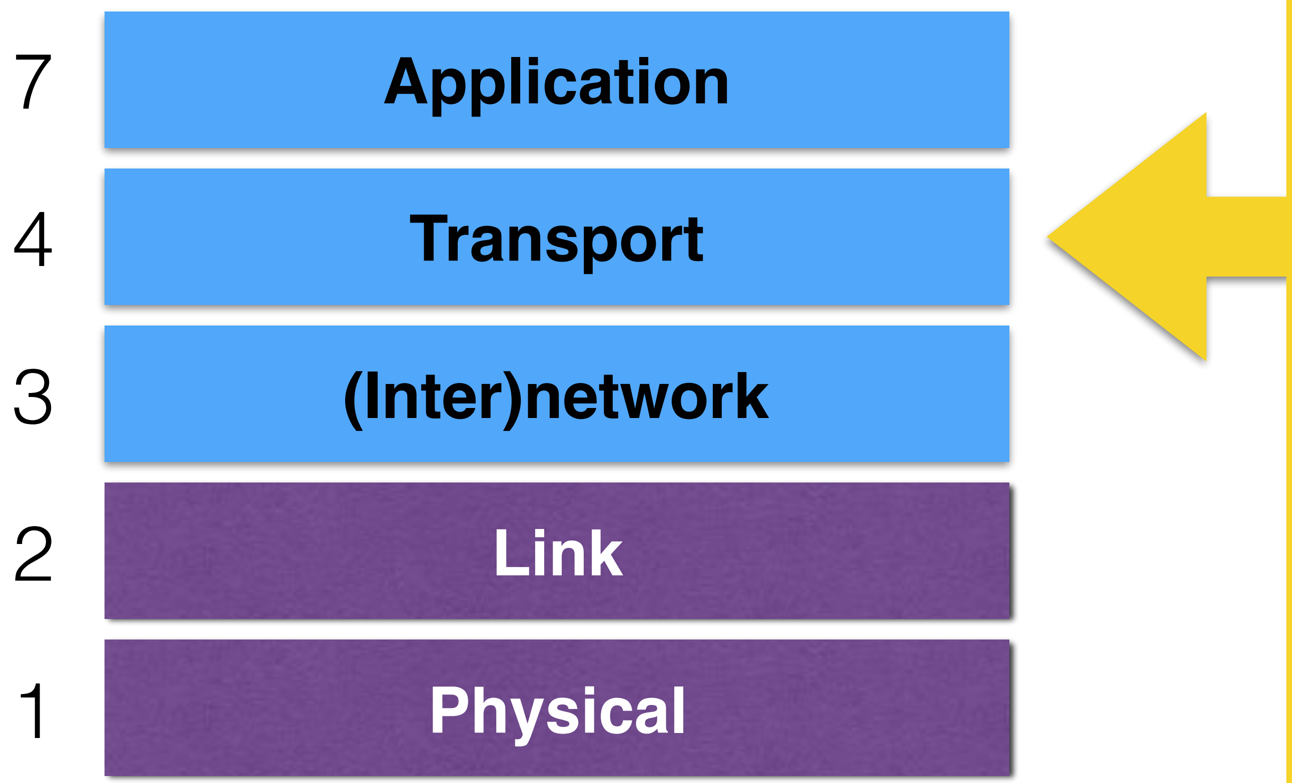
Tunnel mode



The VPN server decrypts and then sends the payload (itself a full IP packet) as if it had just received it from the network

From the client/servers' perspective:
Looks like the client is physically connected to the network!

Layer 4: Transport layer



- End-to-end communication between **processes**
- Different types of services provided:
  - UDP: unreliable *datagrams*
  - TCP: *reliable* byte stream
- “Reliable” = keeps track of what data were received properly and retransmits as necessary

TCP: reliability

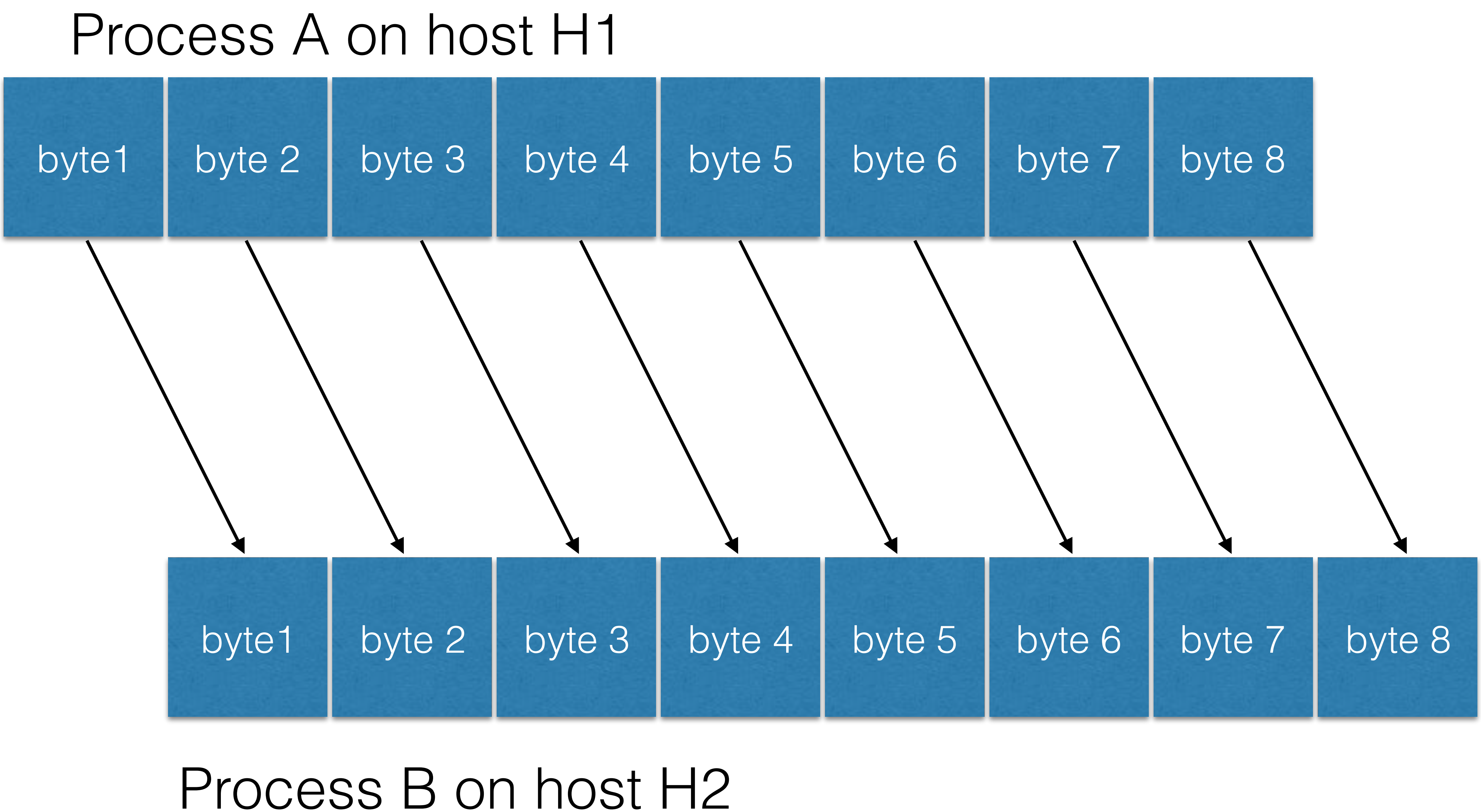
- Given best-effort deliver, the goal is to ensure *reliability*
  - All packets are delivered to applications
  - … in order
  - … unmodified (with reasonably high probability)
- Must robustly detect and retransmit lost data

TCP's bytestream service

- Process A on host 1:
  - Send byte 0, byte 1, byte 2, byte 3, …
- Process B on host 2:
  - Receive byte 0, byte 1, byte 2, byte 3, …
- The applications do **not** see:
  - packet boundaries (looks like a stream of bytes)
  - lost or corrupted packets (they're all correct)
  - retransmissions (they all only appear once)

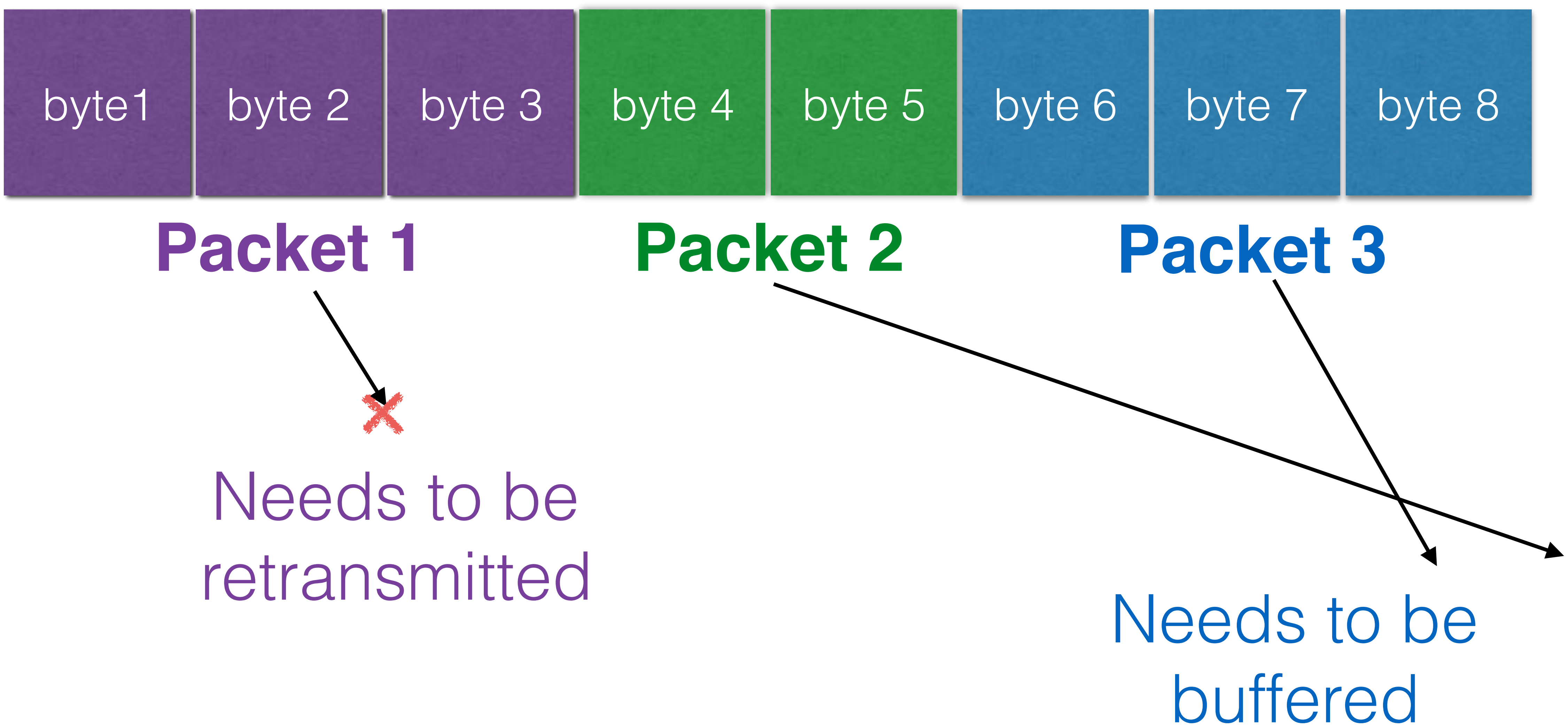
TCP bytestream service

Abstraction: Each *byte* reliably delivered in order



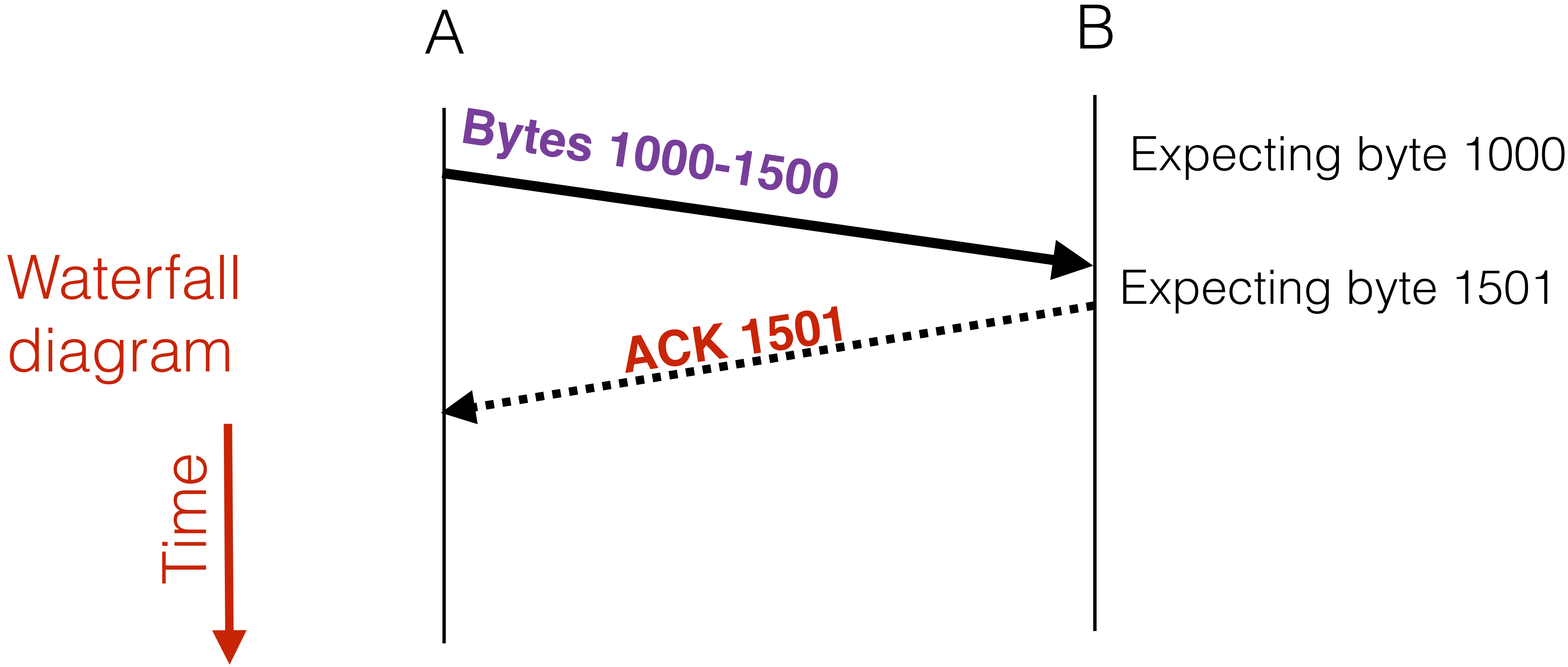
TCP bytestream service

Reality: *Packets* sometimes retransmitted, sometimes arrive out of order



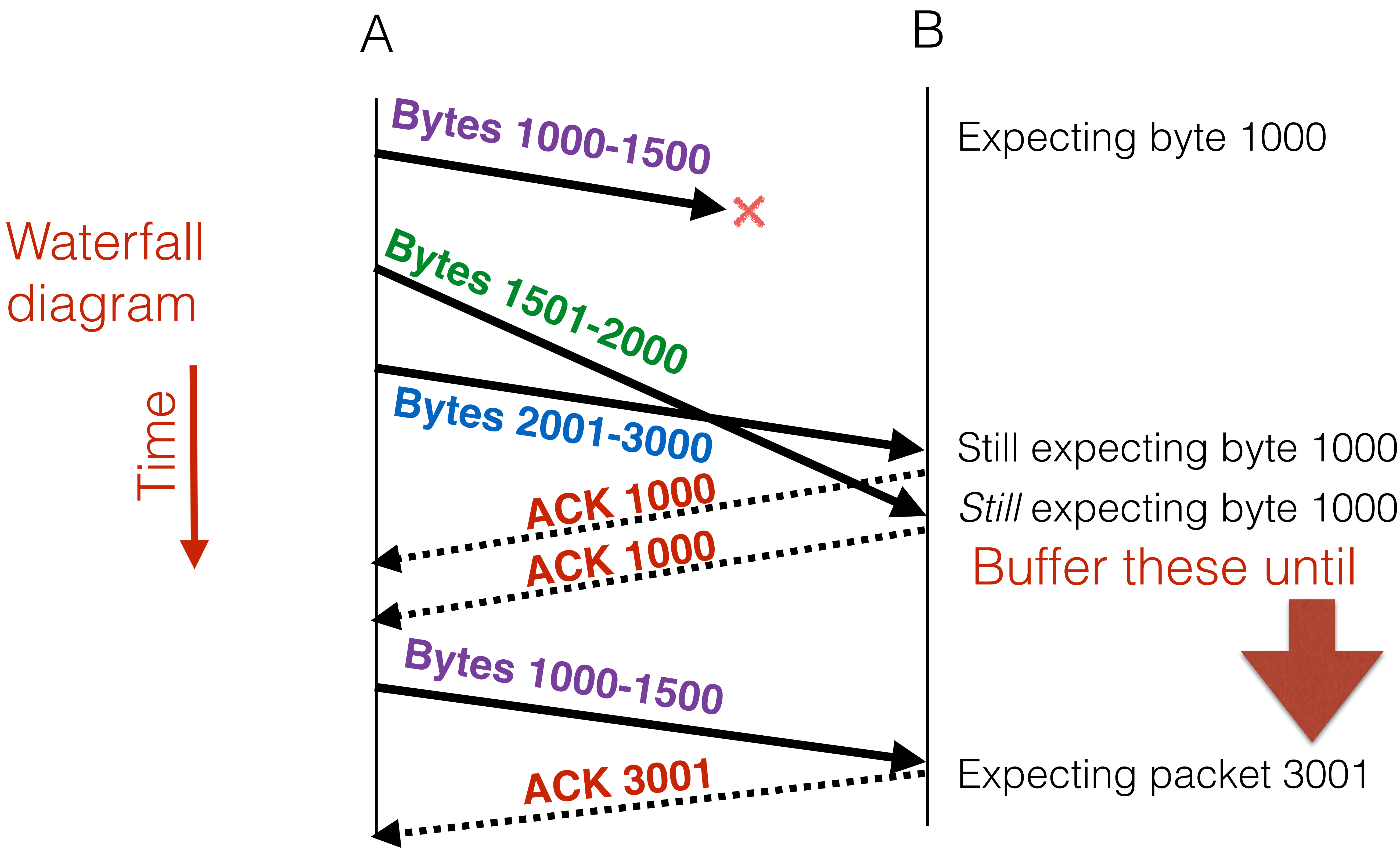
TCP's first job: achieve the abstraction while hiding the reality from the application

How does TCP achieve reliability?



Reliability through acknowledgments
to determine whether something was received.

How does TCP achieve reliability?

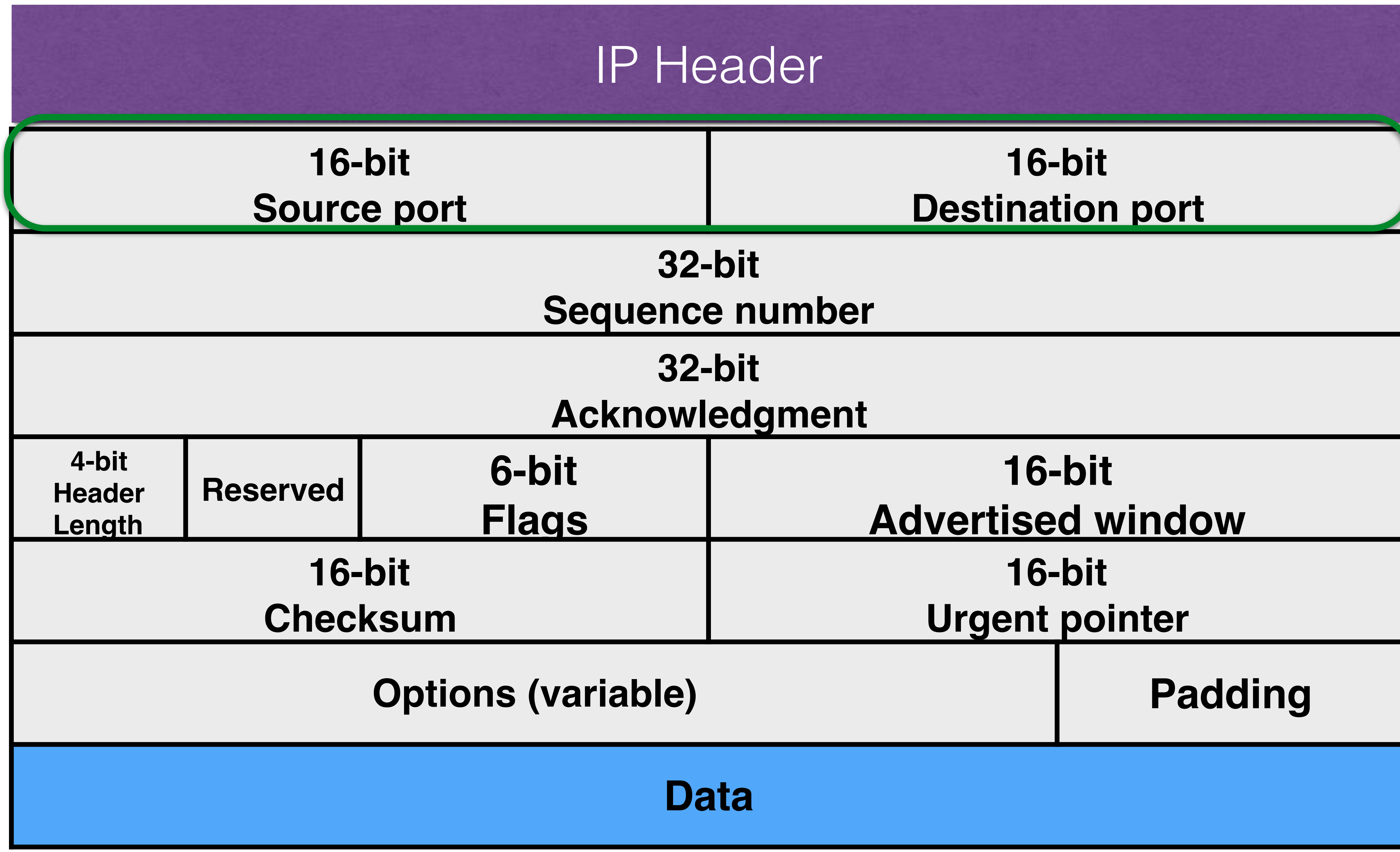


TCP congestion control

TCP's second job: don't break the network!

- Try to use as much of the network as is safe (does not adversely affect others' performance) and efficient (makes use of network capacity)
- Dynamically adapt how quickly you send based on the network path's capacity
- When an ACK doesn't come back, the network may be beyond capacity: slow down.

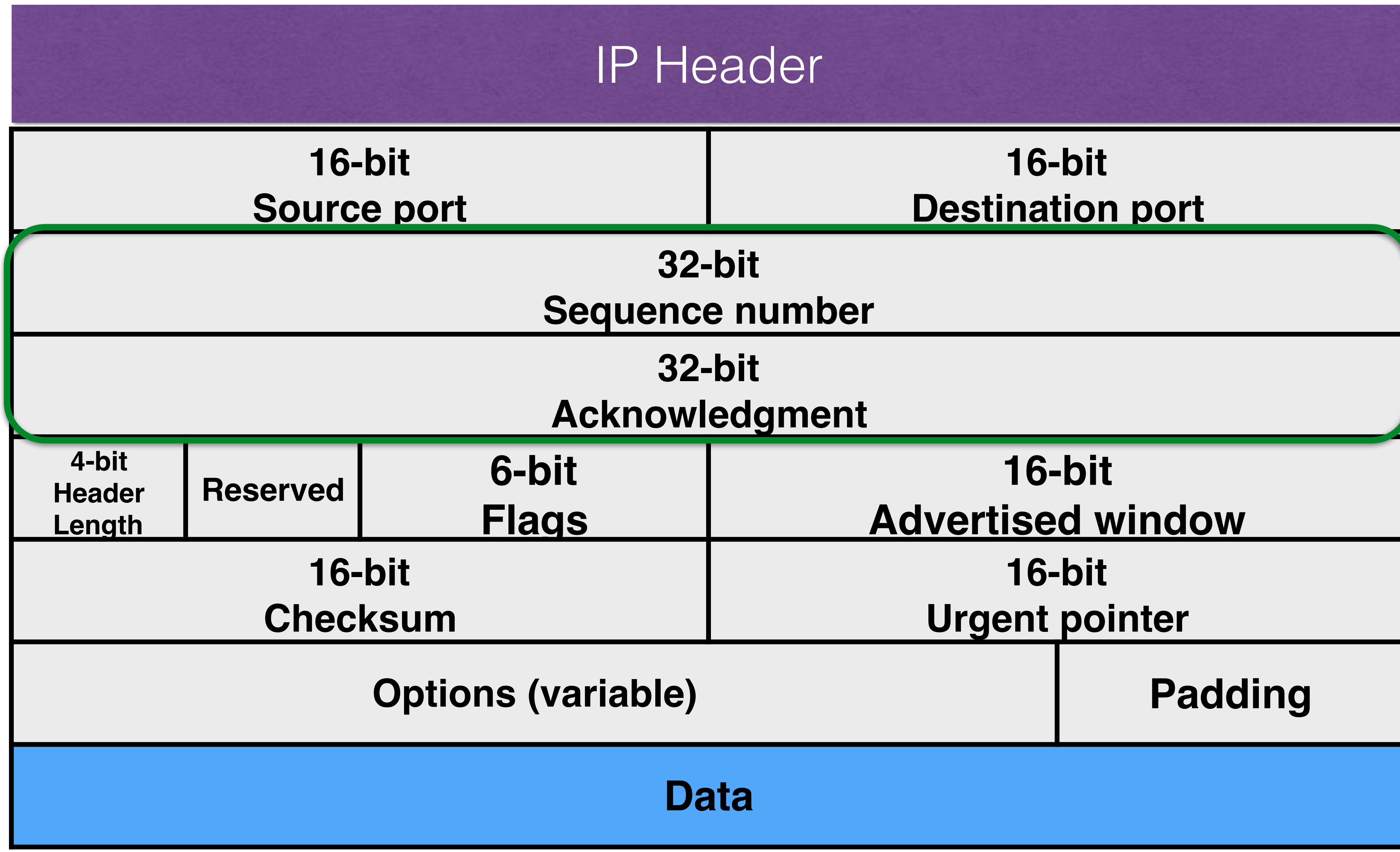
TCP header


IP Header
16-bit
Source port
16-bit
Destination port
32-bit
Sequence number
32-bit
Acknowledgment
4-bit
Header
Length
Reserved
6-bit
Flags
16-bit
Advertised window
16-bit
Checksum
16-bit
Urgent pointer
Options (variable)
Padding
Data

TCP ports

- Ports are associated with **OS processes**
- Sandwiched between IP header and the application data
- {src IP/port, dst IP/port} : this 4-tuple uniquely identifies a TCP connection
- Some port numbers are well-known
  - 80 = HTTP
  - 53 = DNS

TCP header



IP Header
16-bit
Source port
16-bit
Destination port
32-bit
Sequence number
32-bit
Acknowledgment
4-bit
Header
Length
Reserved
6-bit
Flags
16-bit
Advertised window
16-bit
Checksum
16-bit
Urgent pointer
Options (variable)
Padding
Data

TCP seqno

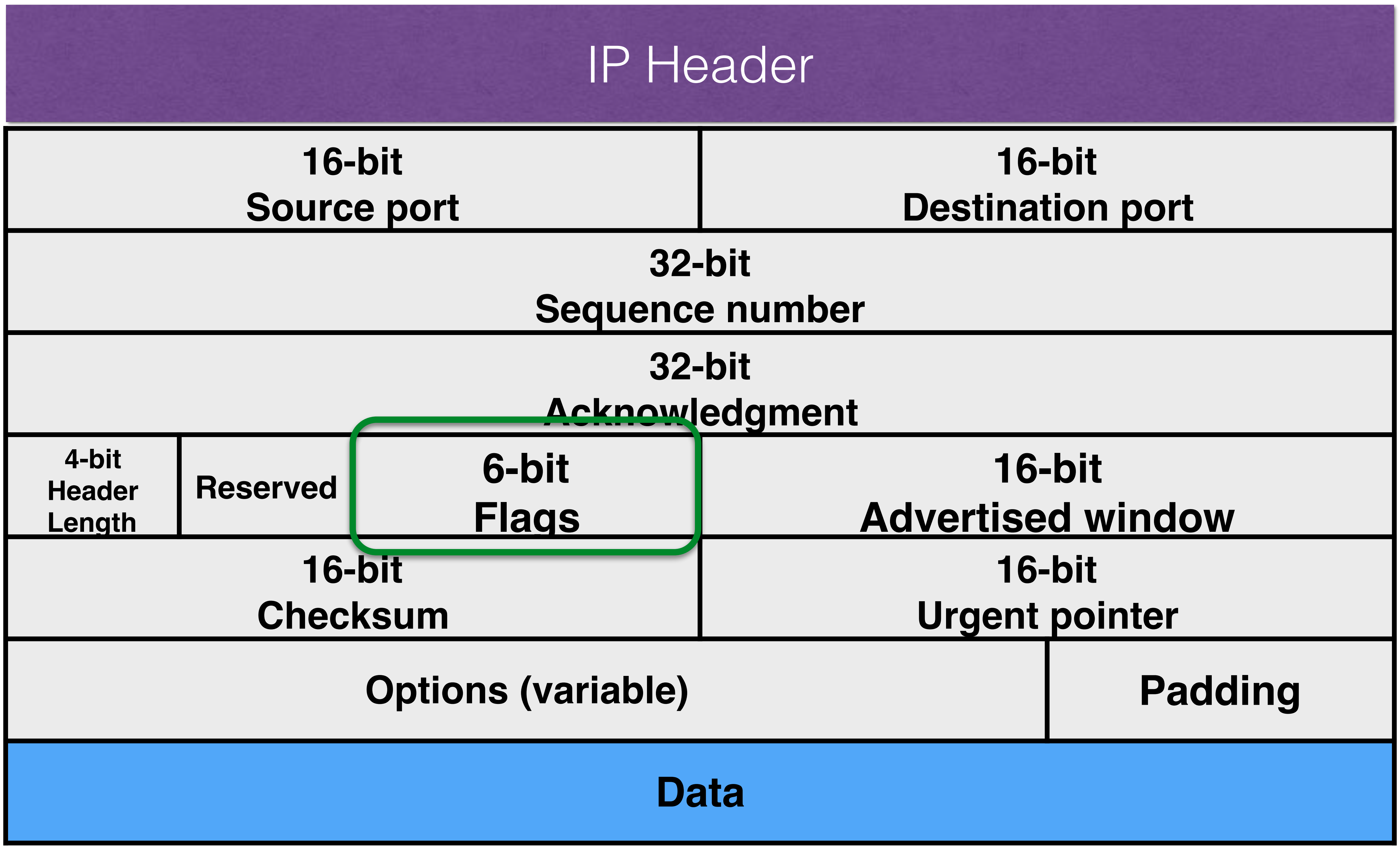
- Each byte in the byte stream has a unique "sequence number"
    - Unique for both directions

- "Sequence number" in the header = sequence number of the ***first*** byte in the packet's data

- Next sequence number = previous seqno + previous packet's data size

- "Acknowledgment" in the header = the ***next*** seqno you expect from the other end-host

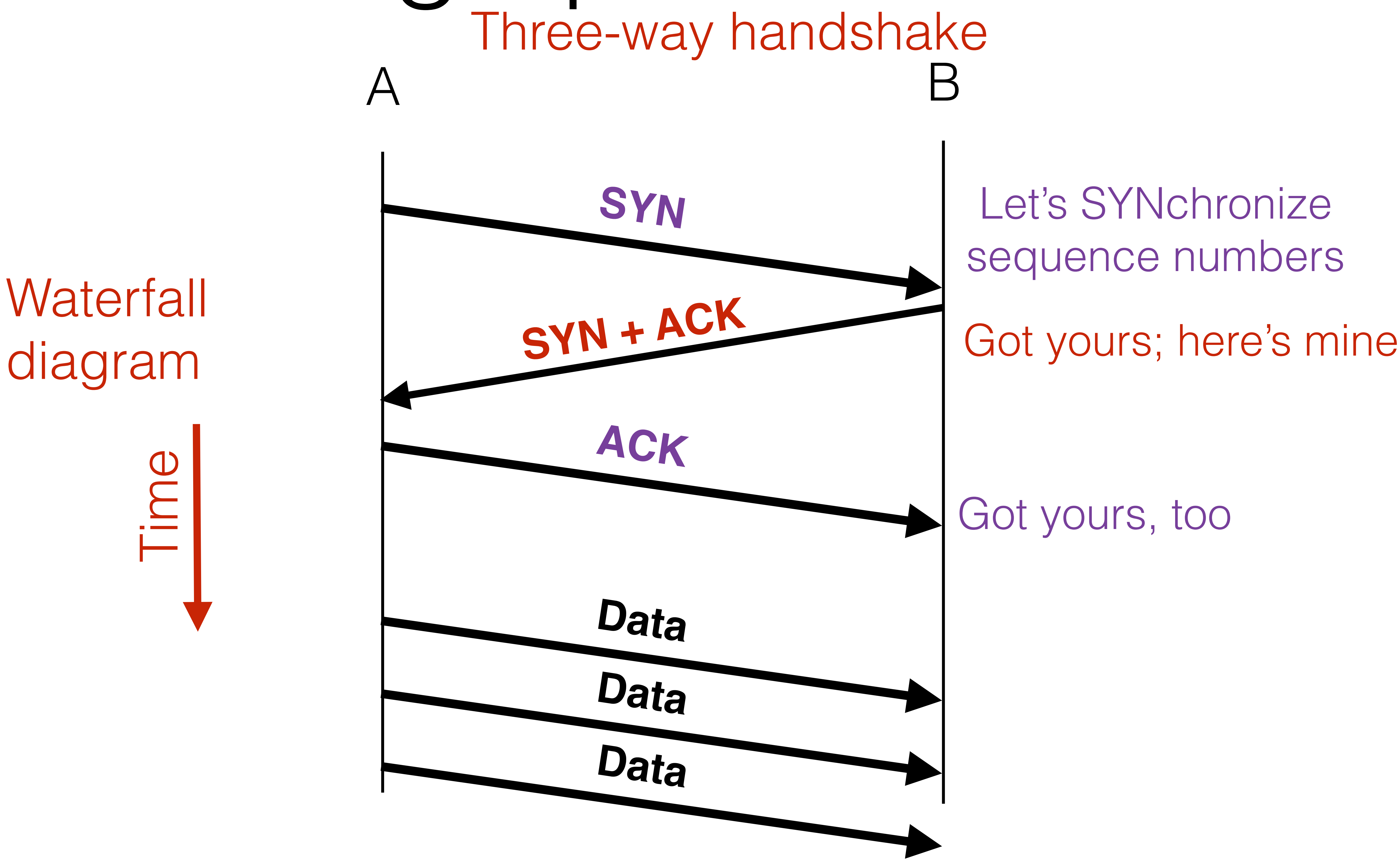
TCP header


IP Header
16-bit
Source port
16-bit
Destination port
32-bit
Sequence number
32-bit
Acknowledgment
4-bit
Header
Length
Reserved
6-bit
Flags
16-bit
Advertised window
16-bit
Checksum
16-bit
Urgent pointer
Options (variable)
Padding
Data

TCP flags

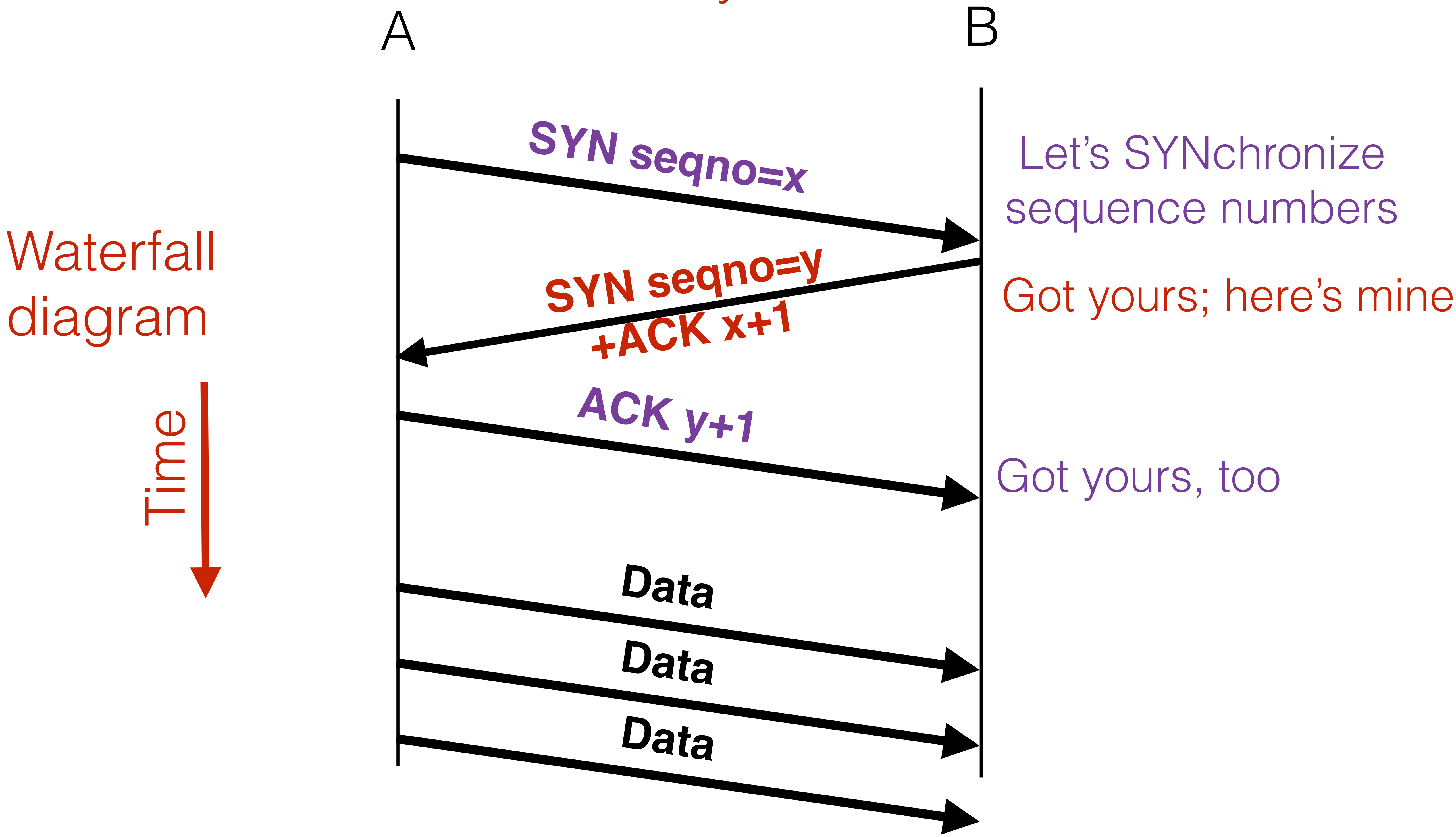
- SYN
  - Used for setting up a connection
- ACK
  - Acknowledgments, for data and “control” packets
- FIN
- RST

Setting up a connection



Setting up a connection

Three-way handshake



TCP flags

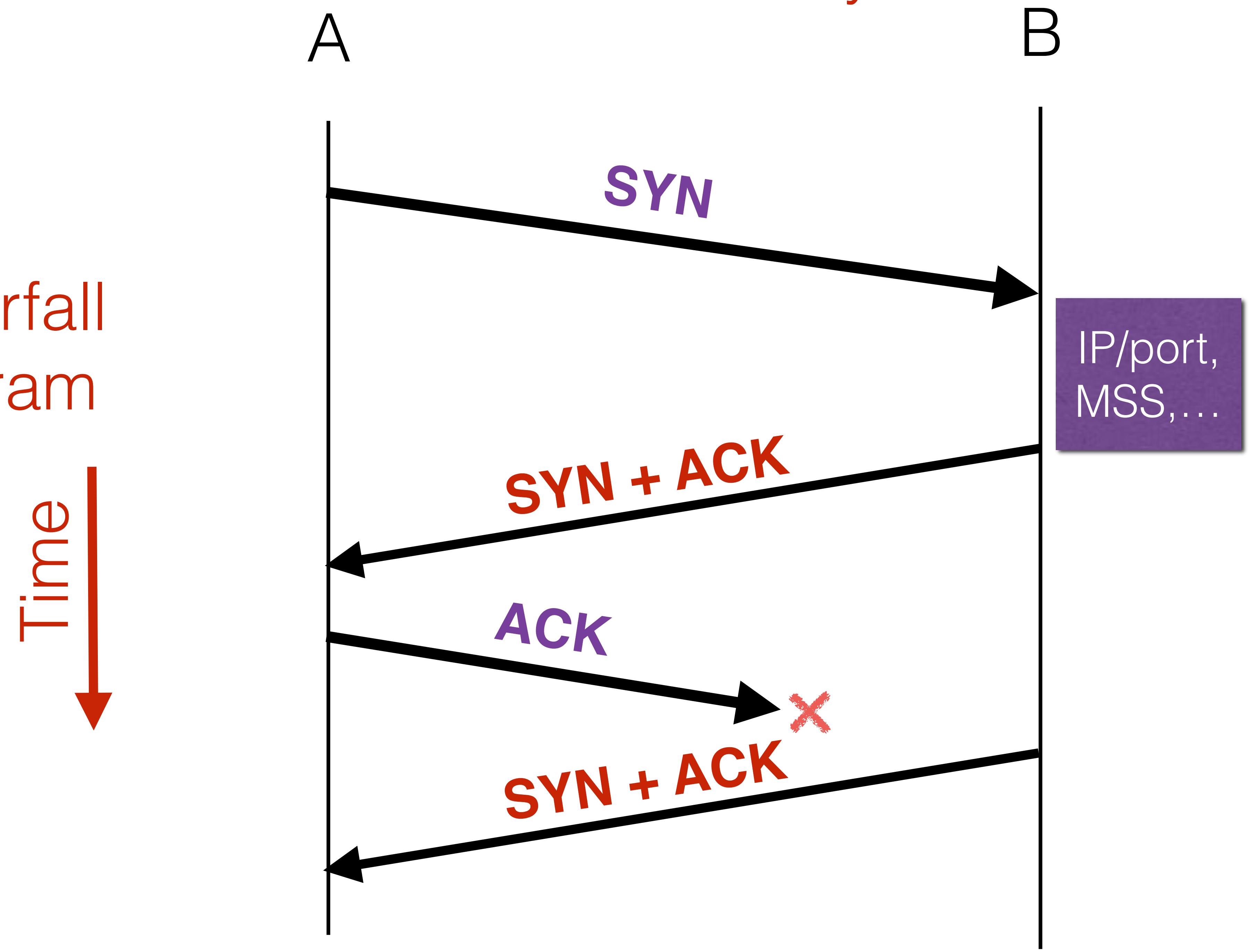
- SYN
- ACK
- FIN: Let’s shut this down (two-way)
  - FIN
  - FIN+ACK
- RST: I’m shutting you down
  - Says “delete all your local state, because I don’t know what you’re talking about

Attacks

- SYN flooding
- Injection attacks
- Opt-ack attack

SYN flooding

Recall the three-way handshake:



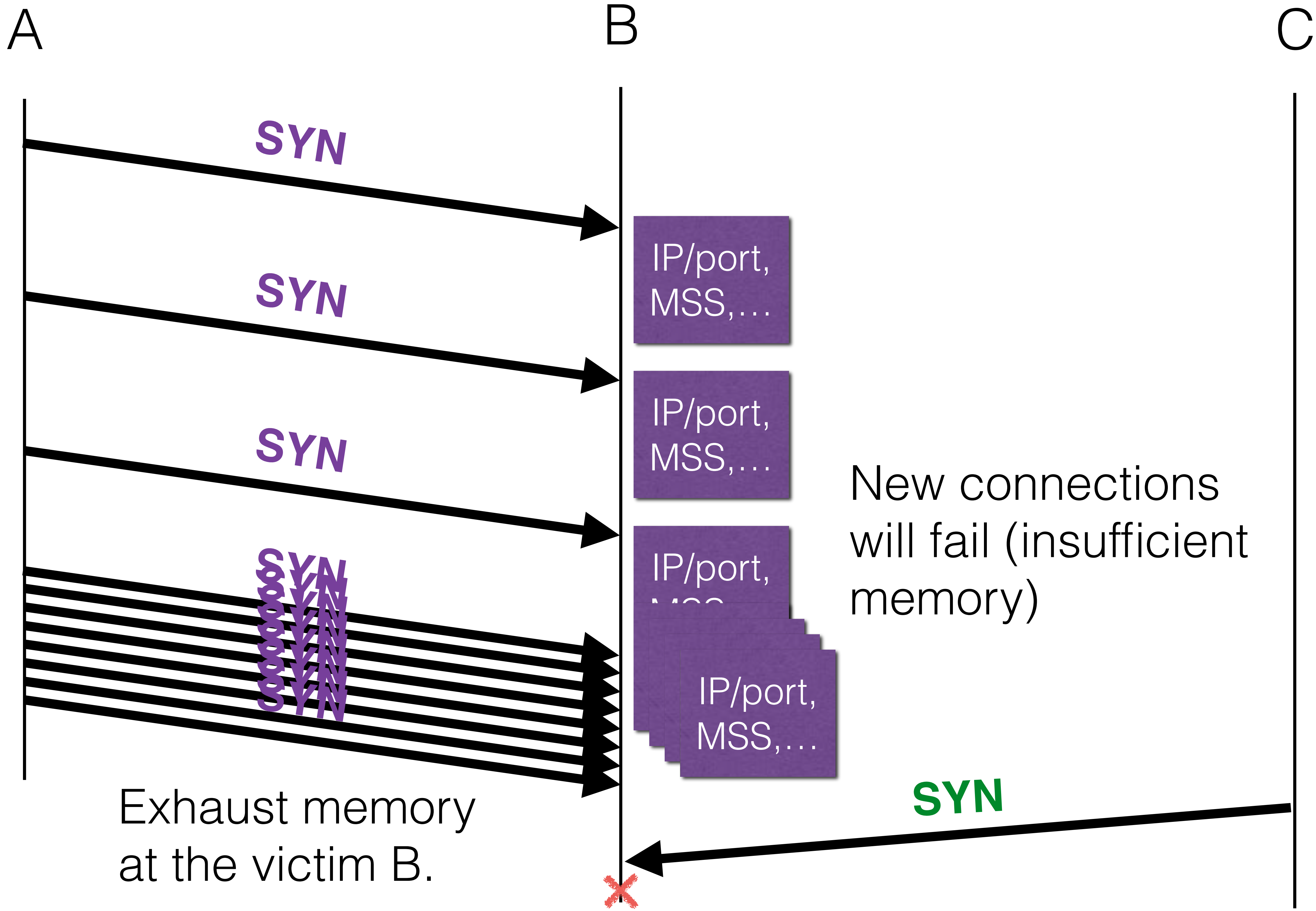
Waterfall diagram

At this point, B allocates state for this new connection (incl. IP, port, maximum segment size)

B will hold onto this local state and retransmit SYN+ACK's until it hears back or times out (up to 63 sec).

SYN flooding

The attack

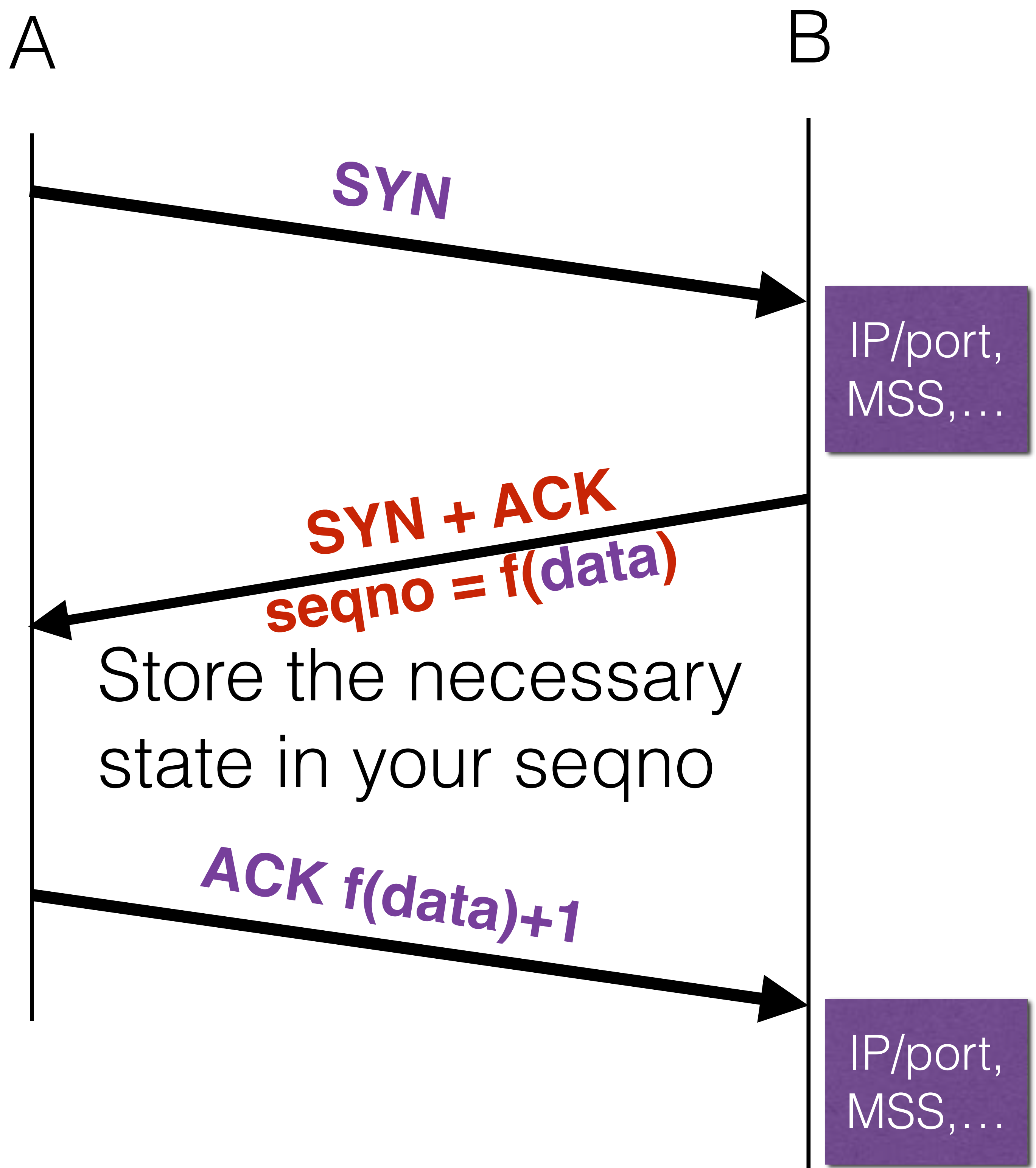


SYN flooding details

- Easy to detect many incomplete handshakes from a single IP address
- *Spoof* the source IP address
  - It's just a field in a header: set it to whatever you like
- Problem: the host who really owns that spoofed IP address may respond to the SYN+ACK with a RST, deleting the local state at the victim
- Ideally, spoof an IP address of a host you know won't respond

SYN cookies

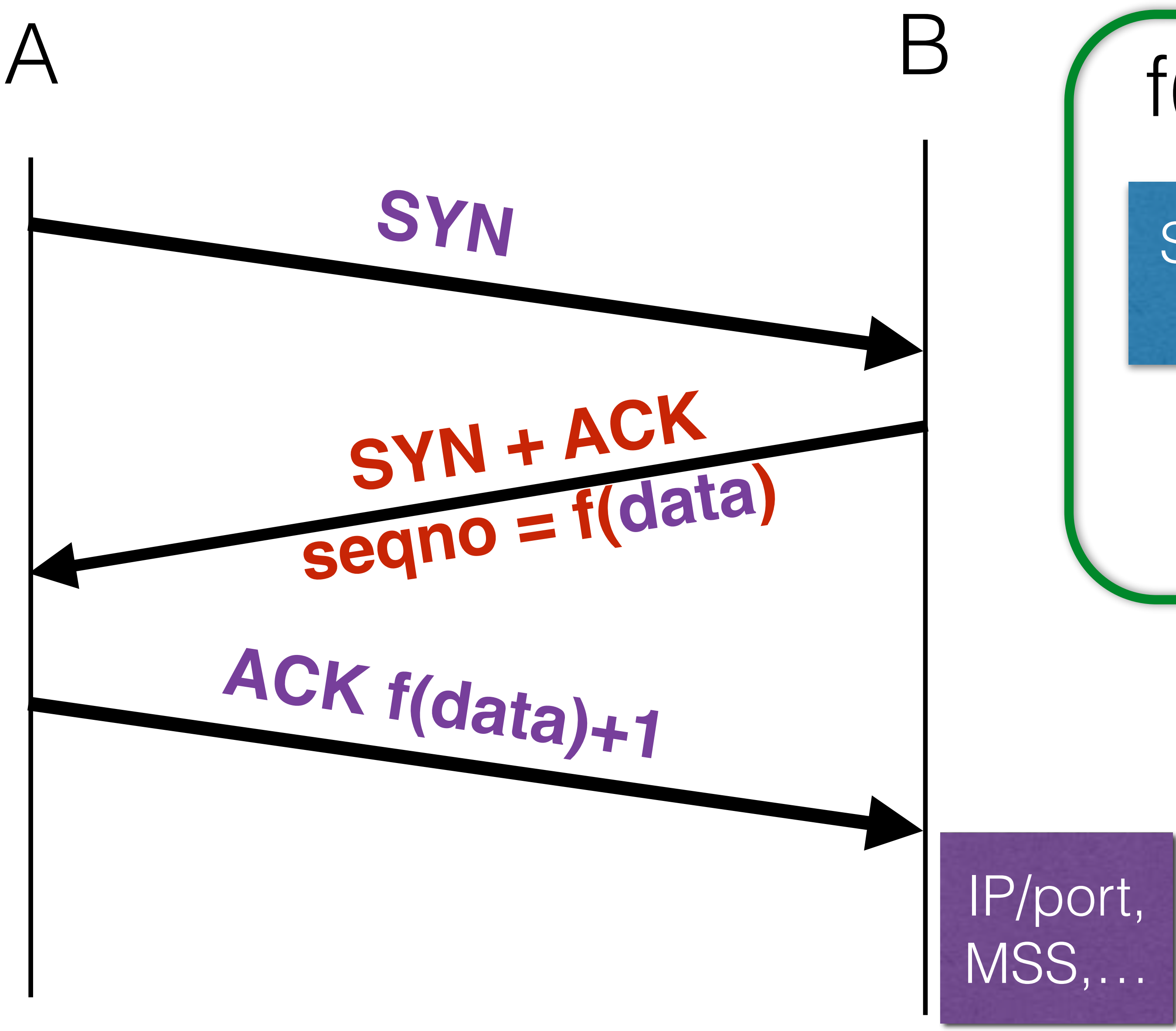
The defense



Rather than store this data, send it to the host who is initiating the connection and have him return it to you

Check that f(data) is valid for this connection. Only at that point do you allocate state.

SYN cookie format



f(.) = 32-bit seqno

Slow-moving timestamp	MSS	Secure hash
Prevents replay attacks	The info we need for this connection	Includes: IPs/ports, MSS, timestamp

The secure hash makes it difficult for the attacker to guess what f() will be, and therefore the attacker cannot guess a correct ACK if he spoofs.

Injection attacks

- Suppose you are on the path between src and dst; what can you do?
  - Trivial to inject packets with the correct sequence number

- What if you are not on the path?
  - Need to guess the sequence number
  - Is this difficult to do?

Initial sequence numbers

- Initial sequence numbers used to be deterministic
- What havoc can we wreak?
  - Send RSTs
  - Inject data packets into an existing connection (TCP veto attacks)
  - *Initiate and use an entire connection without ever hearing the other end*

U.S. Department of Justice
United States Marshals Service

WANTED
BY U.S. MARSHALS

NOTICE TO ARRESTING AGENCY: Before arrest, validate warrant through National Crime Information Center (NCIC).

United States Marshals Service NCIC entry number: (NIC/ W721460021).

NAME: ....MITNICK, KEVIN DAVID

AKS (S): ....MITNIK, KEVIN DAVID
MERRILL, BRIAN ALLEN

DESCRIPTION:

Sex: ....MALE
Race: ....WHITE
Place of Birth: ....VAN NUYS. CALIFORNIA
Date(s) of Birth: 08/06/63; 10/18/70
Height: ....5'11"
Weight: 190
Eyes: ....BLUE
Hair: ....BROWN
Skintone: ....LIGHT
Scars, Marks, Tattoos: ....NONE KNOWN
Social Security Number (s): 550-39-5695
NCIC Fingerprint Classification: ...00PM20PM13DIPM19PM09

ADDRESS AND LOCALE: KNOWN TO RESIDE IN THE SAN FERNANDO VALLEY AREA OF CALIFORNIA AND LAS VEGAS, NEVADA

WANTED FOR: VIOLATION OF SUPERVISED RELEASE
ORIGINAL CHARGES: POSSESSION UNAUTHORIZED ACCESS DEVICE; COMPUTER FRAUD
Warrant Issued: CENTRAL DISTRICT OF CALIFORNIA
Warrant Number: 9312-1112-0154-C

DATE WARRANT ISSUED: NOVEMBER 10, 1992

MISCELLANEOUS INFORMATION: SUBJECT SUFFERS FROM A WEIGHT PROBLEM AND MAY HAVE EXPERIENCED WEIGHT GAIN OR WEIGHT LOSS

VEHICLE/TAG INFORMATION: NONE KNOWN OFTEN USES PUBLIC TRANSPORTATION

If arrested or whereabouts known, notify the local United States Marshals Office, (Telephone: 213-894-2485).

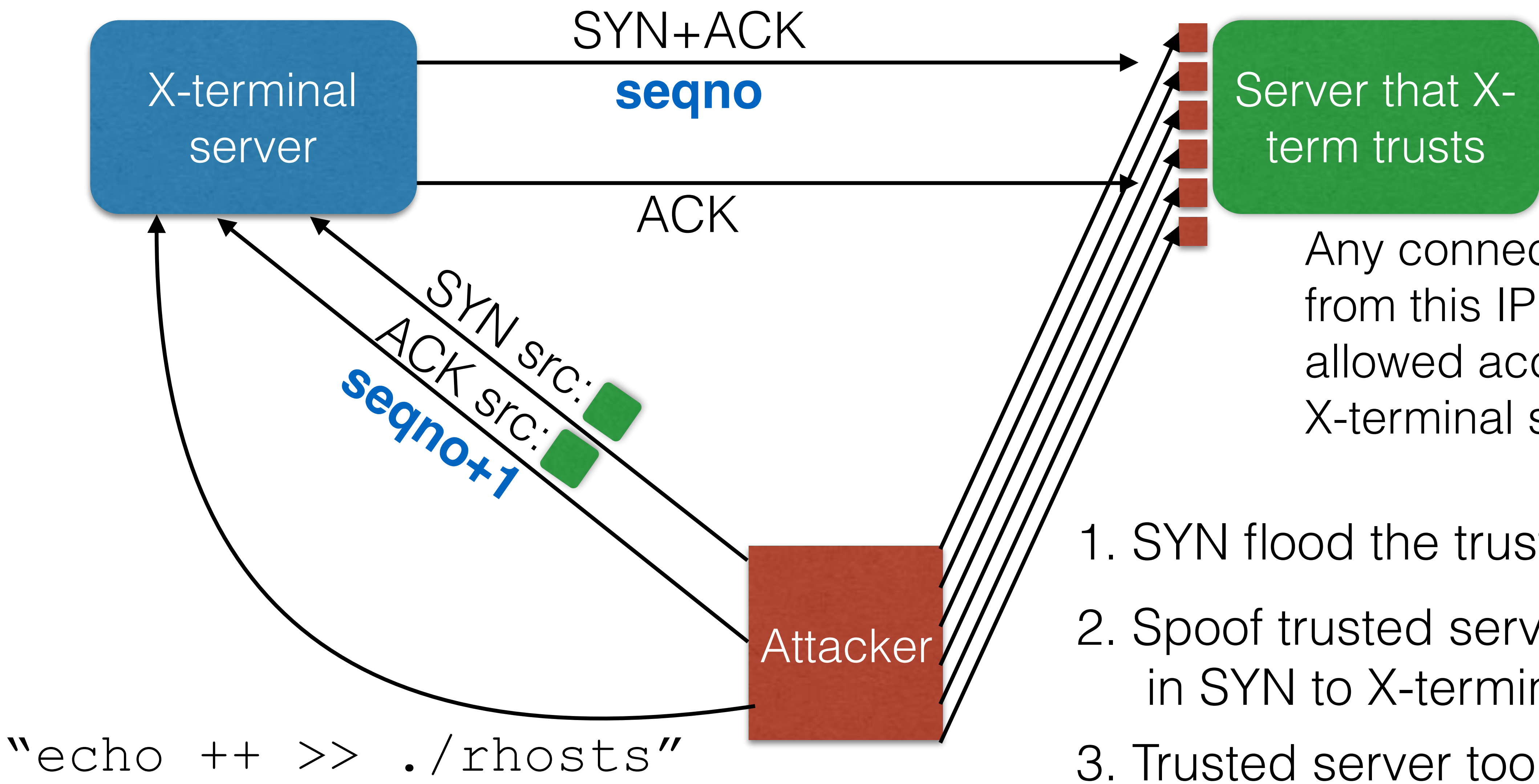
If no answer, call United States Marshals Service Communications Center in McLean Virginia.
Telephone (800)336-0102: (24 hour telephone contact) NLETS access code is VAUSM0000.

Form USM-132
(Rev. 3/2/82)

PRIOR EDITIONS ARE OBSOLETE AND NOT TO BE USED

November 1992

Mitnick attack



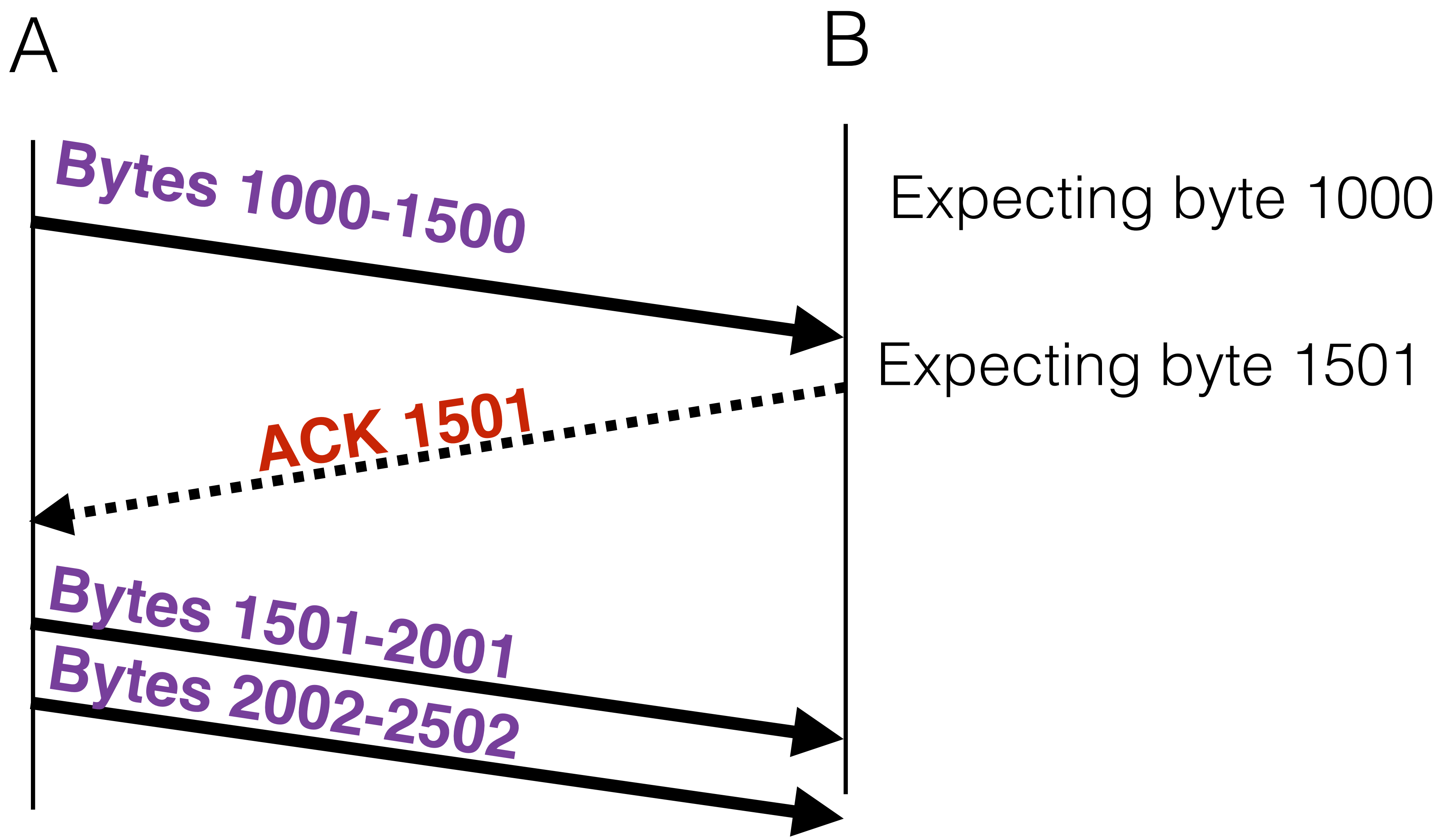
Any connection initiated from this IP address is allowed access to the X-terminal server

1. SYN flood the trusted server
2. Spoof trusted server's IP addr in SYN to X-terminal
3. Trusted server too busy to RST
4. ACK with the guessed **seqno**
5. Grant access to all sources
6. RSTs to trusted server (cleanup)

Defenses

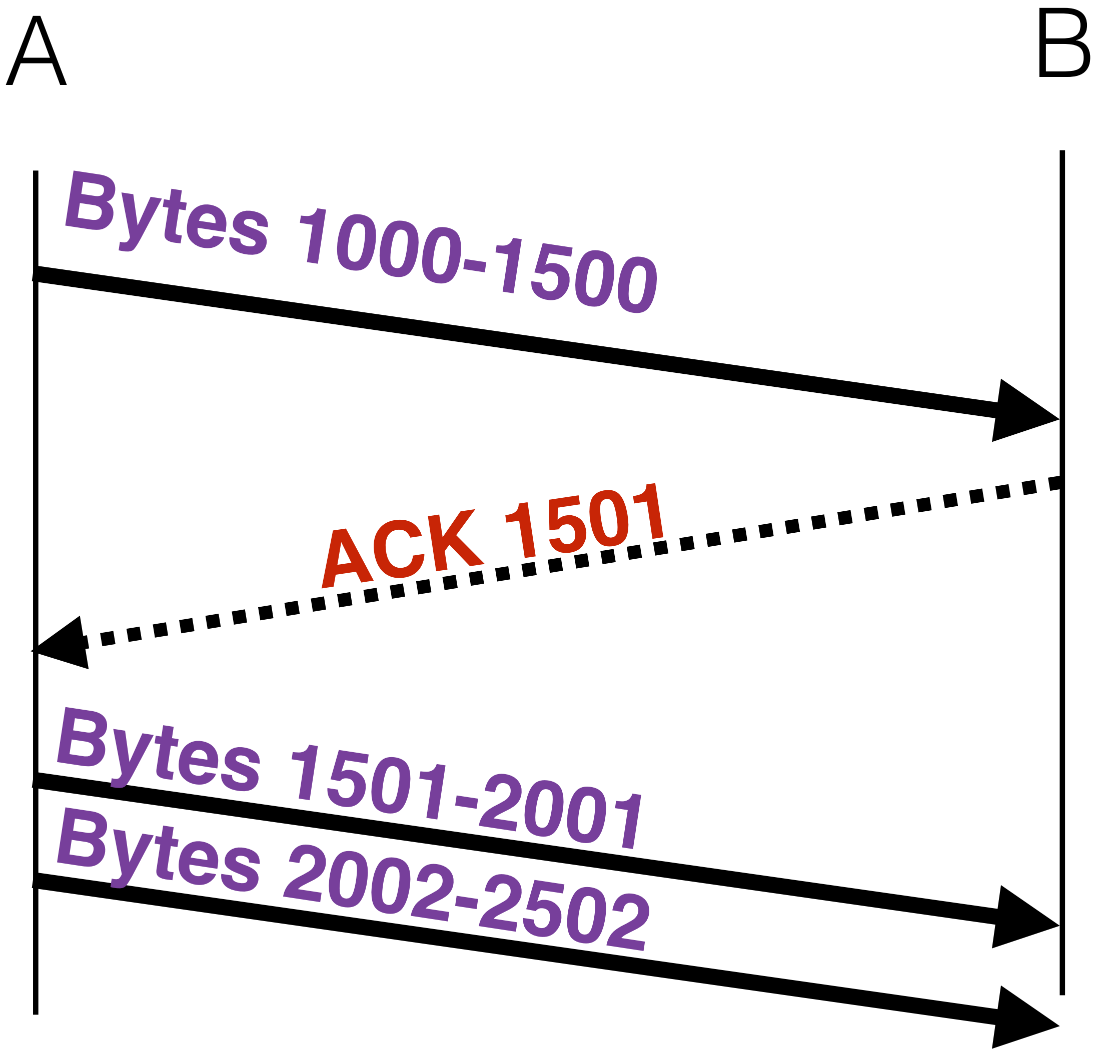
- Initial sequence number must be difficult to predict!

Opt-ack attack



TCP uses ACKs not only for reliability, but also for **congestion control:** the more ACKs come back, the faster I can send

Opt-ack attack

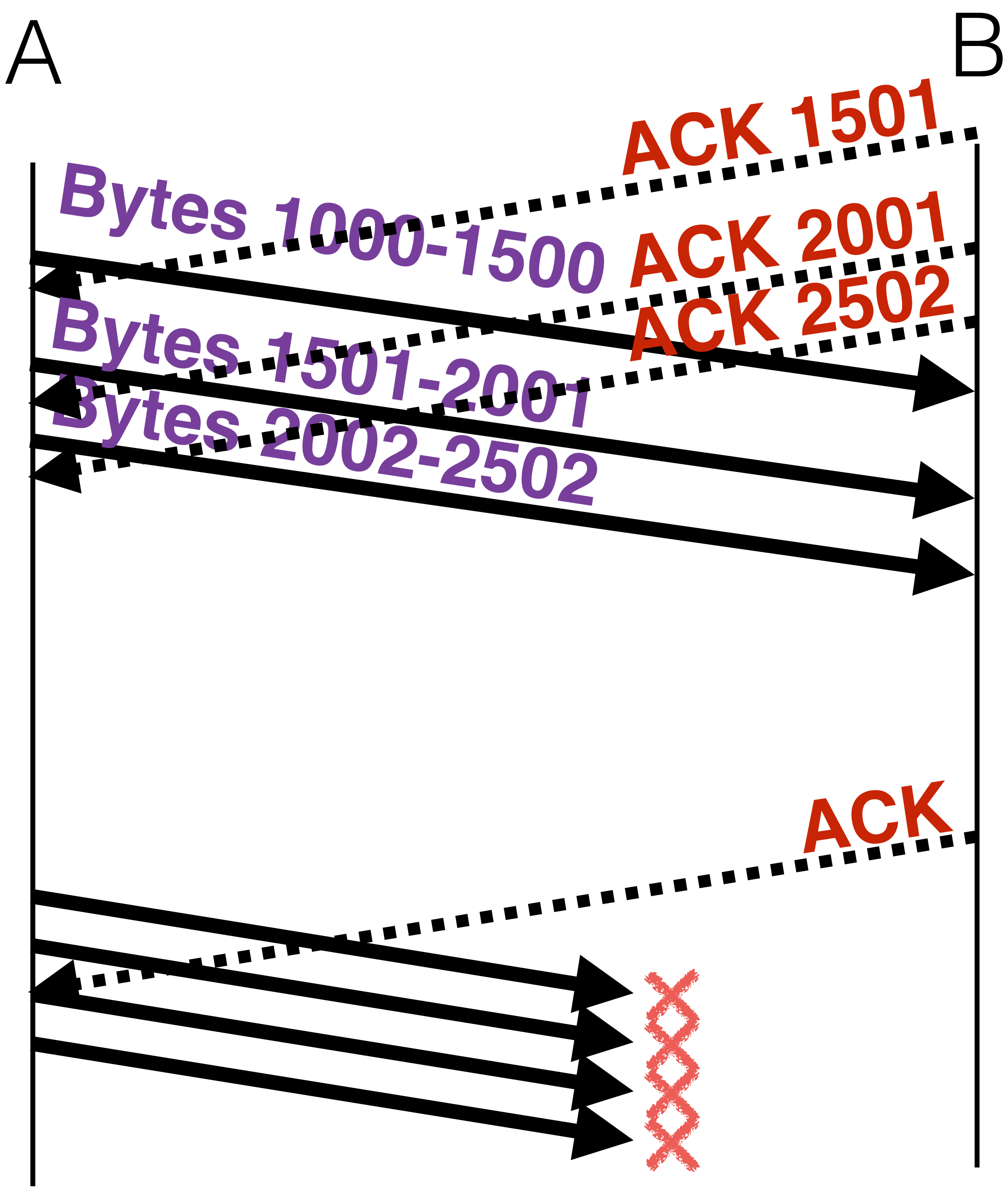


But to get you to send faster, I need to get data in order to ACK, so I need to receive quickly

…or do I?

If I could convince you to send REALLY quickly, then you would effectively DoS your own network!

Opt-ack attack



Then I could ACK early! (“optimistically”)

If I can predict what the last seqno will be *and* when A will send it

A will think “what a fast, legit connection!”

Eventually, A’s outgoing packets will start to get dropped.

But so long as I keep ACKing correctly, it doesn’t matter.

Amplification

- The big deal with this attack is its *Amplification Factor*
  - Attacker sends *x* bytes of data, causing the victim to send many more bytes of data in response
  - Recent examples: NTP, DNSSEC
- Amplified in TCP due to cumulative ACKs
  - “ACK x” says “I’ve seen all bytes up to but not including x”

Opt-ack's amplification factor

- Max bytes sent by victim per ACK:

Packets sent per ACK × Bytes per packet

$$\frac{\text{Max window size}}{\text{MSS}} \times (14 + 40 + \text{MSS})$$

Ethernet (14), TCP/IP (40), Payload (MSS)

- Max ACKs attacker can send per second:

$$\frac{\text{Attacker bandwidth (bytes/sec)}}{(14 + 40)}$$

Size of ACK packet (14 + 40)

Opt-ack's amplification factor

- Boils down to max window size and MSS
  - Default max window size: 65,536
  - Default MSS: 536
- Default amp factor: 65536 * (1/536 + 1/54) ~ **1336x**
- Window scaling lets you increase this by a factor of 2^14
- Window scaling amp factor: ~1336 * 2^14 ~ **22M**
- Using minimum MSS of 88: ~ **32M**

Opt-ack defenses

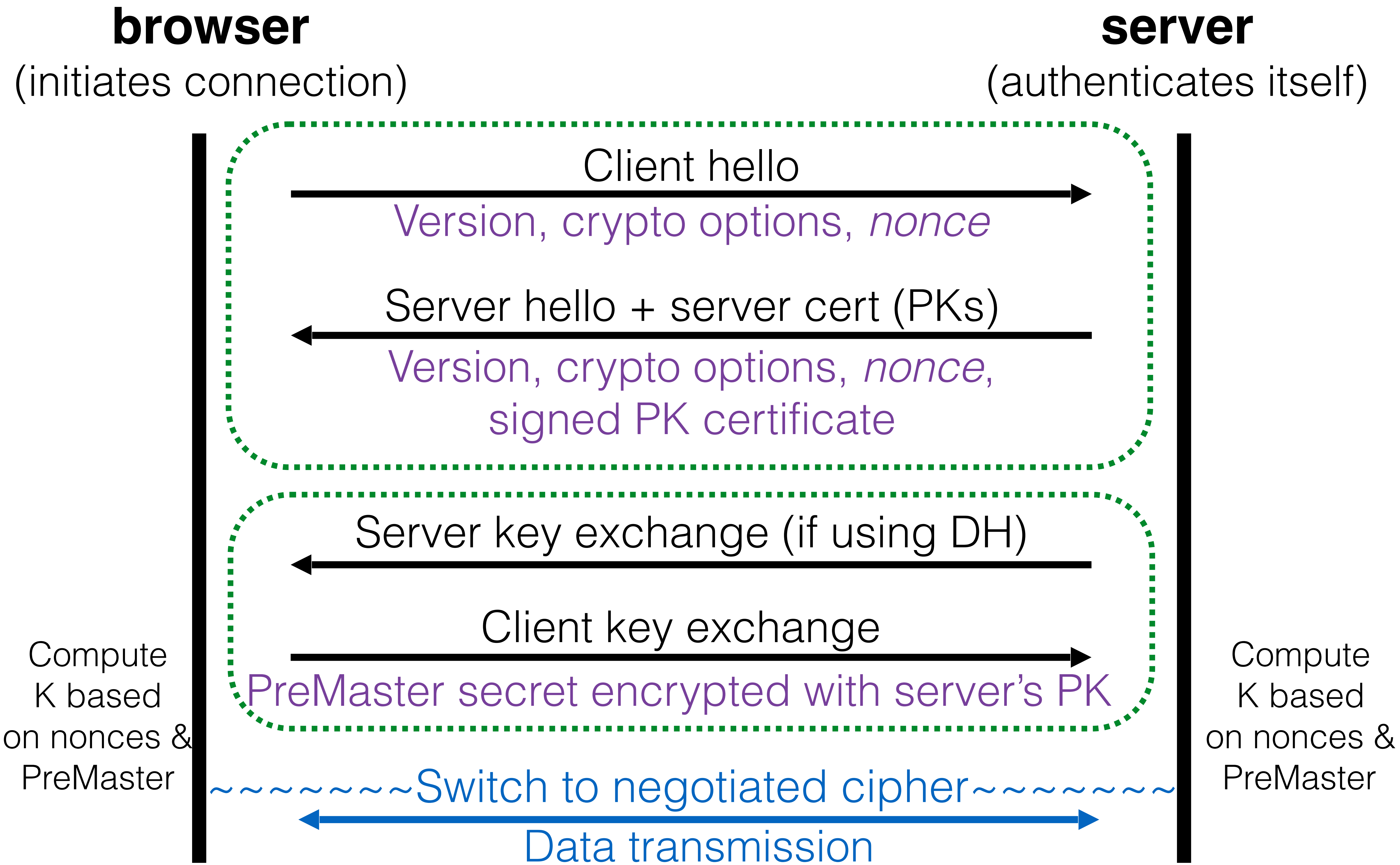
- Is there a way we could defend against opt-ack in a way that is still compatible with existing implementations of TCP?

- An important goal in networking is *incremental deployment*: ideally, we should be able to benefit from a system/modification when even a subset of hosts deploy it.

Transport layer security (TLS)

- Runs on top of TCP/IP
- Protocols for secure comms
  - Confidentiality with block and stream ciphers
  - Integrity with MACs
  - Authenticity with certificates
- Replacement for SSL (secure sockets layer)
  - Several problems including padding attacks

TLS protocol overview



HTTPS

- HTTP “on top of” TLS
- Pros: **Avoid MITM**
  - Includes e.g. reducing video quality, inserting ads
- Cons
  - Takes more time
  - Network service/ISP can’t compress or cache it
  - Network service/ISP wants to insert ads

https://www.eff.org/https-everywhere

Revoking certificates

- When you detect compromise or change keys, you have to notify the CA
- CA then ***revokes*** the certificate
  - Revocation list
  - Online cert status protocol
  - Short expiry times

Revocation list

- CA publishes list of revoked certs
- User (in practice, browser) must periodically download the newest list
  - Check when validating a certificate
- Vulnerability window since last list update
  - Or until certificate expires
- Can be beaten via DOS (why?)

Online certificate status

- During validation, ask CA whether cert is revoked
- Gets rid of vulnerability window
  - But can’t accept any cert if CA is not online!
- And, the CA gets to know where you browse

Short expiration

- Make all certificates have ***very short*** expirations (e.g. 10 min or less)
  - For the most part, renew automatically
- Revocation == decline to renew
- Expensive, not implemented that I’m aware of
  - Also some browsers accept expired certs

Trusting the Trusted Third Party

http://randomrock.com.br/randomrock/rock-n-movies-20-watchmen/

CA compromise

- 2001: Verisign issued two code-signing certificates for Microsoft Corporation
  - To someone who ***didn't actually*** work at MS
  - No functional revocation paradigm
- 2011: Signing keys compromised at Comodo and DigiNotar
  - Bad certs for Google, Yahoo!, Tor, others
  - Seem to have been used mostly in Iran
- Some CAs are less picky than others

Case study: Superfish (Feb 2015)

- Lenovo laptops shipped with “Superfish” adware
- Installs self-signed root cert into browsers
  - MITM on every HTTPS site to inject ads
- Worse: Same private key for every laptop
  - Password = “komodia” (company
- ***Lenovo*** ***“did not find any evidence to substantiate security concerns”***

http://www.sainteldaily.com/archives/11400

http://arstechnica.com/security/2015/02/lenovo-pcs-ship-with-man-in-the-middle-adware-that-breaks-https-connections/

Fixing rogue CA problems

- Limit which CAs can issue for which domains
- Certificate pinning
  - Browser, apps fix certain CA or cert for a server
  - Shipped with product, or on first use
  - Not always appropriate, hard to maintain

Fixing rogue CA problems (2)

- Broad surveillance
  - People on many networks report certs to Notaries
  - Check that others saw the same cert you did
  - Privacy implications
- Public unforgeable audit log
  - Uses crypto, Merkle hash trees
    - Only accept certs published in log
  - Same idea: ***Non-equivocation***
  - Being implemented now

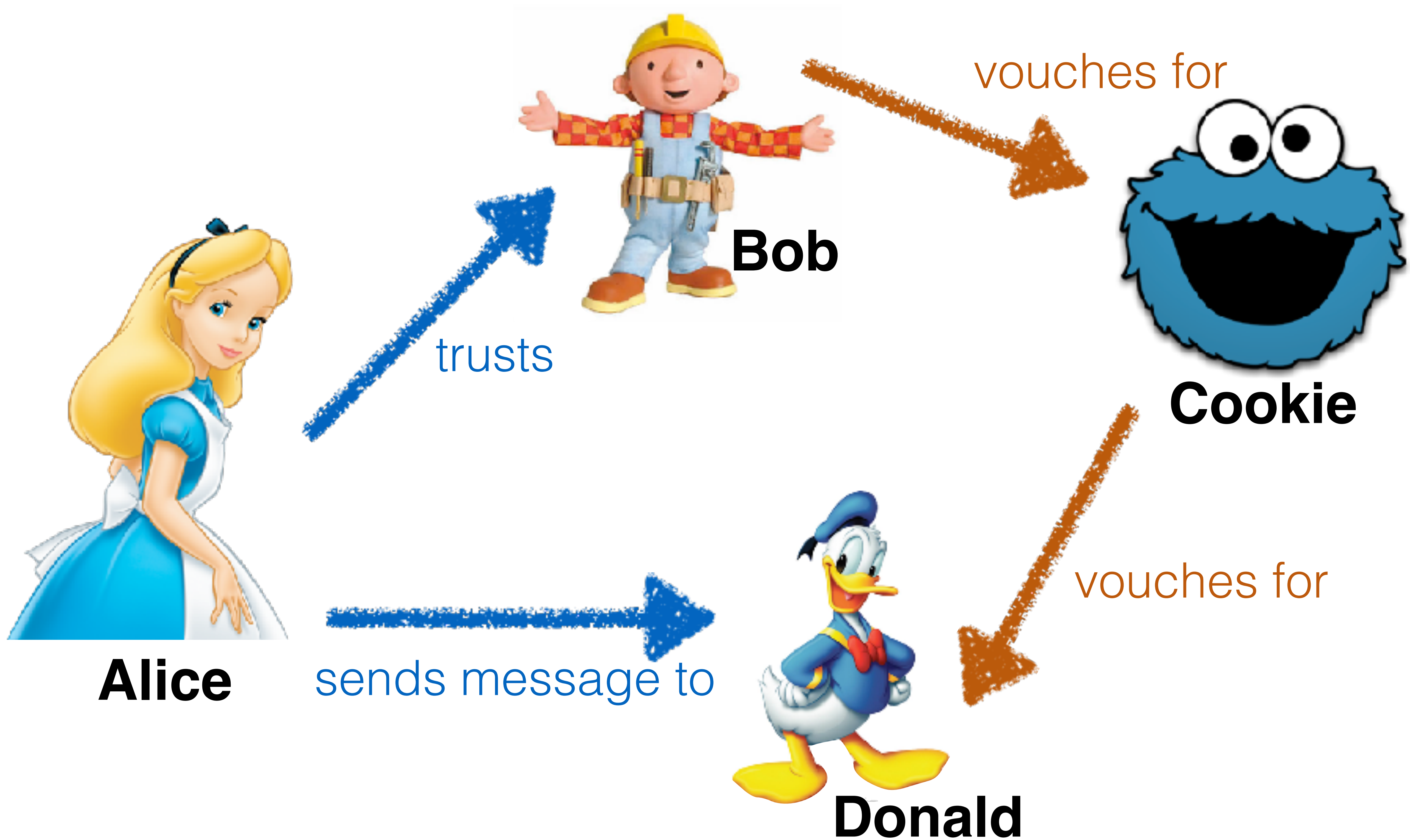
https://www.eff.org/observatory https://www.eff.org/sovereign-keys

Web of trust

Web of trust

- Alternative PKI — not hierarchical
  - Pioneered by PGP
- Don't rely on centralized authorities
- Everyone issues certificates for people they know

Trust chains in web of trust



A matter of trust

- Context:
  - Alice trusts Bob to diligently check identity
  - But Bob is only signing identity, not necessarily belief that Cookie is equally vigilant
- Transitivity: Alice trusts Bob, and Bob trusts Cookie.
  - But does that mean Alice should trust Cookie?
  - Trust for honesty == trust for good judgment?

Web-of-trust in practice

- Automatically find many such paths
  - More, shorter paths = higher confidence?
- Difficult to use
  - Still have bootstrapping problems
  - When should I agree to sign what?
  - Historically, serious UX problems as well