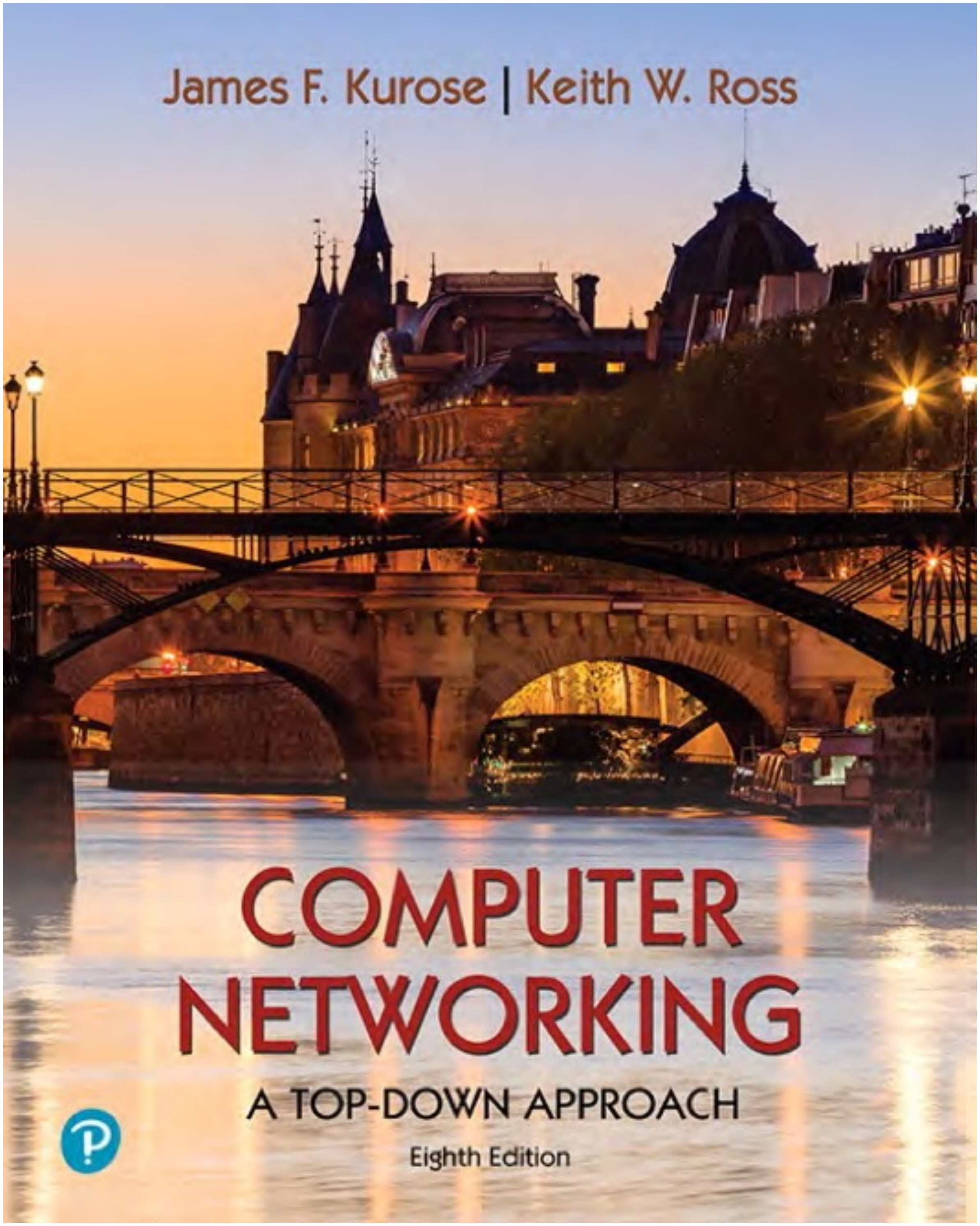
CS118:
Computer Network Fundamentals

Lecture-1: introduction

CS118: explains (roughly) how the Internet works

- Internet: a huge, complex network of networks
- Divide-and-conquer
  - Figure out how many major parts,
  - Learn one piece at a time
- Your job:
  - Read textbook, think, collect a list of questions
    - review every lecture slide deck after each class
  - Ask questions in class/office hours/via Piazza
  - Practice what you learn through homework and projects



Brief Contents

xix

Course assignment and due schedule

Midterm	In-class, Wednesday Feb 5 (**Location TBD**)	
Final	3:00PM-6:00PM Saturday March 21 (**Location TBD**)	
Homework	**Release:** on Thursday of week 1, 3, 5, 7; **Due:** 11:59pm Tuesday of week 3, 5, 7, 9.	
Project 0	**Release**: Monday Jan 6, 2024 (Week 1) **Due:** 11:59pm Wednesday, Jan 15, 2024 (Week 2)	1.5 weeks
Project 1	**Release**: Thursday Jan 16, 2024 (Week 2) **Due:** 11:59pm Sunday, Feb 16, 2024 (Week 6) **Grading:** auto-grading script (sample tests will be provided to let everyone test their code before submission)	4.5 weeks
Project 2	**Release**: Monday, Feb 17, 2024 (Week 7) **Due:** 11:59pm Sunday, March 16, 2024 (Week 10) **Grading:** auto-grading script (sample tests will be provided to let everyone test their code before submission)	4 weeks

FOR ALL OTHER COURSE INFO, PLEASE SEE HTTPS://BRUINLEARN.UCLA.EDU/

Course workload and grading

- Bi-weekly homework assignments
- 3 programming projects,
  0. UDP socket (individual)
  1. Reliable data delivery (2-3 people team)
  2. Secured reliable data delivery (2-3 people team)
- Midterm and final exams (cheat sheets allowed, 2 pages double sided)
- **Strict Grading Policy**
  - Homework: do it yourself; no credit for late submission
  - Project: 20% credit reduction per late day
  - *No make-up exam*

Homework	20%
Programming Projects	25% (5/ 10/ 10)
Midterm	25%
Final exam	30%

2% extra credits based on piazza

1% extra credits course evaluation and TA/LA feedbacks

Class Policy

The following actions are **strictly prohibited**

- Posting/sharing/selling class material, with or without answers, to anyone outside this class, during or after this quarter.
- Use of old homework/midterm/finals in doing homework or exams
- Use ChatGPT in doing assignments
- Making your project code publicly available either during or *after* this quarter
  - *you must use private repository* on GitHub or GitLab

Hints for Getting Good Grade

- Review previous lecture slides
- Read textbook before coming to each lecture
- *Ask questions*
- Get your work done early
  - Lecture slides uploaded to BruinLearn *one day before the lecture*
  - Get HWs and projects done ***before*** the deadline

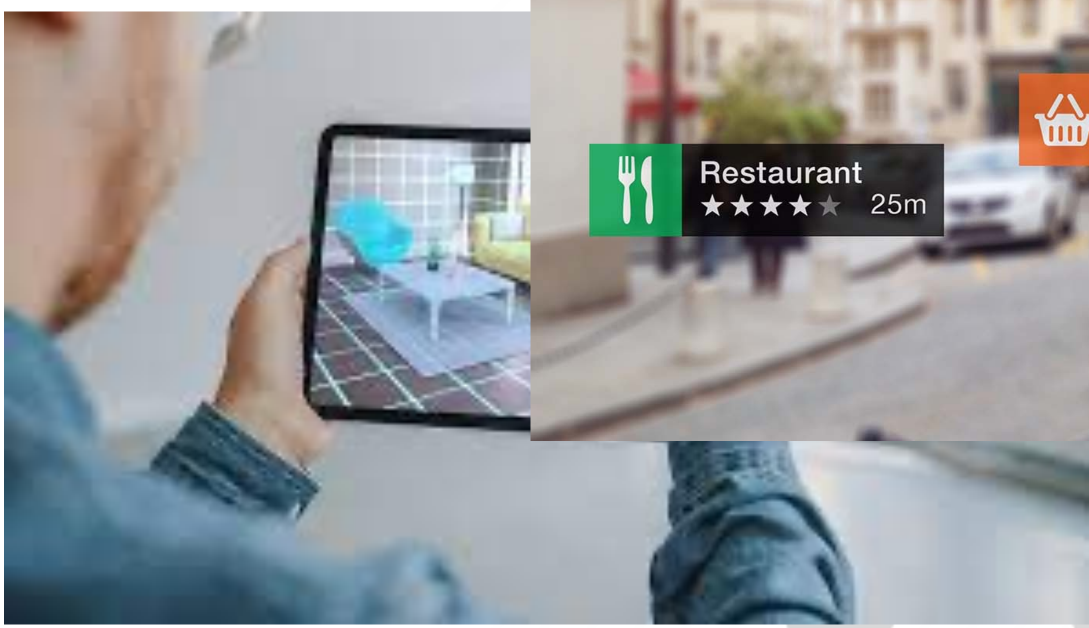
Let's get started

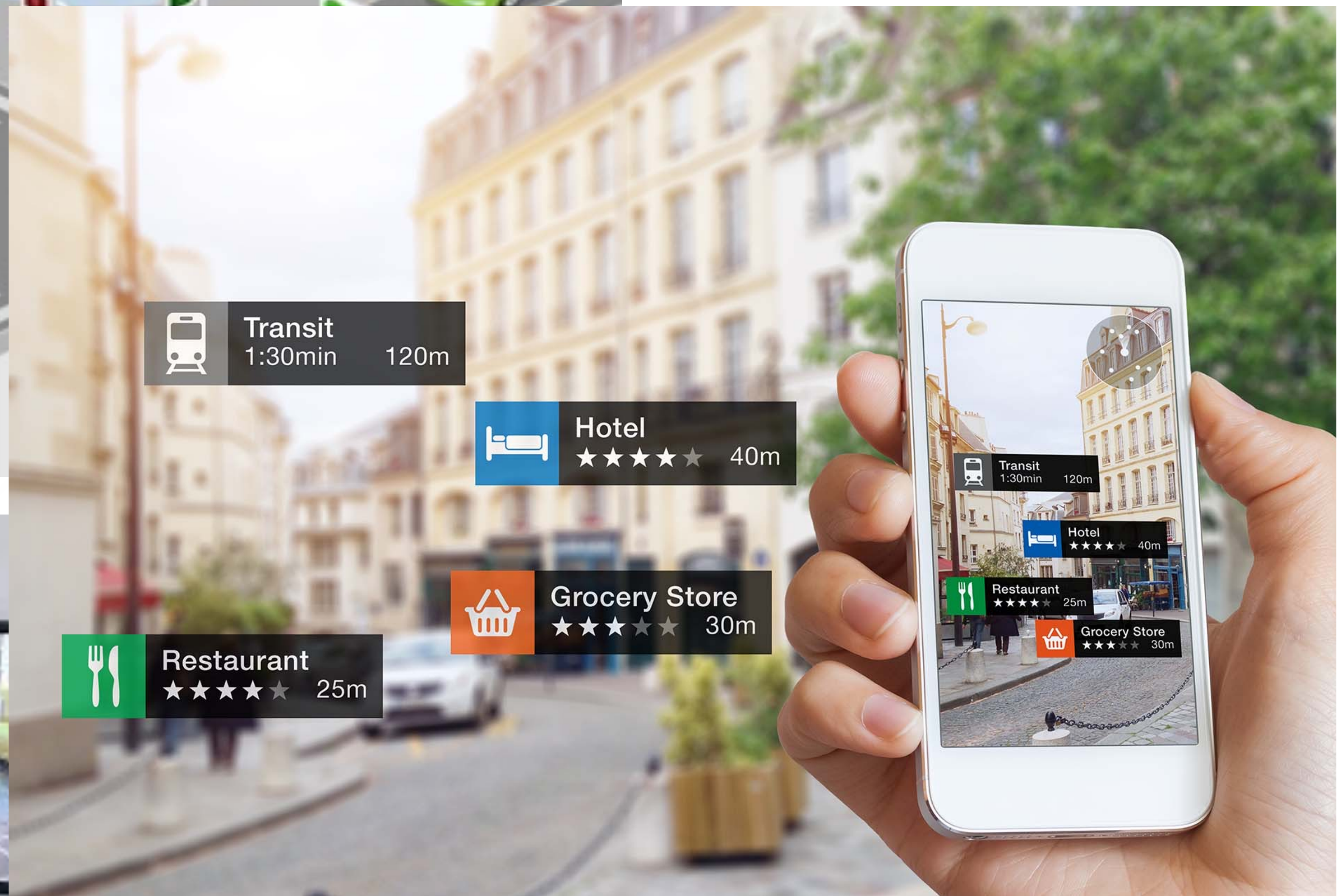
Today we cover the basic concepts in Chapter 1 of the textbook

What is a Computer Network

What is a Computer Network

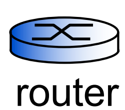
Coming to you ....



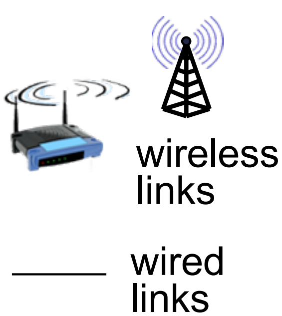


Terminology

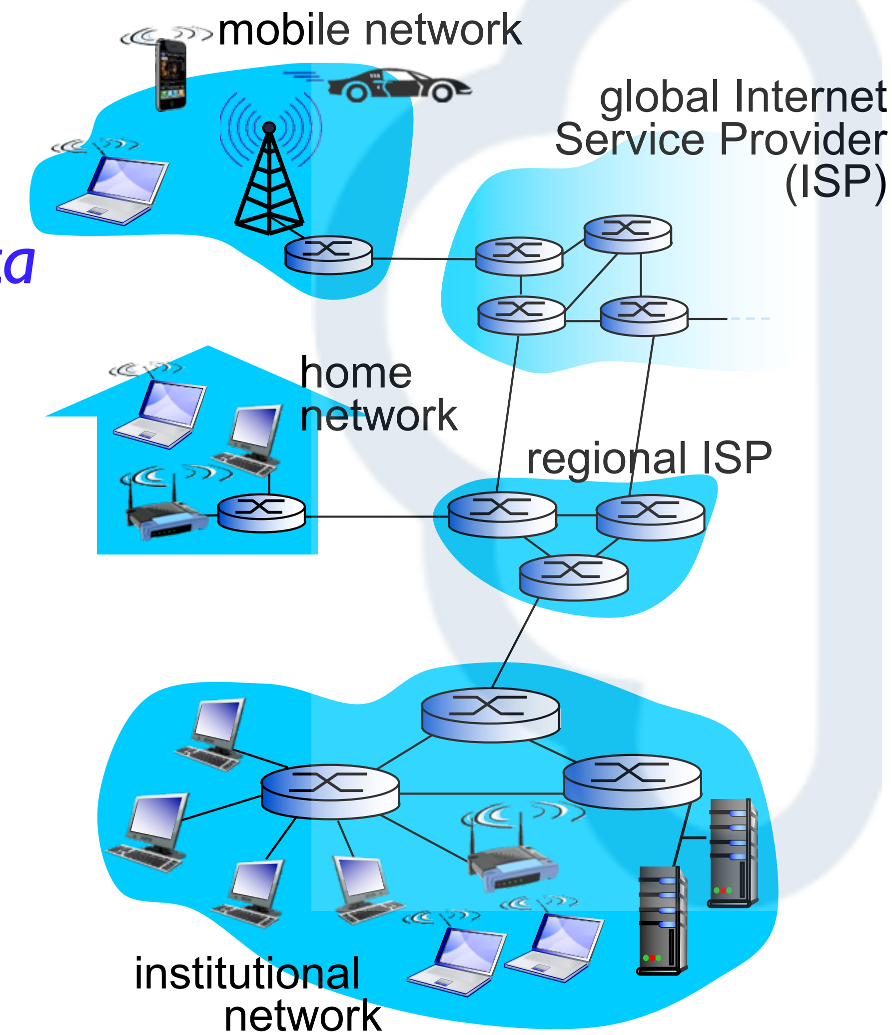
- billions of connected computing devices:
  - *hosts* = *end systems*
  - running *network apps*
  - Apps send/receive *data packets*



- *Routers* = *packet switches* inside network



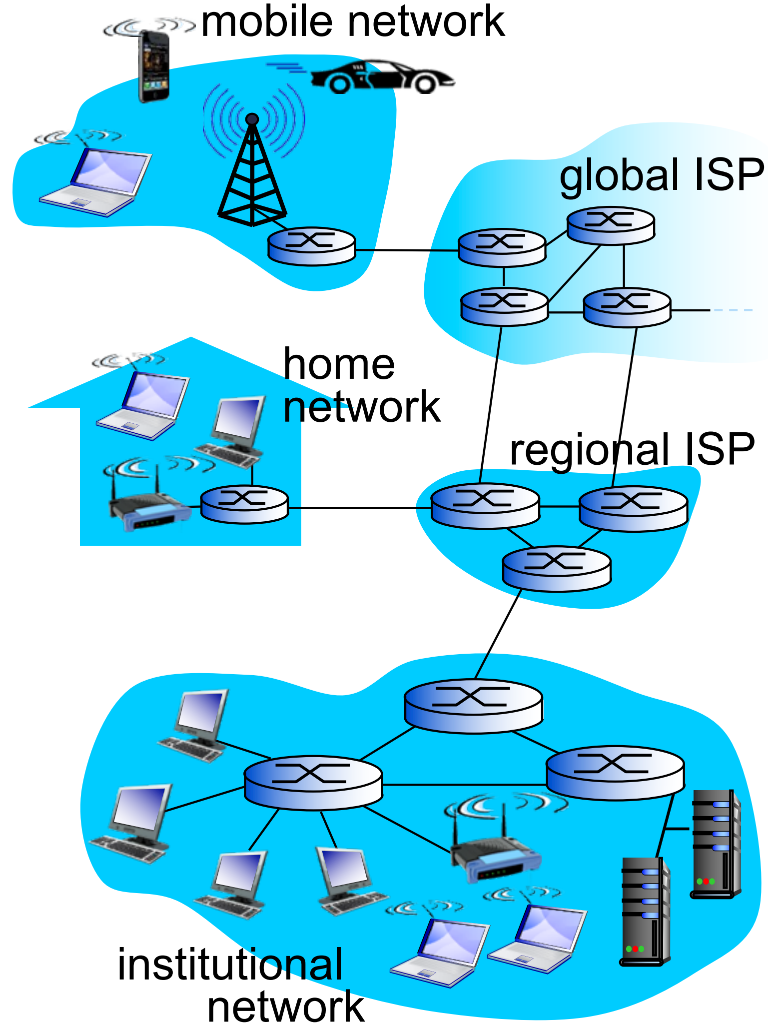
- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate = *bandwidth (BW)*



Recent years witnessed rapid growth of giant cloud service providers

"Nuts and Bolts"

- *Internet:* "network of networks"
  - Interconnected ISPs, enterprise networks, now also cloud service providers
- *Protocols:* define how to send, receive packets
  - e.g., HTTP, TCP, IP, 802.11
- *Internet protocol standards*
  - RFCs: "Request for Comments"
    - https://www.rfc-editor.org/rfc-index.html
    - Developed by Internet Engineering Task Force (IETF)
  - IEEE Standards
  - W3C (World Wide Web Consortium), and others



What is a protocol?

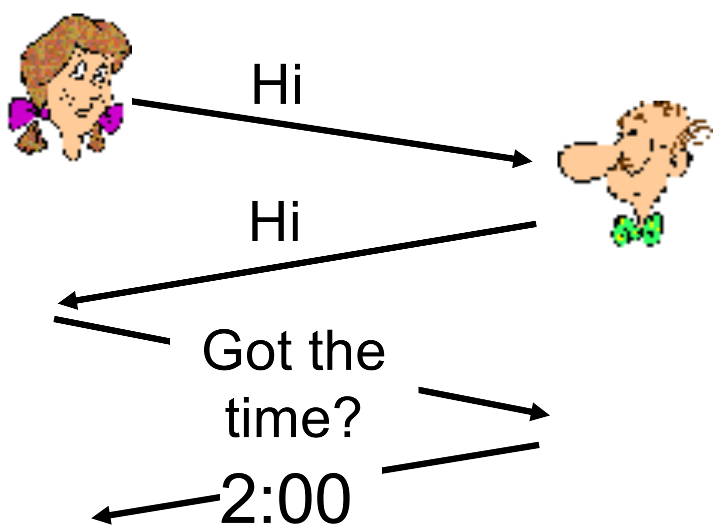
Traffic light protocol

- Green: go
- Red: stop
- Yellow: slow down - stop

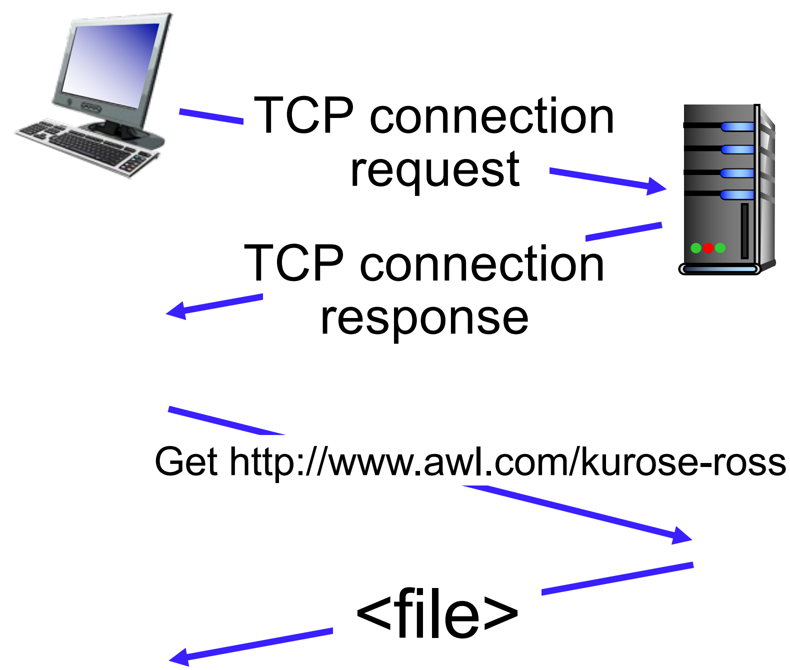
… specific messages sent

… specific actions taken when the messages received

human protocols:

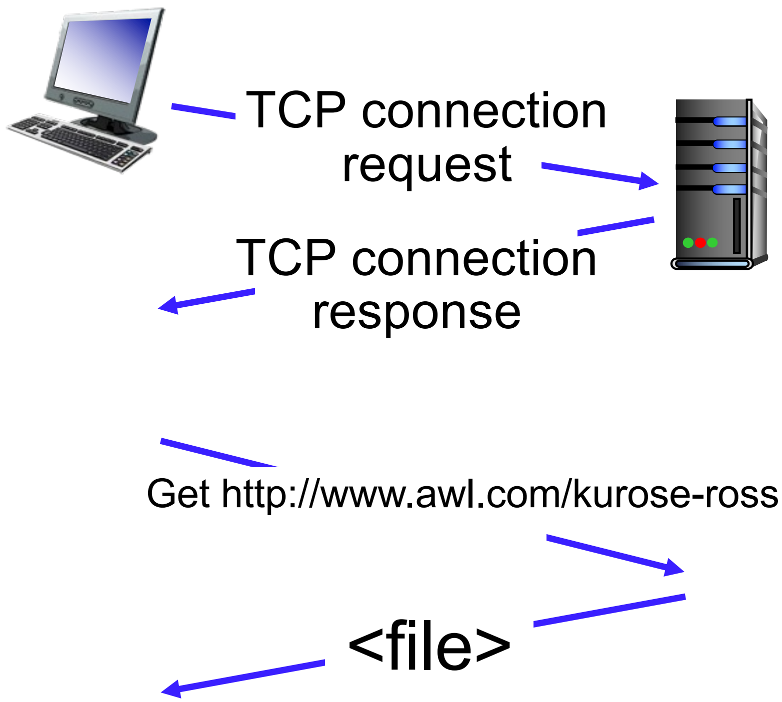


computer protocols:



Internet protocols

computer protocols:



- Communication between machines rather than humans
- all communication activity governed by protocols

> *protocols* define *format*, *order* of *packets sent and received* among network entities, and *actions taken* on packet transmission, receipt

Delivering data over the global Internet is a complicated process, involving many many steps

How to get the work done: divide and conquer

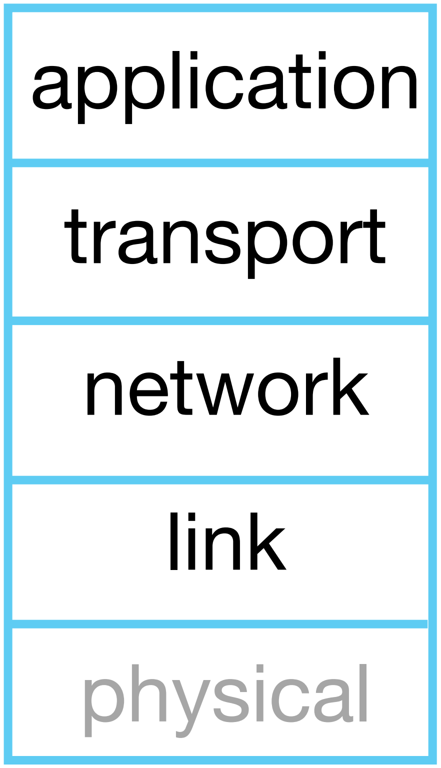
Group functions to a few modules

How many?

Internet protocol stack

- Application layer protocols
  - Support data exchange between application processes
  - Example: SMTP, HTTP, DNS

    (Simple Mail Transfer Protocol)
- Transport layer protocols
  - handling delivery reliability, multiplex within a host
  - Example: TCP, UDP
- Network layer protocols
  - forward packets from source to destination
  - Example: IP
- Link layer protocols
  - transfer data between directly connected network elements
  - Example: Ethernet protocol, WiFi
- Physical layer: bits “on the wire”

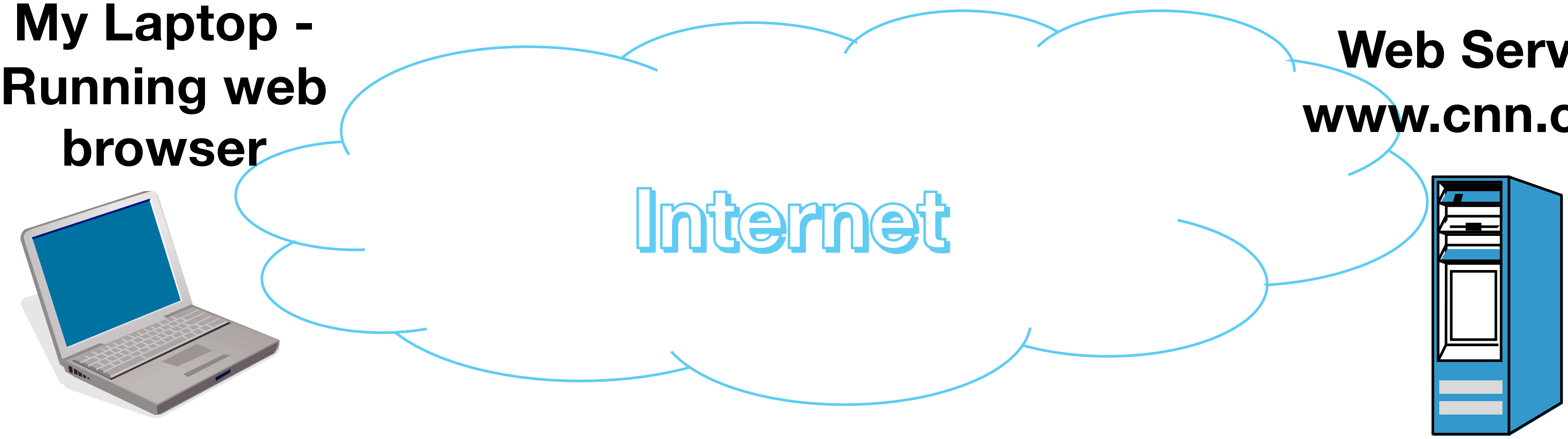


Application View

apps
trans
net
link
phy

My Laptop - Running web browser

Web Server www.cnn.com



client
(Chrome, Safari, Firefox, ...)

server
(Apache, GWS, ...)

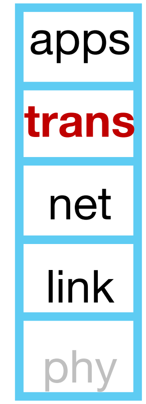
These are *application programs*

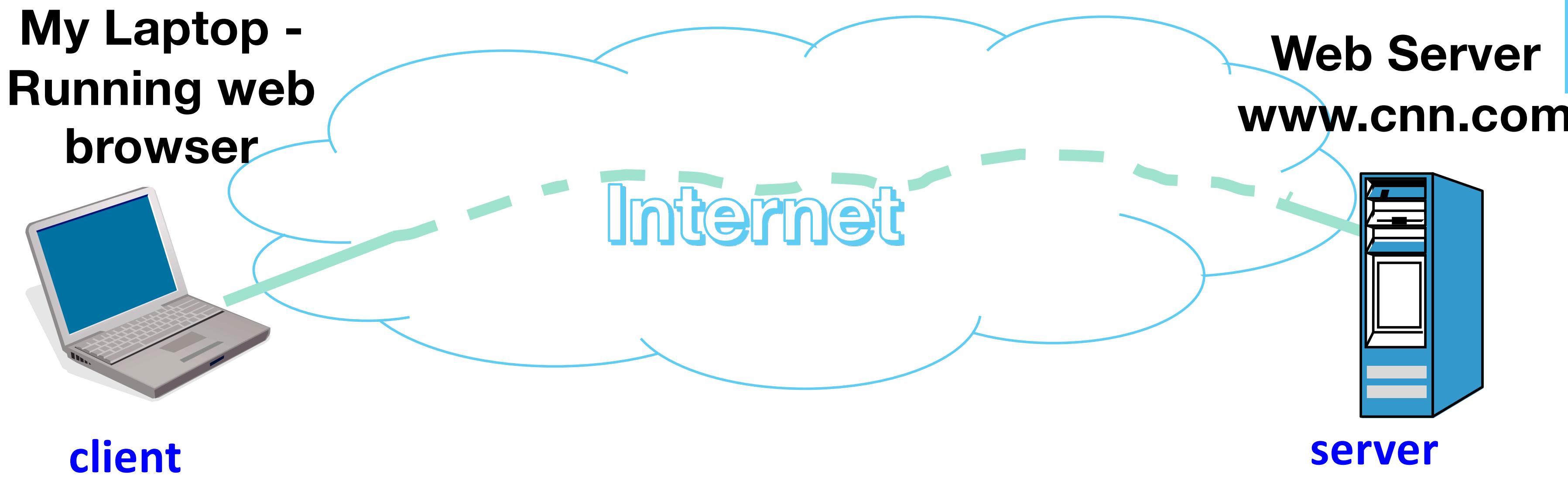
They talk to each other using *application protocols* (web protocol: HTTP)

Application protocols

- Assume network can send data to any hosts on the Internet
- Don't know/*care* <u>how</u> data is sent, and assume all data delivered reliably
- Runs on top of a transport protocol

Transport View

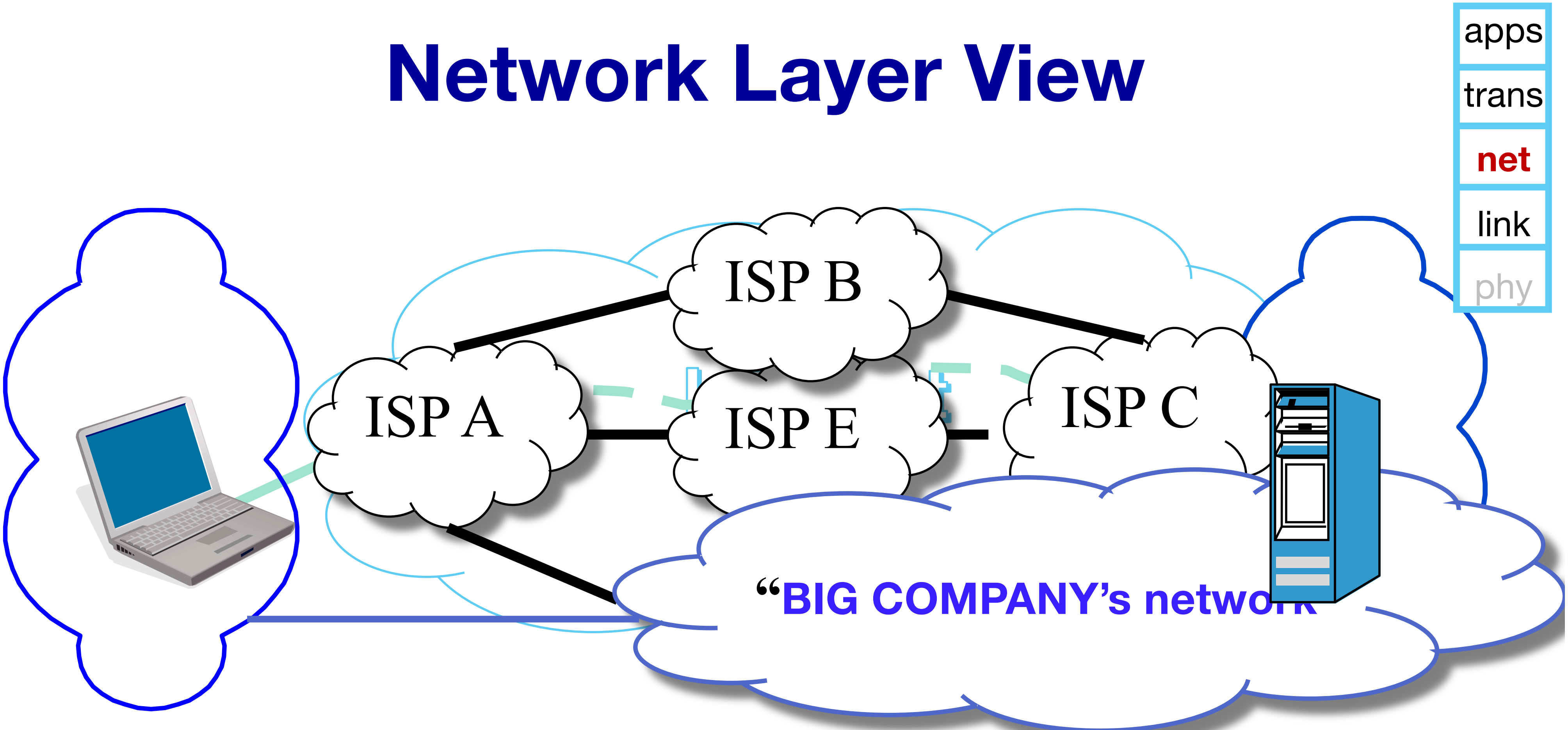




- **A transport protocol**'s job: **delivering data** between the two communicating ends
  - *Don't know or care about which paths data may traverse through the network*
- Multiple transport protocols exist, each offers somewhat different functions (e.g. reliability, congestion control)

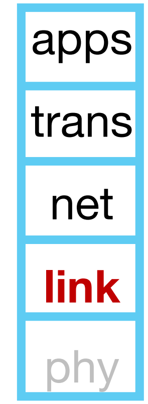
Actually, transport protocols don't do delivery → network protocol's job

Network Layer View



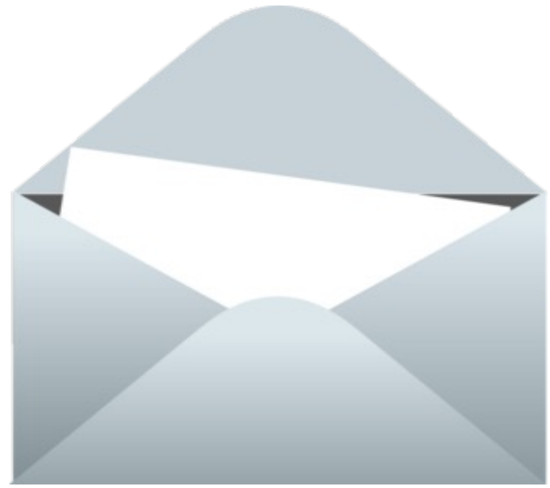
- **network protocol**'s job: forward packets from source to destination host
- A really hard problem: the Internet is large, run by many different parties
  - connection from laptop to CNN.com:
    WiFi → campus backbone → local ISP → other ISP → CNN website

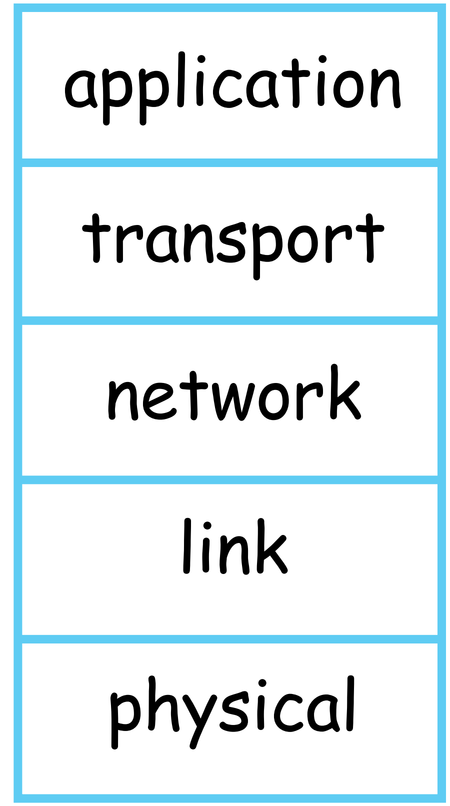
Link Layer View

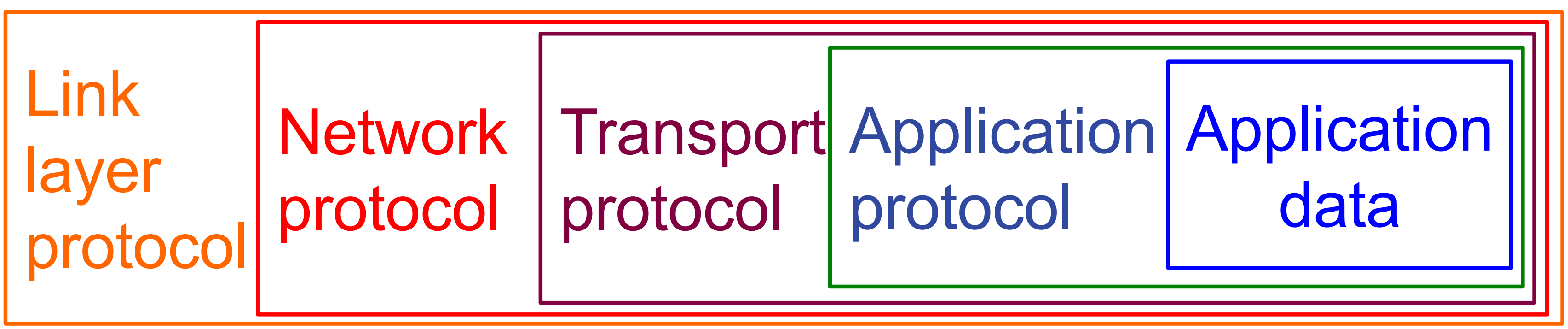


- **Link layer**'s job: Get a packet transmitted across some communication medium to **next hop**
- Different medium → different link layer protocol

What protocol "layer" really means







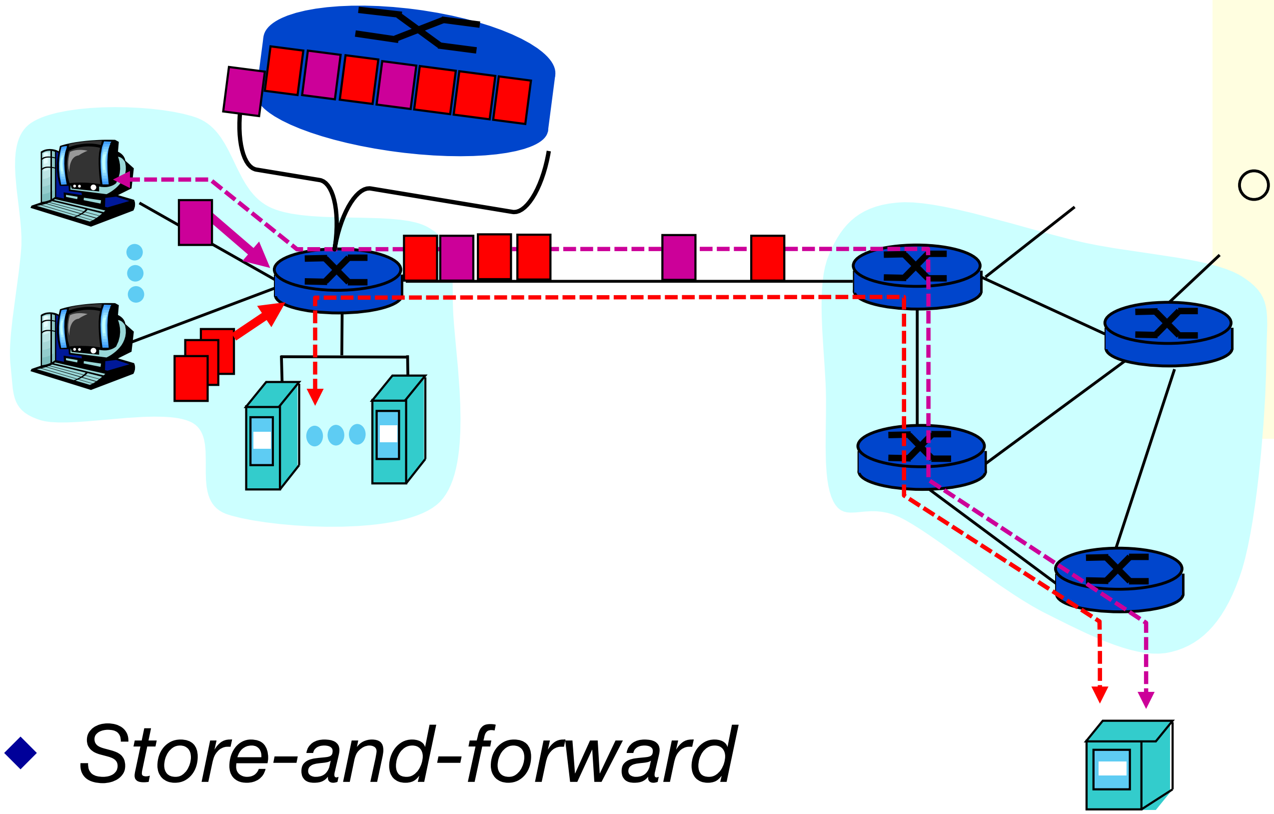
(Tentative) **Schedule of the Quarter**

Week:	1	2	3	4	5
Mon	**1/6** Course intro BW& delay 1	**1/13** HTTP 2	**1/20** Martin Luther King Jr. Day	**1/27** Transport protocols 3	**2/3** Congestion Control 3
Wed	**1/8** Socket programming, Web & HTTP 2	**1/15** DNS 2	**1/22** DNS 2	**1/29** TCP 3	**2/5** **Midterm**

	6	7	8	9	10	
Mon	**2/10** Security 101	**2/17** Presidents' Day	**2/24** Routing algorithms & protocols 5	**3/3** Routing in the Internet 5	**3/10** Hubs and switches 6	**3/21: Final Exam**
Wed	**2/12** Internet Protocol (IP) 4	**2/19** Addressing, NAT, IPv6 4	**2/26** Routing algorithms & protocols 5	**3/5** Link layer (Ethernet) 6	**3/12** Course review	

- The big yellow numbers indicate the chapter numbers in the textbook.

Packet Switching: *Statistical Multiplexing*



- Each node sends packets as soon as link available
- Receiver gets a full packet first, then forwards it towards the destination

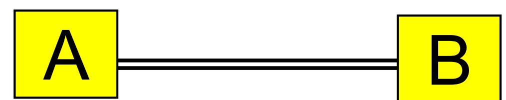
- *Store-and-forward*
- Packet switch can temporarily buffer up packets
  - Introduce *delay*
  - Packets get *dropped* when the queue is full

Network Performance

- 3 basic measurements
  - Throughput (bits/sec, Kbps=1000 bits/sec, Mbps)
  - Loss rate (% of packets lost)
  - Delay (sec, msec)

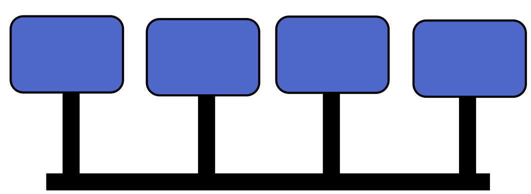
Throughput

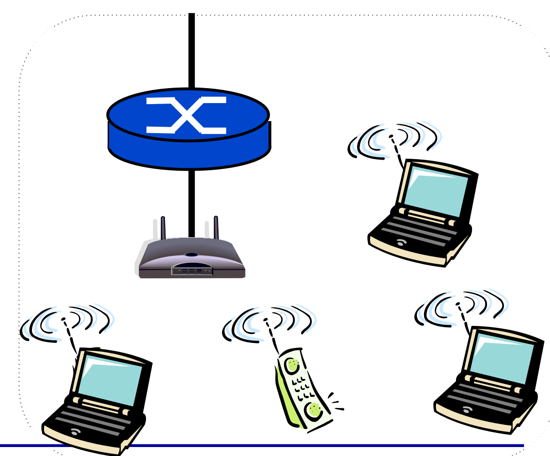
- over a single link: point-to-point



  - Pumping data into the pipe: throughput = link bandwidth
- Multi-access:

  a lot more difficult to measure, *Why?*

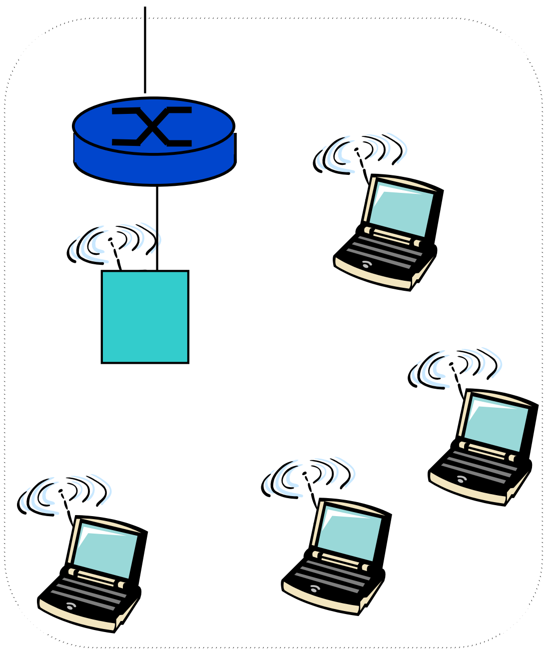




Packet Losses

- Wired links
  - Loss due to transmission errors
  - Loss due to congestion
- Wireless links
  - Limited transmission rate
  - Higher (than wire) bit error rate
  - Host mobility: high variance in the number of hosts sharing the same wireless channel

Do users know there are packet losses?
Do users' performance get affected by losses?



Delay in packet-switched networks

4 sources of delay at each hop

- node processing:
  - check bit errors
  - determine output link
- Queuing = #packets in queue X transmission time of each packet

Transmission = Length / rate

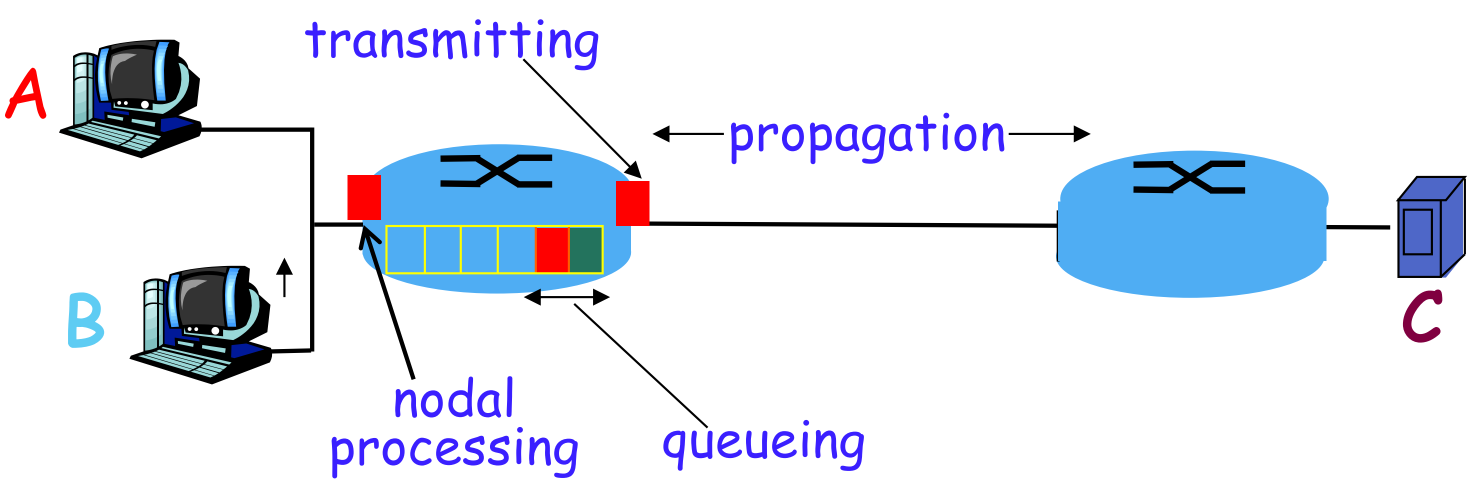
R = link bandwidth (bps)

L = packet length (bits)

Propagation = distance/sec

d = length of physical link

s = propagation speed in medium (~2×10^8 m/sec)

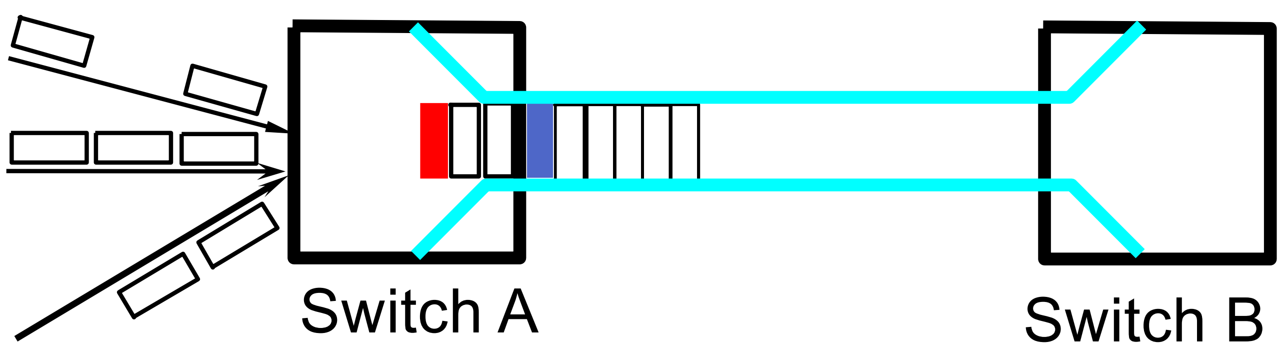


Example: calculating one hop delay

total delay (A→B) = ?

- Queuing delay = ?
- transmission delay = ?
- Propagation delay = ?

link length = 100 km
Bandwidth= 1 Mbps
packet size= 1000 bits
(all pkts equal length)



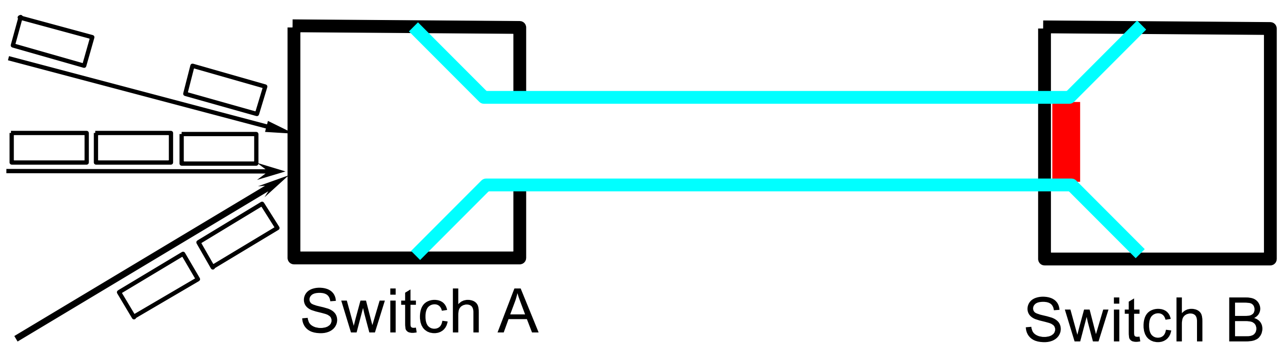
(2.0x10^8 meters/sec in a fiber)

Example: calculating one hop delay

total delay (A→B) = $1ms \times 2 + 1ms + 0.5ms = 3.5ms$

- Queuing delay = **Waiting time for 2 pkts**
- transmission delay = $\frac{1000bits}{1000000bits/\sec}$ = **1 msec**
- Propagation delay = $\frac{100{,}000m}{2 \times 10^8 m/\sec}$ =**0.5 msec**

link length = 100 km
Bandwidth= 1 Mbps
packet size= 1000 bits
(all pkts equal length)



(2.0x10^8 meters/sec in a fiber)

Transmission vs. propagation delay

Transmission delay: L / R

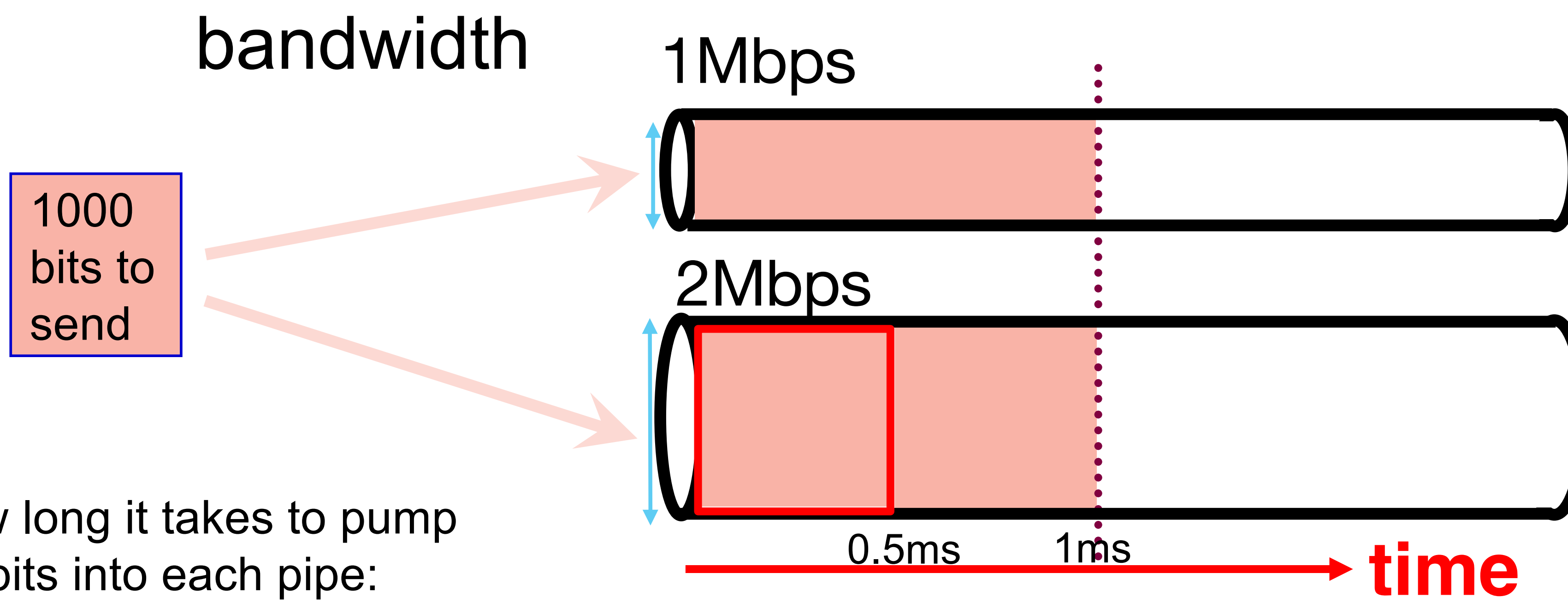
R = link bandwidth (bit-per-second, bps)

L = packet length (bits)

Propagation: d / s

d = length of a physical link

s = signal's propagation speed in the medium (~2×10^8 meter/sec)

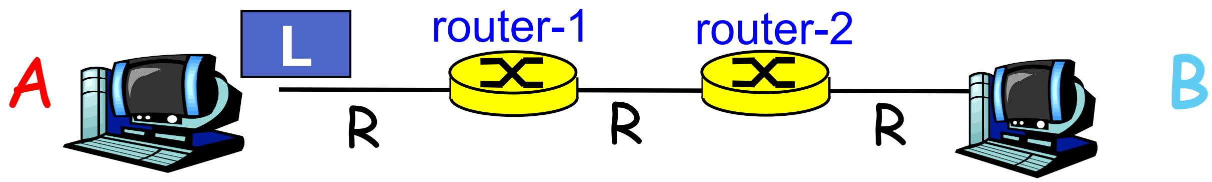


How long it takes to pump 1K bits into each pipe:

(1) $\frac{10^3 bits}{10^6\ bps}$ =1msec

(2) $\frac{10^3 bits}{2\times10^6\ bps}$ =0.5msec

Packet-switching: store-and-forward

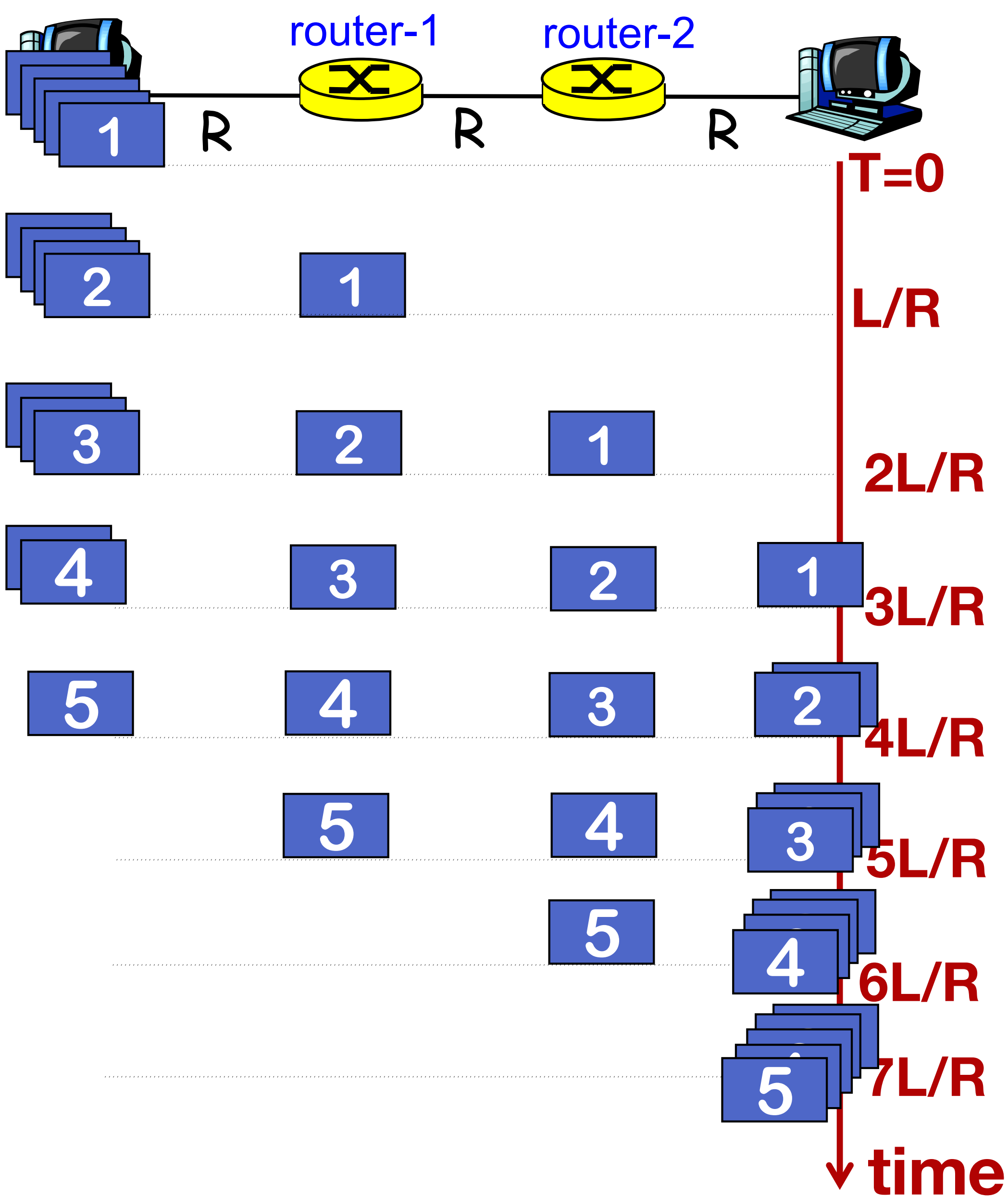


- Takes L/R seconds to transmit (push out) packet of L bits on to link of R bps
- Entire packet must arrive at router before it can be transmitted on next link: *store and forward*

Example 1: send L A → B

- L = 8000 bits (1000 bytes)
- Bandwidth R = 2 Mbps
- *If ignore propagation delay:* i.e. when last bit of a packet left A, it arrives at router-1 instantly
  delay = 3xL/R = 12 msec

Packet-switching: store-and-forward

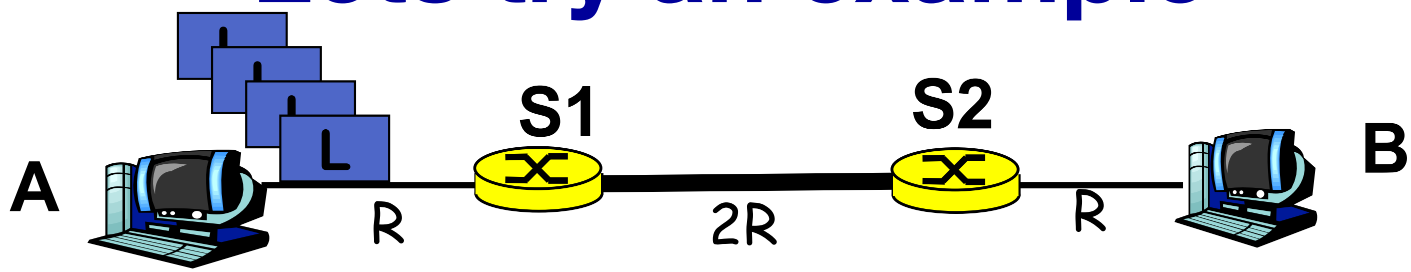


Example 2:

- A sends 5 packets to B
- L = 8000 bits, R = 2 Mbps
  - *Ignore propagation delay*
- How long does it take starting from A sending the first bit of first packet till B receives the last bit of the last packet?

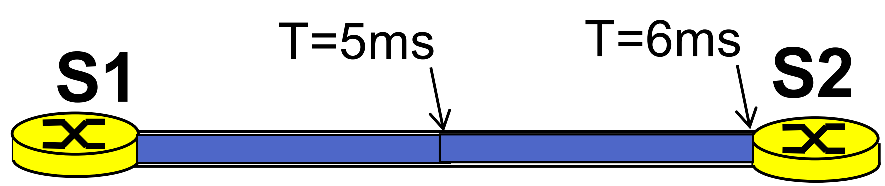
What if one takes into account the propagation delay?

Lets try an example

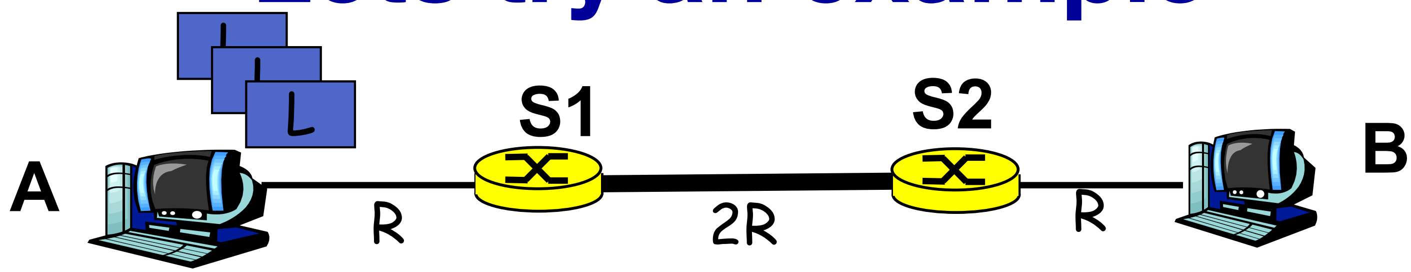


- L = 4000 bits, R = 2 Mbps, A sends 4 packets
  - L/R = 2msec, L/2R = 1ms
- Propagation delay: 2msec for each link
- When will first packet get to S2?
  - A⇨S1: L/R + $D_{propagation}$ = 2ms+2ms
  - S1⇨S2: L/2R + $D_{propagation}$ = 1ms+2ms

When 1st packet arrives at S2, where is the 2nd packet?



Lets try an example



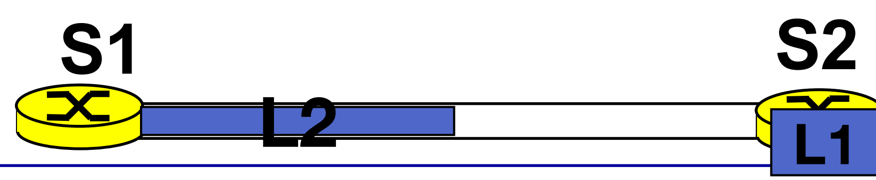
- L/R = 2msec, L/2R = 1ms
- Propagation delay: 2msec for each link
- When 1st packet arrives at S2, where exactly is the 2nd one?

T=7ms

T=4ms: last bit leaves A

T=6ms: last bit arrives S1

T=7ms: last bit leaves S1



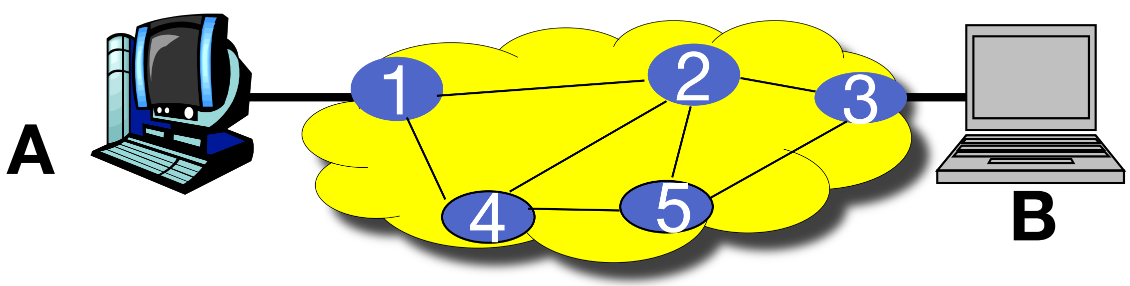
Network latency

- The time to send 1 packet from host A to B
  - sum of delays across each hop along the path

$$Delay_{A-B} = Delay_{A-1} + Delay_{1-2} + Delay_{2-3} + Delay_{3-B}$$

- **RTT**: round-trip-time

$$RTT_{AB} = Delay_{A-B} + Delay_{B-A}$$



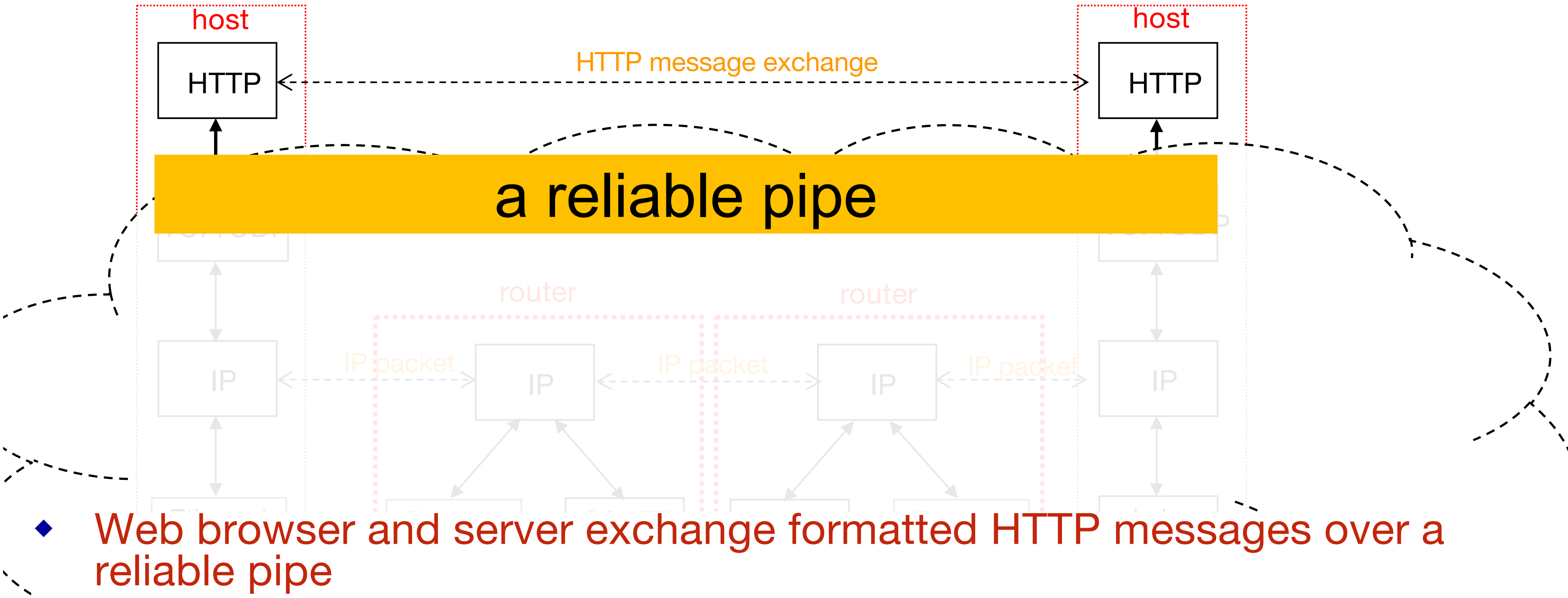
What we covered today

- Internet: made of a huge number of hosts, routers, wired and wireless links
- Hosts: run application protocols to exchange data packets with each other
- Routers: run bunch of protocols to move all packets towards their destinations
- Why protocols are layered
- How to calculate packet delays as they move across a packet-switched network

Lecture 1 Review:
layered protocol architecture

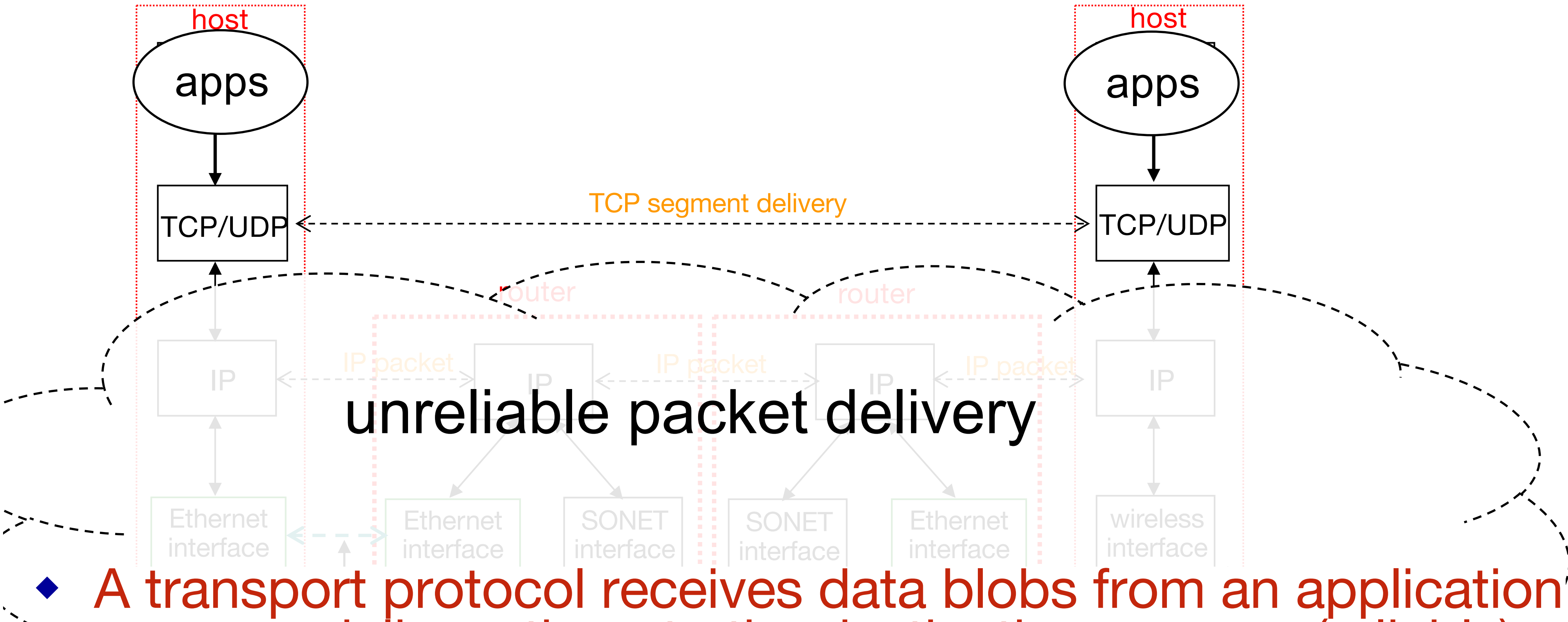
- Concepts:
  - **Internet**: made of a huge number of hosts and routers, interconnected by physical and wireless links
  - **Host**: a computer running applications and bunch of protocols to let apps exchange data with each other
  - **Router**: a packet switch running bunch of protocols to move packets toward their destinations
- Protocols are organized in layers:
  - Application protocols
  - Transport protocols
  - Network protocols
  - Link layer protocols
  - Physical layer
- How to calculate packet delays as they move across one hop

Application protocol’s view of the world



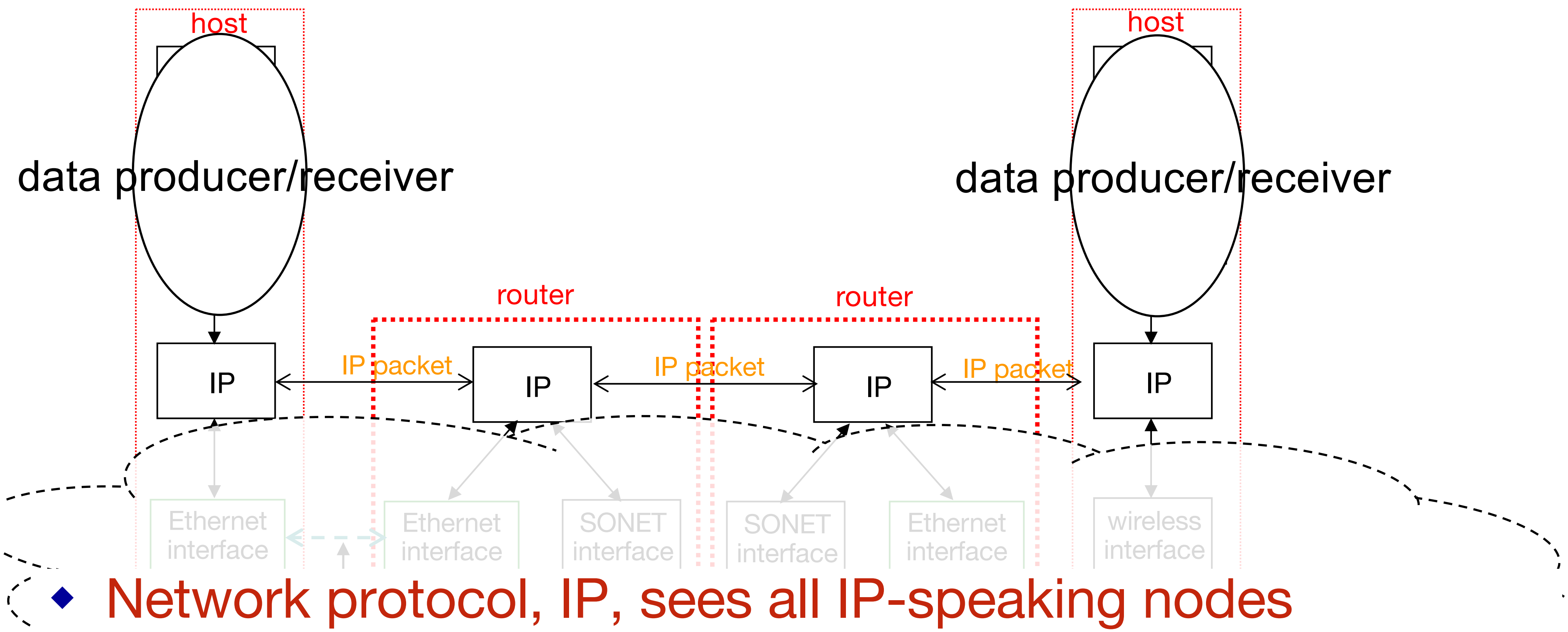
- Web browser and server exchange formatted HTTP messages over a reliable pipe
- As an application protocol, HTTP only concerns with the message’s presentation format
- Application decides where msgs should be delivered to
  - The receiving end is identified by its name, which gets translated to IP address

Transport protocol's view of the world



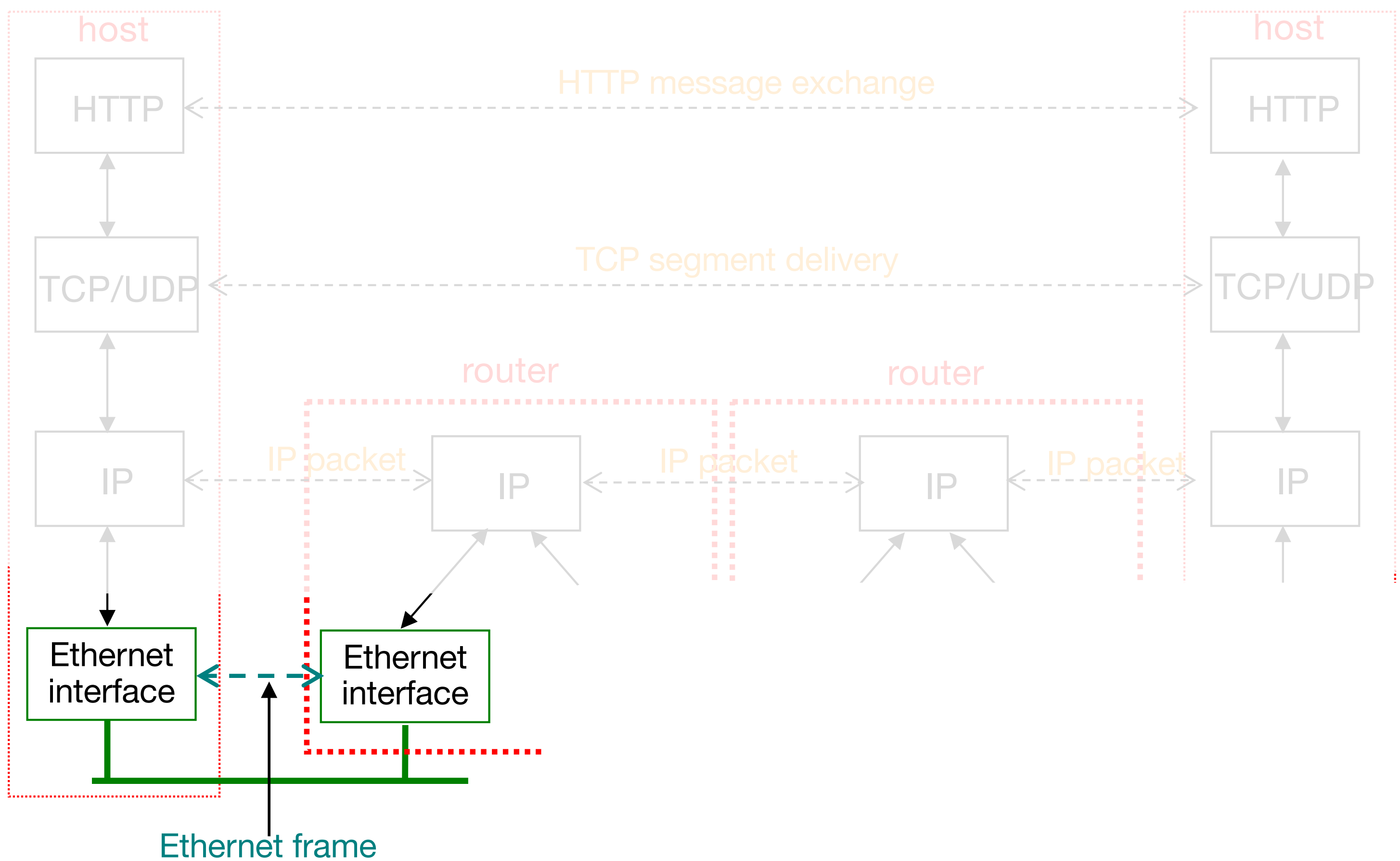
- A transport protocol receives data blobs from an application process, delivers them to the destination process (reliably)
  - Dest. Process is identified by IP address + (trans)port number
- It runs between two processes over an unreliable network (where packets can be garbled, lost, or reordered)

Network protocol's view of the world



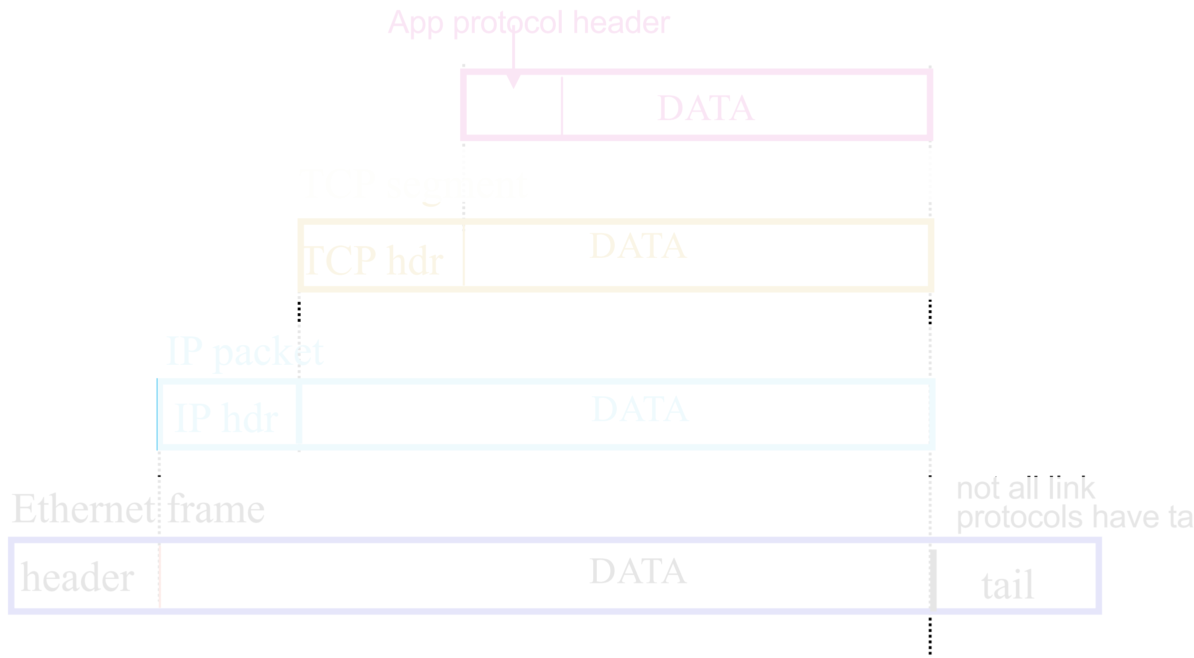
- Network protocol, IP, sees all IP-speaking nodes
- It receives data segments, delivers each of them to its destination IP address (with its best effort)
  - A router forwards packets, without looking inside IP envelope

Link layer protocol's view of the world



- A link layer protocol delivers data frames between two physically connected nodes
  - A link-layer header is added at sending node, removed by the receiving node
  - When a packet moves through the network across multiple hops: link-layer header is added and removed multiple times

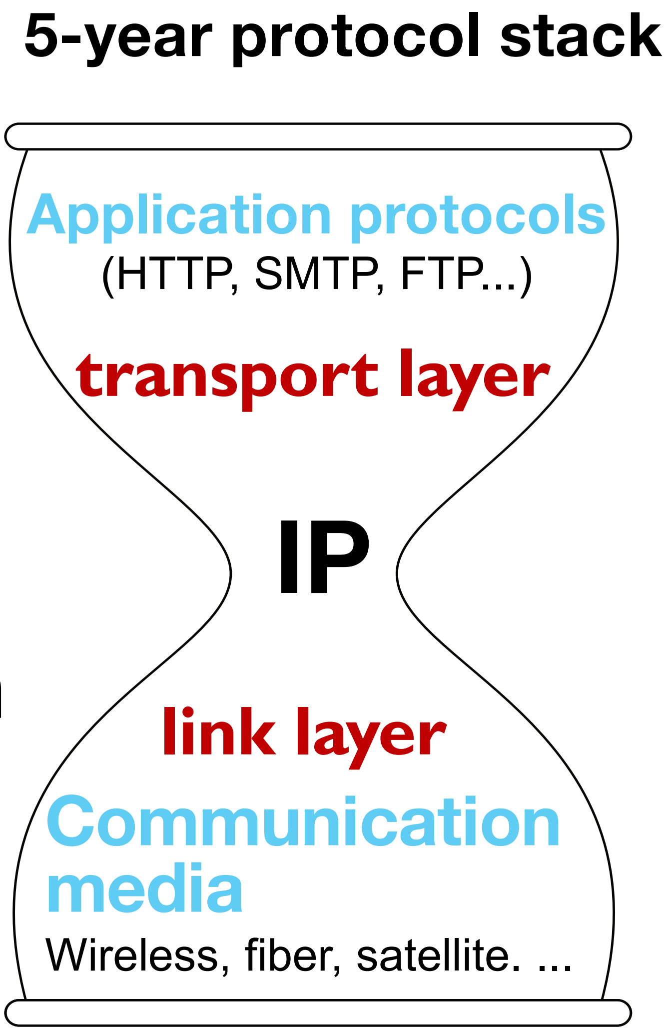
Layered protocol implementation



- protocol header: contains the information one writes on the "envelope"
- all the information, and *only* the information, that's needed to carry out the protocol's functionality

One more question: why 5 layers?

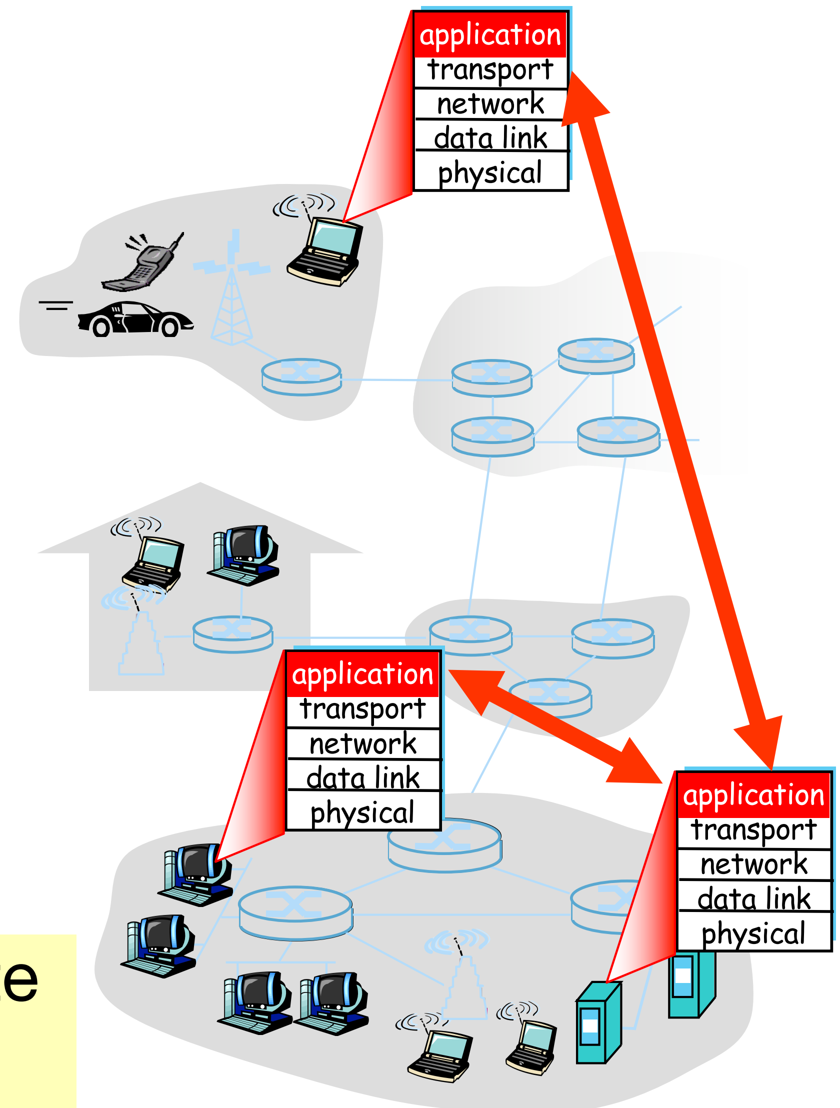
- Two layers are taken as given
  - Multiple different **application protocols**
  - Multiple different **physical communication media types**
- **IP**: the span layer
  - Connecting up all nodes
- **Link layer**: adaptation between IP and physical media
- **Transport**: adaptation between what apps want and what IP offers



Network applications:
how different parties reach each other

Some popular network applications

- Email
- Web
- WhatsApp
- BitTorrent (P2P file sharing)
- Online Games
- YouTube
- Virtual Conferencing



Application processes communicate with each other using application protocols

Client-server application communication model

servers:

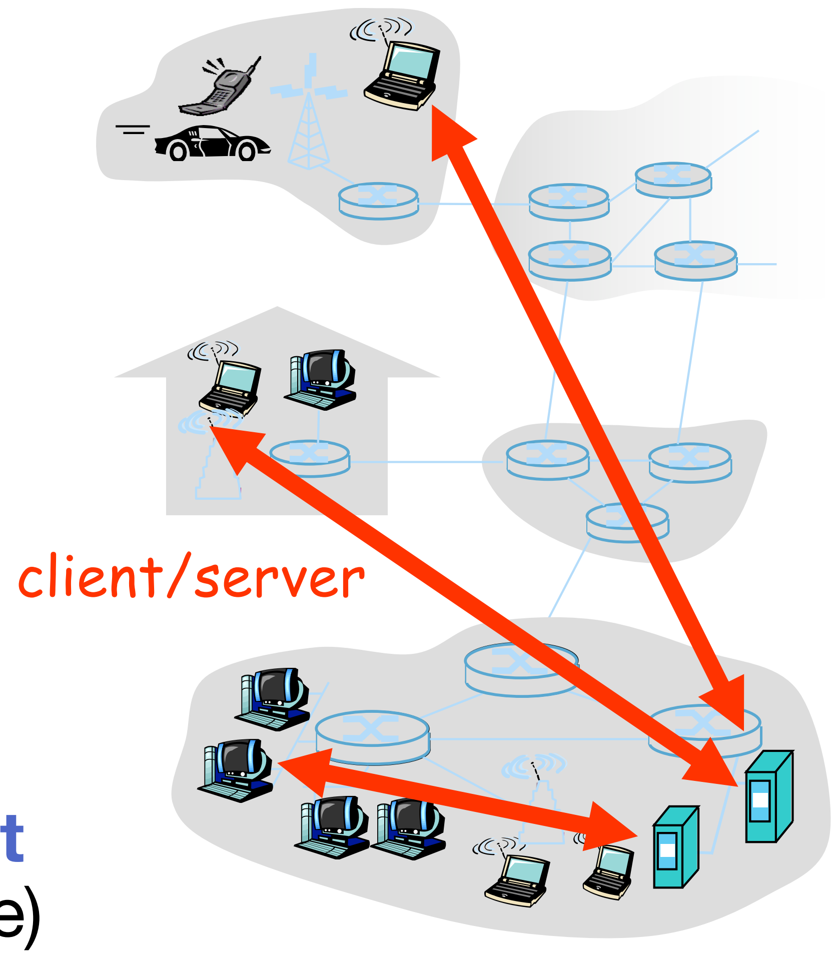
- Reachable by IP address
- **always-on**, waiting for incoming requests from clients

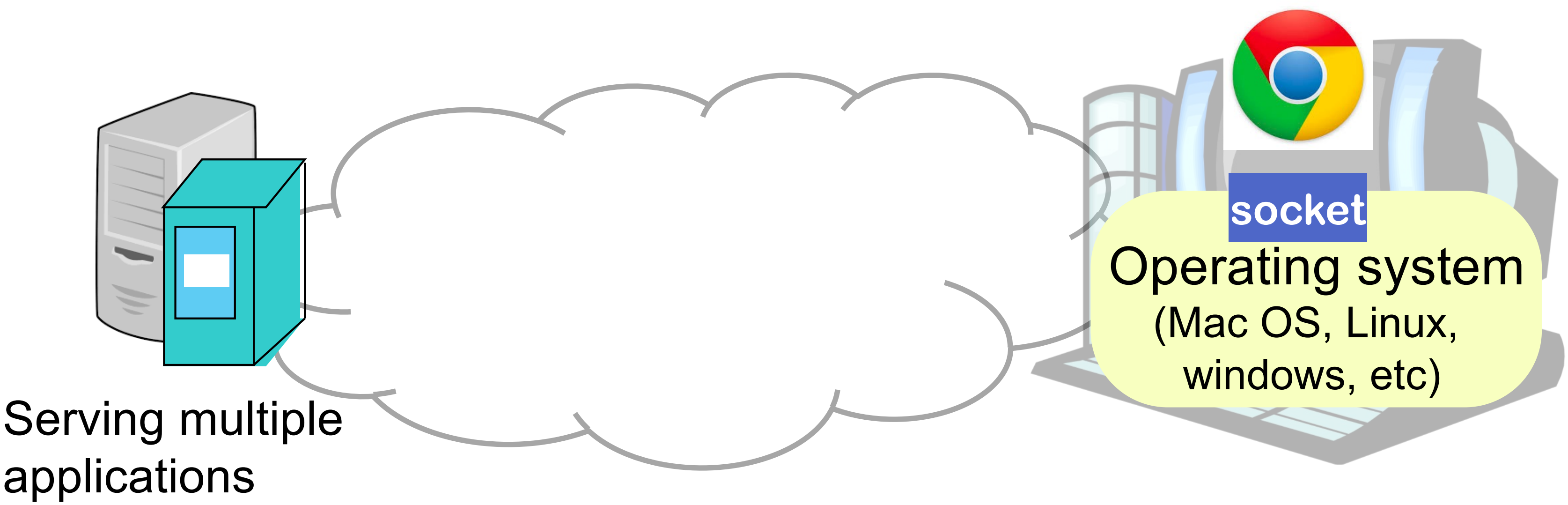
clients:

- Initiate communication with server

Q: How does a client process ***identify*** the server process with which it wants to communicate?

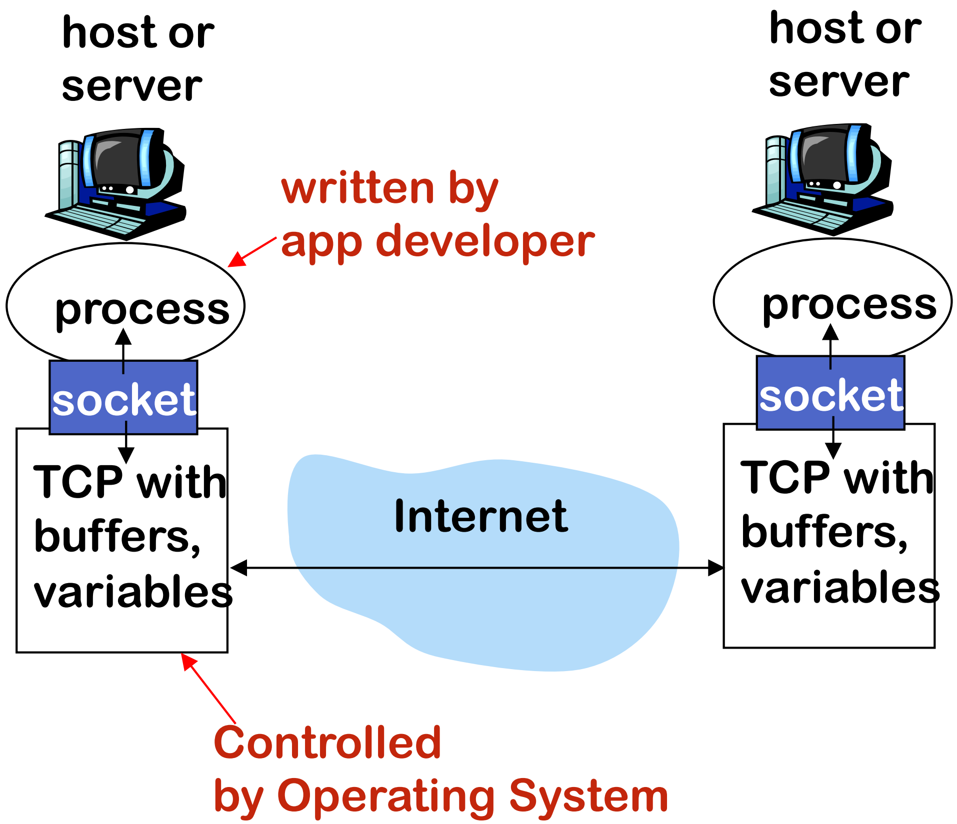
A: Using port numbers via the **socket API** (**A**pplication **P**rogram **I**nterface)




socket
Operating system
(Mac OS, Linux, windows, etc)
Serving multiple applications

Socket

- Process: program running on a host
- Between different hosts: Processes communicate through an application-layer protocol
- A process sends/ receives messages to/from its socket
- A socket analogous to a door:
  - sending process shoves message out of the door
  - transport protocol brings message up to the socket at receiving process



What is "socket"

- A set of system function calls

socket ( ): Create a socket

bind( ): bind a socket to a local IP address and port #

connect( ): initiating connection to another socket

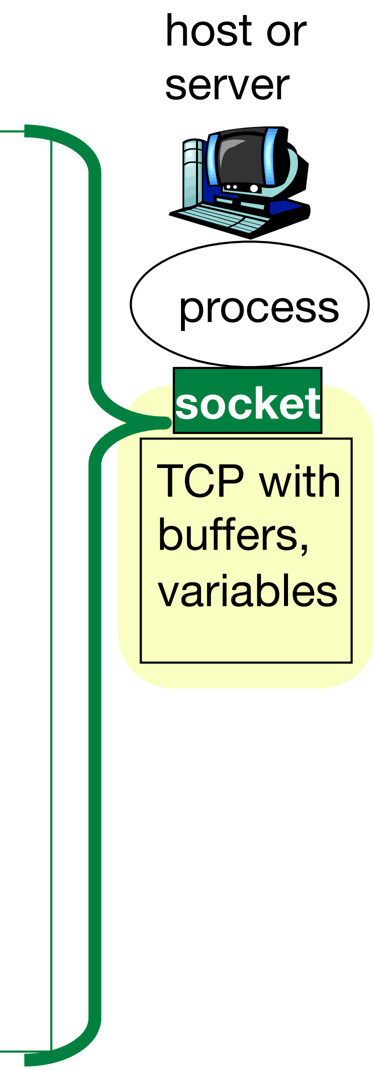
listen( ): passively waiting for connections

accept( ): accept a new connection

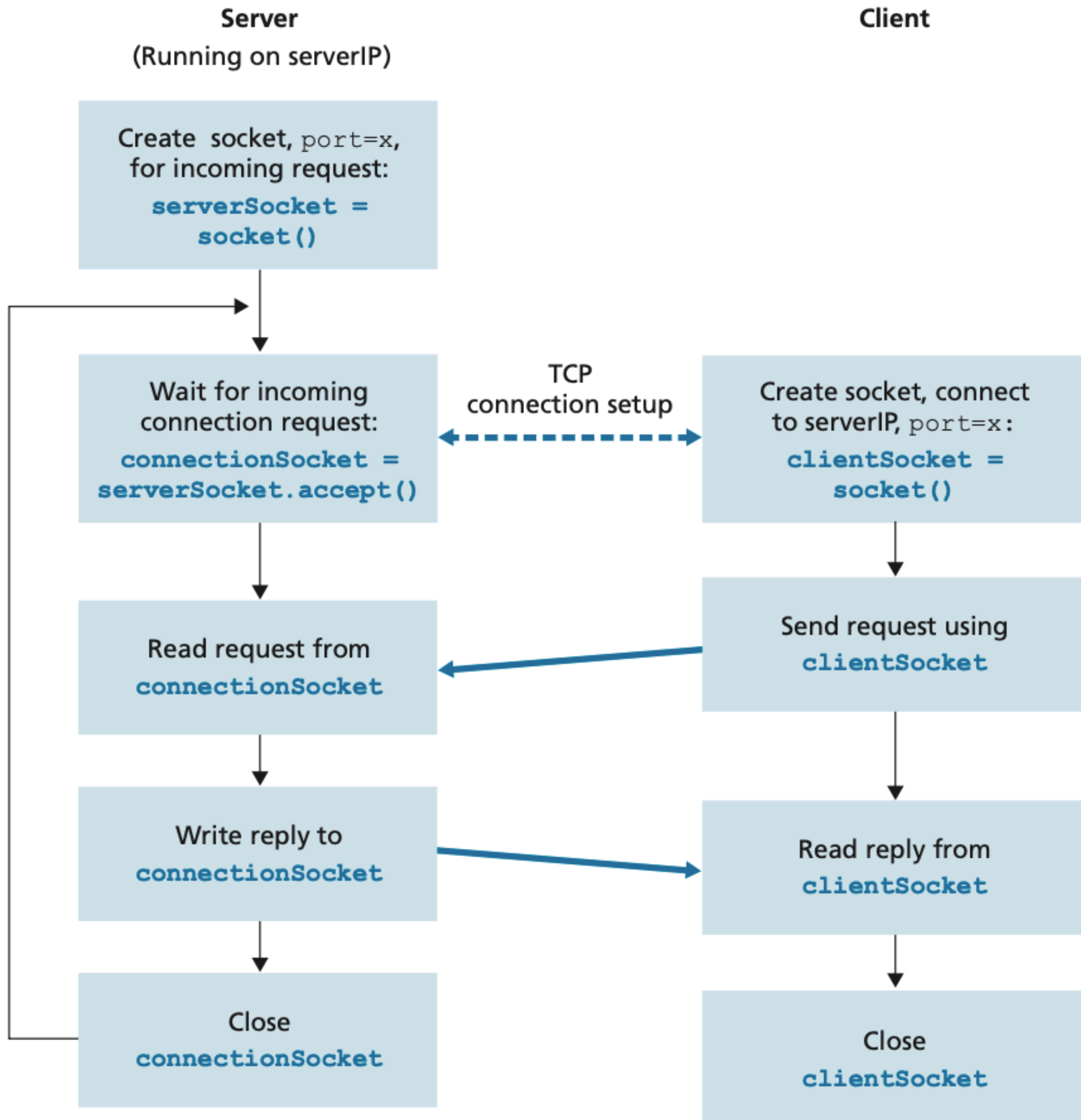
Write( ): write data to a socket

Read( ): read data from a socket

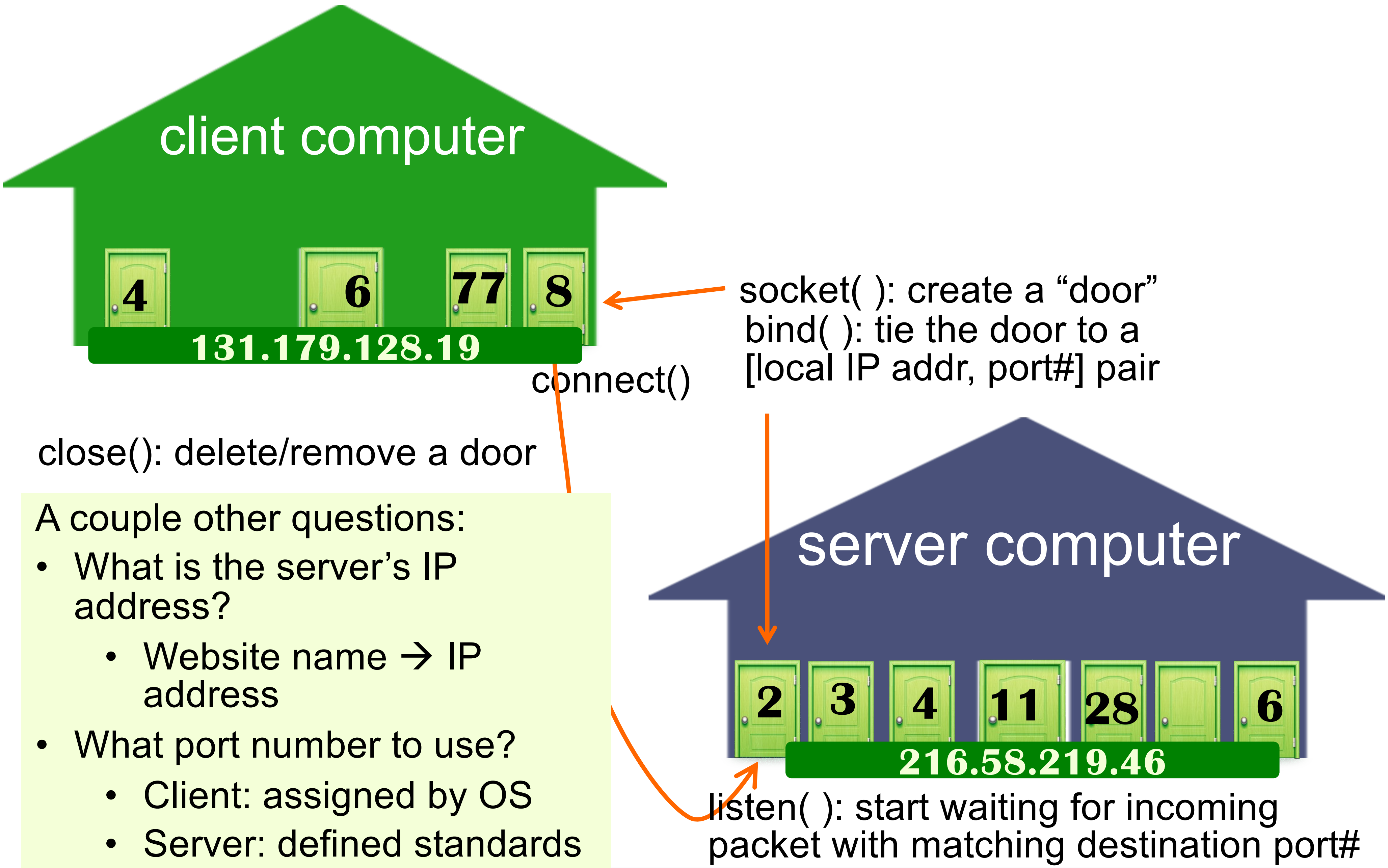
Close( )



What is "socket"



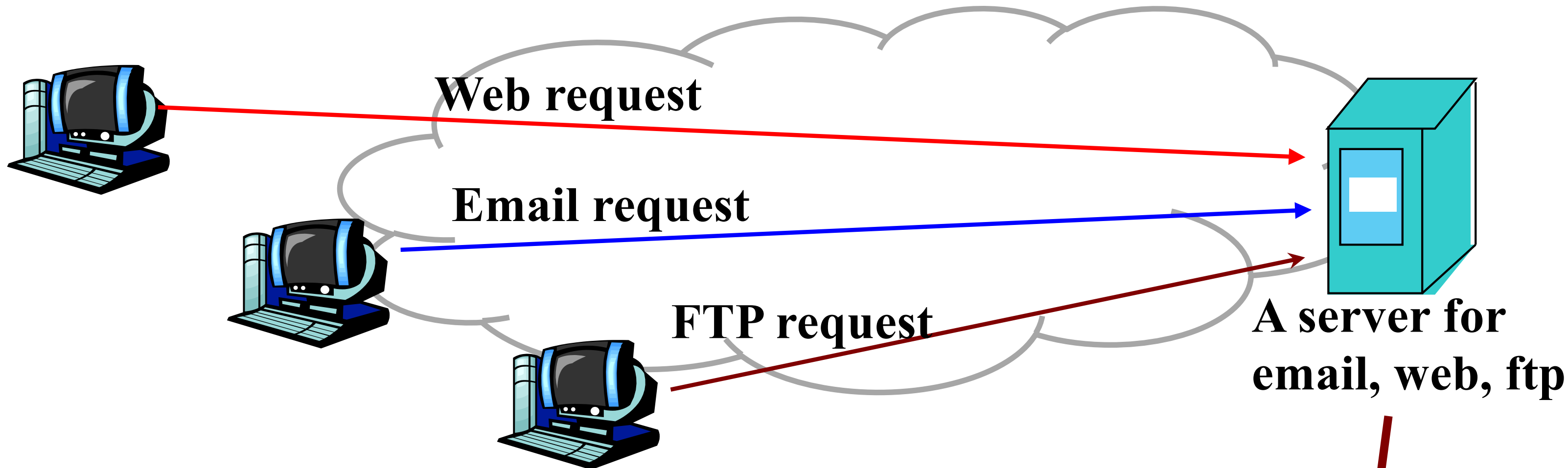
Socket: analogous to a door



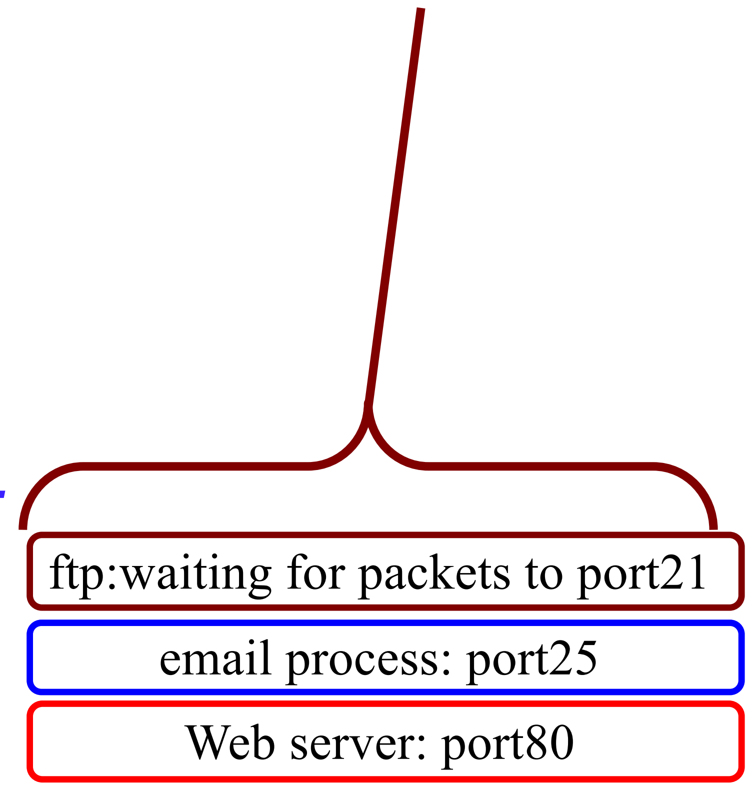
A couple other questions:

- What is the server's IP address?
  - Website name → IP address
- What port number to use?
  - Client: assigned by OS
  - Server: defined standards

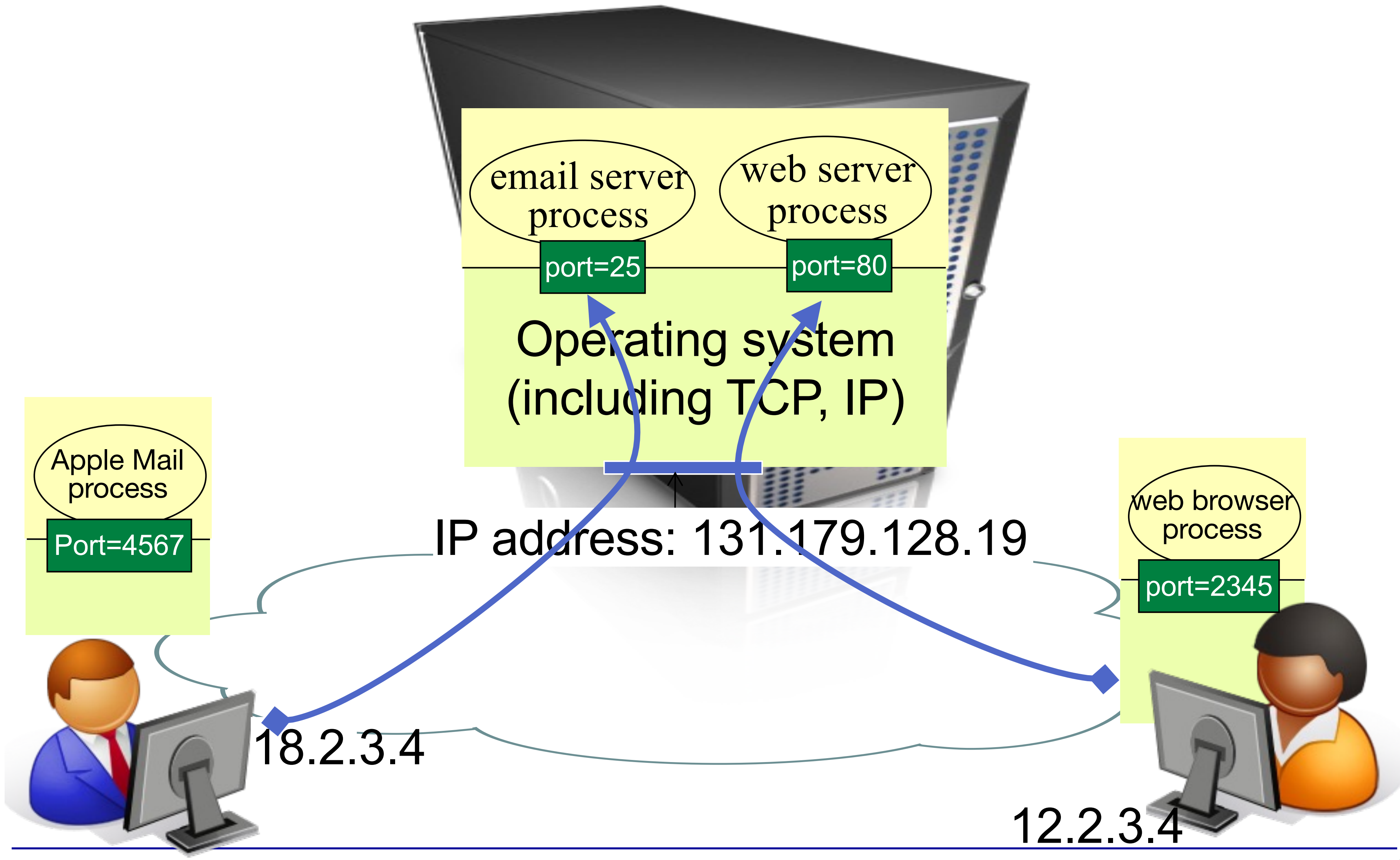
A quick comment about "port"



- Web, Email, FTP all connect client to server by TCP
- The server tells which client wants which service by *well-defined port number*:
  - Web: port 80, ftp 21, mail 25



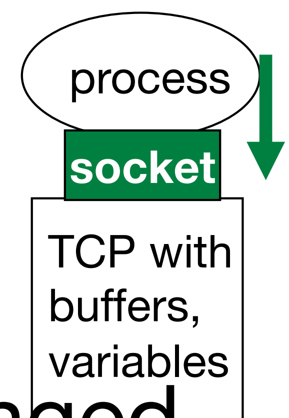
IP address, TCP connection, port number, processes, and sockets



Applications

So far we've talked

- Application process (executing application program)
- Application protocol (used by application processes to exchange data)
- Exactly how data is exchanged
  - Socket
  - Transport protocol
- Lets look at exactly what data is exchanged



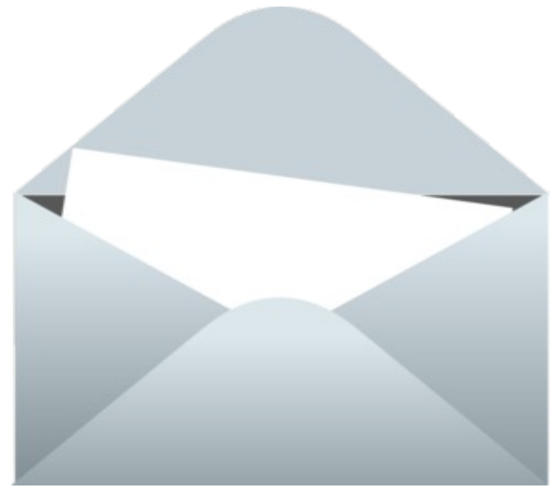
Web and HTTP

- Web page: normally consists of
  - base HTML-file, which includes
  - several referenced objects
- An object can be another HTML file, JPEG image, Java applet, audio file,…
- Each object is addressable by a URL (Universal Resource Locator)

`http://www.someschool.edu:port#/someDept/pic.gif`

Application protocol — host name — path name

What protocol “layer” really means



- application
- transport
- network
- link
- physical

Link layer protocol | Network protocol | Transport protocol | Application protocol | Application data

Acknowledgment

- Slides adapted from S24 CS118 instructed by Prof. Lixia Zhang