Packets and Pockets

CS 168 – Fall 2024 – Discussion 2

(small) Logistics

- Project 1a due on the 10th
- Project 1b due on the 20th

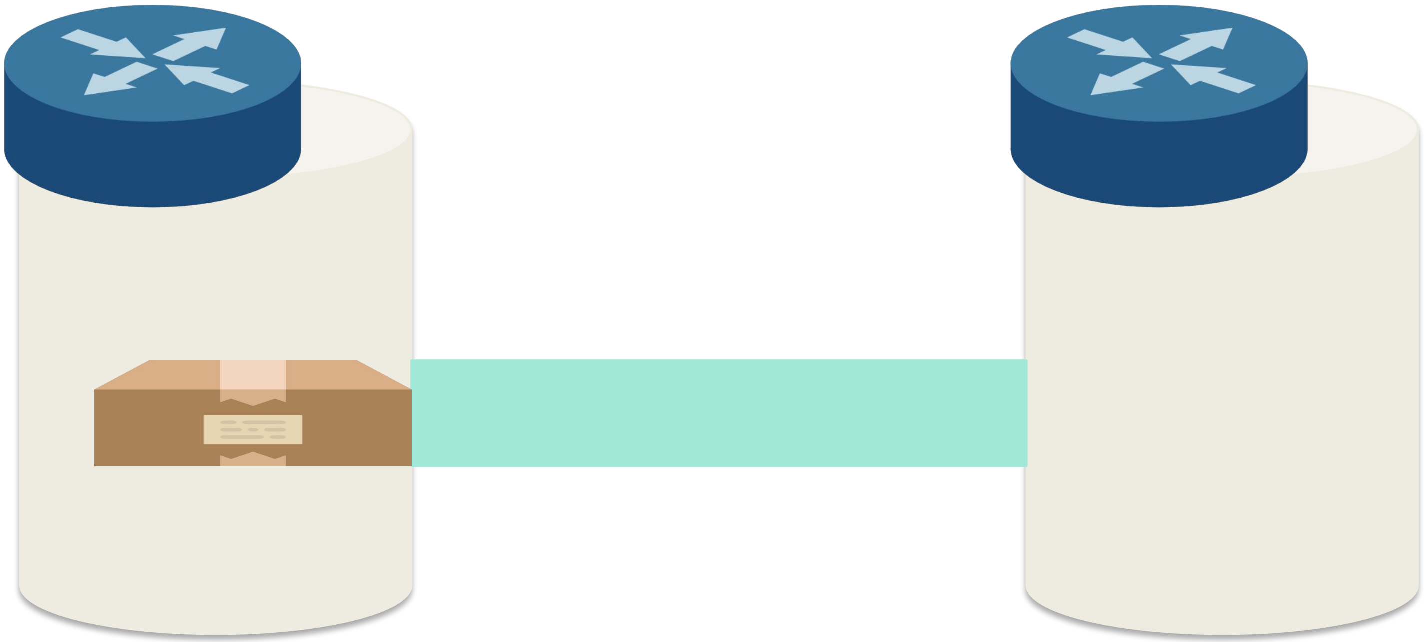
- Midterm on October 15th (faaaar away)

Delays

- How long does it take for your packet to travel through the network?
- It depends on…
  - how much data you're sending and the link speed
    → transmission delay
  - your distance from the destination
    → propagation delay
  - the traffic pattern
    → queuing delay

Transmission Delay

- How long it takes for the all bits in the packet to get on the wire
  - The time between when the first and last bits enter the link
- Limited by the bandwidth
  - Bandwidth: Number of bits you can send through a wire per unit of time



Transmission Delay

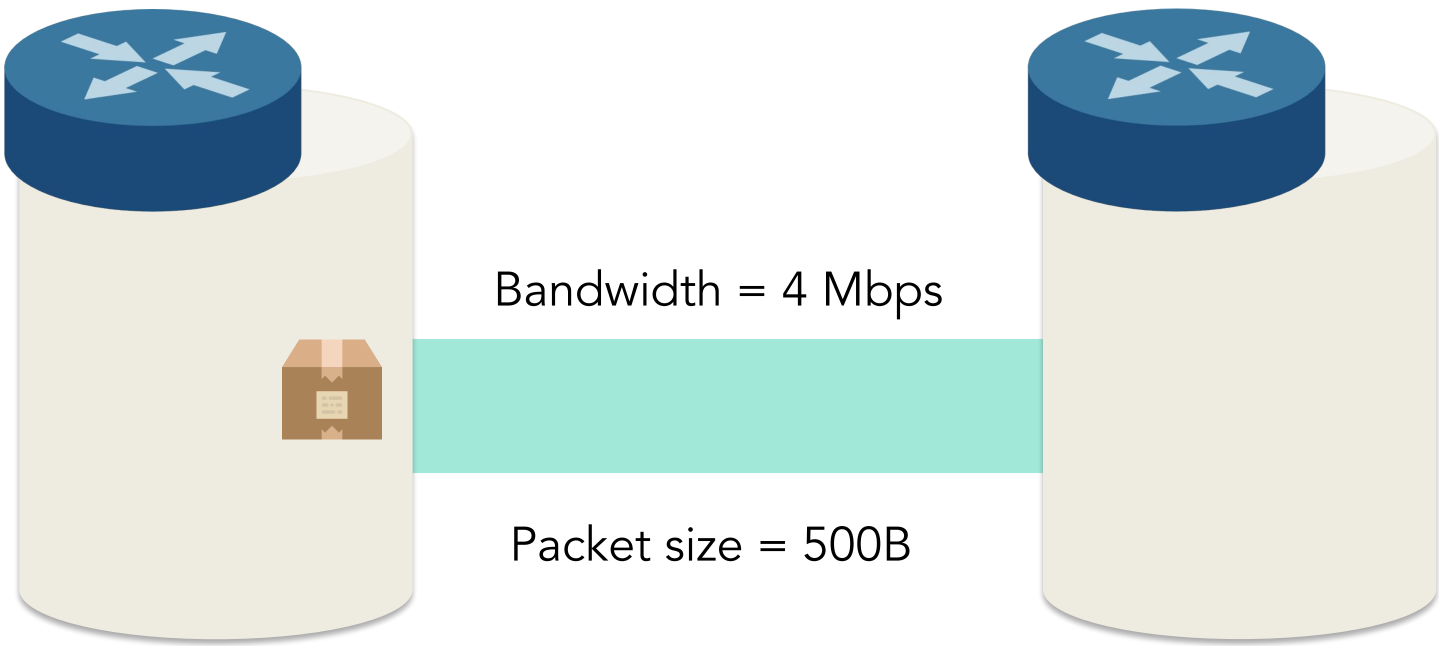
$$\text{Transmission Delay} = \frac{\text{packet size (bytes)}}{\text{bandwidth}}$$

Usually **bits**/second

Transmission Delay: Example

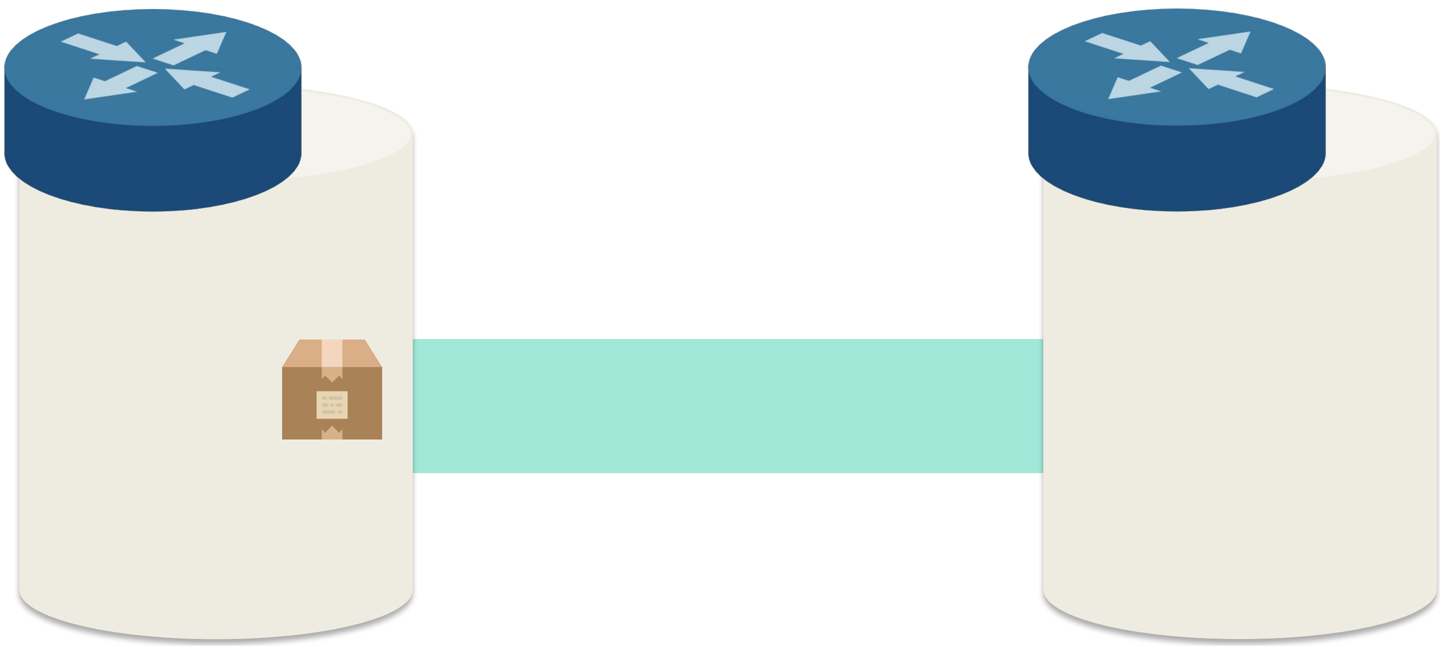
$$\text{Bandwidth} = \frac{4 \cdot 10^6 \text{ bits}}{1 \text{ second}} \cdot \frac{8 \text{ bits}}{1 \text{ byte}} = \frac{1}{2} \cdot 10^6 \frac{\text{Bytes}}{\text{second}}$$

$$\text{Transmission Delay} = \frac{\text{packet size (bytes)}}{\text{bandwidth}} = \frac{500 \text{ Bytes}}{\frac{1}{2} \cdot 10^6 \frac{\text{Bytes}}{\text{second}}} = 1 \text{ ms}$$



Propagation Delay (latency)

- End-to-end transmission time of one bit
- Depends on the length of the link
- Limited by the speed of light (propagation speed of link)
- Does NOT depend on the size of the packet

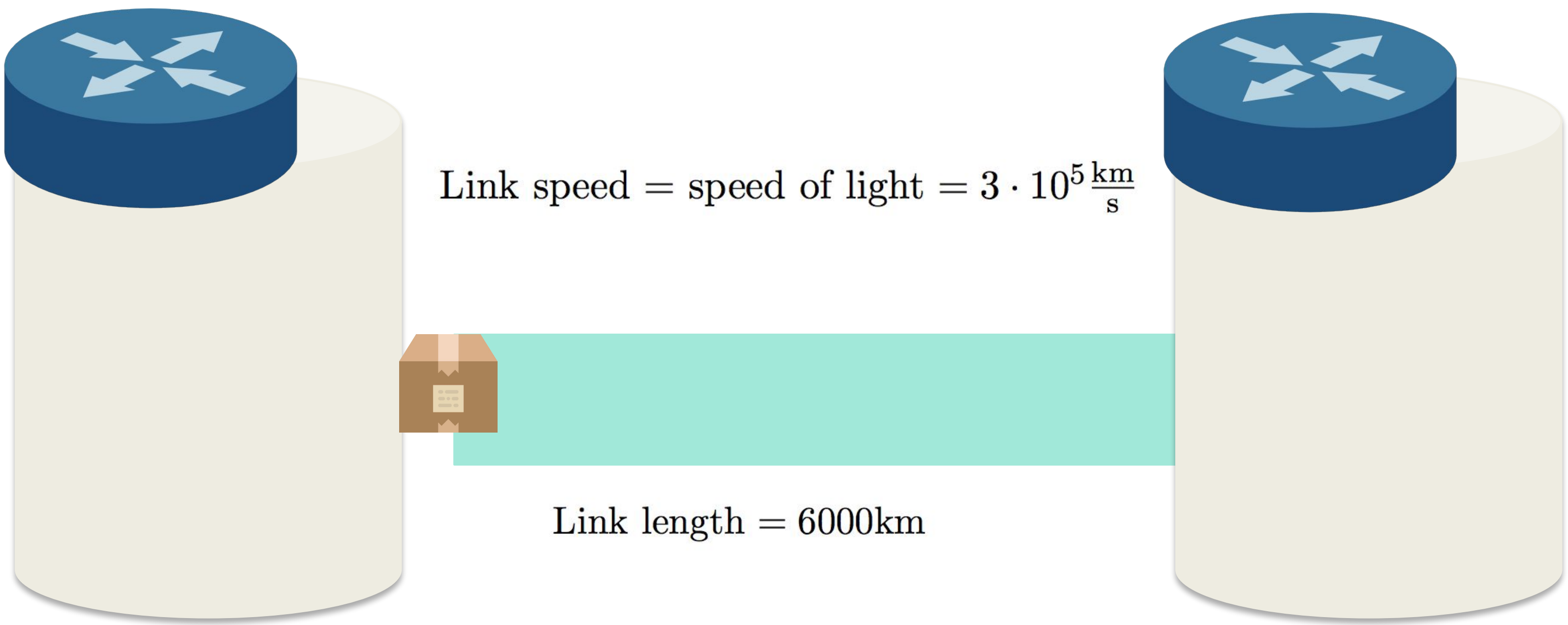


Propagation Delay: Formula

$$\text{Propagation Delay} = \frac{\text{length of link (meters)}}{\text{speed of light (meters/second)}}$$

Propagation Delay: Example

$$\text{Propagation Delay} = \frac{6000\text{km}}{3 \cdot 10^5 \frac{\text{km}}{\text{s}}} = 20\text{ms}$$



Bandwidth Delay Product (BDP)

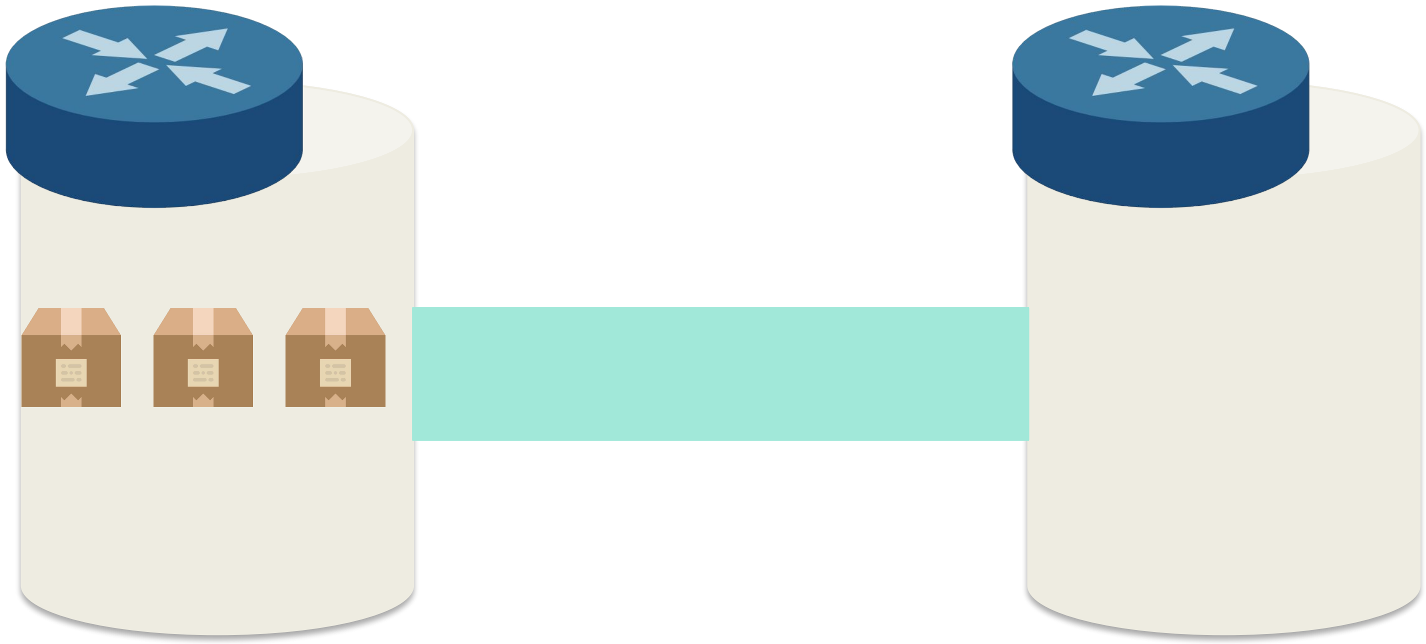
- Now that we know the propagation delay, we can tell how many bits are "in flight" (on the link) at any time

$$\text{Bandwidth Delay Product (BDP)} = \text{Bandwidth} \cdot \text{Propagation Delay}$$

Packets might have to wait before they can be transmitted…

Queuing Delay

- How long the packet waits to get transmitted on the wire
- Happens only when arrival rate is greater than transmission rate
  - more packets are arriving than are getting transmitted



Burstiness and Queues

- How does burstiness affect queuing delays?
  - Bursty flows tend to increase queuing delay

- What happens when the queue is full?
  - Packets are dropped

End-to-End Delay

Sum of all nodal delays on the path

End-to-End Delay = Propagation Delay + Transmission Delay + Queuing Delay

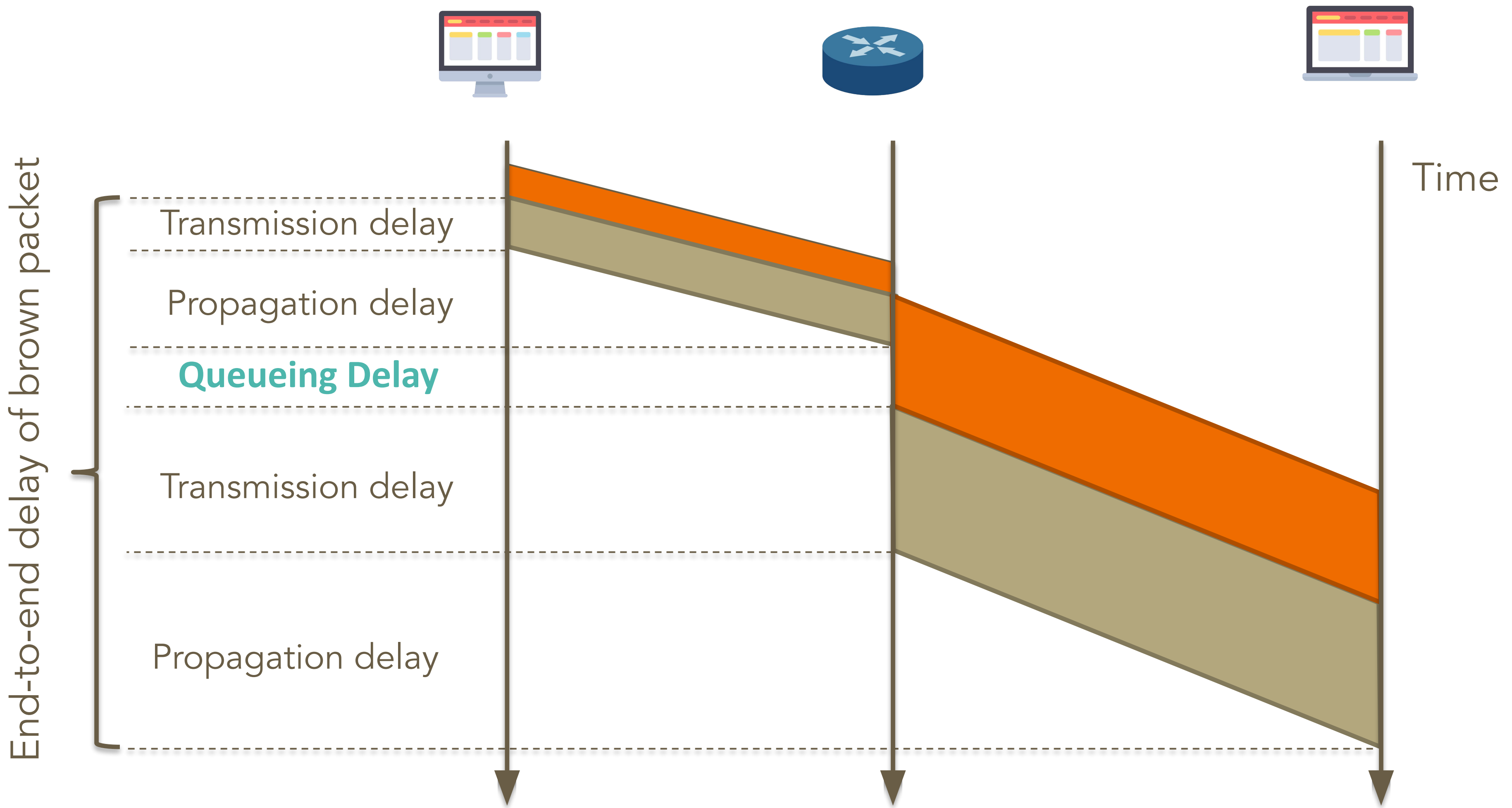
Round Trip Time (RTT)

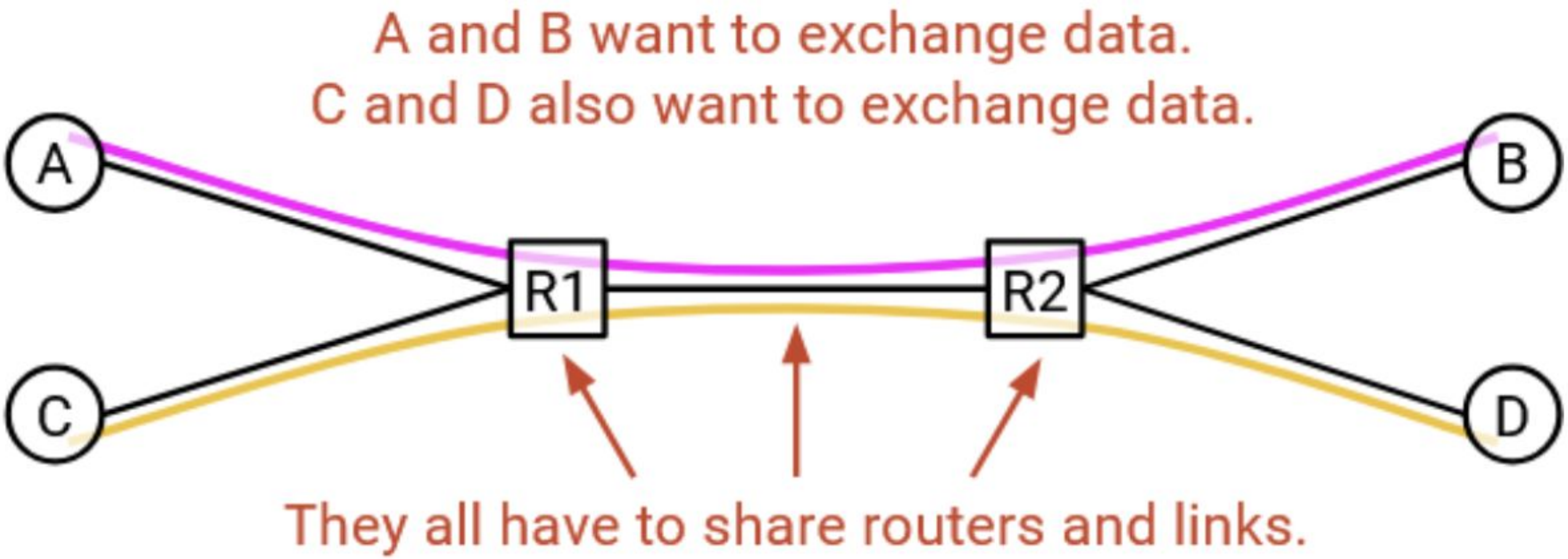
The time it for the packet to reach its destination and receive a response

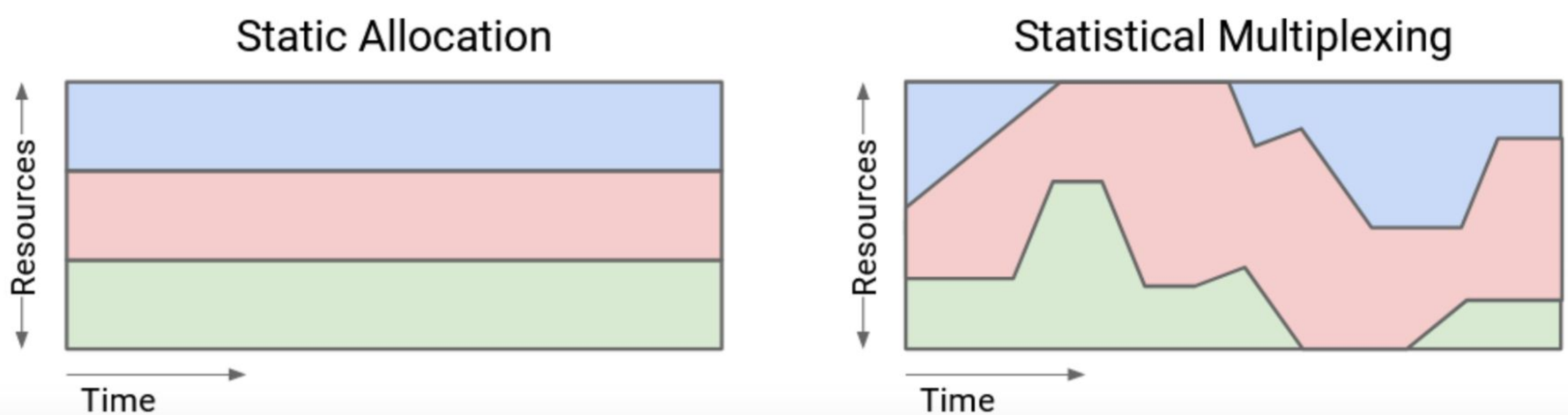
RTT = 2 * (End-to-End Delay)

Visualizing end-to-end delay...

- Two packets, back-to-back

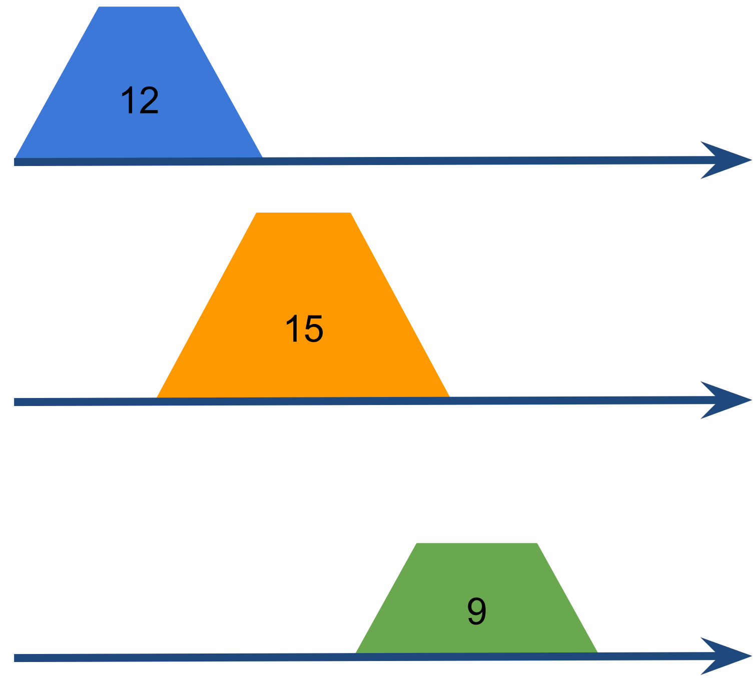


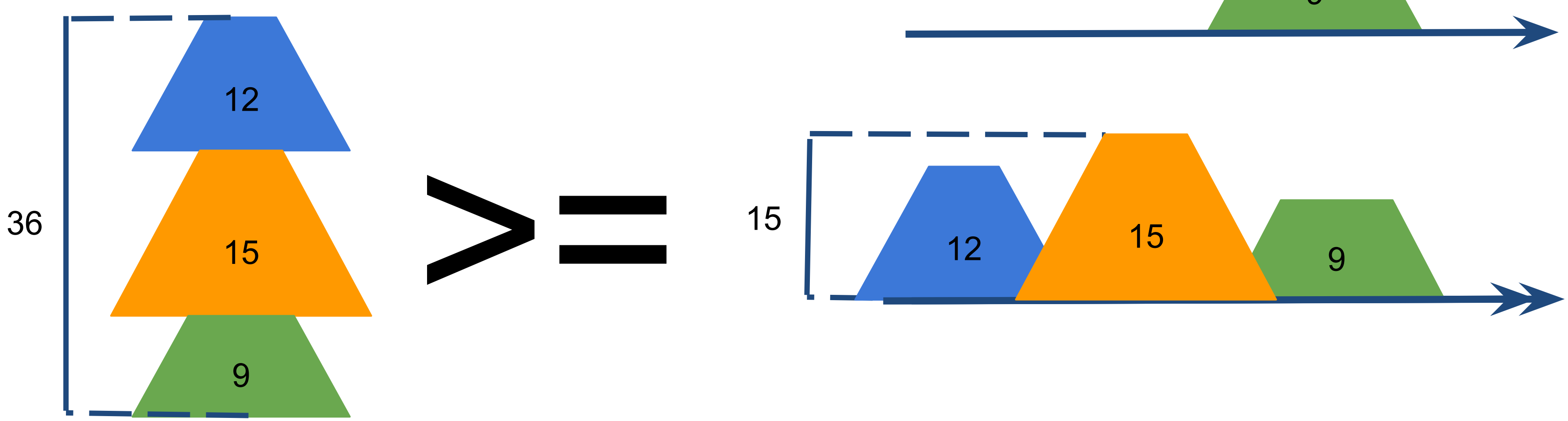

A and B want to exchange data.
C and D also want to exchange data.
A
B
R1
R2
C
D
They all have to share routers and links.


Static Allocation
Resources
Time
Statistical Multiplexing
Resources
Time

Statistical Multiplexing

- Sum of the peaks is always greater than the peak of the sums (peak of the aggregate). Usually much greater.

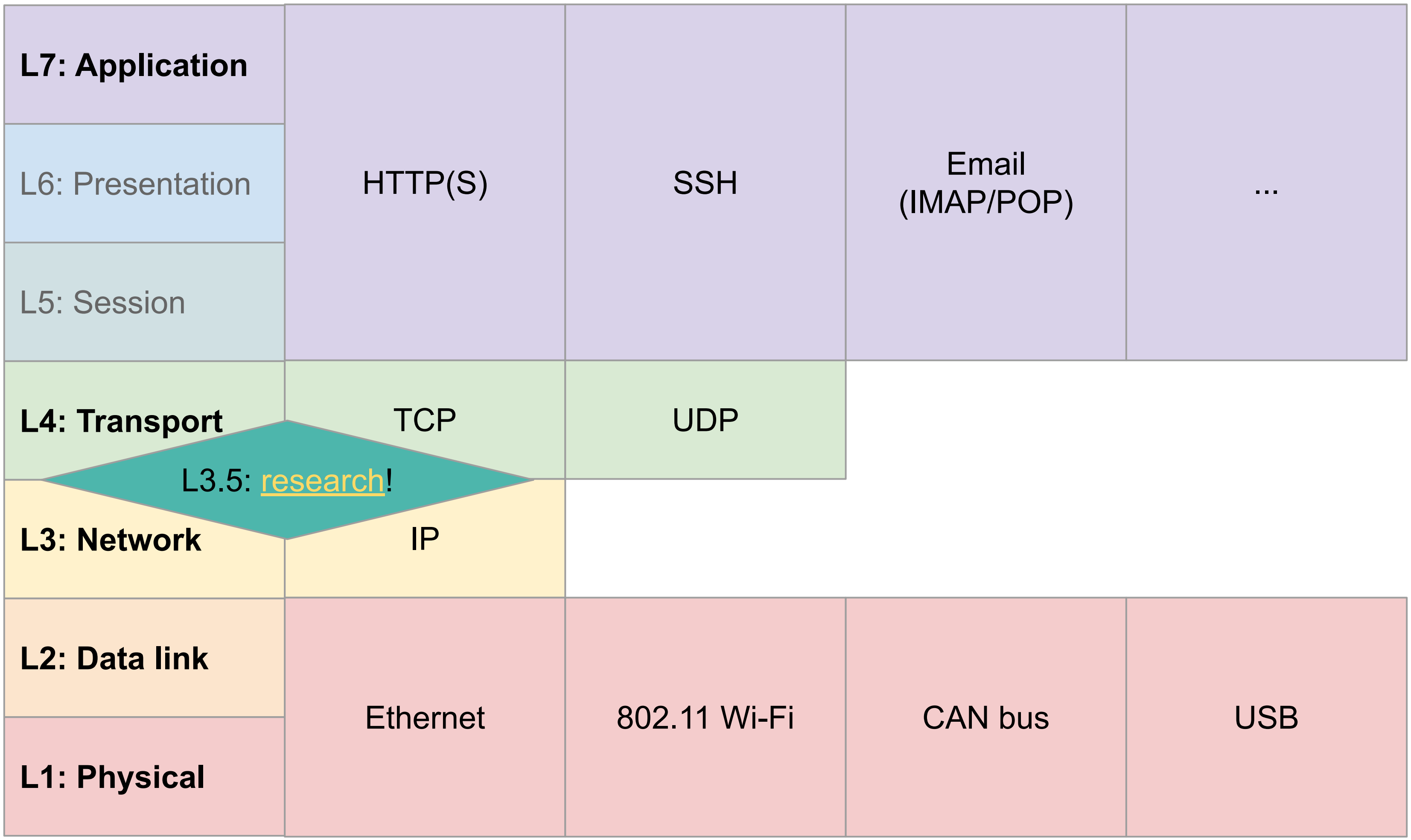




Layering in the Open Systems Interconnection Model

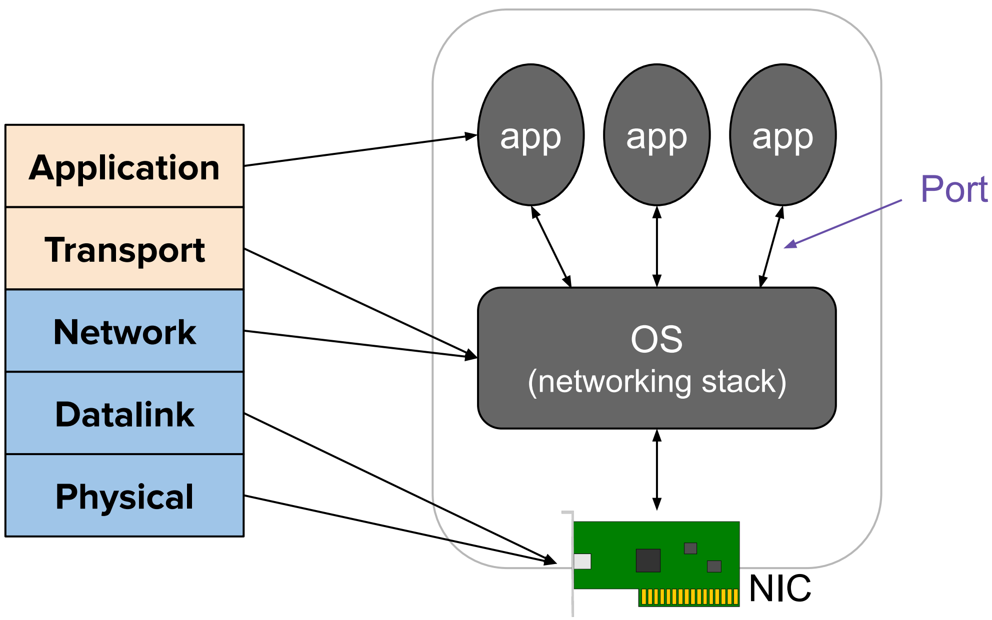
L7: Application	do the thing
L6: Presentation	(ignored here)
L5: Session	(ignored here)
L4: Transport	**beyond** delivery: (un)reliability, packet assembly, congestion control, ...
L3: Network	**global** delivery, best-effort
L2: Data link	**local** delivery, best-effort
L1: Physical	physical transfer of bits

Layering in practice



Sockets

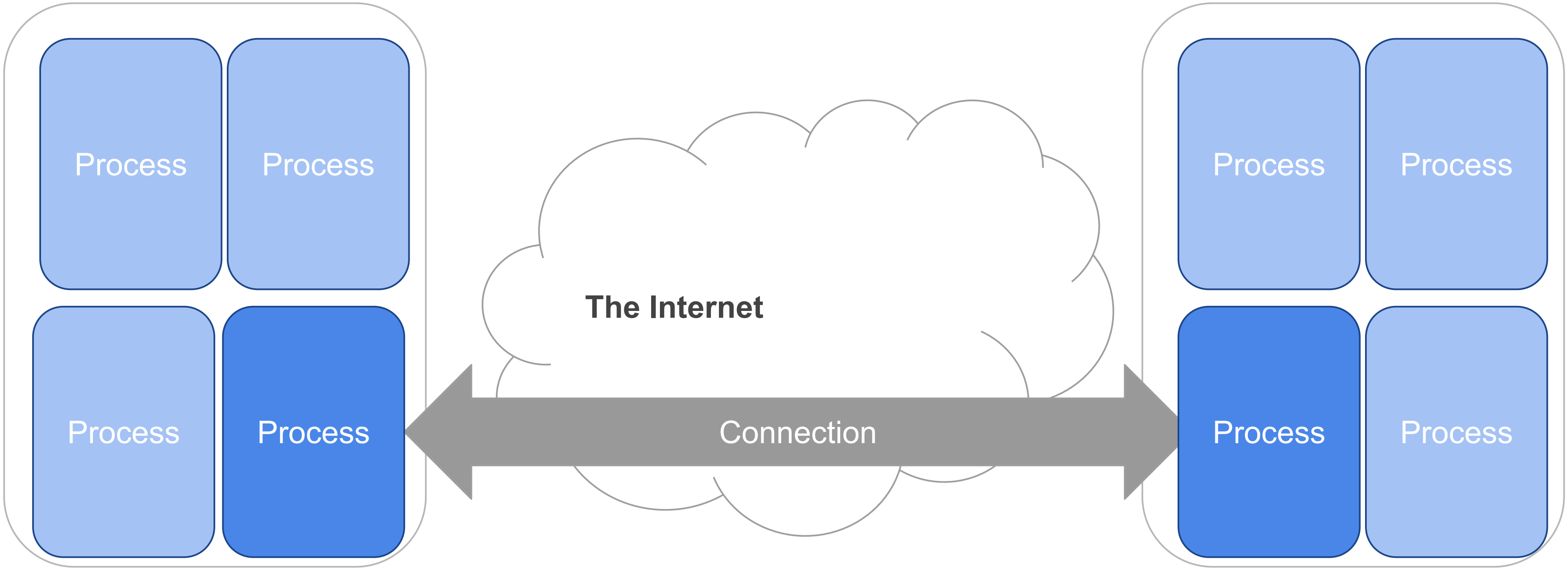
- Endpoint for sending or receiving data across a network
- OS abstraction for **connections**
- Allow L7 applications to operate on data streams (not packets)
    - Connect, listen, accept, send, receive
- Open a socket between:
    - Source IP address : *port*
    - Destination IP address : *port*



(Network Interface Card)

Connection (the basic abstraction)

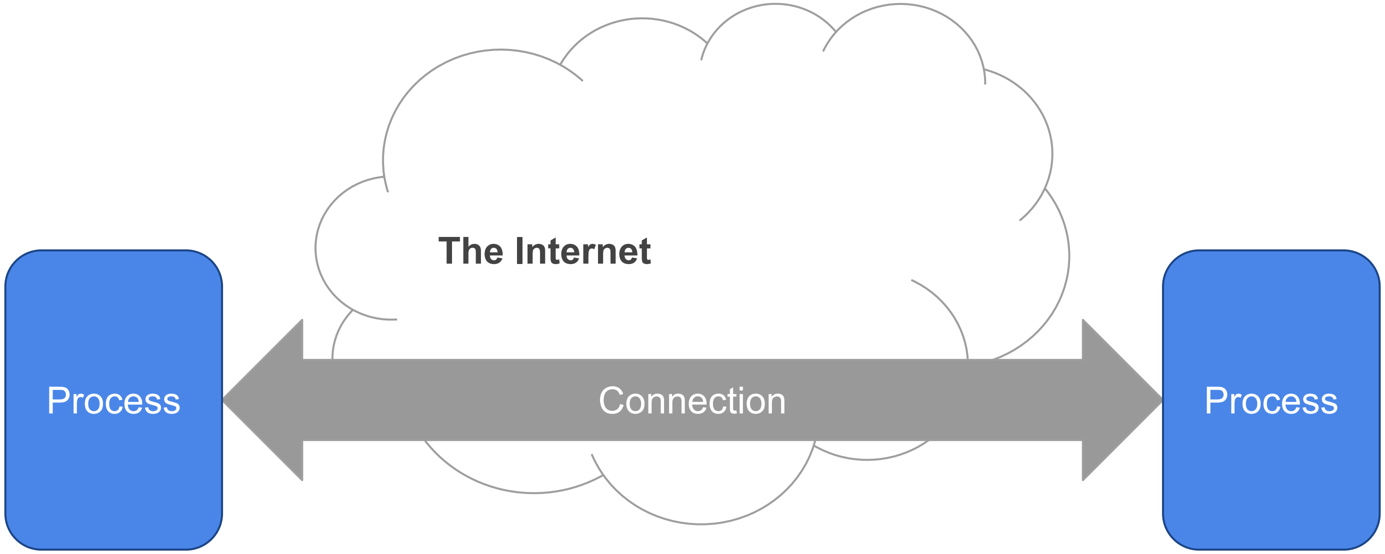
- Pipes data between two processes (on different hosts)
- Data flows both ways!



Connection (the basic abstraction)

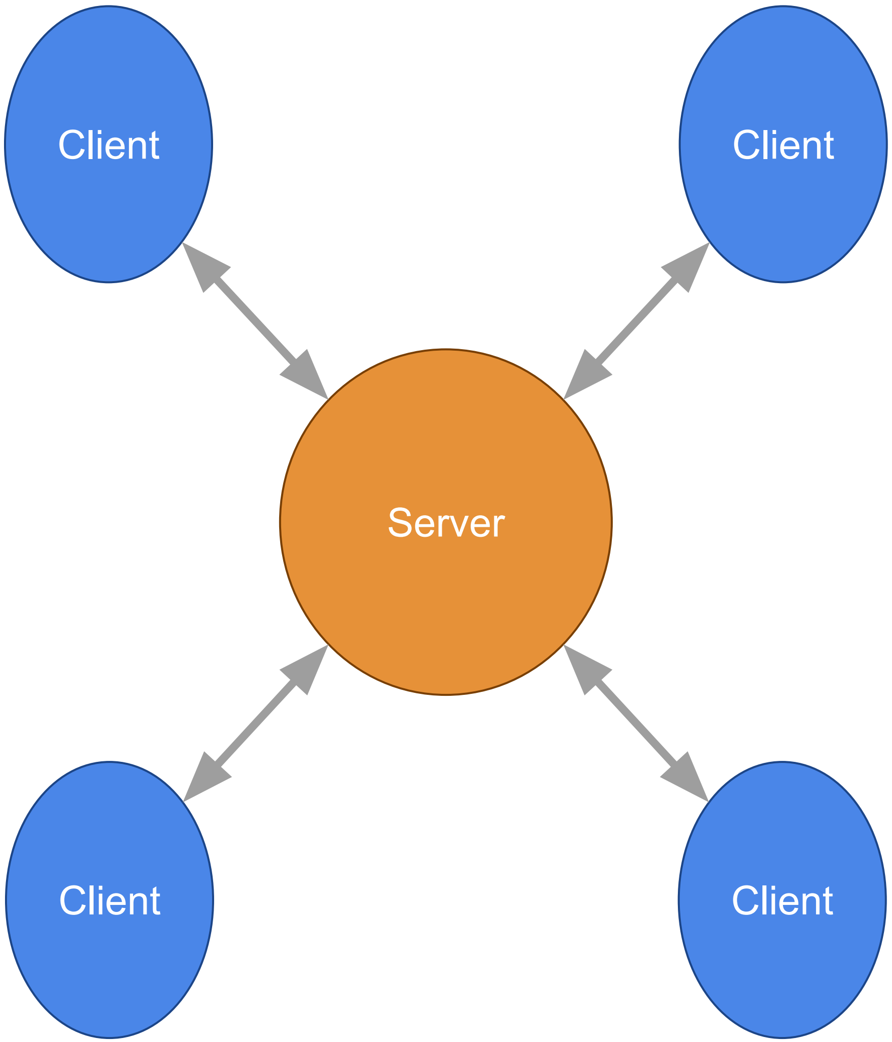
- Data is sent simply as a stream of bits
- Reconstruction of bits only at the endpoints
- The Internet knows nothing* about what it’s transmitting!

* (unless you’re implementing security)



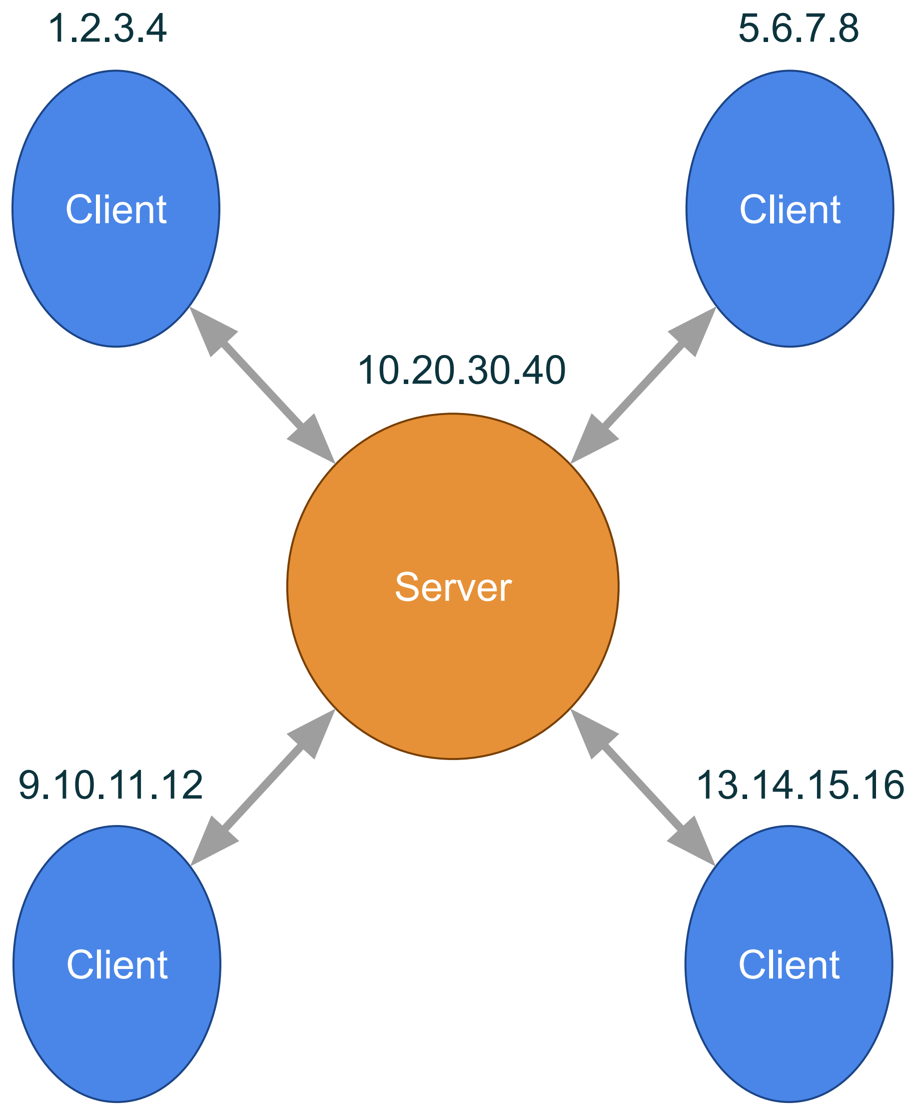
Connections

- Two types of sockets
  - Server and Client
- Servers *listen* for clients to connect to them
  - Wait until a connection is attempted
    - Accept and dispatch connection
  - Usually serving many clients at once
- Clients *initiate* new connections to servers
- Example
  - Server: berkeley.edu
  - Client: Your internet browser



Connections

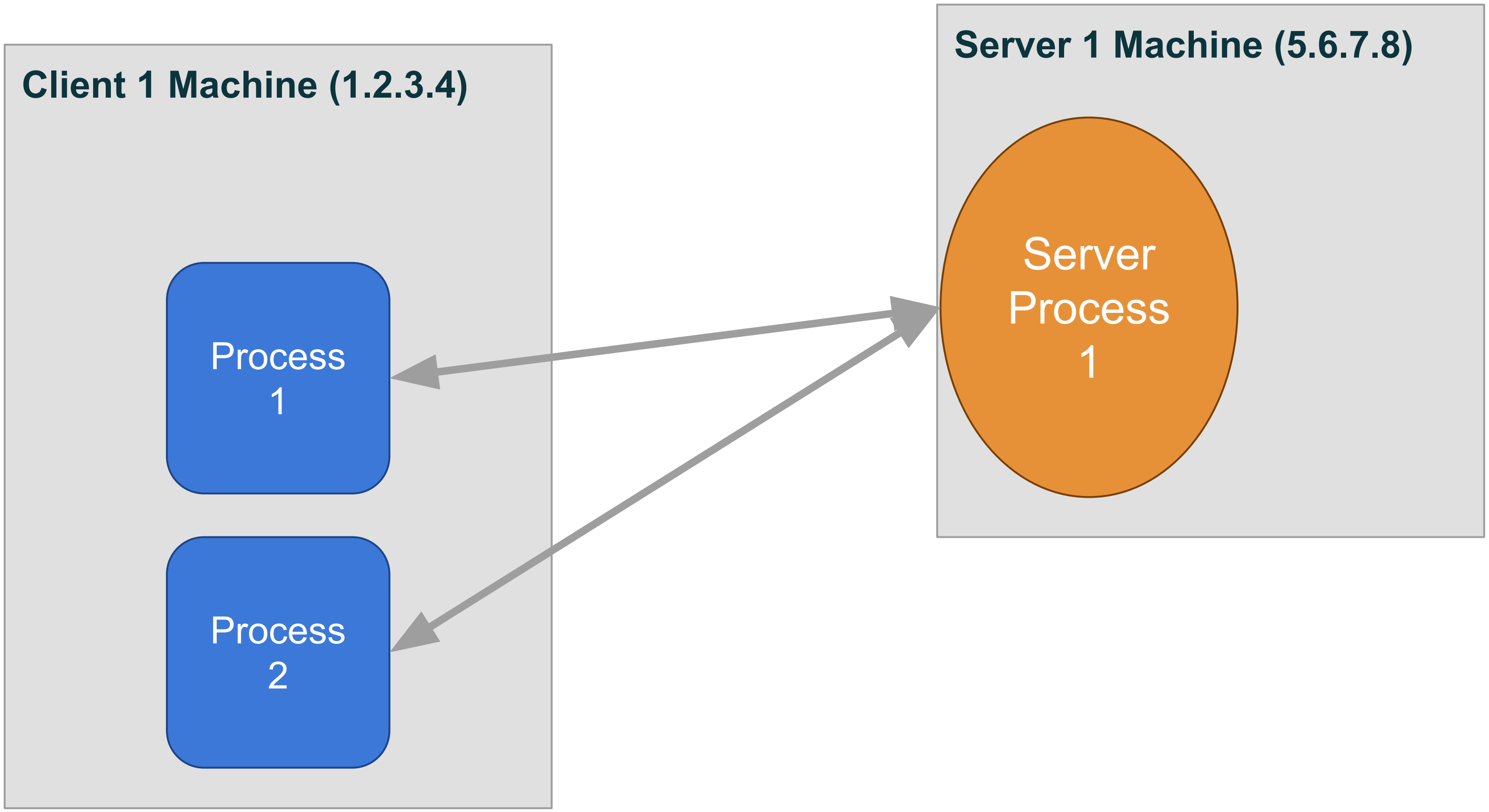
- Hosts have addresses
  - Unique identifier (just like a street address)
- Clients (different users) find servers with their addresses
  - Servers send data back with the client address
- Example addresses ➔



Are addresses enough to make this work?

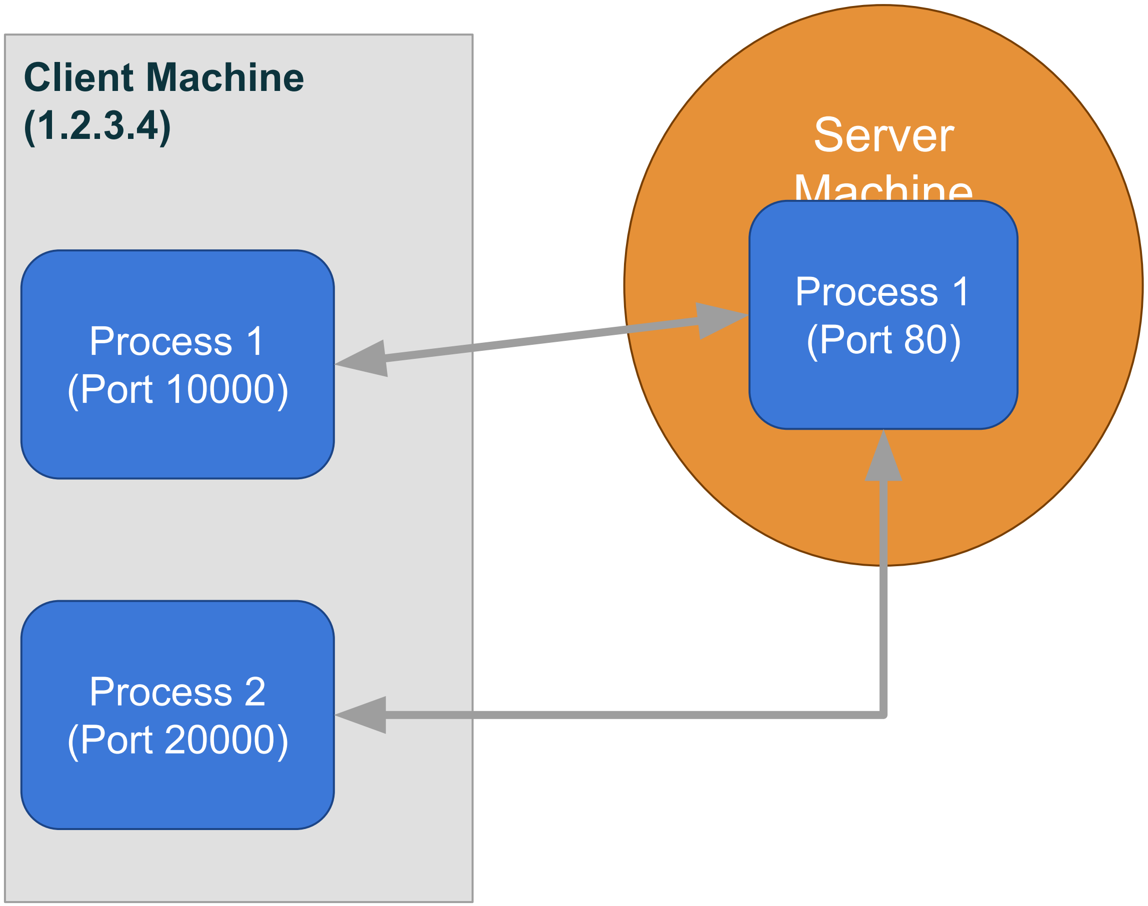
Address aren't enough

How does the client computer know which process (i.e. web browser) to deliver data to?



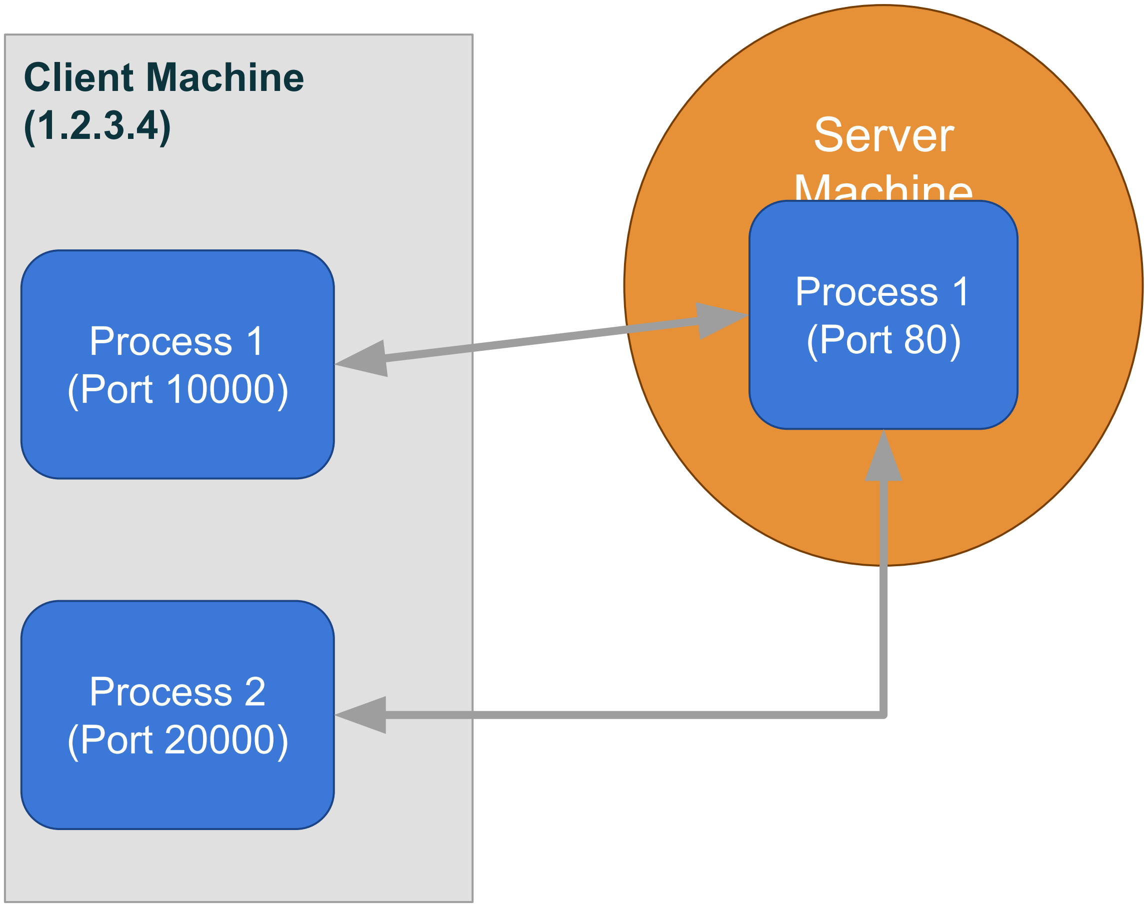
Ports

- Sockets are identified by unique IP:port pairs
- A port is a number that the OS associates with a socket when it is created
  - i.e. sending to address “1.2.3.4:10000” would send data to the socket owned by Process 1



Ports

- Packets carry port number
- Servers listen on a port
  - Which one depends on application
  - HTTP: 80
  - SSH: 22
- Client process connects to well known port
- Client also has a port
  - Randomly assigned by OS
  - Used by OS to send data to correct process



Questions?

Feedback Form:
https://tinyurl.com/cs168-disc-fa24