Midterm Review

CS 326E Elements of Networking

Mikyung Han

mhan@cs.utexas.edu

Time & Place

- Oct 10 Thursday 6-9 PM
- GDC 2.216 (lecture hall)
- Bring your *cheatsheet*, pen, and eraser
- No calculator needed
- No electronic device needed
- Scratch paper provided

Cheatsheet

- You can bring 1 double-sided 8.5x11 paper
- Type or handwritten

Exam Format

- Multiple choice
- Fill in the blank
- Calculations (no need to bring calculator)
- Explain why ABC
- Design your own protocol that does XYZ
- Write code (socket programming)
  - Both UDP and TCP
  - Syntax will be given

Practice Exam Questions

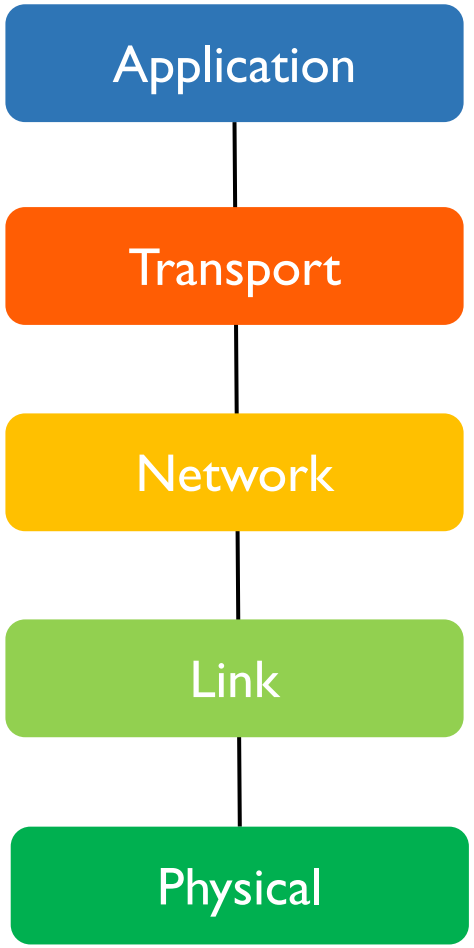
- EX0 and EX1 are your friends
- Pre-class activities and in-class activities
- DNS Dig questions
- Discussions during lecture
  - Why behind what
- Socket programming EX and Project

Topics

Intro and Network Performance

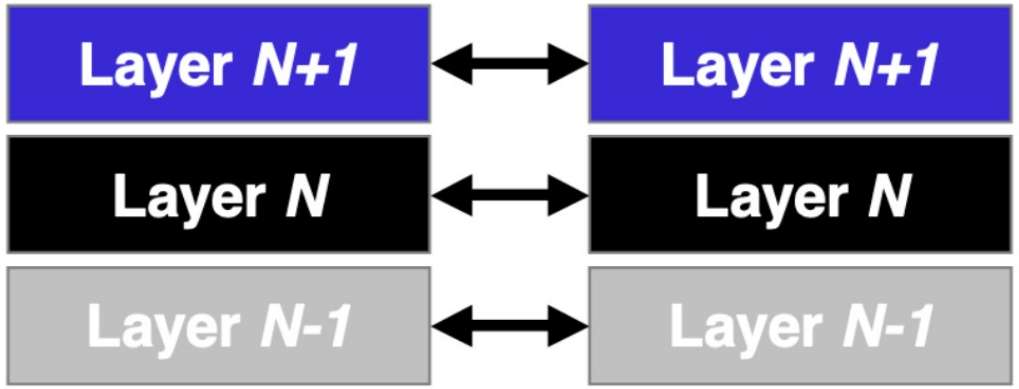
- **Layers and protocol**
  - What and why layer
  - What is protocol
- **4 types of delays**
  - What are they?
- **Packet switching vs circuit switching**
  - In-class exercise

Example Protocols	Layers	Responsible for
FTP, HTTP, SMTP	Application	application specific needs
TCP, UDP	Transport	process to process data transfer
IP	Network	host to host data transfer across different network
Ethernet, WiFi	Link	data transfer between physically adjacent nodes
802.3 PHY	Physical	bit-by-bit or symbol-by-symbol delivery



Protocols and layers are two building blocks in network communication

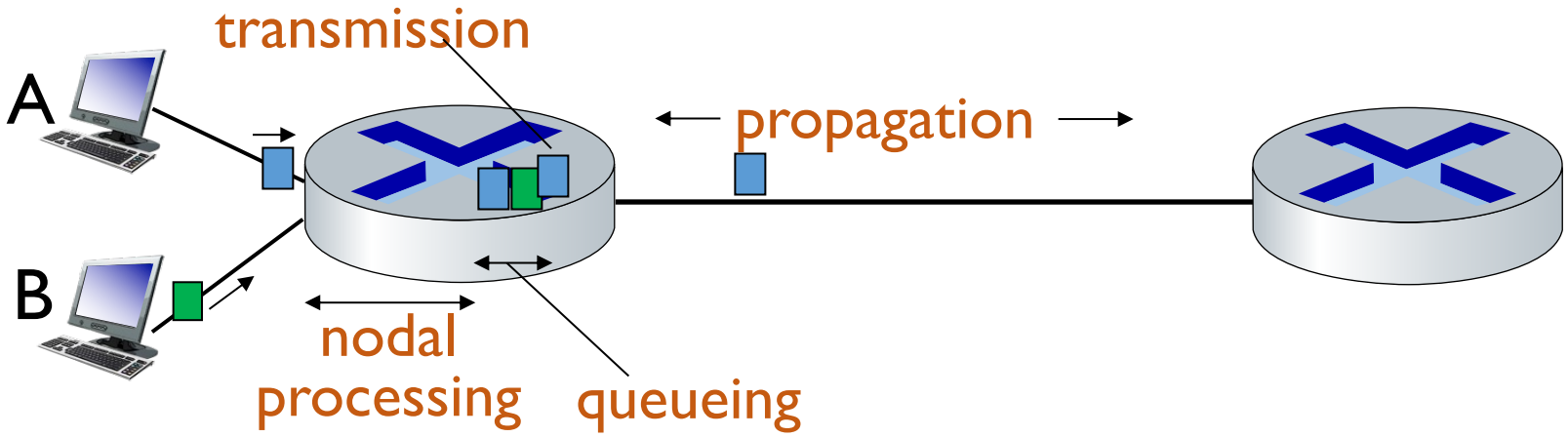
Protocols provides ways for peers to communicate horizontally



Protocol in layer N only interacts with peers in the same layer N

9

Packet delay: four sources



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

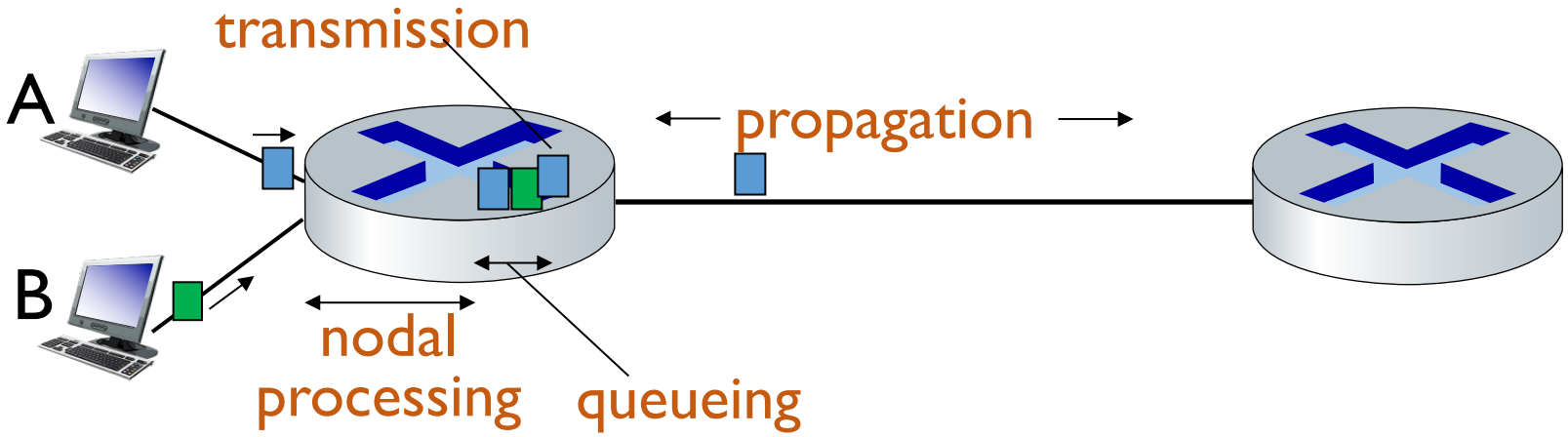
d_{proc}: nodal processing

- check bit errors
- determine output link
- typically < microsecs

d_{queue}: queueing delay

- time waiting at output link for transmission
- depends on congestion level of router

Packet delay: four sources



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{trans}: transmission delay:

- L: packet length (bits)
- R: link transmission rate (bps)
- $d_{trans} = L/R$

d_{prop}: propagation delay:

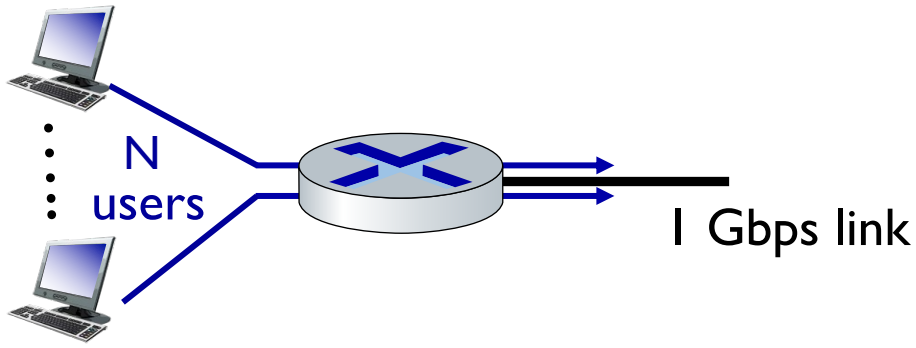
- d: length of physical link
- s: propagation speed (~2×10^8 m/sec)
- $d_{prop} = d/s$

d_{trans} and d_{prop} very different

Packet switching versus circuit switching

example:

- 1 Gb/s link
- each user:
  - 100 Mb/s when "active"
  - active 10% of time (happens randomly)



Q: What is the max number of users that can share this network?

- packet switching: ???

A: Need some assumption on link availability guarantee.

Say we guarantee 99.99% link availability for each user.
Failure rate < 0.0001 (When does failure happen?)

What is the number of users such that the probability of more than 10 users being active simultaneously < 0.0001?

Application Layer

Internet Reference Model

Example Protocols		Responsible for
FTP, HTTP, SMTP	Application	application specific needs
TCP, UDP	Transport	process to process data transfer
IP	Network	host to host data transfer across different network
Ethernet, WiFi	Link	data transfer between physically adjacent nodes
802.3 PHY	Physical	bit-by-bit or symbol-by-symbol delivery

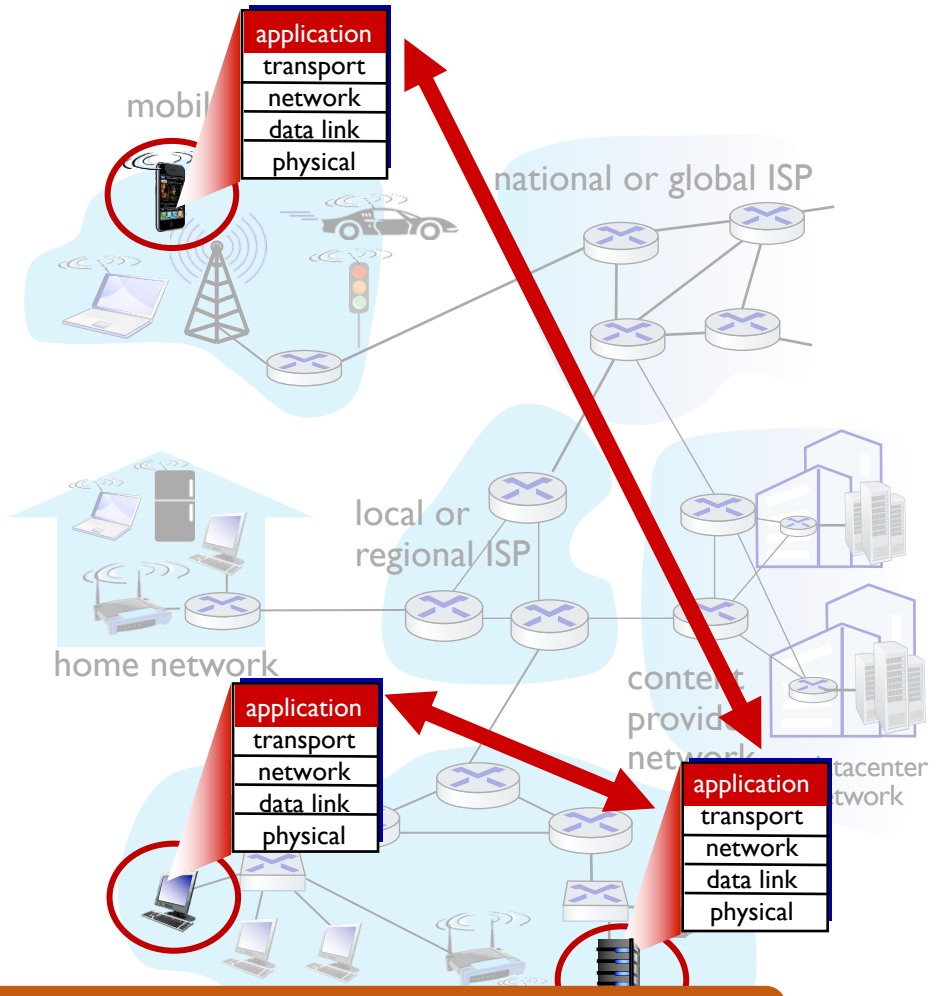
14

Design

- **End to End argument**
  - What is it and why they decided this way?
- **Stateless protocol vs stateful protocol**
  - What are they and when to use what
  - Examples
- **Connection-oriented protocol vs connectionless protocol**
  - What are they and when to use what
  - Examples

Applications only run on the endpoints!

- Network core devices do NOT run user applications
  - No code to write for these ☺
- When developing an app, we only need to consider the two ends
  - server/client or peers



This allowed rapid app development and propagation

DNS

- Distributed hierarchical design
- DNS dig
  - Know how to read results
- Recursive query vs non–recursive query
- Root DNS servers/TLD servers/Authoritative servers
- Caching
- What is the underlying protocol?
  - Why?

HTTP

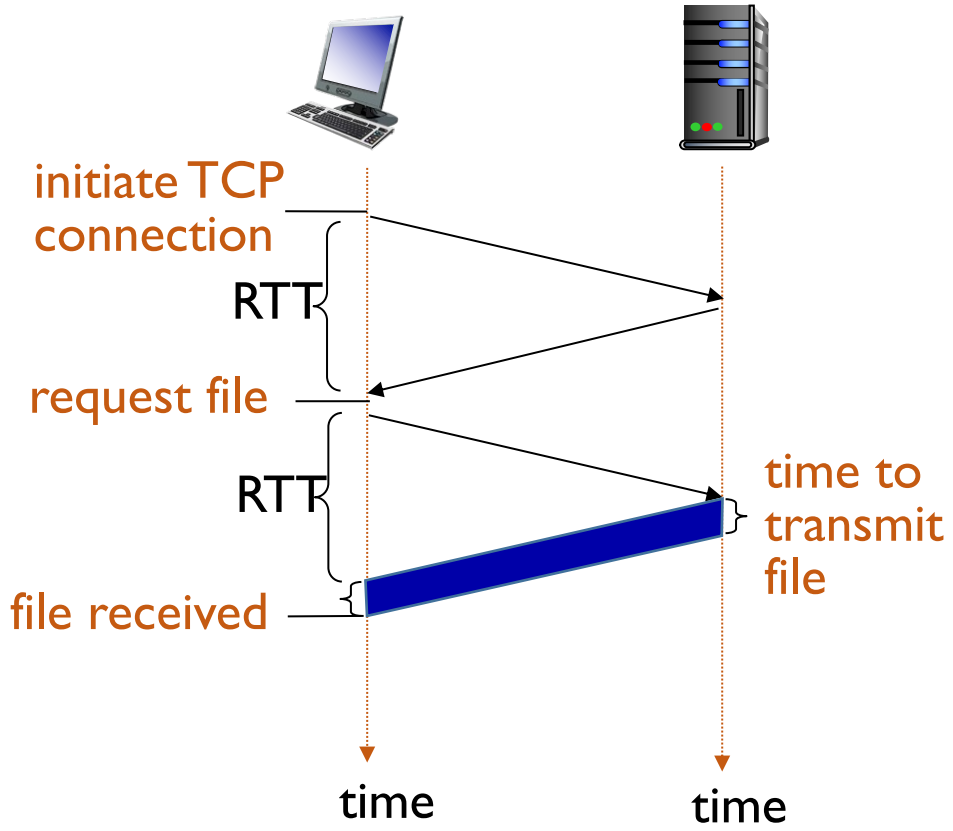
- Difference between HTTP/1.0, HTTP/1.1 and HTTP/2.0
- Persistent connection vs non-persistent
- Head of line blocking
- Framing
- Framing with priority
- What is the underlying protocol?
  - Why?

HTTP 1.0: Non-persistent HTTP takes 2 RTT + object transmission time per object!

RTT (definition): Round trip time between client and server

HTTP response time (per object):

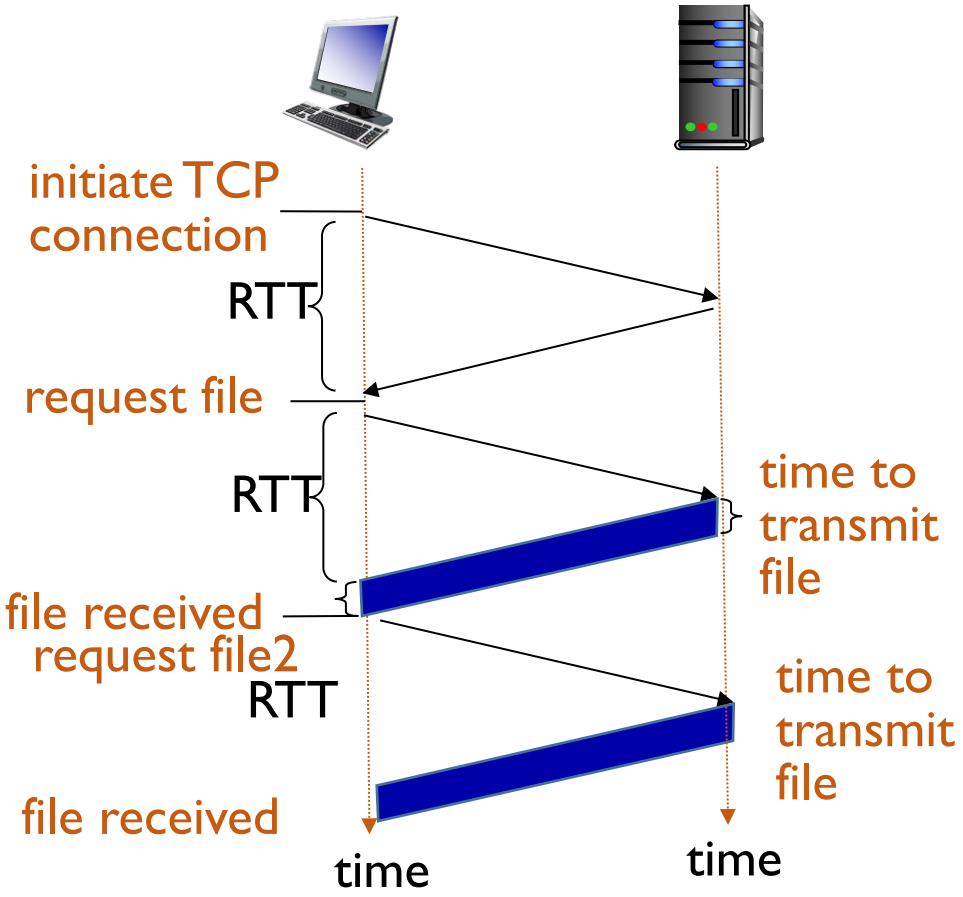
- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- object/file transmission time



This was HTTP 1.0

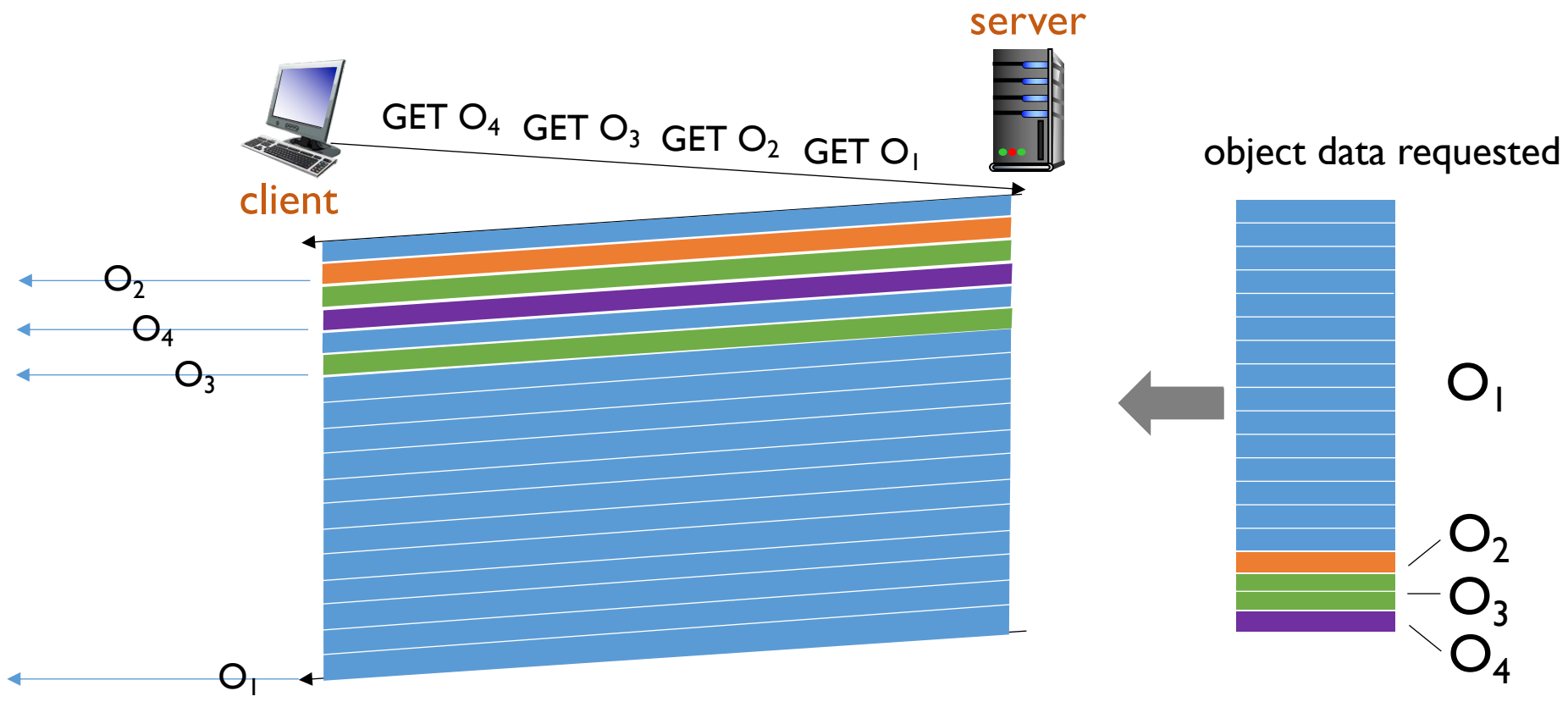
HTTP 1.1: Persistent HTTP reuses the same TCP connection over multiple objects

Initial object 2 RTT
+ subsequent objects 1 RTT
+ all objects TX time



HTTP/2 mitigates HOL blocking

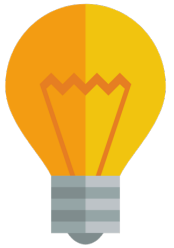
HTTP/2: objects divided into frames, frame transmission interleaved



O_2, O_3, O_4 delivered quickly, O_1 slightly delayed

Transport Layer

Internet Reference Model



Example Protocols		Responsible for
FTP, HTTP, SMTP	Application	application specific needs
TCP, UDP	Transport	process to process data transfer
IP	Network	host to host data transfer across different network
Ethernet, WiFi	Link	data transfer between physically adjacent nodes
802.3 PHY	Physical	bit-by-bit or symbol-by-symbol delivery

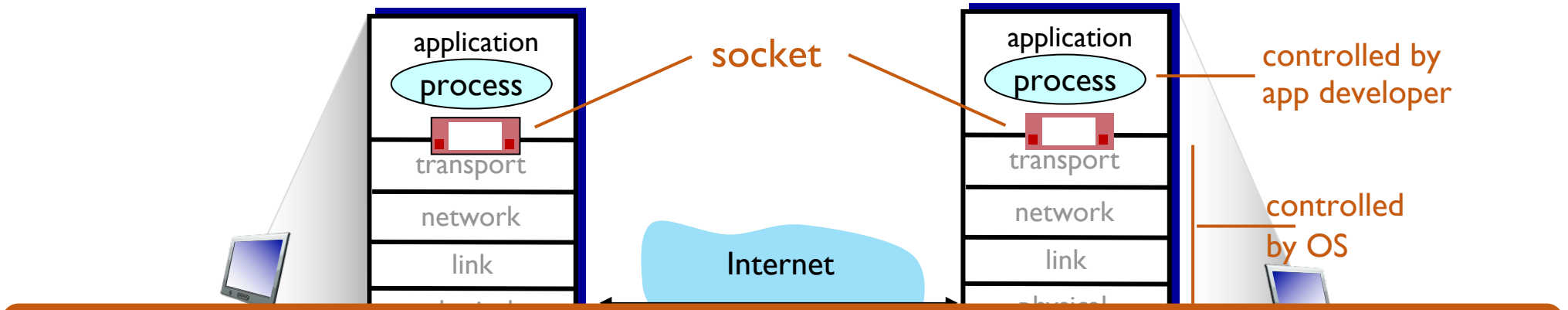
23

Transport Layer

- Socket and port
- TCP vs UDP
- Multiplexing and demultiplexing
- RDT
- Socket programming

What is a Socket?

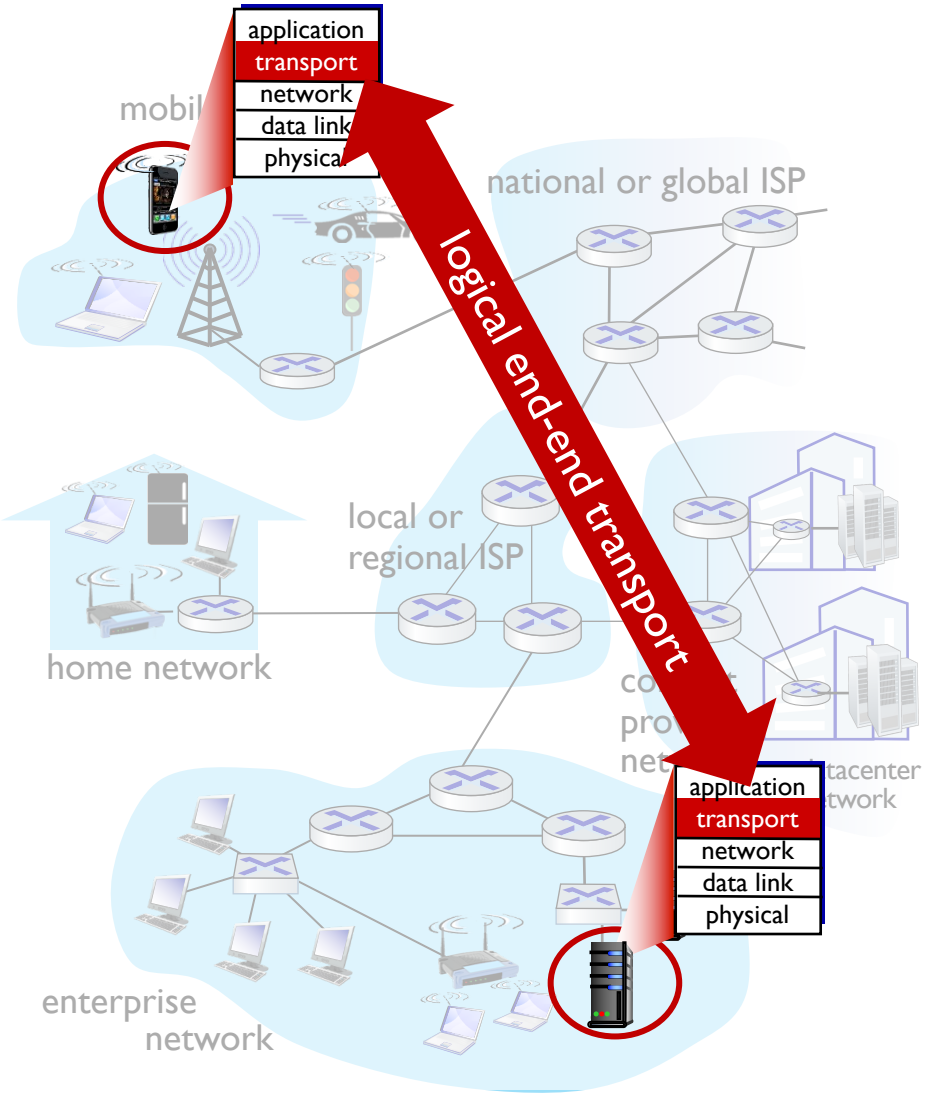
- process sends/receives messages to/from its socket
- socket analogous to a "door"
  - sending process shoves message out the door
  - sending process relies on transport layer to deliver message to socket at receiving process
  - two sockets involved: one on each end



Socket programming question

TCP vs UDP Features

- TCP: Transmission Control Protocol
  - reliable, in-order delivery
  - congestion control
  - flow control
  - Connection-setup
- UDP: User Datagram Protocol
  - unreliable, unordered delivery
  - no-frills extension of "best-effort" IP
- Both has NO guarantee on delay or bandwidth



Socket programming

- TCP socket program
- UDP socket program
- Thread
  - Worker thread vs main thread

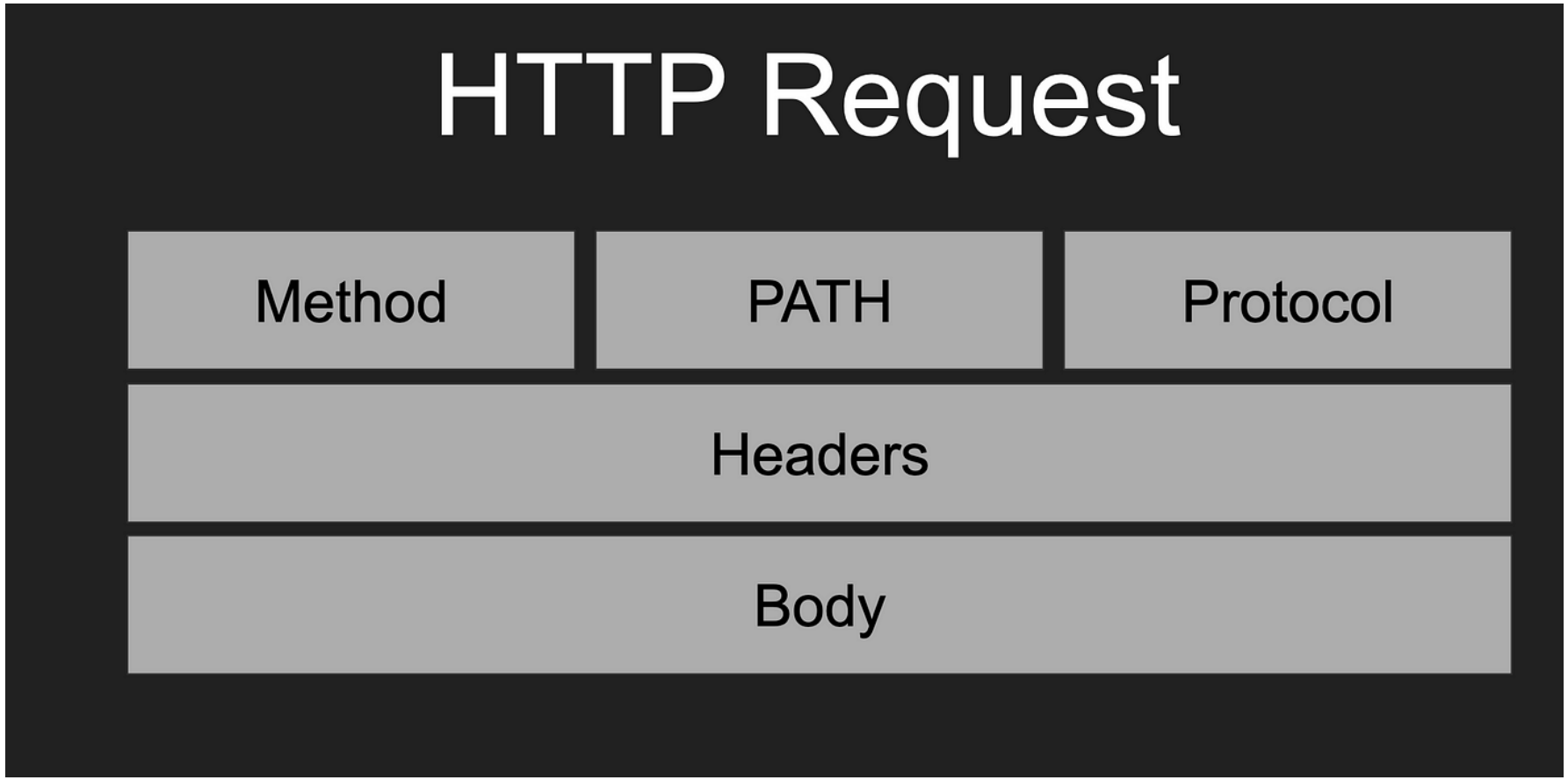
RDT discussions/design choices

- ACK/NAK vs ACK only
- Sequence numbers
- Retransmissions
- Timeout
- Stop and wait

Project related questions

- Know how to read HTTP request and response
- Know how thread works
- Know what each socket is doing

HTTP Request Message Format



HTTP GET request format

- Purpose: retrieves data from the server
- Does GET request have body?

Example Response for HTTP GET

HTTP/1.1 200 OK	**Status Line**
Date: Thu, 20 May 2004 21:12:58 GMT Connection: close	**General Headers**
Server: Apache/1.3.27 Accept-Ranges: bytes	**Response Headers**
Content-Type: text/html Content-Length: 170 Last-Modified: Tue, 18 May 2004 10:14:49 GMT	**Entity Headers**
<html> <head> <title>Welcome to the Amazing Site!</title> </head> <body> This site is under construction. Please come back later. Sorry! </body> </html>	**Message Body**

HTTP CONNECT request

```
CONNECT www.example.com:443 HTTP/1.1
Host: www.example.com:443
Proxy-Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/92.0.4515.159 Safari/537.36
```

- Purpose: Establishes a tunnel to the server, typically used for SSL/TLS connections.

- Does it have body?