EECS 489
Computer Networks

Winter 2025

Mosharaf Chowdhury

Material with thanks to Aditya Akella, Sugih Jamin, Philip Levis, Sylvia Ratnasamy, Peter Steenkiste, and many other colleagues.

Logistics

- You’re ONLY allowed one double-sided 8.5in x 11in note sheet
  - Basic calculators are allowed but you won’t need it

General guidelines (1)

- Test only assumes material covered in lecture, discussion sections, quizzes, and assignments
  - Text: only to clarify details and context for the above
- The test doesn't require you to do complicated calculations
  - Use this as a hint to determine if you're on right track
- You don't need to memorize anything
- You do need to understand how things work

General guidelines (2)

- Be prepared to:
  - Weigh design options outside of the context we studied them in
  - Contemplate new designs we haven't covered in detail but can be put together
    - e.g., I introduce a new IP address format; how does this affect..."
  - Reason from what you know about the pros/cons of solutions we did study

General guidelines (3)

- Exam format
  - MCQ questions
  - Networking use cases
    - Questions not ordered in terms of complexity

General guidelines (4)

- 75-minute midterm exam
  - Friday, Feb. 21 from 7PM - 8:30PM
    - » Room assignments (by the first two letters of your uniqname) are as follows:
      - AA - DB: Chrysler 133
      - DC - ZZ: Chrysler 220
  - SSD accommodations
    - » Friday, Feb. 21 from 6PM at DOW 2150
  - Other conflicts
    - » You have received separate emails by now

What's covered in the midterm?

- Lectures: 1—9 (Up to "More Congestion Control")
- Assignments: 1
- Discussion sections: 1—6 (Content related to lectures 1—9 and assignment 1)

This review

- Walk through what you're expected to know at this point: key topics, important aspects of each
- Not covered in review does NOT imply you don't need to know it
  - But if it's covered today, you should know it
- Summarize, not explain
  - Stop me when you want to discuss something further!

Topics

- Basics (lectures 1–2)
- Application layer (lectures 3–5)
  - HTTP, DNS, CDN, Video Streaming, and Cloud
- Transport layer (lectures 6–9)
  - UDP vs. TCP
  - TCP details: reliability and flow control
  - TCP congestion control: general concepts only

Basic concepts

- You should know:
  - Packet vs. circuit switching
  - Statistical multiplexing
  - Link characteristics
  - Packet delays

Switched networks

- End-systems and networks connected by switches instead of directly connecting them
- Allows us to scale
  - For example, directly connecting N nodes to each other would require N^2 links!

Two approaches to sharing

- Packet switching
  - Network resources consumed on demand per-packet
  - Admission control: per packet
- Circuit switching
  - Network resources reserved a priori at “connection” initiation
  - Admission control: per connection

Statistical multiplexing

- Allowing more demands than the network can handle
  - Hoping that not all demands are required at the same time
  - Good for bursty traffic (average << peak demand)
  - Packet switching exploits statistical multiplexing better than circuit switching

Delay

- Consists of four components
  - Transmission delay
  - Propagation delay
  - Queuing delay
  - Processing delay

due to link properties

due to traffic mix and switch internals

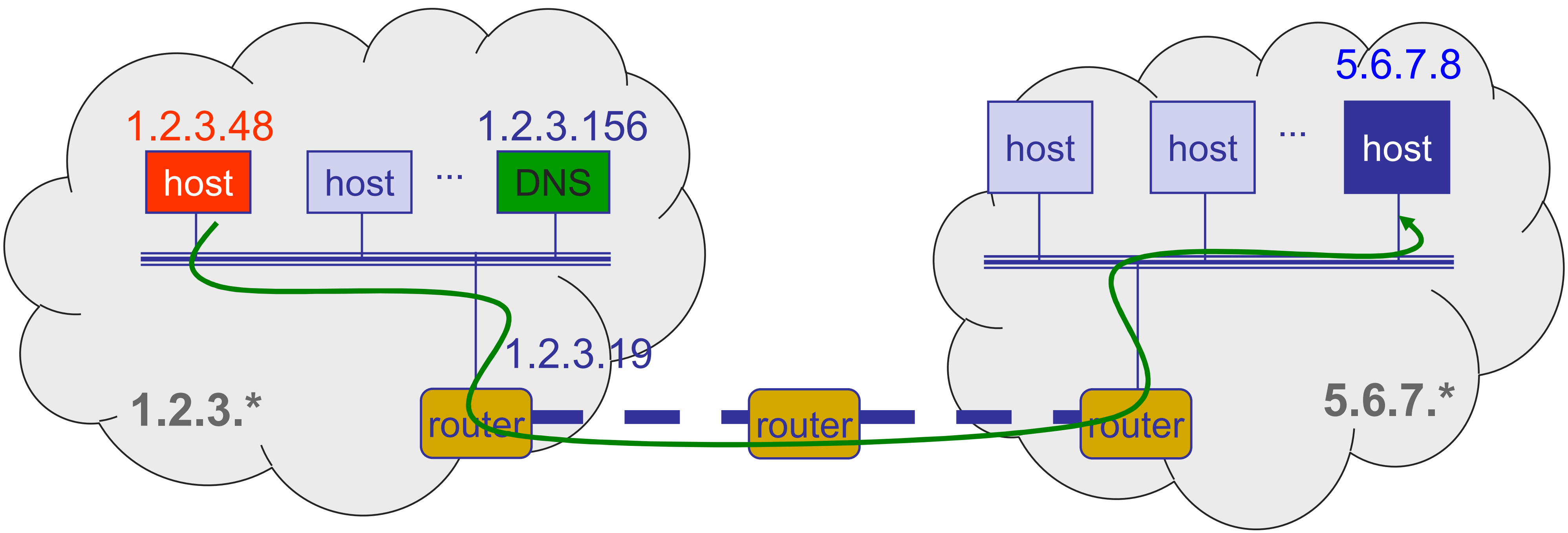
End-to-end delay

transmission
propagation
queueing
processing
transmission
propagation
queueing
processing
transmission
propagation

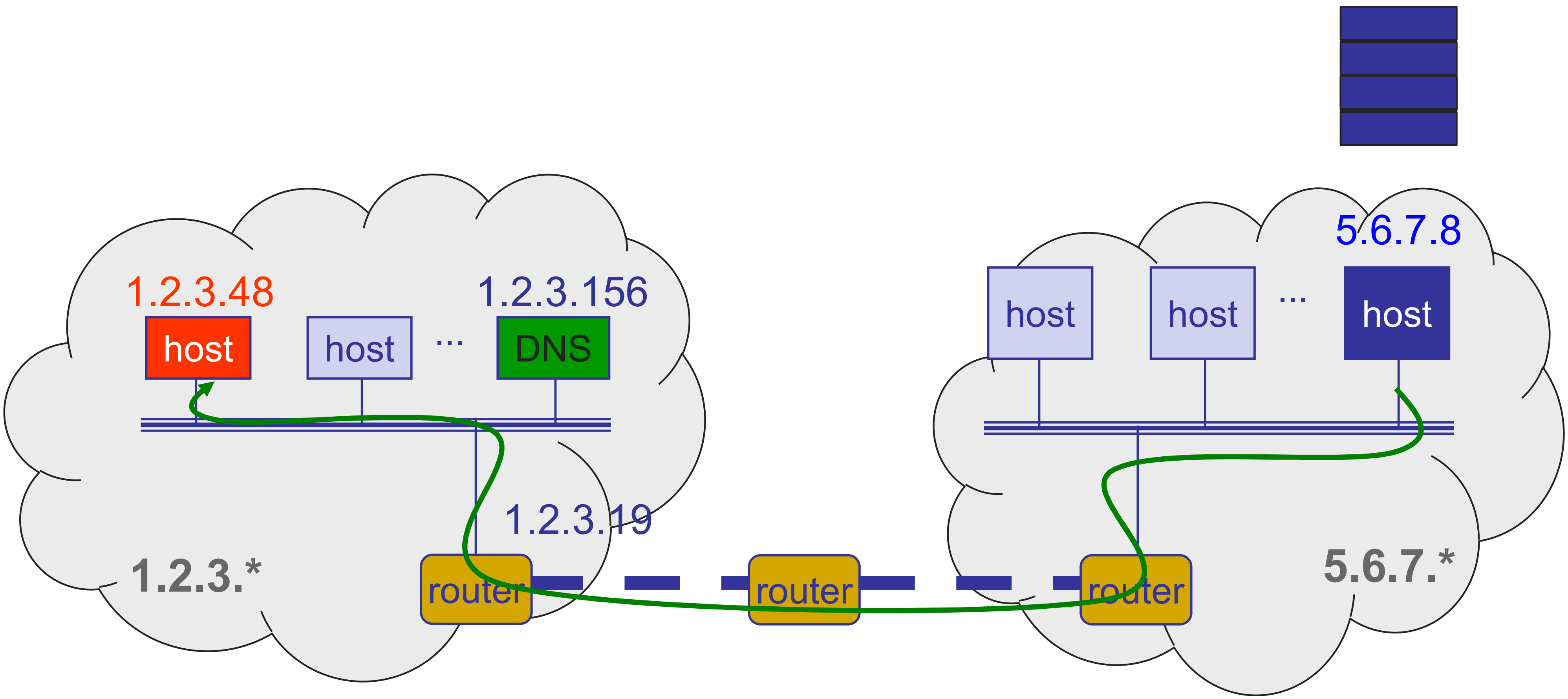
We only consider store-and-forward in this class

What we want

(Some of) What happens…



(More of) What happens



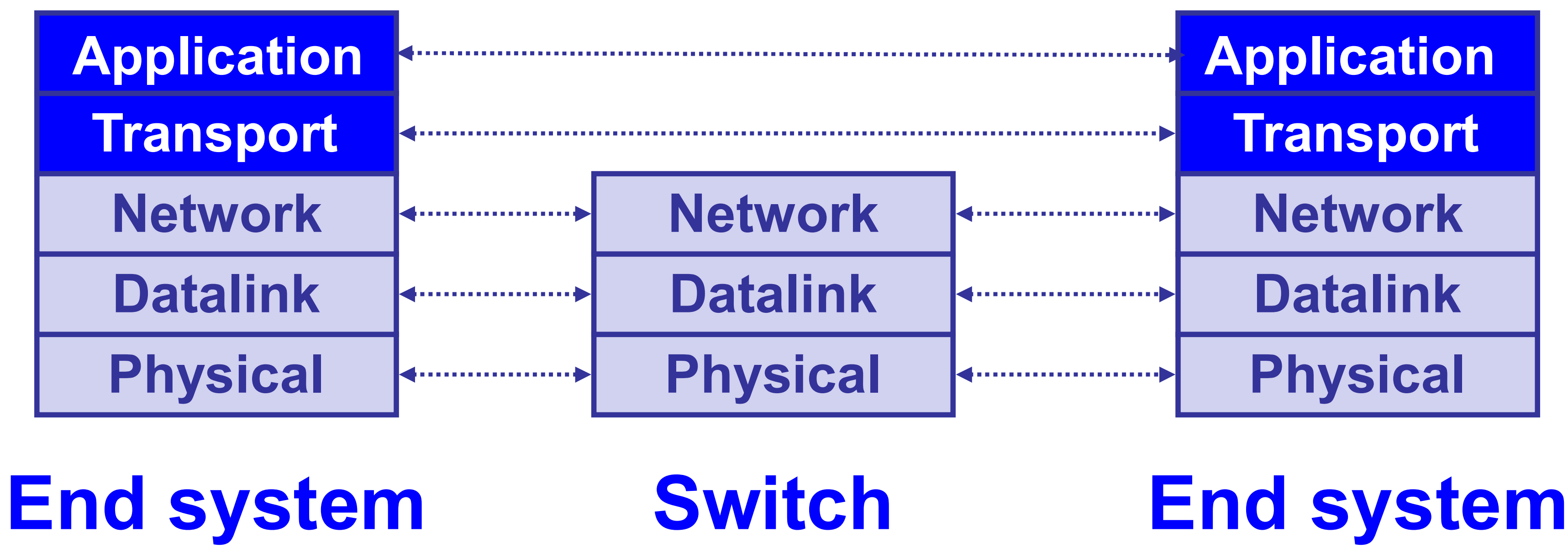
What we get

Layers

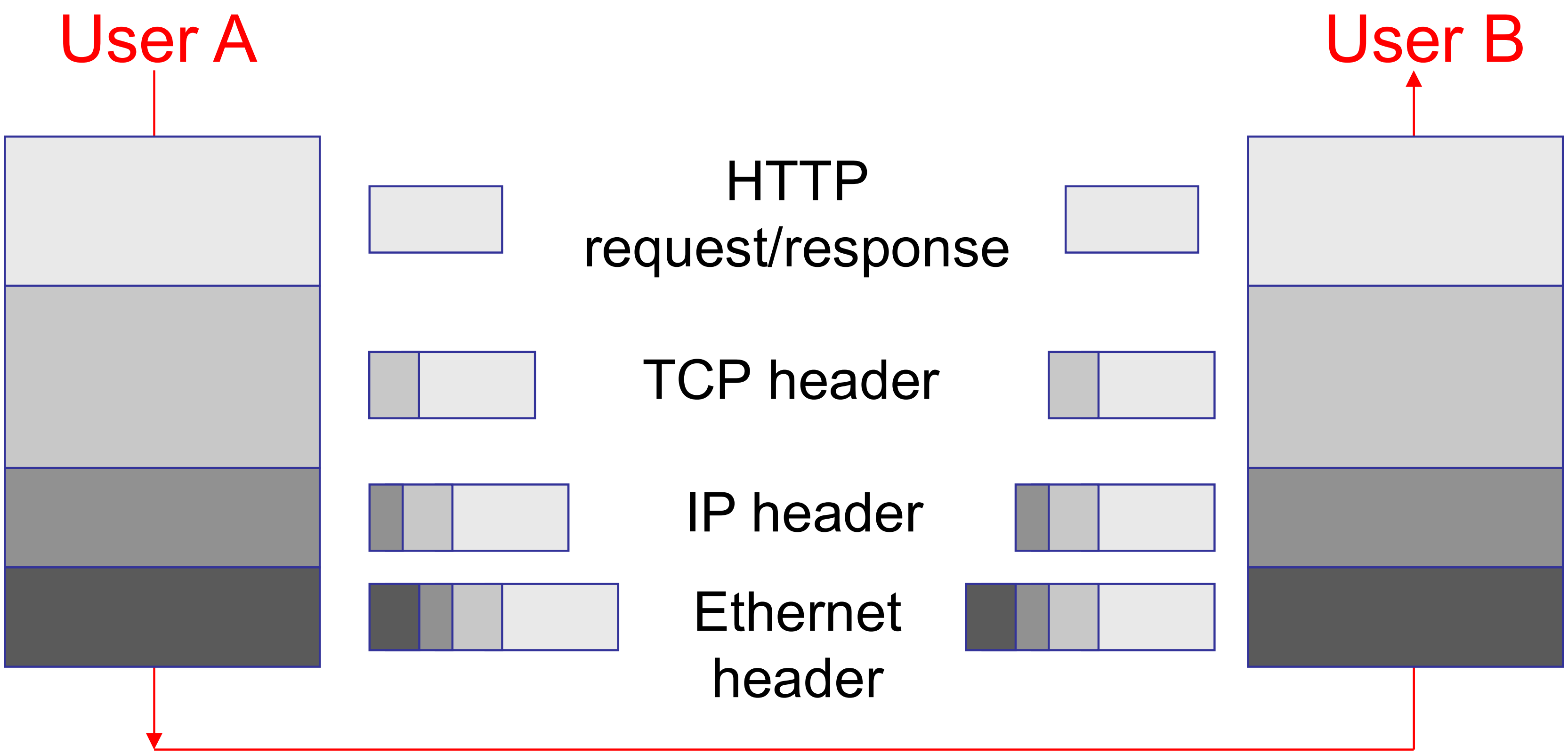
- Layer: a part of a system with well-defined interfaces to other parts
- One layer interacts only with layer above and layer below
- Two layers interact only through the interface between them

Layers in practice

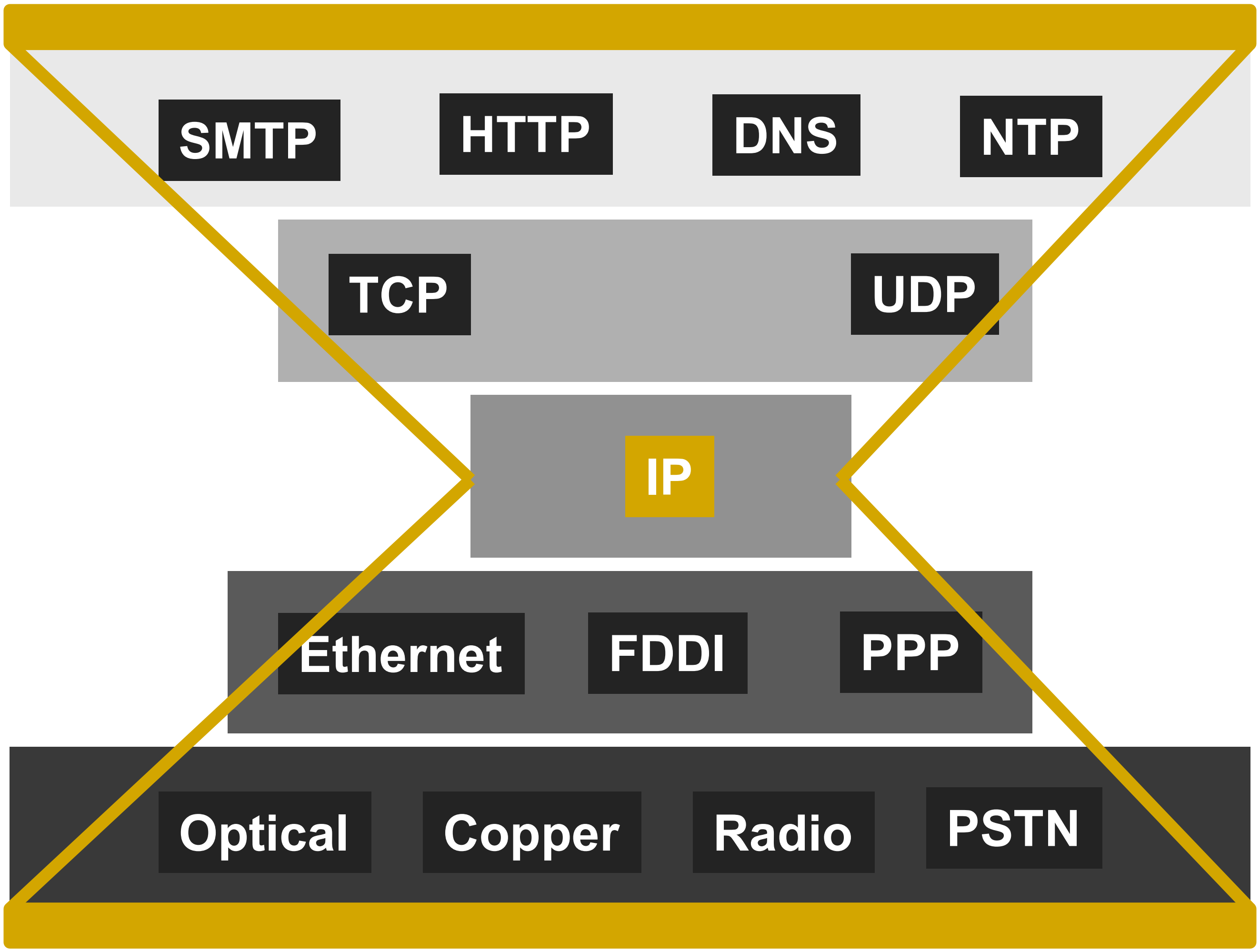
- Lower three layers implemented everywhere
- Top two layers implemented only at hosts



Layer encapsulation: Protocol headers



IP is the narrow waist of the layering hourglass



Topics

- Basics (lectures 1–2)
- Application layer (lectures 3–5)
  - HTTP, DNS, CDN, Video Streaming, and Cloud
- Transport layer (lectures 6–9)
  - UDP vs. TCP
  - TCP details: reliability and flow control
  - TCP congestion control: general concepts only

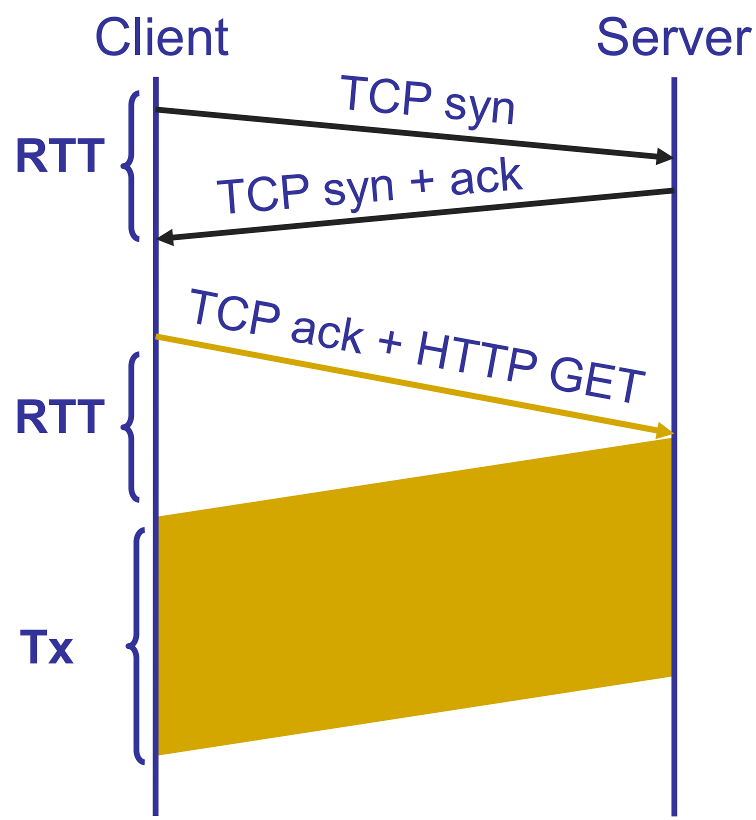
Hyper Text Transfer Protocol (HTTP)

- Client-server architecture
  - Server is “always on” and “well known”
  - Clients initiate contact to server
- Synchronous request/reply protocol
  - Runs over TCP, Port 80
- Stateless
- ASCII format
  - Before HTTP/2

Object request response time

- RTT (round-trip time)
  - Time for a small packet to travel from client to server and back

- Response time
  - 1 RTT for TCP setup
  - 1 RTT for HTTP request and first few bytes
  - Transmission time
  - Total = 2RTT + Transmission Time



Improving HTTP performance

- Optimizing connections using three "P"s
  - Persistent connections
  - Parallel/concurrent connections
  - Pipelined transfers over the same connection
- Caching
  - Forward proxy: close to clients
  - Reverse proxy: close to servers
- Replication

Scorecard: Getting n small objects

- Time dominated by latency

- One-at-a-time: ~2n RTT
- m concurrent: ~2[n/m] RTT
- Persistent: ~ (n+1) RTT
- Pipelined: ~2 RTT
- Pipelined and Persistent: ~2 RTT first time; RTT later for another n from the same site

Scorecard: Getting n large objects each of size F

- Time dominated by TCP throughput B_C ($\le B_L$), where bottleneck link bandwidth is B_L
  - Assuming all TCP connections go through the same bottleneck link
- One-at-a-time: $\sim nF/B_C$
- m concurrent: $\sim nF/(mB'_C)$
  - Assuming each TCP connection gets the same throughput (B'_C), where $mB'_C \le B_L$
- Pipelined and/or persistent: $\sim nF/B_C$
  - The only thing that helps is higher throughput

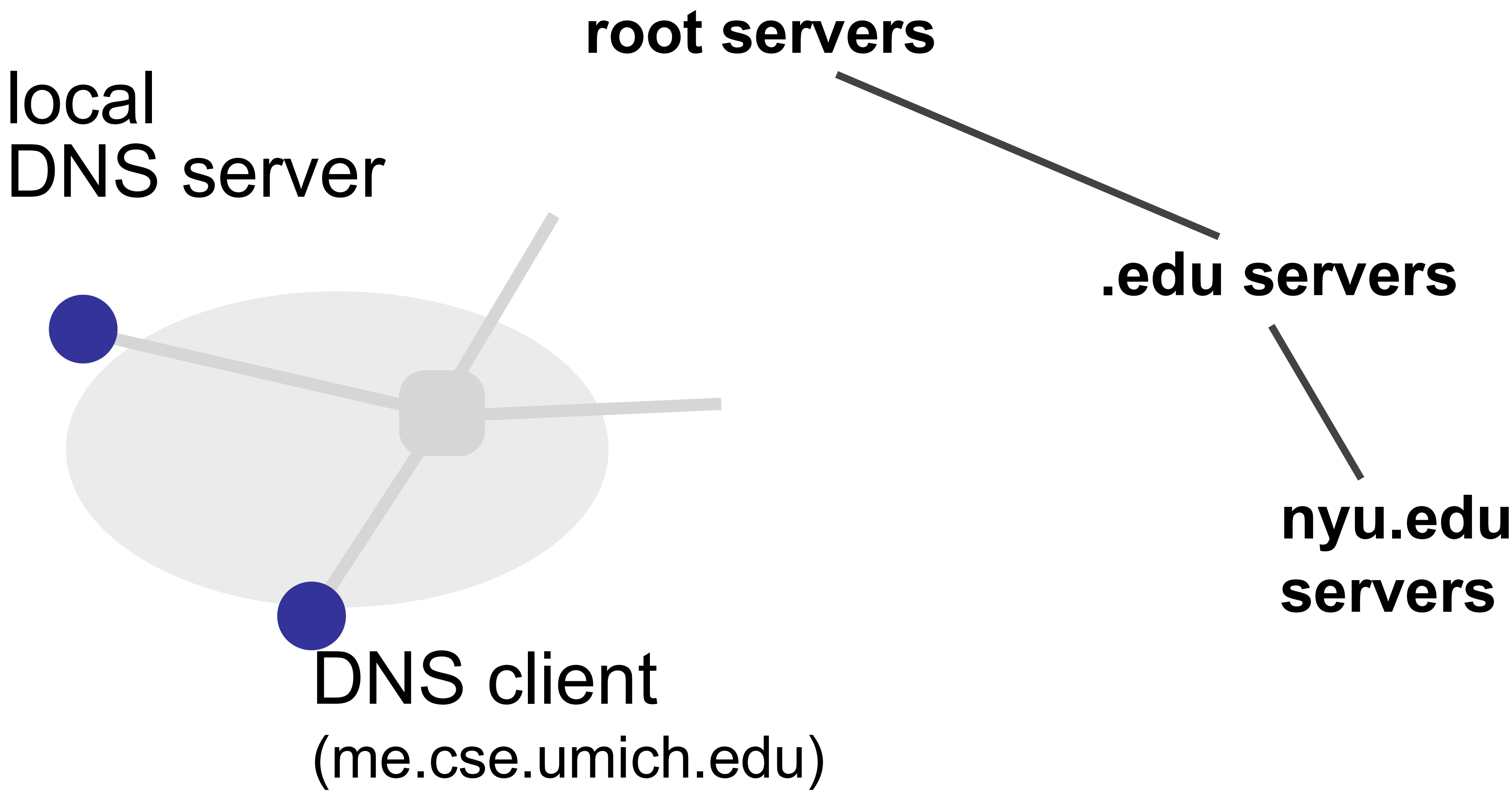
Content Distribution Networks (CDN)

- Caching and replication as a service
- Combination of caching and replication
  - Pull: Direct result of clients' requests (caching)
  - Push: Expectation of high access rate (replication)

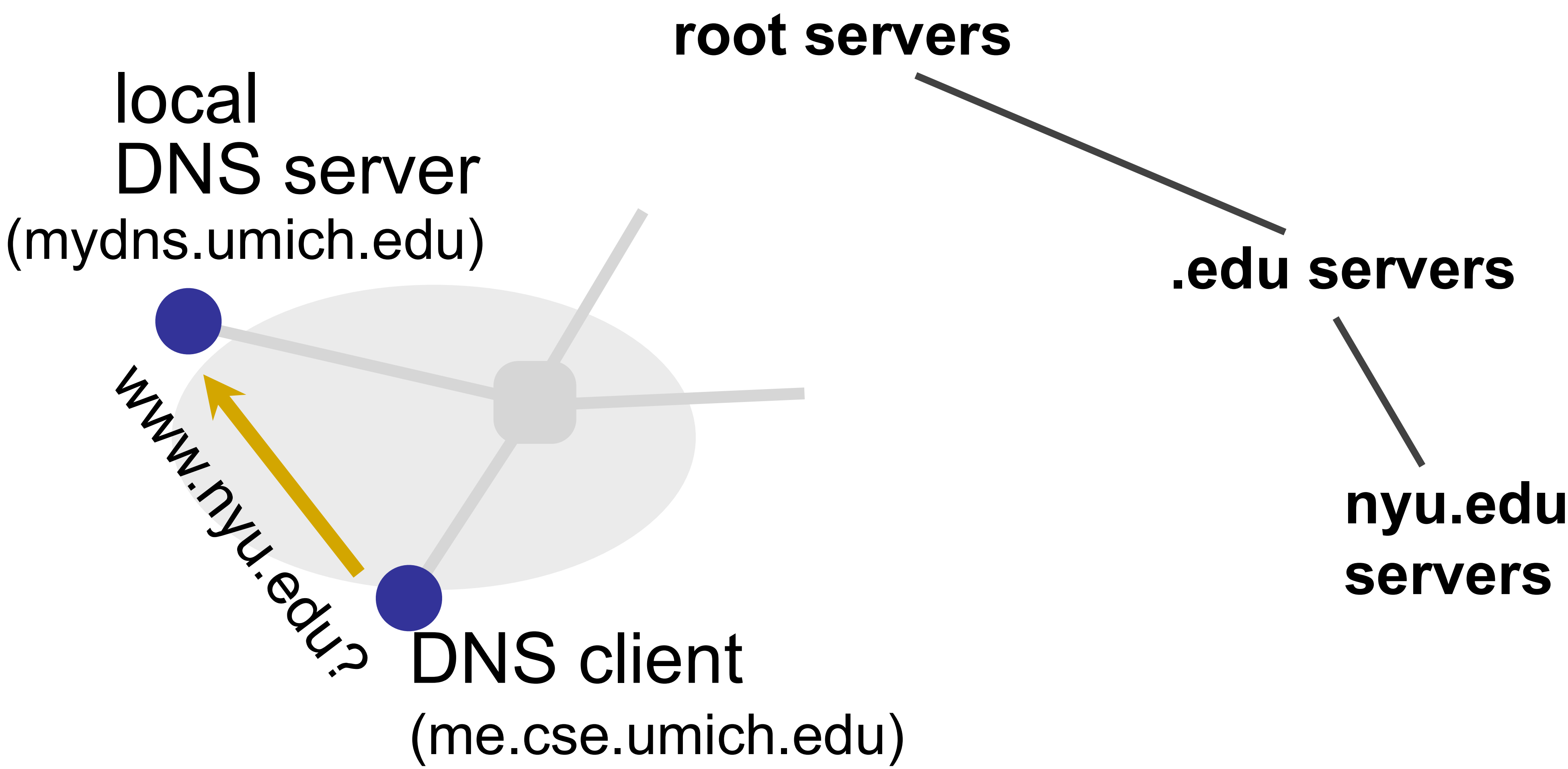
Hierarchies in the DNS

- Three intertwined hierarchies
  - Hierarchical namespace
    - As opposed to flat namespace
  - Hierarchically administered
    - As opposed to centralized
  - (Distributed) hierarchy of servers
    - As opposed to centralized storage

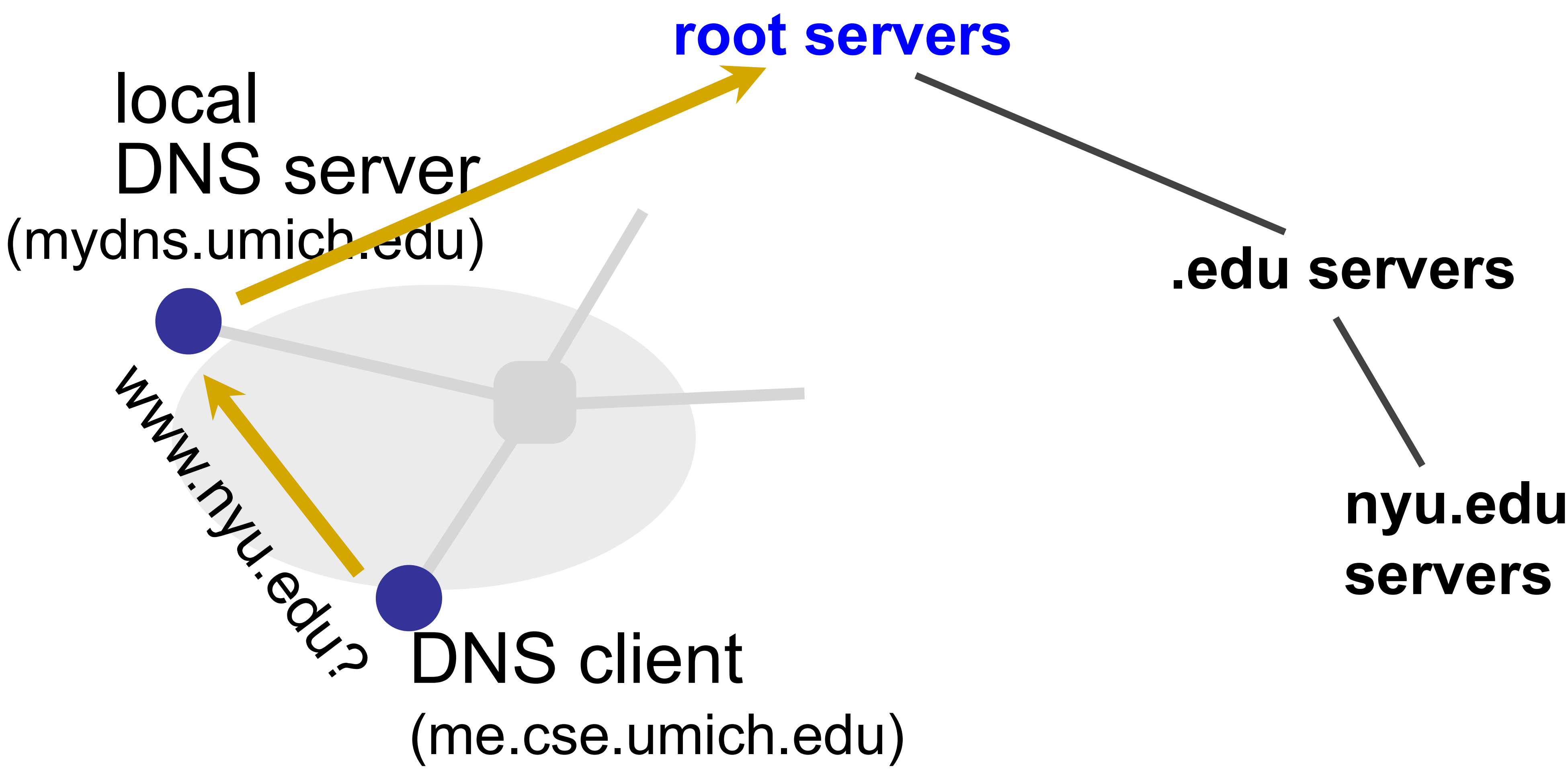
Name resolution



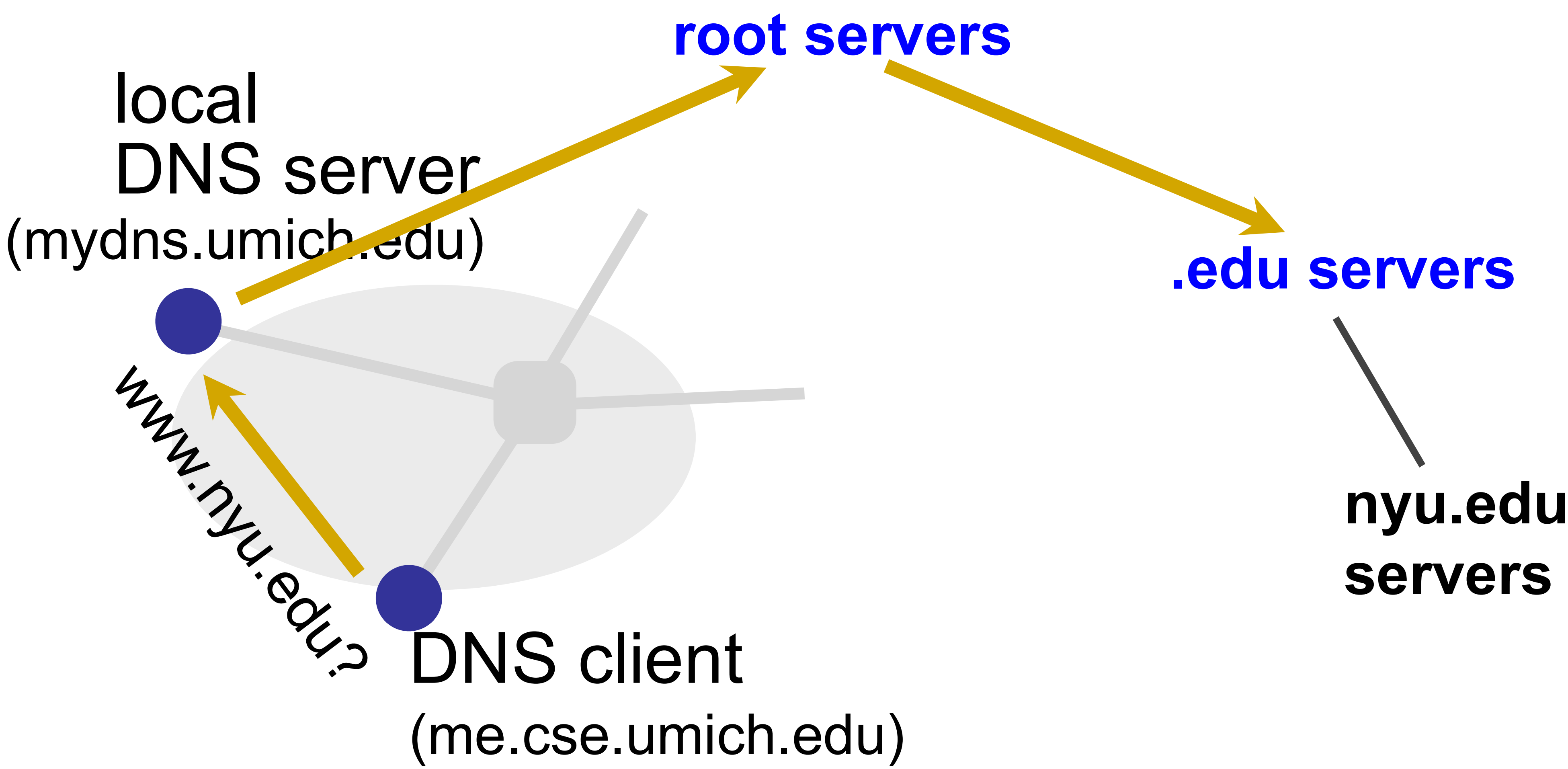
Name resolution



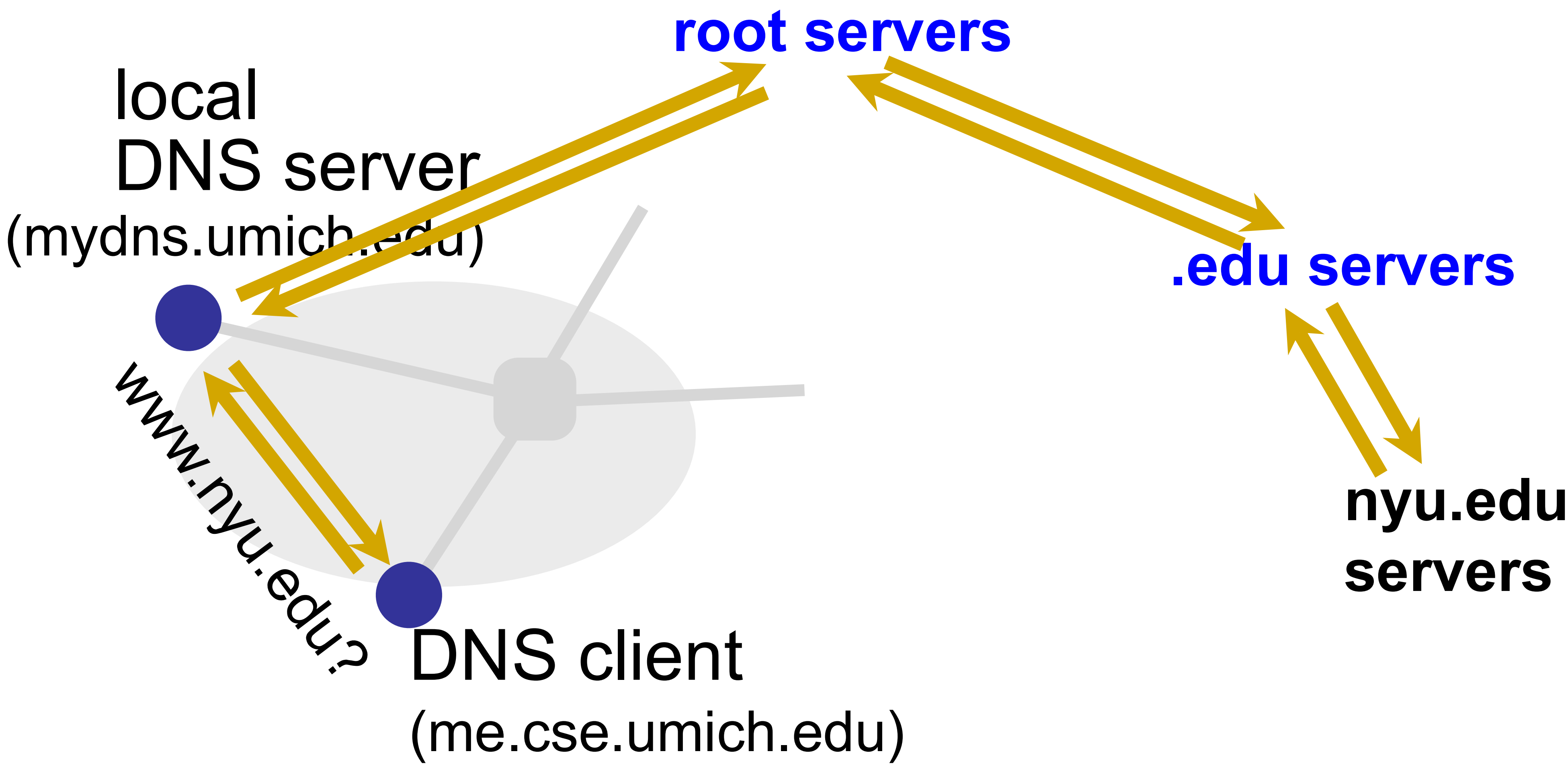
Name resolution



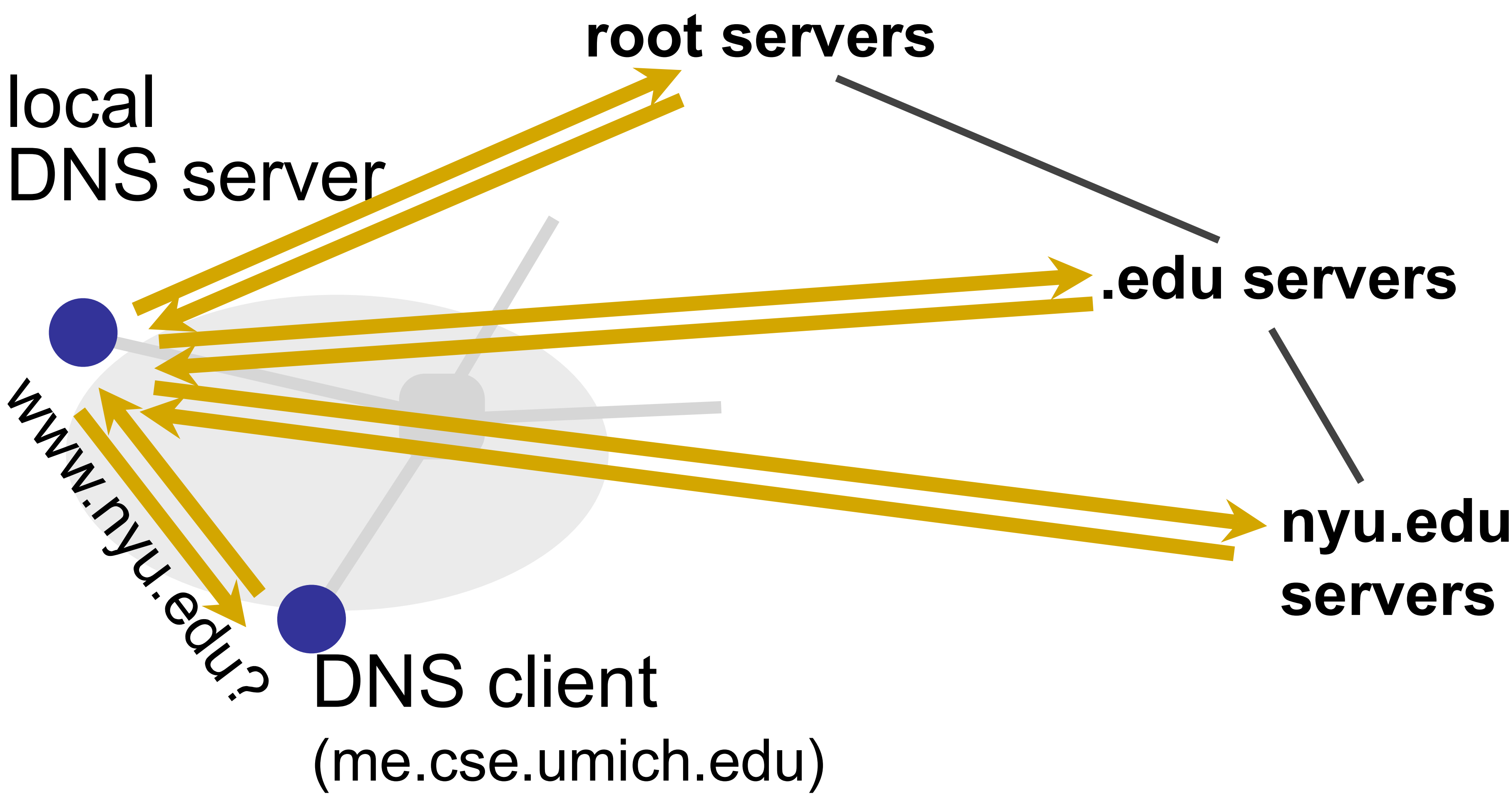
Name resolution



Name resolution: Recursive



Name resolution: Iterative



DNS caching

- Performing all these queries takes time
  - Up to 1-second latency before starting download
- Caching can greatly reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.google.com) visited often
  - Local DNS server often has the information cached
- How DNS caching works
  - DNS servers cache responses to queries
  - Responses include a "time to live" (TTL) field
  - Server deletes cached entry after TTL expires

5-MINUTE BREAK!

Announcements

- Please fill up midterm teaching evaluation

HTTP streaming

- Video is stored at an HTTP server with a URL
- Clients send a GET request for the URL
- Server sends the video file as a stream
- Client first buffers for a while to minimize interruptions later
- Once the buffer reaches a threshold
  - The video plays in the foreground
  - More frames are downloaded in the background

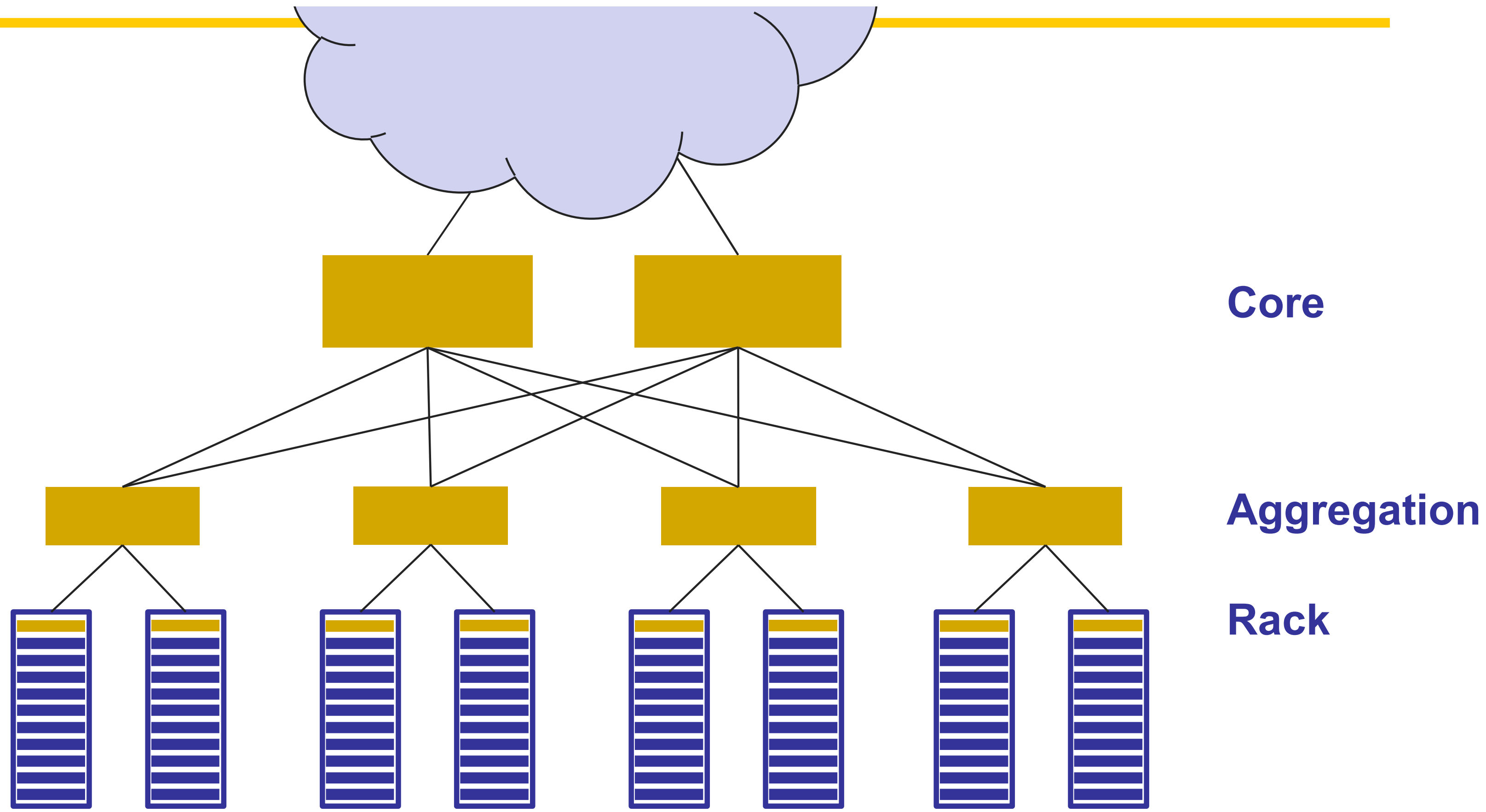
DASH : Dynamic Adaptive Streaming over HTTP

- Keep multiple resolutions of the same video
  - Stored in a manifest file in the HTTP server
- Client asks for the manifest file first to learn about the options
- Asks for chunks at a time and measures available bandwidth while they are downloaded
  - Low bandwidth $\Rightarrow$ switch to lower bitrate
  - High bandwidth $\Rightarrow$ switch to higher bitrate

Applications

- Common theme: parallelism
  - Applications decomposed into tasks
  - Running in parallel on different machines
- Two common paradigms
  - Partition-Aggregate
  - Map-Reduce

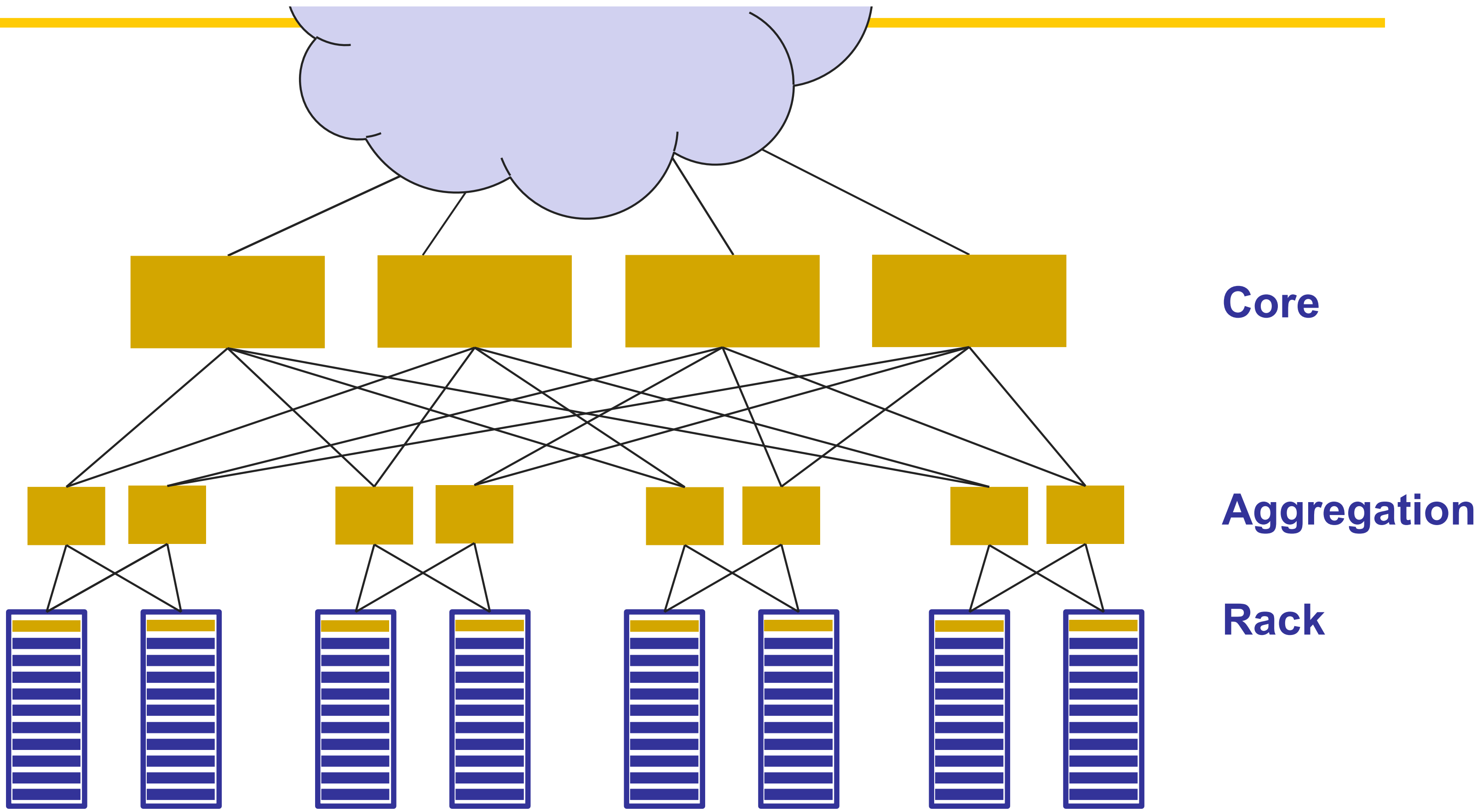
Traditional datacenter networks



Challenges

- Not enough bandwidth
  - Oversubscription: Less bandwidth in the ToR-Agg links than all the servers' bandwidth in the rack
  - Oversubscription ratio: Ratio between bandwidth underneath and bandwidth above
- Not enough paths between server pairs
  - Load balancing issues
  - Failure recovery issues

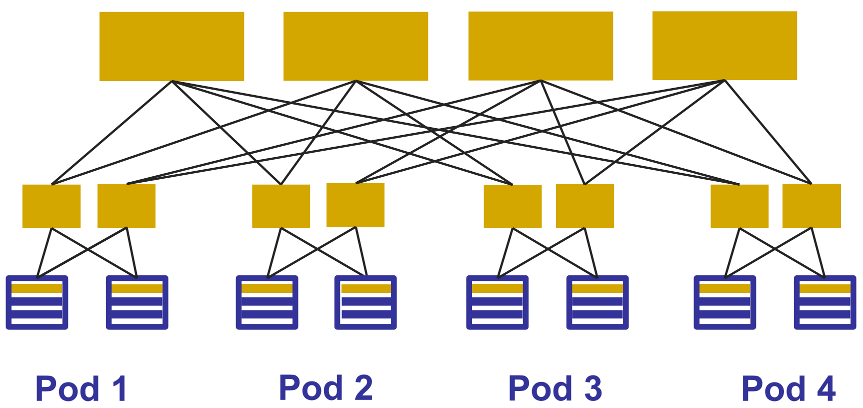
Modern datacenter networks: More bandwidth, more paths



Clos topology

- Multi-stage network
- k pods, where each pod has two layers of k/2 switches
  - k/2 ports up and k/2 down
- All links have the same b/w
- At most $k^3/4$ machines

- Example
  - k = 4
  - 16 machines
- For k=48, 27648 machines



Topics

- Basics (lectures 1–2)
- Application layer (lectures 3–5)
  - HTTP, DNS, CDN, Video Streaming, and Cloud
- Transport layer (lectures 6–9)
  - UDP vs. TCP
  - TCP details: reliability and flow control
  - TCP congestion control: general concepts only

Role of the transport layer

- (1) Communication between application processes
  - Mux and demux from/to application processes
  - Implemented using ports
- (2) Provide common end-to-end services for app layer
  - Reliable, in-order data delivery
  - Well-paced data delivery

UDP vs. TCP

- Both UDP and TCP perform mux/demux via ports

	UDP	**TCP**
Data abstraction	Packets (datagrams)	Stream of bytes of arbitrary length
Service	Best-effort (same as IP)	•Reliability •In-order delivery •Congestion control •Flow control

Reliable transport: General concepts

- Checksums (for error detection)
- Timers (for loss detection)
- Acknowledgments (feedback from receiver)
  - Cumulative: “received everything up to X”
  - Selective: “received X”
- Sequence no (detect duplicates, accounting)
- Sliding windows (for efficiency)

You should know:

- what these concepts are
- why they exist
- how TCP uses them

Designing a reliable transport protocol

- Stop and wait is correct but inefficient
  - Works packet by packet (of size DATA)
  - Throughput is (DATA/ RTT)
- Sliding window: use pipelining to increase throughput
  - n packets at a time results in higher throughput
  - MIN(n*DATA/RTT, Link Bandwidth)

The TCP abstraction

- TCP delivers a reliable, in-order, byte stream
- Reliable: TCP resends lost packets (recursively)
  - Until it gives up and shuts down connection
- In-order: TCP only hands consecutive chunks of data to application
- Byte stream: TCP assumes there is an incoming stream of data, and attempts to deliver it to app

Things to know about TCP

- How TCP achieves reliability
- RTT estimation
- Connection establishment/teardown
- Flow Control
- Congestion Control (concepts only)

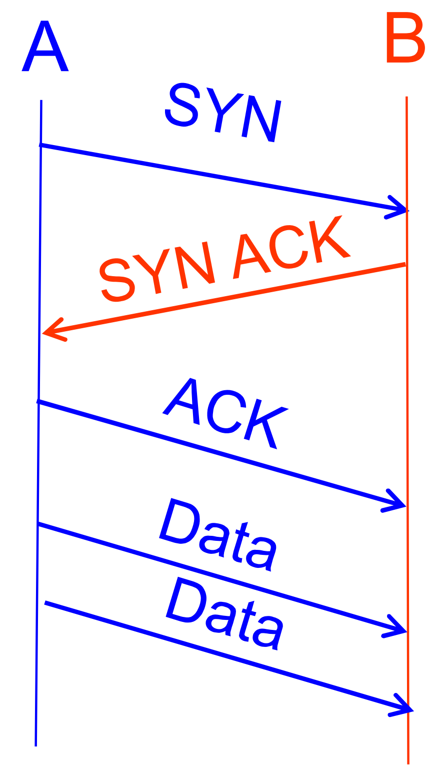
- For each, know how the functionality is implemented and why it is needed

Reliability

- Having TCP take care of it simplifies application development
- How
  - Checksums and timers (for error and loss detection)
  - Fast retransmit (to detect faster-than-timeout loss)
  - Cumulative ACKs (receiver feedback: what’s lost?)
  - Sliding windows (for efficiency)
  - Buffers at sender (hold packets until ACKs arrive)
  - Buffers at receiver (to reorder packets before delivery to application)

Establishing/terminating a TCP connection

- Three-way handshake to establish connection
  - Host A sends a SYN (open; "synchronize sequence numbers") to host B
  - Host B returns a SYN acknowledgment (SYN ACK)
  - Host A sends an ACK to acknowledge the SYN ACK
- Three-way handshake to terminate (normal operation)



Flow control

- Why?
  - TCP at the receiver must buffer a packet until all packets before it (in byte-order) have arrived and the receiving application has consumed available bytes
  - Hence, receiver advances its window when the receiving application consumes data
  - Sender advances its window when new data ACK'd
  - Risk of sender overrunning the receiver's buffers
- How?
  - "Advertised Window" field in TCP header

Congestion control

- Why?
  - Because the network itself can be the bottleneck
  - Should make efficient use of available network capacity
    - While sharing available capacity fairly with other flows
    - And adapting to changes in available capacity
- How?
  - Dynamically adapts the size of the sending window

Put together

- Flow Control
  - Restrict window to RWND to make sure that the receiver isn't overwhelmed
- Congestion Control
  - Restrict window to CWND to make sure that the network isn't overwhelmed
- Together
  - Restrict window to min{RWND, CWND} to make sure that neither the receiver nor the network are overwhelmed

CC implementation

- States at sender
  - CWND (initialized to a small constant)
  - ssthresh (initialized to a large constant)
  - dupACKcount and timer
- Events
  - ACK (new data)
  - dupACK (duplicate ACK for old data)
  - Timeout

Event: ACK (new data)

- If CWND < ssthresh
  - CWND += 1

> - *CWND packets per RTT*
> - *Hence, after one RTT with no drops:*
>   *CWND = 2xCWND*

Event: ACK (new data)

- If CWND < ssthresh
  - CWND += 1

***Slow start** phase*

- Else
  - CWND = CWND + 1/CWND

***Congestion avoidance** phase*

- *CWND packets per RTT*
- *Hence, after one RTT with no drops:*
  CWND = CWND + 1

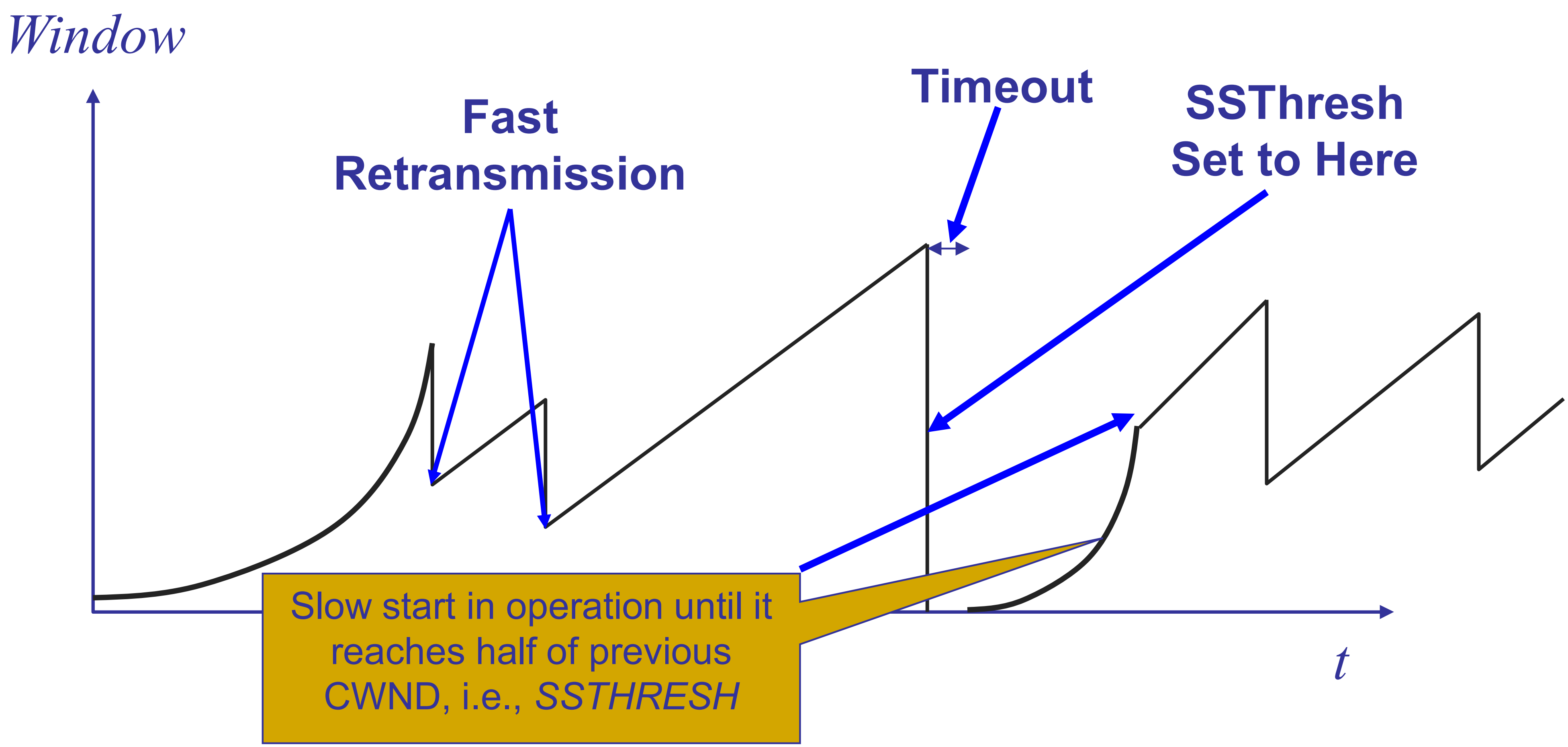
Event: TimeOut

- On Timeout
  - ssthresh ← CWND/2
  - CWND ← 1

Event: dupACK

- dupACKcount ++
- If dupACKcount = 3 /* fast retransmit */
  - ssthresh = CWND/2
  - CWND = CWND/2

Example

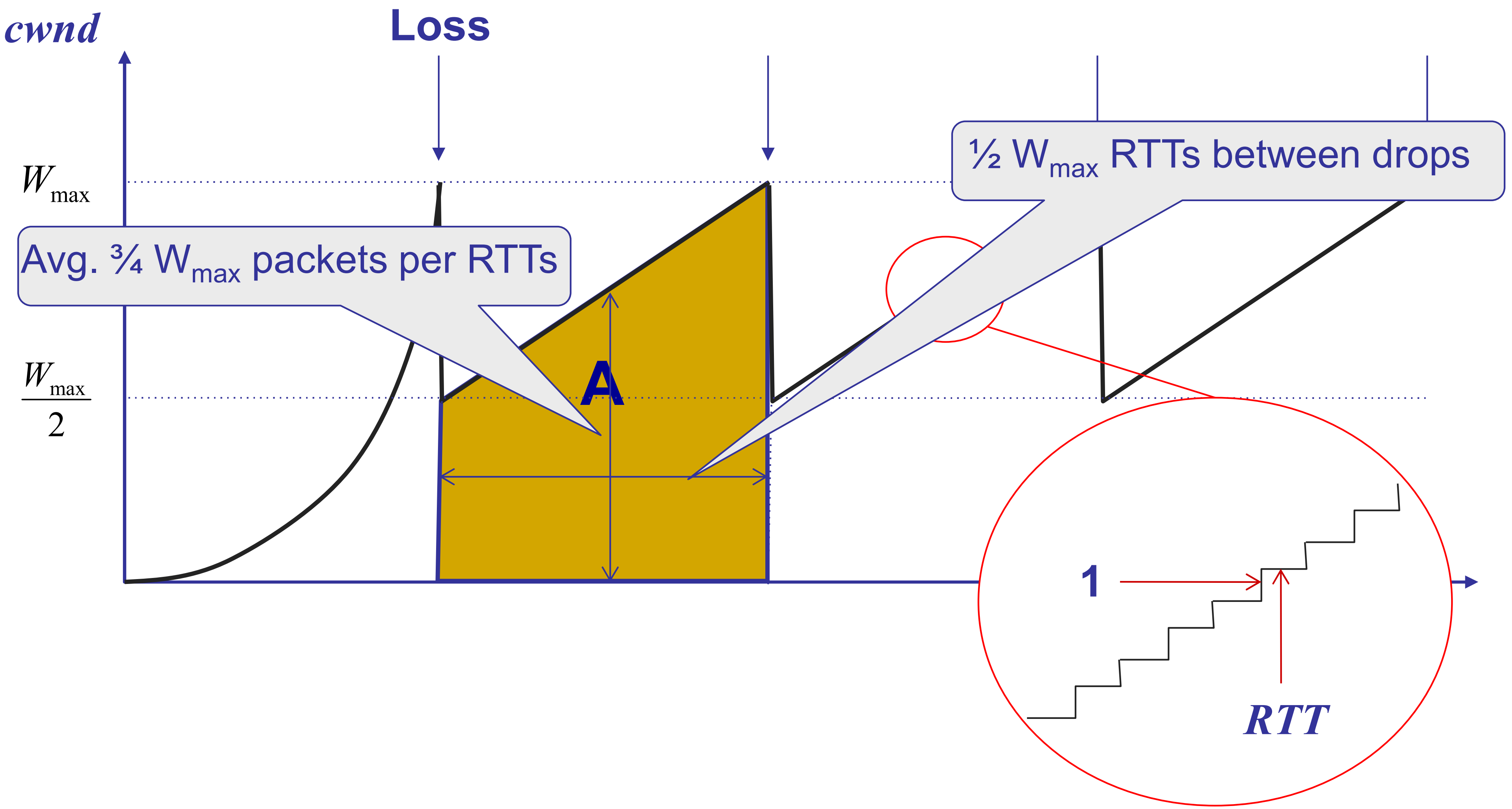


Slow-start restart: Go back to CWND = 1 MSS, but take advantage of knowing the previous value of CWND

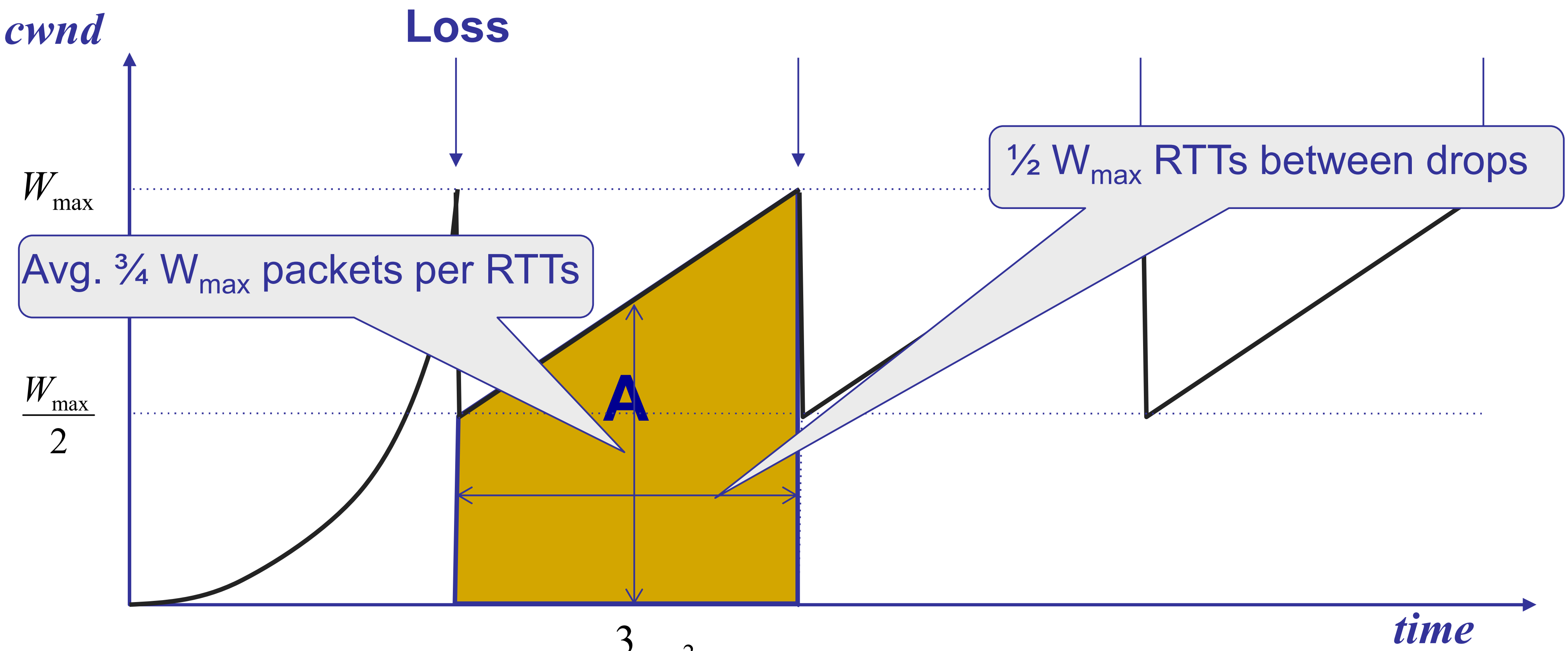
TCP flavors

- TCP-Tahoe
  - CWND =1 on 3 dupACKs
- TCP-Reno
  - CWND =1 on timeout
  - CWND = CWND/2 on 3 dupACKs
- TCP-newReno
  - TCP-Reno + improved fast recovery
- TCP-SACK
  - Incorporates selective acknowledgements

A simple model for TCP throughput



A simple model for TCP throughput



$$A = \frac{3}{8} W_{max}^2$$

Summary

- Practice Exams on GitHub