EECS 489
Computer Networks

Fall 2020

Mosharaf Chowdhury

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

Topics

- Basics (lectures 1–2)
- Application layer (lectures 3–5)
  - HTTP, DNS, and CDN
  - Video Streaming

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!

How are network resources shared?

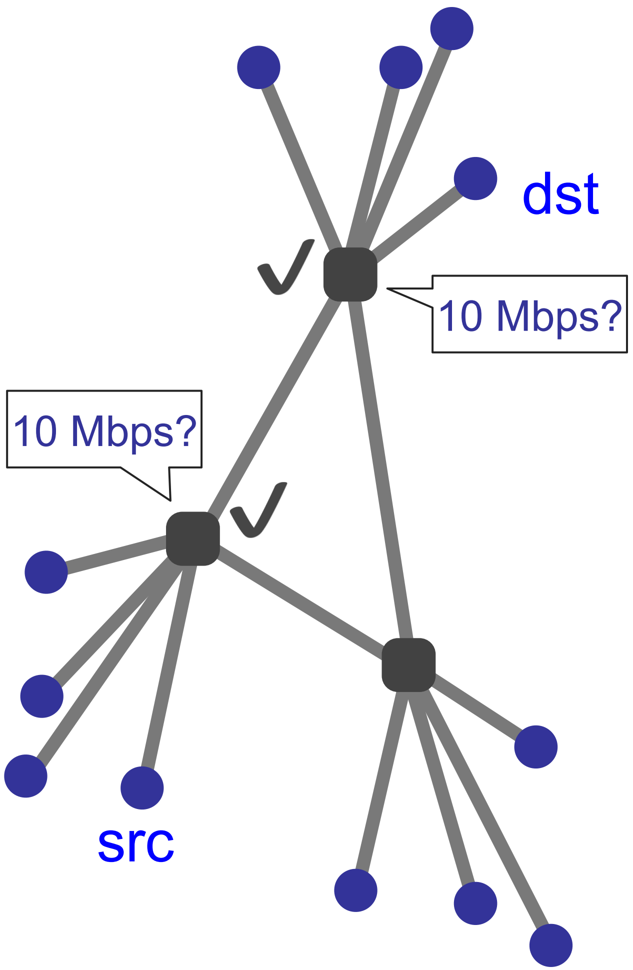
- Two approaches
  - Reservations → circuit switching
  - On-demand → packet switching

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

Circuit switching

1. src sends reservation request to dst
2. Switches create circuit *after* admission control
3. src sends data
4. src sends teardown request



Packet switching

- Data is sent as chunks of formatted bits (Packets)
- Packets consist of a “header” and “payload”
- Switches “forward” packets based on their headers
- Each packet travels independently
- No link resources are reserved

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

Performance metrics

- Delay
- Loss
- Throughput

Delay

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

due to link properties

due to traffic mix and switch internals

A network link

- Link bandwidth
  - Number of bits sent/received per unit time (bits/sec or bps)
- Propagation delay
  - Time for one bit to move through the link (seconds)

Queueing delay

- How long does a packet have to sit in a buffer before it is processed?

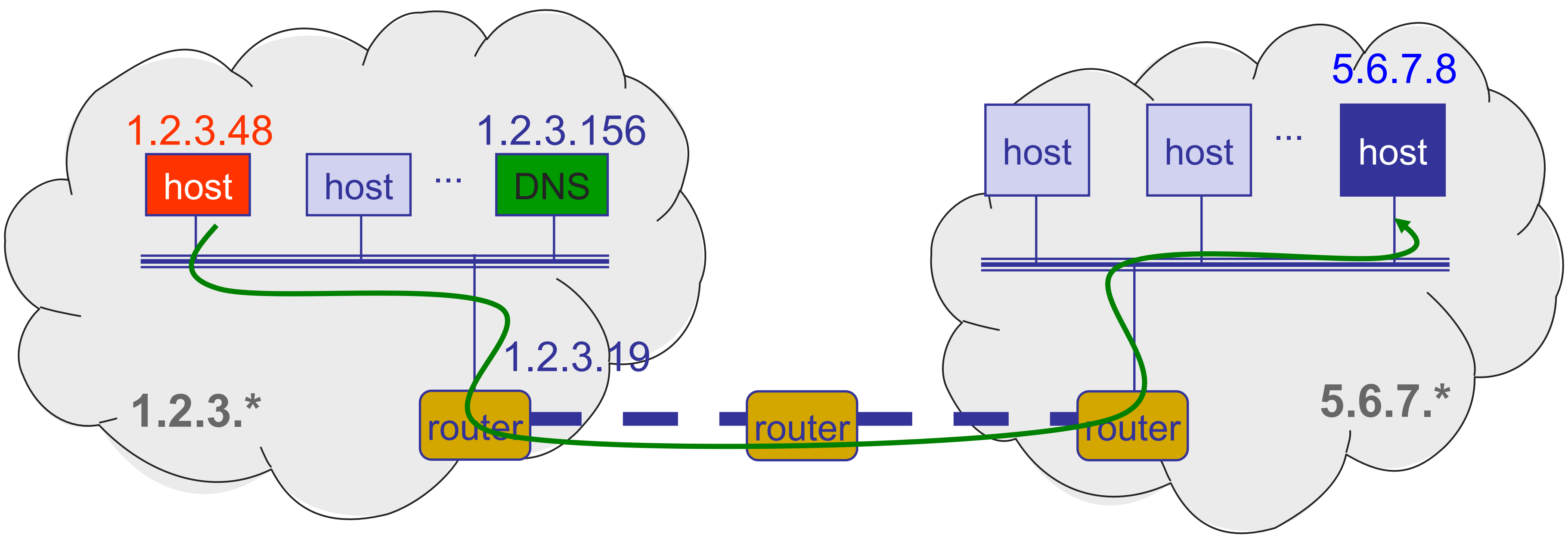
- Characterized with statistical measures
  - Average queuing delay
  - Variance of queuing delay
  - Probability delay exceeds a threshold value

End-to-end delay

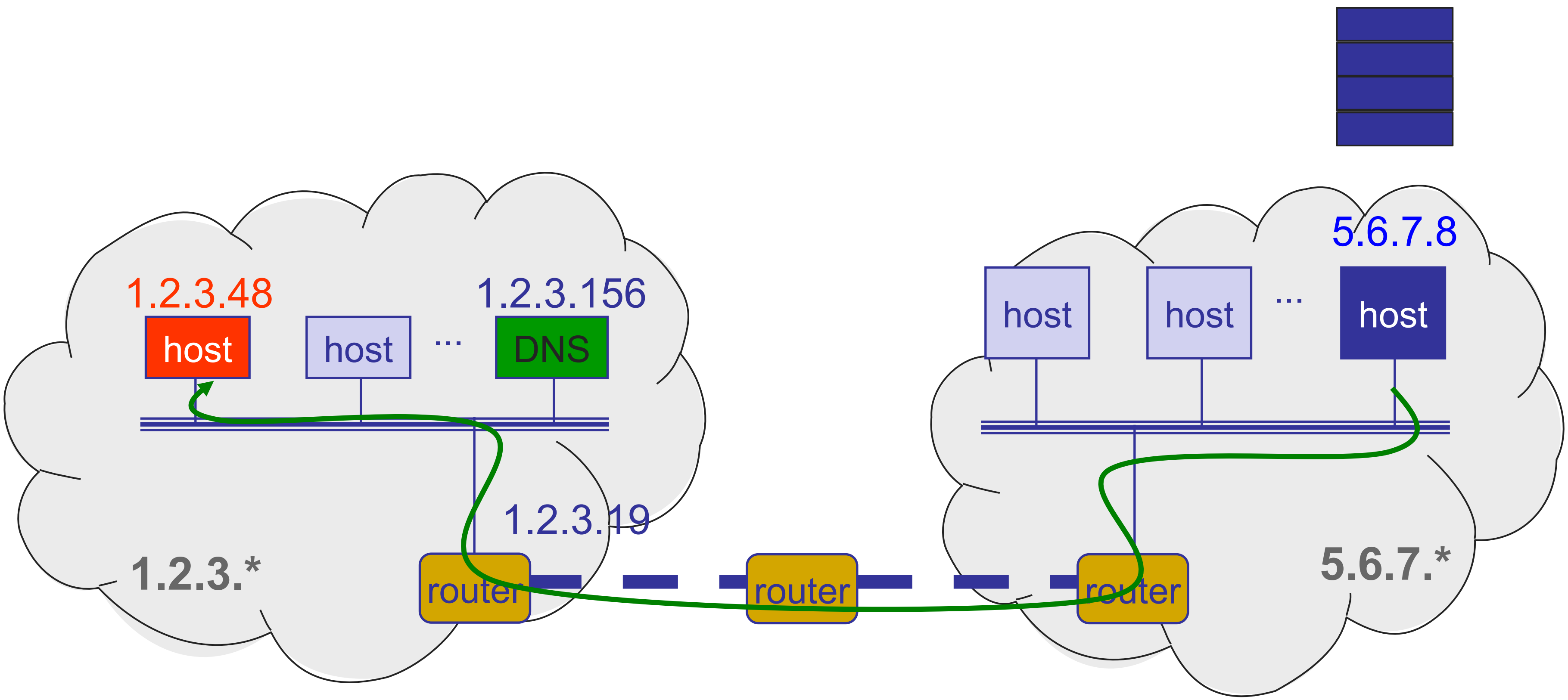
transmission
propagation
queueing
processing
transmission
propagation
queueing
processing
transmission
propagation

What we want

(Some of) What happens…



(More of) What happens



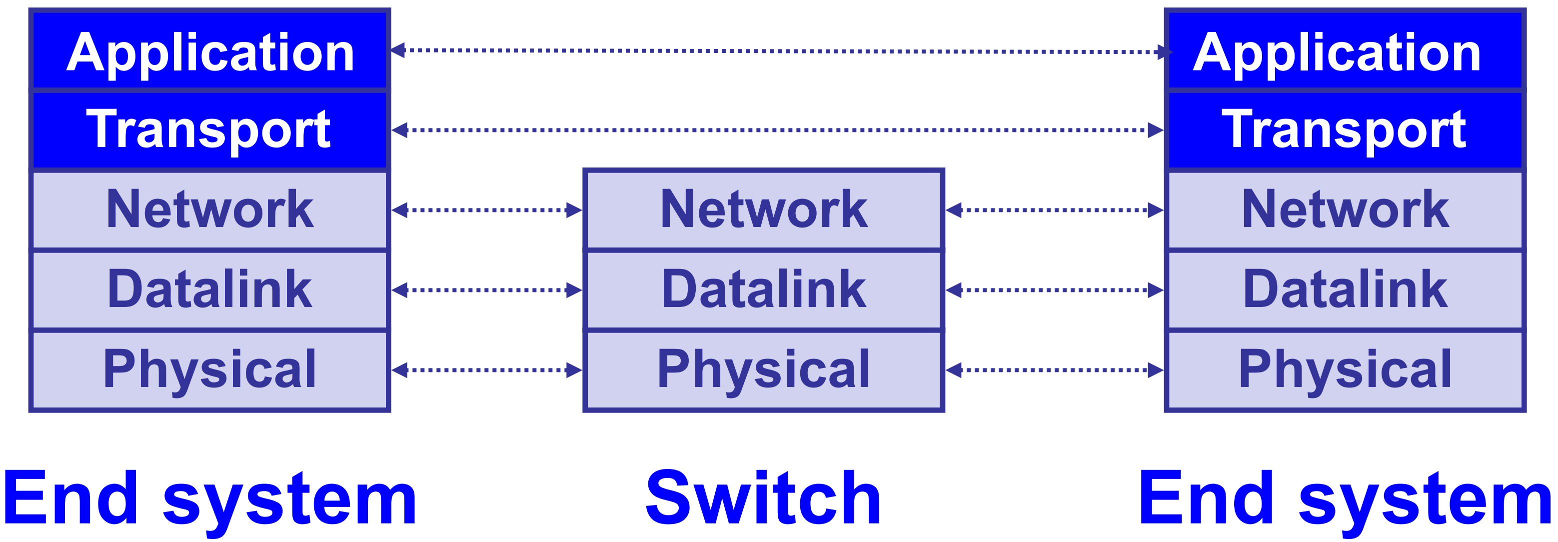
What we get

Layers

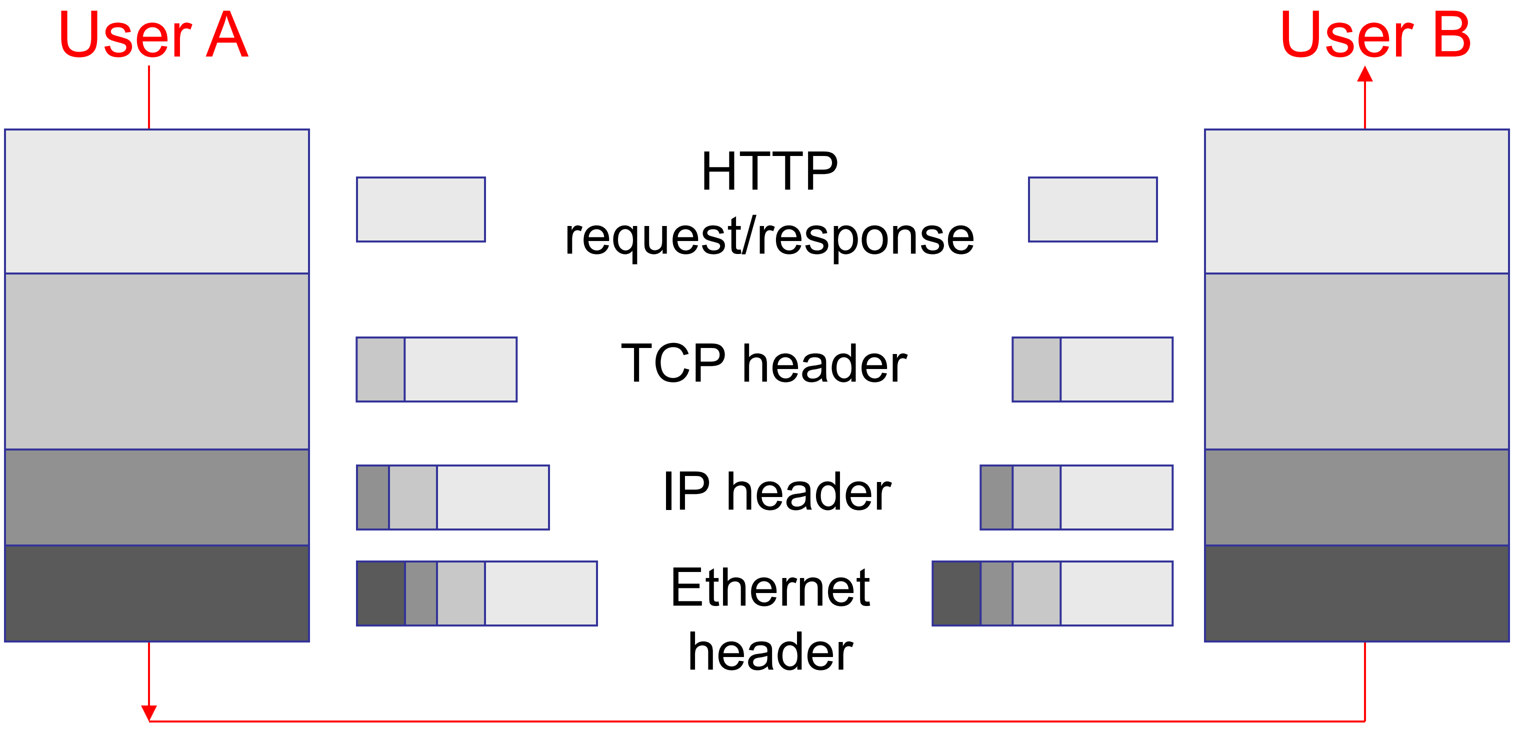
- Layer: a part of a system with well-defined interfaces to other parts
- One layer interacts only with layer above and layer below
- Two layers interact only through the interface between them

Layers in practice

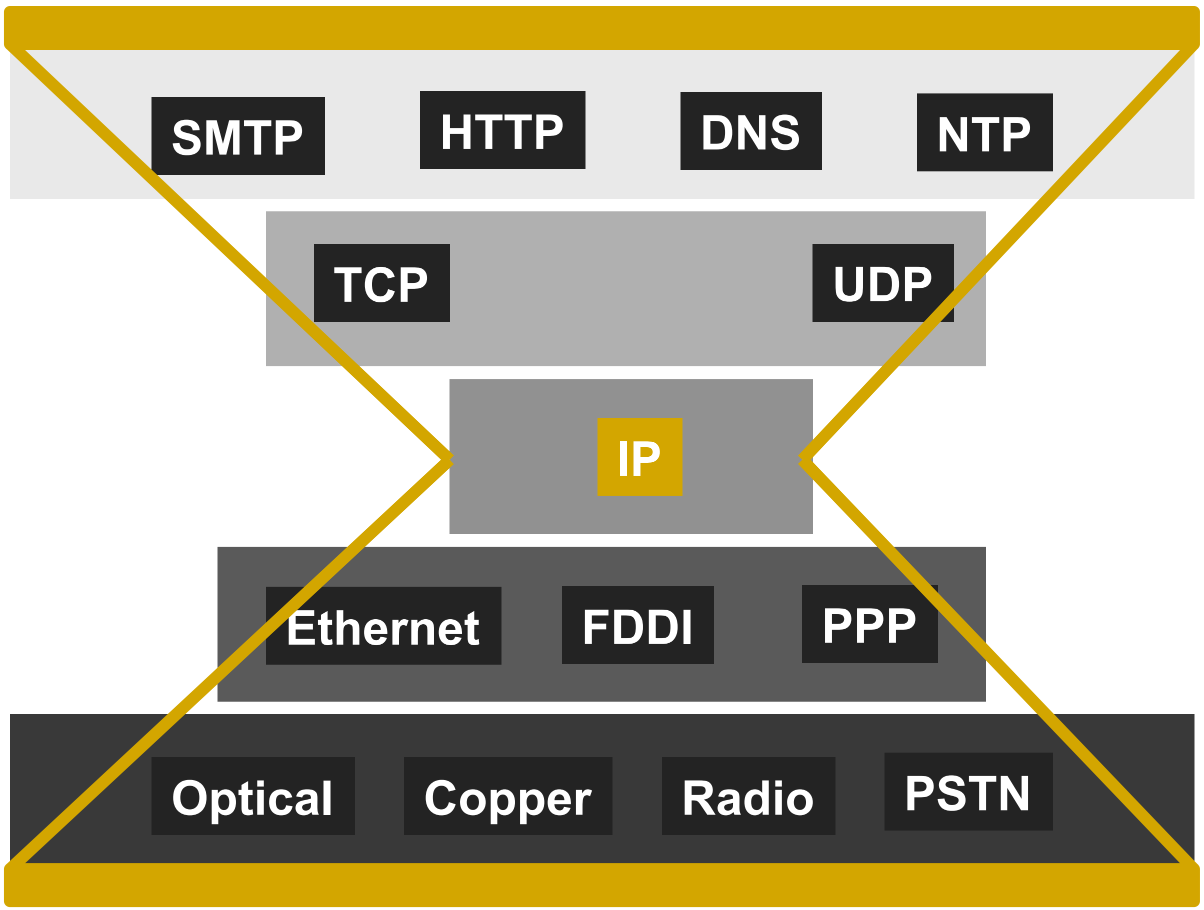
- Lower three layers implemented everywhere
- Top two layers implemented only at hosts



Layer encapsulation: Protocol headers



IP is the narrow waist of the layering hourglass



Placing network functionality

- End-to-end arguments by Saltzer, Reed, and Clark
  - Dumb network and smart end systems
  - Functions that can be *completely* and *correctly* implemented *only* with the knowledge of application end host, should not be pushed into the network
  - Sometimes necessary to break this for performance and policy optimizations
  - Fate sharing: fail together or don't fail at all

5-MINUTE BREAK!

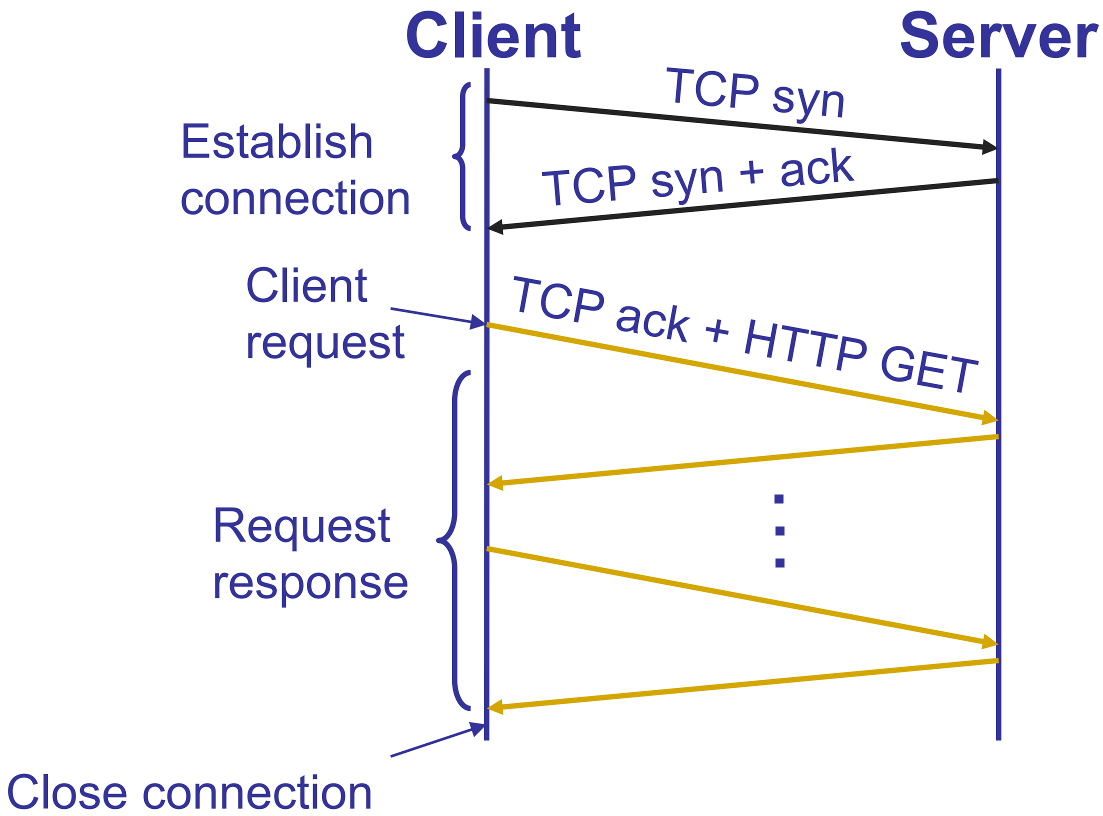
Topics

- Basics (lectures 1–2)
- Application layer (lectures 3–5)
  - HTTP, DNS, and CDN
  - Video Streaming

Hyper Text Transfer Protocol (HTTP)

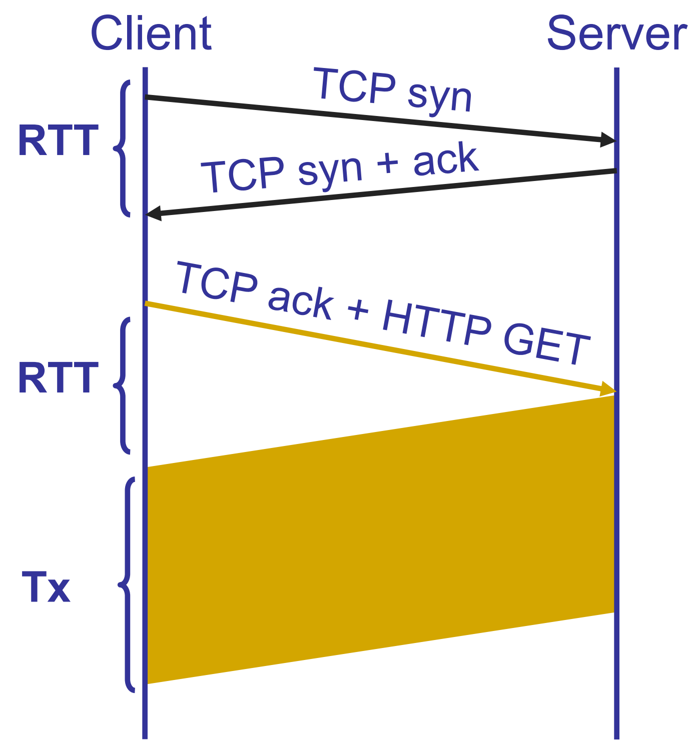
- Client-server architecture
  - Server is “always on” and “well known”
  - Clients initiate contact to server
- Synchronous request/reply protocol
  - Runs over TCP, Port 80
- Stateless
- ASCII format
  - Before HTTP/2

Steps in HTTP request/response



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 (<= B_L), where link bandwidth is referred by B_L

- One-at-a-time: ~ nF/B_C
- m concurrent: ~ $nF/(mB_C)$
  - Assuming each TCP connection gets the same throughput and mB_C <= B_L
- Pipelined and/or persistent: ~ nF/B_C
  - The only thing that helps is higher throughput

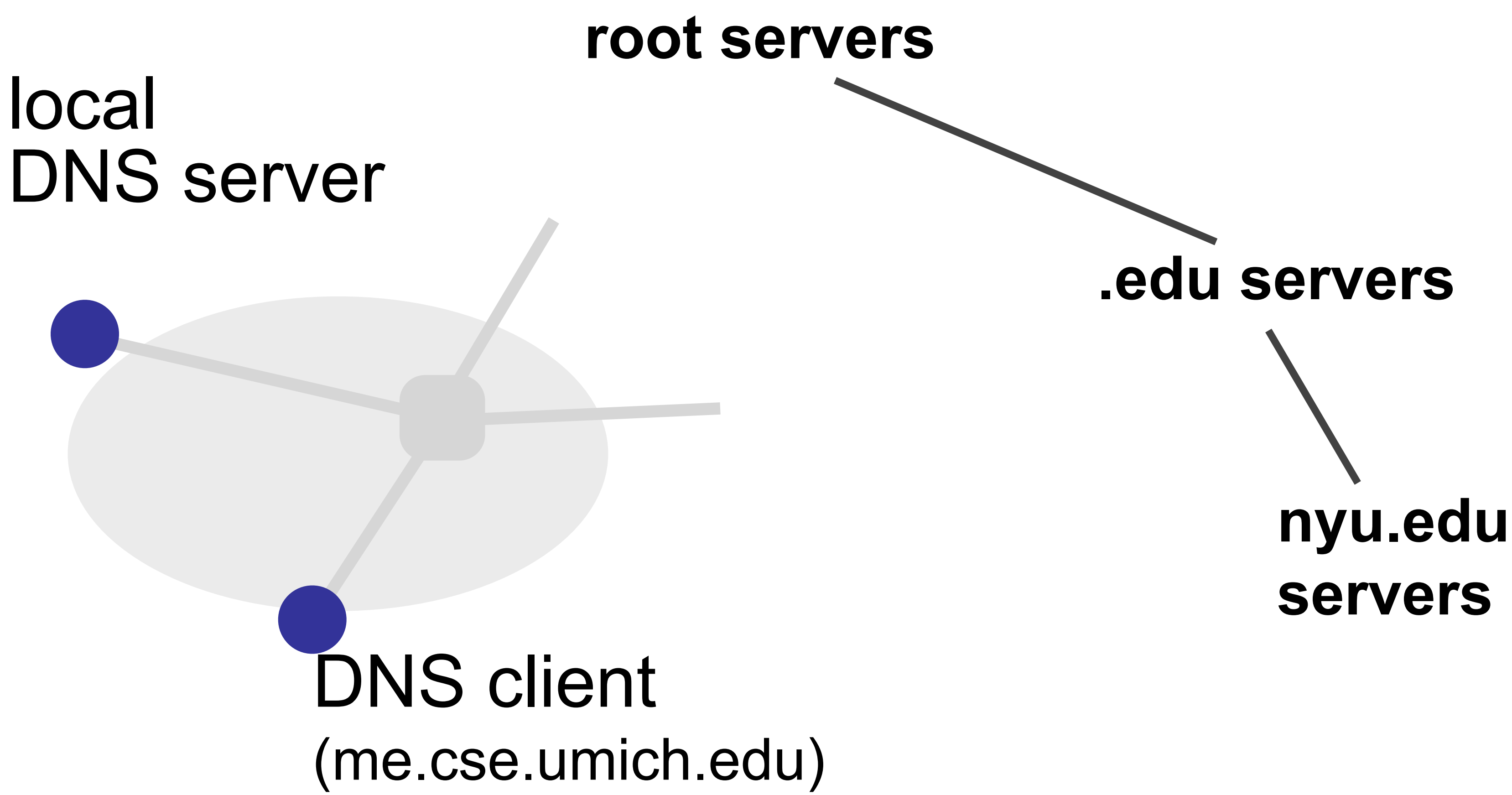
Content Distribution Networks (CDN)

- 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)

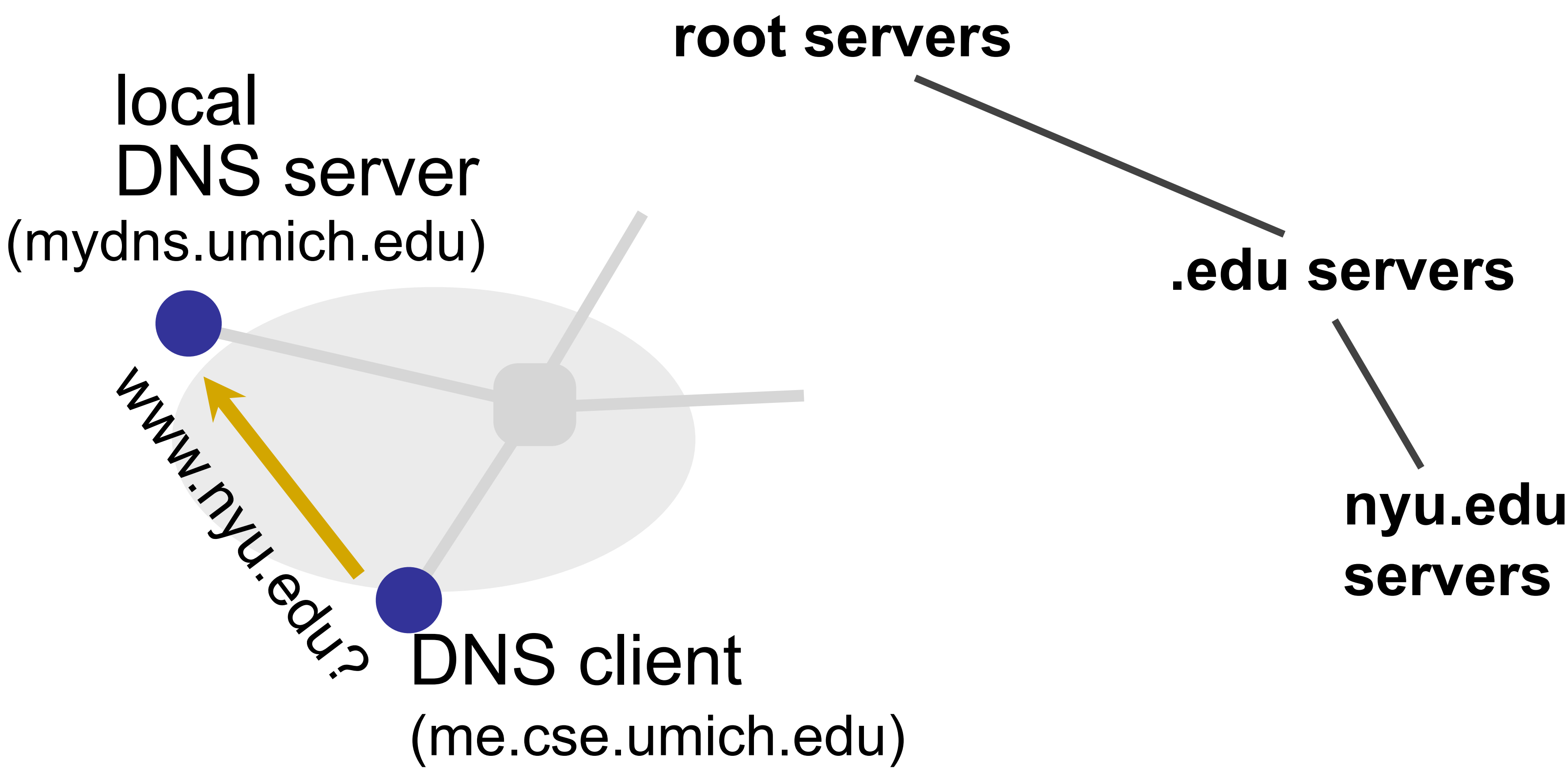
Hierarchies in the DNS

- 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

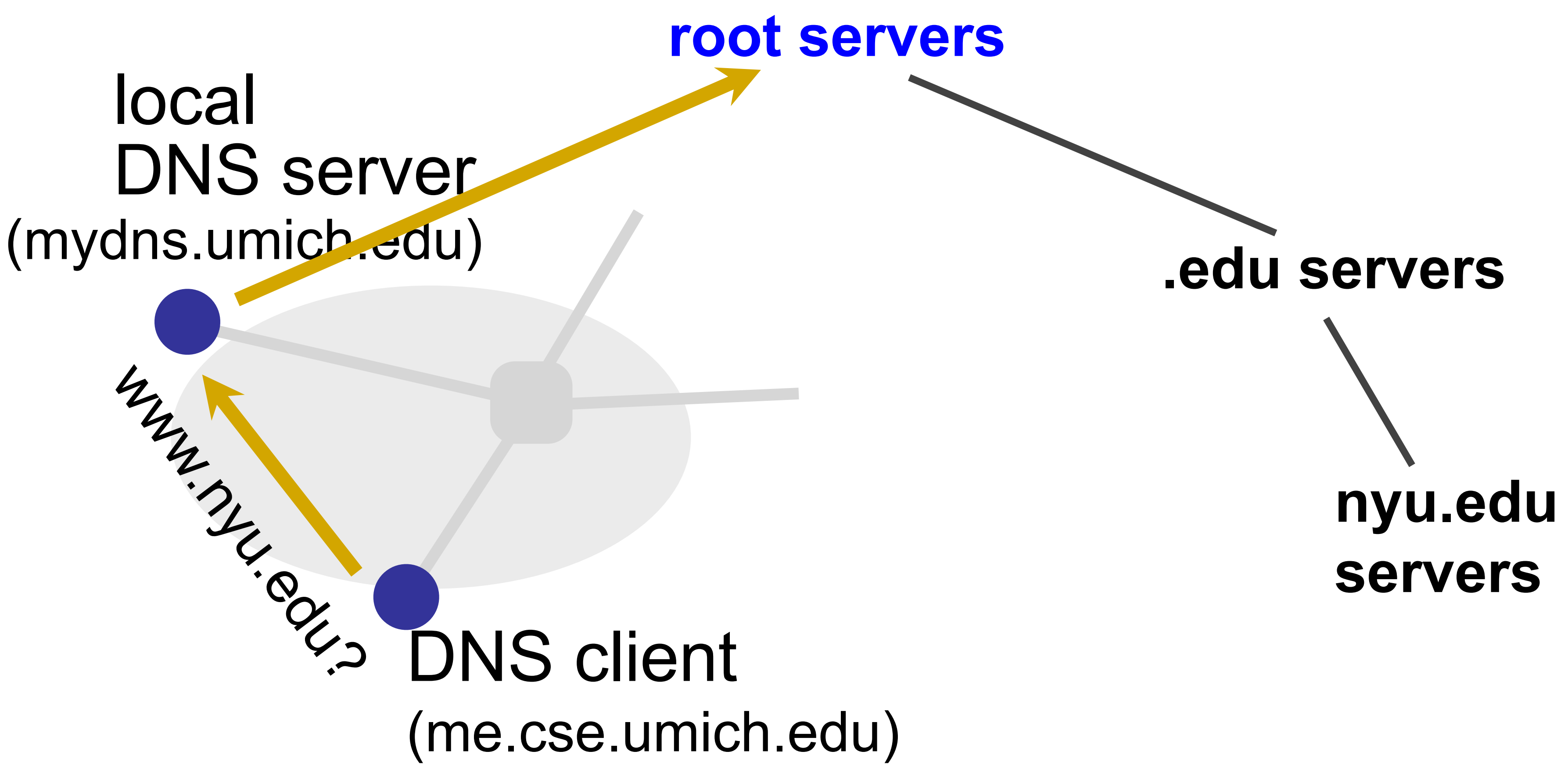
Name resolution



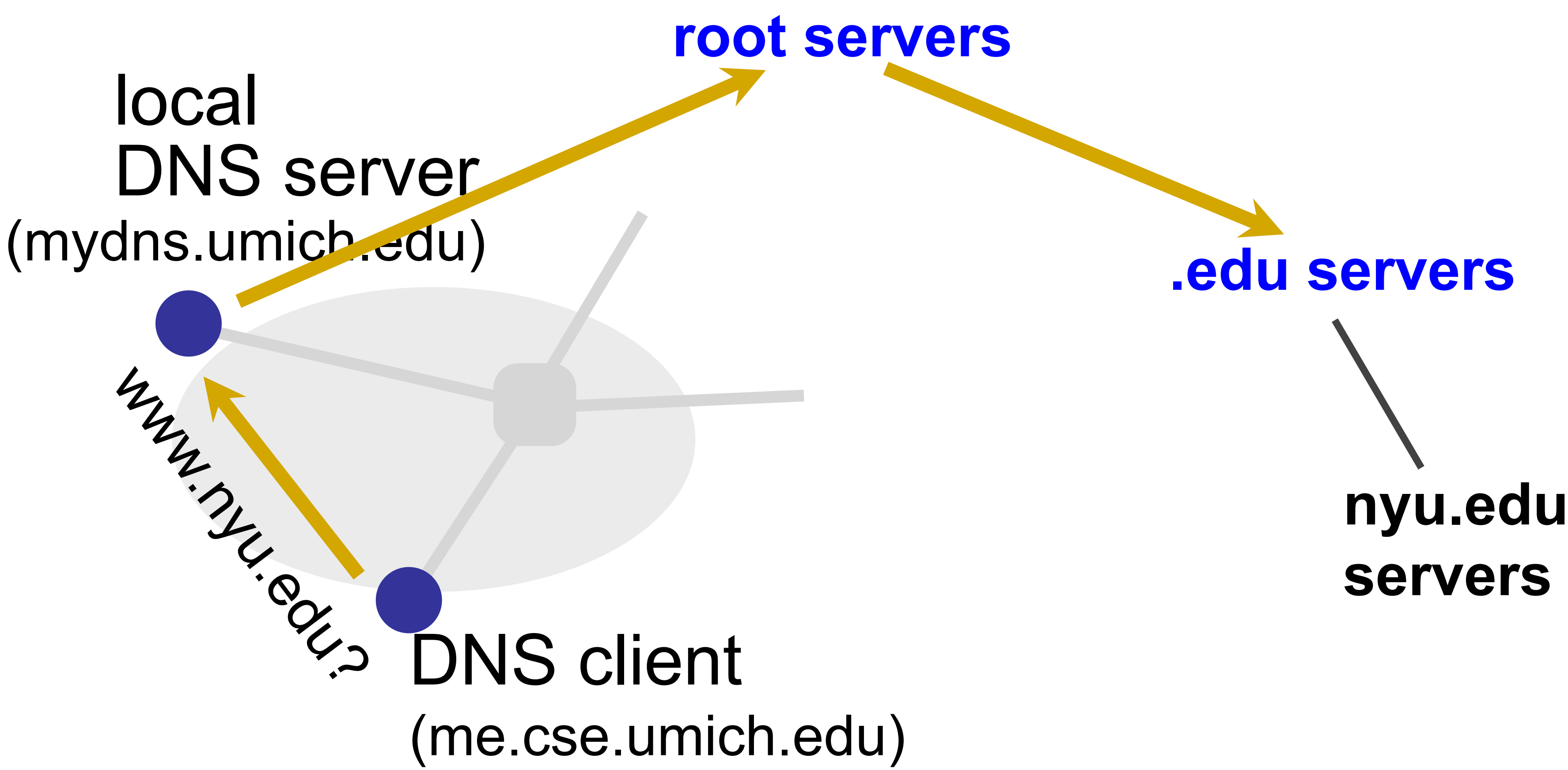
Name resolution



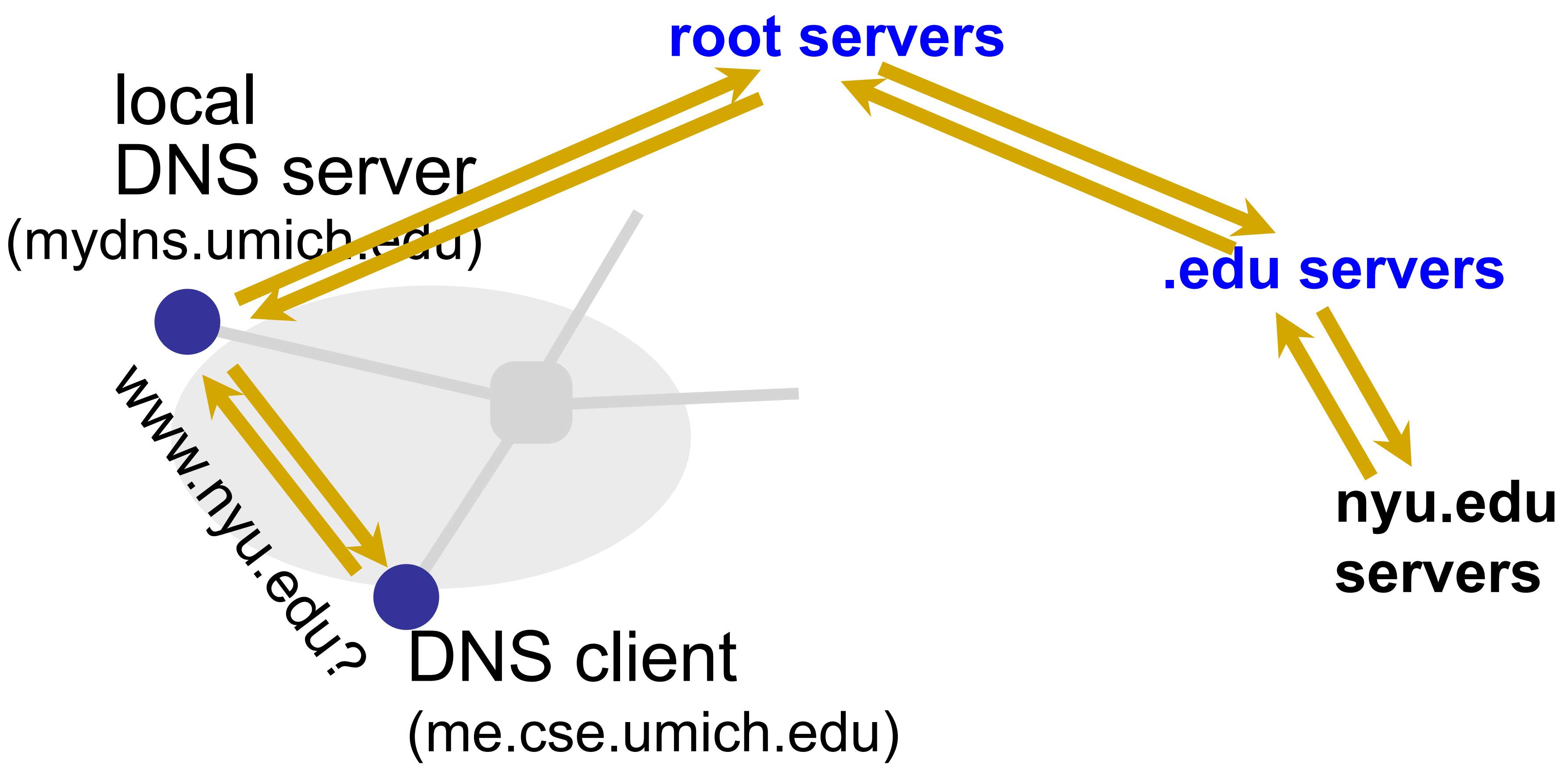
Name resolution



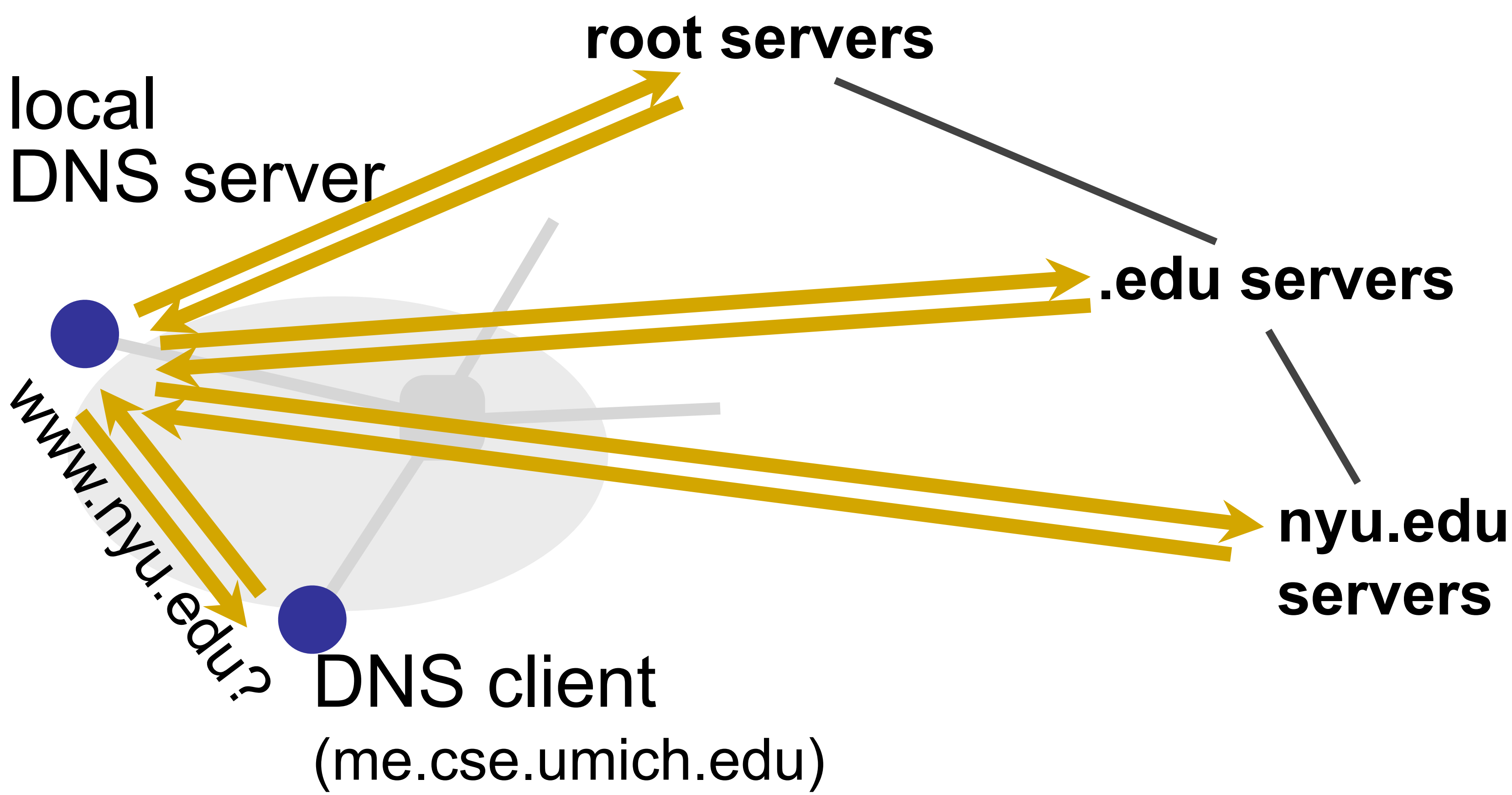
Name resolution



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

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

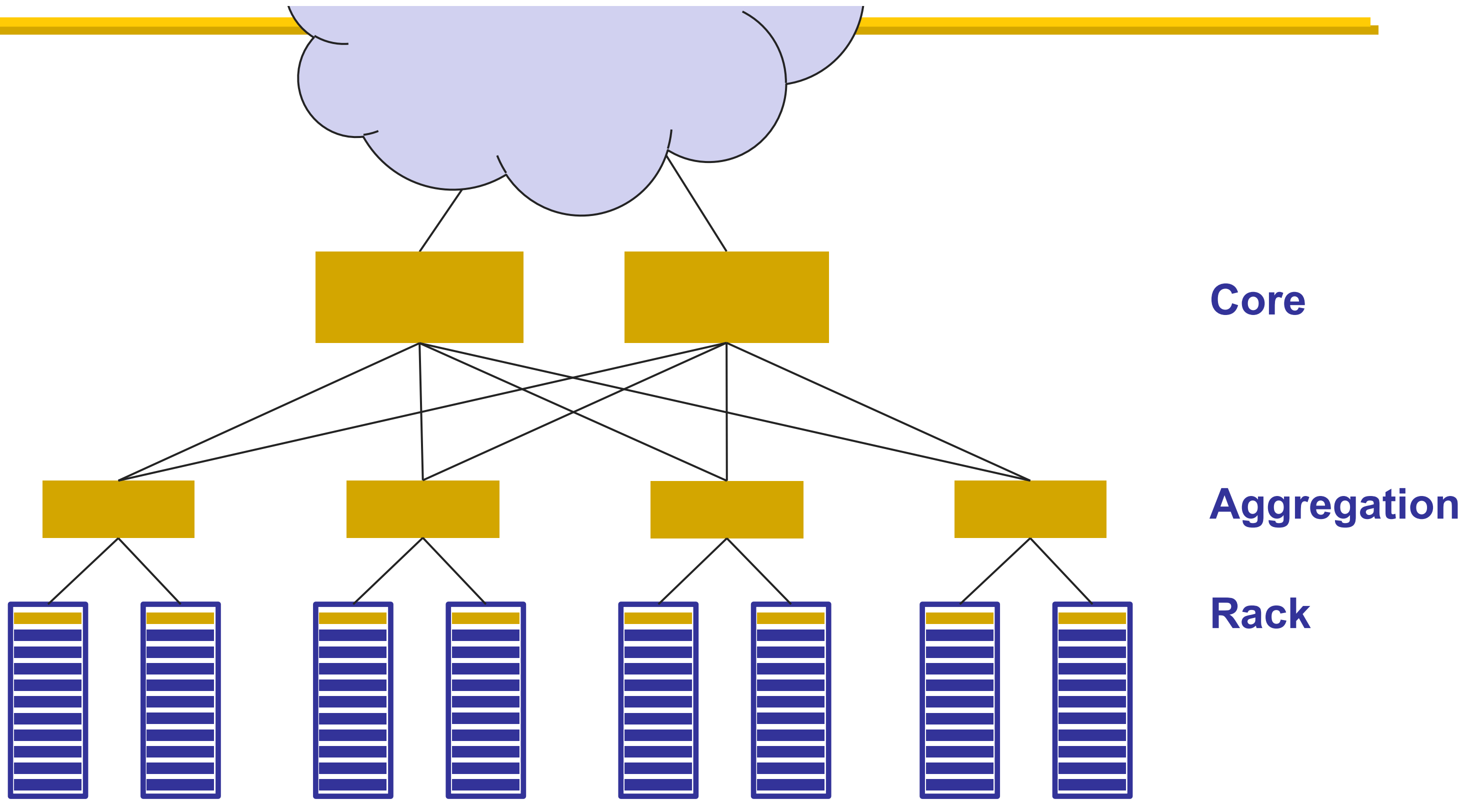
DASH : Dynamic Adaptive Streaming over HTTP

- 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

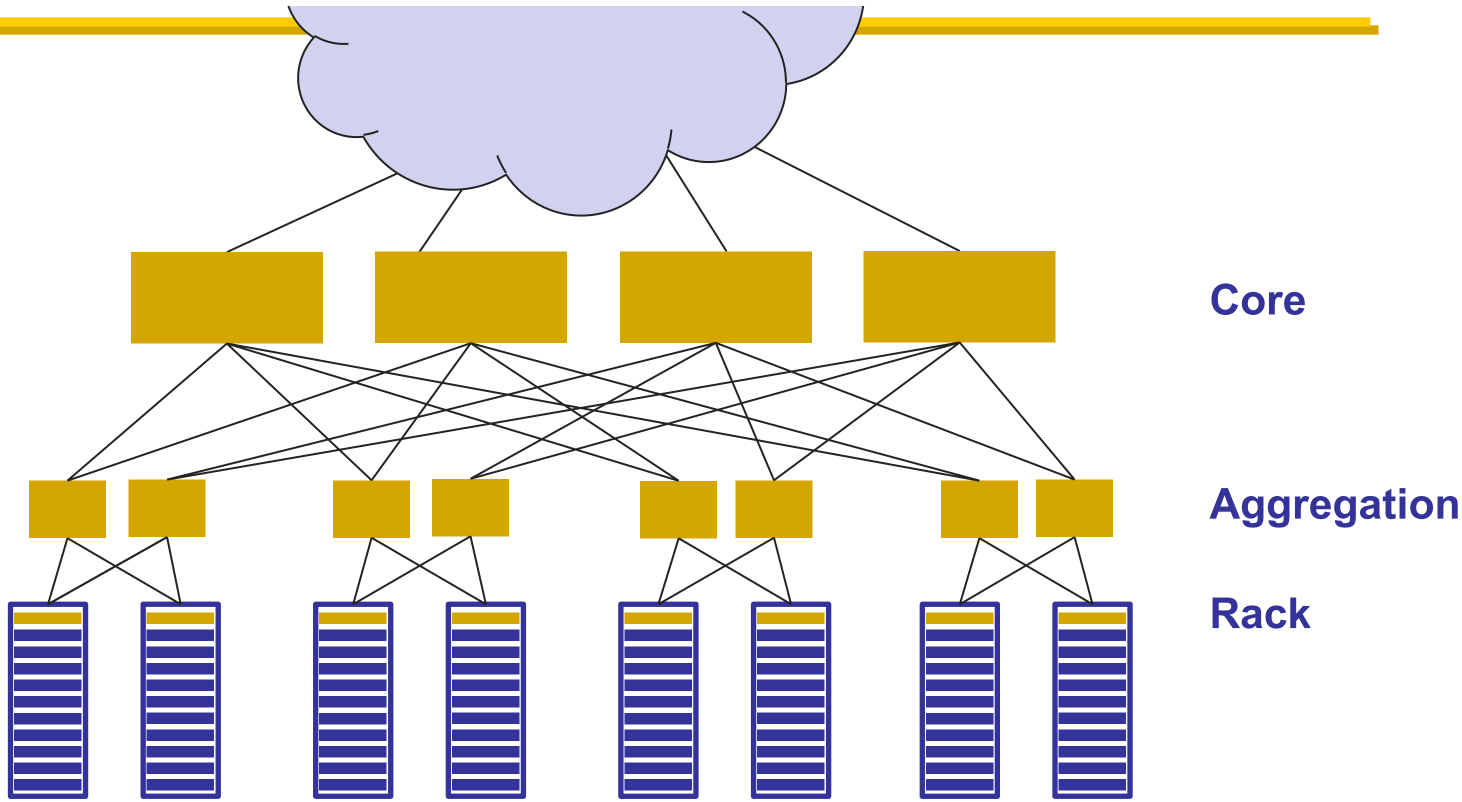
Traditional datacenter networks



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

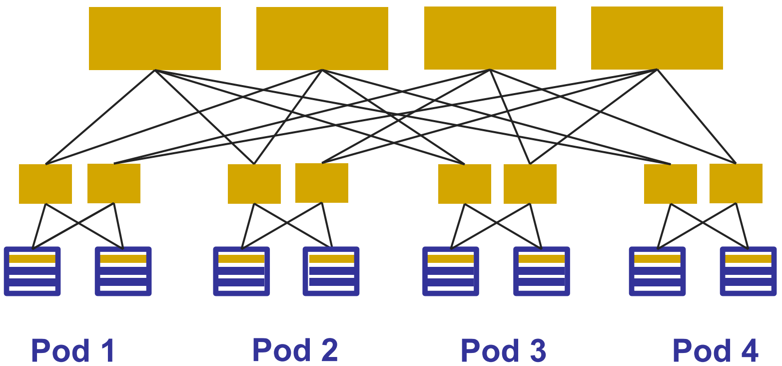
Modern datacenter networks: More bandwidth, more paths



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



Summary

- We will focus on the transport layer from next class