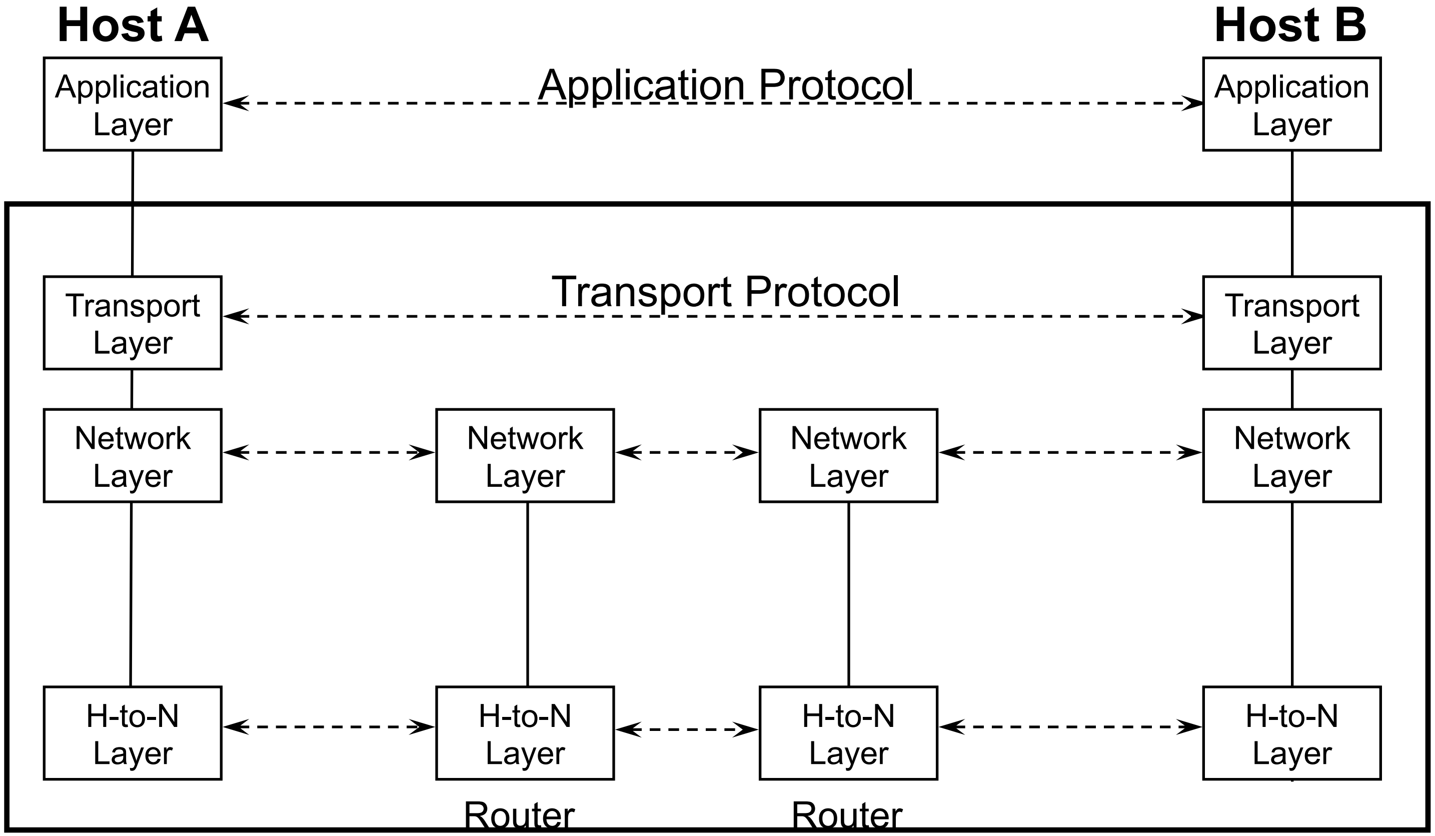
Review for App Layer Protocols

App-Layer Protocols



App-layer protocol defines

- Types of messages exchanged:
  - e.g., request, response
- Message format:
  - Syntax :what fields in messages
  - Semantics: meaning of information in fields
- Rules for when and how processes send & respond to messages

Public-domain protocols:

- defined in RFCs
- DNS, HTTP, FTP, SMTP

Proprietary protocols:

- e.g., Skype, Hangout

Network Application

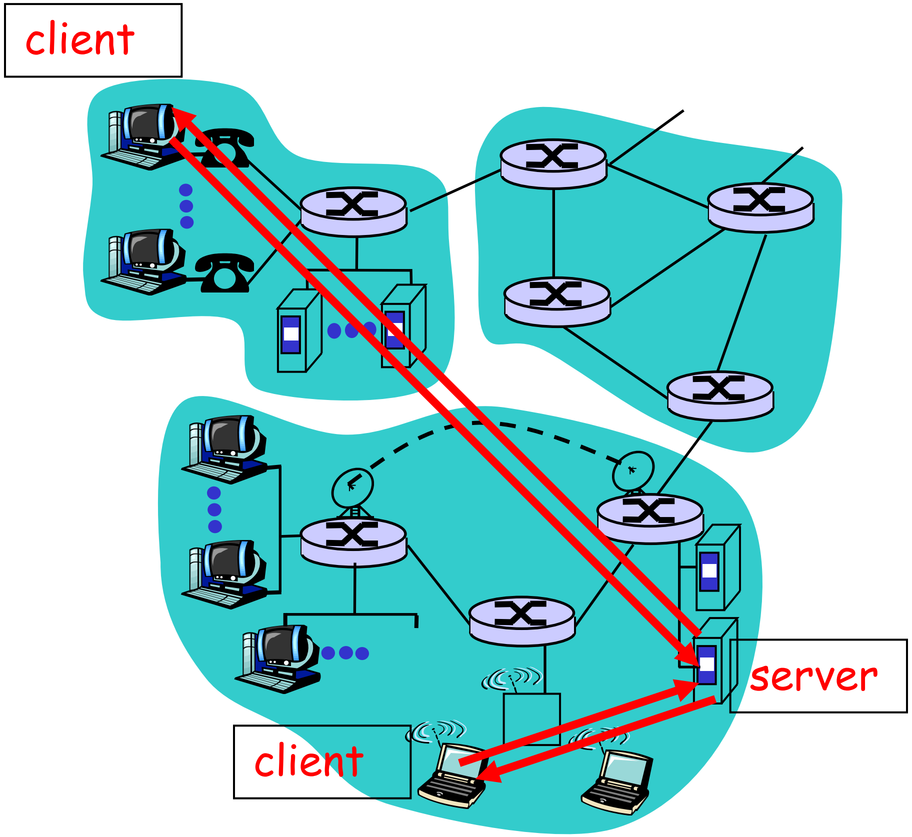
- To communicate, 2 hosts need to identify each other

- Computer network: **IP address**
  - IPv4 (32 bits)
  - IPv6 (128 bits)

- More than one program on a host: **Port #**

- A network connection is a 4-tuple:
  - IP_S, $Port_S$, IP_D, $Port_D$

Client-server architecture (CS)



Server:

- always-on host
- permanent IP address
- server farms for scaling

Clients:

- communicate with server
- may not be always connected
- may have dynamic IP addresses
- do not communicate directly with each other

App-layer protocols

- DNS: Domain Name Service
- HTTP: HyperText Transfer Protocol
- FTP: File Transfer Protocol
- SMTP: Simple Mail Transfer Protocol

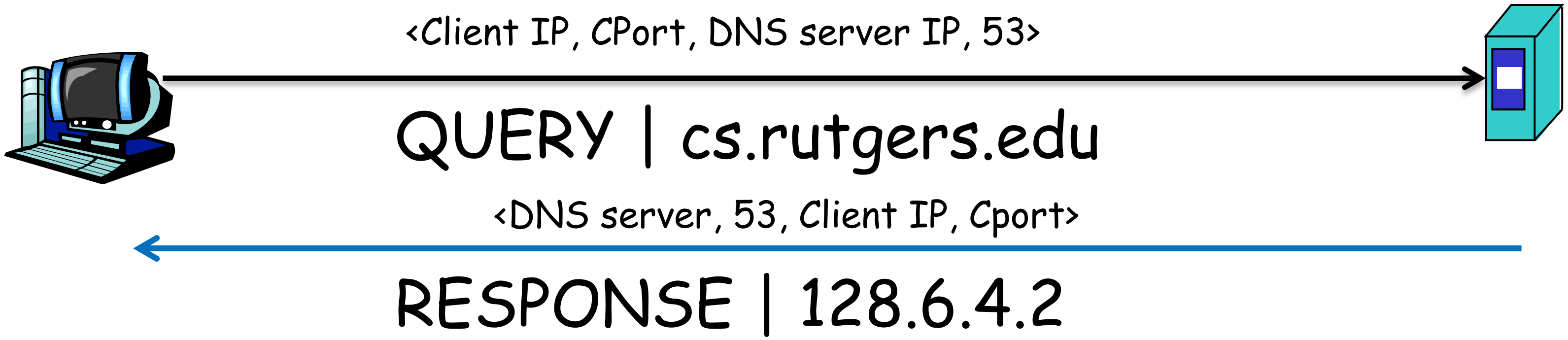
DNS

Domain Name System (DNS)

- For any networked application, we need to know the IP address of a given host name
- Problem:
  - On average, IP addresses have 12 digits
  - We need an easier way to remember IP addresses
- Solution:
  - Use names to refer to hosts
  - Add a service (DNS) to map between host names and IP addresses
  - We call this *Address Resolution*

Simple DNS

DOMAIN NAME	IP ADDRESS
WWW.YAHOO.COM	98.138.253.109
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86



<Client IP, CPort, DNS server IP, 53>

QUERY | cs.rutgers.edu

<DNS server, 53, Client IP, Cport>

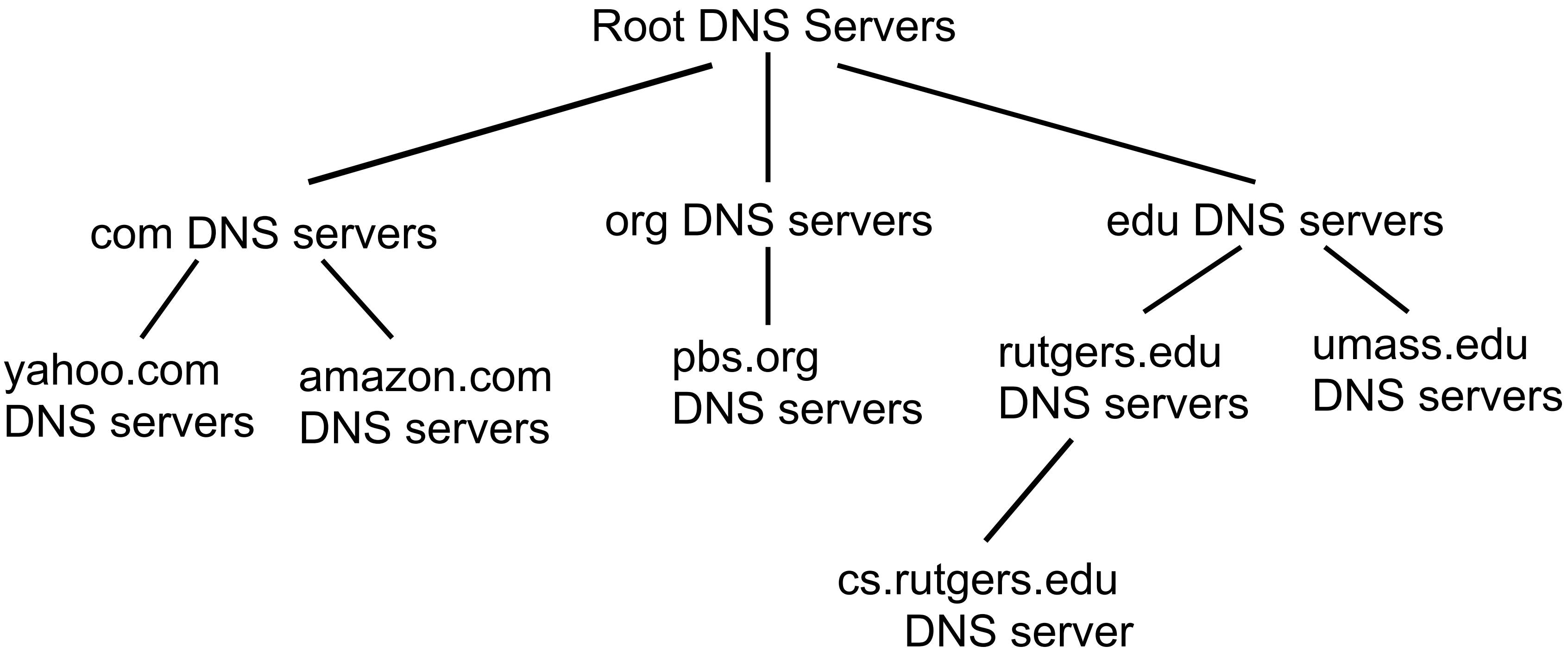
RESPONSE | 128.6.4.2

Centralize DNS?

- □ single point of failure
- □ traffic volume
- □ Distant centralized database
- □ maintenance

doesn't *scale!*

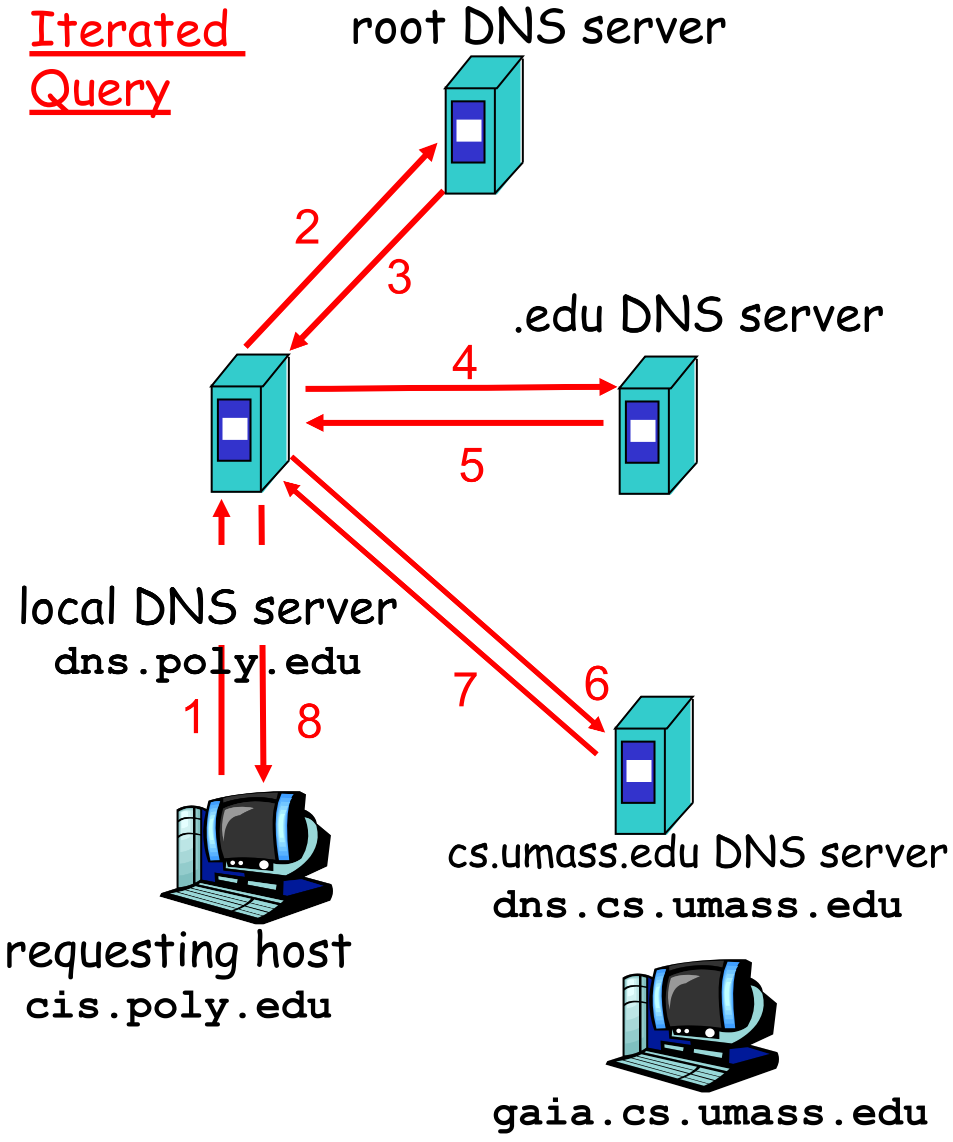
Distributed, Tree-based Database



RFC 1034

2 DNS Query Types

Iterated Query



root DNS server

.edu DNS server

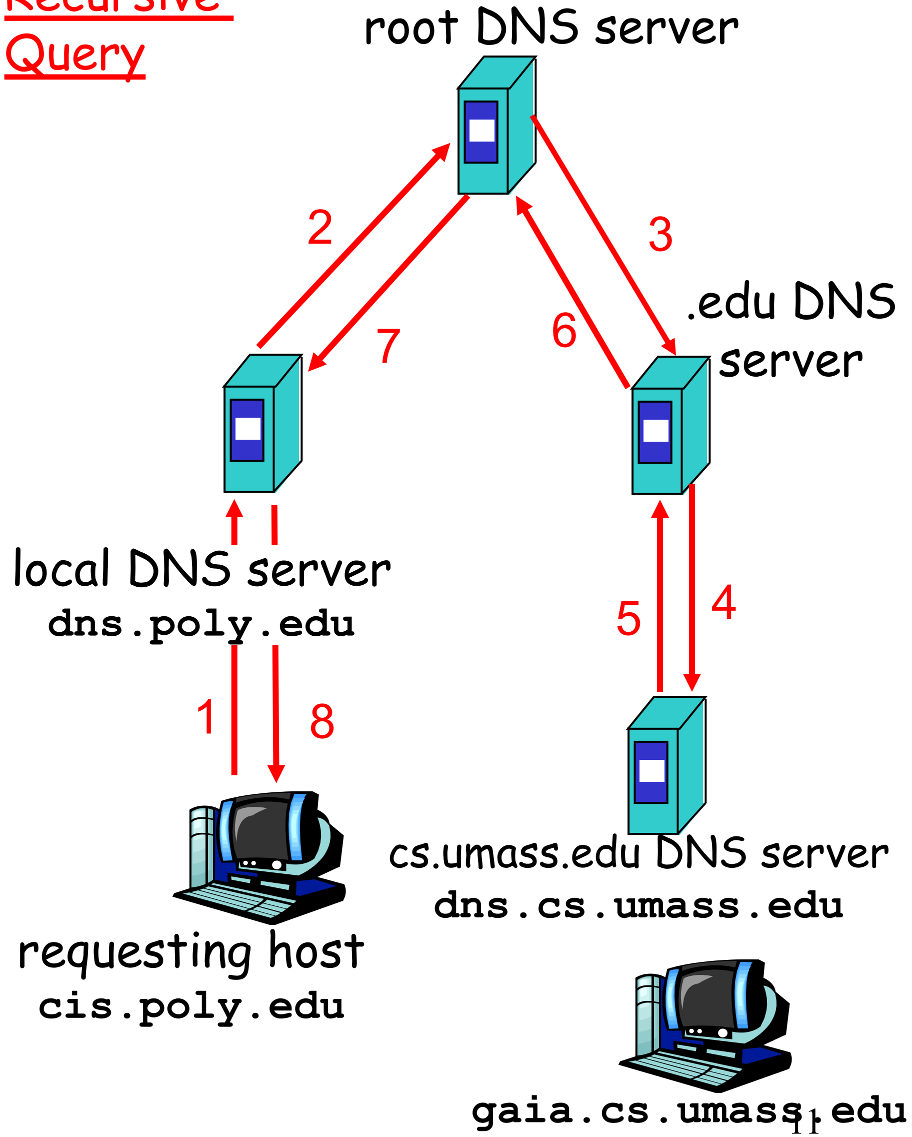
local DNS server
dns.poly.edu

requesting host
cis.poly.edu

cs.umass.edu DNS server
dns.cs.umass.edu

gaia.cs.umass.edu

Recursive Query



root DNS server

.edu DNS server

local DNS server
dns.poly.edu

requesting host
cis.poly.edu

cs.umass.edu DNS server
dns.cs.umass.edu

gaia.cs.umass.edu

DNS: Caching, Updating, Bootstrapping

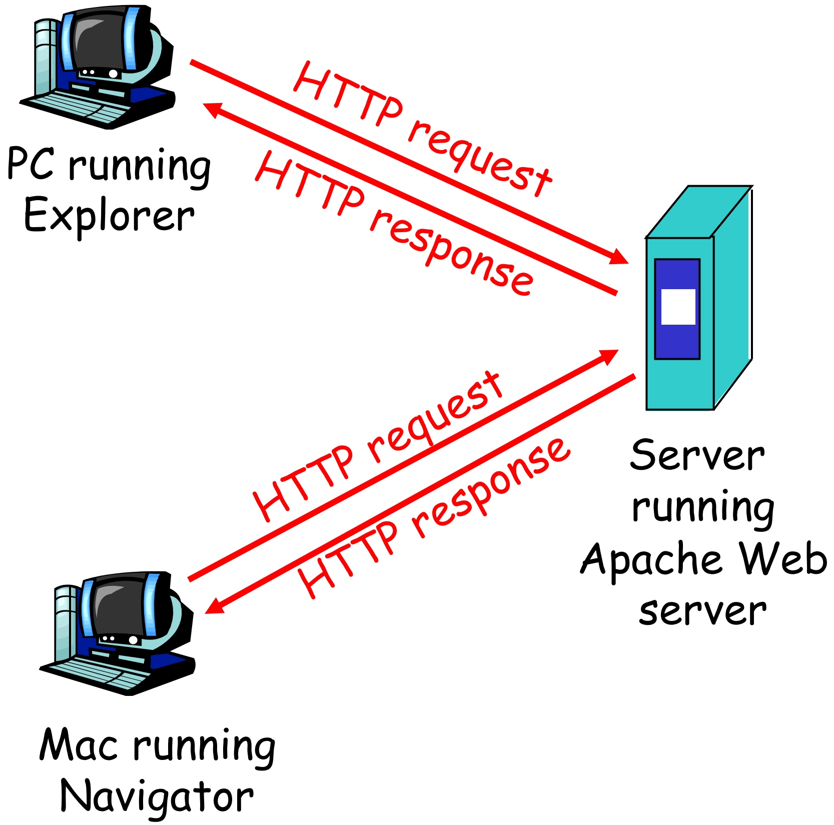
- Once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time
  - TLD (Top Level Domain) servers typically cached in local name servers
    - Thus root name servers not often visited

- How does a host contact the name server if all it has is the name and no IP address?
  - IP address of at least 1 nameserver must be given in advance or with another protocol (DHCP, bootp)

HTTP

HTTP overview

- Web page consists of a base HTML-file which includes several referenced objects addressable by a URL
- Client/Server model
  - *client:* browser that requests, receives, "displays" Web objects
  - *server:* Web server sends objects in response to requests



- Request Message
- Response Message

HTTP connections

Nonpersistent HTTP

- At most one object is sent over a single TCP connection.

Persistent HTTP

- Multiple objects can be sent over a single TCP connection between client and server.

Nonpersistent HTTP issues:

- requires **2 RTTs per object**
  - TCP Connection and HTTP Request
- Browsers can open parallel TCP connections to fetch referenced objects

Persistent HTTP

- server leaves TCP connection open after sending response
- subsequent HTTP messages sent over open connection

Cookie : User-server State

HTTP is "stateless"

- server maintains no information about past client requests
- What state can bring:
  - Authorization, shopping carts, recommendations, user session state

Four components:

1) cookie header line of HTTP *response* message
2) cookie header line in HTTP *request* message
3) cookie file kept on user's host, managed by user's browser
4) back-end database at Web site

Web caches (proxy server)

- Reduce response time for client request.
- Reduce traffic on an institution’s access link.

- browser sends all HTTP requests to cache
  - Miss: cache requests object from origin server, then returns object to client
  - Hit: cache returns object

- guarantees cache content is up-to-date

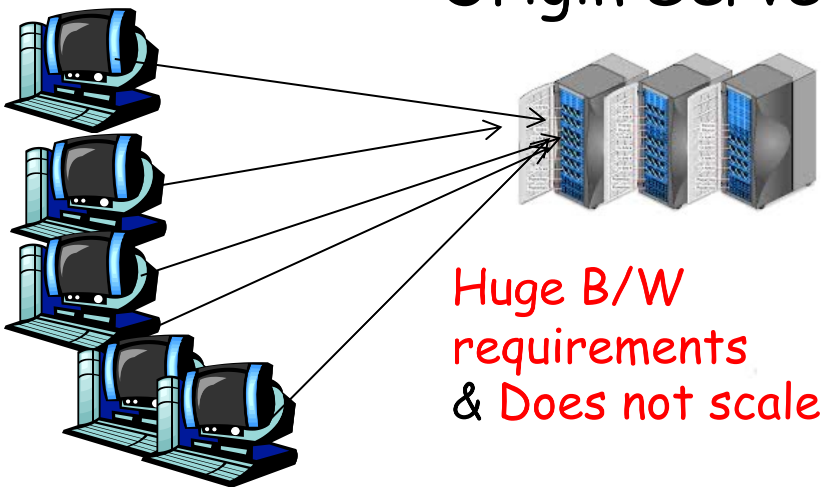
- saves traffic and response time whenever possible

Content Distribution Networks (CDN)

- ❒ Reduce bandwidth Requirement & Traffic of content provider
- ❒ Reduce $$ of maintaining Servers
- ❒ Improve response time to user

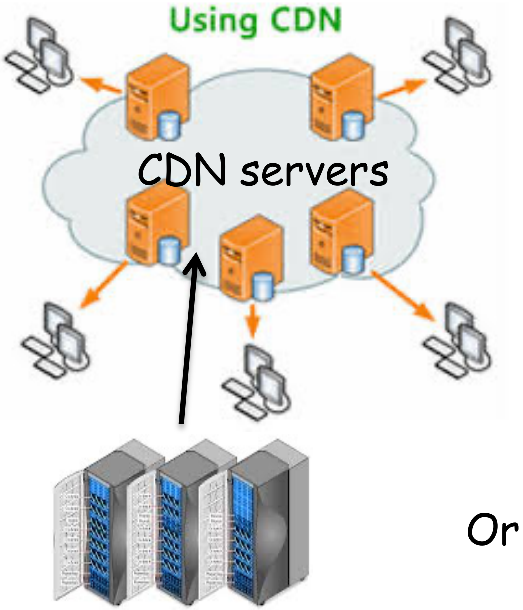
Clients

Origin Server



Huge B/W requirements & Does not scale

Without CDN

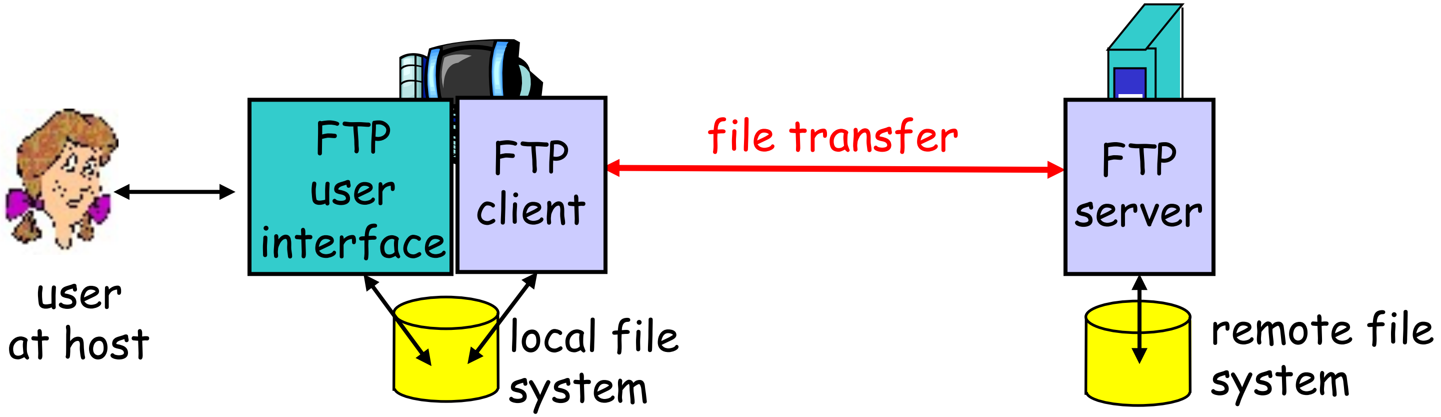


Origin server

With CDN

FTP

FTP: the file transfer protocol



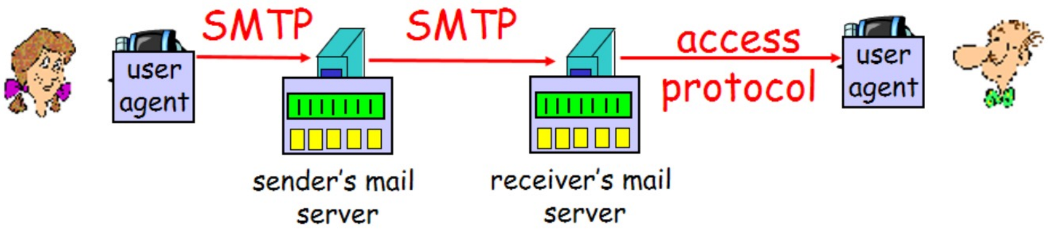
- transfer file to/from remote host
- client/server model
  - *client:* side that initiates transfer (either to/from remote)
  - *server:* remote host
- "out of band" control
  - Control connection port 21 & Data connection port 20
- **Active connection**: data connection initiated form server
- **Passive connection**: : data connection initiated form client
- Key Drawback: Sends passwords in plain ASCII text
- Replaced with sftp instead

SMTP

Electronic Mail

Three Components:

1. User Agents



2. Mail Servers
   - mailbox contains incoming messages for user
   - message queue of outgoing (to be sent) mail messages

- Mail access protocol: retrieval from server
  - POP: Post Office Protocol IMAP: Internet Mail Access Protocol
  - HTTP: Hotmail , Yahoo! Mail, etc.

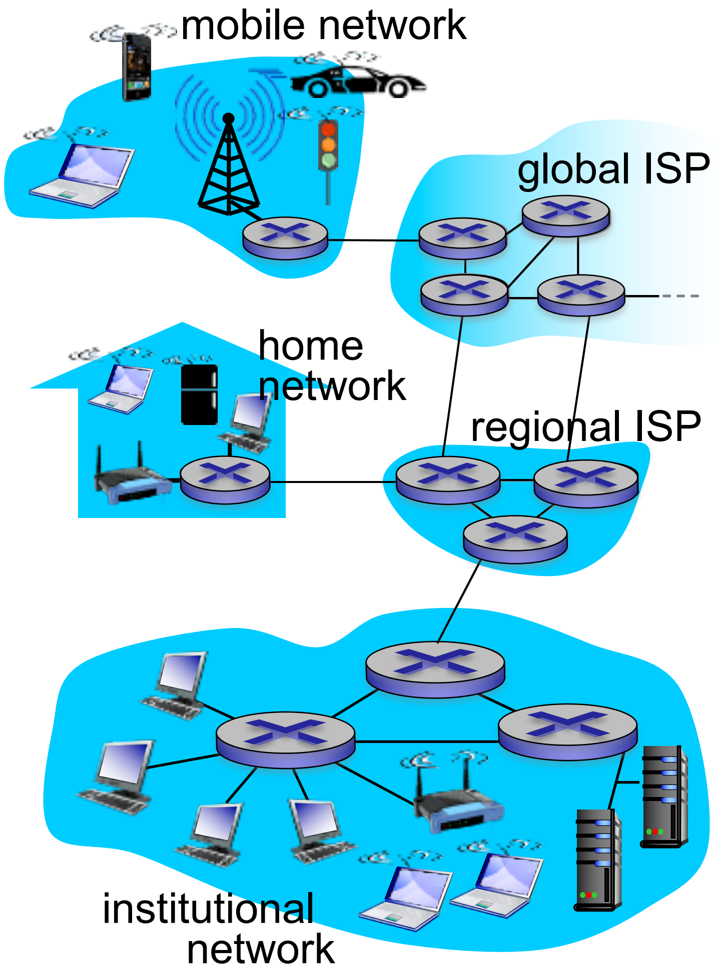
3. SMTP protocol
   - Used to send messages
   - Client: sending user agent or sending mail server
   - server: receiving mail server

Review for Internet Introduction

What's the Internet: Two Views

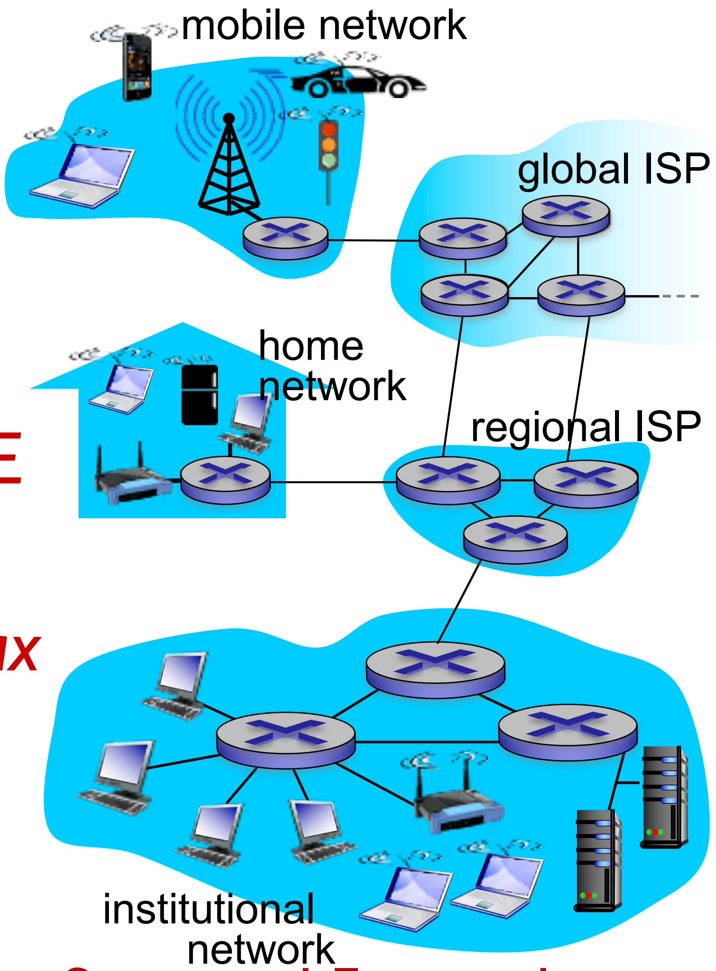
- *View 1: "Component" View*
  - billions of connected *hosts*
  - *routers* and *switches*
  - *protocols* control sending, receiving of messages
  - "network of networks"

- *View 2: Service View*
  - *Infrastructure that provides services to applications:*
  - Web, VoIP, email, games, e-commerce, social nets, …



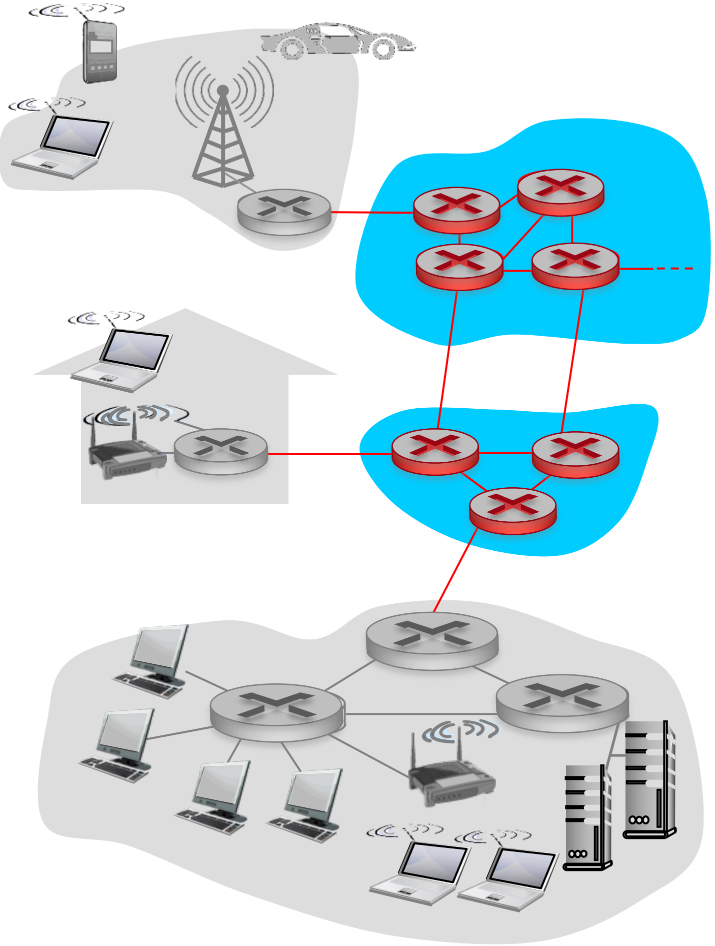
Internet Components

- *Network Edge:*
  - hosts: clients and servers
- *Access networks*
  - *Home: DSL & Cable*
  - *Institutional: Ethernet*
  - *Wireless: WiFi & 3G & 4G LTE*
- *Physical Media*
  - *guided media: copper, fiber, coax*
  - *unguided media: radio*
- *Network Core:*
  - Interconnected Routers
  - Packet Switching: Shared Resources: Store and Forward
  - Circuit Switching: Non-Shared Resources: Reserved Circuit

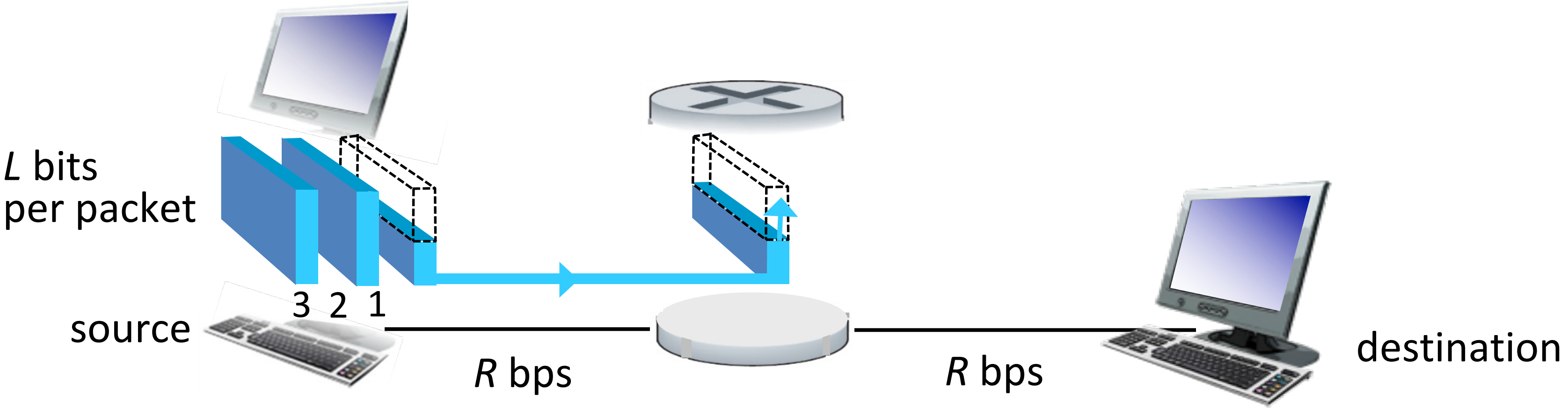


The network core

- ❒ mesh of interconnected routers
  - Packet Switching
    - *store and forward*
  - Circuit Switching
    - *Reserved Resources*



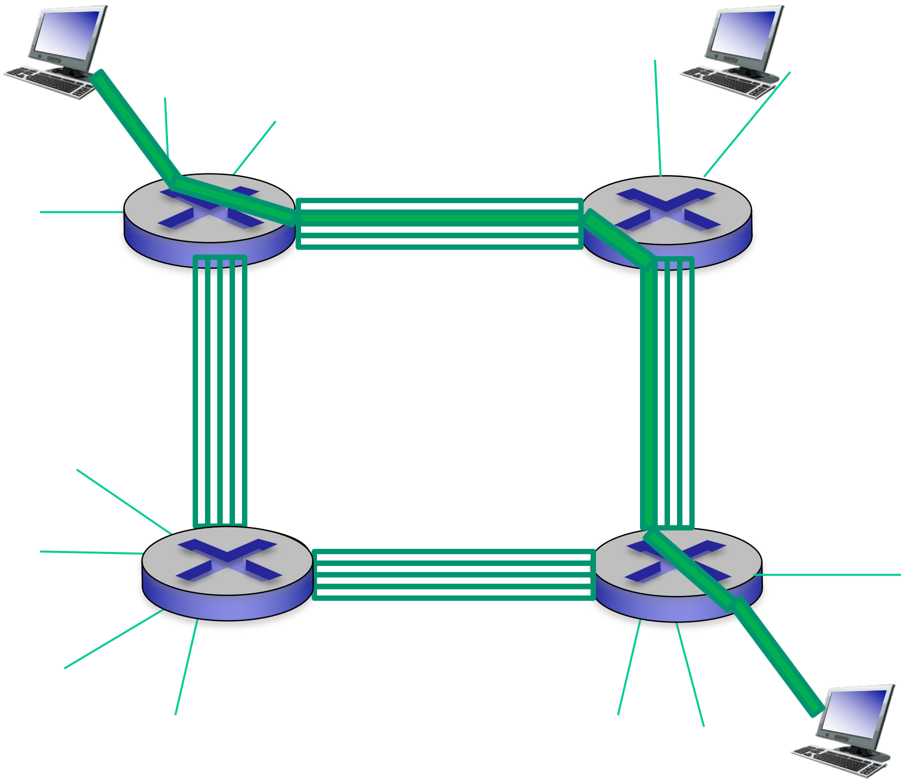
Core 1: Packet-switching



- *store and forward:* entire packet must arrive at router before it can be transmitted on next link

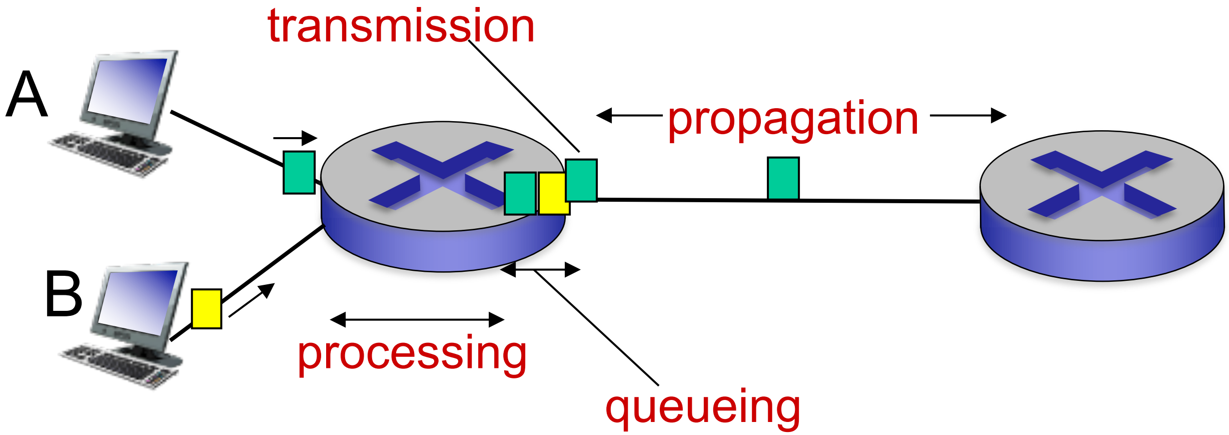
Core 2: Circuit Switching

end-end resources reserved for "call" between source & dest:



- circuit segment idle if not used by call *(no sharing)*
- commonly used in traditional telephone networks

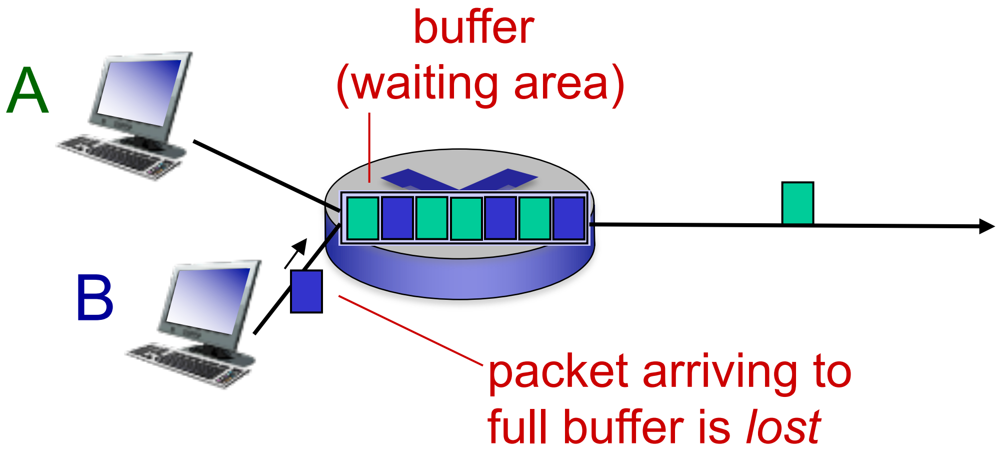
Network Metrics 1: Delay



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

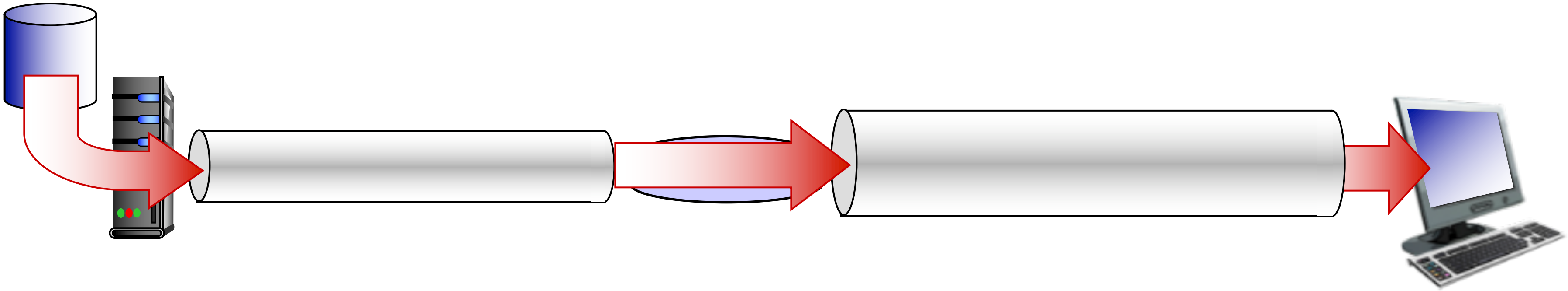
Network Metrics 2: Packet loss

- Queue (aka buffer) has limited capacity
- Packet is dropped if arriving to full queue (aka lost)



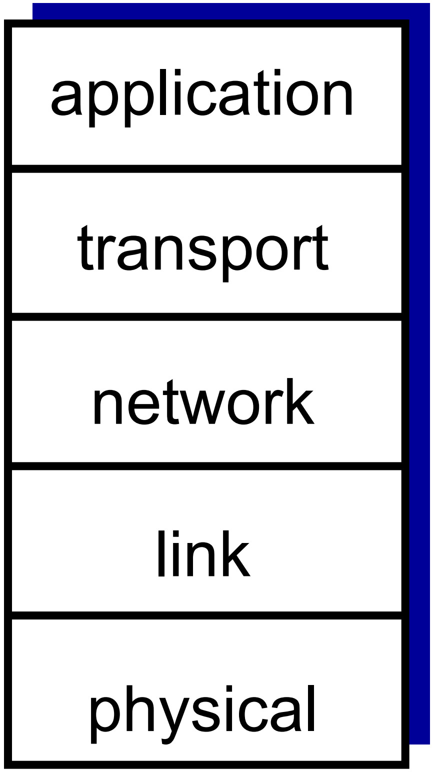
Network Metrics 3: Throughput

- *throughput:* rate (bits/time unit) at which bits transferred between sender/receiver
  - *Real-time:* rate at a given time point
  - *Average:* rate over longer period of time



Internet protocol stack

- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network devices
  - Ethernet, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"



application

transport

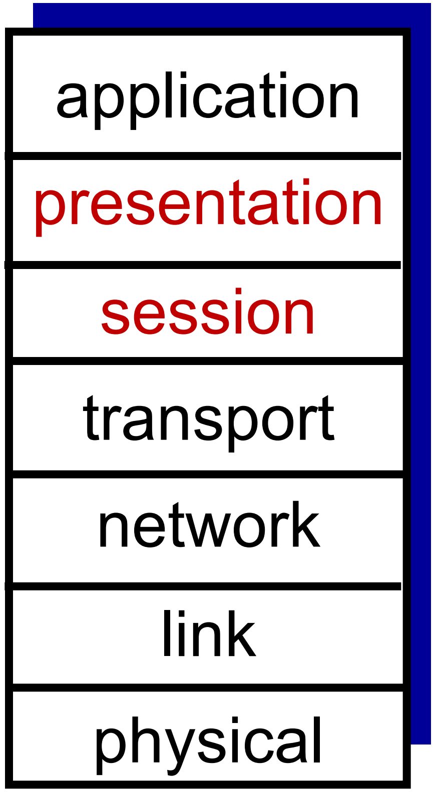
network

link

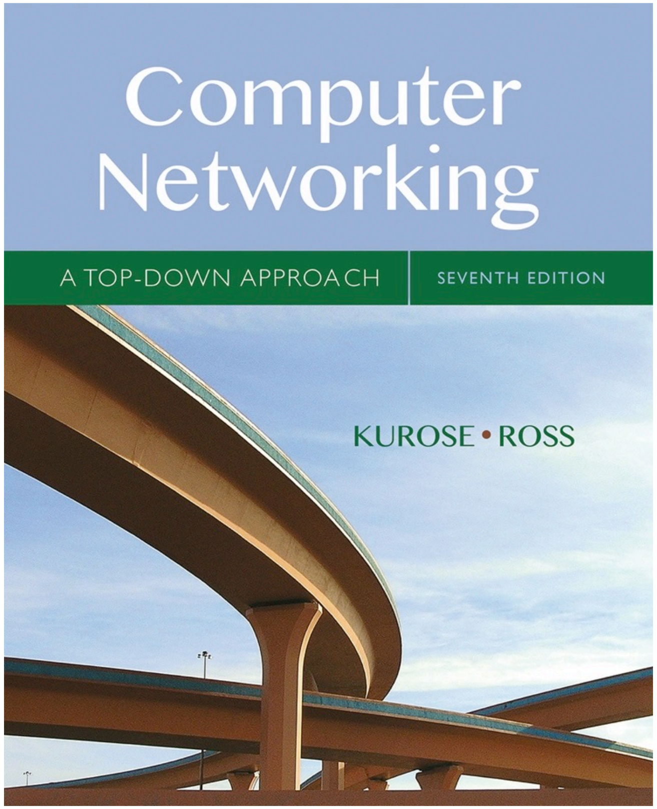
physical

ISO/OSI reference model

- *Open Systems Interconnections*
  - *Presentation:* allow applications to interpret meaning of data, e.g., encryption, compression,
  - *Session:* synchronization, recovery of data exchange
- 5-layer Internet stack "missing" these 2 layers!
  - these services, *if needed,* must be implemented in application



Chapter 3
Transport Layer



Computer Networking: A Top Down Approach

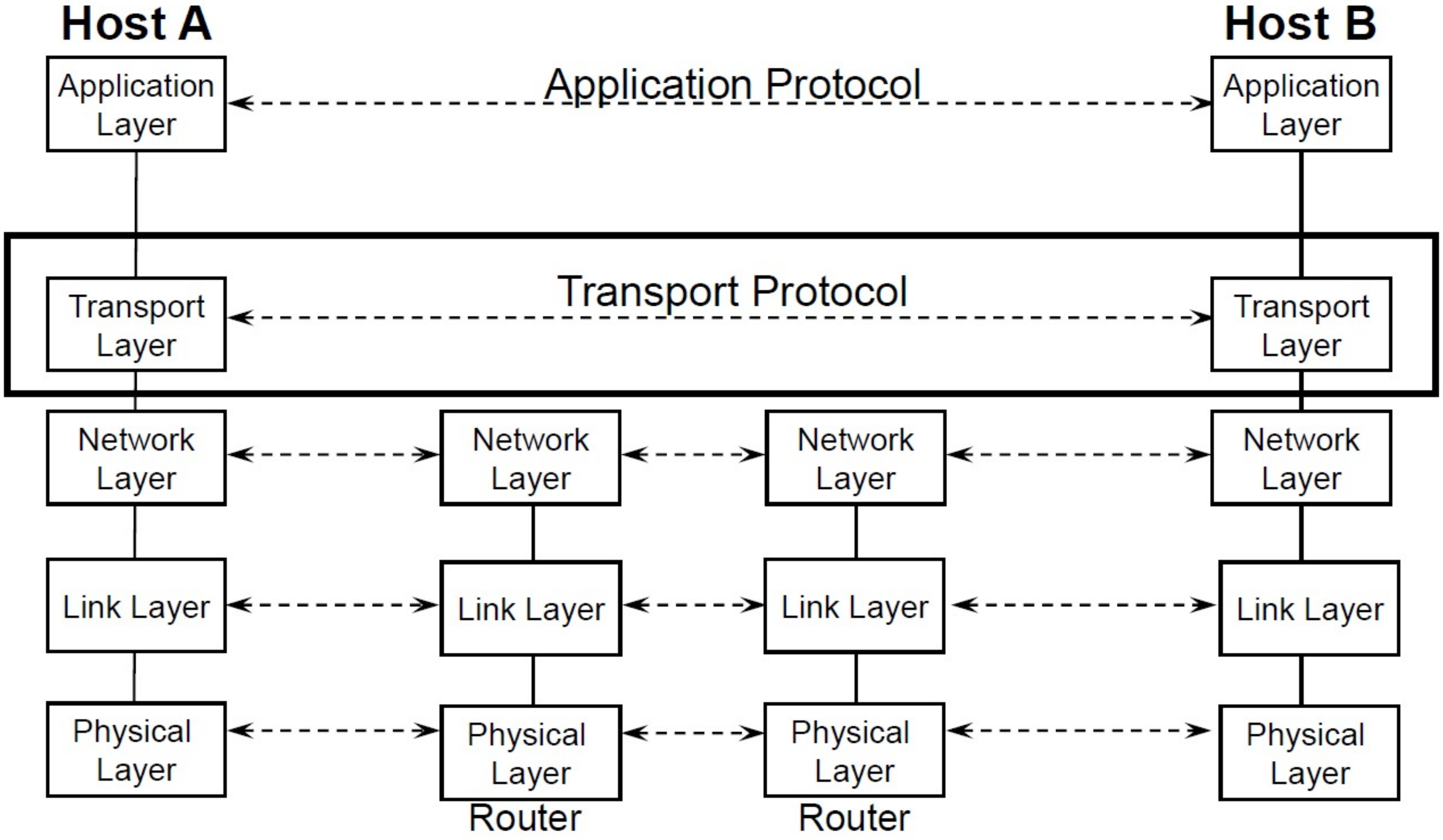
7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Chapter 3: Transport Layer

our goals:

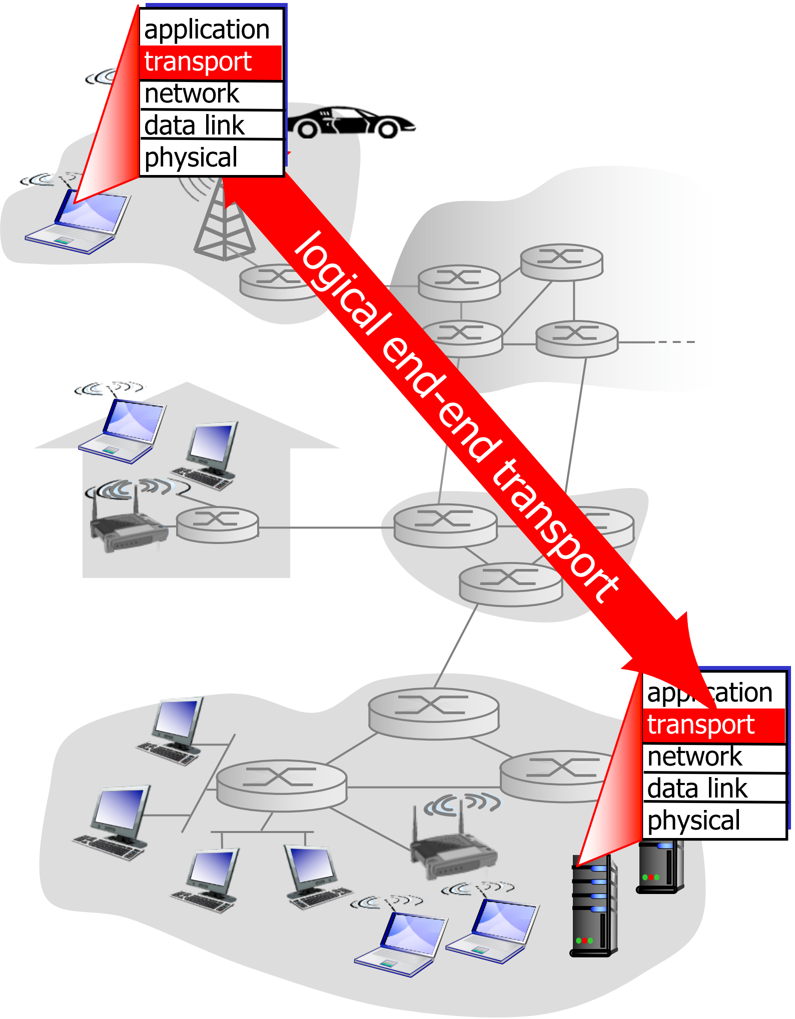
- understand principles behind transport layer services:
  - multiplexing, demultiplexing
  - flow control
  - congestion control
- learn about Internet transport layer protocols:
  - UDP: connectionless transport
  - TCP: connection-oriented reliable transport

Internet Protocol Stack



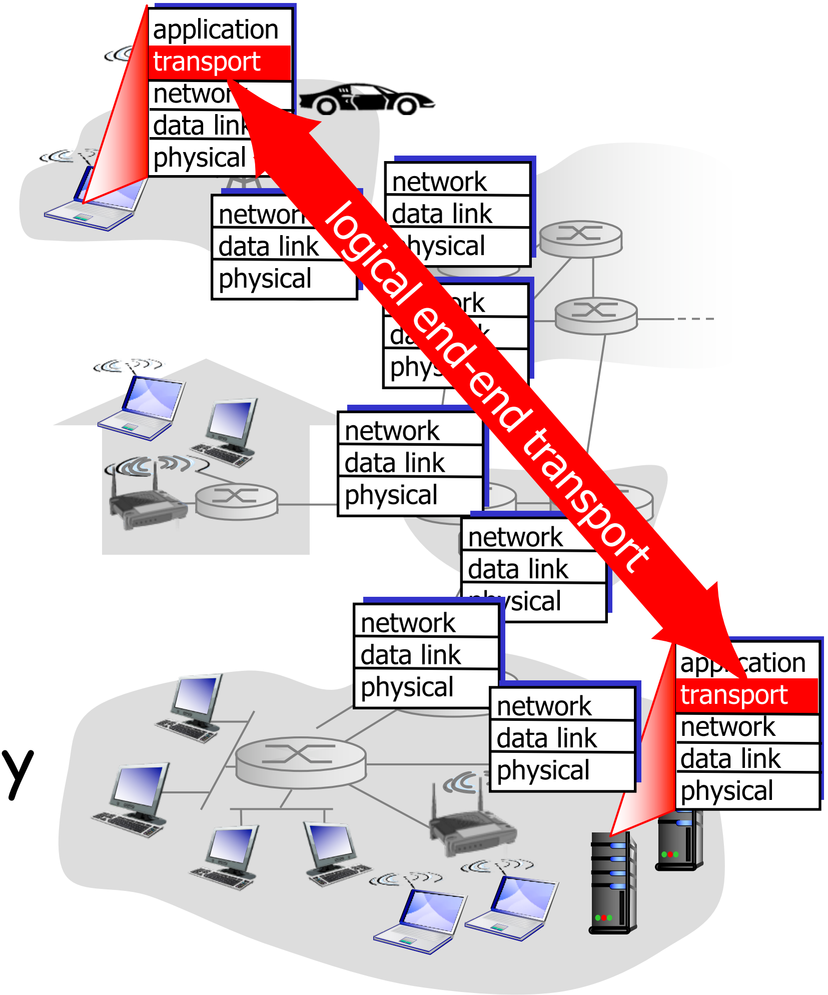
Transport services and protocols

- provide *logical communication* between **app processes** running on **different hosts**
- transport protocols run in end systems
  - **send side:** breaks app messages into *segments*, passes to network layer
  - **rcv side:** reassembles segments into messages, passes to app layer



Internet transport-layer protocols

- Transmission Control Protocol (TCP)
  - reliable, in-order delivery
  - congestion control
  - flow control
  - connection setup



- User Datagram Protocol: UDP
  - unreliable, unordered delivery
  - Simple extension of “best-effort” IP

- Services not available: