Midterm Review

Midterm Exam

- Time: see syllabus in Canvas

- Location: Canvas + Honorlock
  - https://fiuhelp.force.com/canvas/s/article/Honorlock-students

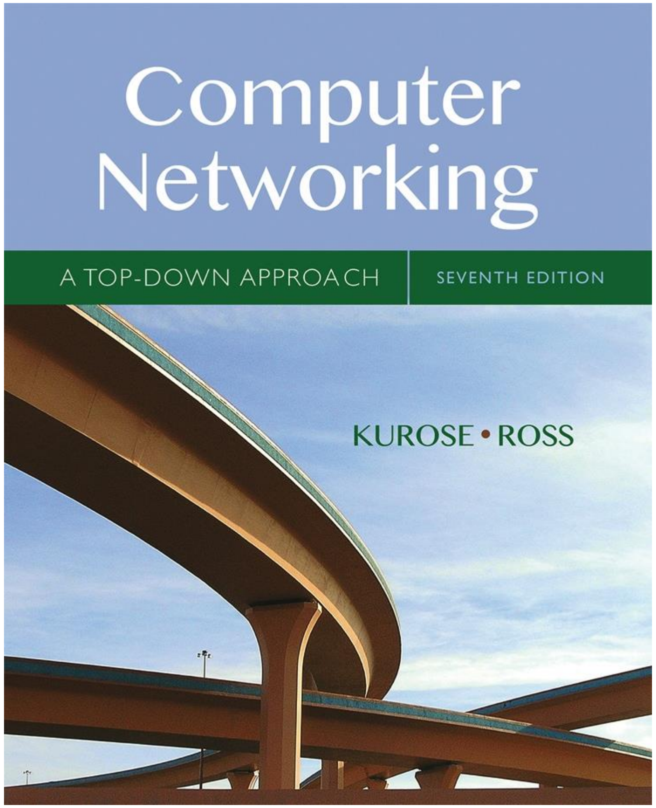
- Scope
  - Chapter 1 – Introduction
  - Chapter 2 – Application layer
  - Chapter 3 – Transport layer

Midterm Exam

- Question format
  - 20 questions (4 points each), similar to quiz questions
  - 4 problems (5 points each), similar to homework problems

- How to prepare
  - Review slides
  - Quiz questions
  - Homework problems

Chapter 1
Introduction



Computer Networking: A Top Down Approach

Slides adopted from original ones provided by the textbook authors.

Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

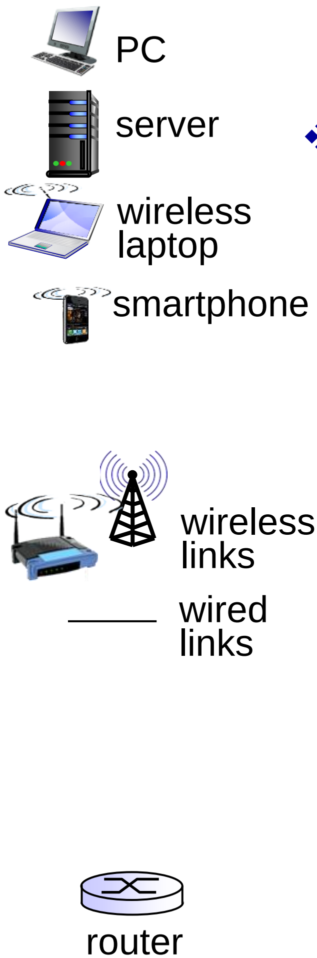
1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

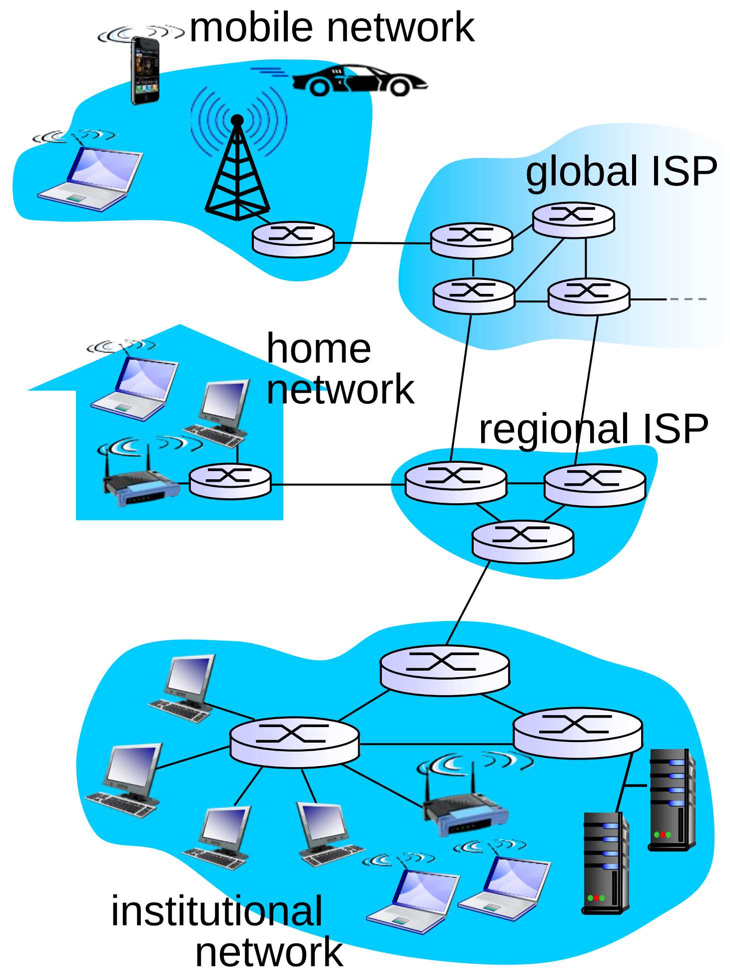
1.6 networks under attack: security

1.7 history

What's the Internet



- *hosts, end systems*
- *communication links*
- *routers and switches*



Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

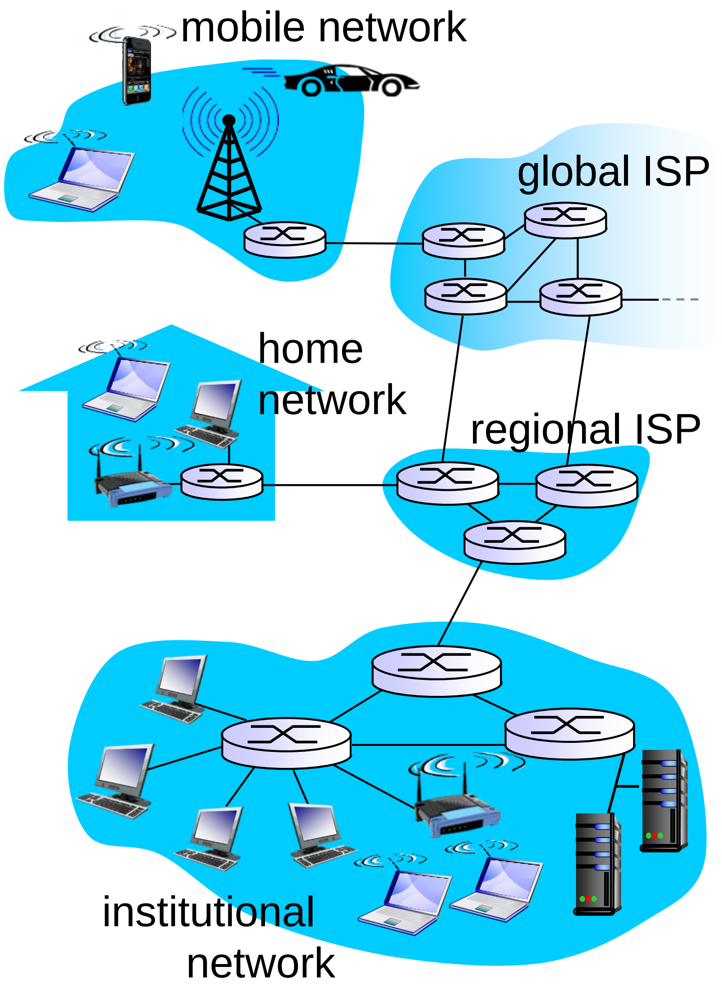
1.5 protocol layers, service models

1.6 networks under attack: security

1.7 history

A closer look at network structure:

- *network edge:*
  - hosts: clients and servers
  - servers often in data centers



Access networks

- DSL: several Mbps, dedicated access
- Cable: tens of Mbps, shared access
- Ethernet: Gbps, for institutional networks
- Wireless: WIFI, 3G/4G cellular

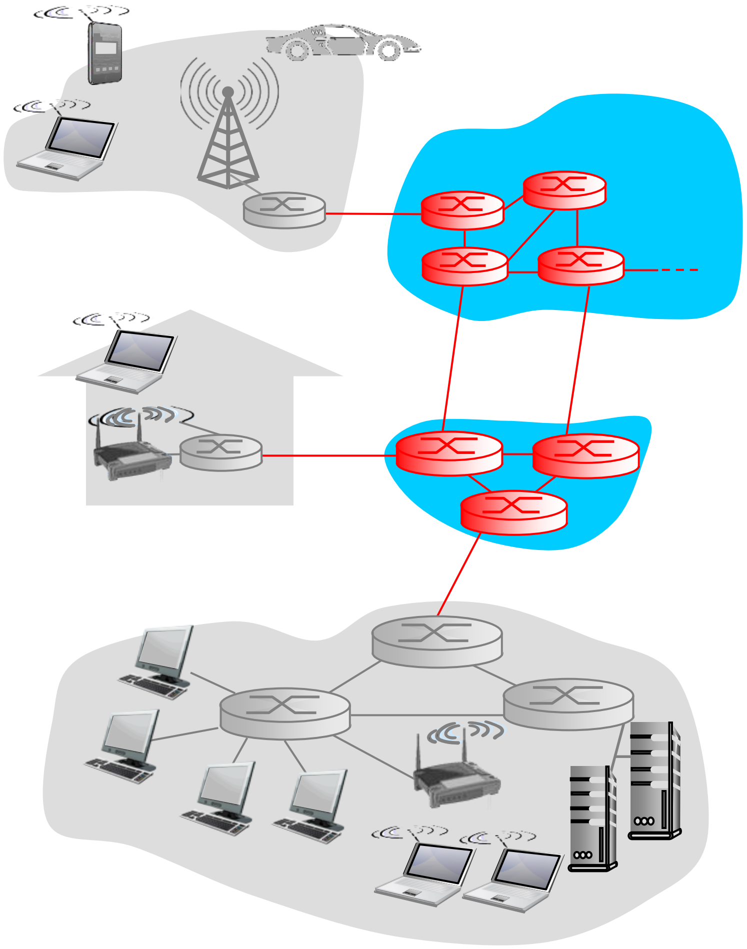
Physical Media

- guided media
  - Twisted pair: Ethernet
  - Coax: cable networks
  - Fiber: optical networks
- unguided media
  - terrestrial microwave
  - LAN (e.g., Wifi)
  - wide-area (e.g., cellular)
  - satellite

Chapter 1: roadmap

The network core

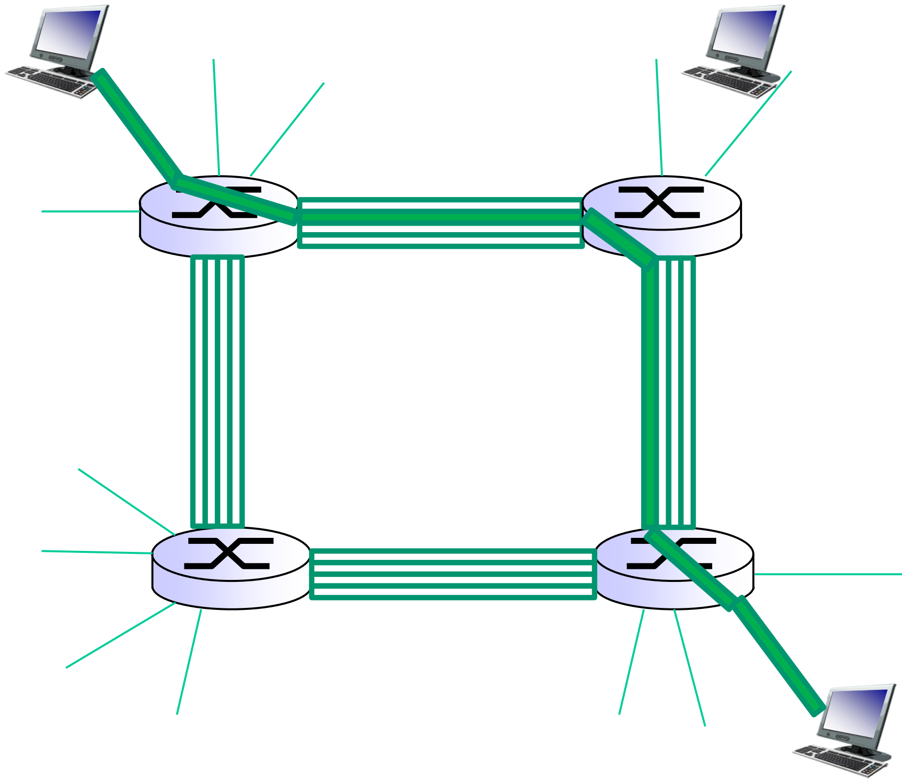
- mesh of interconnected routers
- *the* fundamental question: how is data transferred through net?
  - circuit switching: dedicated circuit per call: telephone net, GSM
  - packet-switching: data sent thru net in discrete "chunks"



Circuit switching

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

- dedicated resources: no sharing
- circuit-like (guaranteed) performance
- call setup required



Network Core: Packet Switching

each end-end data stream divided into *packets*

- user A, B packets *share* network resources
- each packet uses full link bandwidth
- resources used *as needed*

resource contention:

- aggregate resource demand can exceed amount available
- congestion: packets queue, wait for link use
- store and forward: packets move one hop at a time

Packet switching versus circuit switching

	Circuit switching	Packet switching
Pros	• Performance guarantees	• High resource utilization • No call set up
Cons	• Low resource utilization • Call set up	• Congestion

Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

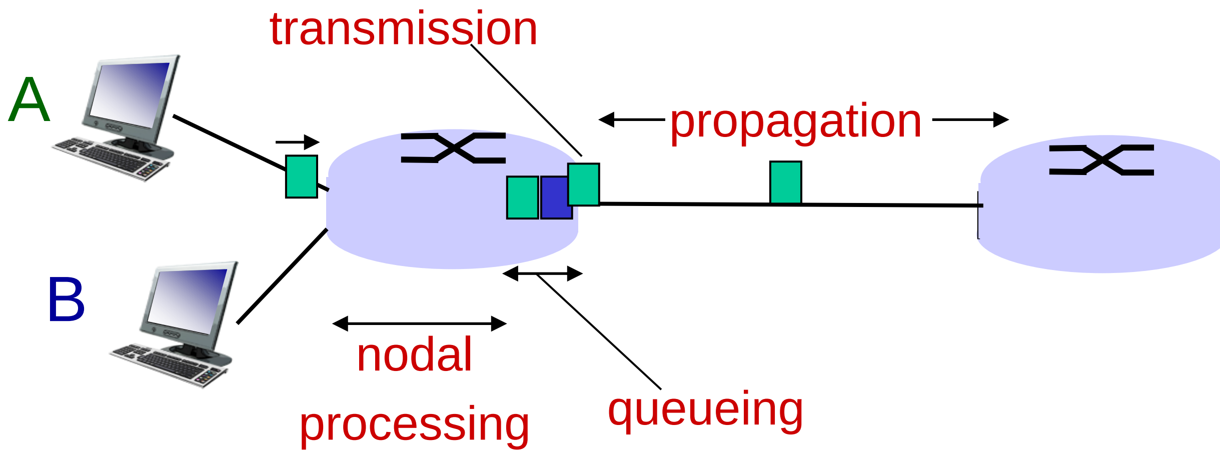
1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

1.6 networks under attack: security

1.7 history

Four sources of packet delay



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

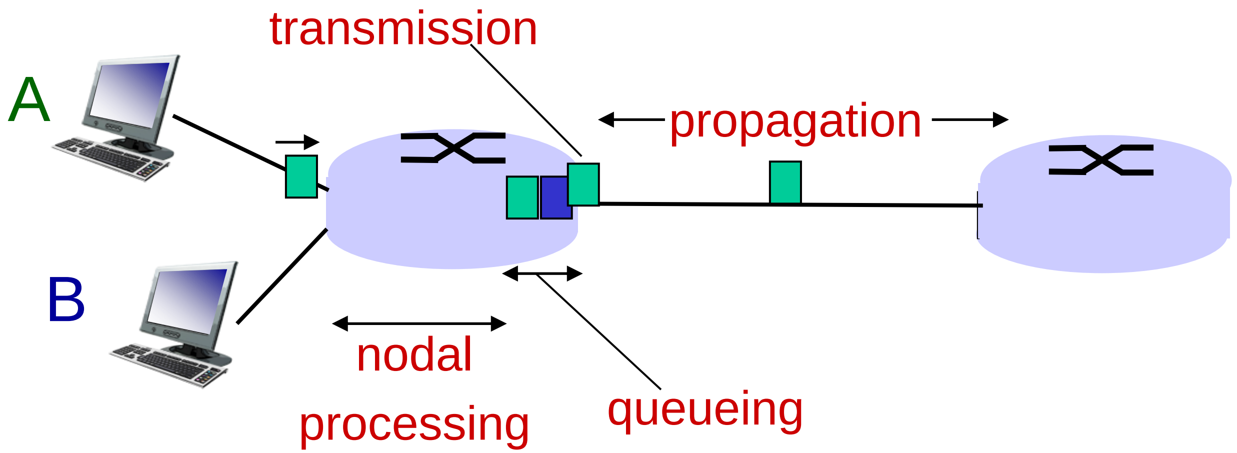
d_{proc}: nodal processing

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

d_{queue}: queueing delay

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

Four sources of packet delay



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

d_{trans}: transmission delay:

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

d_{prop}: propagation delay:

- d: length of physical link
- s: propagation speed in medium ($\sim 2 \times 10^8$ m/sec)
- $d_{prop} = d/s$

Throughput

- *throughput:* rate (bits/time unit) at which bits transferred between sender/receiver
  - *instantaneous:* rate at given point in time
  - *average:* rate over longer period of time
- determined by bottleneck link
  - link on end-end path that constrains end-end throughput

Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge
- end systems, access networks, links

1.3 network core
- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

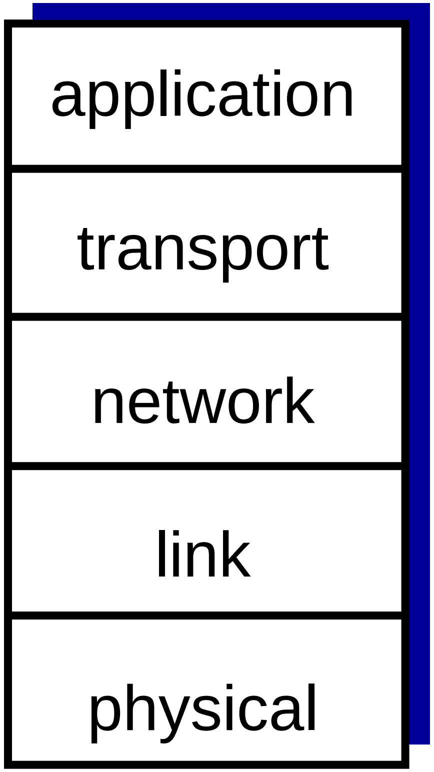
1.5 protocol layers, service models

1.6 networks under attack: security

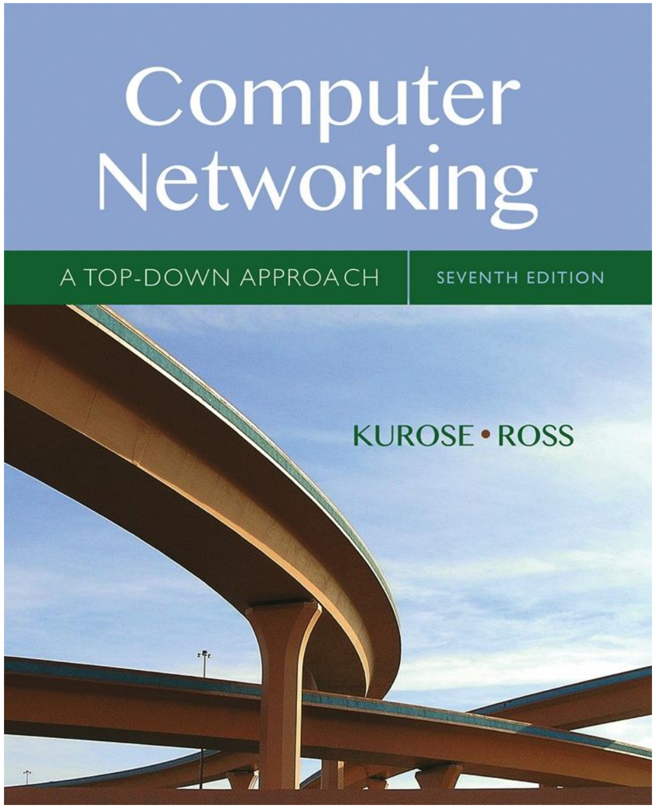
1.7 history

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 elements
  - Ethernet, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"



Chapter 2
Application Layer



Computer Networking: A Top Down Approach

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

Slides adopted from original ones provided by the textbook authors.

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

Application architectures

- Client-server
  - Always-on server, intermittently connected client.
  - Servers are bottlenecks.

- Peer-to-peer (P2P)
  - Peers intermittently connected.
  - Highly scalable but difficult to manage.

Sockets

- Process sends/receives messages to/from its socket.
  - Processes are identified by IP addresses and TCP/UDP port numbers.
- OS provides APIs for creating sockets.

Internet transport protocols services

TCP service:

- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantee, security
- *connection-oriented:* setup required between client and server processes

UDP service:

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, orconnection setup,

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

HTTP overview

uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"

- server maintains no information about past client requests

HTTP connections

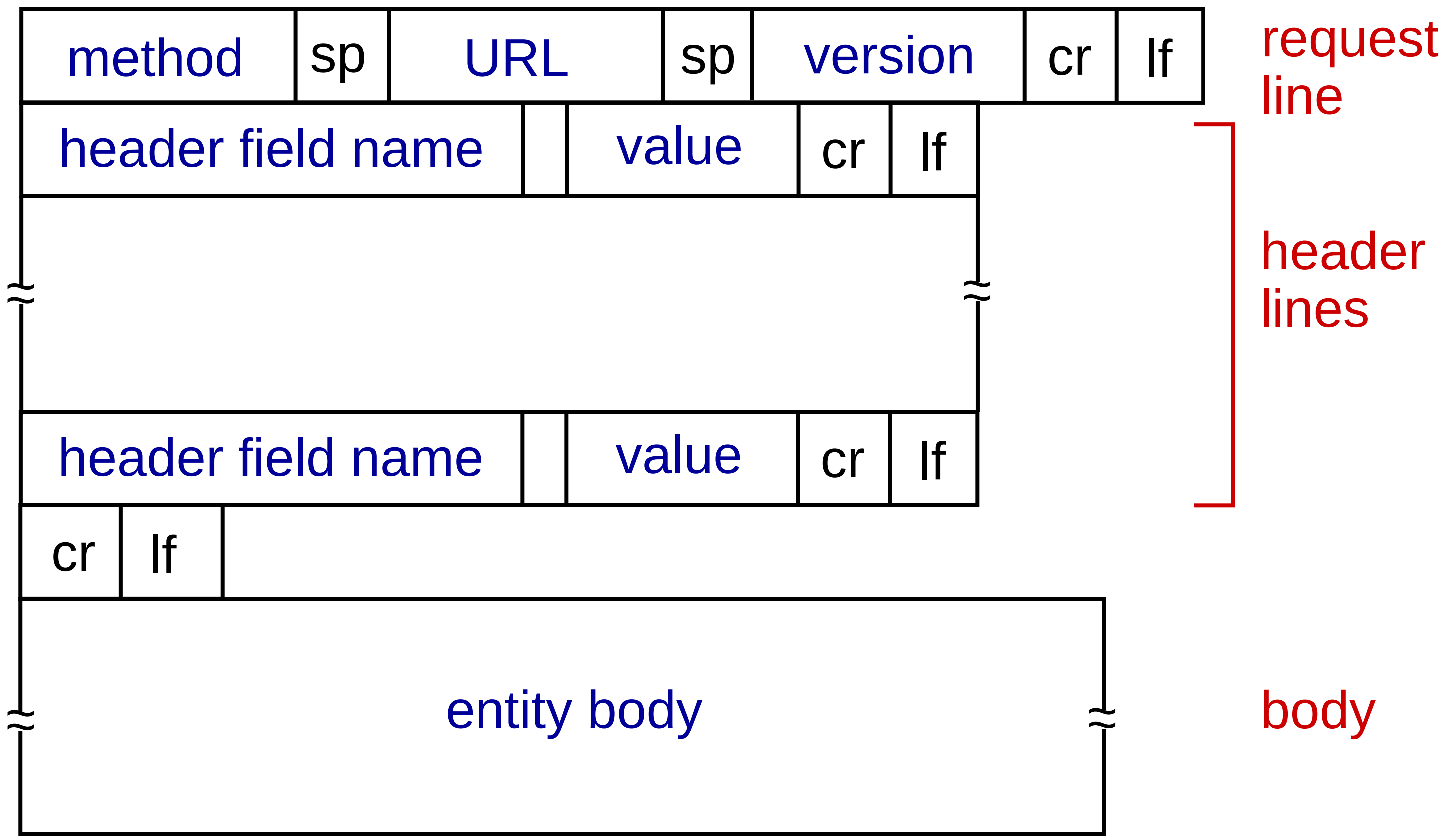
non-persistent HTTP

- at most one object sent over TCP connection
  - connection then closed
- downloading multiple objects required multiple connections
  - use parallel TCP connections to accelerate

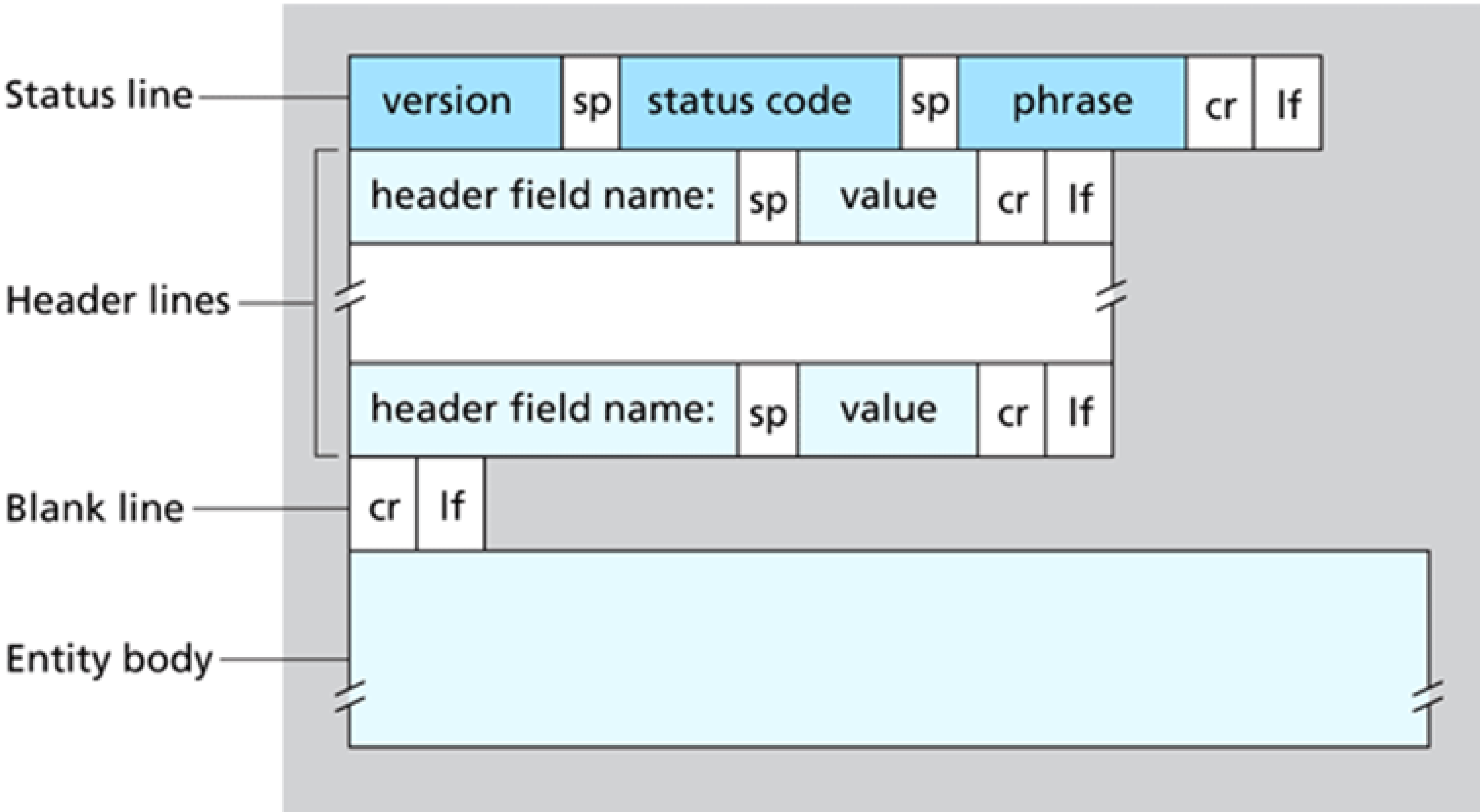
persistent HTTP

- multiple objects can be sent over single TCP connection between client, server

HTTP request message: general format



HTTP response message: general format



User-server state: cookies

- Cookies help web sites remember use states.
- Four components:
  1) set-cookie header line in 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

- Goal: satisfy client request without involving origin server
  - reduce response time and traffic
- user sets browser: Web accesses via cache, browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client
- Conditional GET: don't send object if cache has up-to-date cached version

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

Electronic Mail

- SMTP
  - mail transfer protocol
  - client/server model
  - based on TCP, port 25
- Mail message format defined in RFC 822
  - header and body
- POP3 and IMAP
  - mail access protocols

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

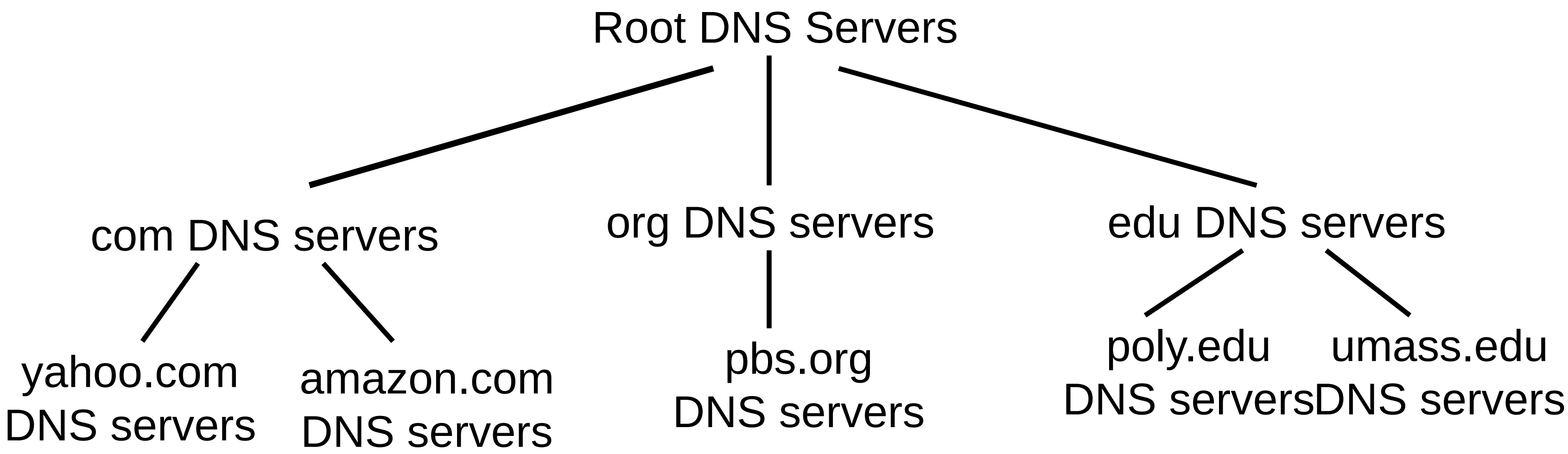
2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

DNS: Domain Name System

- DNS services
  - hostname to IP address translation
  - host aliasing: canonical, alias names
  - mail server aliasing
  - load distribution

Distributed, Hierarchical Database



- Root name servers
- Top-level domain (TLD) servers
- Authoritative DNS servers

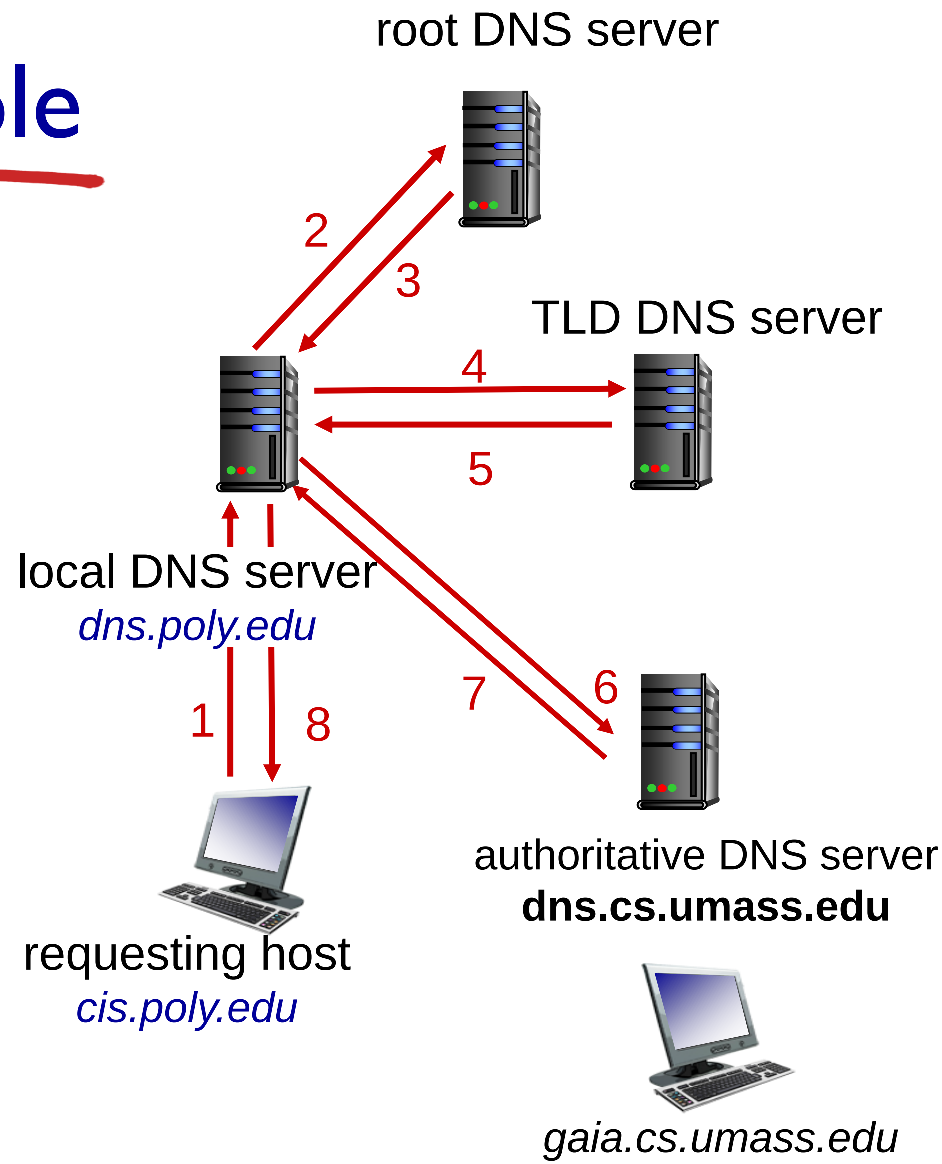
- Local name server

DNS name resolution example

- host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query:

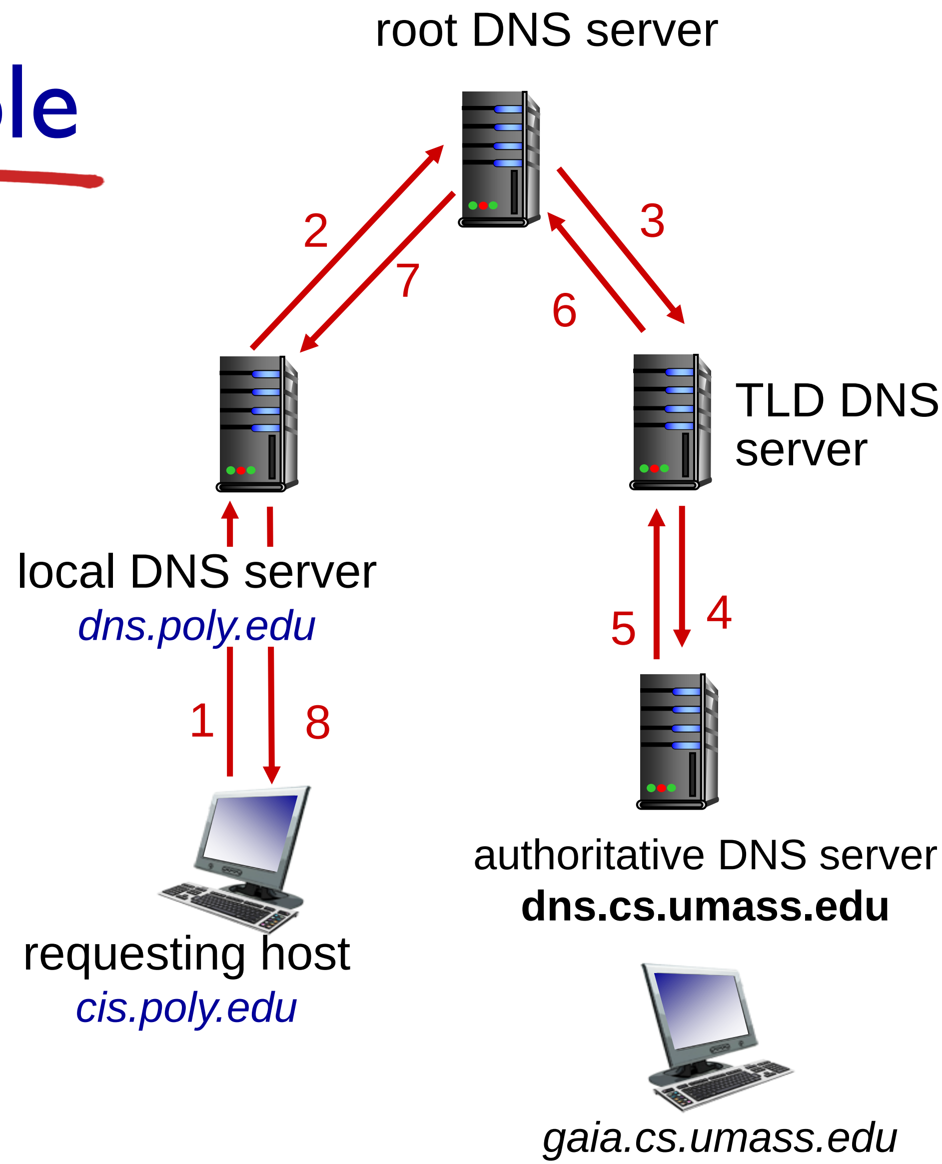
- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”



DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy



DNS: caching, updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time (TTL)
  - TLD servers typically cached in local name servers

- cached entries may be *out-of-date* (best effort name-to-address translation!)
  - if name host changes IP address, may not be known Internet-wide until all TTLs expire

DNS records

DNS: distributed db storing resource records (RR)

RR format: `(name, value, type, ttl)`

type=A

- **`name`** is hostname
- **`value`** is IP address

type=NS

- **`name`** is domain (e.g., foo.com)
- **`value`** is hostname of authoritative name server for this domain

type=CNAME

- **`name`** is alias name for some "canonical" (the real) name
- **`www.ibm.com`** is really **`servereast.backup2.ibm.com`**
- **`value`** is canonical name

type=MX

- **`value`** is name of mailserver associated with **`name`**

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

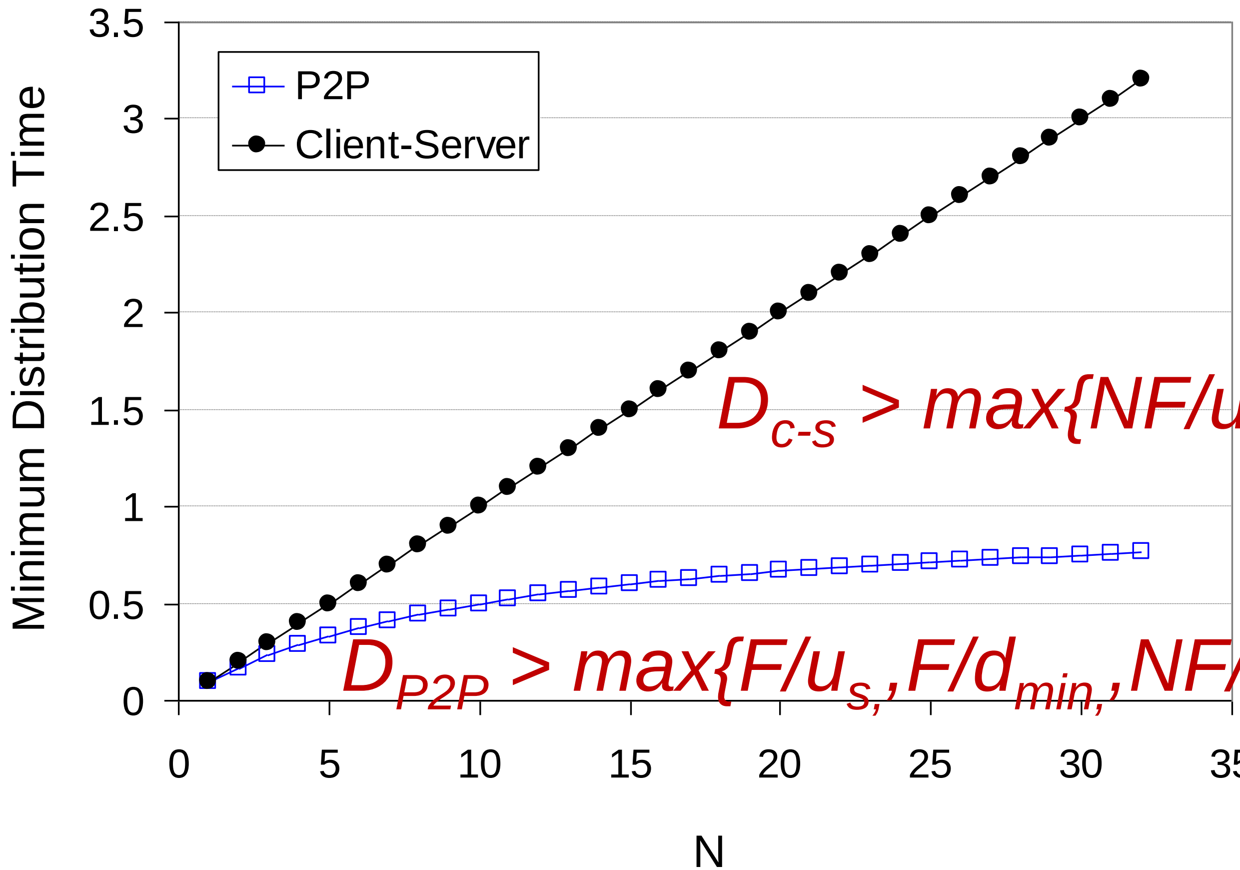
2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

Client-server vs. P2P: example

client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$

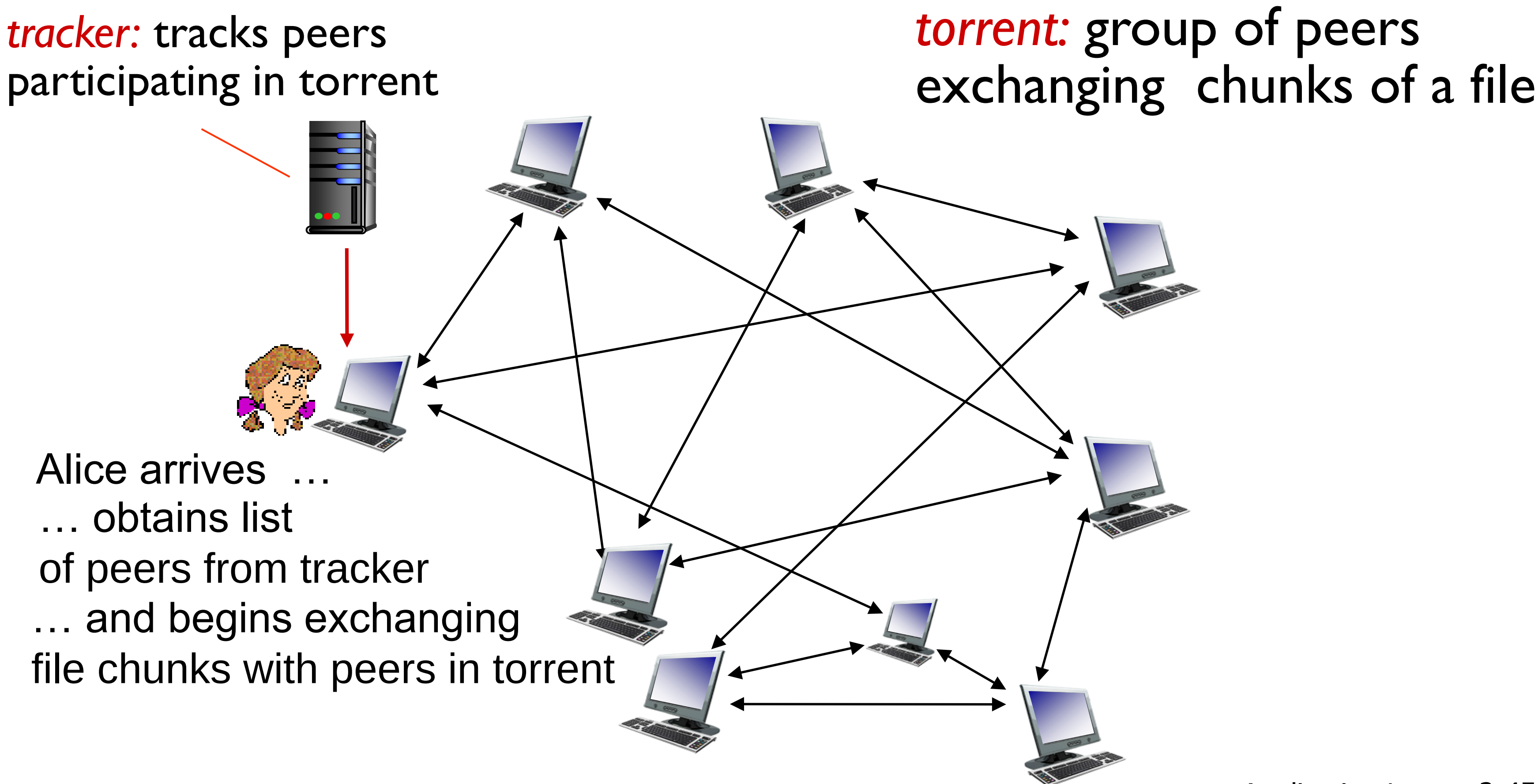


$D_{c\text{-}s} > max\{NF/u_{s,},F/d_{min}\}$

$D_{P2P} > max\{F/u_{s,},F/d_{min,},NF/(u_s + \Sigma u_i)\}$

P2P file distribution: BitTorrent

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks



BitTorrent: requesting, sending file chunks

requesting chunks:

- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

sending chunks: tit-for-tat

- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
  - newly chosen peer may join top 4

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (CDNs)

2.7 socket programming with UDP and TCP

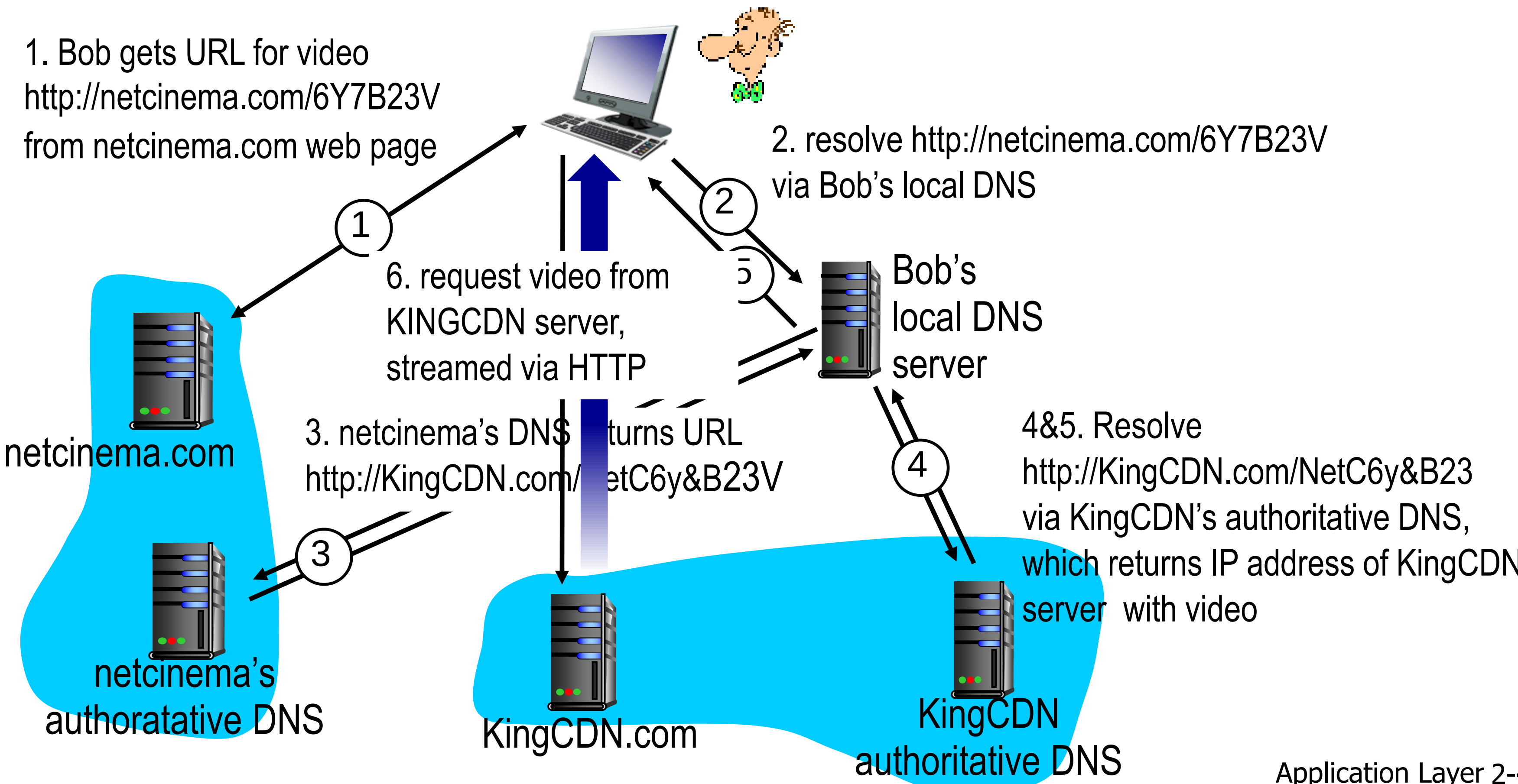
Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file:* provides URLs for different chunks
- *client:*
  - periodically measures server-to-client bandwidth
  - consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time (depending on available bandwidth at time)

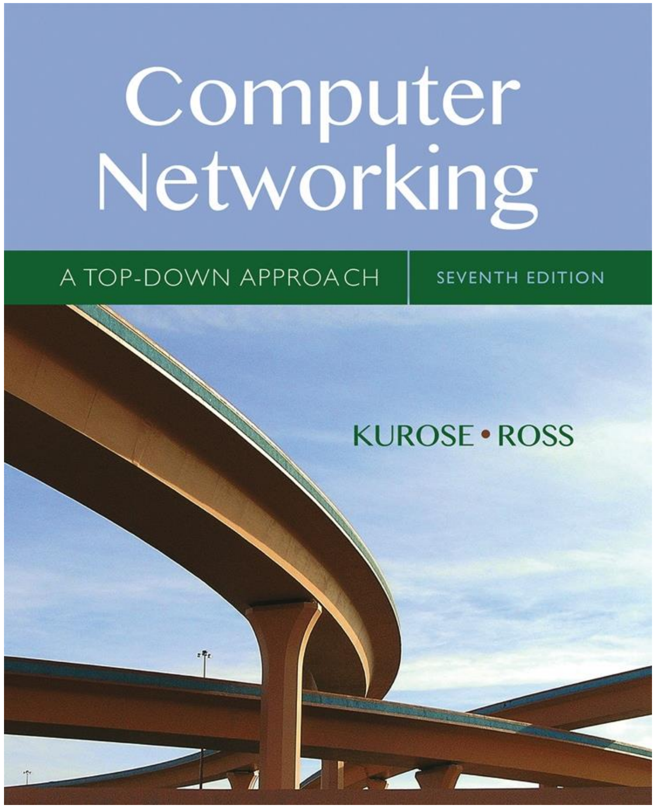
CDN content access: a closer look

Bob (client) requests video http://netcinema.com/6Y7B23V

- video stored in CDN at http://KingCDN.com/NetC6y&B23V



Chapter 3
Transport Layer



Computer Networking: A Top Down Approach

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

Slides adopted from original ones provided by the textbook authors.

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

Transport vs. network layer

- transport layer: logical communication between processes
  - relies on, enhances, network layer services
  - network layer: logical communication between hosts

- two transport-layer protocols
  - TCP: reliable, in-order delivery
  - UDP: unreliable, unordered delivery

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

Demultiplexing

- UDP socket identified by 2-tuple:
  - dest IP address
  - dest port number
- TCP socket identified by 4-tuple:
  - source IP address
  - source port number
  - dest IP address
  - dest port number

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

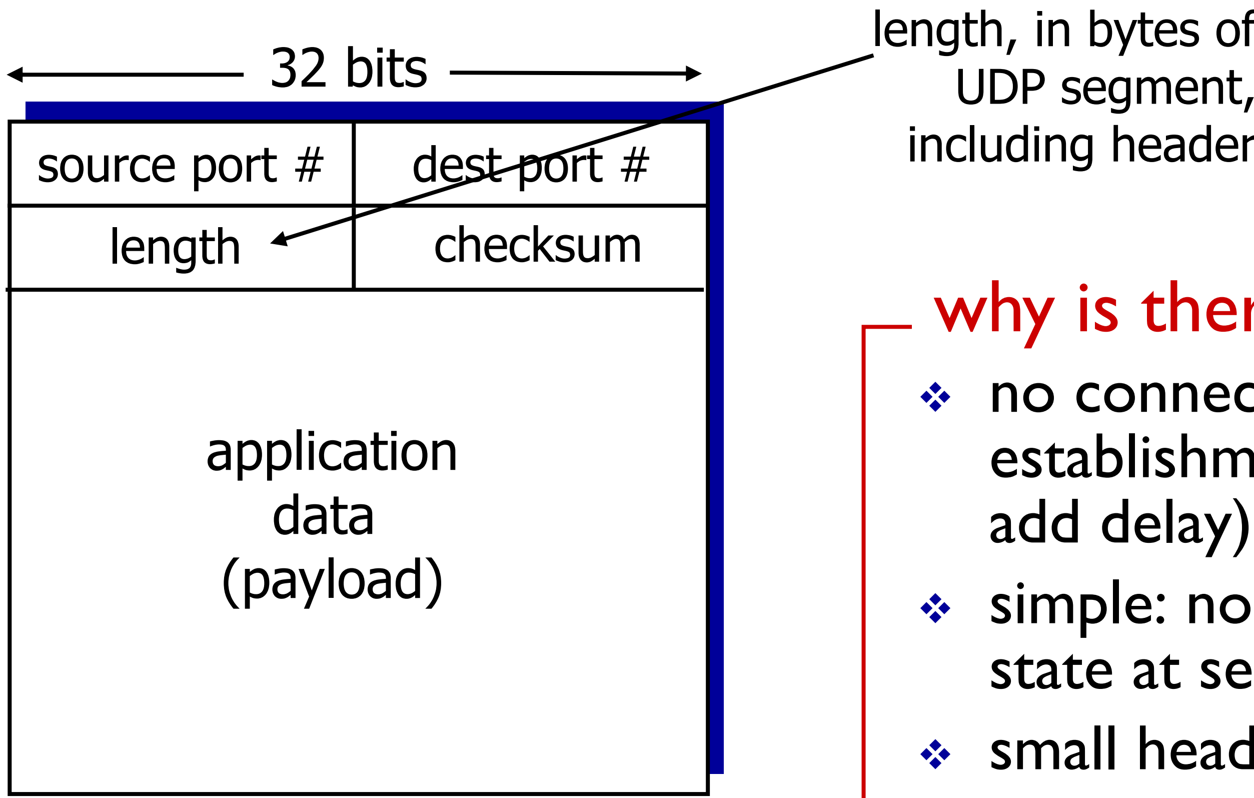
3.7 TCP congestion control

UDP: User Datagram Protocol [RFC 768]

- "best effort" service, UDP segments may be:
  - lost
  - delivered out-of-order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

- UDP use:
  - streaming multimedia apps (loss tolerant, rate sensitive)
  - DNS
  - SNMP

UDP: segment header



UDP segment format

why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small header size
- no congestion control: UDP can blast away as fast as desired

UDP checksum

Goal: detect "errors" (e.g., flipped bits) in transmitted segment

sender:

- treat segment contents, including header fields, as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?* More later ….

Internet checksum: example

example: add two 16-bit integers

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

Reliable Data Transfer

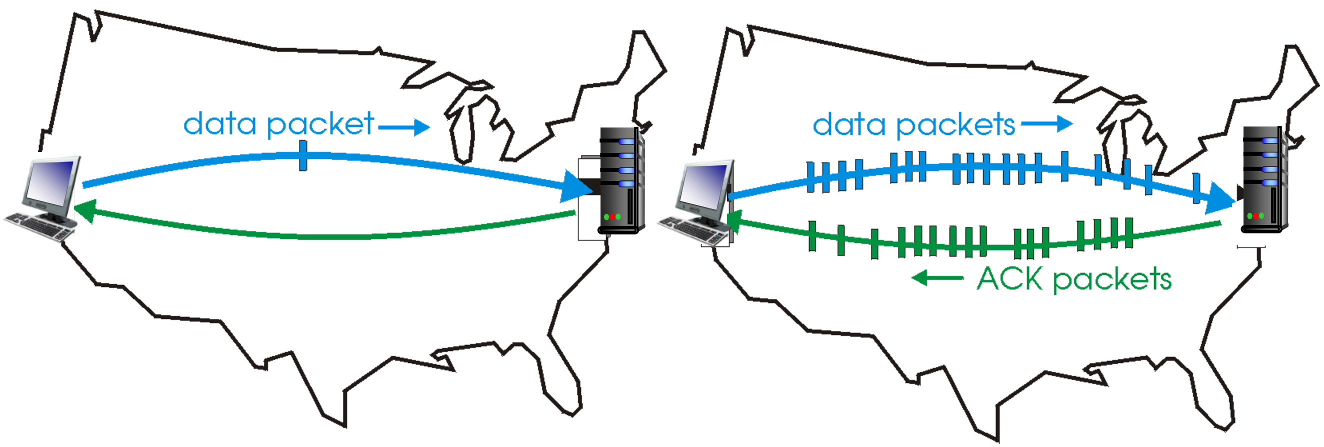
- Challenge: TCP requires reliable data transfer, but the underlying protocol IP is not reliable.

- Possible errors
  - channel with bit errors -> checksum, ACK/NAK, retransmission
  - with corrupted ACK/NAKs -> retransmission, sequence #
  - without NAKs -> ACK retransmission
  - channels with packet loss -> timer

Pipelined protocols

stop-and-wait: one packet at a time

pipelining: sender allows multiple, "in-flight", yet-to-be-acknowledged pkts



(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

utilization: $$U_{sender} = \frac{n\,L / R}{RTT + L / R}$$

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

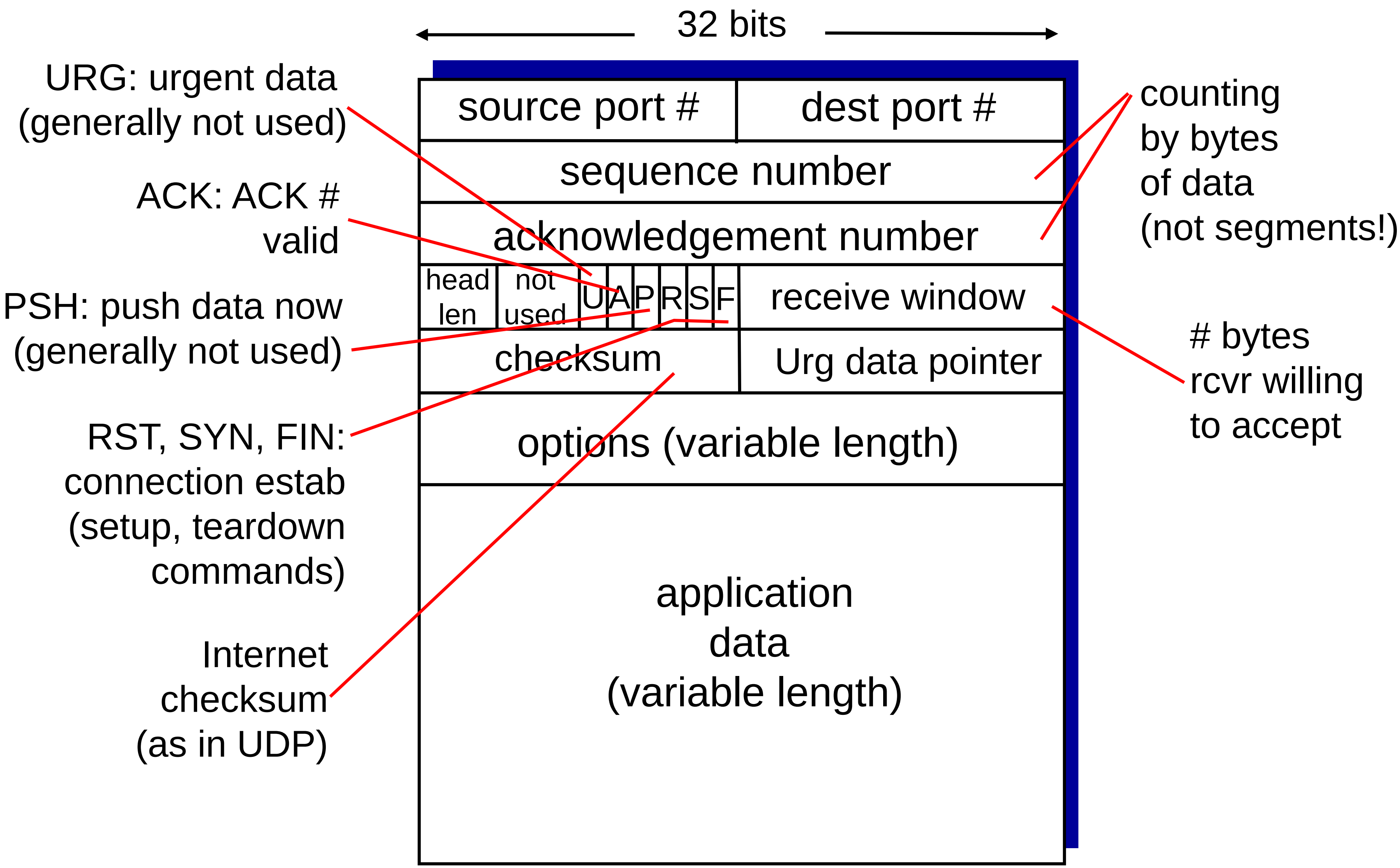
3.6 principles of congestion control

3.7 TCP congestion control

TCP: Overview RFCs: 793,1122,1323, 2018, 2581

- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte stream:*
  - no "message boundaries"
- pipelined:
  - TCP congestion and flow control set window size
- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- connection-oriented:
  - handshaking (exchange of control msgs) inits sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver

TCP segment structure



Maximum segment size (MSS)

- MSS: maximum bytes of TCP payload
- Sequence #: byte-stream # of first byte in segment
- E.g. file size 500,000 bytes, MSS 1,000 bytes

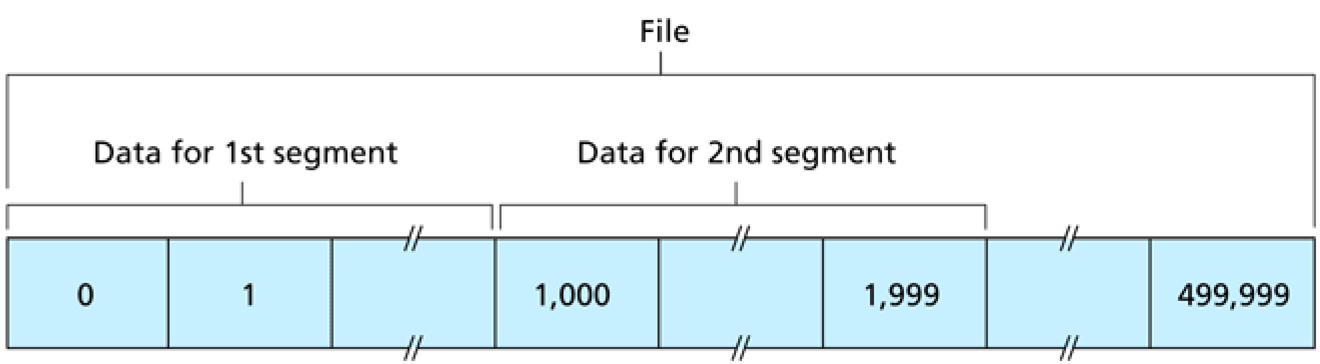


Figure 3.30 ♦ Dividing file data into TCP segments

TCP seq. #'s and ACKs

Seq. #'s:

- byte stream "number" of first byte in segment's data

ACKs:

- seq # of next byte expected from other side
- cumulative ACK

Setting the time out

- `TimeoutInterval = EstimatedRTT + 4*DevRTT`
- `EstimatedRTT = (1-` α`)*EstimatedRTT + `α`*SampleRTT`
- `DevRTT = (1-`β`)*DevRTT + `β`*|SampleRTT-EstimatedRTT|`

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

TCP sender events:

data rcvd from app:

- create segment with seq #
- seq # is byte-stream number of first data byte in segment
- start timer if not already running
  - think of timer as for oldest unacked segment
  - expiration interval: `TimeOutInterval`

timeout:

- retransmit segment that caused timeout
- restart timer

ack rcvd:

- if ack acknowledges previously unacked segments
  - update ACK status
  - start timer if there are still unacked segments
  - triple duplicate ACKs: retransmit

TCP receiver events

event at receiver	*TCP receiver action*
arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
arrival of in-order segment with expected seq #. One other segment has ACK pending	immediately send single cumulative ACK, ACKing both in-order segments
arrival of out-of-order segment higher-than-expect seq. # . Gap detected	immediately send *duplicate ACK,* indicating seq. # of next expected byte
arrival of segment that partially or completely fills gap	immediate send ACK, provided that segment starts at lower end of gap

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

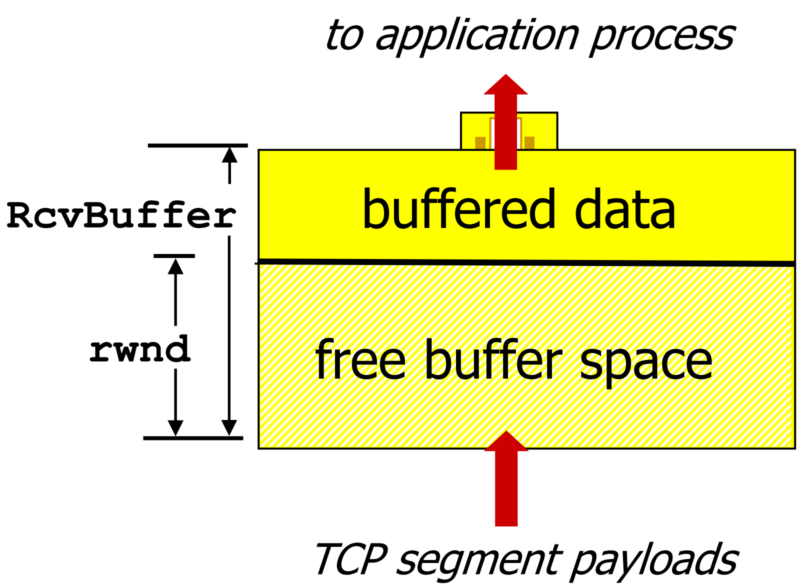
3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

TCP flow control

- receiver "advertises" free buffer space by including **rwnd** value in TCP header of receiver-to-sender segments
  - **RcvBuffer** size set via socket options (typical default is 4096 bytes)
  - many operating systems autoadjust **RcvBuffer**
- sender limits amount of unacked ("in-flight") data to receiver's **rwnd** value
- guarantees receive buffer will not overflow



receiver-side buffering

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

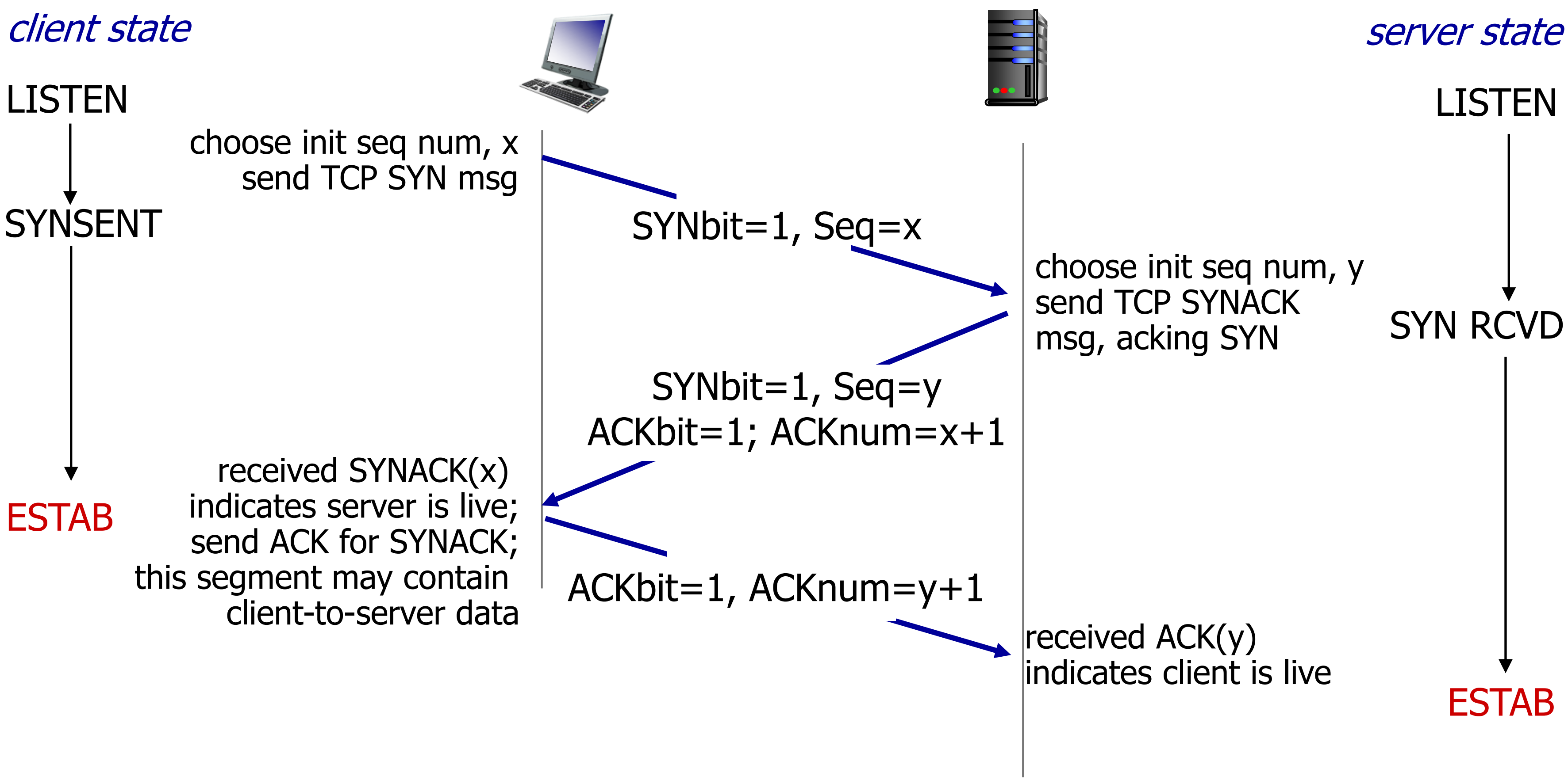
3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

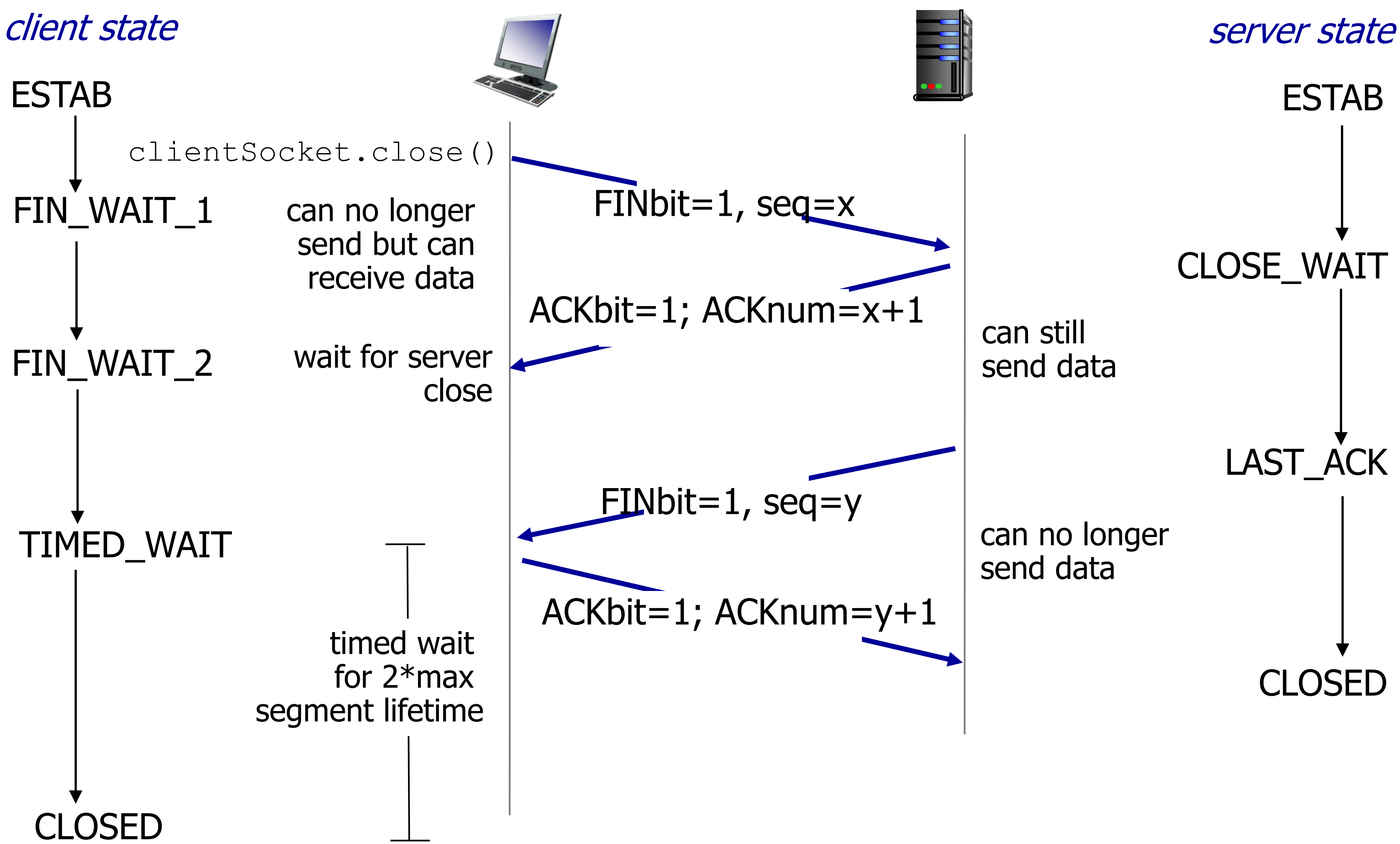
3.6 principles of congestion control

3.7 TCP congestion control

TCP 3-way handshake



TCP: closing a connection



Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

Approaches towards congestion control

two broad approaches towards congestion control:

end-end congestion control:

- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

network-assisted congestion control:

- routers provide feedback to end systems
  - single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
  - explicit rate for sender to send at

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

Summary: TCP Congestion Control

- when **cwnd < ssthresh**, sender in slow-start phase, window grows exponentially.
- when **cwnd >= ssthresh**, sender is in congestion-avoidance phase, window grows linearly.
- when triple duplicate ACK occurs, **ssthresh** set to **cwnd/2**, **cwnd** set to **ssthresh + 3 MSS**
- when timeout occurs, **ssthresh** set to **cwnd/2**, **cwnd** set to 1 MSS.