Lecture 4 – Transport Layer

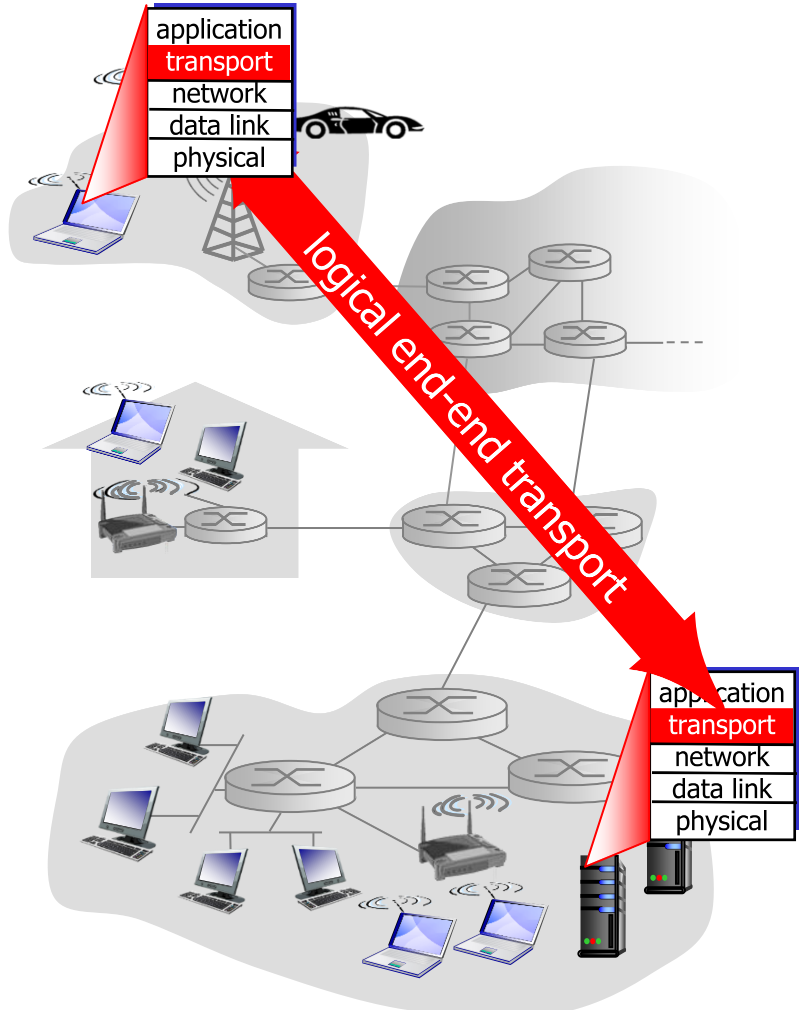
COMP1002 (Cybersecurity and Networks)

Outline

- Transport Layer Services
- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)
  - Segment structure
  - Connection management
  - Reliable data transfer
  - Flow control
  - Congestion control

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
- more than one transport protocol available to apps
  - Internet: TCP and UDP

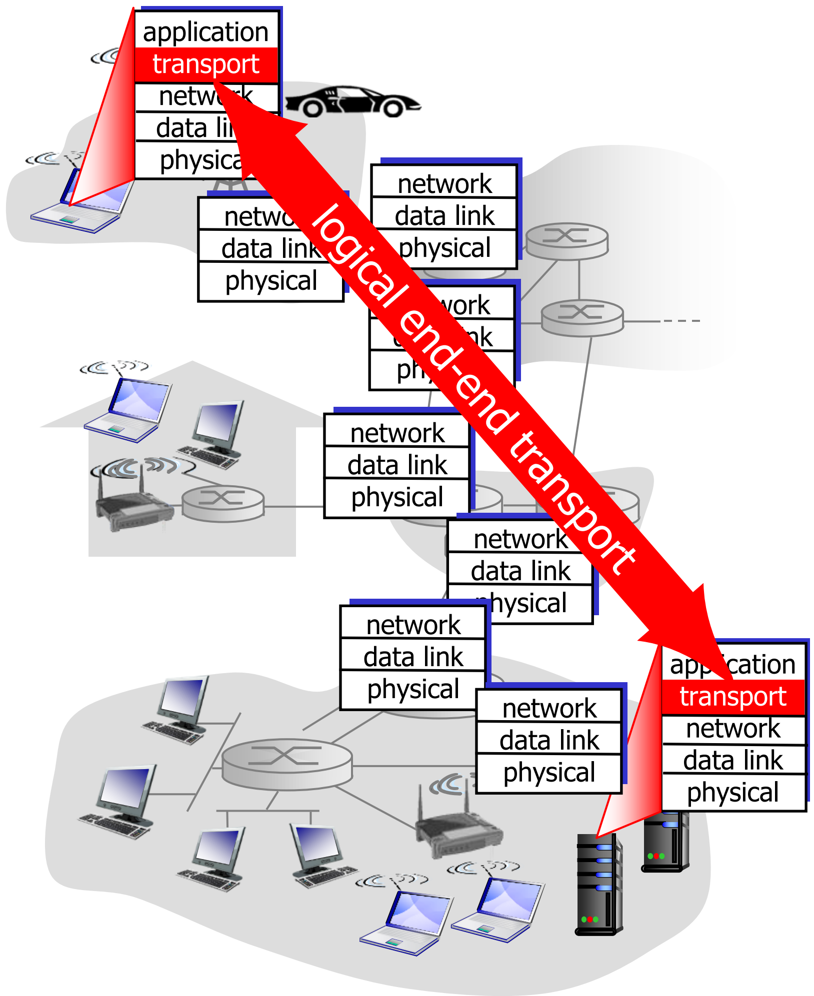


Transport vs. network layer

- Network layer:
  - logical communication between **hosts**
- Transport layer:
  - Logical communication between **processes**
  - Relies on, enhances, network layer services

Internet transport-layer protocols

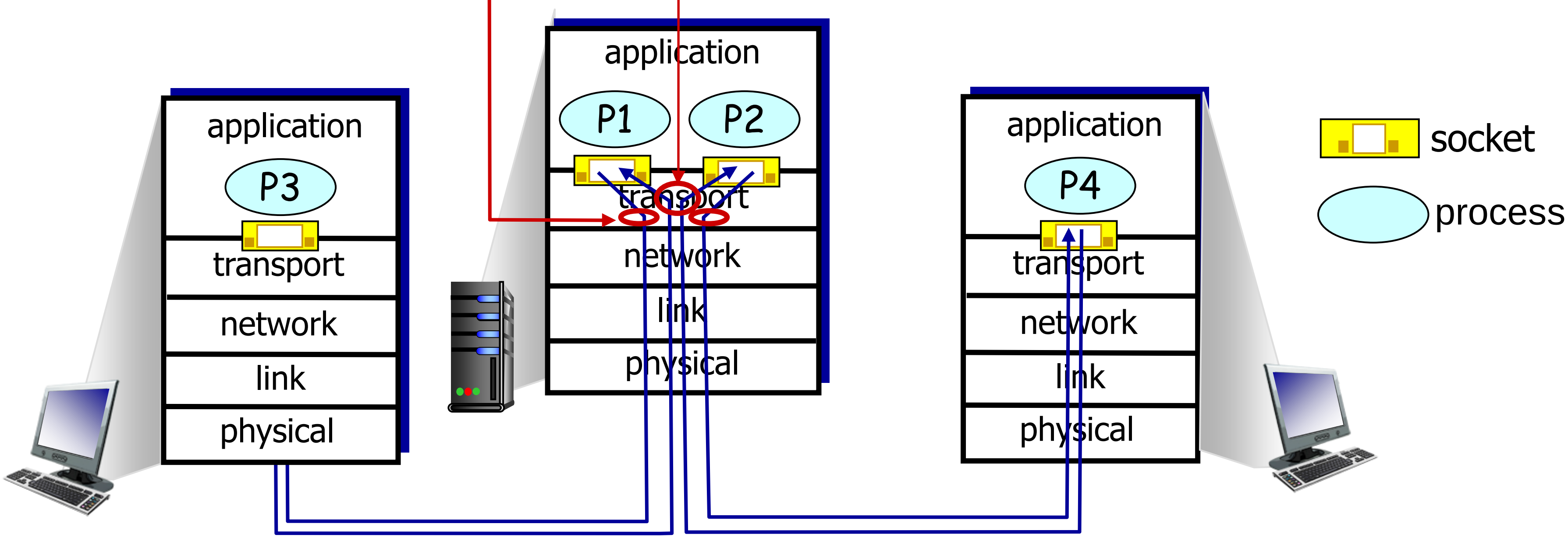
- reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- unreliable, unordered delivery: UDP
  - simple extension of "best-effort" IP
- services not available:
  - delay guarantees
  - bandwidth guarantees



Multiplexing/demultiplexing

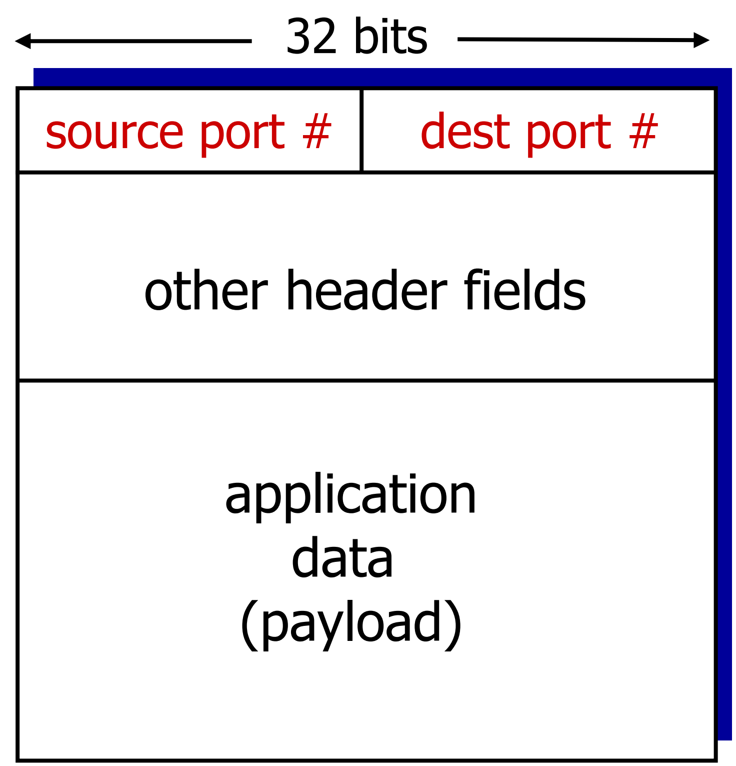
multiplexing at sender:
handle data from multiple sockets, add transport header (later used for demultiplexing)

demultiplexing at receiver:
use header info to deliver received segments to correct socket



How demultiplexing works

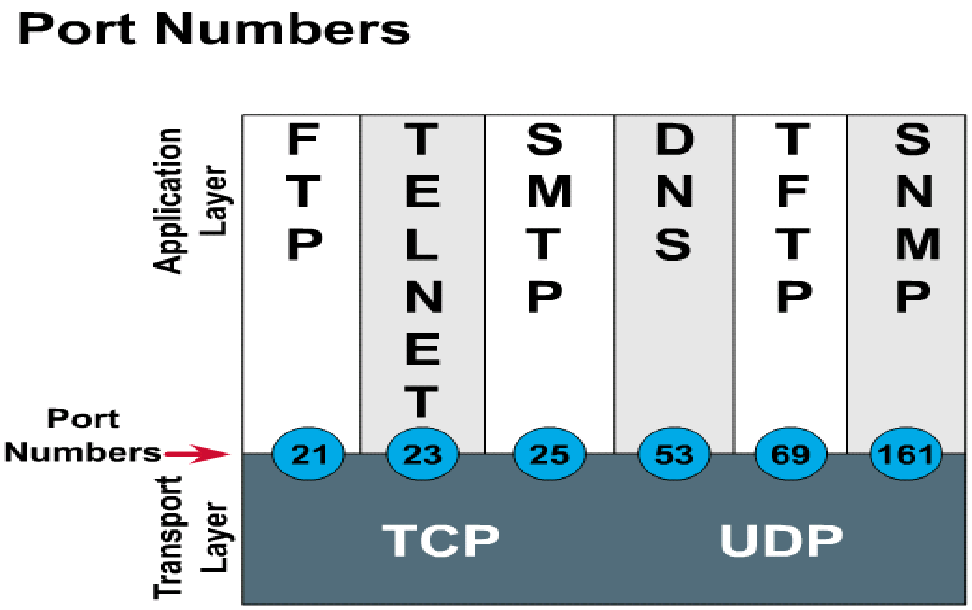
- host receives IP datagrams
  - each datagram has source IP address, destination IP address
  - each datagram carries one transport-layer segment
  - each segment has source, destination port number
- host uses *IP addresses & port numbers* to direct segment to appropriate socket



TCP/UDP segment format

Port numbers

- Historically, ports numbered 1-1023 are well-known/server ports
- Historically, ports numbered >1024 are ephemeral ports



Well known ports are assigned by Internet Assigned Numbers Authority (IANA).
http://www.iana.org

Outline

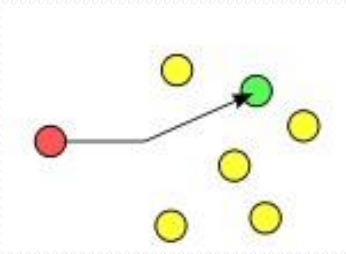
- Transport Layer Services
- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)
  - Segment structure
  - Connection management
  - Reliable data transfer
  - Flow control
  - Congestion control

UDP

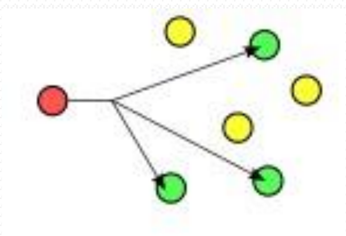
- Defined in RFC 768 - User Datagram Protocol
- “best effort” service, UDP segments may be:
  - Lost
  - Delivered out-of-order to application
- Connectionless
  - No handshaking between UDP sender and receiver
  - Each UDP segment handled independently of others
- No flow control
- No reliability: error detection is possible, but there is no acknowledging mechanism
- Network and processing overheads are low: Only 8 bytes header.

UDP – cont.

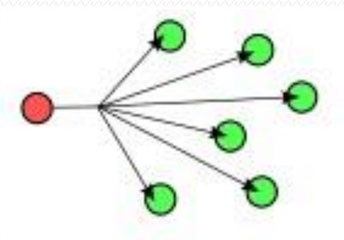
- UDP use:
  - streaming multimedia apps (loss tolerant, rate sensitive)
  - DNS
- reliable transfer over UDP:
  - add reliability at application layer
  - application-specific error recovery!
- UDP supports unicast, multicast, broadcast addresses /applications.



Unicast

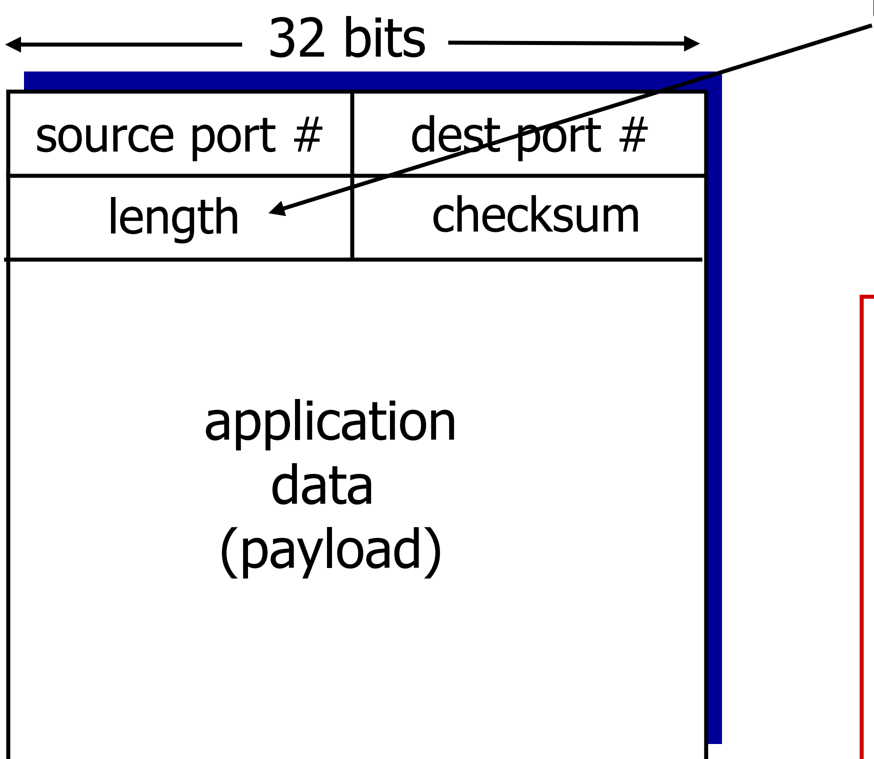


Multicast



Broadcast

UDP: segment header

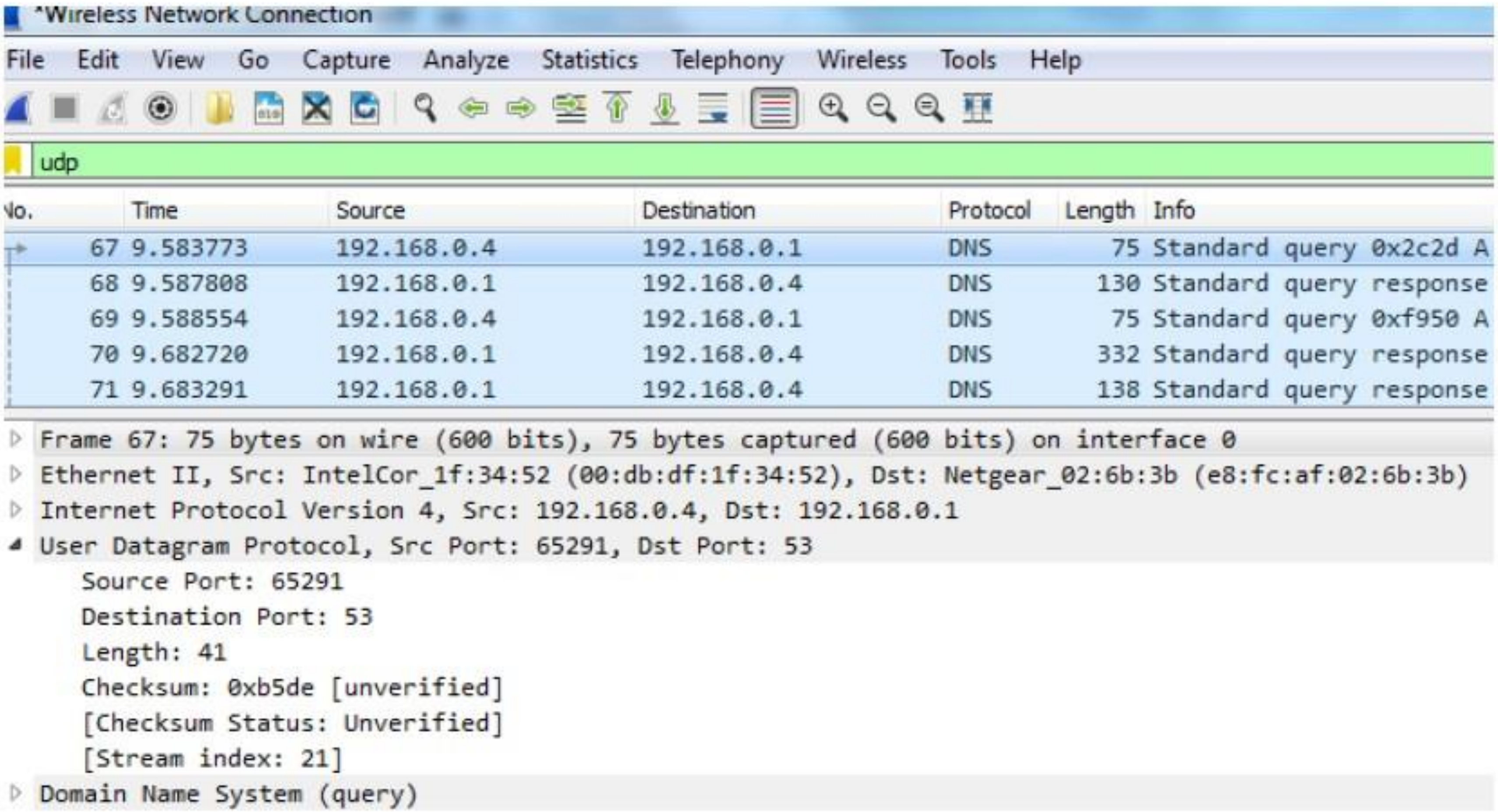


UDP segment format

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 Segment Header



UDP checksum

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

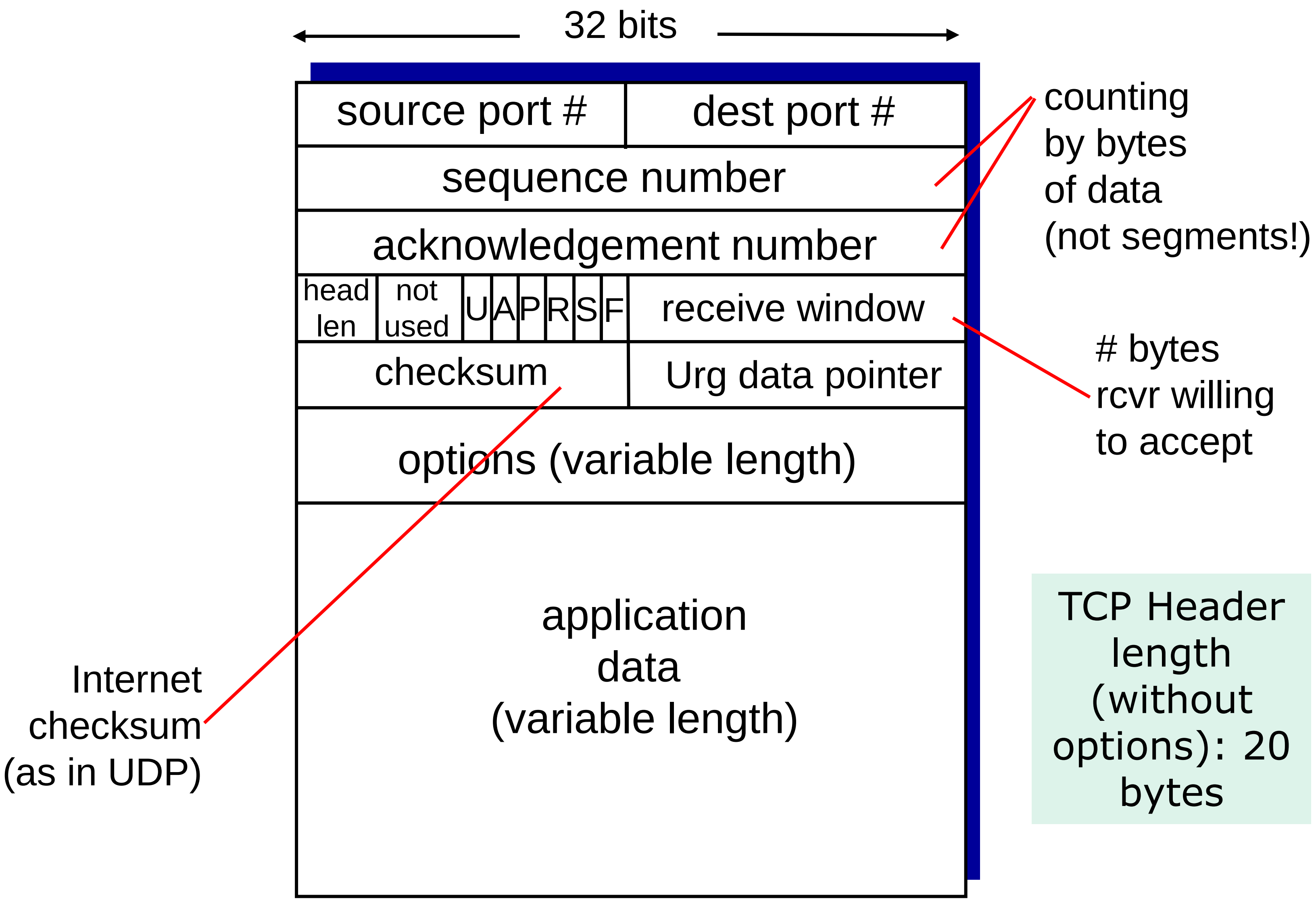
Outline

- Transport Layer Services
- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)
  - Segment structure
  - Connection management
  - Reliable data transfer
  - Flow control
  - Congestion control

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

- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte steam:*
  - 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



TCP Header length (without options): 20 bytes

TCP Segment Header

tcp-ethereal-trace-1.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
12	0.124085	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0
13	0.124185	192.168.1.102	128.119.245.12	TCP	1201	1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147
14	0.169118	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0
15	0.217299	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0
16	0.267802	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=7866 Win=20440 Len=0

```
> Frame 14: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
> Ethernet II, Src: LinksysG_da:af:73 (00:06:25:da:af:73), Dst: Actionte_8a:70:1a (00:20:e0:8a:70:1a)
> Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.1.102
v Transmission Control Protocol, Src Port: 80, Dst Port: 1161, Seq: 1, Ack: 4946, Len: 0
    Source Port: 80
    Destination Port: 1161
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 4946    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window size value: 14600
    [Calculated window size: 14600]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0x6e88 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
```

TCP Flags

- FIN: Gracefully terminate a connection
- SYN: Establish a new TCP session
- RST: (RESET) Abort a TCP session
- PSH: push data now
- ACK: Acknowledge the receipt of data
- URG: Signifies that urgent data is being sent (generally not used)

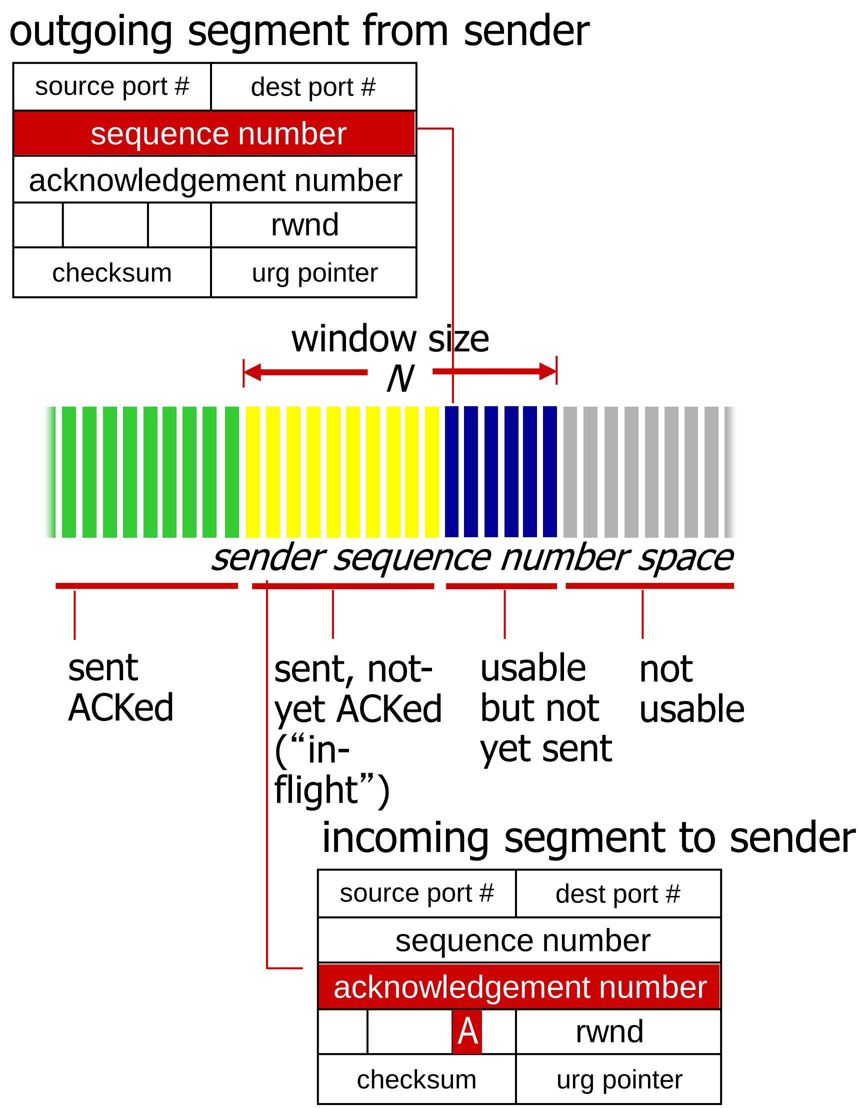
TCP seq. numbers, ACKs

sequence numbers:

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

acknowledgements:

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

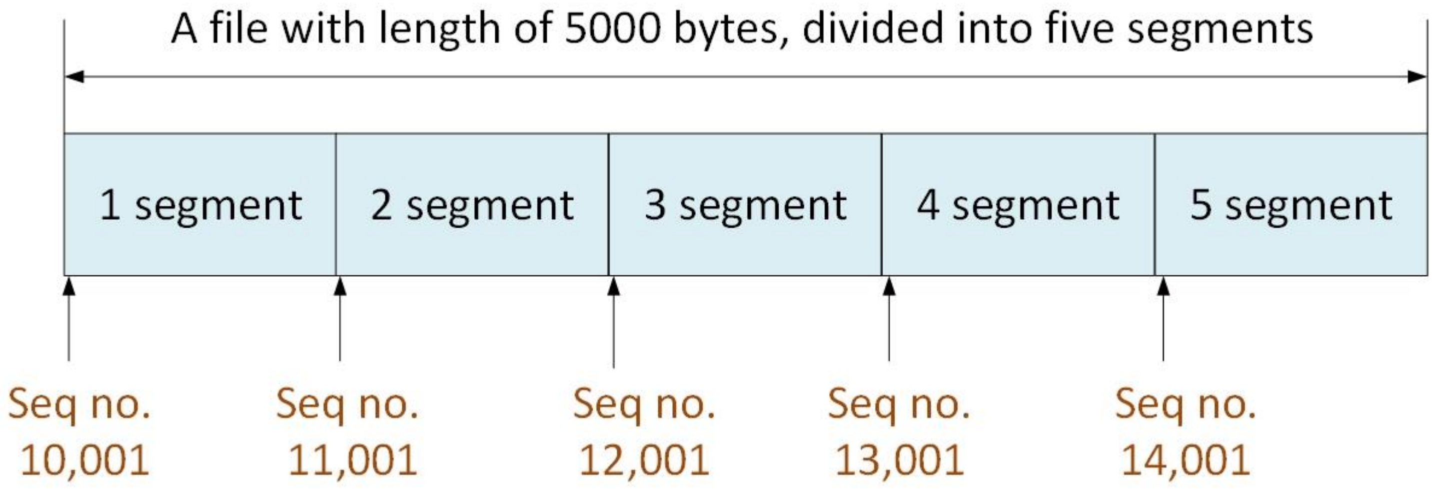


Example

Suppose a TCP connection is transferring a file of 5,000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1,000 bytes?

Example (cont.)

Suppose a TCP connection is transferring a file of 5,000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1,000 bytes?



Note: normally the first seq number is randomly generated.

Example (cont.)

Suppose a TCP connection is transferring a file of 5,000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1,000 bytes?

Solution:

The following shows the seq. number for each segment:

Seg. 1 --> Seq. No: 10,001 Range: 10,001 to 11,000

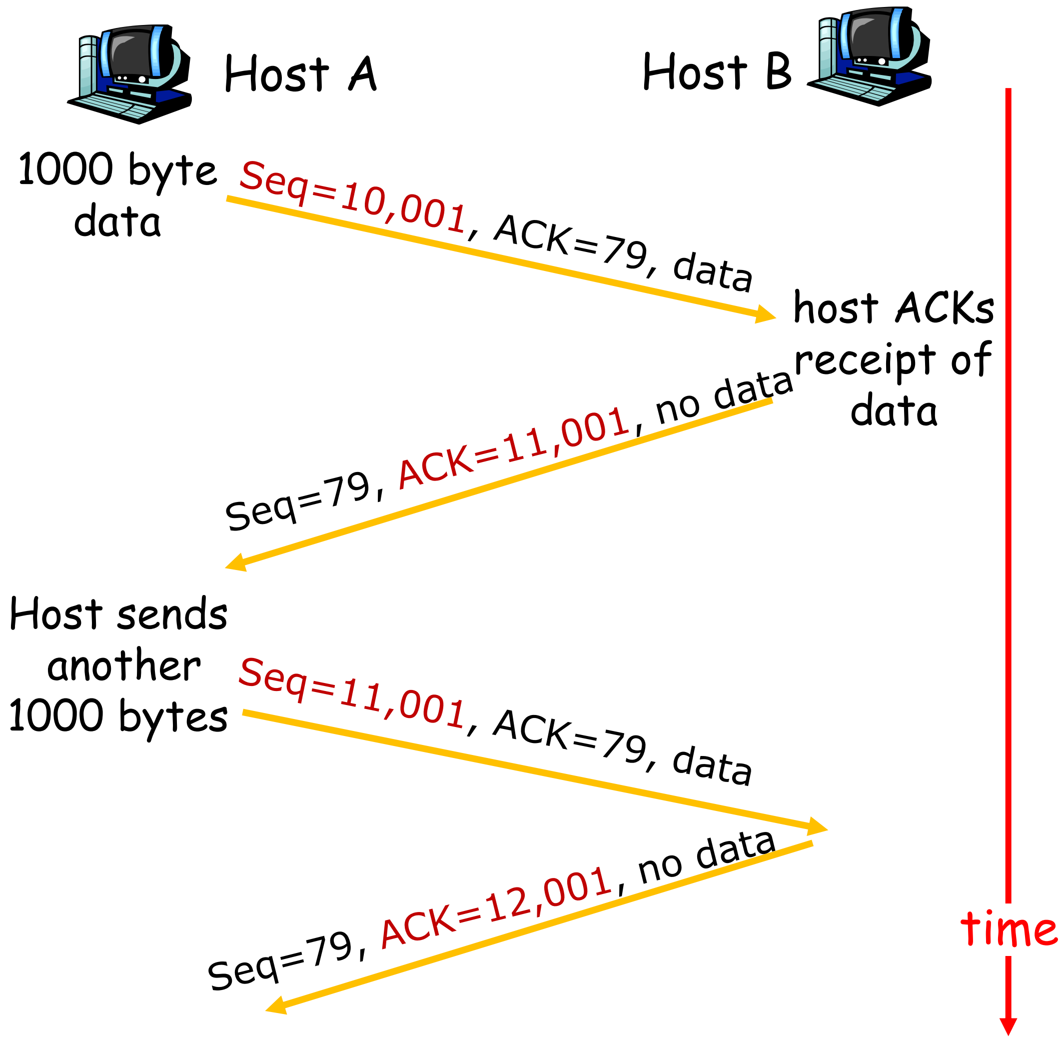
Seg. 2 --> Seq. No: 11,001 Range: 11,001 to 12,000

Seg. 3 --> Seq. No: 12,001 Range: 12,001 to 13,000

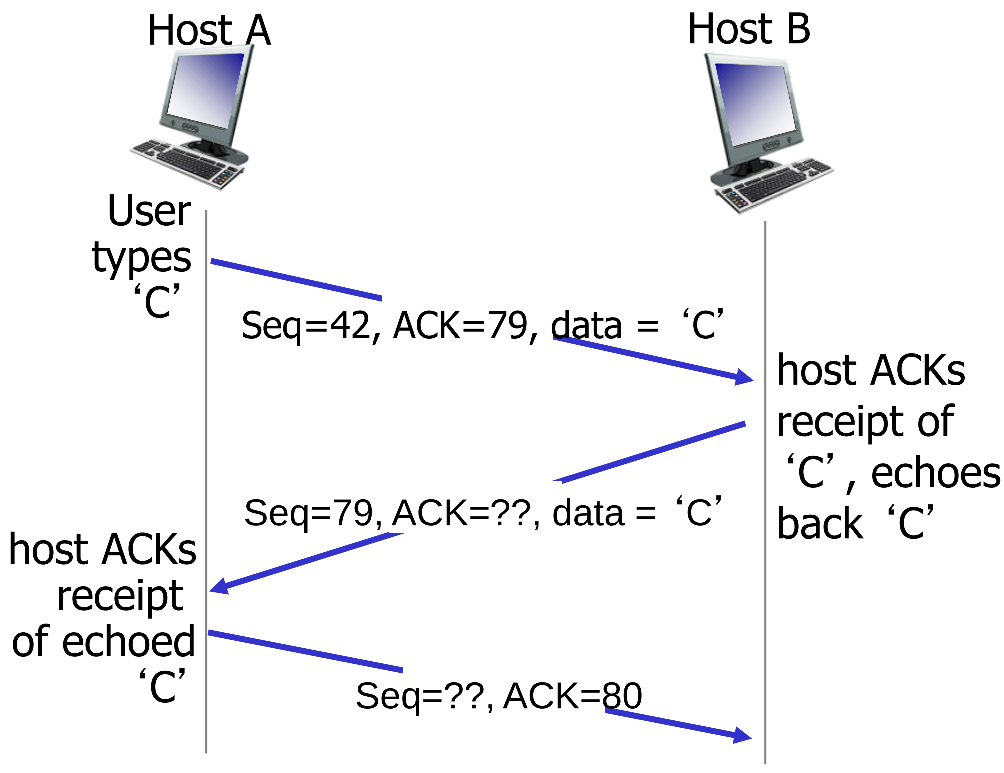
Seg. 4 --> Seq. No. 13,001 Range: 13,001 to 14,000

Seg. 5 --> Seq. No. 14,001 Range: 14,001 to 15,000

TCP seq. #'s and ACKs

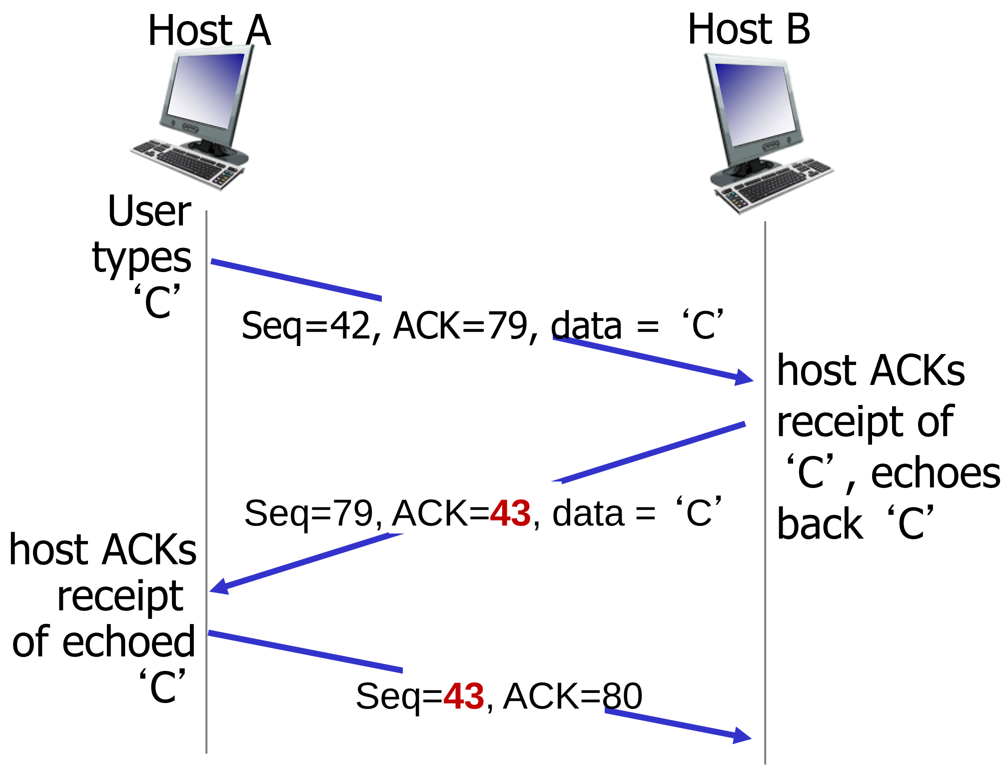


TCP seq. numbers, ACKs



simple telnet scenario

TCP seq. numbers, ACKs



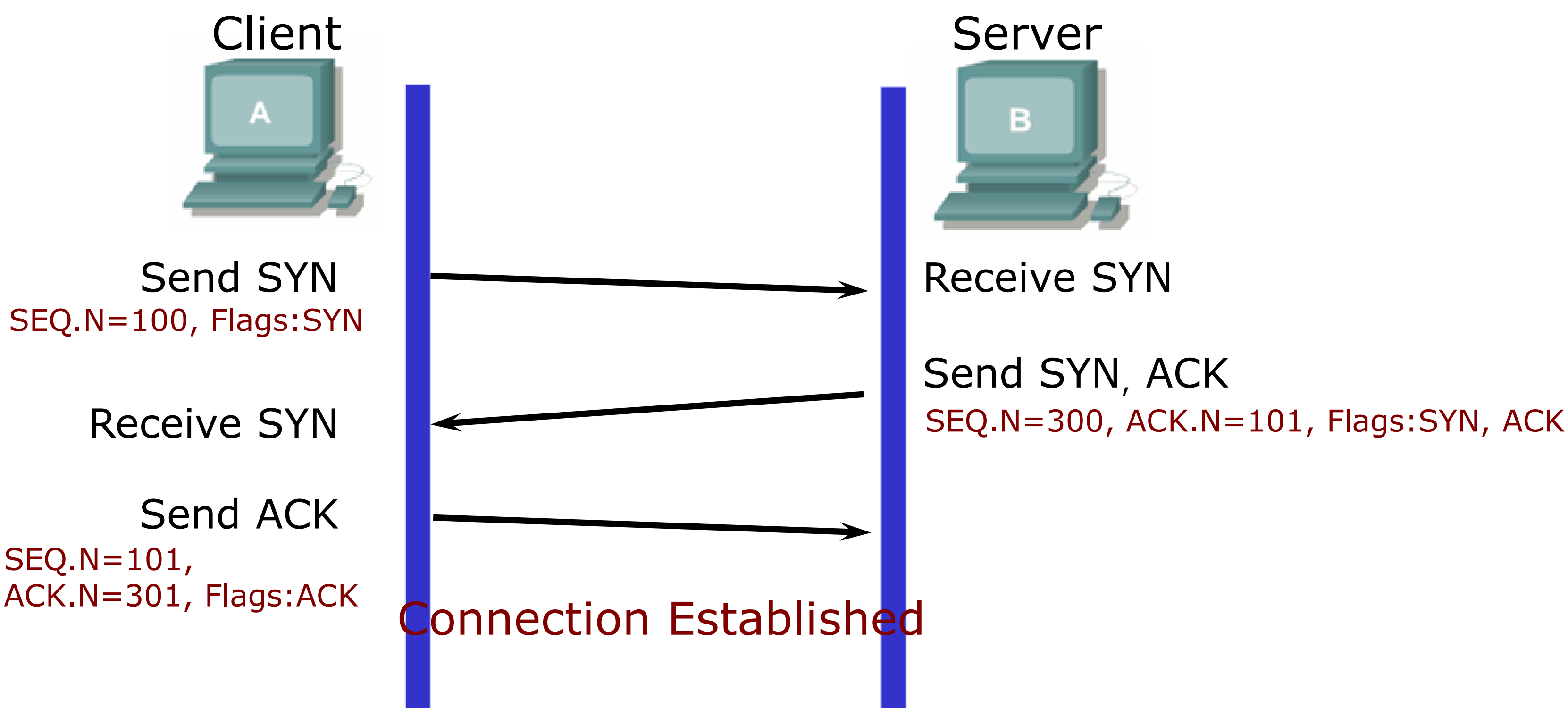
simple telnet scenario

Outline

- Transport Layer Services
- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)
  - Segment structure
  - Connection management
  - Reliable data transfer
  - Flow control
  - Congestion control

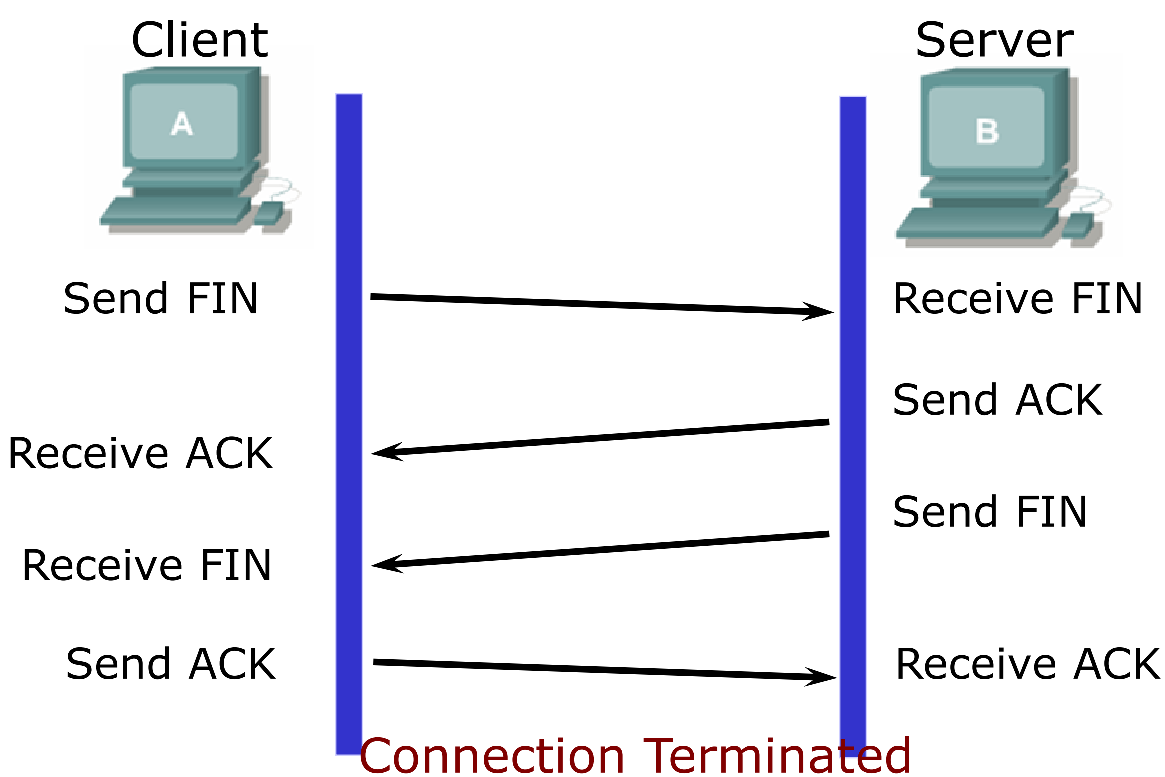
Establishing a TCP Connection

TCP connections are established with a three–way handshake



Gracefully Terminating a TCP Connection

- Either client or server can initiate the termination



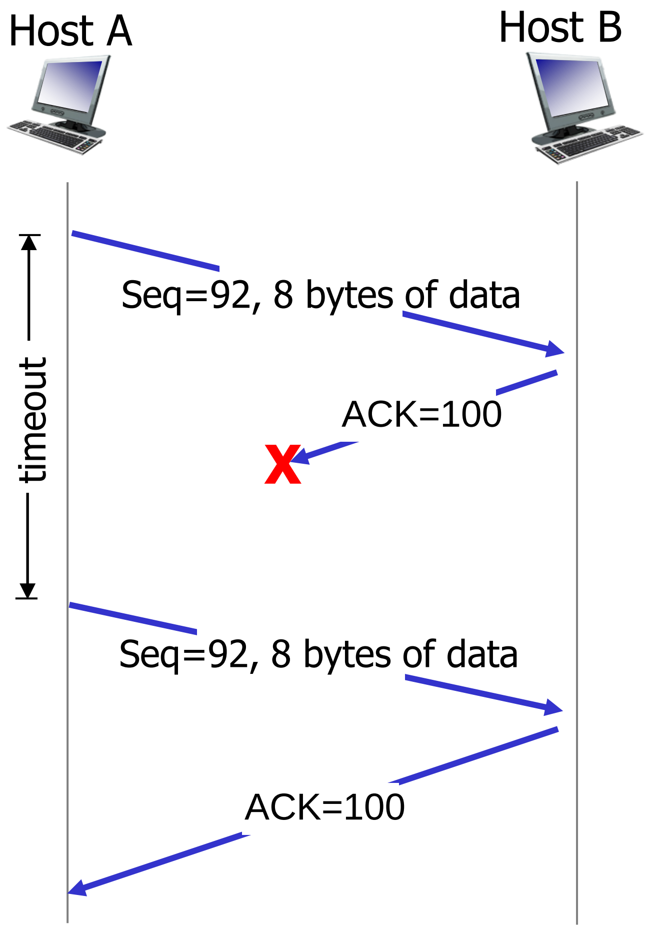
Outline

- Transport Layer Services
- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)
  - Segment structure
  - Connection management
  - Reliable data transfer
  - Flow control
  - Congestion control

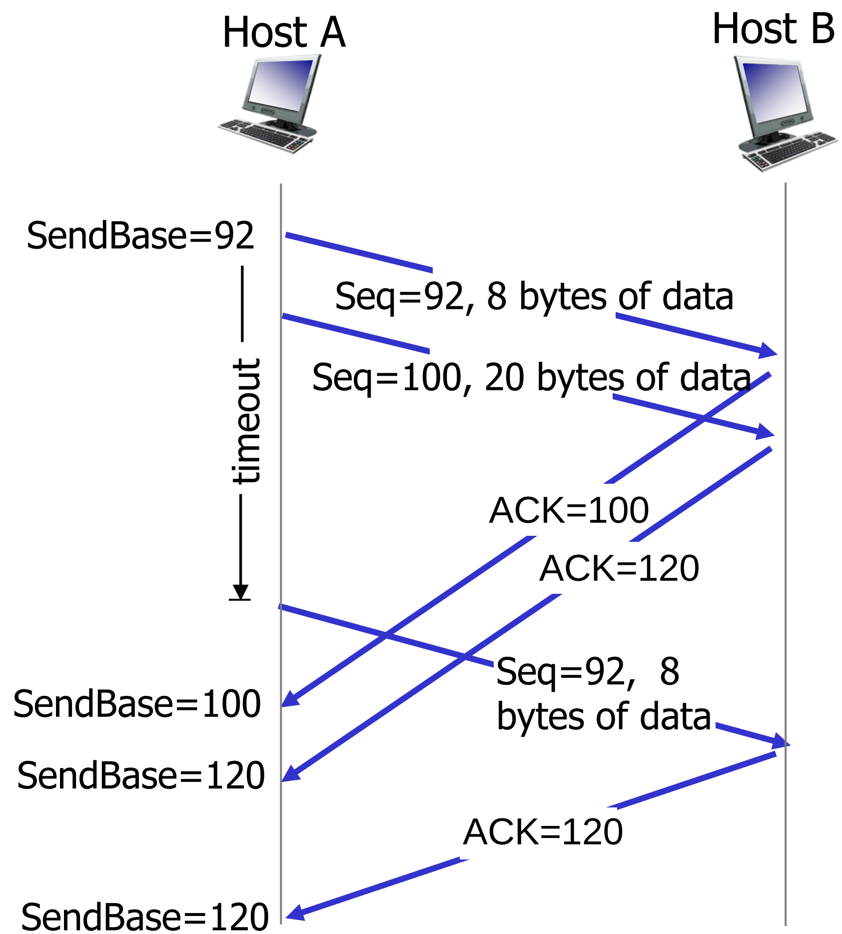
TCP Reliable Data Transfer

- TCP creates reliable data transfer service on top of IP's unreliable service
  - Pipelined segments
  - Cumulative acks
  - Single retransmission timer
- Retransmission triggered by
  - Timeout events
  - Duplicate acks

TCP: retransmission scenarios

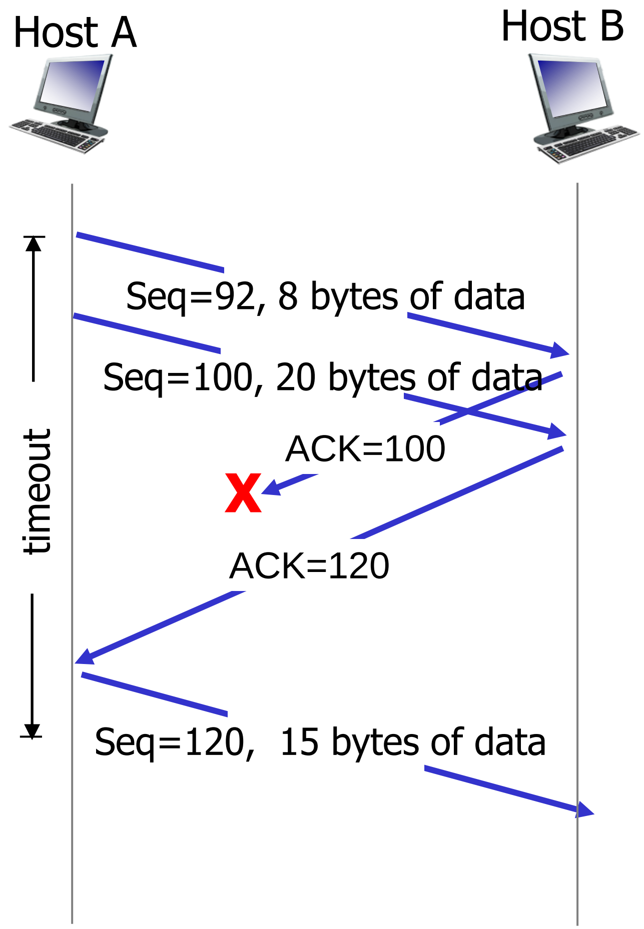


lost ACK scenario



premature timeout

TCP: retransmission scenarios



cumulative ACK

In this scenario, Host A receives an ack. no 120, which means that Host B has received everything up to byte 119. This avoids retransmission of the first segment.

TCP fast retransmit

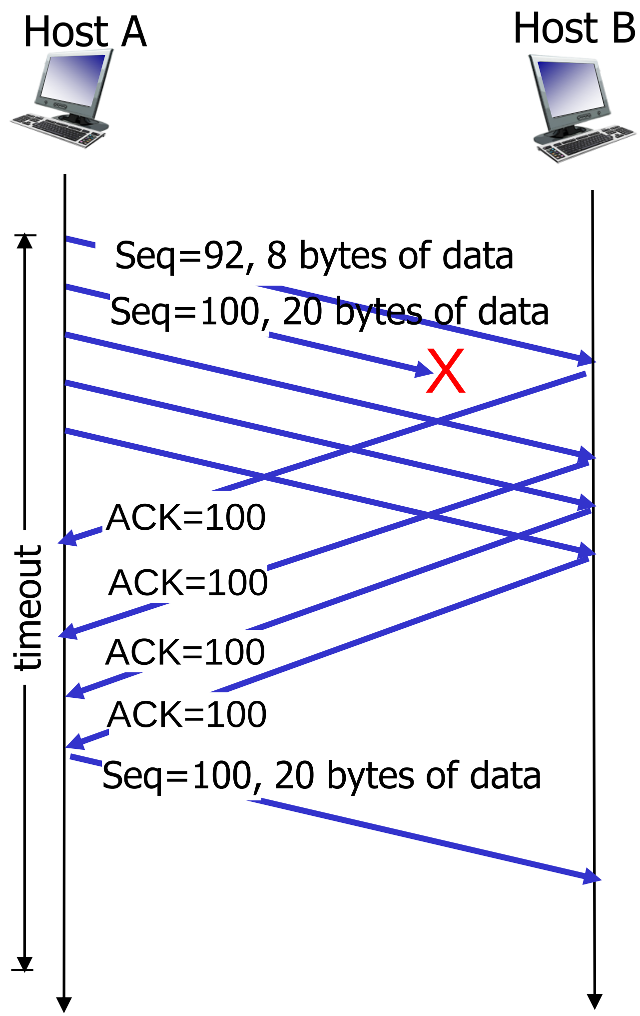
- time-out period often relatively long:
  - long delay before resending lost packet
- detect lost segments via duplicate ACKs.
  - sender often sends many segments back-to-back
  - if segment is lost, there will likely be many duplicate ACKs.

TCP fast retransmit

if sender receives 3 ACKs for same data ("triple duplicate ACKs"), resend unacked segment with smallest seq #

- likely that unacked segment lost, so don't wait for timeout

TCP fast retransmit



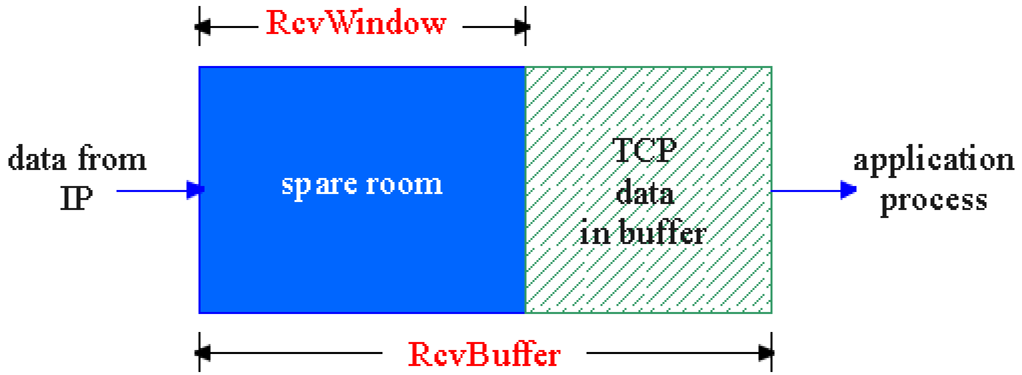
fast retransmit after sender receipt of triple duplicate ACK

Outline

- Transport Layer Services
- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)
  - Segment structure
  - Connection management
  - Reliable data transfer
  - Flow control
  - Congestion control

TCP Flow Control

- receive side of TCP connection has a receive buffer:



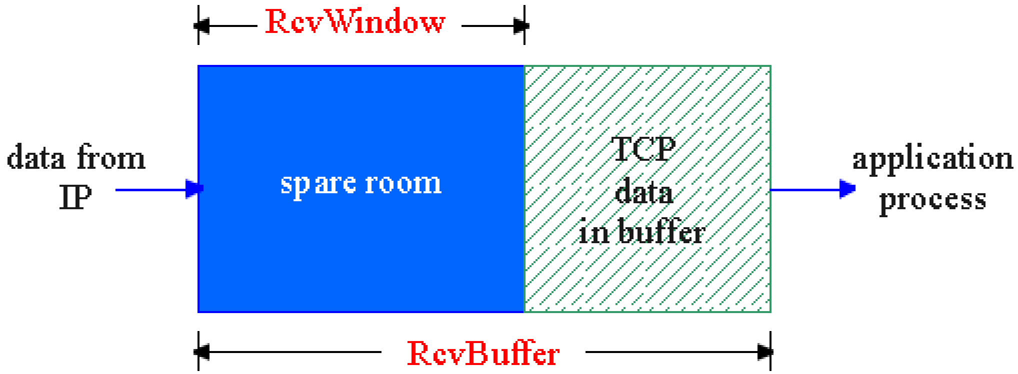
- app process may be slow at reading from buffer

flow control

sender won't overflow receiver's buffer by transmitting too much, too fast

- speed-matching service: matching the send rate to the receiving app's drain rate

TCP Flow control: how it works



- spare room in buffer, RcvWindow

- Rcvr advertises spare room by including value of **`RcvWindow`** in segments
- Sender limits unACKed data to **`RcvWindow`**
  - guarantees receive buffer doesn't overflow

Question: if a TCP sender receives a TCP segment with RcvWindow of zero, what will happen to the sender?

Outline

- Transport Layer Services
- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)
  - Segment structure
  - Connection management
  - Reliable data transfer
  - Flow control
  - Congestion control

Principles of congestion control

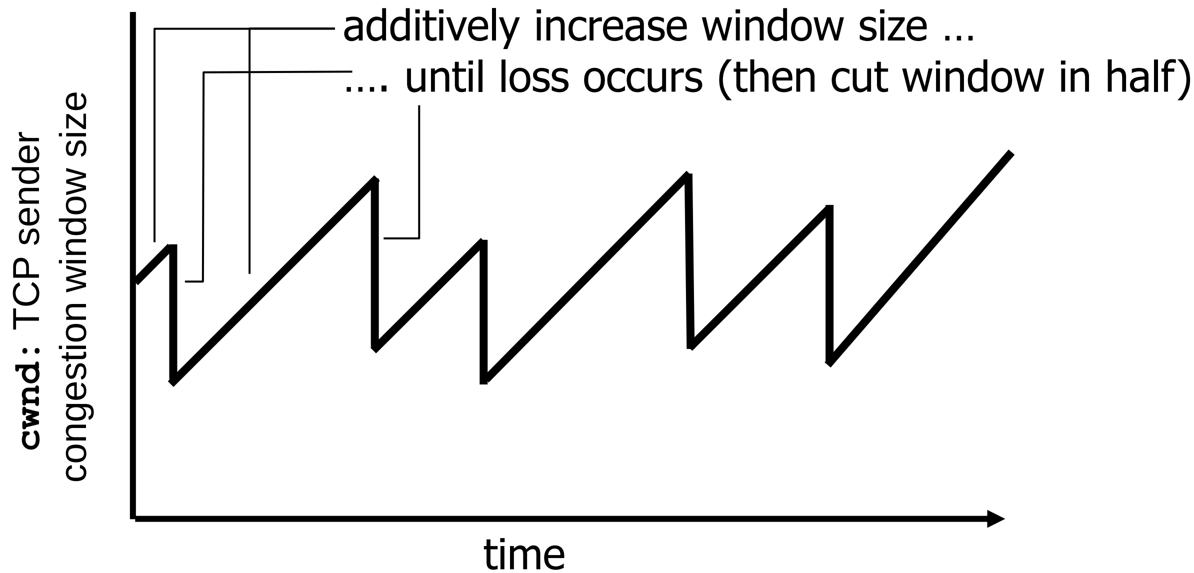
congestion:

- informally: "too many sources sending too much data too fast for *network* to handle"
- different from flow control!
- manifestations:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)
- a top-10 problem in networks!

TCP congestion control: additive increase multiplicative decrease

- *approach:* sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase **`cwnd`** by 1 MSS every RTT until loss detected
  - *multiplicative decrease:* cut **`cwnd`** in half after loss

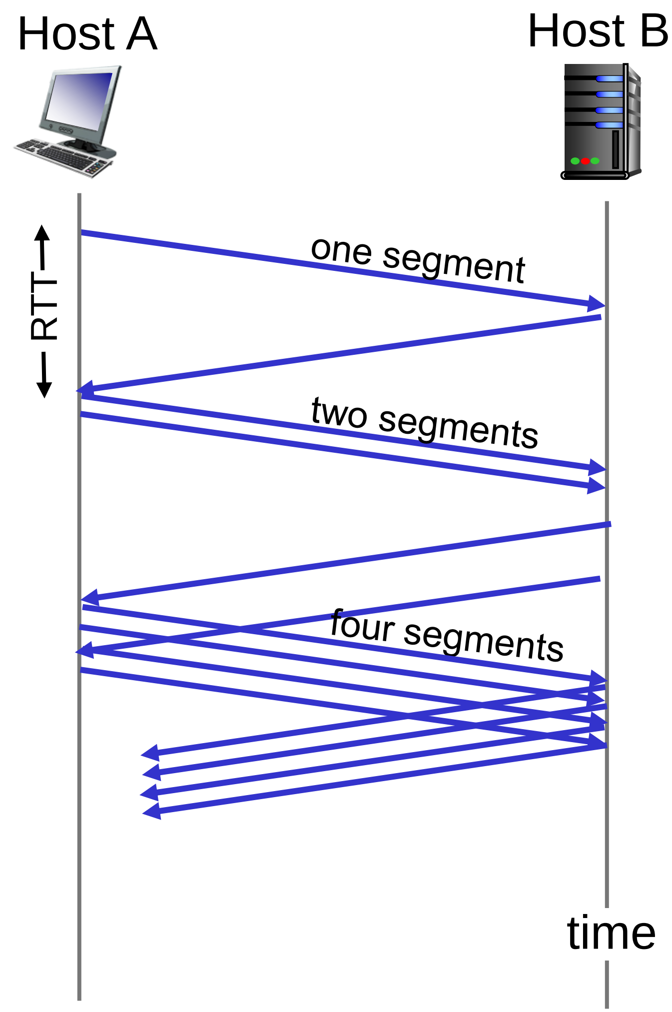
AIMD saw tooth behavior: probing for bandwidth



MSS: maximum segment size
cwnd: congestion window size

TCP Slow Start

- when connection begins, increase rate exponentially until first loss event:
  - initially **cwnd** = 1 MSS
  - double **cwnd** every RTT
  - done by incrementing **cwnd** for every ACK received
- *summary:* initial rate is slow but ramps up exponentially fast
- Aggressive approach



Refinement: inferring loss

- After 3 dup ACKs:
  - **`cwnd`** is cut in half
  - window then grows linearly
- But after timeout event:
  - **`cwnd`** instead set to 1 MSS;
  - window then grows exponentially
  - to a threshold, then grows linearly

Philosophy:

- 3 dup ACKs indicates network capable of delivering some segments
- timeout indicates a "more alarming" congestion scenario

Quiz

Which of the following statements does NOT reflect the characteristics of TCP?

a) Data is broken down into sequenced segments
b) Provides flow control
c) Supports unicast and multicast addresses
d) Each segment is acknowledged and resent if necessary
e) Provides congestion control

Summary

- UDP
  - Unreliable, fast
  - Connectionless
- TCP
  - Reliable: by acknowledgements and retransmission
  - Connection-oriented
  - Flow control and congestion control