The Transport Layer: TCP & Reliable Data Transfer

Smith College, CSC 249
February 15, 2018

Chapter 3: Transport Layer

- TCP Transport layer services:
  - Multiplexing/demultiplexing
  - Connection management
  - Reliable data transfer
    - SEQ and ACK numbers
  - Congestion control
  - Flow control

TCP Connection Management: Set up

Recall: TCP senders and receivers establish a “connection” before exchanging data segments

Three way handshake:

Step 1: client host sends TCP SYN segment to server

- “SYN” for “synchronize” (set SYN bit to 1)
- Specifies (random) initial sequence #
- No data is sent

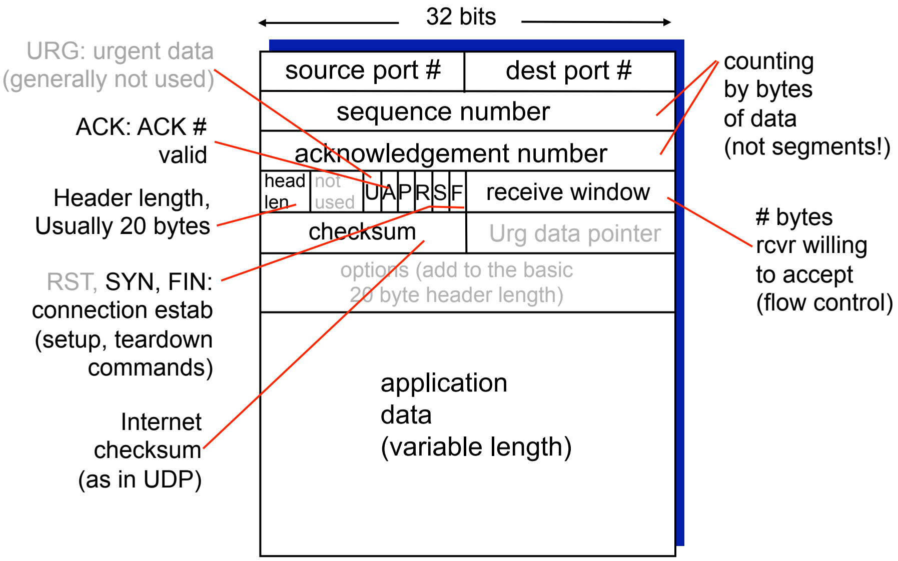
Step 2: server host receives SYN, replies with SYNACK segment

- Set both SYN and ACK bits to 1
- Server allocates buffers and variables
- Specifies its own, server initial sequence #

Step 3: client receives SYNACK, replies with ACK segment

- Client allocates buffers and variables
- This packet may contain data

TCP segment structure



32 bits

URG: urgent data (generally not used)

ACK: ACK # valid

Header length, Usually 20 bytes

RST, SYN, FIN: connection estab (setup, teardown commands)

Internet checksum (as in UDP)

source port # | dest port #

sequence number

acknowledgement number

head len | not used | U | A | P | R | S | F | receive window

checksum | Urg data pointer

options (add to the basic 20 byte header length)

application data (variable length)

counting by bytes of data (not segments!)

# bytes rcvr willing to accept (flow control)

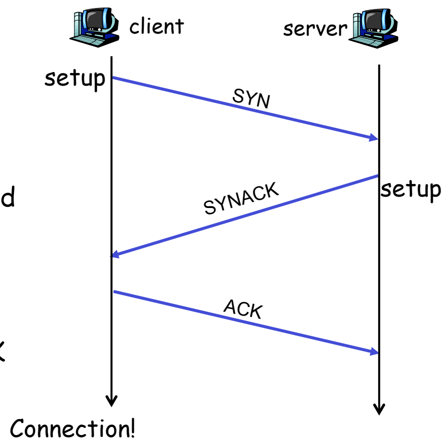
TCP Connection Management: Set up

Set Up:

Step 1: client sends TCP SYN segment
- ❑ Actions at self?
- ❑ Sends data?

Step 2: server receives SYN and replies with SYNACK
- ❑ Actions at self?
- ❑ Sends data?

Step 3: client receives SYNACK and replies with ACK
- ❑ Actions at self?
- ❑ Sends data?

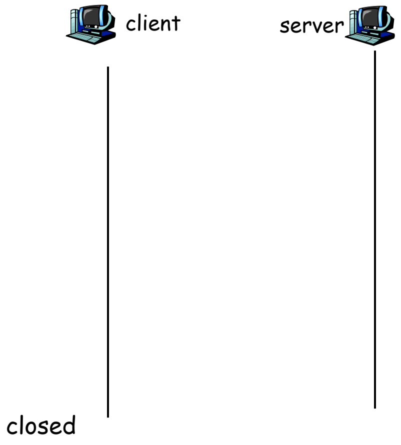


TCP Connection Management: Close

Closing a connection:

How many steps?

What are they?

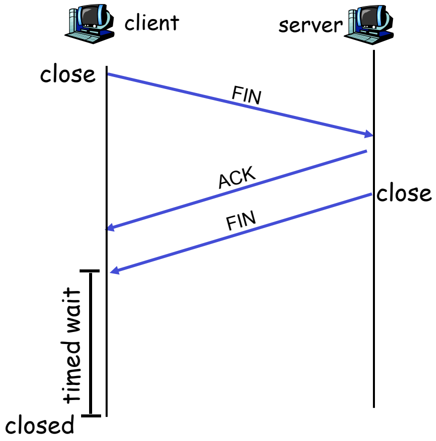


TCP Connection Management: Close

Closing a connection:

Step 1: client end system sends TCP FIN control segment to server

Step 2: server receives FIN, replies with ACK. Closes connection, sends FIN.

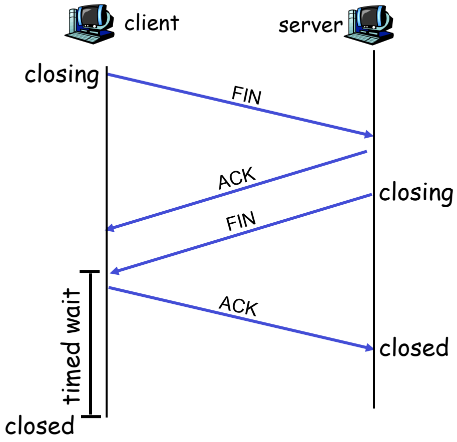


TCP Connection Management (cont.)

Step 3: client receives FIN, replies with ACK.

- Enters “timed wait” - client able to resend final ACK in case it is lost
- Why??

Step 4: server, receives ACK. Connection closed.

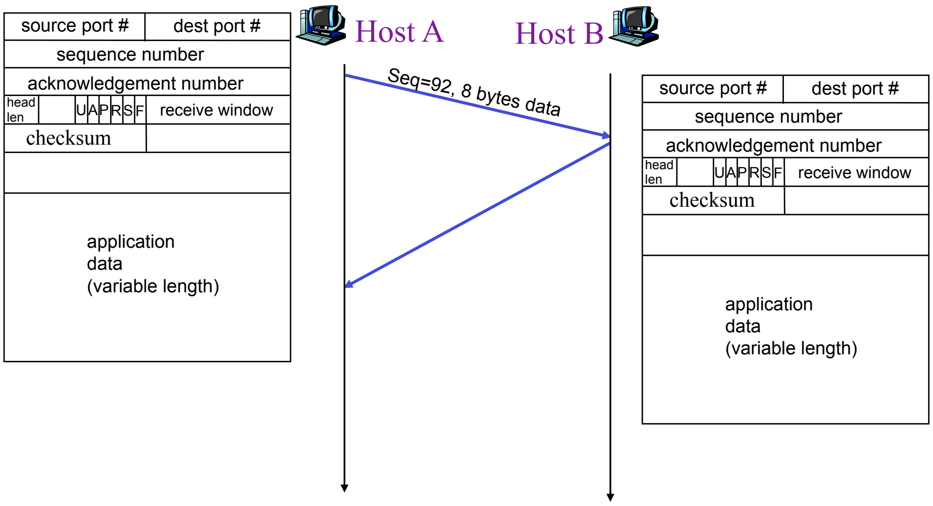


TCP possible sender events:

(1) Data received from application:

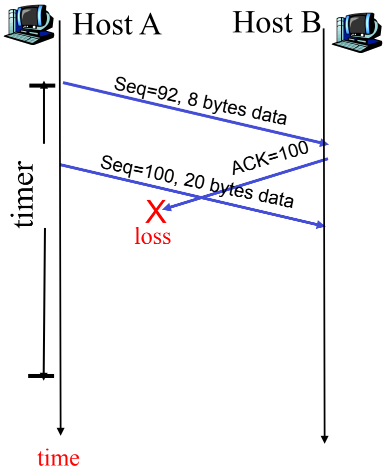
1. Create a segment and assign a SEQ number
   - SEQ # is byte-stream number of first data byte in segment
2. Start timer if it is not already running
   - Timer is for the oldest un-acked segment
   - Expiration interval: `TimeOutInterval`

TCP: SEQ and ACK numbers



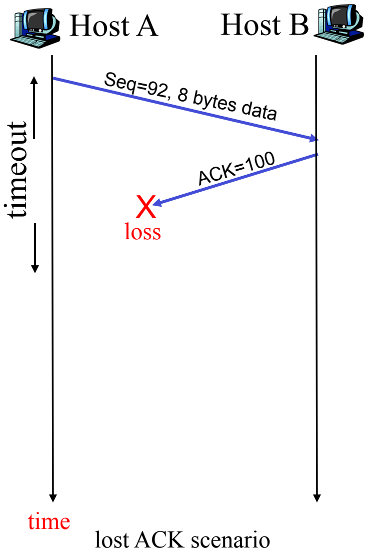
time

TCP: Cumulative ACK



Cumulative ACK scenario

TCP: retransmission from timeout



lost ACK scenario

TCP possible sender events:

(1) Data received from application:

1. Create a segment and assign a SEQ number
   - SEQ # is byte-stream number of first data byte in segment
2. Start timer if it is not already running
   - Timer is for the oldest un-acked segment
   - Expiration interval: `TimeOutInterval`

(2) Timeout (ACK not received):

1. Retransmit segment that caused the timeout
2. Restart the timer

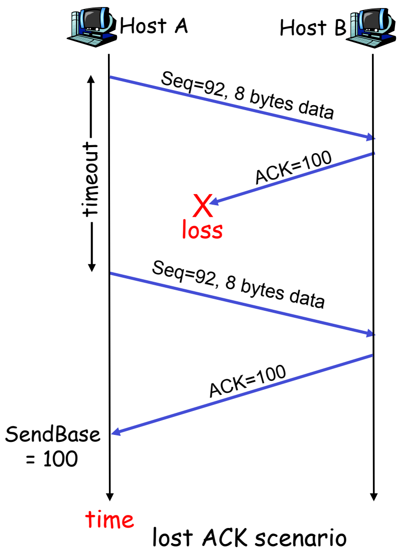
(3) ACK received for previously unacked segments

1. Update what is known to be acked
2. Start timer if there are outstanding segments

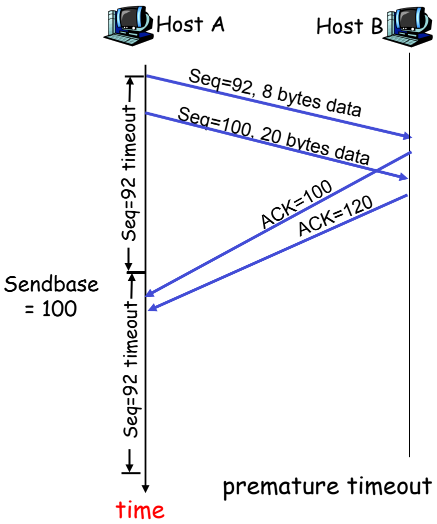
Students' as Transport Layer…

- Working in pairs, send Haiku to each other
  - Let each letter and space represent one byte
- Using blank TCP segments
  - Define and use ACK and SEQ numbers for sending the segments
- Once we see the time involved for our person-transport-layer, define an amount of time for a timer, and have some timeout events
- Communicate with your sending/receiving pairs to be able to dramatize TCP successfully

TCP: retransmission scenarios



lost ACK scenario

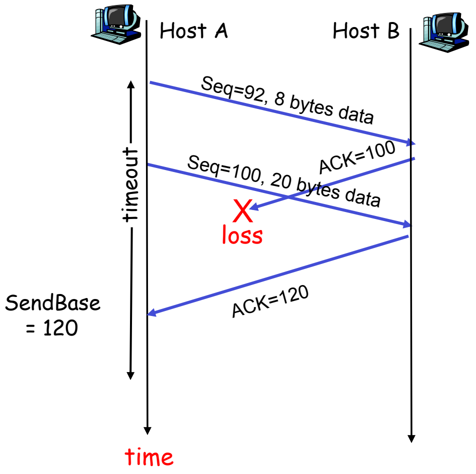


premature timeout

1) What is/was 'A's next step?
2) What does 'B' then do?

15

TCP retransmission scenarios (more)

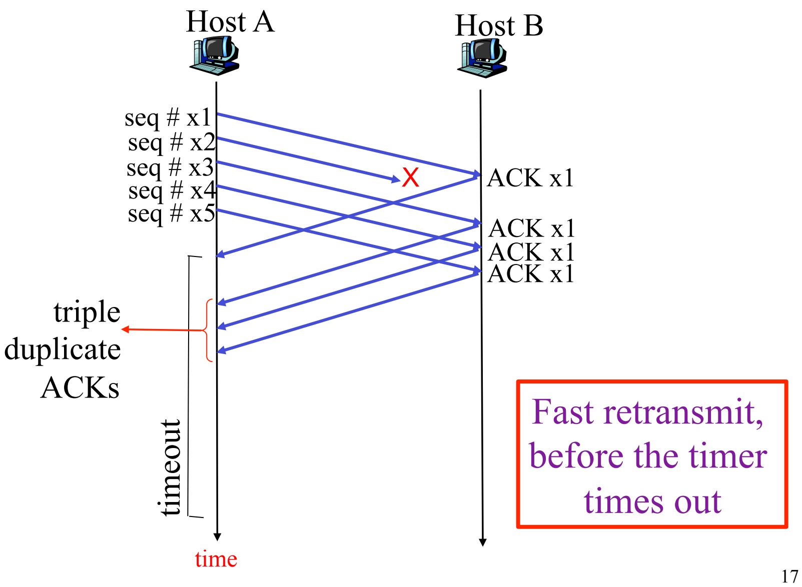


Cumulative ACK scenario

What does 'A' do next?

16

What does ‘A’ do next, and when does it do it?



Fast retransmit, before the timer times out

Discussion Question 1

- Suppose Host A sends two TCP segments back to back to Host B over a TCP connection.
  - What might be the first sequence number?
  - If 20 bytes are sent, what is the second sequence number?
  - Suppose that the first segment is lost but the second segment arrives at B. In the acknowledgment that Host B sends to Host A, what will be the acknowledgment number?

Discussion Question 2

- Consider a reliable protocol that uses only NAKs (no unnecessary ACKs, since most often, things work well!) Suppose the sender sends data infrequently. Would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

Discussion Question 3

- Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

Discussion Question 4: Why Wait for 3 Duplicate ACKs before retransmission?

- Why did the TCP designers choose to have TCP wait until it has received **three** duplicate ACKs before performing a fast retransmit, rather than performing a fast retransmit after the first duplicate ACK for a segment is received?

Discussion Question 4: Why Wait for 3 Duplicate ACKs?

- Suppose packets n, n+1, and n+2 are sent, and that packet n is received and ACKed.
- 2-duplicate ACK policy: If packets n+1 and n+2 are reordered along the end-to-end-path then the receipt of packet n+2 will generate a duplicate ACK for n and would trigger a retransmission.
- 3 duplicate ACK scheme: Trades-off waiting for more packets (rather than just 1) to avoid retransmitting prematurely in the face of packet reordering.
  - This policy could slow things down of course if packet n+1 is lost rather than reordered.

Summary

- Multiplexing and demultiplexing
- Error checking - checksum
- Connection Management
  - SYN and SYNACK packets
- TCP segment header format
- Reliable data transfer
  - Sequence and acknowledgement numbers
  - Discuss use of “NAK”
  - Discuss the purpose of 3 duplicate ACKs