Chapter 3 (Final Study Guide)

TCP RELIABLE DATA TRANSFER

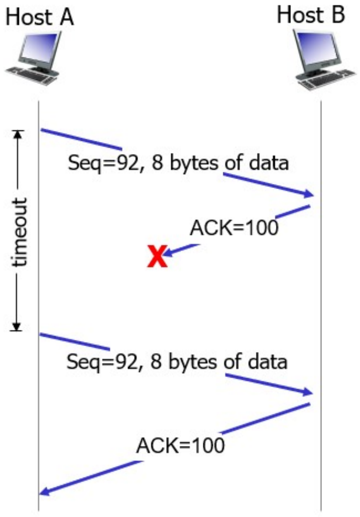
TCP creates rdt service on top of IP's unreliable service

- pipelined segments
- cumulative acks
- single retransmission timer

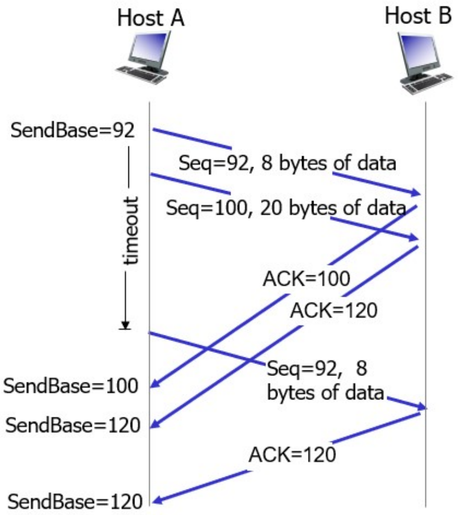
retransmissions triggered by:

- timeout events
- duplicate acks

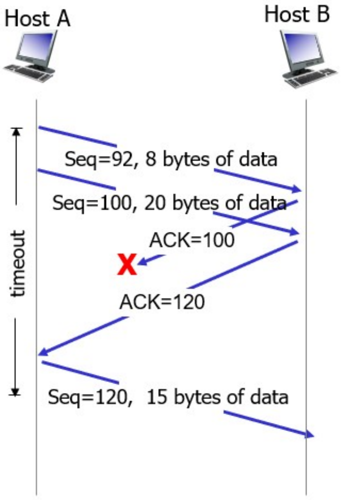
3 TCP Retransmission Scenarios:



lost ACK scenario



premature timeout

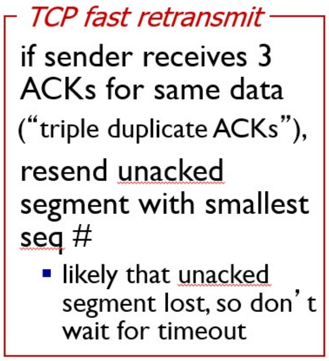


cumulative ACK

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 #

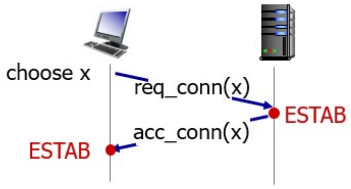
- likely that unacked segment lost, so don't wait for timeout

TCP CONTROL FLOW (Flow Control term important)

Flow Control: receiver controls sender, so sender won't overflow receiver's buffer by transmitting too much, too fast

2-WAY HAND SHAKE

Will 2-way handshake always work in network?
  -variable delays
  -retransmitted messages (e.g. req_conn(x)) due to message loss
  -message reordering
  -can't "see" other side



PRINCIPLES OF CONGESTION (Congestion term important)

Congestion
- informally: "too many sources sending too much data too fast for network to handle"
- different from flow control!
  -Flow control is a speed matching service: matching the rate at which sender is sending against the rate at which the receiving application is reading

***CAUSES/COSTS OF CONGESTION: SCENARIO 1* (3-79 -- 3-86)**

Features:
- two senders, two receivers

- one router, infinite buffers
- no retransmission

Congestion:
- maximum per-connection throughput: R/2
- large delays as arrival rate, l(lambda)in, approaches capacity

CAUSES/COSTS OF CONGESTION: SCENARIO 2

Features:
- one router, finite buffers
- sender retransmission of timed-out packet
    - -application-layer input = application-layer output: l(lambda)in = l(lambda)out
    - -transport-layer input includes retransmissions : l(lambda)in >= l(lambda)in

Congestion:
- Idealization: known loss packets can be lost, dropped at router due to full buffers
- sender only resends if packet known to be lost

Congestion:
- Realistic: duplicates
    - -packets can be lost, dropped at router due to full buffers
    - -sender times out prematurely, sending two copies, both of which are delivered

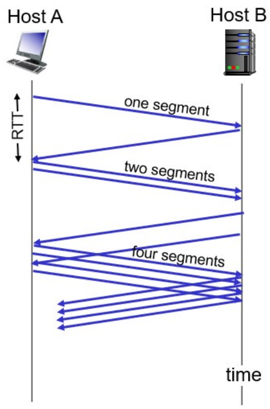
CAUSES/COSTS OF CONGESTION: SCENARIO 3

Features:
- four senders
- multihop paths
- timeout/retransmit

Congestion:
- another “cost” of congestion:
- -when packet dropped, any “upstream transmission capacity used for that packet was wasted

TCP SLOW START

summary: initial rate is slow but ramps up exponentially fast



SEGMENT LOST (Important key terms)

Segment loss detected by a triple duplicate ACK, is set to 1 >set MSS (Congestion window size)

Segment loss detected by a timeout, is set 1 MSS (Congestion window size)

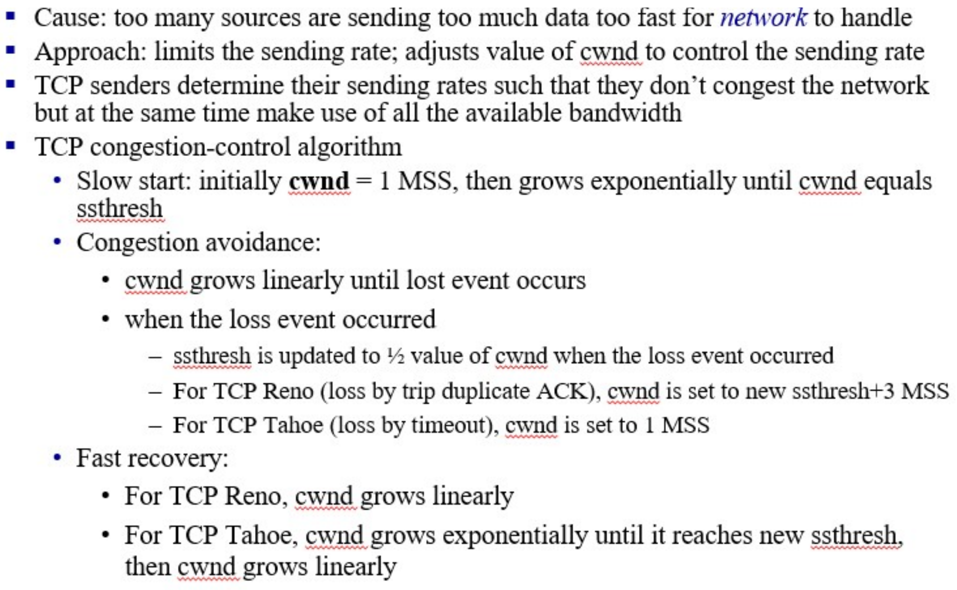
TCP: DETECTING, REACTING TO LOSS

TCP Reno & TCP Tahoe - Window grows exponentially to threshold

TCP Reno - loss indicated by 3 duplicate ACKS (cwnd is cut in half then grows linearly)

TCP Tahoe - Always sets cwnd to 1 when timeout or 3 duplicate acks

TCP CONGESTION CONTROL REVIEW



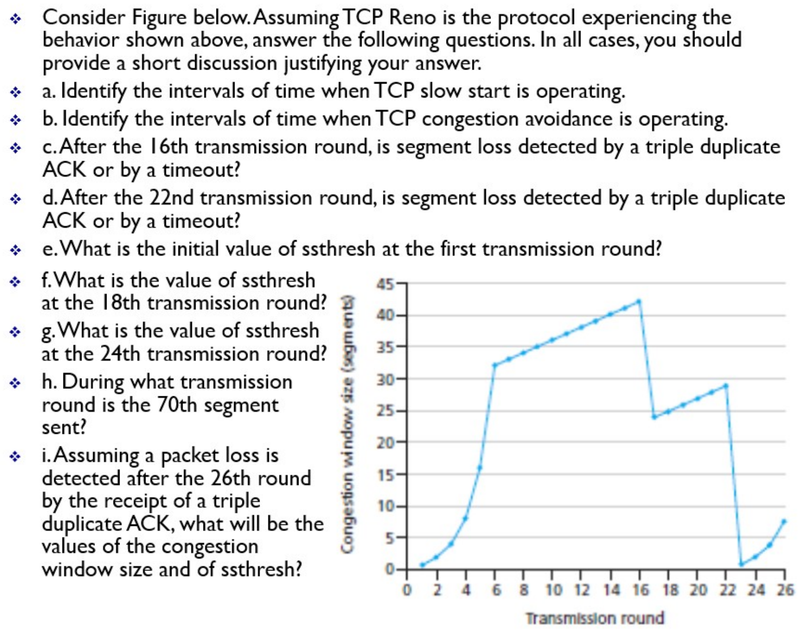
- Cause: too many sources are sending too much data too fast for *network* to handle
- Approach: limits the sending rate; adjusts value of cwnd to control the sending rate
- TCP senders determine their sending rates such that they don't congest the network but at the same time make use of all the available bandwidth
- TCP congestion-control algorithm
  - Slow start: initially **cwnd** = 1 MSS, then grows exponentially until cwnd equals ssthresh
  - Congestion avoidance:
    - cwnd grows linearly until lost event occurs
    - when the loss event occurred
      - ssthresh is updated to ½ value of cwnd when the loss event occurred
      - For TCP Reno (loss by trip duplicate ACK), cwnd is set to new ssthresh+3 MSS
      - For TCP Tahoe (loss by timeout), cwnd is set to 1 MSS
  - Fast recovery:
    - For TCP Reno, cwnd grows linearly
    - For TCP Tahoe, cwnd grows exponentially until it reaches new ssthresh, then cwnd grows linearly

EXPLICIT CONGESTION NOTIFICATION

- two bits in IP header (ToS field) marked by network router to indicate congestion

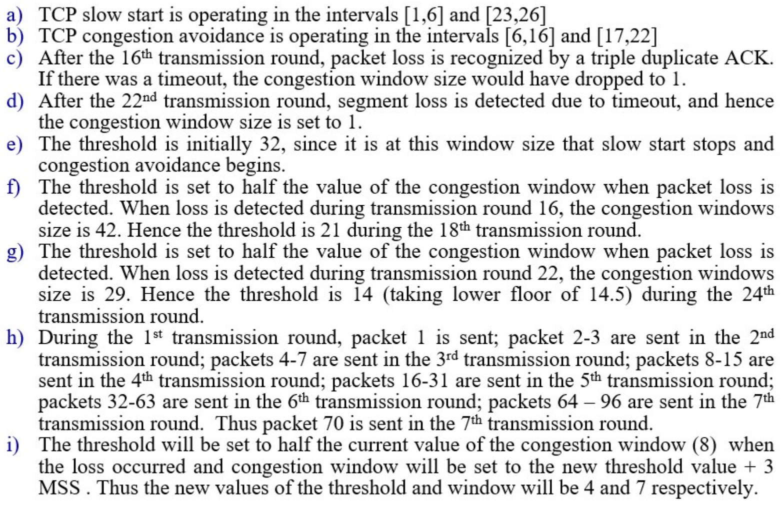
EXAM QUESTION

Question:



- Consider Figure below. Assuming TCP Reno is the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying your answer.
- a. Identify the intervals of time when TCP slow start is operating.
- b. Identify the intervals of time when TCP congestion avoidance is operating.
- c. After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
- d. After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
- e. What is the initial value of ssthresh at the first transmission round?
- f. What is the value of ssthresh at the 18th transmission round?
- g. What is the value of ssthresh at the 24th transmission round?
- h. During what transmission round is the 70th segment sent?
- i. Assuming a packet loss is detected after the 26th round by the receipt of a triple duplicate ACK, what will be the values of the congestion window size and of ssthresh?

Answer:



a) TCP slow start is operating in the intervals [1,6] and [23,26]

b) TCP congestion avoidance is operating in the intervals [6,16] and [17,22]

c) After the 16th transmission round, packet loss is recognized by a triple duplicate ACK. If there was a timeout, the congestion window size would have dropped to 1.

d) After the 22nd transmission round, segment loss is detected due to timeout, and hence the congestion window size is set to 1.

e) The threshold is initially 32, since it is at this window size that slow start stops and congestion avoidance begins.

f) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 16, the congestion windows size is 42. Hence the threshold is 21 during the 18th transmission round.

g) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 22, the congestion windows size is 29. Hence the threshold is 14 (taking lower floor of 14.5) during the 24th transmission round.

h) During the 1st transmission round, packet 1 is sent; packet 2-3 are sent in the 2nd transmission round; packets 4-7 are sent in the 3rd transmission round; packets 8-15 are sent in the 4th transmission round; packets 16-31 are sent in the 5th transmission round; packets 32-63 are sent in the 6th transmission round; packets 64 – 96 are sent in the 7th transmission round. Thus packet 70 is sent in the 7th transmission round.

i) The threshold will be set to half the current value of the congestion window (8) when the loss occurred and congestion window will be set to the new threshold value + 3 MSS . Thus the new values of the threshold and window will be 4 and 7 respectively.

My notes for Final Exam Question

5. 3-91 (final exam question)

a.

Exponential growing graphs

1-6 seconds

23-26

cwnd(congestion window)

b. Congestion windows size grows linearly

6-17seconds

17-23 seconds

c.

Answer: triple duplicate ack

d.
It's been detcted my timeout (reaches 0 after it drops)
Answer: timeout
e.
Slow start ends when the ssthresh starts
Answer: 32
f.
The congestion window is equal to 21. Set the congestion window to half. (The first tip of the graph / 2)
Answer: 21
g.
29/2 (The second TIP of the graph)
Answer: 14
h.

Packet		Segment
1	1	1
2	4	2-3
3	8	4-7
4	16	8-15
5	25	16-31
6	36	32-63
7	33	64-96
8	34	

i.
it is seven equal to new thresshold value + 3
Answer: 7