Tutorial problems for TCP

Review Questions:

1. In TCP, why did we need to introduce sequence numbers?

Sequence numbers are required for a receiver to find out whether an arriving packet contains new data or is a retransmission.

2. In TCP, why did we need to introduce timers?

To handle losses in the channel, if the ACK for a transmitted packet is not received within the duration of the timer for the packet, the packet (or its ACK) is assumed to have been lost. Hence, the packet is retransmitted.

3. Suppose that the round trip time between the sender and the receiver is constant and known to the sender. Would a timer still be necessary in TCP, assuming that packets can be lost? Explain.

A timer would still be necessary in TCP. If the round trip time is known then the only advantage will be that, the sender knows for sure that either the packet or the ACK for the packet has been lost, as compared to the real scenario, where the ACK might still be on the way to the sender, after the timer expires. However, to detect the loss, for each packet, a timer of constant duration will still be necessary at the sender.

4. How does Fast Retransmit work? What is the basic idea behind it?

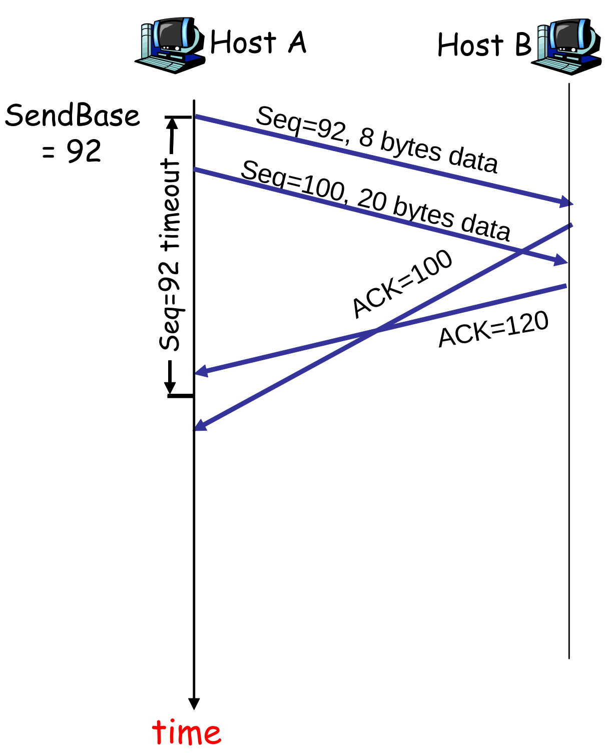
If sender receives 3 duplicate ACKs (totally 4 same ACks) for same data, it assumes that the segment after ACKed data was lost and the sender will resend the segment (before timer expires). The basic idea behind Fast Retransmit is that the sender often sends many segments back-to-back and if a segment is lost, there will likely be many duplicate ACKs for that segment. Therefore, if the sender receives a number of duplicate ACKs for the same segment, it is very likely that the segment is lost.

5. Why is a various-size (credit) sliding window scheme better than a fixed-size sliding window scheme for TCP?

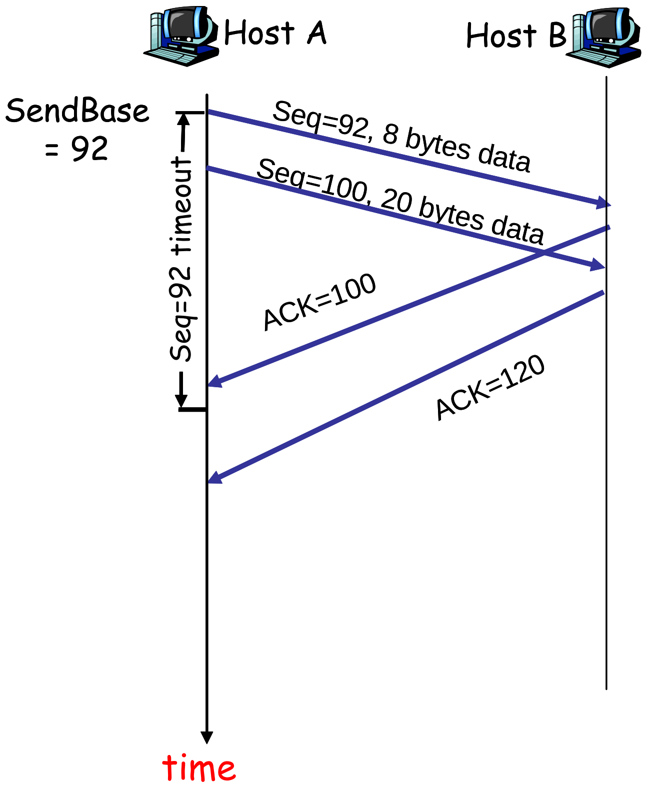
In a fixed scheme, ACK and flow control are synonymous, i.e receiving ACK automatically means the grant of sending more data even though the ACKed data may be still in the buffer of the receiving transport entity. In a credit scheme, ACK and flow control are decoupled so the above problem can be solved.

Problems

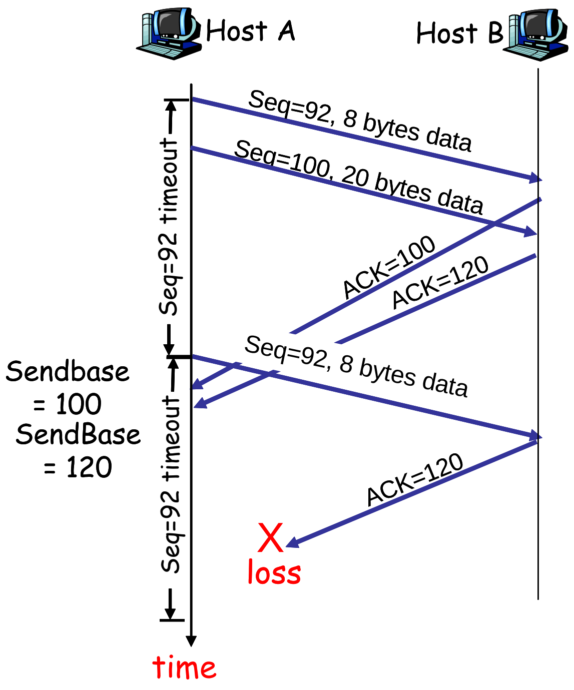
1. Describe what will happen after Host A timeouts in the following four situations.



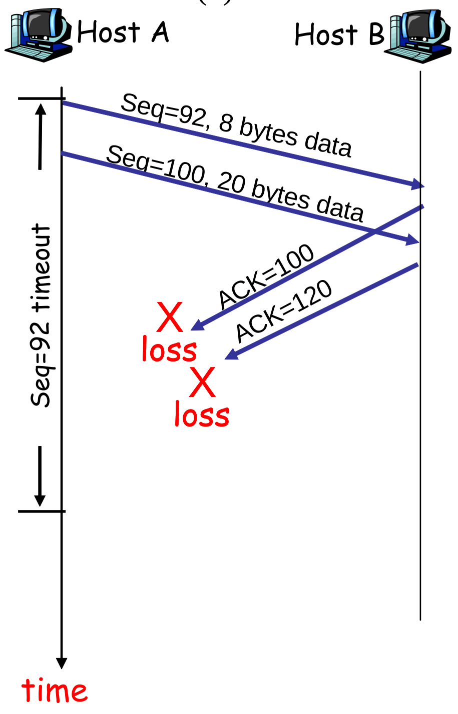
(a)



(b)



(c)



(d)

In (a), (b) and (c), Host A will not retransmit segment after timeout while in (d) Host A will retransmit the segments with sequence number 92 after timeout.

2. a) With reference to Fig. Q2 showing TCP disconnection procedure, when TCP receives a FIN from the other TCP, TCP needs to go through two wait states (CLOSE WAIT and LAST ACK) before closing the connection. Please state the action after TCP receives the FIN and the rationale behind each wait state.
b) Why does the TCP at Site 2 not to send ACK x+1 and "FIN seq=y, ACK x+1" at the same time. What will be the problem if they are sent out at the same time either by sending out "ACK" later or by sending "FIN, ACK" earlier?

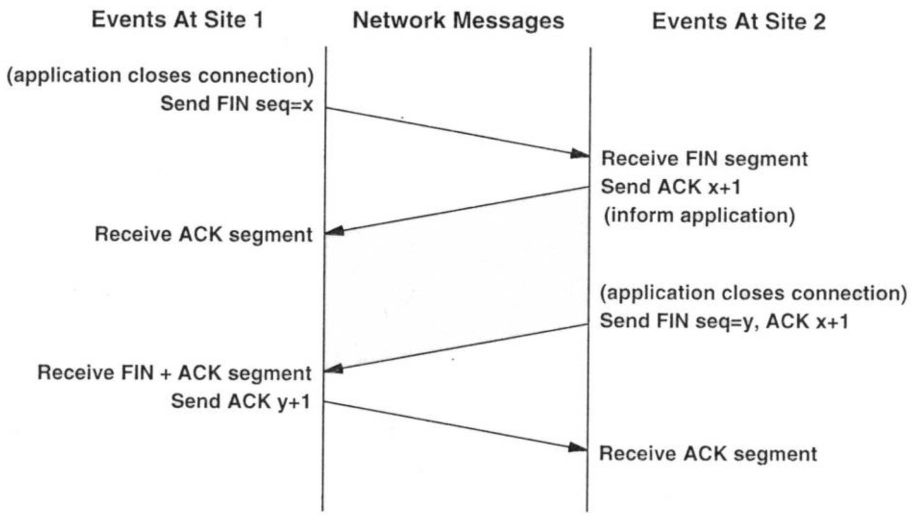


Figure Q2: TCP Connection Termination

a) <u>The Action After TCP Receives The FIN</u>: TCP will send the ACK for that FIN. This assures that the other TCP knows that its FIN has been received. Also, TCP informs its application user about receiving the FIN from the other TCP.
<u>CLOSE WAIT</u>: TCP waits for the CLOSE signal from its application and then send out a FIN to the other TCP. This assures that the other TCP knows that it have sent out all its segments.
<u>LAST ACK</u>: TCP waits for receiving ACK to its FIN from the other TCP. It assures that the FIN has been received by the other TCP.

b) When TCP wants to ACK to the FIN from the other side, its application may still have data to send. If TCP sends out "ACK" later, the other side may unnecessarily resend the FIN and waste bandwidth resource. On the other hand, if TCP sends out "FIN, ACK" earlier, the TCP of this side (or the other side) may close its connection before the application of this side finishes sending out all its data.

3. One difficulty with the original TCP EA-RTT estimator (Exponential Average Round-Trip Time estimator) is the choice of an initial value. In the absence of any special

knowledge of network conditions, the typical approach is to pick an arbitrary value, such as 3 seconds, and hope that this will converge quickly to an accurate value. If this estimate is too small, TCP will perform unnecessary retransmissions. If it is too large, TCP will wait a long time before retransmitting if the first segment is lost. Also, the convergence may be slow, as this problem indicates. Note that

$$\text{EA-RTT}(K + 1) = \alpha \times \text{EA-RTT}(K) + (1 - \alpha) \times \text{RTT}(K).$$

a) Choose $\alpha = 0.90$ and EA-RTT(0) = 1 seconds, and assume all measured RTT values = 4 second and no packet loss. What is EA-RTT(20)? *Hint:* The equation for calculating EA-RTT can be rewritten to simplify the calculation, using the equation $(1 + \ldots + \alpha^{n-2} + \alpha^{n-1}) = (1 - \alpha^{n})/(1 - \alpha)$.

b) Now let $\alpha = 0.25$ and EA-RTT(0) = 4 second and assume measured RTT values = 1 seconds and no packet loss. What is EA-RTT(20)?

a) $\text{EA-RTT}(n) = \alpha^{n}\,\text{EA-RTT}(0) + (1 - \alpha)\ \text{RTT}\ (\alpha^{n-1} + \alpha^{n-2} + \ldots + 1)$

$\qquad = \alpha^{n}\,\text{EA-RTT}(0) + \ \text{RTT}\ (1 - \alpha^{n})$

=> EA-RTT(20) = 3.64 sec

b) EA-RTT(20) = 1.00 sec.

4. A TCP entity opens a connection and uses slow start. Approximately how many round-trip times are required before TCP can send *N* segments at one round?

TCP initializes the congestion window to 1, sends an initial segment, and waits. When the ACK arrives, it increases the congestion window to 2, sends 2 segments, and waits. When the 2 ACKs arrive, they each increase the congestion window by one, so that it can send 4 segments. In general, it takes $\log_2 N$ round trips before TCP can send N segments.

5. Consider the following plot of TCP window as a function of time.

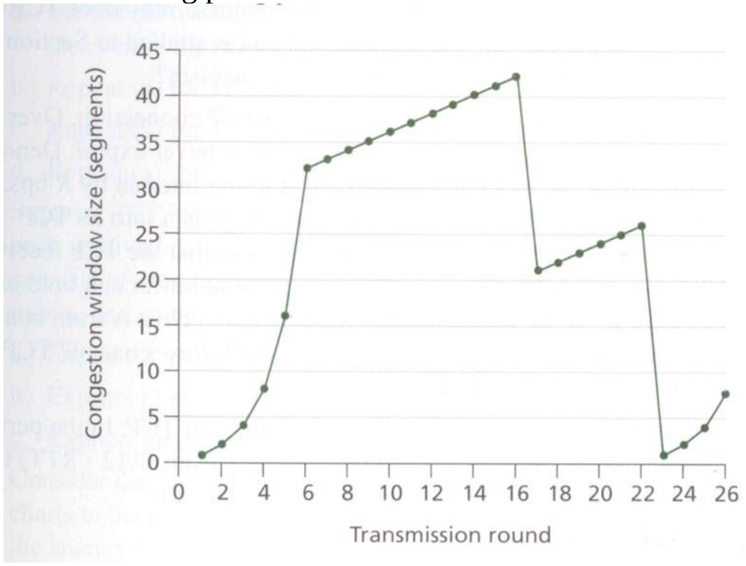


Figure Q3: TCP congestion control

Assuming TCP Reno (i.e. Slow Start + Congestion Avoidance + Fast Retransmit + Fast Recovery) is the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying the 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 Threshold at the first transmission round?
f) What is the value of Threshold at the 18th transmission round?
g) What is the value of Threshold 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 Threshold?

a) TCP slowstart is operating in the intervals [1,5] and [23,26]: double the previous window size
b) TCP congestion avoidance is operating in the intervals [6,16] and [17,22]: linearly increase the window size
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 slowstart 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 26. Hence the threshold is 13 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 congestion window and threshold will be set to half the current value of the congestion window (8) when the loss occurred. Thus the new values of the threshold and window will be 4.

6. In this problem we consider the delay introduced by the TCP slow-start phase. Consider a client and a Web server directly connected by one link of rate R. Suppose the client wants to retrieve an object whose size is exactly equal to 15 S, where S is the maximum segment size (MSS). Denote the round-trip time between client and server as RTT (assume to be constant). Ignoring protocol headers, determine the time to retrieve the object (including TCP connection establishment) when

a) 4 S/R > S/R + RTT > 2S/R

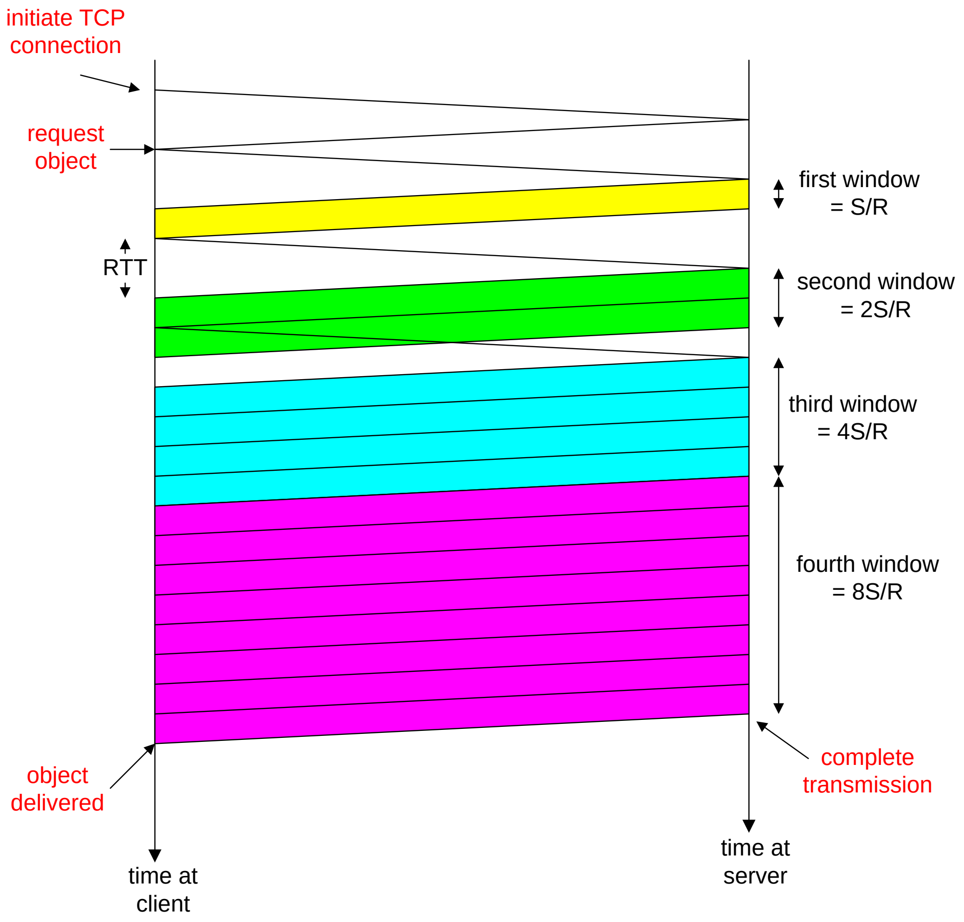
b) S/R + RTT > 4 S/R

a) Referring to the figure below, we see that the total delay is

RTT + RTT + S/R + RTT + S/R + RTT + 12S/R = 4RTT + 14 S/R

b) Similarly, the delay in this case is:

RTT+RTT + S/R + RTT + S/R + RTT + S/R + RTT + 8S/R = 5RTT +11 S/R


initiate TCP connection
request object
RTT
first window = S/R
second window = 2S/R
third window = 4S/R
fourth window = 8S/R
object delivered
complete transmission
time at client
time at server