Ch3

R4.

Describe why an application developer might choose to run an application over UDP rather than TCP.

Answer:
An application developer may not want its application to use TCP's congestion control, which can throttle the application's sending rate at times of congestion. Often, designers of IP telephony and IP videoconference applications choose to run their applications over UDP because they want to avoid TCP's congestion control. Also, some applications do not need the reliable data transfer provided by TCP.

R7.

Suppose a process in Host C has a UDP socket with port number 6789.Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789. Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts?

Answer:
Yes, both segments will be directed to the same socket. For each received segment, at the socket interface, the operating system will provide the process with the IP addresses to determine the origins of the individual segments.

R8.

Suppose that a Web server runs in Host C on port 80. Suppose this Web server uses persistent connections, and is currently receiving requests from two different Hosts, A and B. Are all of the requests being sent through the same socket at Host C? If they are being passed through different sockets, do both of the sockets have port 80? Discuss and explain.

Answer:
For each persistent connection, the Web server creates a separate "connection socket". Each connection socket is identified with a four-tuple: (source IP address,

source port number, destination IP address, destination port number). When host C receives and IP datagram, it examines these four fields in the datagram/segment to determine to which socket it should pass the payload of the TCP segment. Thus, the requests from A and B pass through different sockets. The identifier for both of these sockets has 80 for the destination port; however, the identifiers for these sockets have different values for source IP addresses. Unlike UDP, when the transport layer passes a TCP segment's payload to the application process, it does not specify the source IP address, as this is implicitly specified by the socket identifier.

R9.

In out rdt protocols, why did we need to introduce sequence numbers?

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

R10.

In our rdt protocols, why did we need to introduce timers?

Answer:
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 or NACK) is assumed to have been lost. Hence, the packet is retransmitted.

R11.

Suppose that the roundtrip delay between sender and receiver is constant and known to the sender. Would a timer still be necessary in protocol rdt 3.0, assuming that packets can be lost? Explain.

Answer:
A timer would still be necessary in the protocol rdt 3.0. 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 (or NACK) for the packet has been lost, as compared to the real scenario, where the ACK (or NACK) 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.

R14.

True or false?

a. Host A is sending host B a large file over a TCP connection. Assume host B has no data to send A. Host B will not send acknowledgements to host A because host B cannot piggyback the acknowledgements on data.
b. The size of the TCP rwnd never changes throughout the duration of the connection.
c. Suppose Host A is sending Host B a large file over a TCP connection. The number of unacknowledged bytes that A sends cannot exceed the size of the receive buffer.
d. Suppose host A is sending a large file to host B over a TCP connection. If the sequence number for a segment of this connection is m, then the sequence number for the subsequent segment will necessarily be m+1.
e. The TCP segment has a field in its header for rwnd.
f. Suppose that the last SampleRTT in a TCP connection is equal to 1 sec. The current value of TimeoutInterval for the connection will necessarily be >= 1 sec.
g. Suppose host A sends one segment with sequence number 38 and 4 bytes of data over a TCP connection to Host B. Then in this same segment the acknowledgement number is necessarily 42.

Answer:

a) false b) false c) true d) false e) true f) false g) false

R15.

Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has sequence number 90; the second has sequence number 110.

a. How much data is in the first segment?
b. Suppose that the first segment is lost but the second segment arrives at B. In the acknowledgement that Host B sends to Host A, what will be the acknowledgement number?

Answer:

a) 20 bytes b) ack number = 90

P6.

Consider our motivation for correcting protocol rdt 2.1. Show that the receiver, shown in Figure 3.57, when operating with the sender shown in Figure 3.11, can lead the sender and receiver to enter into a deadlock state, where each is waiting for an event that will never occur.

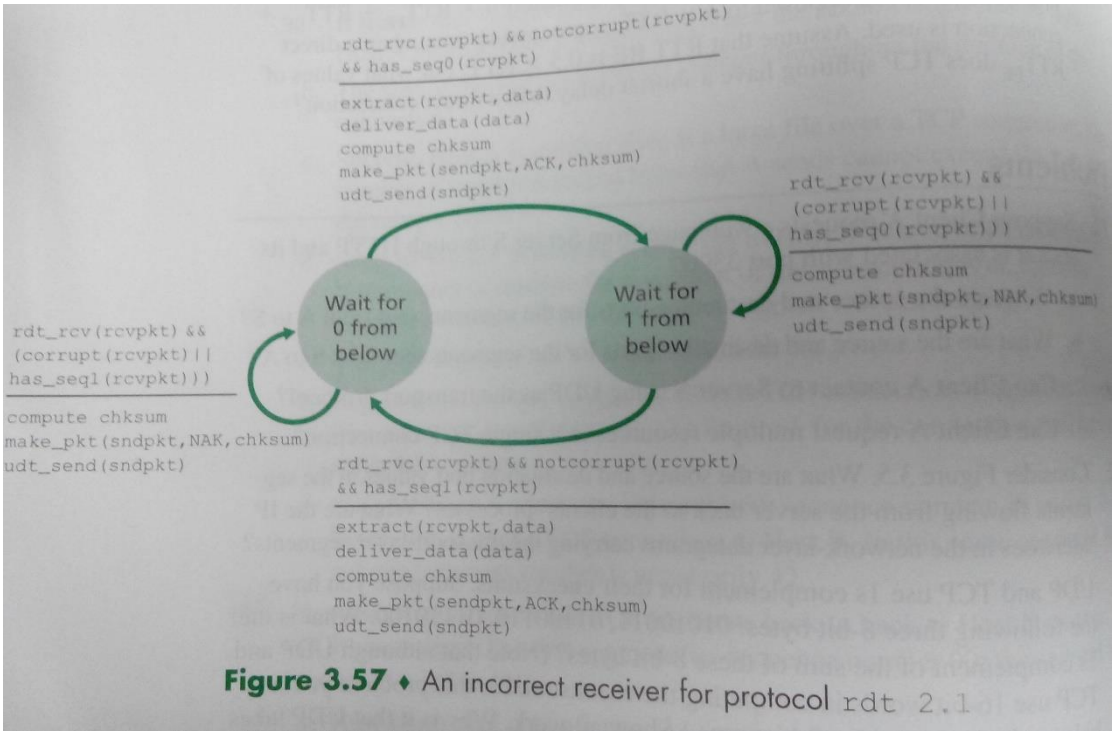


Figure 3.57 ♦ An incorrect receiver for protocol rdt 2.1

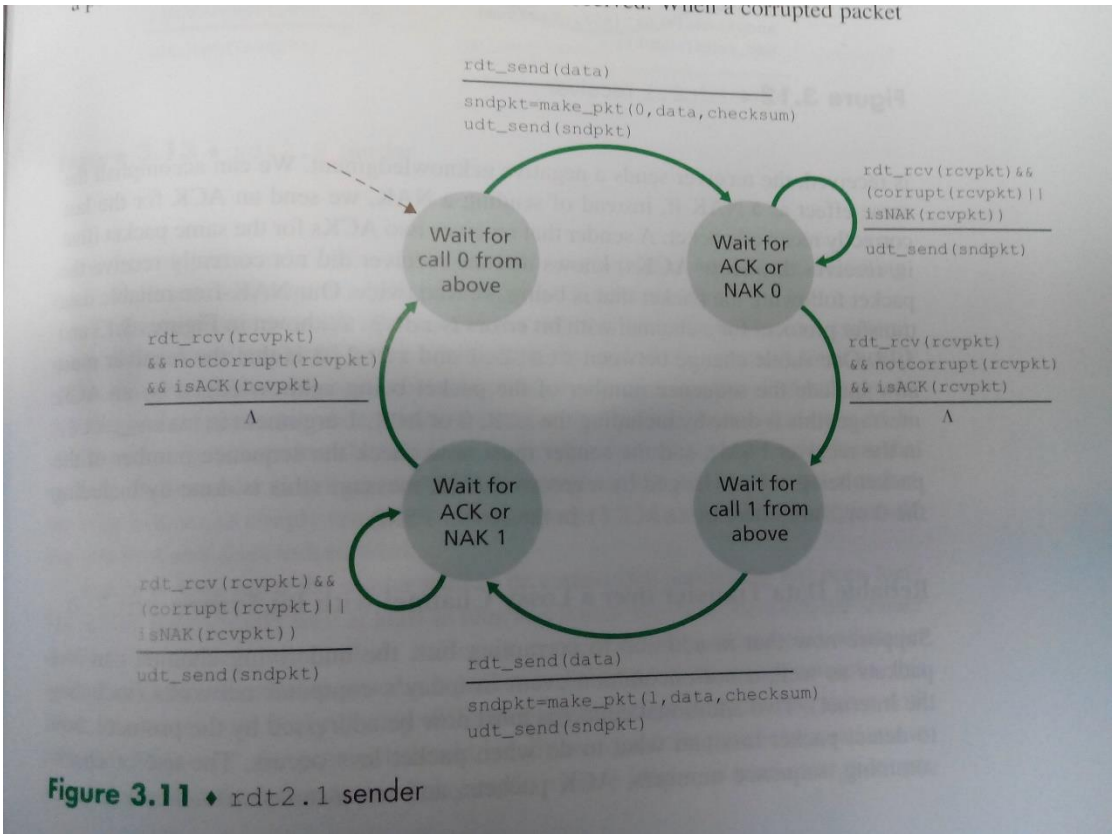


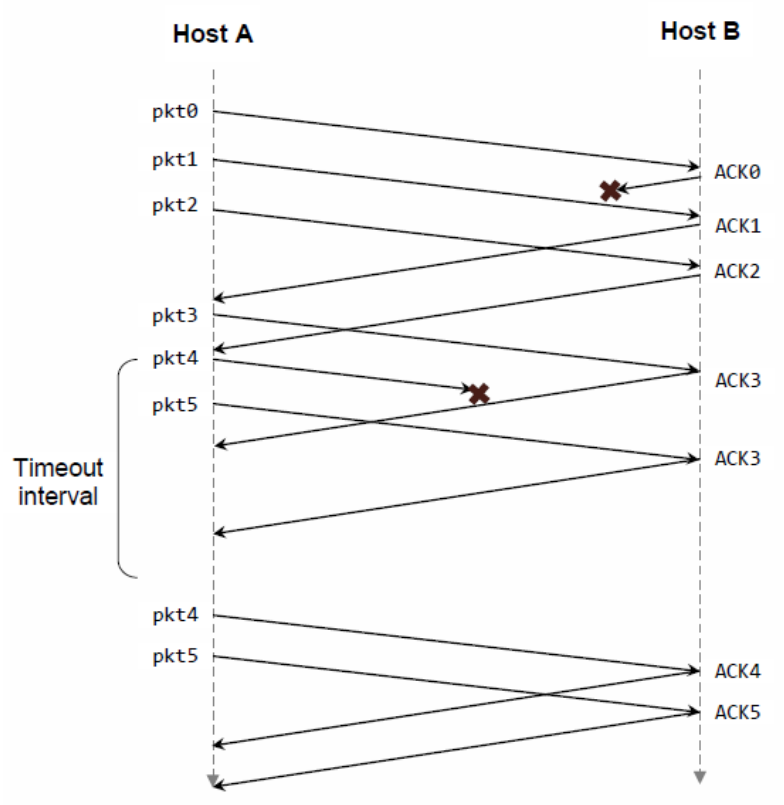
Figure 3.11 ♦ rdt2.1 sender

Answer:
Suppose the sender is in state “Wait for call 1 from above” and the receiver (the receiver shown in the homework problem) is in state “Wait for 1 from below.” The sender sends a packet with sequence number 1, and transitions to “Wait for ACK or NAK 1,” waiting for an ACK or NAK. Suppose now the receiver receives the packet with sequence number 1 correctly, sends an ACK, and transitions to state “Wait for 0 from below,” waiting for a data packet with sequence number 0. However, the ACK is corrupted. When the rdt2.1 sender gets the corrupted ACK, it resends the packet with sequence number 1. However, the receiver is waiting for a packet with sequence number 0 and (as shown in the home work problem) always sends a NAK when it doesn't get a packet with sequence number 0. Hence the sender will always be sending a packet with sequence number 1, and the receiver will always be NAKing that packet. Neither will progress forward from that state.

P19.

Suppose Host A and Host B use a GBN protocol with windows size N =3 and a long-enough range of sequence numbers. Assume Host A sends six application messages to Host B and that all messages are correctly received, except for the first acknowledgement and the fifth data segment. Draw a timing diagram (similar to Figure 3.22), showing the data segments and the acknowledgements sent along with the corresponding sequence and acknowledgement numbers, respectively.

Answer:



P20.

Consider a scenario in which Host A and Host B want to send messages to Host C. Host A and C are connected by a channel that can lose and corrupt (but not reorder) messages. Host B and C are connected by another channel (independent of the channel connecting A and C) with the same properties. The transport layer at Host C should alternate in delivering messages from A and B to the layer above (that is, it should first deliver the data from a packet from A, then the data from a packet from B, and so on). Design a stop-and-wait error-control for reliably transferring packets from A and B to C, with alternating delivery at C as described above. Give FSM descriptions of A and C. (Hint: The FSM for B should be essentially the same as for A.) Also, give a description of the packet format(s) used.

Answer:

Receiver:

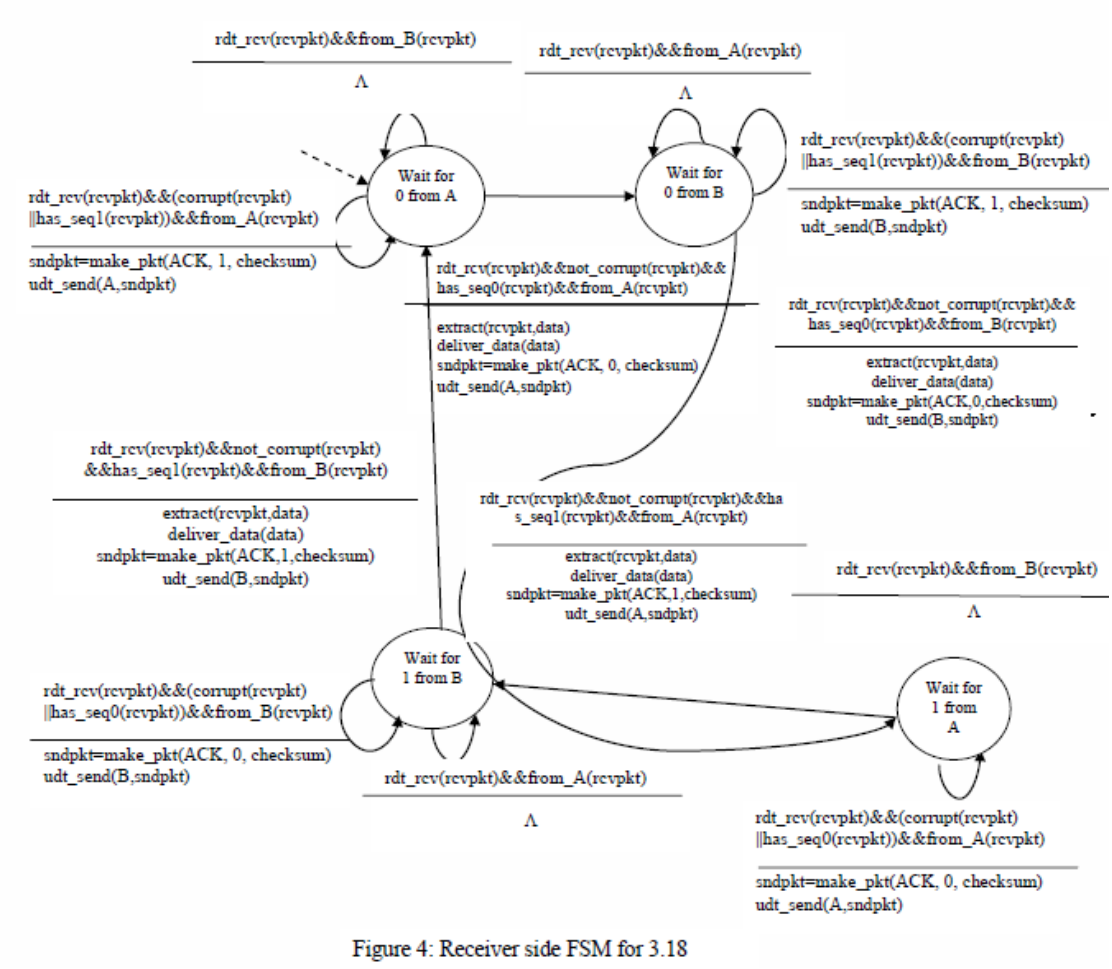
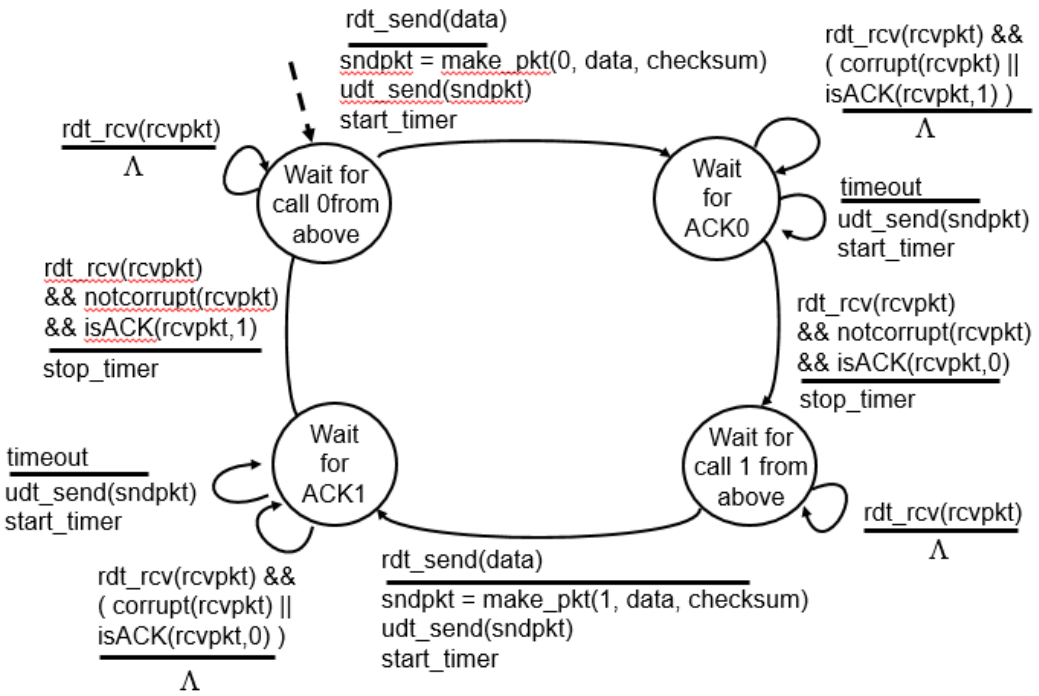


Figure 4: Receiver side FSM for 3.18

Sender: rdt3.0



P32.

Consider the TCP procedure for estimating RTT. Suppose that α= 0.1. Let $SampleRTT_1$ be the most recent sample RTT, let $SampleRTT_2$ be the next most recent sample RTT, and so on.

a. For a given TCP connection procedure, suppose four acknowledgements have been returned with corresponding sample RTTs: $SampleRTT_4$, $SampleRTT_3$, $SampleRTT_2$, and $SampleRTT_1$. Express EstimatedRTT in terms of the four sample RTTs.

b. Generalize your formula for n sample RTTs.

c. For the formula in part (b) let n approach infinity. Comment on why this averaging procedure is called an exponential moving average.

Answer:

a)

Denote $EstimatedRTT^{(n)}$ for the estimate after the nth sample.

$$\begin{aligned} EstimatedRTT^{(4)} &= xSampleRTT_1 + \\ &\quad (1-x)[xSampleRTT_2 + \\ &\quad (1-x)[xSampleRTT_3 + (1-x)SampleRTT_4]] \\ &= xSampleRTT_1 + (1-x)xSampleRTT_2 \\ &\quad + (1-x)^2 xSampleRTT_3 + (1-x)^3 SampleRTT_4 \end{aligned}$$

b)

$$\begin{aligned} EstimatedRTT^{(n)} &= x\sum_{j=1}^{n-1}(1-x)^{j-1}SampleRTT_j \\ &\quad + (1-x)^{n-1} SampleRTT_n \end{aligned}$$

c)

$$\begin{aligned} EstimatedRTT^{(\infty)} &= \frac{x}{1-x}\sum_{j=1}^{\infty}(1-x)^j SampleRTT_j \\ &= \frac{1}{9}\sum_{j=1}^{\infty}.9^j SampleRTT_j \end{aligned}$$

The weight given to past samples decays exponentially.

P40.

Consider Figure 3.58. 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?

j. Suppose TCP Tahoe is used (instead of TCP Reno), and assume that triple duplicate ACKs are received at the 16th round. What are the ssthresh and the congestion window size at the 19th round?

k. Again suppose TCP Tahoe is used, and there is a timeout event at 22nd round. How many packets have been sent out from 17th round till 22nd round, inclusive?

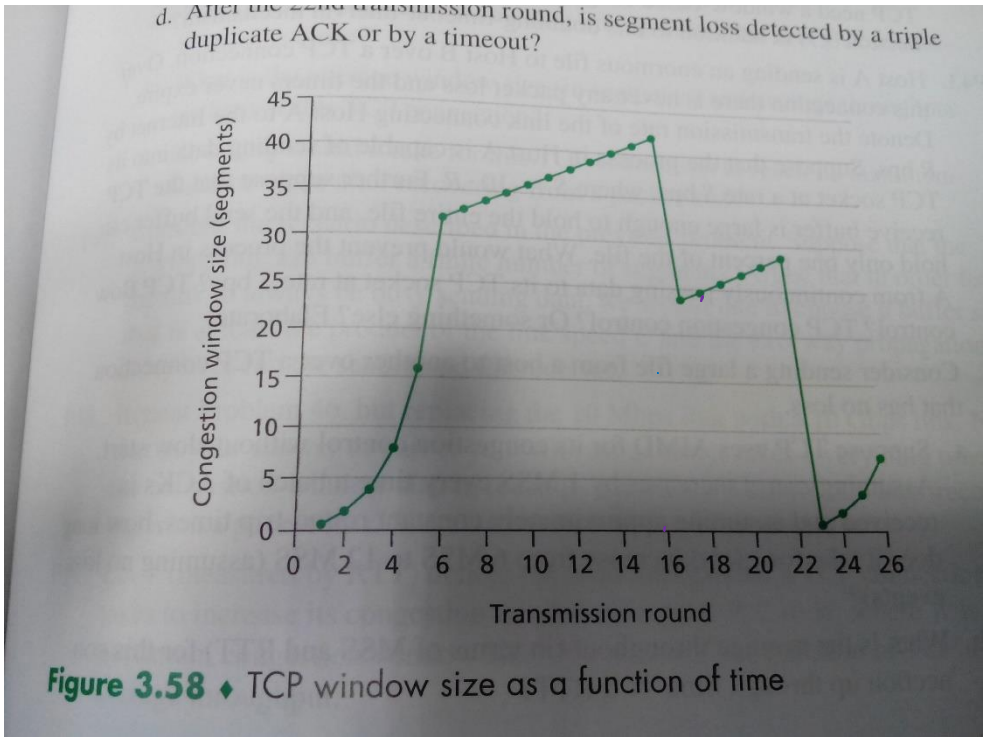


Figure 3.58 ♦ TCP window size as a function of time

Answer:

a) TCP slowstart 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.

j) threshold is 21, and congestion window size is 1.Then, the windows size will increase to 4 at round 19.

k) round 17, 1 packet; round 18, 2 packets; round 19, 4 packets; round 20, 8 packets; round 21, 16 packets; round 22, 21 packets. So, the total number is 52.