Problem 1

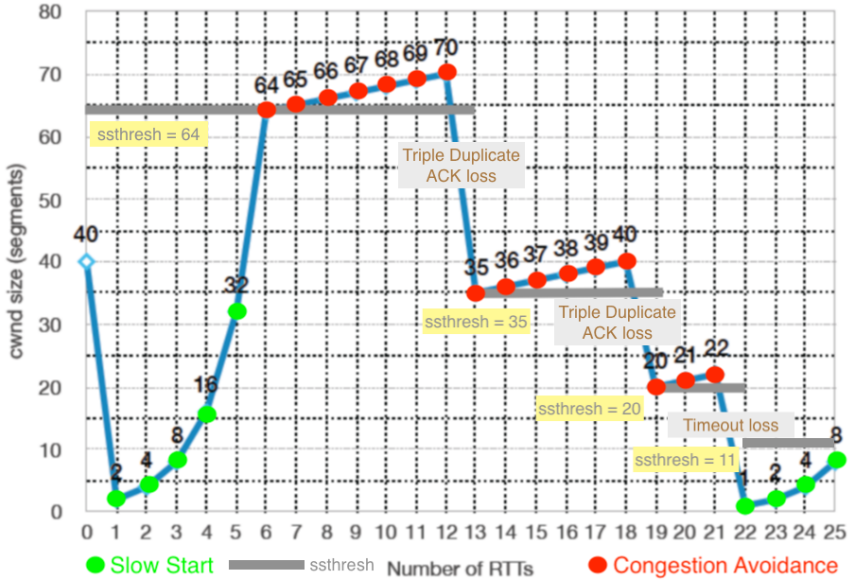

Figure 1: TCP congestion control.

Assume that a TCP connection has been running between hosts A and B for sometime, so that the number of RTTs shown in the above graph are with respect to the time when you started observing the the cwnd value of this connection. Hosts A and B use TCP Reno (with Fast Retransmit and Fast Recovery).

1. **On the graph above**, identify the time periods when TCP slow start is operating.
2. **On the graph above**, identify the time periods when TCP congestion avoidance is operating (AIMD).
3. For each loss event, specify whether it was detected by a triple duplicate ACK or by a timeout
4. For each loss event, indicate the value of the slow start threshold (ssthresh)

1. The green points (Slow Start) on the graph above indicate time periods of $RTT = [1, 6), [22, 25]$
2. The red points (Congestion Avoidance) on the graph above indicate time periods of $RTT = [6, 12], [13, 18], [19, 21]$
3. The graph has three labels for each loss event:
   (a) The first loss event happens between $RTT = (12, 13)$ and was detected by a triple duplicate ACK because *cwnd* only halved and didn't get reset to 1 *segment*.
   (b) The second loss event happens between $RTT = (18, 19)$ and was detected by a triple duplicate ACK because *cwnd* only halved and didn't get reset to 1 *segment*.
   (c) The third loss event happens between $RTT = (21, 22)$ and was detected by a timeout because *cwnd* was reset to 1 *segment*.
4. *ssthresh* values after each loss event:
   (a) Loss at $RTT = (12, 13)$: $ssthresh = 35$
   (b) Loss at $RTT = (18, 19)$: $ssthresh = 20$
   (c) Loss at $RTT = (21, 22)$: $ssthresh = 11$

Problem 2

Assume that host A sets up a TCP connection with host B to send data. Assume that A's `ssthresh` value is 64 KB and TCP segment size is 3 KB.

1. How many bytes have been transmitted after 3 RTTs assuming no losses? Show your work.
2. Now, suppose that after the third RTT, a loss occurs which results in the TCP sender's retransmission timer to expire. What actions will TCP congestion control take in this case?
3. Assuming no further packet loss occurs from then on, how many RTTs does it take to transmit an additional 22 KB of data?

time (RTT)	0	1	2	3	timeout	4	5	6	7	8
ssthresh (KB)	64KB	64KB	64KB	64KB	timeout	12KB	12KB	12KB	12KB	12KB
cwnd (segments)	1	2	4	8	timeout	1	2	4	5	6
cwnd (KB)	3KB	6KB	12KB	24KB	timeout	3KB	6KB	12KB	15KB	18KB
transmitted (KB)	0KB	3KB	9KB	21KB	timeout	0KB	3KB	9KB	21KB	36KB

1. Looking at the table, after the 3rd RTT we've transmitted 21KB of data assuming no losses.
2. TCP congestion control sets:
   (a) $ssthresh = \frac{cwnd}{2} = \frac{24KB}{2} = 12KB$
   (b) $cwnd = 1segment = 3KB$
3. Assuming no further packet loss, it takes another 4 RTT to transmit an additional 22KB of data after the loss after the 3rd RTT to transmit an additional $22KB$ of data starting from $RTT = 4$.

Problem 3

True or False? Briefly explain your answer in a single sentence.

1. Host A is sending Host B a large file over a TCP connection. Assume Host B has no data to send Host A. Host B will not send acknowledgments to Host A because Host B cannot piggyback the acknowledgments on data.
2. The size of the TCP *rwnd* never changes throughout the duration of the connection.
3. 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.
4. 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.
5. The TCP segment has a field in its header for *rwnd*.
6. 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 $\geq$ 1 sec.
7. Suppose Host A sends one segment with sequence number 38 and 4 bytes of data over a TCP connection to Host B. In this same segment the acknowledgment number is necessarily 42.

1. False. Host B will always send ACKs back to A when over a TCP connection.
2. False. *rwnd* dynamically changes throughout the connection based on the space available at each respective host.
3. True. Host A will never send more than what B's receive buffer can handle, so the worst case of having every byte sent from A be unacknowledged still cannot exceed the size of the receive buffer.
4. False. If the sequence number for a segment is m, then the sequence number for the subsequent segment will be $m+$ the number of bytes sent from A to B in the segment with sequence number m, which is not necessarily 1.
5. True. Each TCP segment has a header field for *rwnd* which tells the other host how many bytes it can receive.
6. False. The current value of TimeoutInterval adjusts itself based on all of the SampleRTT's that happen. Each new SampleRTT will only slightly affect the current TimeoutInterval value. If SRTT was much less than 1 second before the most recent SampleRTT, then the timer is likely to have a value that is $<$ 1 sec.
7. False. The acknowledgement number of this segment is the sum of the previous segment from B to A's sequence number and the number of bytes received from B.

Problem 4

As we have discussed in the class, a timer is a useful component in various protocol designs: because a communicating end cannot see what is going on either inside the network or at the other end, when needed it sets up an "alarm", and takes some action when the alarm goes off.

1. Does HTTP use any timers? If so, please briefly describe how each is used. If not, please explain why it does not need one.
2. Does DNS use any timers? If so, please briefly describe how each is used. If not, please explain why it does not need one.
3. Does TCP use any timer? If so, please briefly describe how each is used. If not, please explain why it does not need one.
4. Does UDP use any timer? If so, please briefly describe how each is used. If not, please explain why it does not need one.

1. Yes. HTTP uses timers in case an HTTP request that is sent out does not see a response for some reason. Although it is built on TCP, there is still value in retransmitting a request if no response is received within a specific time frame. This is useful in detecting if a server is down or if a request takes too long for an unknown reason as we can simply retransmit the request. Generally, there are idempotent HTTP operations that support retransmissions as multiple requests are equivalent to a single request.
2. Yes. DNS generally uses UDP to send queries. Because there is no guarantee of the query being sent or a response being received, a timer is required and implemented by the DNS itself. This ensures that a host will eventually receive a response from the caching resolver. If there was no timer, and the UDP packets get lost, then the host could be waiting for the cache resolver forever.
3. Yes. TCP uses timer in the form of RTOs which are calculated by taking $SampleRTT$s to update $SRTT$ and $DevRTT$ values, which are finally used to update RTO with the equation $RTO = SRTT + 4 \times DevRTT$. A timer is needed so that lost packets can be retransmitted. This allows TCP to provide the illusion of a reliable connection despite being built on an inherently unreliable network layer.
4. No. UDP is a connectionless, best-attempt protocol and does not guarantee reliability. This means that a timer is not needed, as it only promises to attempt to send packets. Once it is sent, the UDP protocol is no longer responsible for any response or even guarantee of the packets reaching the destination.