Transport Congestion Control - Answers

Q1) Figure 1 shows the trace of Congestion window for a particular TCP implementation. Would it be a TCP Reno or TCP Tahoe? Why?

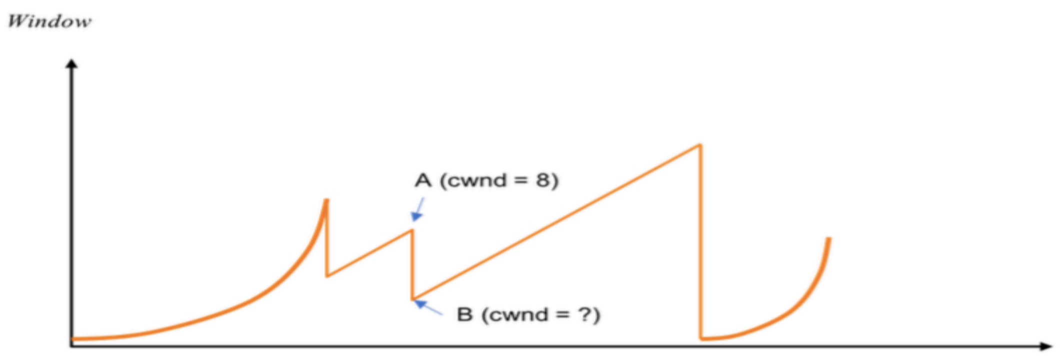

Figure 1. TCP Congestion Window trace

Answer: It is TCP Reno. Note that Congestion Window is reduced to half twice in this trace. TCP Tahoe would always reduce its Congestion Window to 1 in reposne to a congestion event.

Q2) In Figure 1, what would be the value of TCP Congestion window at point B?

Answer: 8/2 = 4 as the congestion window is halved.

Q3) Figure 2 shows congestion window traces for both TCP Tahoe and TCP Reno where up to transmission round of 8, both follow the same blue curve, but after that TCP Tahoe follows the blue curve and TCP Reno follow the black curve.

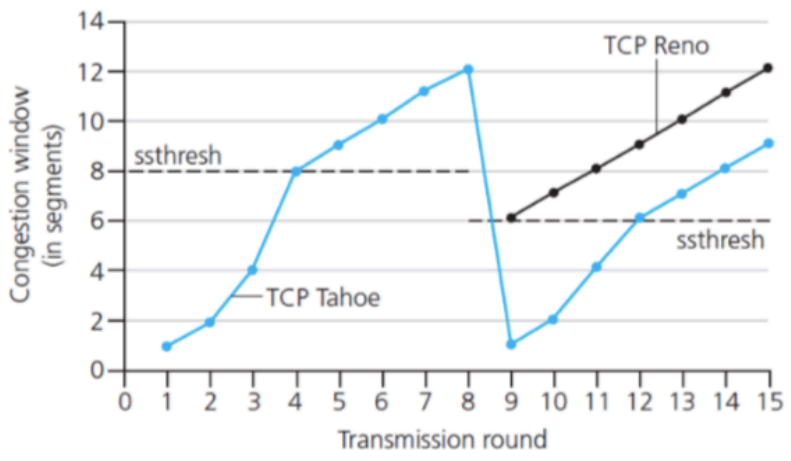

Figure 2: Congestion window trace for TCP Tahoe and TCP Reno

Answer the following Questions:

¶a) What has happened at transmission round No 8?

Answer: A Triple Duplicate ACK has been observed. Note different reactions for TCP Tahoe and TCP Reno.

¶b) Identify the regions where Slow Start (SS) and Congestion Avoidance (CA) are in operation?

Answer:

SS =

Between Rounds 1-4 (both TCP Tahoe and Reno) and

Between Rounds 9-12 for TCP Tahoe

CA=

Between Rounds 4-8 (both TCP Tahoe and Reno)

Between Rounds 9-15 for TCP Reno

Between Rounds 12-15 for TCP Tahoe

¶c) How many total segments has been transferred by TCP Tahoe and TCP Reno at the end of round 15?

Answer:

Add all the congestion window sizes till round 15.

TCP Tahoe: 1+2+4+8+9+10+11+12+1+2+4+6+7+8+9=94 segments

TCP Reno: 1+2+4+8+9+10+11+12+6+7+8+9+10+11+12=120 segments

Q4) TCP waits until it has received three duplicate ACKs before performing a fast retransmit. Why do you think the TCP designers chose not to perform a fast retransmit after the first duplicate ACK for a segment is received?

Answer: Suppose packets n, n+1, and n+2 are sent, and that packet n is received and ACKed. If packets n+1 and n+2 are reordered along the end-to-end-path (i.e., are received in the order n+2, n+1) then the receipt of packet n+2 will generate a duplicate ack for n and would trigger a retransmission under a policy of waiting only for first duplicate ACK for retransmission.

By waiting for a triple duplicate ACK, it must be the case that three packets after packet n are correctly received, while n+1 was not received. The designers of the triple duplicate ACK scheme probably felt that waiting for three packets (rather than 1) was the right trade-off between triggering a quick retransmission when needed, but not retransmitting prematurely in the face of packet reordering.

Q5) Consider sending a large file from a host to another over a TCP connection that has no loss. Suppose TCP uses AIMD for its congestion control with the SSthrehold at 6 MSS. Assuming approximately constant round-trip times, how long does it take for cwnd to increase to 12 MSS (assuming no loss events, neglect TCP connection establishment time)?

Answer:

RTT : Data transferred in this RTT , Total Data transferred

1 RTT : 1, 1

2 RTT : 2, 3

3 RTT : 4, 7

4 RTT : 6, 13

5 RTT : 7, 20

6 RTT : 8, 28

7 RTT : 9, 37

8 RTT : 10, 47

9 RTT : 11, 58

10 RTT : 12, 70

Note that the Cwnd would have reached value of 12 MSS at the end of 9th RTT. At the end of 10 RTT, we would have received back all ACKs for the 12 MSS transferred in the 10 RTT.

What is the average throughout (in terms of MSS and RTT) for this connection up through time = 6 RTT?

Answer:

At end of 6 RTT, 28 MSS were sent (and acknowledged). Thus, we can say that the average throughput up to time 6 RTT was (28 MSS)/(6 RTT) = 4.66 MSS/RTT.