Comnet – HW4

Oz Cabiri & Guy Harem

Part2, Theoretical Questions:

Question 1:

Packets are of size 1200 MBps, Transmission speed is 10Gbps, and the Propagation delay is 20ms.

a) Packet transmission time:

$T_{trans} = \frac{PacketSize}{Bandwith} = \frac{1200*8}{10*10^9} = \frac{960}{10^9} = 960nsec$

$RTT = 2 * PropagationDelay = 2 * 20ms = 40ms$

$$UtilizationTime = \frac{T_{trans}}{RTT + T_{trans}} = \frac{960 * 10^{-9}}{40 * 10^{-6} + 960 * 10^{9}} = 2.4 * 10^{-5}$$

b) In Go-Back-N, we are using pipelining, and we send a full window of (N) packets without waiting for acks.
Given we have 8 bits to represent SEQ numbers, we can represent 2^8 SEQ numbers, so in GBN, the largest window possible (to avoid ambiguity) is N = 2^8-1= 255 to fully utilize the channel.

The utilization of GBN would be N times better than Stop-and-Wait, but it cannot pass 100% utilization, so:

$UtilizationTime = \min\left(1, \frac{N*T_{trans}}{RTT+T_{trans}}\right) = \min(1, 0.00612) = 0.00612$

Question 2:

We want to transfer a file of size L, from host A to host B, using a TCP connection with both transmitter/receiver windows of size W.

W (WindowSize) is limited to 16 bits, and the SEQ number is limited to 32 bits.

a) **Assuming the channel is FIFO (in-order transmission):**
   Yes, it is sufficient, to prevent ambiguity in a sliding window protocol, the SEQ number space must be at least (2*WindowSize).
   The question didn't specifically point which algorithm we use (GBN or SR), but for both, the SEQ Space is much bigger than the required unique SEQ numbers (N+1 for GBN, 2N for SR):

$$SEQ\ Space = 2^{32} \gg 2 * WindowSize = 2 * 2^{16} = 2^{17}$$

   Since 2^32 is much larger than 2^17, we have sufficient SEQ unique numbers to transmit a file of any size L (even infinite) with no ambiguity errors.

b) **Assuming real-world behavior (Non-FIFO):**
   Yes, the bounds are sufficient, in reality, packets cannot survive in the network forever due to TTL, so an "old" packet will be discarded by the network long before the SEQ numbers wrap around (2^32) and reach the current window again.

Question 3:

In this question, we're using TCP Reno.

For Time 6:

1mss can be if we encountered a timeout (cwnd = 1).

Time	CWND	SSTHRESH	Event
1	28 mss	32 mss	Timeout
2	1 mss	28/2 = 14 mss	
3	2 mss	14 mss	
4	4 mss	14 mss	
5	8 mss	14 mss	Timeout
6	**1 mss**	8/2 = 4 mss	

4 mss can be if we encountered 3-dup acks (cwnd = cwnd/2).

Time	CWND	SSTHRESH	Event
1	28 mss	32 mss	Timeout
2	1 mss	28/2 = 14 mss	
3	2 mss	14 mss	
4	4 mss	14 mss	
5	8 mss	14 mss	3-dup acks
6	**4 mss**	8/2 = 4 mss	

9 mss can be if we passed the ssthresh and the growth is linear. (+lower bound)

Time	CWND	SSTHRESH	Event
1	28 mss	32 mss	
2	32 mss	32 mss	
3	33 mss	32 mss	3-dup acks
4	16 mss	33/2 = 16 mss	3-dup acks
5	8 mss	16/2 = 8 mss	
6	**9 mss**	8 mss	

14 mss can be if ssthresh is 14, we still haven't reached it, and after exponential growth we reach to the ssthresh = 14.

Time	CWND	SSTHRESH	Event
1	28 mss	32 mss	Timeout
2	1 mss	28/2 = 14 mss	
3	2 mss	14 mss	
4	4 mss	14 mss	
5	8 mss	14 mss	
6	**14 mss**	14 mss	

16 mss is not possible, exponential grown cannot surpass the ssthresh at first reach, we grow to its value (even if ssthresh < 2*CWND).

*We didn't talk about it in class, but the ssthresh can only grow when a congestion occurred (3-dup acks or RTO), being set to ssthresh = cwnd/2.