TÉCNICO LISBOA

RC 17/18	**LAB ASSIGNMENT**	**Number:**	5
Problems		**Issue Date:**	18 Oct 2017
Application and Transport Layer		**Issue Date:**	

Preliminary Notes

All exercises in this assignment are adapted from the 7th edition of *Computer Networking: a top-down approach* by Kurose and Ross.

These exercises allow a better understanding about Transport Layer protocols: UDP and TCP.

Exercises

1. Suppose that the five measured $SampleRTT$ values are 106 ms , 120 ms, 140 ms, 90 ms, and 115 ms. Compute the $EstimatedRTT$ after each of these $SampleRTT$ values is obtained, using a value of $\alpha = 0.125$ and assuming that the value of $EstimatedRTT$ was 100 ms just before the first of these five samples were obtained. Compute also the $DevRTT$ after each sample is obtained, assuming a value of $\beta = 0.25$ and assuming that the value of $DevRTT$ was 5 ms just before the first of these five samples was obtained. Lastly, compute the TCP $TimeoutInterval$ after each of these samples is obtained.

Solution:

$DevRTT_n = (1 - \beta) * DevRTT_{n-1} + \beta * |SampleRTT_n - EstimatedRTT_{n-1}|$

$EstimatedRTT_n = (1 - \alpha) * EstimatedRTT_{n-1} + \alpha * SampleRTT_n$

$TimeoutInterval_n = EstimatedRTT_n + 4 * DevRTT_n$

$DevRTT_0 = 5$ ms; $EstimatedRTT_0 = 100$ ms

$TimeoutInterval_0 = 100 + 4 * 5 = 120$ ms

$SampleRTT_1 = 106$ ms

$DevRTT_1 = (1 - 0.25) * 5 + 0.25 * |106 - 100| = 5.25$ ms

$EstimatedRTT_1 = (1 - 0.125) * 100 + 0.125 * 106 = 100.75$ ms

$TimeoutInterval_1 = 100.75 + 4 * 5.25 = 121.75$ ms

$SampleRTT_2 = 120$ ms

$DevRTT_2 = (1 - 0.25) * 5.25 + 0.25 * |120 - 100.75| = 8.75$ ms

$EstimatedRTT_2 = (1 - 0.125) * 100.75 + 0.125 * 106 = 103.16$ ms

$TimeoutInterval_2 = 103.16 + 4 * 8.75 = 138.16$ ms

$SampleRTT_3 = 140$ ms

$DevRTT_3 = (1 - 0.25) * 8.75 + 0.25 * |140 - 103.16| = 15.77$ ms

$EstimatedRTT_3 = (1 - 0.125) * 103.16 + 0.125 * 140 = 107.76$ ms

$TimeoutInterval_3 = 107.76 + 4 * 15.77 = 170.84$ ms

$SampleRTT_4 = 90$ ms

$DevRTT_4 = (1 - 0.25) * 15.77 + 0.25 * |90 - 107.76| = 16.27$ ms

$EstimatedRTT_4 = (1 - 0.125) * 105.54 + 0.125 * 115 = 106.72$ ms

$TimeoutInterval_4 = 106.72 + 4 * 16.27 = 170.62$ ms

$SampleRTT_5 = 115$ ms

$DevRTT_5 = (1 - 0.25) * 16.27 + 0.25 * |115 - 105.54| = 14.57$ ms

$EstimatedRTT_5 = (1 - 0.125) * 105.54 + 0.125 * 115 = 103.16$ ms

$TimeoutInterval_5 = 103.16 + 4 * 8.75 = 138.16$ ms

2. Compare the protocols Go-Back-N (GBN), Selective-Repeat (SR) and TCP (no delayed ACK). Assume that the timeout values for all three protocols are sufficiently long such that 5 consecutive data segments and their corresponding ACKs can be received (if not lost in the channel) by the receiving host (Host B) and the sending host (Host A) respectively. Suppose Host A sends 5 data segments to Host B, and the 2^{nd} segment (sent from A) is lost. In the end, all 5 data segments have been correctly received by Host B.

   (a) How many segments has Host A sent in total and how many ACKs has Host B sent in total? What are their sequence numbers? Answer this question for all three protocols.

   Solution:
   GoBackN:

   - A sends: 9 segments; Sequence Numbers: 1,2,3,4,5,2,3,4,5
   - B sends: 8 ACK's; Acknowledgment Numbers: 1,1,1,1,2,3,4,5

Selective Repeat:

- A sends: 6 segments; Sequence Numbers: 1,2,3,4,5,2
- B sends: 5 ACK's; Acknowledgment Numbers: 1,3,4,5,2

TCP:

- A sends: 6 segments; Sequence Numbers: 1,2,3,4,5,2
- B sends: 5 ACK's; Acknowledgment Numbers: 2,2,2,2,6

(b) If the timeout values for all three protocols are much longer than 5 RTT. then which protocol successfully delivers all five data segments in shortest time interval?

Solution:
TCP uses fast retransmit without waiting until timeout, so TCP would be the fastest one.

3. Consider that only a single TCP (Reno) connection uses one 10 Mbit s^{-1} link which does not buffer any data. Suppose that this link is the only congested link between the sending and receiver and the receiver's receive buffer is much larger than the congestion window. We also make the following assumptions: each TCP segment size is $1,500$ B; the two-way propagation delay of this connection is 150 ms; and this TCP connection is always in congestion avoidance phase, that is, ignore slow start.

(a) What is the maximum window size (in segments) that this TCP connection can achieve?

Solution:
Let W denote the max window size in segments. Then, $W \times \frac{MSS}{RTT} = 10$ Mbit s^{-1}, as packets will be dropped if the maximum sending rate exceeds link capacity, which means that a greater W is "possible" but irrelevant for efficiency. We have $RTT = 150$ ms $= 0.15$ s and $MSS = 1500$ B $= 1500 \times 8$ bit. Thus, we have $W \times \frac{1500\times 8}{0.15} = 10 \times 10^6 \equiv W = \frac{0.15\times 10\times 10^6}{1500\times 8} = 125$ segments.

(b) What is the average window size (in segments) and average throughput (in bit s^{-1}) of this TCP connection?

Solution:
As congestion window size varies from $\frac{W}{2}$ to W, then the average window

size is $\frac{3\times W/2}{2} = 0.75 = \lceil 93.75 \rceil = 94$ segments. Average throughput is $\frac{94\times1500\times8}{0.15} = 7.52$ Mbit s^{-1}.

(c) How long would it take for this TCP connection to reach its maximum window again after recovering from a packet loss?

Solution:
When there is a packet loss, the congestion window size becomes $\frac{W}{2} = \frac{125}{2} = 62$ segments. Then, $(125 - 62) \times 0.15 = 9.45$ s, the number of RTT that TCP connection needs in order to increase its window size from 62 to 125. Recall the window size increases by one MSS in each RTT.

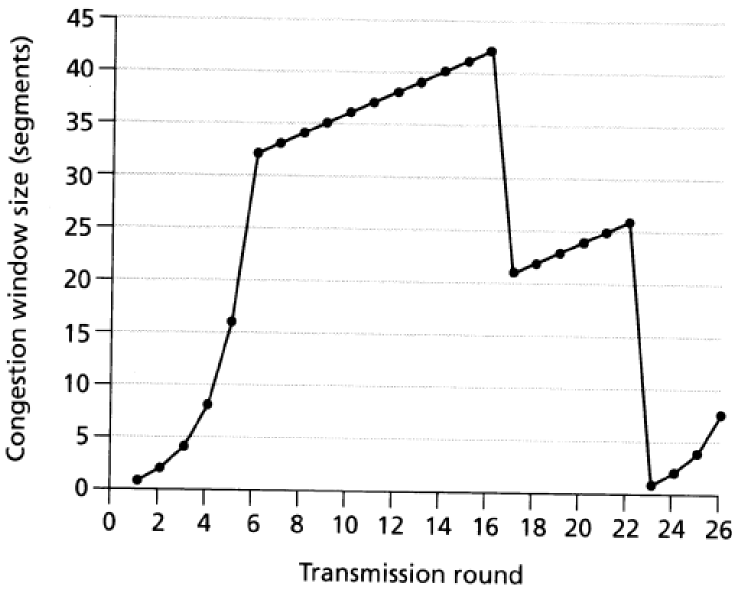

Figure 1: TCP congestion window size as a function of time

4. Consider Figure 1. Assuming TCP Reno is the protocol experiencing the behavior shown in the figure, 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.

Solution:
$[1, 6]$ and $[23, 26]$

(b) Identify the intervals of time when TCP congestion avoidance is operating.

Solution:
$[6, 16]$ and $[17, 22]$

(c) After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?

Solution:
Triple Duplicate ACK

(d) After the 22th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?

Solution:
Timeout

(e) What is the initial value of $ssthresh$ at the first transmission round?

Solution:
32 segments

(f) What is the value of $ssthresh$ at the 18th transmission round?

Solution:
21 segments

(g) What is the value of $ssthresh$ at the 24th transmission round?

Solution:
14 segments

(h) During what transmission round is the 70th segment sent?

Solution:
After the 6th transmission TCP already sent $1 + 2 + 4 + 8 + 16 + 32 = 63$ segments and after the 7th transmission TCP already had sent $1 + 2 + 4 + 8 + 16 + 32 + 33 = 96$ segments. Thus packet 70 is sent in the 7th transmission

(i) Assuming a packet loss is detected after the 26th round by receipt of a triple duplicate ACK, what will be the values of the congestion window size and if $ssthresh$?

Solution:
When a retransmit by triple duplicate ACK occurs in TCP Reno, $ssthresh_n = \frac{cwsize_{n-1}}{2}$ segments and $cwsize_n = ssthresh_n + 3$ segments. Then $ssthresh = 4$ and $cwsize = 7$

(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?

Solution:
When a retransmit by triple duplicate ACK occurs in TCP Tahoe, $ssthresh_n = \frac{cwsize_{n-1}}{2}$ segments and $cwsize_n = 1$ segment. Then $ssthresh = 21$ and $cwsize = 1$

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

Solution:
$1 + 2 + 4 + 8 + 16 + 21 = 52$ segments

5. Why was it necessary to introduce sequence numbers in order to create a reliable data transfer protocol?

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

6. Why was it necessary to introduce timers in order to create a reliable data transfer protocol?

Solution:
Timeout are requires 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.

7. In this exercise, answer with True (T) or False (F), in the space provided at the beginning of each question.

(a) __F__ Host *A* is sending Host *B* a large file over a TCP connection. Assume Host **B** has no data to send to Host **A**. Host **B** will not send acknowledgements to Host **A** because Host **B** cannot piggyback the acknowledgements on data.

(b) __F__ The size of the TCP $rwnd$ never changes throughout the duration of the connection.

(c) __T__ Suppose Host **A** is sending Host **B** a large file over a TCP connection. The number of unacknowledged bytes that **A** sends cannot exceeded the size of receive buffer

(d) __T__ The TCP segment has a field in its header for $rwnd$.

(e) __F__ Suppose that the last $SampleRTT$ in a TCP connection is equal to 1 s. The current value of $TimeoutInterval$ for the connection will necessarily be ≥ 1 s.

(f) __F__ Suppose Host **A** sends one segment with sequence number 38 and 4 B of data over a TCP connection to Host **B**. In this same segment the acknowledgment number is necessarily 42.