CSC 458/2209: Computer Networking Systems, Winter 2025

Department of Computer Science, University of Toronto

Problem Set 2 - Solutions **Date:** March 29, 2025

1a)

6 packets will be sent.
Time = 0, Packet1 is sent with content: H
Time = 2, E is buffered
Time = 4, L is added to buffer
Time = 4.58, ACK arrives, Packet2 is sent with content EL
Time = 6, L is added to buffer
Time = 8, O is added to buffer
Time = 9.16, ACK arrives, Packet3 is sent with content LO
Time = 10, _ is added to buffer
Time = 12, W is added to buffer
Time = 13.74, ACK arrives, Packet4 is sent with content _W
Time = 14, O is added to buffer
Time = 16, R is added to buffer
Time = 18, L is added to buffer
Time = 18.32, ACK arrives, Packet5 is sent with content ORL
Time = 20, Packet6 is sent with content D
The last packet is sent as soon as the connection is closed. No need to wait here.

1b)

With Nagel's algorithm, packet departure times are going to be similar to 1a as long as we do not reach the MSS. Since we send 40 bytes every 0.1 seconds, the buffer will have MSS bytes of data almost every 1.1 seconds (0.1 x 458/40 to be more precise). Once we have a full MSS, the packet will be sent out. So the mouse is going to be perceived to move every 1.1 seconds. Without Nagel's algorithm, the mouse will move more smoothly (with 0.1 second jumps).

1c)

With MSS equal to 80 bytes, we would have a full MSS every 0.2 seconds, and mouse movements will seem smoother even with Nagel's algorithm.

2a)

After the first sample, we will have EstimatedRTT = 0.5 * 1000 + 0.5 * 10.2 = 551. Continuing this, the EstimatedRTT will become 326.5, 214.25, 158.13, 130.06, 116.03, 109.2, 105.51, 103.75… We want to have error less than 0.1, which means that the EstimatedRTT should be between 95 and 105. It means after the 9^{th} sample, when the EstimatedRTT becomes 103.05, the error is less than 0.05.

For $\alpha = 0.4$, we will have the following values for EstimatedRTT: 461.2, 245.68, 159.47, 124.99, 111.20, 105.68, 103.47, … Which means that the error becomes less than 0.1 after the 8th sample.

2b)

First estimate: $1000\alpha + 102(1-\alpha) = 1000\alpha + 100 - 102\alpha = 898\alpha + 102$
Second estimate: $(898\alpha + 102)\alpha + 102\ (1 - \alpha) = 898\alpha^2 + 102\alpha + 102 - 102\alpha = 898\alpha^2 + 102$
Third estimate: $(898\alpha^2 + 102)\alpha + 102\ (1 - \alpha) = 898\alpha^3 + 102\alpha + 102 - 102\alpha = 898\alpha^3 + 102$
Fourth estimate: $(898\alpha^3 + 102)\alpha + 102\ (1 - \alpha) = 898\alpha^4 + 102\alpha + 102 - 102\alpha = 898\alpha^4 + 102$
Therefore, we should have: $898\alpha^4 + 102 < 110$, which means that $898\alpha^4 < 8$, which means $\alpha^4 < 8/898=0.00089$, which means $\alpha < 0.307$

3a)

The bottleneck link rate is 2 packet/second, and the round-trip propagation delay is 4 seconds so we can have 2x4 packets in the links connecting the source to the destination and back. With 4 packets of buffering, the congestion window can grow to 12 packets. If it goes beyond that, it will lead to a packet drop.

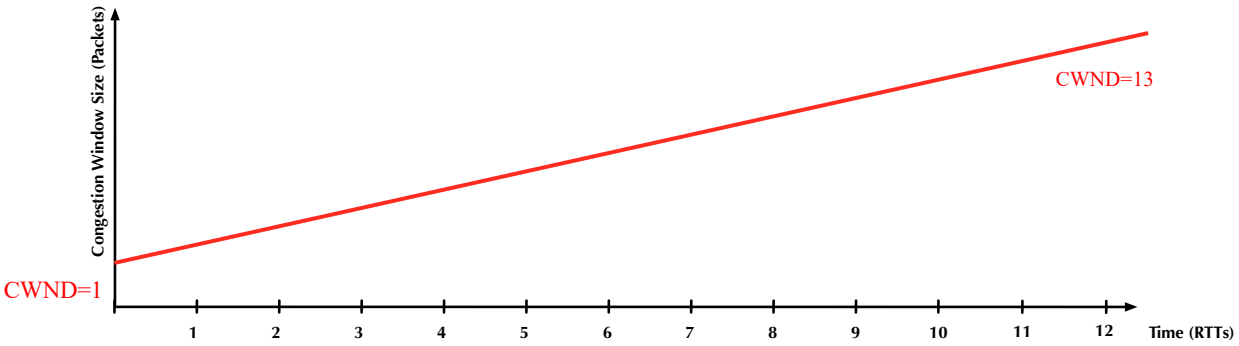
Here, we have ignored the transmission time for simplicity. If you have decided to include that in your calculations, the delay would be 4.7 seconds and the maximum congestion window size would be 13. We are going with the previous assumption here, but this would be also an acceptable solution (as long as you make the assumptions you have made clear).

3b)

Here, the congestion window goes from 1 to 12 without any drops. As soon as the 13th packet is injected to the network, we see a packet drop. We have assumed once the drop happens the source becomes aware of it in the next RTT. This is a simplifying assumption (and not accurate as source realizes RTTs based on sequences numbers). If you assume source recognizes the drop in the next RTT that is also acceptable.

3c)

Here we show the congestion window size at the end of each RTT (time 0 is when congestion window is set to 1, at the end of 1st RTT congestion window would be 2, and so on).
Depending on the assumptions you make about packet drops it is OK if that halving happens at time 12 or 13.



3d)

At the end of 1st RTT, we have one packet in flight, and buffer is empty. As the congestion window size grows up to 8, all packets will fill up the links. The buffer might temporarily hold one packet occasionally (when there are two packets back-to-back), but we can ignore that here. At the end of the 9th RTT, we have one packet in the buffer. From there on, the buffer occupancy grows by 1 packet every RTT till it becomes full (at the end of the 12th RTT). There will be a packet loss after that, but that is beyond 12 RTTs that we are considering in this problem.

3e)

We will solve this part in two different ways.

In the first approach, we find the delay of each packet and take the average. We do this for the first 12 RTTs as the problem states.

Without any queueing delay, each packet has three transmission delays and 3 propagation delays:
Link 1 (S to R) : 0.1 (transmission time) + 1 (propagation time) = 1.1
Link 2 (R to D) : 0.5 (transmission time) + 1 (propagation time) = 2
Link 3 (D to S) : 0.1 (transmission time) + 1 (propagation time) = 1.1

The total end-to-end latency here would be 4.2 seconds for each packet.

When the buffer is not empty, we will add 0.5 second of delay for each packet in the buffer. With that, the first 8 RTTs will have packets with delay 4.2. The 9th will have 4.7, 10th RTT will have a delay of 5.2, and so on.

We need to multiply the end-to-end latency of packets sent during that RTT with the number of packets during that RTT and sum up to get the total delay observed by all packets. Then, divide this value by the total number of packets.

Total delay: 1 x 4.2 + 2 x 4.2 + 3 x 4.2 + 4 x 4.2 + 5 x 4.2 + 6 x 4.2 + 7 x 4.2 + 8 * 4.2 + 9 x 4.7 + 10 x 5.2 + 11 x 5.7 + 12 x 6.2 = 382.6 seconds

Total number of packets: 1 + 2 + 3 + … + 12 = 78

Average delay = 382.6 / 78 = 4.9 seconds.

Slight differences with these numbers based on assumptions you have made in the previous part are still acceptable.

In the second approach (which is simpler but a bit less accurate) we do not average over all individual packets. We just measure the average queue occupancy (which is 0 for the first 8 RTTs and grows to 4 during the next 4 RTTs) and add that to the transmission and propagation delay measured above to measure average end-to-end latency. This is an approximation of the solution, but is considered an acceptable solution here.

4a)

P_avg_len = 0.01 * (50 - 30)/(70-30) = 0.01 * 20 / 40 = 0.005
P_drop = p_avg_len / (1-count x P_avg_len) = 0.005 / 1 = 0.005

4b)

P_avg_len = 0.01 * (65 - 30)/(70-30) = 0.01 * 35 / 40 = 0.0875
P_drop = p_avg_len / (1-count x P_avg_len) = 0.00875 / (1 - 50 * 0.00875) / 1 = 0.00875 / (1 – 0.4375) = 0.00875 / 0.5625 = 0.016

5a)

The IP address of web.cs.toronto.edu is 198.185.159.145 and port for HTTPS is 443.
Source IP: 1.2.3.1
Source Port: 2209
Destination IP: 198.185.159.145
Destination Port: 443

5b)

Source IP: 5.6.7.8
Source Port: 2209
Destination IP: 198.185.159.145
Destination Port: 443

5c)

Source IP: 198.185.159.145
Source Port: 443

Destination IP: 5.6.7.8
Destination Port: 2209

5d)

Source IP: 198.185.159.145
Source Port: 443
Destination IP: 1.2.3.1
Destination Port: 2209

5e)

Network Address Translation Table	
WAN Side Address (IP, Port #)	LAN Side Address (IP, Port #)
5.6.7.8, 2209	1.2.3.1, 2209