Computer Communications and Networks (COMN) 2022/23, Semester 1

Assignment 2 Results Sheet

Forename and Surname:	Andreas Hiropedi
Matriculation Number:	s2015345

Question 1 – Number of retransmissions and throughput with different retransmission timeout values with stop-and-wait protocol. For each value of retransmission timeout, run the experiments for **5 times** and write down the **average number of retransmissions** and the **average throughput**.

Retransmission timeout (ms)	Average number of retransmissions	Average throughput (Kilobytes per second)
5	454	63
10	243	65
15	117	66
20	**110**	**68**
25	112	57
30	110	56
40	109	56
50	104	48
75	105	41
100	106	36

Question 2 – Discuss the impact of retransmission timeout value on the number of retransmissions and throughput. Indicate the optimal timeout value from a communication efficiency viewpoint (i.e., the timeout that minimizes the number of retransmissions while ensuring a high throughput).

The optimal timeout value from a communication efficiency viewpoint is 20ms: this is because it ensures a high throughput (highest throughput actually based on the results), whilst ensuring a relatively low number of retransmissions (110, with only very high, inefficient values of retransmission timeouts doing a bit better in this regard).

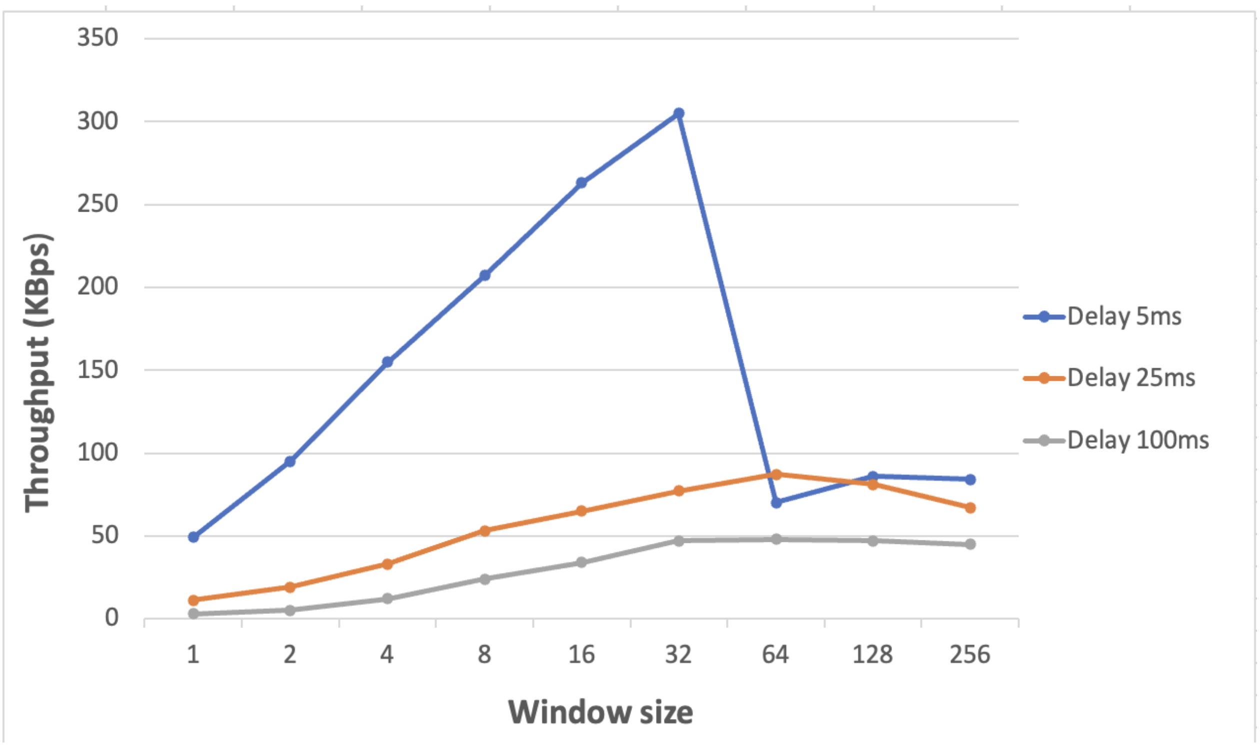
The pattern for the number of retransmissions that we notice here is similar to that of **diminishing returns**: up to an optimal point (namely 20ms), we notice a constant decrease in the number of retransmissions as we increase the retransmission timeout. However, beyond this optimal point, the values no longer decrease (the number of retransmissions seems to fluctuate within the range of 105-110). This is to be expected: as we allow more time before a retransmission occurs, it is normal that we get a much lower number of retransmissions than when we allow very little time (this would explain the first part, before the optimal point). Beyond the optimal point, we can assume that the larger timeout value has very little impact on the number of retransmissions, and hence why the observed values aren't significantly lower than the optimal 110.

This differs from the pattern for throughput, where we see that there is an **inverse diminishing returns relationship** between retransmission timeout and throughput: as the retransmission timeout increases, up until the optimal point of 20ms, we notice an increase in throughput. However, beyond that point, we notice that, as the retransmission timeout is further increased, the throughput decreases. This is to be expected: we are allowing for longer waiting periods (longer than optimal) before retransmitting packets, which would result in lower traffic in the network, and hence the lower throughput.

Question 3 – Experimentation with Go-Back-N. For each value of window size, run the experiments for **5 times** and write down the **average throughput**.

	Average throughput (Kilobytes per second)		
Window Size	**Delay = 5ms**	**Delay = 25ms**	**Delay = 100ms**
1	49	11	3
2	95	19	5
4	155	33	12
8	207	53	24
16	263	65	34
32	305	77	47
64	70	87	48
128	86	81	47
256	84	67	45

Create a graph as shown below using the results from the above table:



Question 4 – Discuss your results from Question 3.

Based on both the graph and the table, we see that there is an **inverse diminishing returns relationship** between window size and throughput: as the window size increases, up until the optimal point, we notice an increase in throughput. However, beyond that point, we notice that, as the window size is further increased, the throughput then decreases or fluctuates. This seems to be the case for all three delays (5ms, 25ms and 100ms).

However, one thing worth noting is that, as the delay is increased, the sharp change beyond the optimal point is less noticeable (for example, we notice a very sharp decline from a window size 32 to 64 for the 5ms delay, where 32 is the optimal window size; in the case of 25ms and 100ms delays, where 64 appears to be the optimal window size, we notice the decline between 64 and 128 to be a lot less steep).

Another interesting observation is that the optimal points differ between the three delays: as mentioned earlier, for a 5ms delay, 32 seems to be the optimal window size, whereas we have an optimal window size of 64 for 25ms and 100ms delays. This does, however, make sense: since we are allowing for larger delays, we can afford to send more packets as part of the same window (we expect the transmission to take longer due to increased number of packets, but the link delay should cover this, hence resulting in better throughput for higher window values).

Lastly, it is worth pointing out that, for each delay, a different retransmission timeout value was used. For the 5ms delay, we used the optimal value of 20ms retransmission timeout from question 1. For the 25ms delay, we used a retransmission timeout value of 50ms, and for the 100ms delay we used a retransmission timeout value of 200ms. This is because, based on the observed performance, a retransmission timeout value of roughly double the link delay would result in consistently good throughput relative to smaller/ larger retransmission timeout values (the data followed a similar pattern to the one observed in question 1).

Question 5 – Experimentation with Selective Repeat. For each value of window size, run the experiments for **5 times** and write down the **average throughput**.

	Average throughput (Kilobytes per second)
Window Size	**Delay = 25ms**
1	14
2	22
4	48
8	77
16	94
32	65

Question 6 - Compare the throughput obtained when using "Selective Repeat" with the corresponding results you got from the "Go Back N" experiment and explain the reasons behind any differences.

We notice that for smaller window sizes (up until 4), the throughput for "Selective Repeat" and "Go Back N" is very similar; this could be due to the very small window size, meaning that the efficiency of neither protocol can stand out, as both seem to require larger windows to observe any differences.

However, as we go up from 4 to 16, we notice that "Selective Repeat" starts to have much better throughput compared to "Go Back N". This is because the "Go Back N" protocol resends all packets within a window in case a packet is lost (not ACK-ed), whereas "Selective Repeat" only resends the packets that were lost (not ACK-ed) within that window. This results in less overall traffic on the link when "Selective Repeat" is used, and hence the higher throughput.

Lastly, for window size 32 (and potentially further on as well), we notice that "Selective Repeat" performs worse than "Go Back N". This could be because, for larger windows, the advantage of only retransmitting lost packets from "Selective Repeat" is no longer that effective, and it may take longer to identify the lost packets rather than just simply resending all of them, as is the case with "Go Back N".

Question 7 – Experimentation with *iperf*. For each value of window size, run the experiments for **5 times** and write down the **average throughput**.

	Average throughput (Kilobytes per second)
Window Size (KB)	**Delay = 25ms**
1	40
2	72
4	130
8	164
16	210
32	142

Question 8 - Compare the throughput obtained when using "Selective Repeat" and "Go Back N" with the corresponding results you got from the *iperf* experiment and explain the reasons behind any differences.

The throughput generated by the iperf experiment appears to be much greater than both "Selective Repeat" and "Go Back N". The optimal window size based on the iperf experiment would be 16, which is the same as "Selective Repeat", but not "Go Back N". This may be because the protocol rules implemented as part of "Selective Repeat" are more similar to those used by TCP, whereas the much simpler "Go Back N" uses a more distinct set of rules.

One reason for the difference in throughput could be that iperf adds twice the allocated value of packet size (the additional resources are used for administrative purposes and internal kernel structures). As a result, to make the comparison more fair, we should consider each throughput value in the table but halved. This would result in throughput values that are slightly closer to those of "Selective Repeat" and "Go Back N", but nonetheless better throughput values than either protocol.