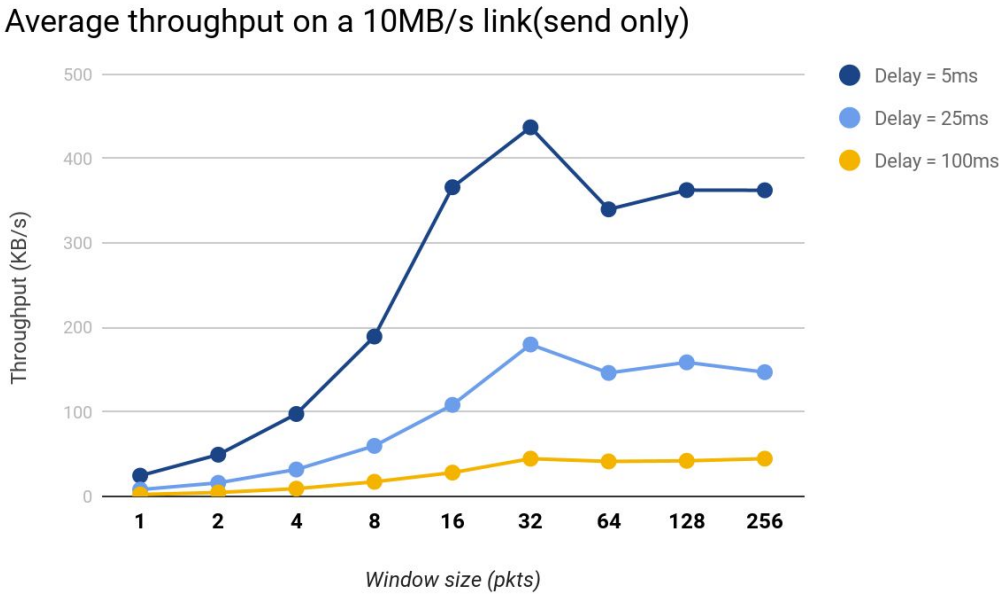
Computer Communications and Networks (COMN)
2017/18, Semester 1
Assignment Part 2 Results Sheet

Forename and Surname:	Michael Michaelides
Matriculation Number:	s1447836

Question 1 – Experimentation with Go-Back-N. For each value of window size, run the experiments for **5 times** and write down **average throughput**.

	Average throughput (Kilobytes per second)		
Window Size	**Delay = 5ms**	**Delay = 25ms**	**Delay = 100ms**
1	24.67	8.19	2.38
2	49.54	16.18	4.77
4	97.70	32.01	9.28
8	189.44	59.95	17.42
16	366.23	108.51	28.29
32	437.16	180.02	44.80
64	340.03	146.33	41.53
128	362.82	158.87	42.28
256	362.62	147.22	44.80



Question 2 – Discuss your results from Question 1.

First of all, as expected, the lower the propagation delay results in higher throughput because the packets arrive faster at their destination. Additionally, it seems that there is an optimal window w such that if window is less than w, then performance is suboptimal and if the window is greater than that, then performance drops again. In this case, the optimal window is 32 segments and each segment is approximately 1KB hence the optimal window is 32KB.

The problem with low windows is that the sender sends a few packets and then idles for as long as required till the acknowledgements arrive. For that time the link is idling and hence the link utilization is low.

On the other hand, the problem with large windows in the go back n protocol is the large amount of retransmissions. Since usually the whole window (from base to nextSeq-1) is resent for each packet loss, having large windows causes large amounts of traffic being retransmitted for each packet lost. This becomes inefficient because the sender wastes a large amount of time resending the window (and causes network congestion) and hence it requires more time to send the whole file and the throughput decreases.

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

	Average throughput (Kilobytes per second)
Window Size	**Delay = 25ms**
1	8.22
2	16.38
4	31.66
8	59.8
16	111.96
32	203.31
64	310.83
128	352.25
256	384.5

Question 4 - 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.

The difference is that the throughput with selective repeat keeps increasing while the window size and doesn't reach a maximum, unlike go back N that reaches a maximum and then throughput decreases again.

The reason is that a selective repeat client doesn't resend the whole window in the case of a single packet loss, it just resends that packet instead. For this, by increasing the window size we only make sure that the sender will idle less waiting for the acknowledgements and hence the link is used more efficiently. Any packet that is lost will be sent back independently, irrelevantly of the window size and hence there is no performance penalty for increasing the window size. As long as the sending rate doesn't exceed the capacity of the link, then there is no increase in packet loss and hence only the throughput increases. The maximum throughput will be reached either when the send rate approaches the capacity of the link or when the window size fits the whole file in one go, which is the best case because the sender will send all the file in one go and then just resend individual lost packets.

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

Window Size (KB)	Average throughput (Kilobytes per second) Delay = 25ms
1	14.15
2	17.5
4	38.36
8	61.34
16	73.28
32	91.2

Question 6 - 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 results are very similar up to the window of 8KB in all 3 cases, however then both selective repeat and go back n methods double up the throughput at the window of 16KB, in contrast to the case above that just keeps increasing steadily. After that, both the first two cases keep increasing but go back n starts decreasing after the window of 32KB. However, GBN and SR greatly outperform the performance obtained by iperf with TCP. The reason is that TCP has overhead to accommodate for few extra mechanisms, apart from just reliable transmission, that are slowing down the transmission.

First of all, the TCP header is much larger than the header of 3 bytes we used and hence iperf is sending larger amounts of data.

The main reason why TCP is slow though is because of a mechanism called congestion control, used to treat congestion on the network. TCP starts slow, setting the window to 1MSS and then doubling it each time until a threshold and then it just increases the window by 1MSS each time. Hence, it doesn't utilise the maximum window capacity as my implementations do all the time and this makes TCP slow. Another reason is that TCP takes drastic actions on packet loss or triple duplicate ack by either halving the window or setting it to 1MSS and steadily increasing it. This slows down the transmission but ensures that congestion on the network is treated quickly, something that doesn't affect us in this case and it only slows down the transmission. With my implementations, on packet loss the sender keeps sending at maximum speed and hence it has an advantage.