Computer Communications and Networks (COMN)
2016/17, Semester 2

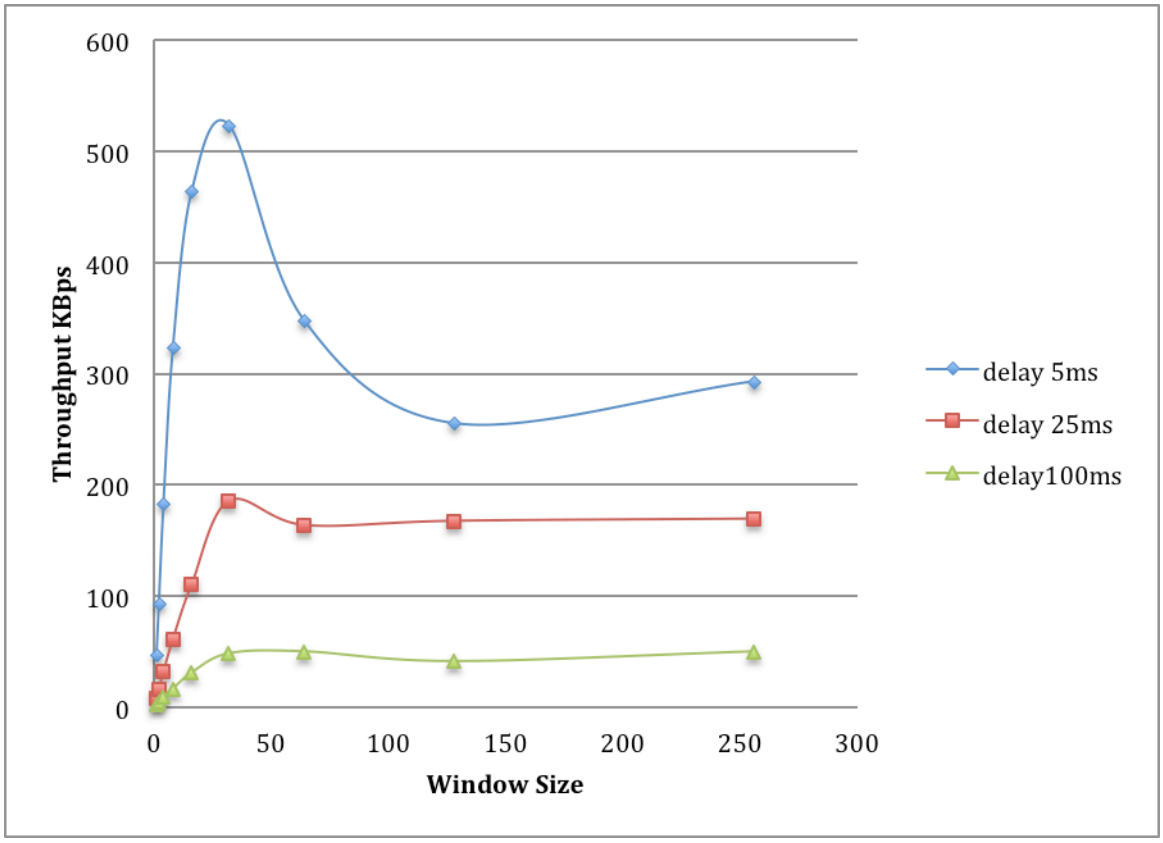
Assignment Part 2 Results Sheet

Forename and Surname:	Isabella Chan
Matriculation Number:	s1330027

Question 1 – Experimentation with Go-Back-N:

	Throughput (Kilobytes per second)		
Window Size	**Delay = 5ms retryTimeout = 20ms**	**Delay = 25ms** retryTimeout = 100ms	**Delay = 100ms retryTimeout = 400ms**
1	47.11033333	8.2007	2.4555
2	93.019	16.19	4.87
4	183.7703333	31.96566667	9.7385
8	323.2296667	61.08	16.665
16	463.736	110.295	31.26
32	522.9853333	175.3733333	48.48666667
64	347.6386667	163.59	50.21
128	255.2633333	167.3333333	41.4445
256	293.041	169.3333333	50.175

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



Question 2 – Discuss your results from Question 1.

The plot shows that smaller delays result in greater throughput, which makes sense because it takes shorter time for a packet to get from the sender to the receiver. There is a correlation between throughput and window size. For smaller delays, there is an obvious optimal value for window size in order to maximize the throughput value. This is because the window size has to be carefully chosen such that there won't be too many packets in the window when the base packet's ack is missing, thus reducing the amount of retransmissions.

For larger delays, the optimal window size is not so obvious as throughput increases as window size increases and than saturates. It takes longer for an ack packet to arrive to the sender, and window will be filled quickly, preventing more packet transmission. Throughput is much smaller for longer delays because of that.

The performances saturate because the rate of packet loss remains the same, such that once a certain number of packets have been sent and stored in the window and retransmitted, the missing ack should arrive before causing anymore retransmission. Therefore the throughput becomes constant once the window size gets too big.

It shows that GoBackN is efficient for shorter delays with a window size of around 32. Effectiveness of this protocol reduces as delay increases and window size increases.

Question 3 – Experimentation with Selective Repeat

Window Size	Throughput (Kilobytes per second) Delay = 25ms retryTimeout = 100ms
1	7.34
2	14.287
4	28.37845
8	54.6455
16	101.965
32	187.8

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.

Selective repeat should have a higher throughput than GoBackN's since it is more efficient by not retransmitting all packets every time time out occurs. The results show that at 32 window size, the throughput with SR is higher than GBN, which is because window size does not matter in SR now since it retransmit only individual packet but not all packets in the window.

The results still show that throughput doubles as window size increases, although SR should have a generally higher throughput than GBN. The results in this table might be affected by the way I implemented the timers. However, the result with larger window size is still apparent.

The throughput for SR should steadily increase and window size does not matter. Using too small a window size might hinder the performance since less packet can be sent/ack'd.

Question 5 – Experimentation with *iperf*

Window Size (KB)	Throughput (Kilobytes per second) Delay = 25ms
1	14.125
2	16.75
4	39.376
8	67.125
16	92.25
32	69.375

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 speed of transfer for TCP is slower than UDP (SR and GBN). It has an optimal window size of 16 KB. TCP is slower because it is reliable, a connection orientated protocol and ordered, and thus there is additional overhead and processing time. It is reliable means it provides error detection, congestion control and retransmission of lost packets. It is a connection-orientated protocol means that a connection or socket must first be established before data can flow and data travels both ways. It is ordered means that it uses sequence numbers to ensure that packets are reconstructed in the correct order.