Computer Communications and Networks (COMN) 2019/20, Semester 2

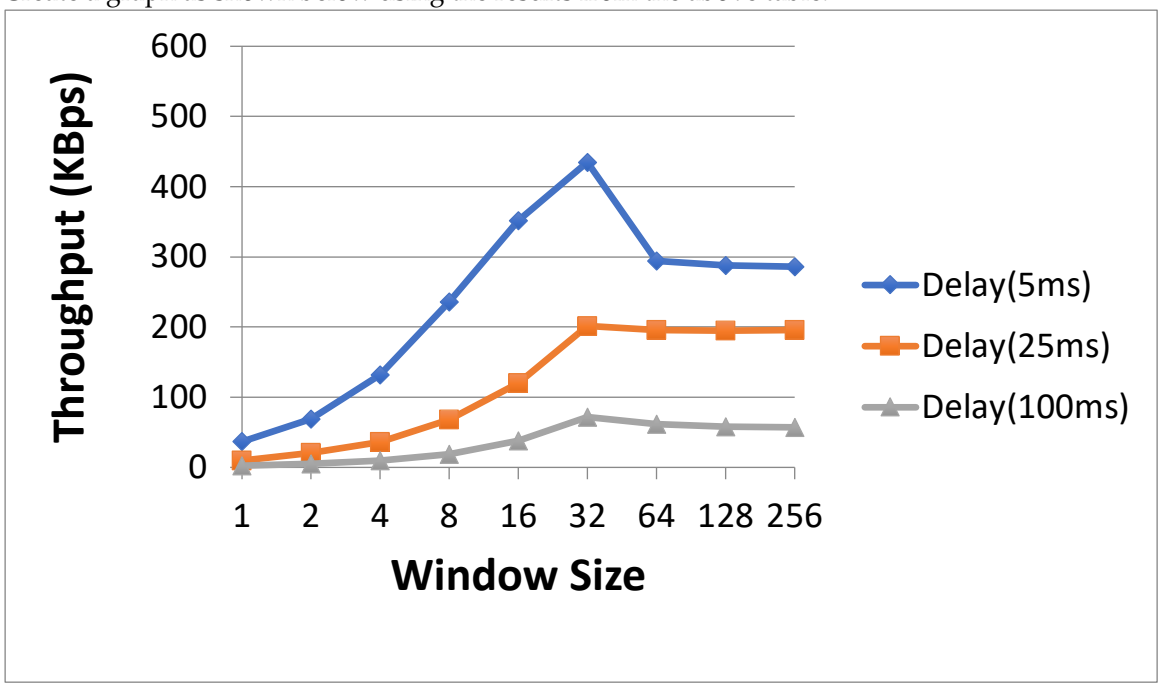
Assignment Part 2 Results Sheet

Forename and Surname:	Billy Byiringiro
Matriculation Number:	S1794094

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	36.66	9.74	2.45
2	68.58	20.87	4.72
4	131.62	36.06	9.82
8	235.42	68.24	19.23
16	352.05	120.17	37.98
32	434.91	201.55	71.75
64	294.34	196.09	61.44
128	287.48	194.56	58.18
256	286.10	195.64	57.02

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



Question 2 – Discuss your results from Question 1.

First of all, I set the timeout values for the two cases, 25ms and 100ms, to 60ms and 210ms respectively. For the 25ms case, the round-trip delay is equal to 50ms and as the packets are arriving in bunches, I assumed the node processing delay may be more from the one I assumed from the first coursework, therefore, I set to 10ms, so the total timeout becomes 50ms roundtrip delay plus 10ms for nodal-processing. The same applies for 100ms case. These minimizes the number of retransmissions and increases throughput.

Analysis for my results from Question 1:

From all cases the throughput seems to increase by a factor of 2 as the window size increase by the power of 2 until window size reaches 32 then it becomes constant or slightly decreases. This justifiable because when the window size increases, we are making most out the specified bandwidth:10Mbits/s. However, as the window size becomes larger, the increase of retransmission increases too, that is, when a single packet is lost (as we are using "Go Back N" (GBN) algorithm, the packets after it are retransmitted. That's why, the window size of 32 is providing highest throughput as it is good enough to carry enough packets and also small enough to avoid retransmission of too many packets. The other reason, as DummyNet link queue size is set to 50 by default, when window size if greater than 50, there is a chance that the queue might be full and drop some packets, which in turn cause decrease of throughput for window size 64, 128, and 256.

Another thing to note is how a slight deference in delays -- 5ms, 25ms, 100ms -- highly affects throughput, as there is slow response time between sender and receiver. And these delays affect the set timeout delay which direly affect GBN as it uses cumulative acknowledgement.

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	9.24
2	19.51
4	37.74
8	70.24
16	131.89
32	248.59

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.

Both algorithms' throughput is almost the same when the window size is small, but as the window size increases, "Selective Repeat" outperforms "Go Back N" algorithm. Which is reasonable because as the window size increases, the number of retransmissions highly increases for "Go Back N" as it uses cumulative acknowledgment where all packets are retransmitted if a packet with a sequence number that precedes theirs is lost , while for "Selective Repeat", only the lost packets is retransmitted.

I also would like to mention that 'Selective Repeat' throughput seems to be more stable on a different window size. While for GBN, the throughput fluctuates as it depends of the position of a lost value in a sent window, and that location is random.

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	13.12
2	22.50
4	47.25
8	65.37
16	83.75
32	98.73

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.

Iperf transmission throughput is slightly better than both "Selective Repeat" and "Go back N" when the window size if less than 8, and starts increasing slowly as the window size increases, hence being outperformed by both algorithms.

It seems to behave like "Selective Repeat" and "Go back N" when window size is small than 8: doubling the throughput as the window size increases by factor of 2. But is start increasing by roughly the previous increase then by the factor of ½ of the previous value. The reasons being, by default, Iperf uses TCP and that trends looks very similarly to how TCP works, TCP first increases the window size until a packet is lost, then halve by 2, then keep it constant and adjust it once in a while, which also enforces congestion control of the TCP. The is because TCP adds various headers on packets and have to do checksum and set up and end connection, which is something we didn't not implement is GBN and SR which solely uses UDP.