Computer Communications and Networks (COMN) 2021/22, Semester 2

Assignment 2 Results Sheet

Forename and Surname:	Weiyu Tu
Matriculation Number:	s1820742

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 **average number of retransmissions** and **average throughput**.

Retransmission timeout (ms)	Average number of re-transmissions	Average throughput (Kilobytes per second)
5	1688.8	65.365
10	970.8	62.478
15	125.4	56.032
20	111.4	53.375
25	98.6	50.818
30	99.2	52.278
40	101.2	41.986
50	95.8	34.032
75	105.2	28.941
100	103.6	25.503

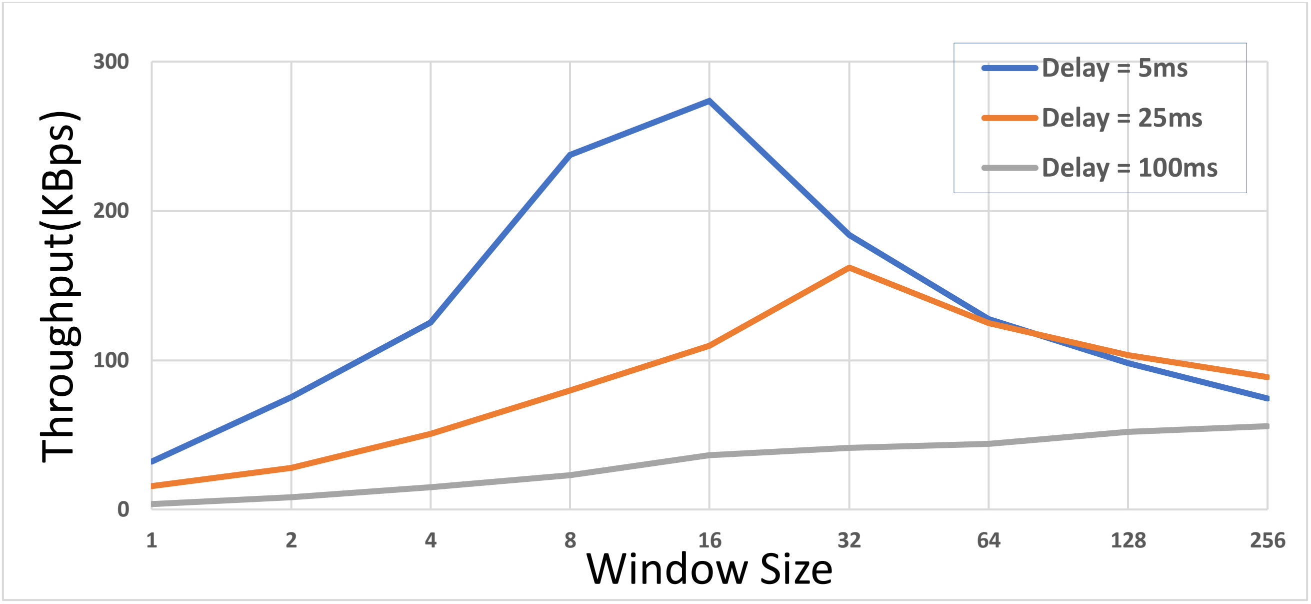
Question 2 – Discuss the impact of retransmission timeout value on the number of retransmissions and throughput. Indicate the optimal timeout value from a communication efficiency viewpoint (i.e., the timeout that minimizes the number of retransmissions while ensuring a high throughput).

When the transmission timeout values are 5ms and 10ms, the number of retransmissions is dramatically high. They have a very high probability since the long RTT takes up most of their timeout limits. For example, when retransmission timeout is 5ms while the average RTT is set to 10ms, the packages spent a long time on the way to the destination and easily to be timed out even if they have not arrived yet. The case is further worsened since timeouts will trigger more retransmission packages and more duplicate ACKs. The situation is mitigated when the average retransmission timeout is once larger than the RTT: the average number of re-transmissions dropped to 125.3 from 970.8 as retry time limitation extends from 10ms to 15ms. However, as a 5% number of packets are lost during transmission, even when the timeout value goes larger, the throughput is reduced because of the poor utilisation rate. For example, when retransmission timeout is 50ms, the observed average number of retransmissions divided by the total number of packages transmitted is 95/(879*2)=0.054 which is close to the TC 5% lose rate. Through this experiment, the optimal timeout value is 25ms as it has a relatively low number of retransmission and high throughput.

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

	Average throughput (Kilobytes per second)		
Window Size	**Delay = 5ms**	**Delay = 25ms**	**Delay = 100ms**
1	32.074	15.658	3.610
2	75.439	27.868	7.998
4	125.302	50.490	14.745
8	237.259	79.738	22.774
16	273.657	109.786	36.340
32	183.954	162.018	41.250
64	127.579	124.650	43.814
128	97.968	103.264	51.887
256	74.397	88.816	55.799

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



Question 4 – Discuss your results from Question 3.

When the timeout value is fixed to Q2's optimal which is 25ms when the delay is 5ms. When the delay time is 25ms and 100ms, I choose 55ms and 205ms as timeout respectively.
Theoretically, if the RTT (one way delay*2) time is larger than the timeout limit, the packets could easily be delicately transmitted. The actual packets' transmission time can be monitored by the using ping tool. The time used for 64bytes-packets from localhost is about 50ms-55ms for 25ms delay cases and 200ms-205ms when the delay is 100ms which proves the assumption above.
Redundant packets are also regarded as a kind of pollution to the network. So, I adjust the retransmission timeout value at least two times larger than the network delay. However, the retransmission time cannot be too large as the truly lost packages are queueing for retransmitting. The utilization of the channel is also considered.
As shown in the picture, the throughputs under different RTT delay conditions with small window sizes are low. The throughputs with very large window sizes are not high enough as well. The throughput reaches the peak value when window size equals 16 when the delay is 5ms, 32 when the delay is 25ms. For the case of a 100ms network delay, even the throughput is slightly increasing, the throughputs of whatever window size are not optimum. From the essence of the invention of the sliding window, it is used to solve the problem of low-utilized networks and there should exist an optimal window size for different delay times. In general, as the delay time grows larger, the optimal window size grows larger as well. However, just like the 100ms delay case, under the serious network congestion, Go-Back-N cannot help any. The small window size will increase the duration of sending out all their packages while the large window size leads to a huge retransmission workload even there is only a single packet loss.

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

	Average throughput (Kilobytes per second)
Window Size	**Delay = 25ms**
1	16.243
2	31.770
4	89.363
8	183.816
16	254.601
32	386.715

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.

When window sizes are small, both SR and GBN performs similarly as they undergo similar logic. A small number of duplicate transmissions of all packets within the window does not affect the network or let the network to be congested. However, SR shows its advantage in the sizes of the sliding-windows bomb. In GBN, once the retransmission is triggered because there is a single package loss, packets within the window are all required to transmit again even others have been already acknowledged by the receiver. When GBN's network bandwidth is fully utilized, for example, when window size equals 64, the duplicate packets on the channel seriously reduce the throughputs, the transmission rate of SR is still optimal. There won't be any redundant ACK and packets implicated by a single packet transmission timeout. Therefore, SR should transmit faster than the GBN.

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

	Average throughput (Kilobytes per second)
Window Size (KB)	**Delay = 25ms**
1	13.63
2	25.63
4	26.00
8	57.50
16	79.13
32	94.20

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 average throughput by using *iperf* for every window size is smaller than both GBN and SR when delay=25ms.
Without only considering file integrity, introducing a reliable transmission under UDP protocol, TCP includes more services such as handshaking, congestion control and flow control which trades off the speed of file transfer for a stable and secure connection. TCP performance depends strongly on the congestion mechanisms that slow down the segment transmission when a segment is lost. What's more, our problem is packet-oriented. In theory, UDP is more efficient if applications send a very small amount of information sporadically just like our problem, as setting up a TCP connection takes time and is useless if the message can be carried in a single IP packet. Facing small losses, for example, our *tc* setting (5%-10% loss rate), TCP cannot distinguish whether it is a package loss or traffic congestion and put much effort into congestion control when UDP simply recovers the data by duplicate package transmissions quickly. Hence, GBN or SR behaves better than *iperf* in this case.