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

Assignment 2 Results Sheet

Forename and Surname:	Andy Fu
Matriculation Number:	1923846

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	1852.0	77.3
10	980.0	76.7
15	95.2	76.0
20	95.4	72.6
25	90.8	70.6
30	94.2	67.4
40	101.8	62.6
50	92.2	59.8
75	90.6	51.9
100	94.4	44.8

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).

My implementation does not have a rate restriction on sending packets, as I found that stop-and-wait does not cause traffic congestion when the packet is small and the loss rate is 5%.

The RTT for this section is set to 10ms, the actual average time between sending a packet and receiving the ack is slightly more than 10ms since there are other kinds of delay including the processing time. Therefore, when we are using 5ms and 10ms as the retransmission timeout, the sender usually cannot receive the ack before the timeout, thus the average numbers of retransmission are high. Note when the retry timeout is

5ms, the sender could resend more times (twice compared to 10ms) before receiving the ack, so the retransmission number is the highest.

For retransmission timeout longer than 15ms, the average number of retransmissions drops to between 90 to 100, and the number remains roughly the same if the retransmission timeout keeps increasing after 15ms, since the RTT is 10ms, there is no difference for waiting longer. Note the minimum expected number of retransmissions is around 88, since there are 879 packets, there is a 5% of loss rate in both directions (sending packets and sending acks) this number can only be achieved under a very ideal network condition.

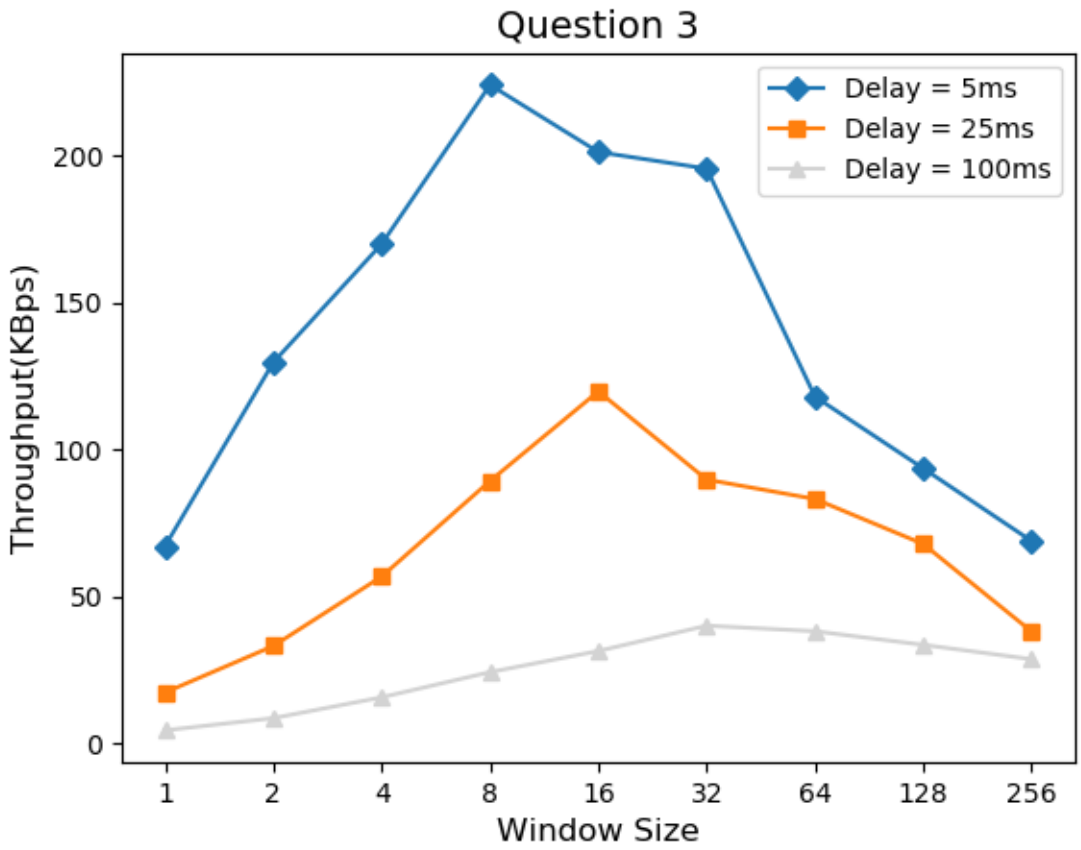
With 5ms and 10ms timeout values, the sender sends redundant packets before receiving the ack. It is less likely to lose multiple packets at the same time; therefore, the sender is less likely to wait longer than 2 RTTs due to the packet loss, sender with small timeout values tends to be faster. Apart from that, a short timeout also means the sender wait and wastes less time before resending, so increasing the timeout will decrease the throughput in general.

If we are in a network with sufficiently high bandwidth, resending redundant packets will not cause congestion or slow down the transmission, then it might be feasible to consider using a 5ms timeout to speed up the transmission. However, in reality, the bandwidth is often limited, thus a retry timeout that is slightly longer than the RTT is the most optimal one. In this case, 15ms is the best timeout value.

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	72.9	17.3	4.4
2	129.6	33.1	8.6
4	169.8	56.7	15.6
8	223.9	89.2	24.2
16	201.1	119.7	31.4
32	195.5	89.7	40.0
64	117.2	83.1	38.1
128	93.5	67.8	33.5
256	68.8	38.0	28.6

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



Question 4 – Discuss your results from Question 3.

I chose 15ms, 55ms, and 205ms as retransmission timeout for propagation delay of 5ms, 25ms, and 100ms, the idea is to choose values slightly larger than the total delay, since the total delay is slightly longer than 2 * propagation delay (I have observed using ping command), RTT plus 5ms is safe to use.

In general, the propagation delay affects throughput directly, longer propagation delay and timeout will make the sender to wait longer before knowing whether the retransmission is required. Since the file of 879kB can be sent within 1 second if the loss rate is 0 and bandwidth is 10mbit/s (if we strictly limit the transmission rate), wasting 205ms for a packet loss will slow down the transmission significantly.

When the window size is 1, the go-back-n protocol is the same as the stop-and-wait protocol, when the window size increases, the sender could send more packets after the timeout, thus the ratio of the packet sent vs. time spent on waiting for ack is higher, the throughput is higher. But when the window size is larger than a threshold, the probability of packet loss within the window increases, and the timer is more likely to be reset due to the acknowledgement of previous packets. For example, if the nth packet in the window is lost, but all previous n-1 packets are fine, the sender will reset the timer after receiving every single previous ACK,

thus more time is wasted before knowing the whether the retransmission is required, the throughput will drop if the window size is too large.

Moreover, if we do not restrict the rate of sending packets, a larger window size is likely to cause traffic congestion, and therefore reduce the throughput. But if we restrict the rate of sending packets, there would be no need to set a big window size. For instance, if we want to send 256 packets (256KB) within 15ms, the rate of sending packets exceeds the bandwidth; If we always send 256 packets within 205ms (10mbit/s), but set the timeout value as 15ms, the last packet in the window will never be sent before the first timeout as long as there is a single packet loss.

The threshold of window size is dependent on the propagation delay and loss rate. According to the graph, the optimal window size is 8 when the delay is 5ms, the optimal window size is 16 and 32 when the delay is 25ms and 100ms. Please note the throughput of the GBN protocol is highly unstable, and the average throughput may vary wildly (5 repeats are not sufficient to give a good statistical conclusion). For example, if successive packets are lost, the sender waits a shorter time before resending.

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	17.2
2	32.7
4	58.9
8	101.1
16	145.9
32	234.1

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 the window size is small (1, 2, 4), Selective-Repeat (SR) and Go-Back-N perform equally well, but Go-Back-N (GBN) performs better than Selective-Repeat when the window size is larger, the larger the window size is, the more Selective-Repeat outperforms Go-Back-N.

Because when a packet is lost, the SR protocol can still send other packets with larger sequence number, but the GBN stops and wait for a timeout. Moreover, since the SR protocol has a separate logical timer for

each packet, the timer will not be reset because of the ACKs for other packets, the sender will retransmit packets sooner. While the GBN usually takes longer before reacting to a packet loss.

The SR protocol will utilize the bandwidth better as packets are buffered at the receiver side, but the GBN receiver will drop many packets because of a single packet loss. Thus, given a fixed bandwidth, the SR protocol could transmit files with higher efficiency (higher throughput).

Because of the nature of the GBN protocol, its throughput drops when the window size is too large, however, the throughput of the SR protocol can keep increasing as the window size increases, as long as there are sufficient bandwidth and buffer size.

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	11.8
2	23.5
4	26.3
8	57.9
16	93.7
32	131.3

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 of the iperf experiment shows a similar increasing trend as the Selective-Repeat protocol, but the iperf is slower, unlike the Go-Back-N protocol, the throughput did not drop if we increase the window size from 16 to 32.

The reason is that iperf is using a TCP connection, it spent time on the TCP three-way-handshake, and wasted time on error detection after each packet arrives (since we have assumed no bit error), but my SR protocol and the GBN protocol transmit directly without any handshake or checksum. Therefore, given a small window size (0 to 16) the iperf always has a smaller throughput compared to the two other protocols.

However, the TCP receiver does not drop the received packet because of a single packet loss, so its performance does not drop if we continue to increase the window size, eventually, the iperf will have a

higher throughput compared to the GBN protocol if the window size is large (>= 32). But theoretically, the iperf will never have a greater throughput compared to the SR protocol, as this is a trade-off between transmission efficiency and reliability, the TCP is much more reliable than my SR implementation (especially when there are bit errors).