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

Assignment 2 Worksheet

Forename and Surname:	Emir Ersanli
Matriculation Number:	S221285

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	1912	45.27
10	1115	48.83
15	261	58.23
20	154	59.42
25	97	57.75
30	98	57.43
40	96	53.68
50	89	52.4
75	95	45.76
100	91	34.88

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).

In examining the impact of retransmission timeout on the number of retransmissions and throughput, we observe a distinct relationship. As the timeout value increases, the average number of retransmissions tends to decrease sharply up to a timeout of 15 ms, beyond which the decrease stabilizes. This indicates that shorter timeouts lead to premature retransmissions, while timeouts above 20 ms do not contribute to significant further reductions in retransmissions, likely due to the packet loss rates and network delays.

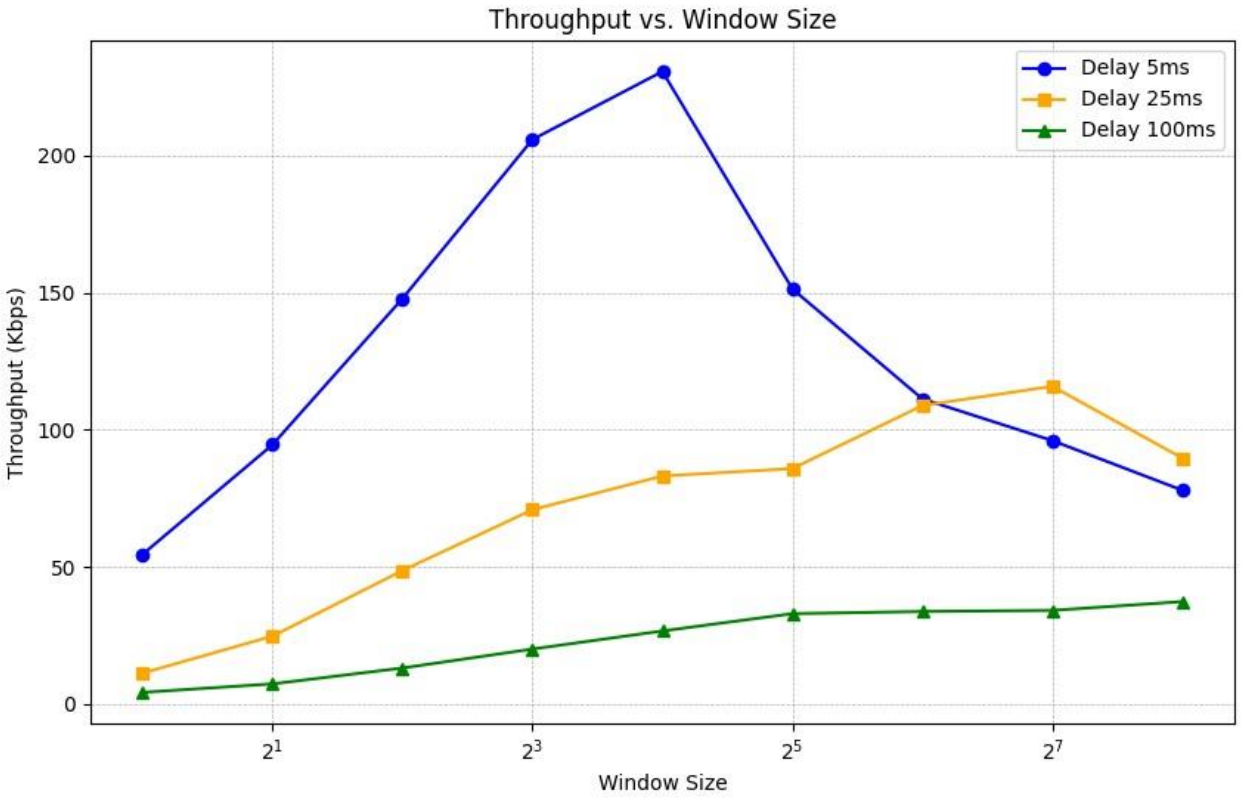
Regarding throughput, there is an increase up to a timeout of 20 ms, after which it begins to decrease. This pattern suggests that up to a certain timeout value, the protocol benefits from waiting slightly longer for acknowledgments, which reduces unnecessary retransmissions and utilizes network resources more efficiently. However, setting the timeout too high causes the protocol to wait excessively before retransmitting a lost packet, leading to lower throughput due to increased latency.

Considering both the average number of retransmissions and throughput, the optimal retransmission timeout value for communication efficiency in my data is found to be between 15 and 20 ms. This range provides a balanced trade-off, reducing retransmissions substantially while maintaining high throughput.

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	54.6	11.3	4.3
2	94.6	24.8	7.4
4	147.9	48.7	13.18
8	205.9	70.9	20.1
16	230.7	83.2	26.7
32	151.4	85.9	33.0
64	111.3	108.9	33,8
128	96.1	115.9	34.2
256	78.0	89.7	37.4

Create a graph using the results from the above table (empty example gvraph shown below):



Question 4 – Discuss your results from Question 3.

In the experimentation with the Go-Back-N protocol, various window sizes and one-way propagation delay values were tested to observe their impact on the throughput. The retransmission timeouts were chosen to be larger than the round-trip time (RTT) to avoid premature retransmissions, which could result in unnecessary traffic and reduce throughput efficiency.

For a delay of 5ms, the retransmission timeout was set to 20ms. This value is based on the findings from the previous part of the project, where it was determined to be the optimal timeout for the given conditions. This setup aims to ensure that the sender waits sufficiently long for acknowledgments before deeming the packets as lost. The resulting data shows that the throughput increases as the window size grows, peaking at a window size of 16. This indicates that with a small propagation delay, a moderately sized window allows for efficient utilization of the network, with multiple packets being in transit, which increases throughput.

However, as the window size continues to increase beyond 16, the throughput begins to decrease. This decrease can be attributed to the receiver being unable to process incoming packets quickly enough, causing the sender to wait for acknowledgments and thereby reducing the window's effectiveness.

When the propagation delay increases to 25ms and 100ms, the retransmission timeout values were chosen as 60ms and 205ms, respectively. These values ensure that the timeout is significantly greater than the propagation delay, accounting for the time it takes for a packet to make a round trip, plus some additional processing time. With the increased delay, the window size at which the throughput peaks also increases, being 32 and 64, respectively, for the two delays. This phenomenon can be explained by the increased delay, which allows for a larger number of packets to be in flight without overwhelming the receiver. As such, a larger window size becomes more suitable for higher delays. The throughput peaks at different window sizes for different propagation delays, illustrating the importance of selecting an appropriate window size relative to the delay. For lower delays, smaller window sizes are more effective, while for higher delays, larger windows are needed to maintain a steady flow of packets without overburdening the receiver. This balance is crucial for optimizing throughput in Go-Back-N protocols.

In conclusion, my results align with the principles of window-based transmission protocols. The choice of window size relative to the propagation delay is a critical factor in optimizing throughput. The optimal window size tends to increase with higher propagation delays, and there is a peak throughput that can be achieved by tuning the window size appropriately. Too small a window underutilizes the network, while too large a window can overwhelm the receiver and degrade performance.

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

Window Size	Average throughput (Kilobytes per second) Delay = 25ms
1	13.5
2	26.8
4	44.6
8	86.5
16	141.8
32	185.2

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.

From my Go-Back-N trials, a notable peak throughput of 230.7 kilobytes per second was observed at a window size of 16. However, as the window size further increased, the throughput decreased, highlighting a tipping point beyond which the efficiency of Go-Back-N degrades. This decline can be linked to the protocol's fundamental limitation where, upon detecting a single packet loss, all subsequent packets in the window are retransmitted, even if they were received successfully by the receiver. This leads to an unnecessary retransmission of packets, thereby congesting the network and reducing the effective throughput.

In contrast, the Selective Repeat protocol displayed a steady increase in throughput without a decline within the tested window sizes, achieving a throughput of 185.2 kilobytes per second at a window size of 32. This increase can be attributed to the protocol's mechanism of selectively retransmitting only those packets that are detected as lost or received out of order. By avoiding redundant retransmissions, Selective Repeat makes better use of the network's bandwidth and maintains a higher throughput across larger window sizes.
This difference is further accentuated by the fact that Selective Repeat can handle out-of-order packet arrival more efficiently. Since it doesn't require packets to be processed in a strictly sequential order like Go-Back-N, it's able to maintain higher throughput levels under conditions where packet ordering and loss are prevalent, such as in networks with higher latency or variability.

Moreover, the implementation strategies adopted for each protocol in my experiments could have influenced the results. The use of non-blocking sockets in Go-Back-N might be less efficient in handling retransmissions compared to the multi-threaded approach of Selective Repeat. Multi-threading allows for concurrent processing of packet acknowledgments and retransmissions, which can lead to a more responsive and fluid handling of network events, thus optimizing throughput.

In summary, the higher throughput observed with Selective Repeat in my experiments is primarily due to its more efficient packet handling strategy that minimizes the retransmission of successfully received packets, complemented by the protocol's ability to handle out-of-order packets without the need for sequential processing. The multi-threaded implementation likely contributes to this efficiency by enhancing the protocol's responsiveness to network events

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

Window Size (KB)	Average throughput (Kilobytes per second) Delay = 25ms
1	13.5
2	23.8
4	37.3
8	64.8
16	82.2
32	100.2

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.

Selective Repeat displays higher throughput compared to iperf in the 25ms delay scenario due to protocol differences. While Selective Repeat transmits over UDP, which doesn't establish a connection or employ congestion control, iperf uses TCP, which includes a handshake process and adjusts its rate upon detecting packet loss. These TCP features ensure reliability but can reduce throughput under conditions of high latency or loss, which explains why Selective Repeat outperforms iperf at larger window sizes in these conditions.