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

Assignment 2 Worksheet

Forename and Surname:	Jacob Inwald
Matriculation Number:	s2150204

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	2422	79.07
10	1535	78.39
15	93	73.13
20	90	69.62
25	91	68.42
30	92	66.52
40	104	60.23
50	93	58.91
75	90	51.03
100	101	42.8

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).

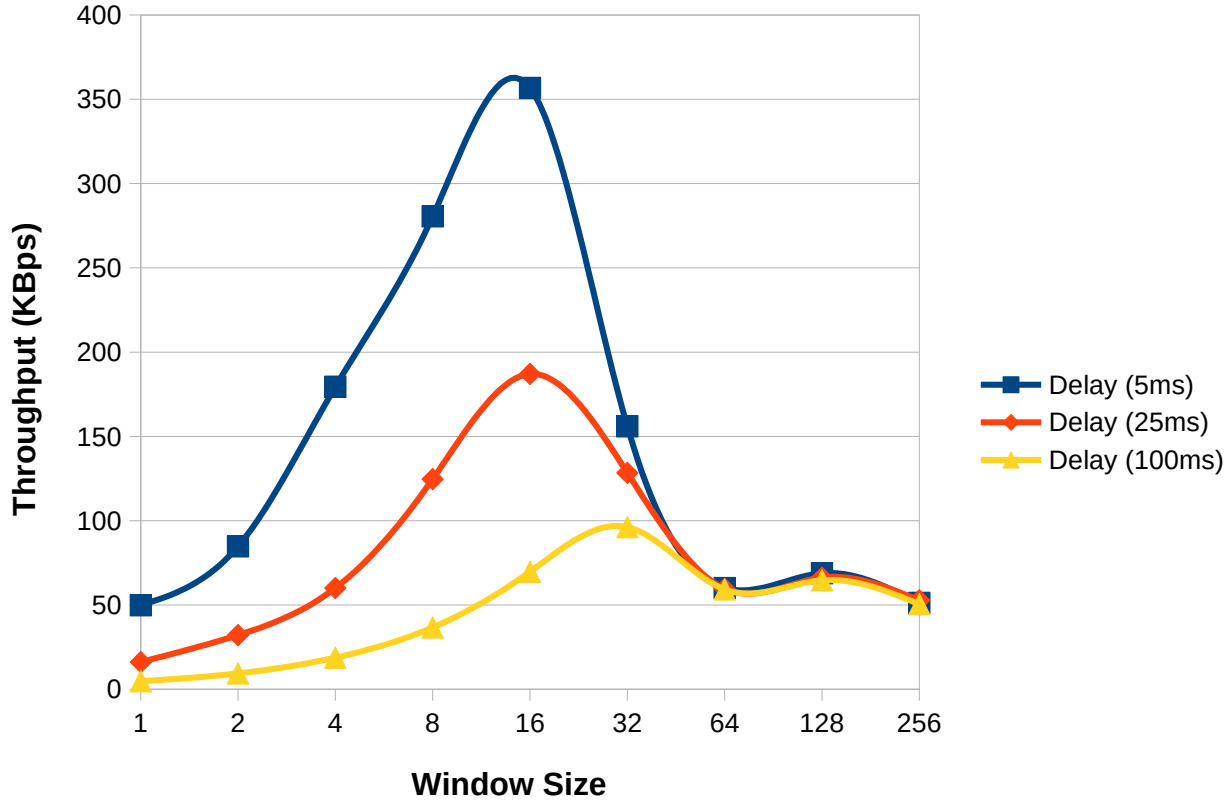
The retransmission timeout value significantly affects both the number of retransmissions and throughput in the stop-and-wait protocol. As the timeout value increases from 5ms to 25ms, the number of retransmissions drops dramatically (from 2422 to 91), demonstrating that shorter timeout values can cause unnecessary retransmissions when packets or ACKs are simply delayed rather than lost. However, after the 25ms mark, the number of retransmissions stabilizes, indicating that most legitimate packet losses are being captured without premature retransmissions. Throughput decreases as timeout increases likely because longer timeouts mean the system waits longer before retransmitting genuinely lost packets.

From a communication efficiency viewpoint, the optimal retransmission timeout value appears to be around 15-20ms. At 15ms, throughput is high (73.13 KBps) while retransmissions have already dropped significantly to 93. At 20ms, retransmissions reach their minimum (90) with a still-competitive throughput of 69.62 KBps. A timeout value of 20ms appears to provide the lowest retransmission rate, while still ensuring a high thoughput, making it the most efficient choice for this network configuration.

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	49.84	16.15	4.73
2	84.79	32.13	9.40
4	179.45	60.06	18.73
8	280.52	124.69	36.66
16	356.77	187.04	69.69
32	155.98	128.41	96.19
64	60.06	59.64	59.25
128	68.84	66.27	64.65
256	51.26	52.77	50.51

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



Question 4 – Discuss your results from Question 3.

The results from Question 3 demonstrate several key behaviors of the Go-Back-N protocol under different propagation delay conditions. For all three delay settings, throughput initially increases with window size, reaches a peak, and then declines. The optimal window size varies based on delay. For 5ms delay, peak throughput occurs at window size 16 (356.77 Kbps). For 25ms delay, peak throughput also occurs at window size 16 (187.04 Kbps). For 100ms delay, peak throughput occurs at window size 32 (96.19 Kbps).

Larger propagation delays result in lower overall throughput across all window sizes, which is expected since the time required for acknowledgments to return increases. The significant drop in throughput after the optimal window size for all propogation delays can be attributed to the inherent inefficiency of Go-Back-N when packet loss occurs. With larger windows, a single packet loss results in retransmission of more packets, causing network congestion and reduced efficiency. For larger delays, the optimal window size increases because a larger window allows more packets to be "in flight" during the longer round-trip time, making better use of the available bandwidth.

The results show that optimal window size should increase proportionally with delay to maximize channel utilization. However, due to the inefficiencies of Go-Back-N with packet loss, extremely large windows eventually become counterproductive regardless of delay.

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	15.46
2	31.82
4	60.32
8	113.55
16	204.46
32	298.29

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.

Comparing Selective Repeat with Go-Back-N at 25ms delay shows significant differences in throughput performance, especially at larger window sizes. At window size 1, both protocols behave similarly (15.46 KBps for Selective Repeat vs. 16.15 KBps for Go-Back-N) because with a window size of 1, both protocols effectively function as Stop-and-Wait. As window size increases, both protocols show improved throughput, but Selective Repeat demonstrates superior scaling at larger window sizes. At window size 16, Selective Repeat achieves 204.46 KBps vs. Go-Back-N's 187.04 KBps (~9.3% better). At window size 32, Selective Repeat achieves 298.29 KBps vs. Go-Back-N's 128.41 KBps (~132% better).

While Go-Back-N's throughput peaks at window size 16 and then declines, Selective Repeat continues to show improved throughput through window size 32 (the largest tested). These differences can be explained by the fundamental distinction between the two protocols: when packet loss occurs, Go-Back-N retransmits all packets from the lost packet to the end of the current window, while Selective Repeat only retransmits the specific lost packets. With a 5% packet loss rate in our experiments, Go-Back-N wastes significant bandwidth retransmitting correctly received packets, especially with larger window sizes. Selective Repeat's more efficient handling of packet loss allows it to maintain high throughput even with larger windows. The performance gap becomes more pronounced at window size 32 because the probability of at least one packet loss within the window increases with window size, making Go-Back-N's inefficiency more apparent while Selective Repeat continues to benefit from increased parallelism.

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	14.57
2	30.5
4	32.23
8	73.07
16	101.91
32	108.00

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 command *iperf3 -f K -c localhost -M 1024 -t 20 -w {N}* was used to collect the data presented in Question 7 for a given window size N. *iperf3* was used over *iperf* as it provided accurate TCP window sizes when set, whereas *iperf* would change the configured window size. Further to this, the OS minimum TCP window size was lowered by editing system configurations with *sysctl*, to allow the small window sizes used. All data collected used 25ms propogation delay, a 5% packet loss and a 10 Mb bandwidth.

Comparing the iperf (TCP) results with the Selective Repeat and Go-Back-N protocols (both UDP-based) at 25ms delay reveals several interesting differences. At small window sizes (1-2), all three protocols perform similarly, with throughput values in the range of 14-32 KBps. This indicates that when parallelism is limited, the core protocol differences have minimal impact. At window size 8, Selective Repeat (113.55 KBps) and Go-Back-N (124.69 KBps) both outperform TCP (73.07 KBps) At even larger window sizes (16-32), the performance difference becomes more pronounced, with window size 32: Selective Repeat achieves 298.29 KBps, Go-Back-N 128.41 KBps, and TCP only 108.00 Kbps.

These differences can be attributed to several factors. Firstly, TCP headers are larger than the minimal headers used in our UDP-based implementations, reducing the effective payload per packet. Next, TCP is designed to be network-friendly and avoid congestion, sometimes at the expense of maximum throughput. TCP is also a slower protocol than our UDP implementation, including packet validation with checksums, and will rate-limit itself. Finally,the TCP implementation handles various network conditions and optimizations beyond our simplified protocols, introducing additional processing overhead.

Selective Repeat clearly outperforms both TCP and Go-Back-N at larger window sizes due to its efficient handling of packet loss while maintaining a large transmission window. Go-Back-N initially performs well but suffers at larger window sizes due to its inefficient retransmission strategy. TCP, while robust and universally applicable, demonstrates more conservative throughput scaling with window size in this specific controlled environment with 5% packet loss.