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

## Assignment 2 Results Sheet

| | |
|---|---|
| **Forename and Surname:** | Patryk Marciniak |
| **Matriculation Number:** | s1828233 |

**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**.

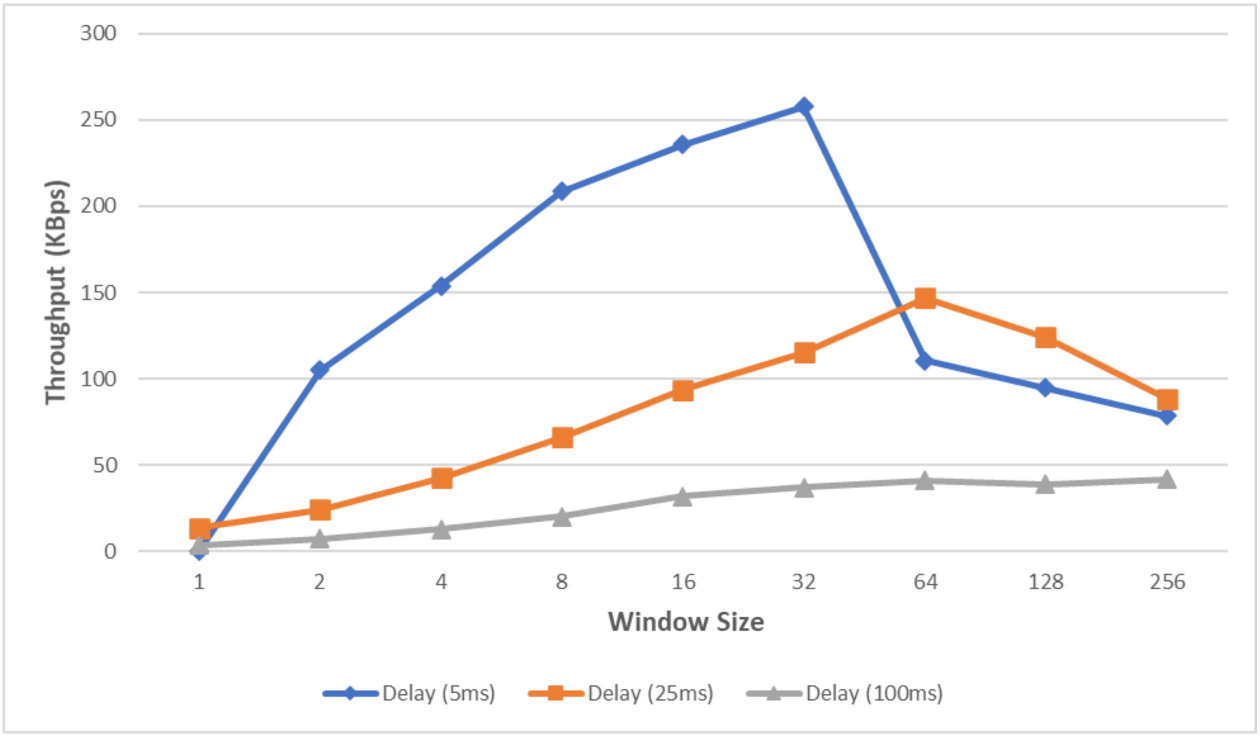
| Retransmission timeout (ms) | Average number of re-transmissions | Average throughput (Kilobytes per second) |
|---|---|---|
| 5 | 1538.8 | 68.62 |
| 10 | 788.4 | 63.36 |
| 15 | 215.6 | 62.22 |
| 20 | 121 | 61.48 |
| 25 | 101.8 | 60.1 |
| 30 | 108.6 | 54.25 |
| 40 | 106.4 | 51.37 |
| 50 | 104.6 | 48.75 |
| 75 | 100.6 | 41.1 |
| 100 | 105.6 | 36.22 |

**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).

For lower timeouts(5 and 10ms) the number of retransmissions is very high, this is because with a 5ms one-way propagation delay we have an expected RTT of 10ms meaning that the receiver should receive the ACK 10ms after the package is sent, but the short 5ms/10ms timeout is reached before this happens and the packet is resent hence the high number of retransmissions. With a 15ms timeout, the retransmissions are still high but much lower than with 5/10ms, likely because 15ms is only a bit higher than the RTT so some packets still get resent unnecessarily as they do with the 5/10ms timeout. After we exceed 20ms timeout the number of retransmissions stabilises because some packets will just be lost no matter how long we wait. It is also for this reason that the throughput decreases with a higher timeout. because some packets will not be received no matter what but the sender waits, wasting time and hurting throughput. The optimal timeout value is 20ms as it offers the best balance of a low number of retransmissions and a good throughput.

**Question 3** – 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 | 60.24 | 13.45 | 3.79 |
| 2 | 105.13 | 24.14 | 7.18 |
| 4 | 153.72 | 42.44 | 12.91 |
| 8 | 208.51 | 66.1 | 20.1 |
| 16 | 235.58 | 93.52 | 32.01 |
| 32 | 257.78 | 115.21 | 36.97 |
| 64 | 110.33 | 146.87 | 41.28 |
| 128 | 94.63 | 124.22 | 38.9 |
| 256 | 78.54 | 88.48 | 41.82 |



**Question 4** – Discuss your results from Question 3.

The timeout value at 25ms delay was set to 65ms and for 100ms delay it was set to 220ms, these values were chosen based on the optimal values found for 5ms delay where the optimal timeout was RTT + 10ms, so I used RTT + 15ms for 25ms delay and RTT + 20ms for 100ms delay since the delay is longer I added a bit more headroom for the higher timeouts.

For the 5ms delay, the throughput rises until a window size of 32 where it reaches a peak value of 258KBps, after that with window size 64 and more the throughput drops drastically, this is likely because the Go-Back-N protocol needs to retransmit all the packets in the window if the first packet in the window times out. With a large window size, there is a higher chance of packet loss and more packets will need to be retransmitted. Another compounding reason for this drop in throughput may be down to the receiver being overloaded by the arrival of so much data at once from the large window size and thus may not be able to send the ACKs to the sender on time causing it to time out.

With the long delays of 25ms and 100ms, the throughput is generally lower, likely because of the long delay and longer timeout, although from the above results especially for 25ms delay it can be seen that the optimal window size is larger than for a shorter delay. This suggests that we can generalize and say that to increase throughput for long delays we can increase window size.

**Question 5** – 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 | 18.11 |
| 2 | 34.56 |
| 4 | 64.59 |
| 8 | 115.36 |
| 16 | 185.25 |
| 32 | 314.53 |

**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.

As identified in the previous question a timeout of 65ms was used for the delay of 25ms.

Compared to Go Back N, Selective Repeat offers higher throughput for all the window sizes tested. The gains over Go Back N are smaller at lower window sizes because GBN doesn't need too much time to retransmit a smaller window. However, because Selective Repeat does not need to retransmit the whole window when packets time out, the throughput gains at larger window sizes become significantly higher over GBN which will need to retransmit the whole window. So overall we can say that Selective Repeat is more efficient than GBN as it wastes less bandwidth on retransmission however, comes at a cost of significantly higher complexity and is more resource-demanding making it less feasible to implement in practice.

**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 | 14.03 |
| 2 | 27.65 |
| 4 | 31.02 |
| 8 | 71.48 |
| 16 | 115.1 |
| 32 | 93.92 |

**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.

Using iperf the throughput is a bit lower compared to Go Back N and significantly lower relative to Selective Repeat. This is because iperf uses the TCP protocol which sacrifices speed to improve reliability as opposed to the UDP protocol implemented in GBN and SR.

Some reasons why TCP is slower include the fact that it must first establish a connection between server and client via handshake which takes up time and hurts performance, also the TCP header is much larger as it includes fields for the sequence number, acknowledgement, etc. which UDP does not have and thus its header is smaller. Since the TCP header is larger it means we can fit less data into each TCP packet compared to a UDP packet which may hurt throughput.

TCP also has flow and congestion control which control the rate at which the sender is sending the packets so that the receiver is not overwhelmed, while UDP just lets the sender send as much as it can which in effect results in higher throughput.