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

## Assignment 2 Results Sheet

| | |
|---|---|
| **Forename and Surname:** | Eoin Reid |
| **Matriculation Number:** | s1858933 |

**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 | 3267.8 | 64.7 |
| 10 | 1617.8 | 69.8 |
| 15 | 180.2 | 70.2 |
| 20 | 113 | 71 |
| 25 | 105 | 68 |
| 30 | 107.6 | 66.4 |
| 40 | 116.4 | 60 |
| 50 | 121.8 | 48.4 |
| 75 | 125.6 | 46 |
| 100 | 135.6 | 35.8 |

**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).

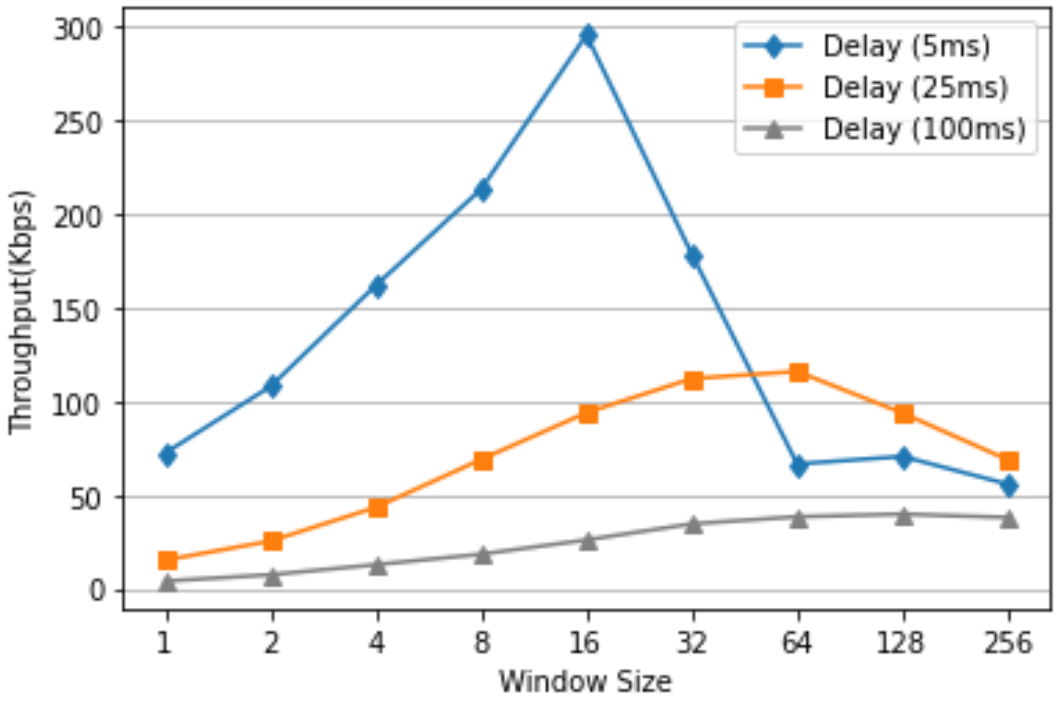
In order to send the file we have to send almost 800 packets, with 5% packet loss we expect to have to retransmit at least around 80 packets as 5% of both packets and acknowledgements will be lost. Initially we see the number of retransmissions is extremely high. This is expected as our retransmission timeout is much lower than the average round trip time (RTT) of ~20, therefore we won't wait long enough to receive the acknowledgement and will retransmit many packets unnecessarily.

As we increase the retransmission timeout we see a dramatic drop in the number of retransmissions. When increasing timeout from 5 to 10 and again from 10 to 15 we see a huge change in retransmissions

This is because, by increasing our timeout, we allow for more acknowledgements to be received before we attempt to retransmit, therefore reducing the number of unnecessary retransmissions.
When changing from 15 to 20 we see a smaller but still significant drop in the number of retransmissions, then thereafter we don't see much of a change at all. This again is unsurprising as with 80-100 packets/acks being lost, after a certain point it doesn't matter how long we wait as we are retransmitting roughly the actual number of packets that were lost/whose ack was lost. It makes sense for this plateau in retransmissions to occur at retransmission time of around 20/25 as this is close to the RTT. Therefore if we are waiting as long as or longer than the RTT then we will receive all acknowledgements for packets that aren't lost and won't wrongly retransmit any packets.
In terms of throughput we see a high throughput for low retransmission timeouts, after around 20/25ms timeout we see the throughput start to drop the longer the timeout. This is because in the event of a lost packet we are waiting longer than the RTT and therefore longer than we actually need to to see if we receive an ack. This timeout means that we pause for longer before retransmitting and causes our file to take much longer to transfer.
The optimal Retransmission timeout would be 20ms as this achieves a high throughput while achieving a relatively low number of retransmissions.

**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**.

| | | | |
|---|---|---|---|
| 1 | 72.4 | 15.2 | 4 |
| 2 | 108 | 25.4 | 7.4 |
| 4 | 162 | 43.4 | 12.8 |
| 8 | 213.4 | 69 | 18.4 |
| 16 | 295.8 | 94 | 26 |
| 32 | 178.6 | 112.2 | 34.6 |
| 64 | 66.6 | 116 | 38.4 |
| 128 | 70.6 | 93.8 | 39.8 |
| 256 | 55.8 | 68.8 | 38 |




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

For each delay I wanted to have a timeout that was larger than the average round trip time to send a packet and receive its acknowledgement. The average RTT for 5ms, 25 ms, and 100 ms is 10ms, 50ms, and 200ms respectively. Therefore I chose a timeout larger than this for each. I used timeouts of 15ms for 5ms propagation delay, 60ms for 25ms delay and 210ms for delay of 100ms.

Initially for all three delays, increasing the window size increases the throughput. However, we can see that in each of the cases it reaches a peak after which increasing the window size actually has a negative impact on the throughput. This is to be expected as Go-Back-N retransmits the whole window size in the event of lost packets. Therefore in the case of large windows, if one packet or its acknowledgement is lost we end up retransmitting 64, 128 or 256 packets slowing down the program and overwhelming the receiver. This is as a result of the time it takes to retransmit these packets and the receiver receiving packets quicker than it is able to acknowledge them causing a bottleneck in the transmission.

In the cases of higher propagation delay we see that this peak comes at a higher window size. This is because the receiver is less likely to become overwhelmed by incoming packets as they are arriving at a slower rate.

For 5 ms delay we see that the peak occurs at window size of 16. For 25ms the peak occurs at window size of 64, and lastly for 100ms delay we can see the peak occurs at window size = 128.

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

| | |
|---|---|
| 1 | 15.6 |
| 2 | 25.4 |
| 4 | 45.6 |
| 8 | 83.6 |
| 16 | 121.2 |
| 32 | 190.6 |

**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.

For Question 5 I used a retry timeout of 60ms for the same reasons as explained in question 4.
Comparing my Selective Repeat and Go-Back-N results for the same propagation delay we can see that initially both perform extremely similarly, however, after a window size of 4 we start to see Selective Repeats throughput perform better. Initially the performance is only slightly better (for window size of 8 SR has throughput 83.6 to GBNs 69) however as the window size increases the difference becomes more stark. For window size of 32 SR has almost triple GBNs throughput (SR has throughput 190.6 to GBNS 68.8).
These results make sense; the issue with Go-Back-N is that in the case of lost packets we retransmit a whole window of packets, often unnecessarily retransmitting the full window when only one or two packets were actually lost. For small window sizes this difference is negligible, however as we increase the window size and we are retransmitting large windows anytime we lose a packet the difference becomes significant. Selective Repeat aims to solve this problem by only retransmitting the packets that we haven't received an ack for in the selected timeout. Therefore for higher window sizes we will still only be retransmitting lost or slow packets rather than full windows allowing for significantly better performance.

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

| | |
|---|---|
| 1 | 11.26 |
| 2 | 20 |
| 4 | 26.73 |
| 8 | 52.09 |
| 16 | 72.34 |
| 32 | 80.90 |

**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.

Comparing these results to Selective Repeat and Go-Back-N we can note that it exhibits a similar pattern to SR, that is that throughput steadily increases as we increase the window size. Performance for small windows is similar to both however after window size of 4 we see that it's throughput is generally lower. This could be explained by the fact that iperf utilises TCP whereas the GBN and SR implementations use UDP which is comparatively faster. This is because of many features within TCP itself such as; larger header allowing for a smaller payload therefore increasing time taken to send a file, checksum calculations, congestion control among others.