**CSC 573 – Internet Protocols**

**Project #1**

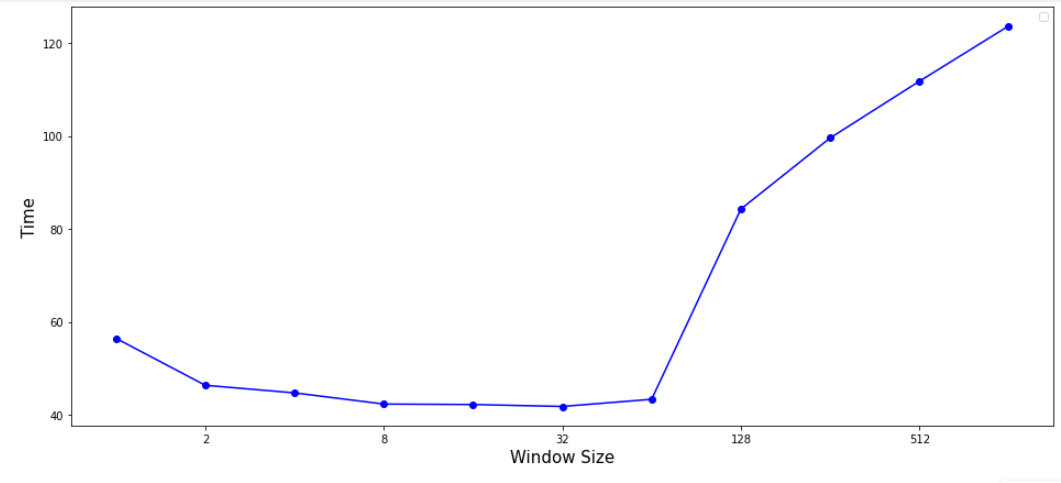
**Archit Kwatra – 200316390**

Size of the file is 1.1MB for all the tasks

TASK – 1

|  |  |
| --- | --- |
| **Window Size** | **Time** |
| 1 | 56.432 |
| 2 | 46.3342 |
| 4 | 44.7 |
| 8 | 42.294 |
| 16 | 42.192 |
| 32 | 41.78 |
| 64 | 43.32 |
| 128 | 84.245 |
| 256 | 99.545 |
| 512 | 111.753 |
| 1024 | 123.65 |

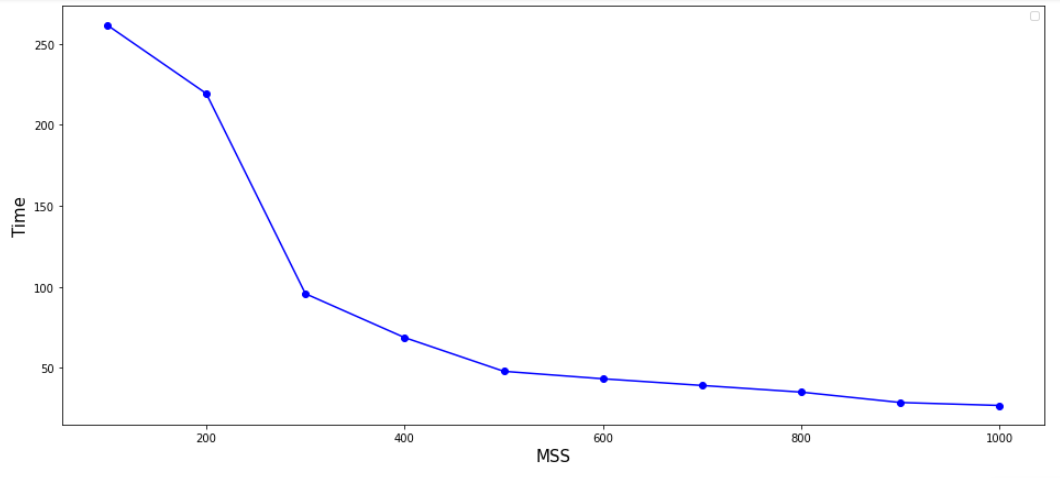
Below is the graph for the average delay vs window size. We can see that the graph is a “U” shaped, which is indicative that neither too large values not too small values for the window size are desirable. The reason could be that too small values might lead to reciever not acknowledging the sent packets and thus causing delay. On the other hand, when the window size is too large, the sender has to send all the packets which were not acknowledged, if an acknowledgement is lost and thus increasing the dalay once again. Therefore, on optiomal value of window size should be chosen. According to the graph, an optmal value could be between 8 to 64.



TASK – 2

As evident from the graph, larger the MSS value, lesser will be the delay since we are sending more data with each packet. However, too large values could be troublesome.

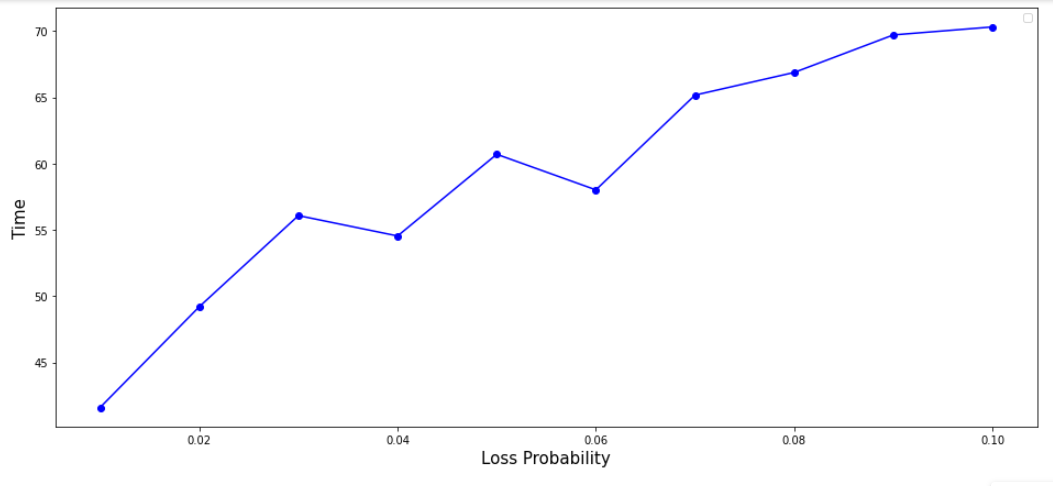
|  |  |
| --- | --- |
| **MSS** | **Time** |
| 100 | 261.654 |
| 200 | 219.3459 |
| 300 | 95.762 |
| 400 | 68.6732 |
| 500 | 47.8456 |
| 600 | 43.22 |
| 700 | 39.098 |
| 800 | 34.983 |
| 900 | 28.567 |
| 1000 | 26.755 |



TASK – 3

|  |  |
| --- | --- |
| **Loss Probability** | **Time** |
| 0.01 | 41.6 |
| 0.02 | 49.2 |
| 0.03 | 56.076 |
| 0.04 | 54.54 |
| 0.05 | 60.702 |
| 0.06 | 58.018 |
| 0.07 | 65.168 |
| 0.08 | 66.8765 |
| 0.09 | 69.707 |
| 0.10 | 70.311 |

The graph is an increasing curve, though not a linear increase, which shows that delay will be reduced since there is lesser chance of discarding a packet at lower probability as compared to discarding a packet at a higher probability and hence we don’t need to resend the packet for a discarded one. Therefore, a lower value of the probability is more desirable.

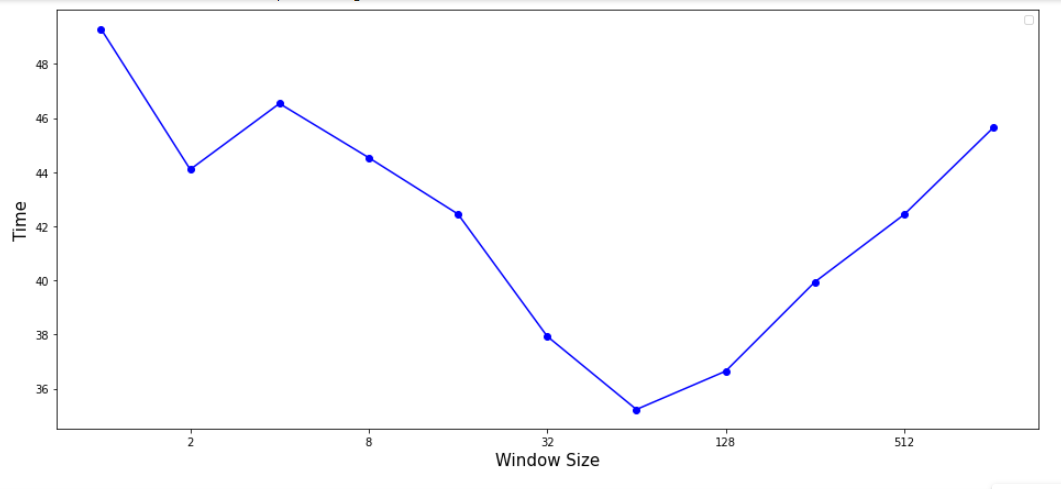


Selective ARQ –

TASK – 1

|  |  |
| --- | --- |
| **Window Size** | **Time** |
| 1 | 49.29 |
| 2 | 44.098 |
| 4 | 46.534 |
| 8 | 44.534 |
| 16 | 42.45 |
| 32 | 37.934 |
| 64 | 35.23 |
| 128 | 36.645 |
| 256 | 39.941 |
| 512 | 42.432 |
| 1024 | 45.6454 |

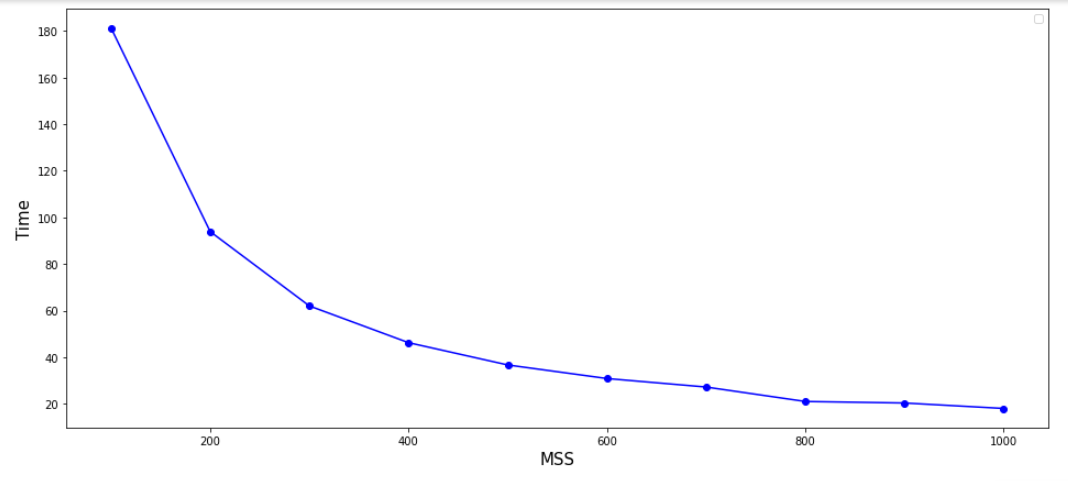
The selective Repeat ARQ shows an improvement over the Go-Back-N algorithm since in this case, only the negative ACK packets are retransmitted. The best value for this algorithm is for a window size = 64.



TASK – 2

|  |  |
| --- | --- |
| **MSS** | **Time** |
| 100 | 181.32 |
| 200 | 93.90 |
| 300 | 62 |
| 400 | 46.234 |
| 500 | 36.67 |
| 600 | 30.849 |
| 700 | 27.19 |
| 800 | 20.998 |
| 900 | 20.341 |
| 1000 | 18 |

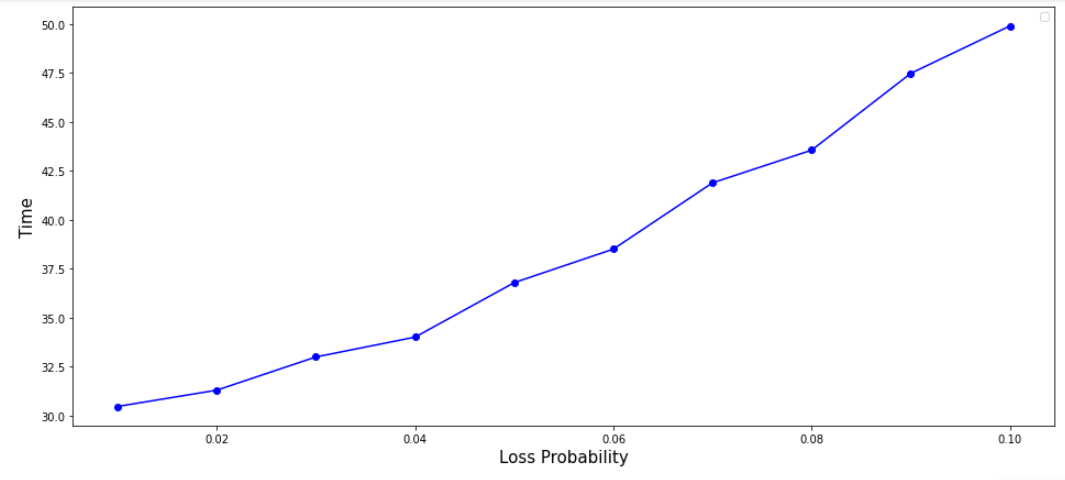
Similar to Go\_back\_N algorithm, the delay decreases with an increase in the MSS value since more data is sent with each packet.



TASK – 3

|  |  |
| --- | --- |
| **Loss Probability** | **Time** |
| 0.01 | 30.456 |
| 0.02 | 31.287 |
| 0.03 | 32.987 |
| 0.04 | 34 |
| 0.05 | 36.786 |
| 0.06 | 38.5 |
| 0.07 | 41.889 |
| 0.08 | 43.56 |
| 0.09 | 47.487 |
| 0.10 | 49.91 |

This graph is also similar to Go\_Back\_N algorithm. The delay increases with an increase in the loss probability.



Some of my other observations:

Sometimes, the total time for the file transfer varies immensely. The reason could be attributed to some congestion in practical scenarios; however, I was using the same system for both client and server.

The file transfer speed increased when transferring the file for multiple times (without any change in the values).