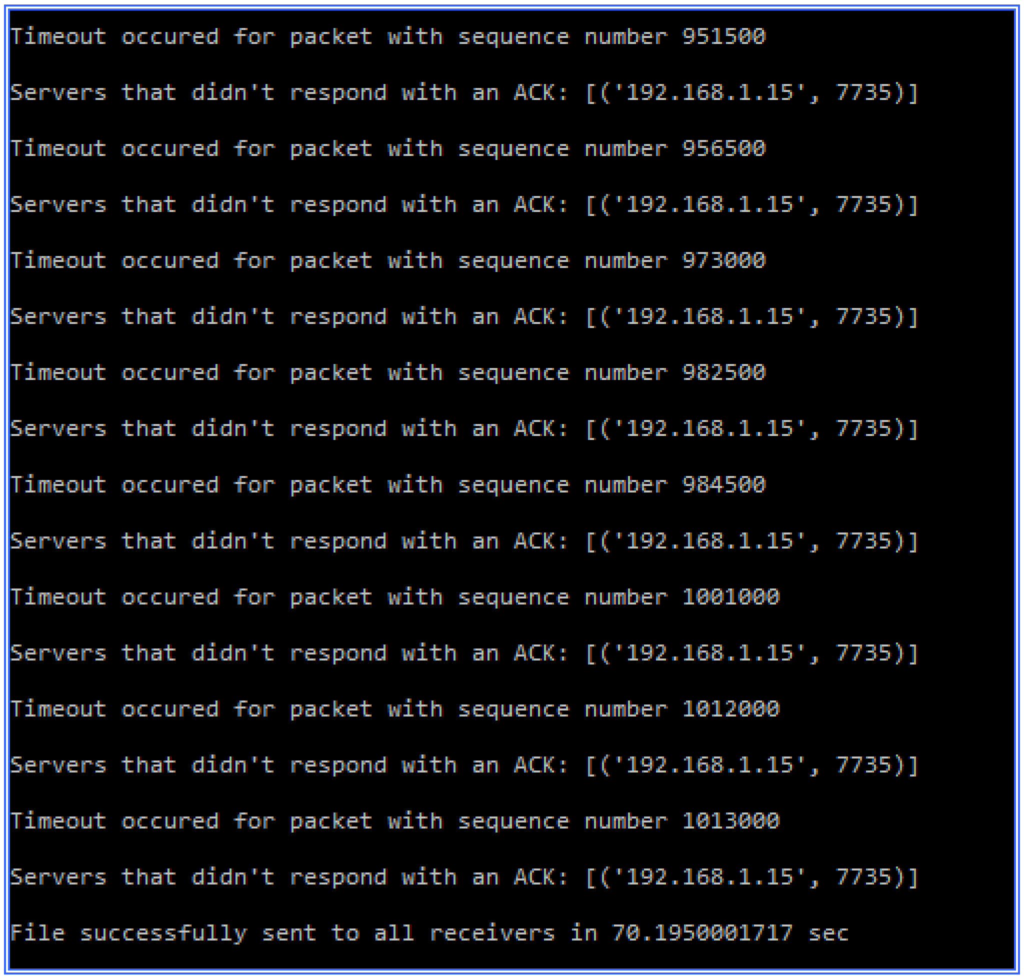
IP Project #2

Vellaiyur Chellaram Charan Ram (cvellai) - 200158572
Kashyap Sivasubramanian (ksivasu) - 200200471

Outputs Obtained

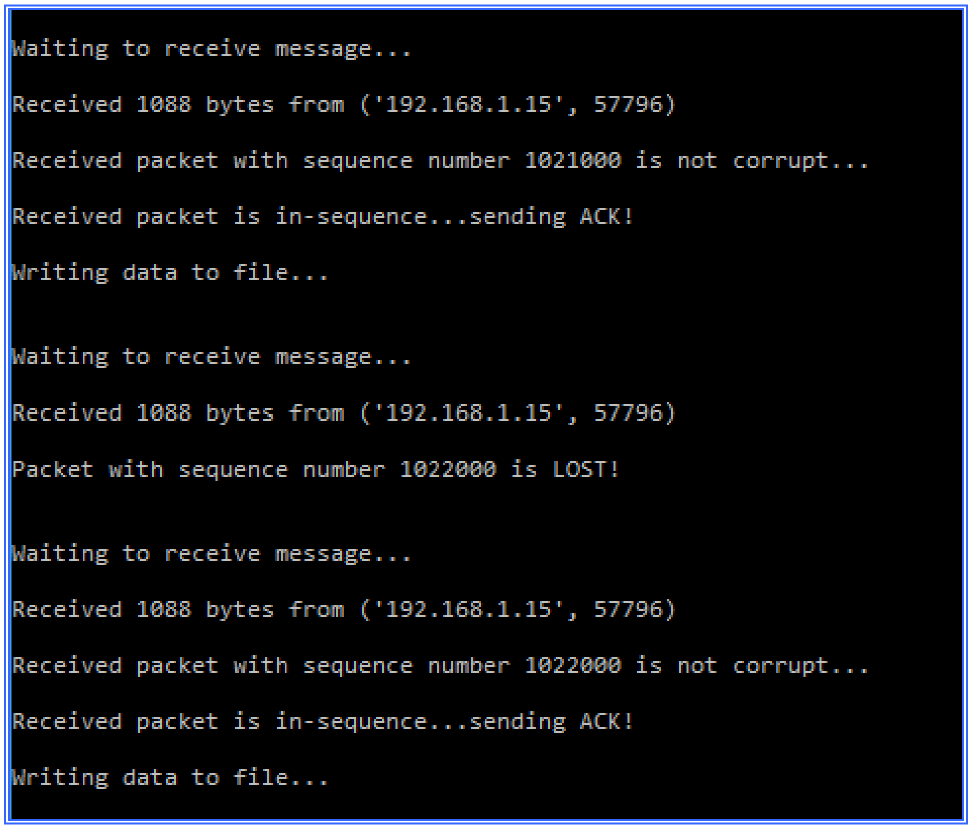
These are the sample output snippets on the client and server side:



```
Timeout occured for packet with sequence number 951500

Servers that didn't respond with an ACK: [('192.168.1.15', 7735)]

Timeout occured for packet with sequence number 956500

Servers that didn't respond with an ACK: [('192.168.1.15', 7735)]

Timeout occured for packet with sequence number 973000

Servers that didn't respond with an ACK: [('192.168.1.15', 7735)]

Timeout occured for packet with sequence number 982500

Servers that didn't respond with an ACK: [('192.168.1.15', 7735)]

Timeout occured for packet with sequence number 984500

Servers that didn't respond with an ACK: [('192.168.1.15', 7735)]

Timeout occured for packet with sequence number 1001000

Servers that didn't respond with an ACK: [('192.168.1.15', 7735)]

Timeout occured for packet with sequence number 1012000

Servers that didn't respond with an ACK: [('192.168.1.15', 7735)]

Timeout occured for packet with sequence number 1013000

Servers that didn't respond with an ACK: [('192.168.1.15', 7735)]

File successfully sent to all receivers in 70.1950001717 sec
```

Client Side

The sequence number here has been implemented based on the number of bytes sent. For example, if the first packet sent has 1000 bytes with sequence number 0, the receiver sends an ACK with sequence number 0, indicating it has received the first 1000 bytes and that it’s waiting for the next packet with sequence number 1000.



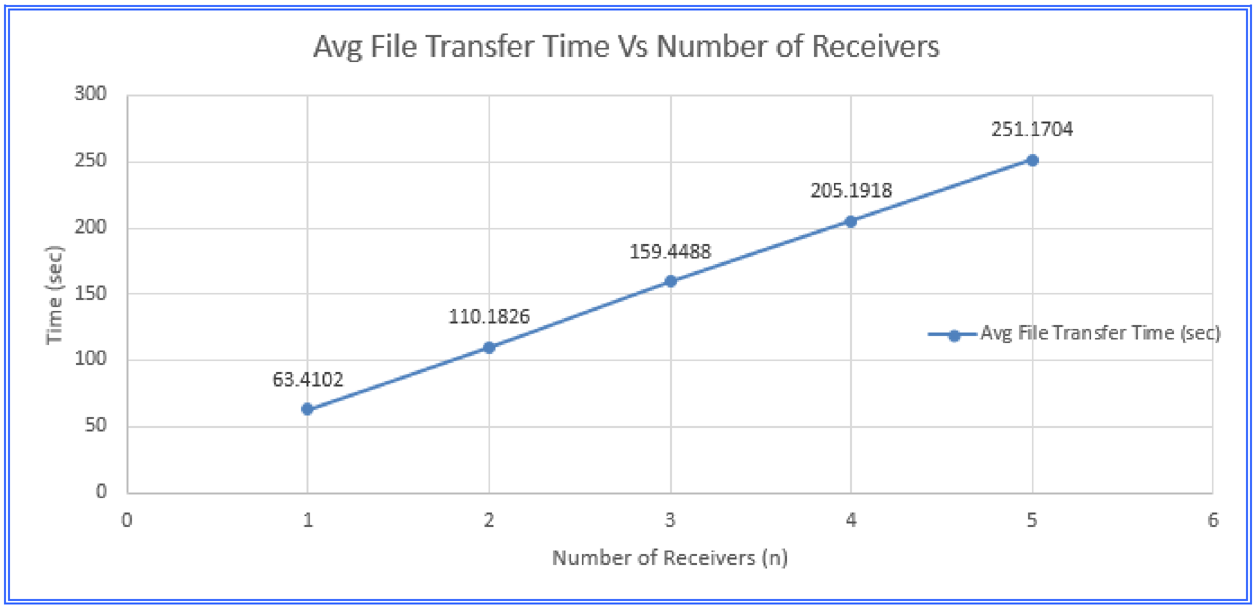
```
Waiting to receive message...

Received 1088 bytes from ('192.168.1.15', 57796)

Received packet with sequence number 1021000 is not corrupt...

Received packet is in-sequence...sending ACK!

Writing data to file...


Waiting to receive message...

Received 1088 bytes from ('192.168.1.15', 57796)

Packet with sequence number 1022000 is LOST!


Waiting to receive message...

Received 1088 bytes from ('192.168.1.15', 57796)

Received packet with sequence number 1022000 is not corrupt...

Received packet is in-sequence...sending ACK!

Writing data to file...
```

Server Side

The snippet above showing the server side tells us the number of bytes received from the client, and whether the packet received is in order or not. If the packet received is lost, it displays the sequence number of the packet that is lost.

Results Obtained

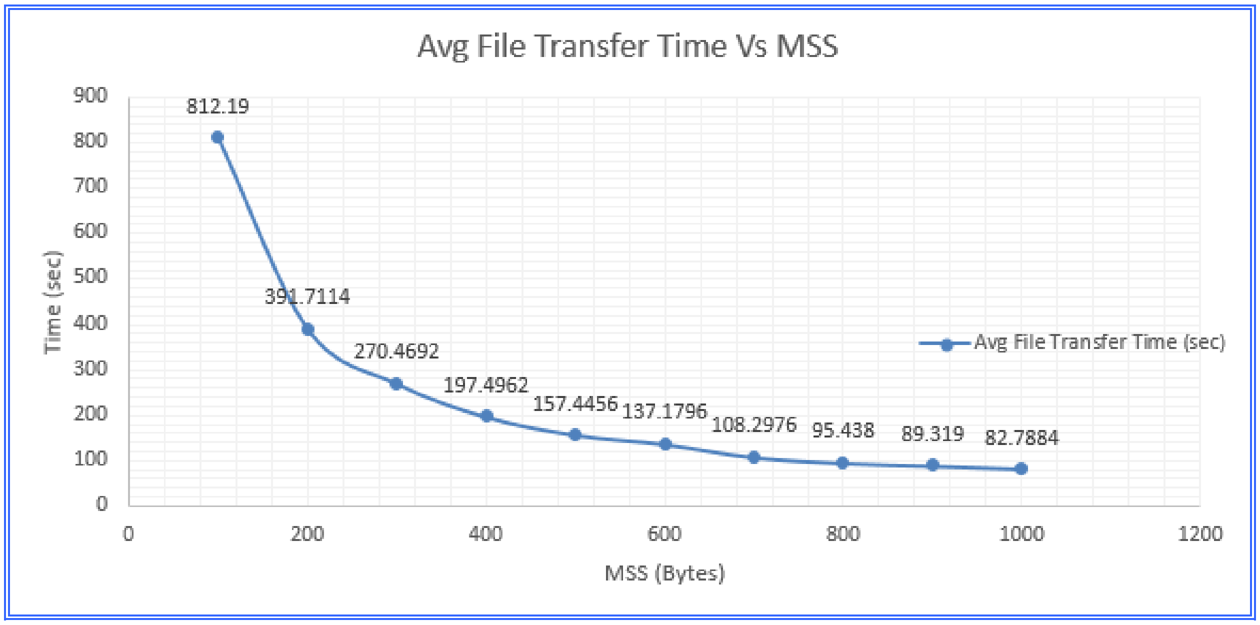
Task 1



The Average File Transfer Time is plotted against the Number of Receivers as shown in the figure above. The Task was performed by keeping the **Probability of loss** in the server side and **MSS** as constants. The values were kept at **0.05** and **500** respectively.

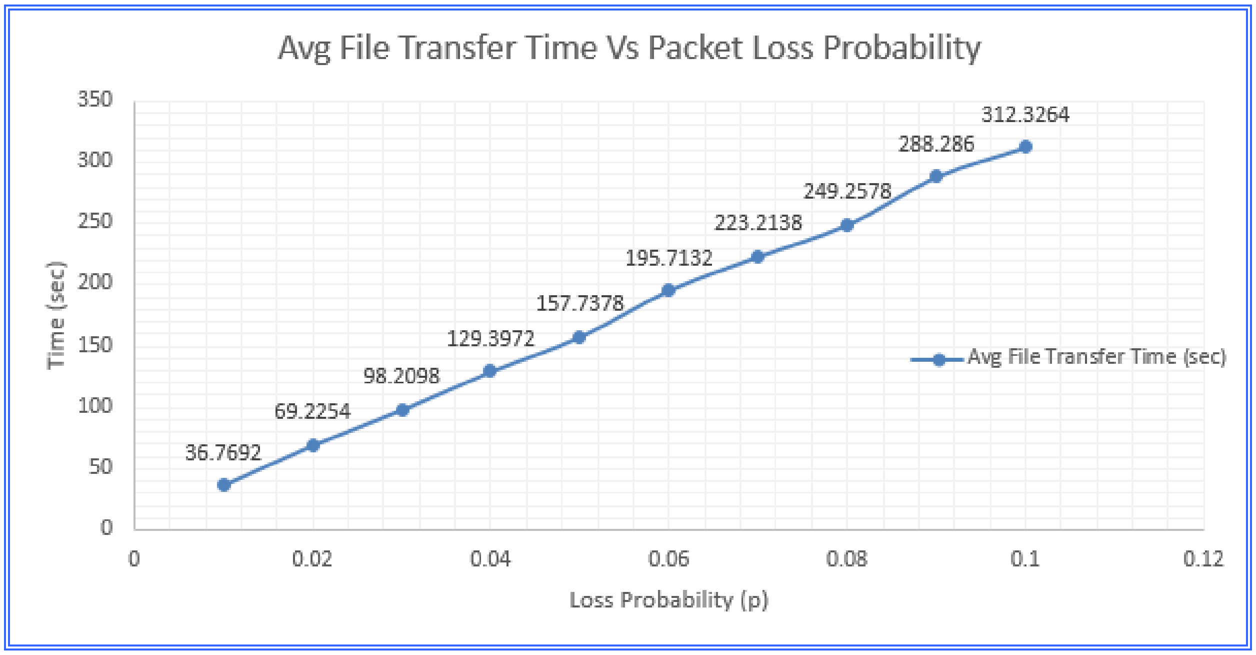
We can see that the Average Time to transfer files to n receivers increase as the number of receivers increases in a Stop and Wait ARQ model. In the Stop and Wait model, the client moves on to the next segment only when it has received an ACK for the previously sent segment from all its receivers. This consecutively prolongs the file transfer time as the receiver count keeps on increasing.

Task 2



In Task 2, Average File Transfer Time is plotted versus MSS keeping the **Number of Servers** and **Probability of Loss** at the server as constants. They are kept at **3** and **0.05** respectively.
From the above graph, we can see that the Average File Transfer Time deceases as MSS increases. This is the expected behavior as a larger number of bytes are sent at one go as MSS increases.

Task 3



Task 3 was performed keeping the value of **MSS** and the **Number of Servers** constant. The values were **500** and **3** respectively. As the value of loss probability increases, the average time taken to transfer the file to all the servers also increases. This can be understood by the fact that as the loss probability becomes larger, the number of retransmissions required to transmit a file increases which results in an overall increase in the file transfer time.

Conclusions

From the above three tasks we can draw the following conclusions:

1. We can infer that Stop and Wait ARQ model is not suited for implementations wherein a large number of hosts need to be serviced. This can be seen form Task 1, where the delays for file transfer increases almost linearly with the number of servers in the network.
2. As observed from Task 2, the value of the MSS has a significant influence on the file transfer time. In actuality, the MSS value should neither be too small to cause numerous unnecessary transmissions nor be too large so as to increase retransmissions by being more prone to errors.
3. From Task 3, we can see that as the Loss Probability increases, the average time to transfer a file also increases. So, the Stop and Wait ARQ model becomes inefficient for systems which are lossy, due to the larger number of retransmissions involved.