CN-3530/CS 301 Assignment 2

1. Stop and Wait Protocol

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 **average throughput**.

Retransmission timeout (ms)	Average number of re-transmissions	Average throughput (Kilobytes per second)
5	9300.2	53.36044
10	2602.6	63.46298
15	2341	59.35123
20	188.2	62.49117
25	132.6	58.27574
30	145.8	53.93188
40	126	50.01772
50	133.6	44.81977
75	129	39.53269
100	106.8	37.84428

Question 2 – Discuss the impact of retransmission timeout value on number of retransmissions and throughput. Indicate the optimal timeout value from communication efficiency viewpoint (i.e., the timeout that minimizes the number of retransmissions and keeps the throughput as high as possible).

The optimal timeout value that could be chosen here would be a timeout of ***25ms.*** As upon sorting by smallest number of retransmissions, the highest throughput achieved is at 25ms as the next highest is achieved at 20ms but the number of retransmissions would increase from 132 to 188 with an increase in throughput by a small amount.
The impact of retransmission timeout on the number of retransmissions is that the number of retransmissions decrease with the increase in timeout time as the sender will wait for longer time to receive the acknowledgement from the receiver.
The impact of retransmission time on the throughput is that the throughput decreases with the increase in timeout time as the sender can now wait a longer time to receive ACK from receiver.

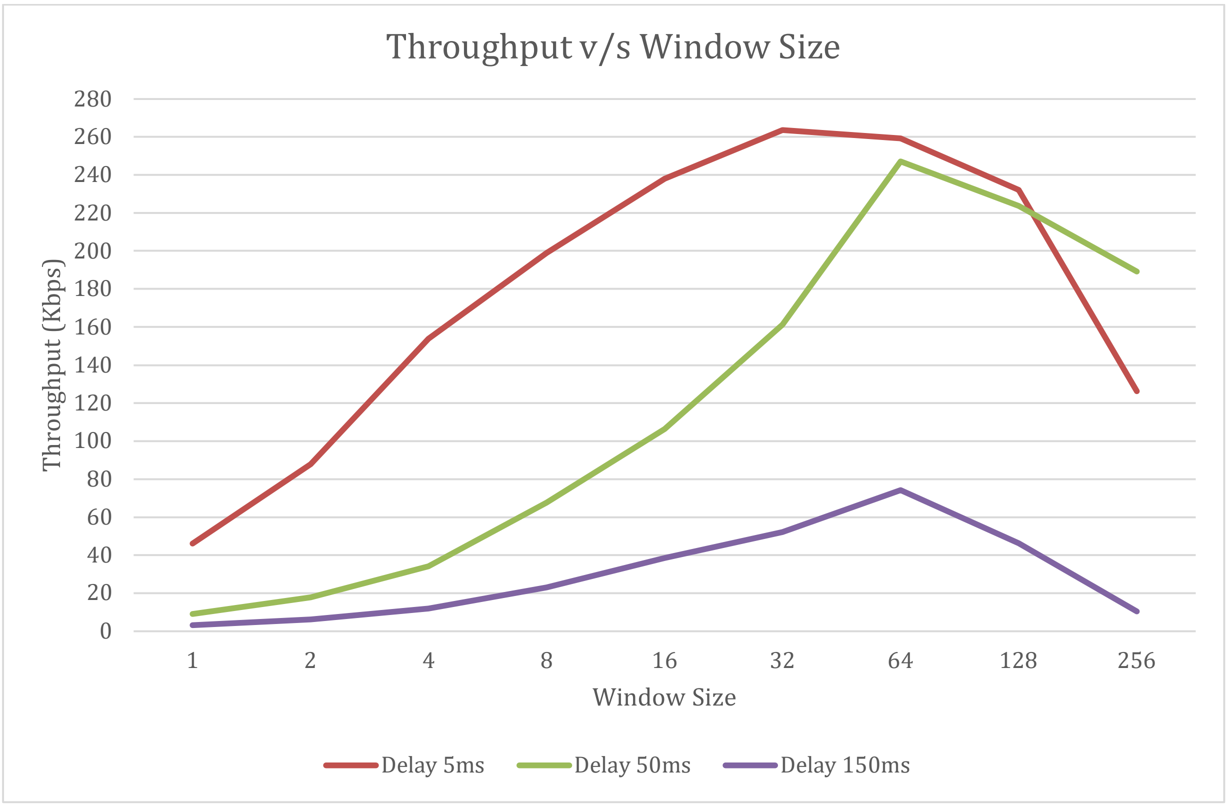
In the program, the entire image is first being read and put into a dictionary with key value being chunk number and data being the binary byte form of the image chunk. We iterate through the dictionary and create a packet to be sent which consists of a sequence number (which is the chunk number), byte flag indicating the last of the file and the image data. The entire packet is of size 1024 bytes. Once the packet is sent, the sender receives for the ACK from the server. If it receives the correct ACK then it accepts and moves ahead, if it receives an incorrect ACK then it resends the packet and if a timeout occurs, then it resends the packet.
On the receiver end, it always waits for a packet to be sent by the sender, it also keeps track of the chunk number (aka the sequence number), if it receives the correct packet then it sends an ACK to the sender, if it receives a duplicate packet, then it resends the ACK as it might be possible that the ACK got lost during its travel to the sender. When the receiver, get the packet the first time, then it stores in a dictionary with key value being the chunk number and the data being the image data. In the end, once it receives the last packet, it closes the connection and saves the file.

2. Go back N Protocol

Question 1 – Experimentation with Go-Back-N. For each value of window size, run the experiments **5 times** and write down the **average throughput**.

	Average throughput (Kilobytes per second)		
Window Size	**Delay = 5ms**	**Delay = 50ms**	**Delay = 150ms**
1	46.06915854	9.003367101	3.160847087
2	87.82733646	17.86333965	6.200039121
4	153.8621552	34.09672753	11.96892597
8	198.9518953	67.63202962	23.01317916
16	238.0707546	106.3312243	38.43324793
32	263.6265616	161.2673457	52.10483808
64	259.301267	247.1031537	74.14711415
128	232.2244588	223.6355069	46.16287438
256	126.1211789	189.1997845	10.45336138

Create a graph similar to the one shown below using the results from the above table: (Edit: change delays to 5ms, 50ms and 150 ms as mentioned in the assignment statement)



Question 2 – Discuss your results from Question 1.

We can observe that the throughput increases with the increase in window size upto a certain optimum value after which it decreases. The reason behind this is that when the window size is small, the sender waits for each ACK for each packet and will send the next packet only when the ACK is received. As the window size is increasing, the number of packets sent is more, so the sender can receive the ACKs faster as compared to before hence the increasing throughput, but at a certain value it starts decreasing, because as the window size increases, and if a packet loss occurs at the beginning of the window then it will have to resend all those other packets again, which decreases the throughput as now the sender will wait a longer time to receive the ACKs.

In the case of 5ms delay, the timeout time used is 25ms, the throughput here is a lot more than the other delay values as the sender waits a lesser time for the packet to return and the optimal value for window size occurs at 32. 64 can also be used as its value is very close to 32.

In the case of 50ms delay, timeout time used is 250ms (value has been adjusted to account for the propagation delay), the optimum value occurs at 64 window size.

In the case of 150ms delay, the timeout time used is 500ms (value has been adjusted to account for the propagation delay), the optimum value occurs at 64 window size.

The program reads the image in the same manner as before as done in stop and wait. The receiver side program is same as before. In the sender side, there are 2 functions, one for sending the data and one for receiving the ACKs. In the main function, the *send_func* thread is started and inside that the receive_func thread is started. A mutex lock is used to lock acquire the thread resources. All shared variables are put as global inside the program, so that both the threads can access them and modify them.

In send_func, the thread acquires a lock when it is sending the packet from the window and releases when its done. In receive_func, the thread acquires a lock when it identifies that the socket has received an ACK, if the sequence number of ACK is same as the base then it proceeds ahead by shifting the base by 1, if it receives an out of order or incorrect ACK then it releases the lock and goes into exception case. In the exception case, when timeout has occurred, it will acquire a lock and change the next_sequence_number back to base and releases the lock.

Once the receiver gets all the chunks, it exits the program and saves the file.

Name: Tanmay Garg
Date: 18th Nov 2022
Signature: TG