INTERNET PROTOCOLS

PROJECT 2

Point to Multipoint Reliable Data Transfer Protocol

BHARATH VENKATESH

Student ID: 001037220

SRIDEVI JANTLI

Student ID: 200019504

Objective:

- Encapsulating application data into transport layer segments by including transport headers.
- Implementing mechanisms for buffering and managing data to be delivered to multiple destinations.
- Verifying the correctness of data by computing checksums.
- Using the UDP socket interface.

Design:

- UDP Server: The Servers are the receivers of the data sent by the Client which does the following functionalities:
    - Receive Data: The data is received using the UDP socket.
    - Checksum Calculation: Compute the checksum of the entire segment received and verify if the checksum has all 1s.
    - Probabilistic Discard: Discard the segments received if the random number generated is less than the probability specified. This is used to introduce the packet loss functionality.
    - Write Data: Write the received data to the specified file.
    - Send ACK: Send an ACK back to the client.

- UDP Client: The Client is a sender which sends data to multiple servers on a well defined port and does the following functionalities:
    - Read File: Read the data from the file specified by the user.
    - Framing Headers: Frame the header with sequence number, data indicator and checksum, FIN.
    - Checksum Calculation: Compute the checksum for the data segment and header.
    - Send Data: Send the data to multiple servers using UDP socket.
    - Wait for ACK: Wait for the ACK from the receivers for the segment sent.

Implementation:

Programming Language: Java 1.7 [Using open-jdk]

Server:

- The Server is of type Server class and holds the filename to write and the socket to receive the data from the client.
- Checksum for the header is calculated using HeadCheckSum function.
- Checksum for the data is calculated using ByteDataCheckSum function.
- Final Checksum for the entire segment is calculated using ComputeFinalChecksum.

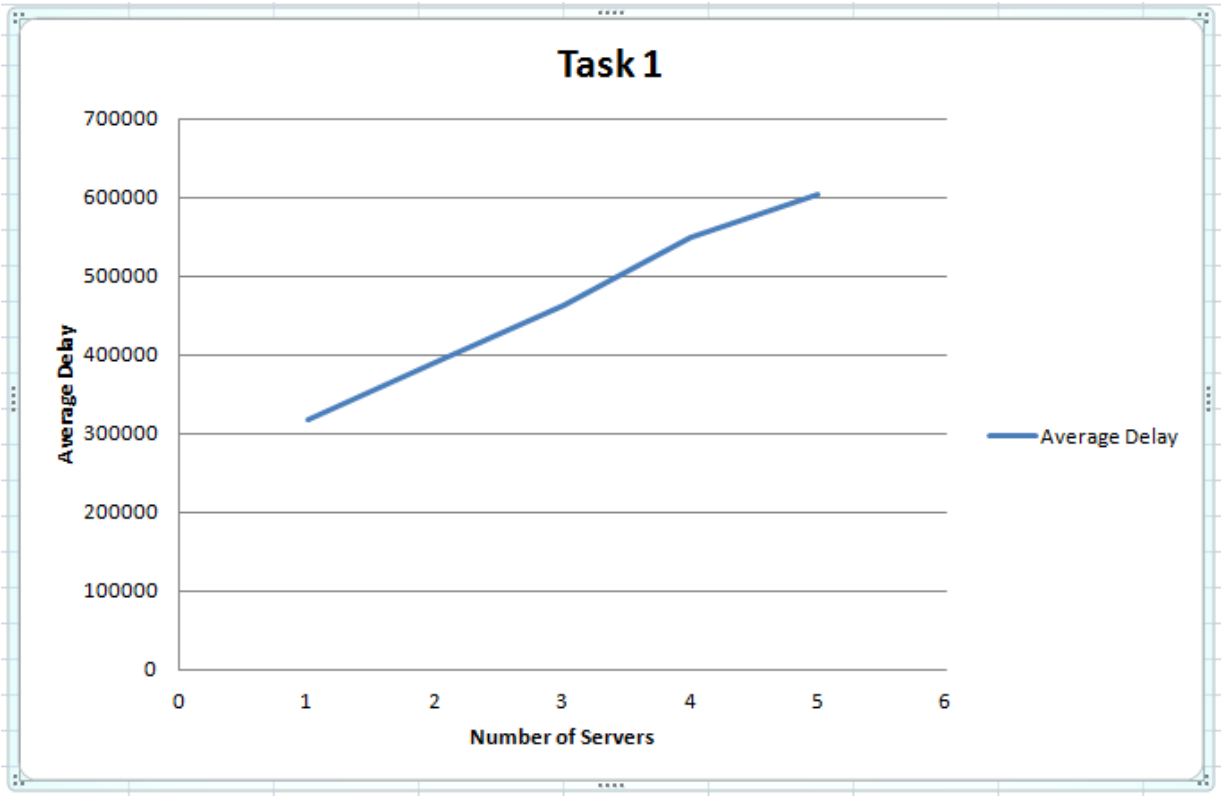
- Ignore the segment if the computed checksum is not all one's or if the random number generated is less than the loss probability.
- Write the data to the file and send an ACK back to the Client.

Client:

- The Client is of type Client class and holds the sequence number, data indicator, FIN and the file which is sent to the server.
- Checksum for the header is calculated using HeadCheckSum function.
- Checksum for the data is calculated using ByteDataCheckSum function.
- Final Checksum for the entire segment is calculated using ComputeFinalChecksum.
- Send segments to all the Servers and wait for an ACK from all of them.
- If ACK not received from all the Servers within in the timeout time then resend the segment to the all the Servers from which ACK was not received.

Task 1 Effect of the Receiver Set Size n:

Number of Servers	Average Delay
5	604555
4	548747.6
3	463761.8
2	389629.8
1	318469.8



Observation:

We see a linear increase in the average delay time as the number of servers increases.

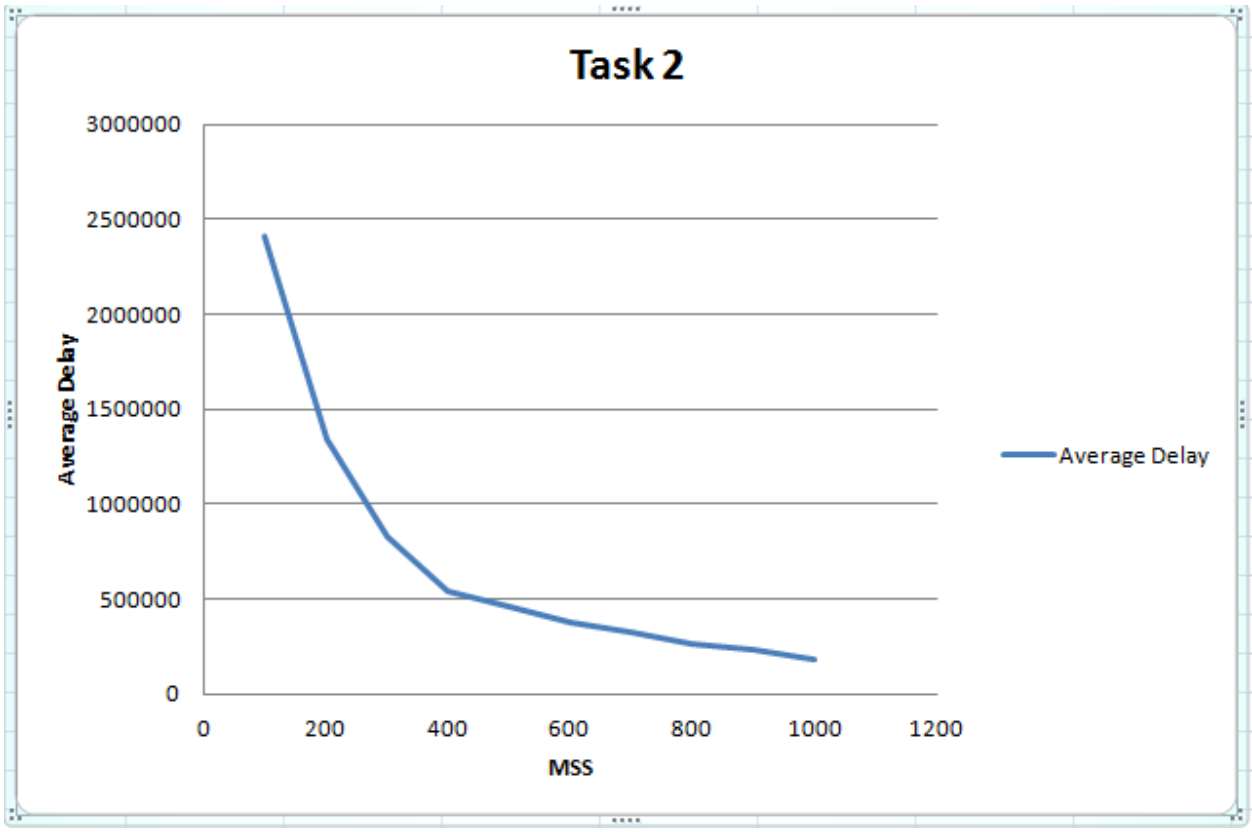
Explanation:

We see that linear increase in the delay due to the following factors.

- Transmission time increases with the increase in the number of servers. The Client has to now has to send packets to n servers.
- As the number of servers increase the probability of one of the server generating a random number less than the specified probability increases n folds.
- Another reason being that, the client has to wait for ACKs from all the servers before transmitting the next packet, meaning if any of the server has a packet loss then the Client defers sending of the next packet to all the servers.

Task 2: Effect of MSS

MSS	Average Delay
100	2414219.2
200	1339543
300	831574.6
400	542589
500	456827.6
600	374054.8
700	324875.2
800	270241.6
900	232301.2
1000	181079



Observation:

As the segment size increases from 100 to 500 we see an exponential decrease in the average delay. Further from 500 to 1000 there is a linear decrease.

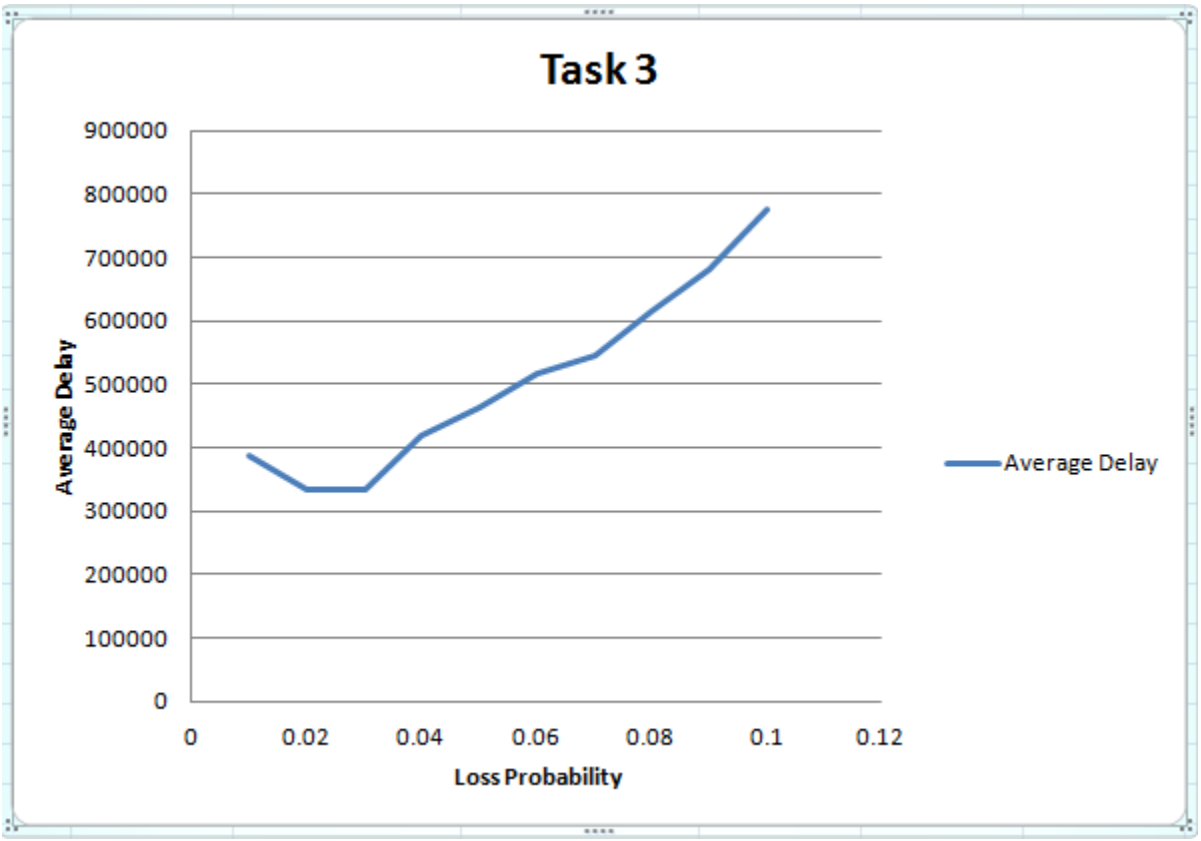
Explanation:

We see the decrease in the delay due to the following factors:

- The decrease in the delay is due to the fact that number of transmissions gets reduced with the increase in MSS.
- Initially we see the exponential decrease because the segment size is much lesser than the capacity of the socket thus the socket is underutilized.
- Another reason for this being that, the overhead of the header is considerably high for a small MSS.
- After some increase in the MSS we see that the exponential decrease turns to linear decrease as, the header for the segment being transferred is now negligible compared to the size of the actual data being transferred
- Also as the MSS increases the utilization of the bandwidth increases accordingly till it reaches a threshold.

Task 3: Effect of Loss Probability p

Loss Probability	Average Delay
0.01	388077
0.02	334312.6
0.03	332698.6
0.04	419392
0.05	462897.8
0.06	516964.4
0.07	544079
0.08	614966.8
0.09	680654.4
0.1	774859.2



Observation:

We see an increase in the delay as the loss probability increases from 0.01 to 0.10. The slight variance seen between 0.01 to 0.03 can be accredited to the randomness of the function that generates the loss probability at the server side.

Explanation:

- The increase in delay is due to the fact that as the loss probability increases, the number of retransmissions increase requiring more duplicate transmissions from the Client to the server.
- As the loss probability further increases from 0.1 to 0.5 and further, we expect an exponential increase in the delay, as this would result in numerous retransmissions for a single packet to be delivered from the sender to the receiver.

Conclusion:

- The Peer to Peer Multipoint protocol that we designed can be seen to be ideally performing when the number of servers are minimal with least loss probability and a very high MSS.
- Since such an ideal situation is not practical to be obtained we see that since the delay increases as the number of servers increase to negate that effect the loss probability should be decreases appropriately and the corresponding MSS size should be increased.
- Task1 indicates that it is desirable to *limit the number of servers* supported in such a design to contain the delay incurred to an acceptable limit.
- We see from the tasks that we performed that as the loss probability increases above a certain limit the delay increases exponentially. So it desirable to have *very minimal loss probability.*
- We see that as the size of the MSS is increased from 100 to 1000 the delay decreases. This indicates that at MSS 100 the network was underutilized. Thus to increase the scalabaility of the servers it is required that we maintain a *high MSS thus increasing the utilization.*
- It is also to be noted that other traffic existing on the network might impact the productivity of the network.

Scenario,Run	n	Delay	Average
1,1	5	604372	
1,2	5	598432	
1,3	5	619345	604555
1,4	5	593274	
1,5	5	607352	
1,1	4	533616	
1,2	4	554438	
1,3	4	573345	548747.6
1,4	4	524518	
1,5	4	557821	
1,1	3	465115	
1,2	3	457662	
1,3	3	483561	463761.8
1,4	3	473435	
1,5	3	439036	
1,1	2	396470	
1,2	2	379047	
1,3	2	367434	389629.8
1,4	2	406524	
1,5	2	398674	
1,1	1	324417	
1,2	1	325603	
1,3	1	319045	318469.8
1,4	1	308392	
1,5	1	314892	

Scenario,Run	MSS	Delay	Average
2,1	100	2364771	
2,2	100	2436805	
2,3	100	2459624	2414219.2
2,4	100	2399780	
2,5	100	2410116	
2,1	200	1321289	
2,2	200	1365994	
2,3	200	1287531	1339543
2,4	200	1358469	
2,5	200	1364432	
2,1	300	800209	
2,2	300	886425	
2,3	300	796332	831574.6
2,4	300	847956	
2,5	300	826951	
2,1	400	539515	
2,2	400	567244	
2,3	400	521393	542589
2,4	400	517952	
2,5	400	566841	
2,1	500	465823	
2,2	500	448521	
2,3	500	498362	456827.6
2,4	500	412765	
2,5	500	458667	

Scenario,Run	MSS	Delay	Average
2,1	600	385781	
2,2	600	356972	
2,3	600	346976	374054.8
2,4	600	405622	
2,5	600	374923	
2,1	700	321863	
2,2	700	366964	
2,3	700	310896	324875.2
2,4	700	305697	
2,5	700	318956	
2,1	800	281583	
2,2	800	256981	
2,3	800	265712	270241.6
2,4	800	274695	
2,5	800	269537	
2,1	900	240991	
2,2	900	224935	
2,3	900	213649	232301.2
2,4	900	245516	
2,5	900	236415	
2,1	1000	193090	
2,2	1000	186473	
2,3	1000	169524	181079
2,4	1000	196476	
2,5	1000	159832	

Scenario,Run	Loss Probability	Delay	Average
3,1	0.01	390355	
3,2	0.01	389422	
3,3	0.01	376491	388077
3,4	0.01	395601	
3,5	0.01	388516	
3,1	0.02	308289	
3,2	0.02	359264	
3,3	0.02	308793	334312.6
3,4	0.02	346921	
3,5	0.02	348296	
3,1	0.03	355261	
3,2	0.03	321008	
3,3	0.03	329648	332698.6
3,4	0.03	310749	
3,5	0.03	346827	
3,1	0.04	406050	
3,2	0.04	416982	
3,3	0.04	412494	419392
3,4	0.04	435716	
3,5	0.04	425718	
3,1	0.05	445672	
3,2	0.05	456913	
3,3	0.05	496473	462897.8
3,4	0.05	438511	
3,5	0.05	476920	

Scenario,Run	Loss Probability	Delay	Average
3,1	0.06	503351	
3,2	0.06	516924	
3,3	0.06	523640	516964.4
3,4	0.06	517942	
3,5	0.06	522965	
3,1	0.07	524835	
3,2	0.07	569133	
3,3	0.07	542865	544079
3,4	0.07	526719	
3,5	0.07	556843	
3,1	0.08	602384	
3,2	0.08	639475	
3,3	0.08	619460	614966.8
3,4	0.08	612746	
3,5	0.08	600769	
3,1	0.09	664708	
3,2	0.09	674932	
3,3	0.09	696513	680654.4
3,4	0.09	688016	
3,5	0.09	679103	
3,1	0.1	795913	
3,2	0.1	765891	
3,3	0.1	783647	774859.2
3,4	0.1	766014	
3,5	0.1	762831	