IP Project Report – 2

Team Members – Atul Sharma, Shubham Balsaraf, Rashi Agarwal

Project Objective:

In this project, we implemented point-to-multipoint reliable data transfer protocol using the Stop-and-Wait automatic repeat request (ARQ) scheme. We carried out a number of experiments to evaluate its performance.

The P2MP-FTP Client (Sender):

The client(sender) reads data from a file specified in the command line, and it calls rdt_send() to transfer the data to the P2MP-FTP servers.

The client transmits each of the segment separately to each of the receivers, and waits until it has received ACKs from every receiver before it can transmit the next segment. Every time a segment is transmitted, the sender sets a timeout counter. If the counter expires before ACKs from all receivers have been received, then the sender re-transmits the segment, but only to those receivers from which it has not received an ACK yet. This process repeats until all ACKs have been received.

The header of the segment contains three fields:

-> A 32-bit sequence number,

-> A 16-bit checksum of the data part, computed in the same way as the UDP checksum, and

-> A 16-bit field that has the value 0101010101010101, indicating that this is a data packet.

The P2MP-FTP Server (Receiver):

The server listens on the well-known port 7735. It implements the receive side of the Stop-and-Wait protocol. When it receives the data packet, it computes the checksum and checks the received data into a file whose name is provided in the command line. If the packet received is out-of-sequence, and ACK for the last received in-sequence packet in send, if the checksum is incorrect, the receiver does nothing.

The ACK segment consists of three fields and no data:

-> The 32-bit sequence number that is being ACKed,

-> A 16-bit field that is all zeroes, and

-> A 16-bit field that has the value 1010101010101010, indicating that this is an ACK packet.

Retransmission:

The retransmission occurs in the following cases:

-> The receiver(server) does not Acknowledge data due to error in receiving data and timeout occurs on the sender(client) side.

-> When receiver(server) does not Acknowledge data due to wrong checksum and timeout occurs on the sender(client) side. (the random (r) value is less than probability)

-> When the receiver(server) send an acknowledgement, but it gets lost somewhere in the transit and timeout occurs on the sender(client) side.

Offline Experiments:

The file size taken for the experiments is 2MB.

We tested using 5 servers in different virtual machines binded to local host.

The RTT value was determined by performing traceroute to one of the servers and the average of 10 ms was obtained. The timeout value was set to 5*RTT = 0.05 seconds

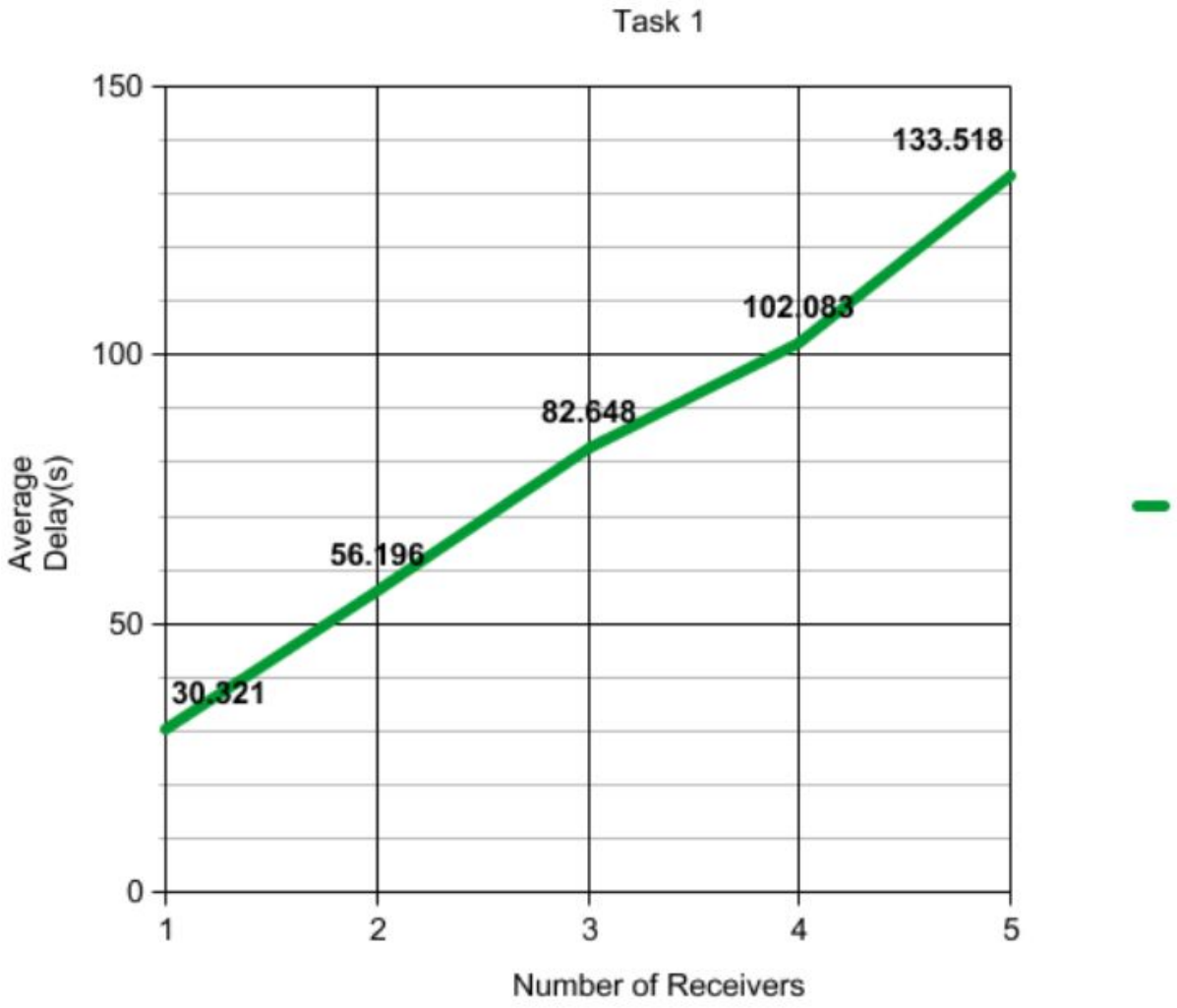
The following results were obtained:

Task1: Effect of the Receiver Set Size n

For the first task, the MSS was set to 500 bytes and the loss probability p = 0.05. After that the receivers(n) are varied – 1,2,3,4,5. For each value of n, transmit the file 5 times, time the data transfer(delay), and compute the average delay over the file transmissions.

Graph indicates the average delay against n

TASK 1 Timeout = 0.05 seconds					
Receivers	1	2	3	4	5
	30.203	54.741	82.654	101.654	133.679
	28.502	52.912	80.125	103.364	130.648
	30.111	59.782	84.565	106.032	135.489
	31.246	56.961	76.356	100.029	128.364
	31.642	55.8	82.982	101.654	136.032
Average Delay(s)	30.321	56.196	82.648	102.083	133.518



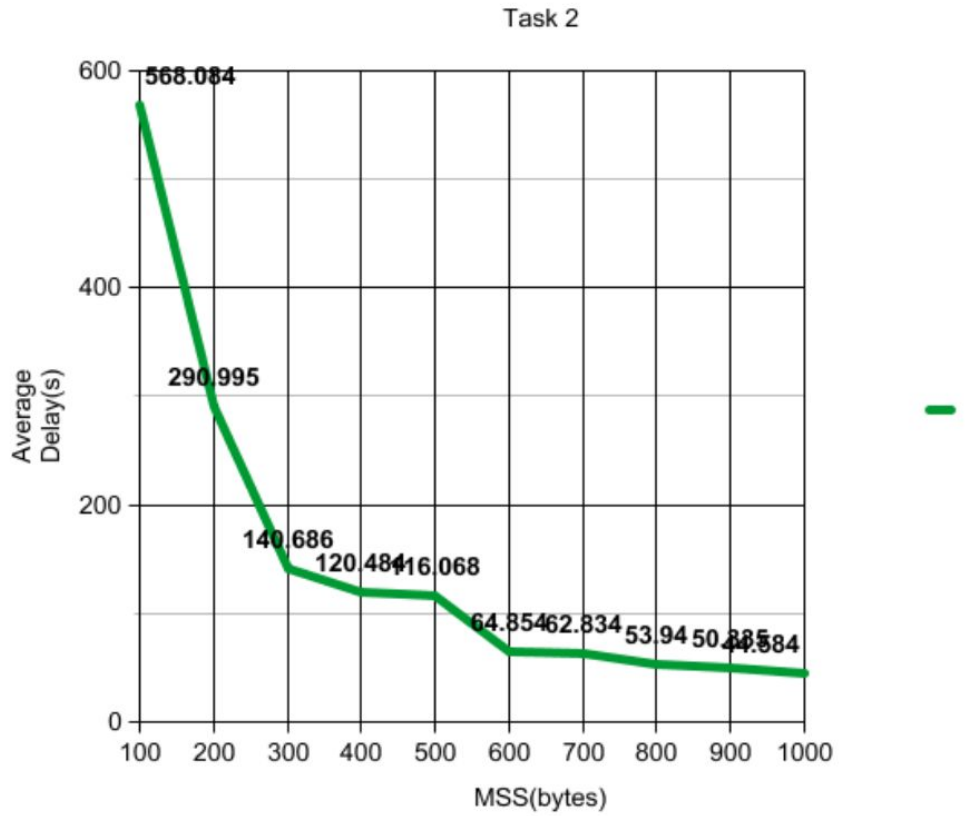
With the increase in the number of receivers and constant MSS and loss probability, the file takes more time to transfer i.e. the average delay increases. The delay increases because the sender(client) has to wait for acknowledgements from all the receivers in order to transmit next segment.

Task 2: Effect of MSS

The number of receivers(n) are set to 3 and the loss probability is set to p= 0.05. The MSS is varied from 100 bytes to 1000bytes in increments of 100 bytes. For each MSS value, the file is transmitted 5 times, average delay is computed over the five transmissions.

Graph indicates plot of average delay vs the MSS value:

TASK 2										
MSS	100	200	300	400	500	600	700	800	900	1000
	568.978	290.454	140	120	116.469	64.782	60.432	53.939	50.236	44.566
	565.85	290.336	140	125	114.546	64.011	64.465	52.649	51.362	46.102
	569.154	290.363	136	123	113.125	61.125	62.123	51.822	49.923	45.203
	571.322	292.987	142	117	117.89	68.165	63.003	53.987	50.822	43.625
	566.982	288.364	142	116	118.923	63.652	61.23	54.004	49.682	44.265
Average Delay(s)	568.084	290.995	140.686	120.484	116.068	64.854	62.834	53.94	50.335	44.584



From the graph we can infer that as the size of MSS increases, the delay decreases exponentially. When the MSS increases, the amount of data that can be received in a segment will be more, which means that the number of packets required to transmit the same amount of data is lower. So, per-packet processing time is less. The minimum delay increases, since packets are now bigger, per flow delays between packets are now also higher.

Task 3: Effect of loss probability

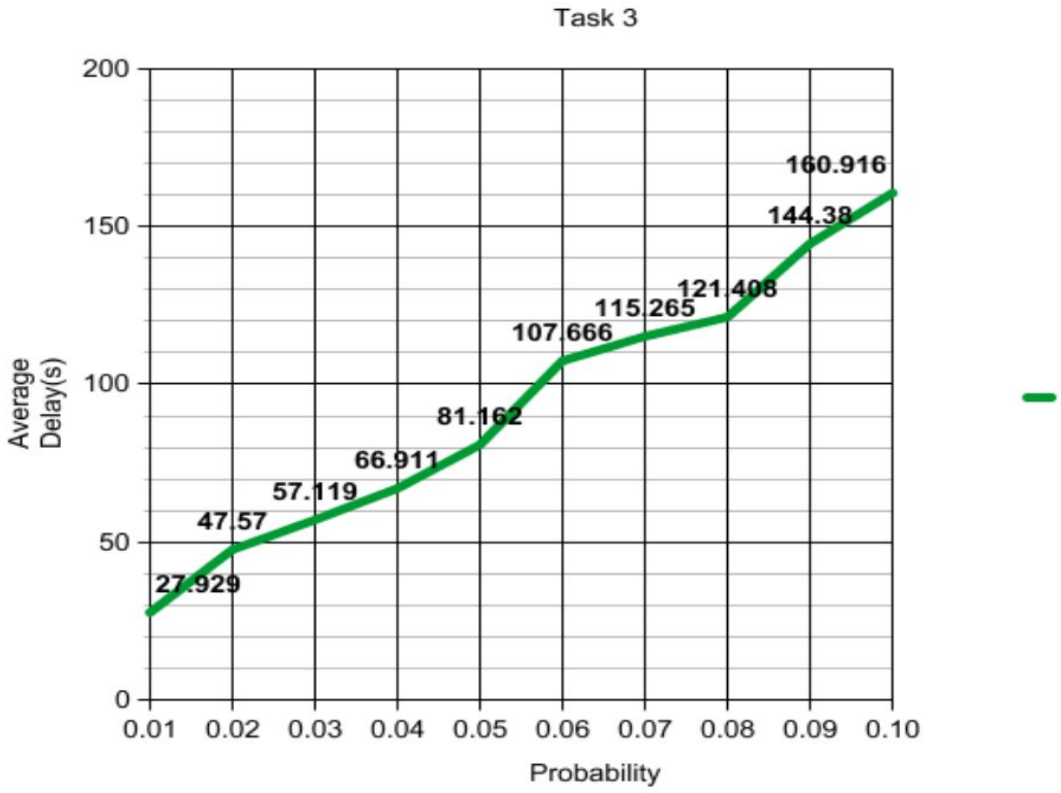
The MSS value is set to 500 bytes and the number of receivers are set to 3. The value of probability is varied from 0.01 to 0.10 in increments of 0.01. For each value of p, the file is transmitted 5 times, and average delay is computed over the five transfers.

Graph indicates average delay vs probability:

TASK 3										
Probability	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
	26.917837	47.610364	57.001	66.922	82.012	107.65	115.201	121.812	144.65	161.822
	27.368273	46.936484	56.652	65.989	80.702	108.788	116.548	122.003	142.989	159.908
	27.028936	47.024768	56.366	67.166	80.898	105.998	114.639	120.509	145.854	160.202
	27.196378	46.937481	57.989	66.877	81.032	106.807	115.209	121.382	144.365	160.89
Average Delay(s)	27.929	47.57	57.119	66.911	81.162	107.666	115.265	121.408	144.38	160.916

We can observe an increase in the average delay as probability of loss increases. The random value is calculated between 0 to 1 and is compared with probability which we vary from 0.01 to 0.1. When r value is less than p, the packet is retransmitted by the client.

As we increase probability value from 0.01 to 0.1, the chance of r being greater than probability decreases but not linearly as the r variable is random.



Conclusion:

The following are the conclusions based on the offline tasks conducted:

1) Link is underutilized because we have used Stop and Wait protocol which means at a time only one segment is sent to the receiver. Instead, Go Back N or selective repeat protocol would be more efficient and practical.
2) From Task 1 we observed that average delay increases with the number of receivers. Hence the protocol does not include stability and is inefficient when large numbers of receivers come into the picture.