KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)

CSC/ECE 573 – Internet Protocols
PROJECT 2 REPORT:

Overview:

This project specifies the utilization of point to multipoint reliable data transfer i.e P2MP rdt using protocols likes Stop-and-Wait (SAW), Automatic Repeat Request (ARQ) and UDP.

The Data to be transferred through this rdt methodology is first encapsulated in segments with appropriate header fields (8 byte) and transported as if the data transfer were reliable using checksums and window sizing management.

The scheme of the project uses Python and several hosts that are configured as either clients or servers. There is only one single client which serves as a sender for all servers (i.e. receivers). Thus, a single data segment is broadcasted at a time to all receivers. The sender must wait to receive an acknowledgement message from each separate receiver before continuing with the next data segment. All data transfer occurs from the sender to the receivers. The only messages that travel from the receivers to the sender are acknowledgements.

Client side specifications:

1.) Reads data from the file that needs to be transmitted to the servers

2.) Calls the rdt_send() –

Gathers all P2MP-FTP protocol data fields together to make up a datagram of 1 MSS and send it to the list of P2MP-FTP Servers. The mss value is given on the command line.

3.) Implements sending side of ARQ :

Reads enough raw from the file to build 1 MSS packet worth of data. Calls helper functions to calculate checksum (sum of all 16 bit words and take complement of it), construct the header and transmit the packet.

4.) Calls the rdt_send_datagram() function:

Sends datagram to all the P2MP-FTP Servers via multi-cast. Create new UDP socket with timeout interval to transfer datagram to P2MP-FTP Servers. The multi-cast technique is used to send the same (one) datagram to all P2MP-FTP Servers. After datagram is sent to all P2MP-FTP Servers, it waits for ACK responses from P2MP-FTP Servers. Calls helper function to determine what P2MP-FTP Servers received data packets correctly.

5.) Extract Server ACK - Extracts the ACK received from P2MP-FTP Server as response. It extracts the packet received from the socket and verifies whether this is really ACK received from one of the P2MP-FTP Servers as response after sending a data packet. This packet can be corrupted, or not P2MP-FTP protocol packet or not even from P2MP-FTP Server. Each case is handled gracefully with an exception. If this is expected ACK, it updates the host object accordingly. Verifies whether ACK is received from all P2MP-FTP Servers. It must not ACK the segment correctly, only need to check if the response is received.

Header field present in each MSS contains following information:

1. 32 bit seq number
2. 16 bit UDP checksum
3. 16 bit value indicating data packet as
   - DATA_PACKET = 0101010101010101
   - LAST_DATA_PACKET = 0101010101010111

Server side specifications:

1.) Listens on port number #7735 that is specified on the command line while running each server. This may change if we are using the servers in the NCSU EOS machines.

2.) Computes the UDP checksum to create a reliable data transfer.

3.) Sends ack's to the client using UDP connection.

4.) Writes data into the file whose name is specified on the command line.

5.) If out-of-order packet sequence is received then the ACK for the last received in-sequenced packet is sent as displayed in below screenshots. If checksum computations is displaying wrong packet reception then do nothing.

ACK field information:

1. 32 bit sequence number that is being Acked.
2. 16 bit field with all 0's
3. 16 bit value indicating ACK packet as 1010101010101010

Probabilistic Loss Service:

1.) The value of probabilistic constant p that is specified at the server side on the command line is compared with a random value that is generated by the server code.

2.) The value of p needs to be specified always in a range of (0,1)

3.) If the random value generated r<=p, then reject the packet & no other action needs to be taken. So more the value of p more will be the Packet loss occurring at the Server side.

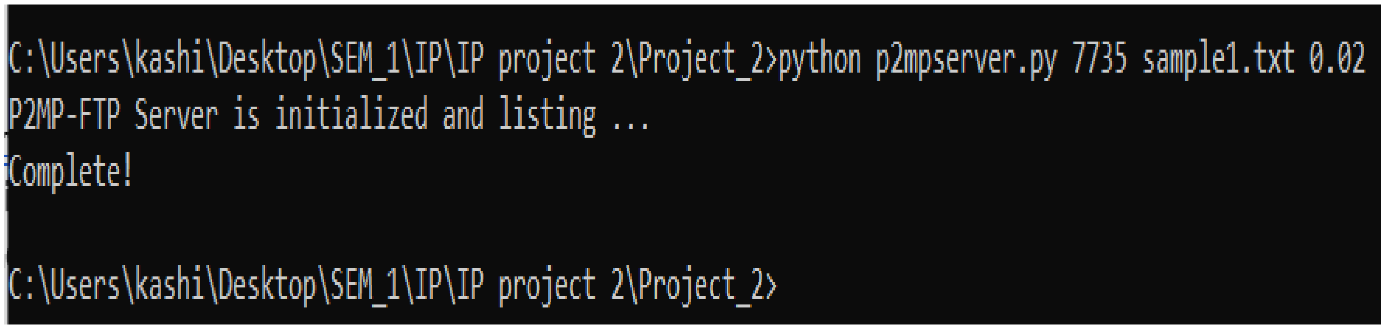
KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)

Command Line Arguments:

Format of command line: Server (receiver):

The P2MP-FTP server must be invoked as follows:
p2mpserver port# file-name p
where port# is the port number to which the server is listening (for this project, this port number is always constant for all the servers), file-name is the name of the file where the data will be written, and p is the packet loss probability discussed above.

$> python p2mpserver.py 7735 sample1.txt 0.02



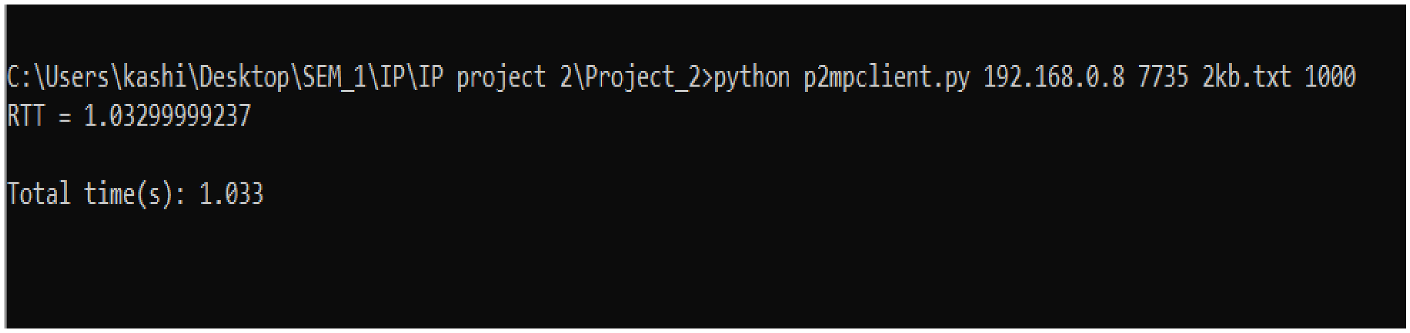
```
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\Project_2>python p2mpserver.py 7735 sample1.txt 0.02
P2MP-FTP Server is initialized and listing ...
Complete!

C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\Project_2>
```

Fig 1.

Format of command line: Client (sender):

The P2MP-FTP client must be invoked as follows:
p2mpclient server-1 server-2 server-3 server-port# file-name MSS
where server-i is the host name where the i-th server (receiver) runs, i = 1, 2, 3
server-port# is the port number of the server (i.e., 7735),
file-name is the name of the file to be transferred,
and MSS is the maximum segment size.

$> python p2mpclient.py 192.168.0.8 7735 2kb.txt 1000



```
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\Project_2>python p2mpclient.py 192.168.0.8 7735 2kb.txt 1000
RTT = 1.03299999237

Total time(s): 1.033
```

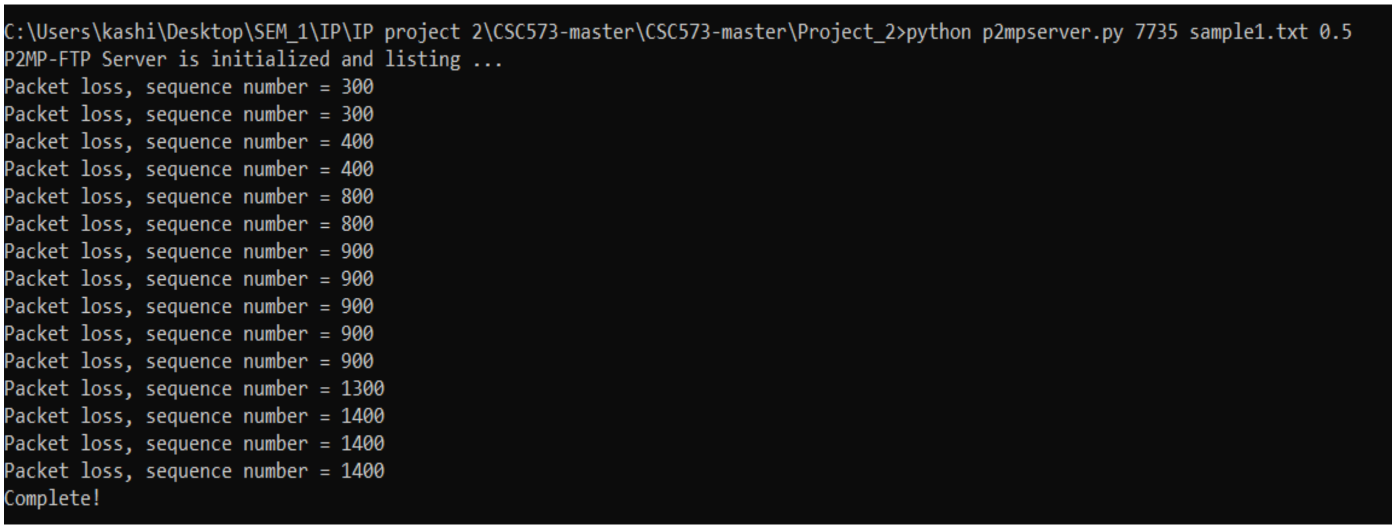
Fig2.

KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)

OUTPUT:

Output 1 : If the sequence number X is discarded by the probabilistic loss service, i.e r<=p

Server side(here p = 0.5)

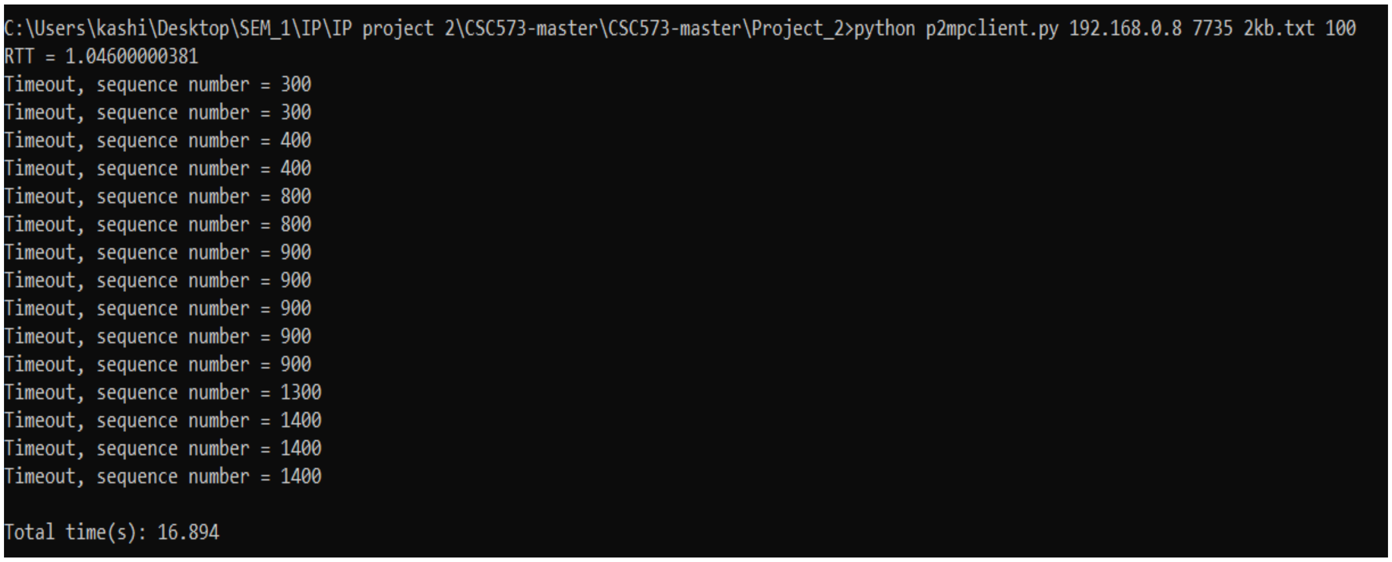


```
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\CSC573-master\CSC573-master\Project_2>python p2mpserver.py 7735 sample1.txt 0.5
P2MP-FTP Server is initialized and listing ...
Packet loss, sequence number = 300
Packet loss, sequence number = 300
Packet loss, sequence number = 400
Packet loss, sequence number = 400
Packet loss, sequence number = 800
Packet loss, sequence number = 800
Packet loss, sequence number = 900
Packet loss, sequence number = 900
Packet loss, sequence number = 900
Packet loss, sequence number = 900
Packet loss, sequence number = 900
Packet loss, sequence number = 1300
Packet loss, sequence number = 1400
Packet loss, sequence number = 1400
Packet loss, sequence number = 1400
Complete!
```

Fig 3.

Output 2 : If the timeout occurs for a packet with sequence number Y,

Client side



```
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\CSC573-master\CSC573-master\Project_2>python p2mpclient.py 192.168.0.8 7735 2kb.txt 100
RTT = 1.04600000381
Timeout, sequence number = 300
Timeout, sequence number = 300
Timeout, sequence number = 400
Timeout, sequence number = 400
Timeout, sequence number = 800
Timeout, sequence number = 800
Timeout, sequence number = 900
Timeout, sequence number = 900
Timeout, sequence number = 900
Timeout, sequence number = 900
Timeout, sequence number = 900
Timeout, sequence number = 1300
Timeout, sequence number = 1400
Timeout, sequence number = 1400
Timeout, sequence number = 1400

Total time(s): 16.894
```

Fig 4.

Total delay time: 16.894 secs.

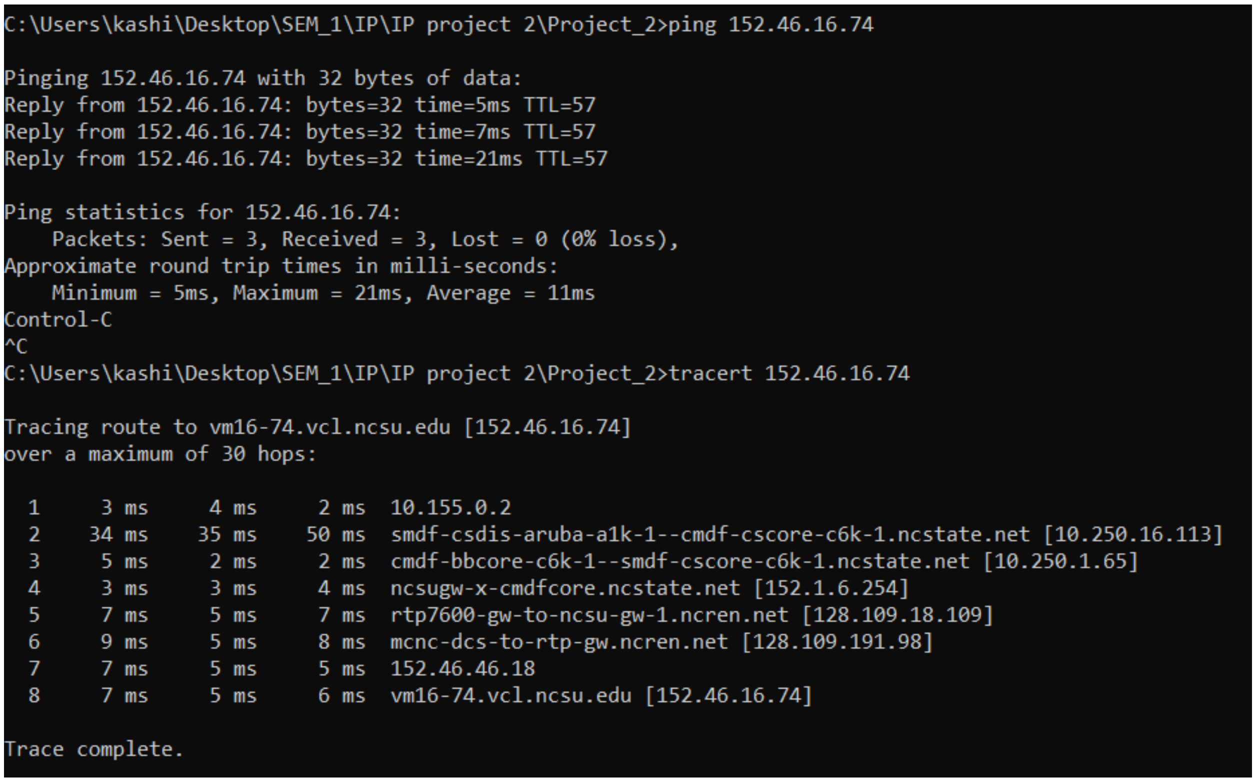
KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)

Test Environment

The test was ran over NCSU campus network. We reserved 5 NCSU EOS machines with ip addresses as specified below:

152.46.16.74
152.46.16.123
152.46.19.175
152.46.18.200
152.7.99.139

and ran the receivers (P2MP-FTP server) on them. And the sender (P2MP-FTP client) was ran on our laptop which was connected to NCSU WIFI. The route from sender to receiver had multiple-hops as shown below:



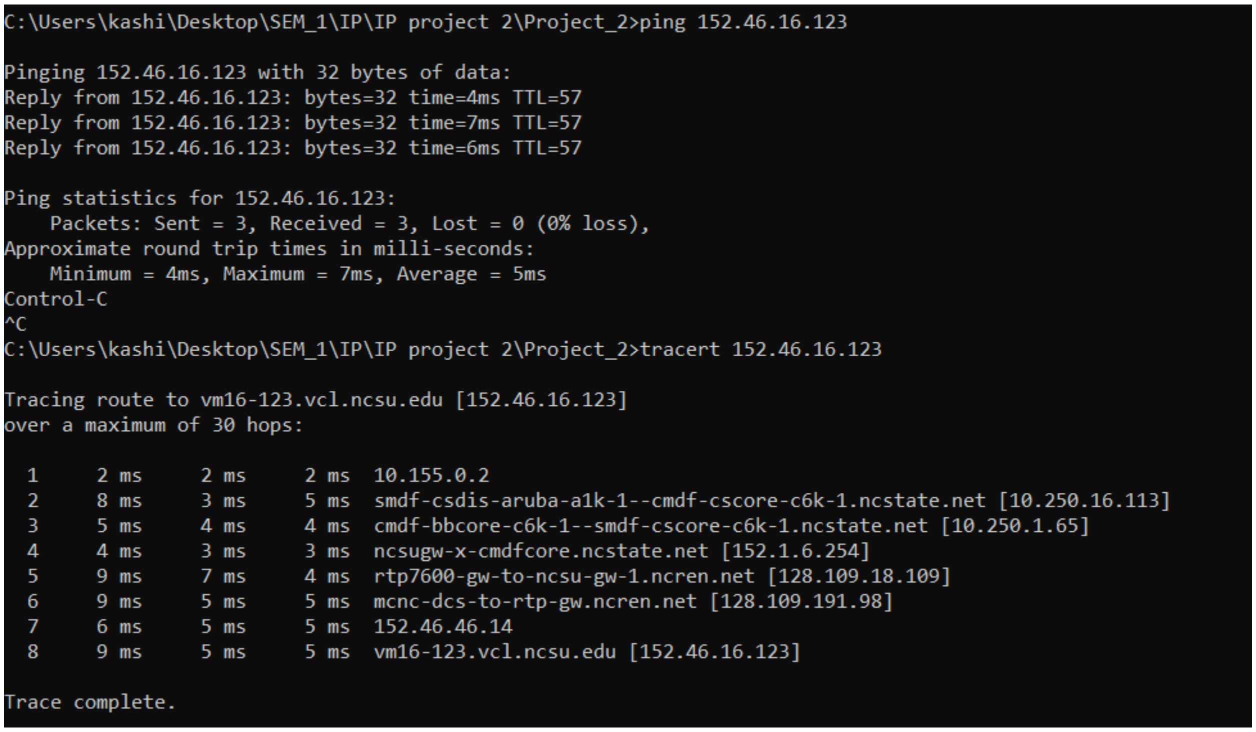
```
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\Project_2>ping 152.46.16.74

Pinging 152.46.16.74 with 32 bytes of data:
Reply from 152.46.16.74: bytes=32 time=5ms TTL=57
Reply from 152.46.16.74: bytes=32 time=7ms TTL=57
Reply from 152.46.16.74: bytes=32 time=21ms TTL=57

Ping statistics for 152.46.16.74:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 5ms, Maximum = 21ms, Average = 11ms
Control-C
^C
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\Project_2>tracert 152.46.16.74

Tracing route to vm16-74.vcl.ncsu.edu [152.46.16.74]
over a maximum of 30 hops:

  1     3 ms     4 ms     2 ms  10.155.0.2
  2    34 ms    35 ms    50 ms  smdf-csdis-aruba-a1k-1--cmdf-cscore-c6k-1.ncstate.net [10.250.16.113]
  3     5 ms     2 ms     2 ms  cmdf-bbcore-c6k-1--smdf-cscore-c6k-1.ncstate.net [10.250.1.65]
  4     3 ms     3 ms     4 ms  ncsugw-x-cmdfcore.ncstate.net [152.1.6.254]
  5     7 ms     5 ms     7 ms  rtp7600-gw-to-ncsu-gw-1.ncren.net [128.109.18.109]
  6     9 ms     5 ms     8 ms  mcnc-dcs-to-rtp-gw.ncren.net [128.109.191.98]
  7     7 ms     5 ms     5 ms  152.46.46.18
  8     7 ms     5 ms     6 ms  vm16-74.vcl.ncsu.edu [152.46.16.74]

Trace complete.
```

Fig 5.

How did we set the rtt value and timeout_interval:

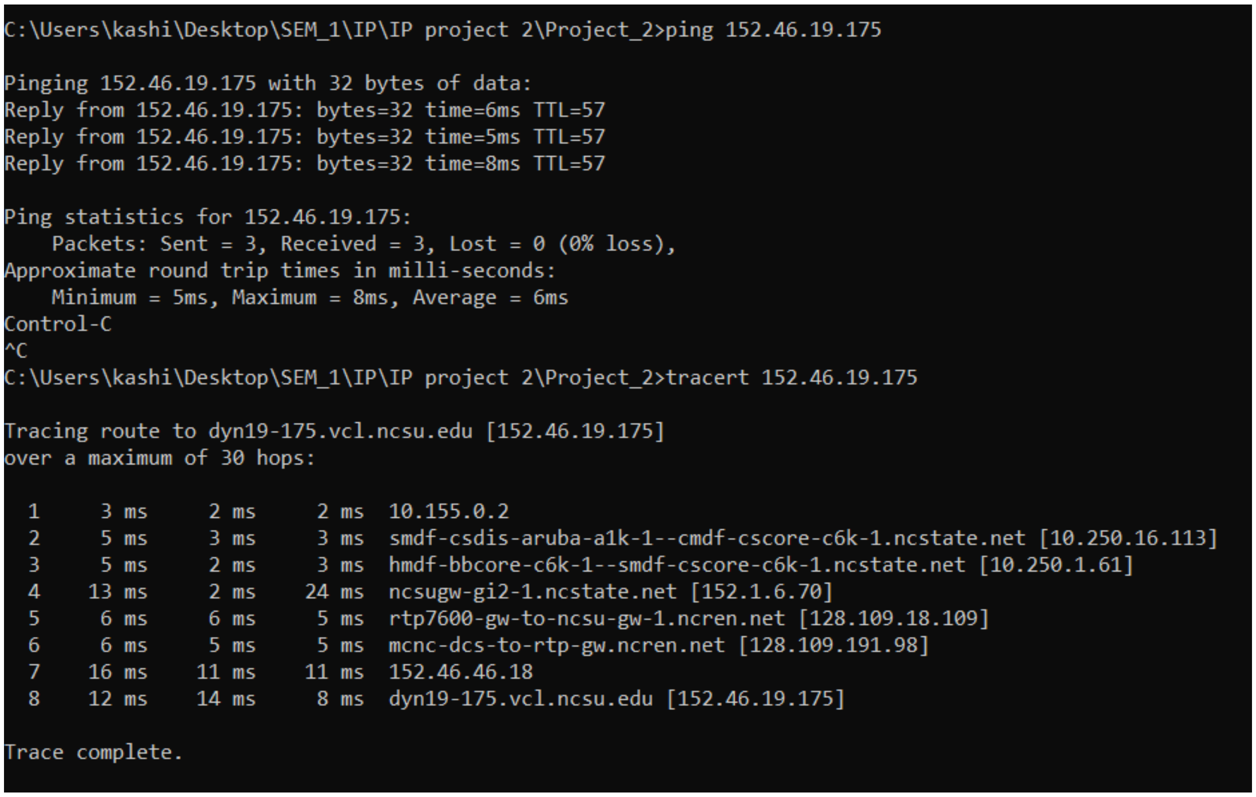
```
rtt = http_request.elapsed.total_seconds()
rtt = time.time() - start_time
if rtt > timeout_interval:
    timeout_interval = rtt
```

KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)



```
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\Project_2>ping 152.46.16.123

Pinging 152.46.16.123 with 32 bytes of data:
Reply from 152.46.16.123: bytes=32 time=4ms TTL=57
Reply from 152.46.16.123: bytes=32 time=7ms TTL=57
Reply from 152.46.16.123: bytes=32 time=6ms TTL=57

Ping statistics for 152.46.16.123:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 7ms, Average = 5ms
Control-C
^C
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\Project_2>tracert 152.46.16.123

Tracing route to vm16-123.vcl.ncsu.edu [152.46.16.123]
over a maximum of 30 hops:

  1     2 ms     2 ms     2 ms  10.155.0.2
  2     8 ms     3 ms     5 ms  smdf-csdis-aruba-a1k-1--cmdf-cscore-c6k-1.ncstate.net [10.250.16.113]
  3     5 ms     4 ms     4 ms  cmdf-bbcore-c6k-1--smdf-cscore-c6k-1.ncstate.net [10.250.1.65]
  4     4 ms     3 ms     3 ms  ncsugw-x-cmdfcore.ncstate.net [152.1.6.254]
  5     9 ms     7 ms     4 ms  rtp7600-gw-to-ncsu-gw-1.ncren.net [128.109.18.109]
  6     9 ms     5 ms     5 ms  mcnc-dcs-to-rtp-gw.ncren.net [128.109.191.98]
  7     6 ms     5 ms     5 ms  152.46.46.14
  8     9 ms     5 ms     5 ms  vm16-123.vcl.ncsu.edu [152.46.16.123]

Trace complete.
```

Fig 6.



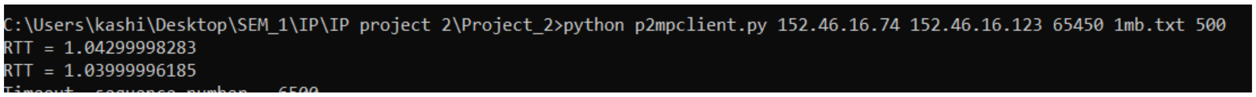
```
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\Project_2>ping 152.46.19.175

Pinging 152.46.19.175 with 32 bytes of data:
Reply from 152.46.19.175: bytes=32 time=6ms TTL=57
Reply from 152.46.19.175: bytes=32 time=5ms TTL=57
Reply from 152.46.19.175: bytes=32 time=8ms TTL=57

Ping statistics for 152.46.19.175:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 5ms, Maximum = 8ms, Average = 6ms
Control-C
^C
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\Project_2>tracert 152.46.19.175

Tracing route to dyn19-175.vcl.ncsu.edu [152.46.19.175]
over a maximum of 30 hops:

  1     3 ms     2 ms     2 ms  10.155.0.2
  2     5 ms     3 ms     3 ms  smdf-csdis-aruba-a1k-1--cmdf-cscore-c6k-1.ncstate.net [10.250.16.113]
  3     5 ms     2 ms     3 ms  hmdf-bbcore-c6k-1--smdf-cscore-c6k-1.ncstate.net [10.250.1.61]
  4    13 ms     2 ms    24 ms  ncsugw-gi2-1.ncstate.net [152.1.6.70]
  5     6 ms     6 ms     5 ms  rtp7600-gw-to-ncsu-gw-1.ncren.net [128.109.18.109]
  6     6 ms     5 ms     5 ms  mcnc-dcs-to-rtp-gw.ncren.net [128.109.191.98]
  7    16 ms    11 ms    11 ms  152.46.46.18
  8    12 ms    14 ms     8 ms  dyn19-175.vcl.ncsu.edu [152.46.19.175]

Trace complete.
```

Fig 7.

KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)

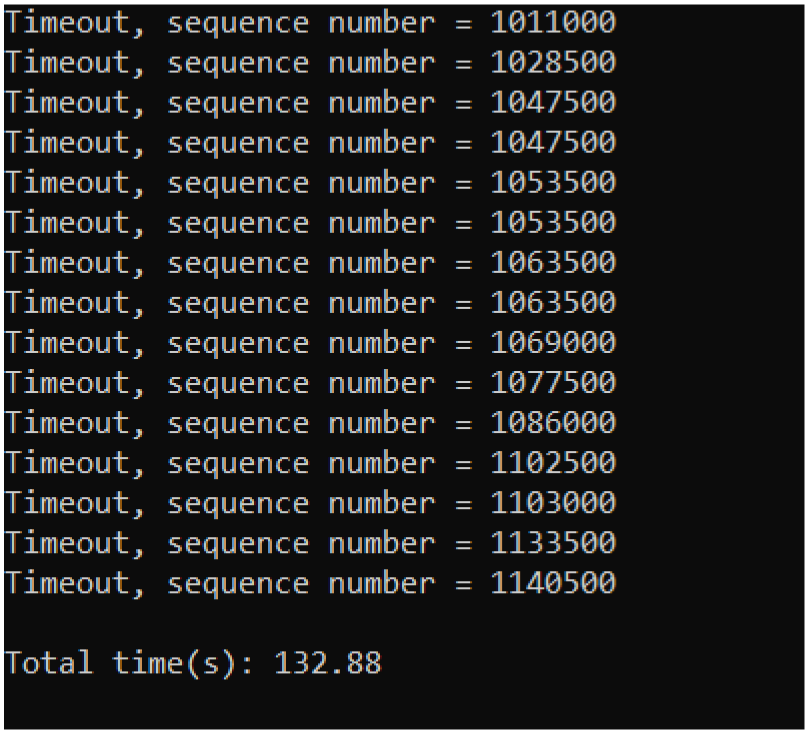
Output

The snapshots below show a sample of the sender or client output.



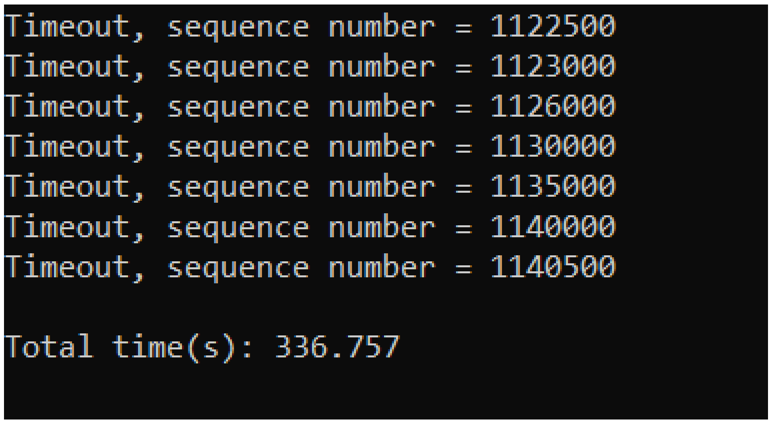
```
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\Project_2>python p2mpclient.py 152.46.16.74 152.46.16.123 65450 1mb.txt 500
RTT = 1.04299998283
RTT = 1.03999996185
Timeout, sequence number = 6500
```

Fig 8.



```
Timeout, sequence number = 1011000
Timeout, sequence number = 1028500
Timeout, sequence number = 1047500
Timeout, sequence number = 1047500
Timeout, sequence number = 1053500
Timeout, sequence number = 1053500
Timeout, sequence number = 1063500
Timeout, sequence number = 1063500
Timeout, sequence number = 1069000
Timeout, sequence number = 1077500
Timeout, sequence number = 1086000
Timeout, sequence number = 1102500
Timeout, sequence number = 1103000
Timeout, sequence number = 1133500
Timeout, sequence number = 1140500

Total time(s): 132.88
```

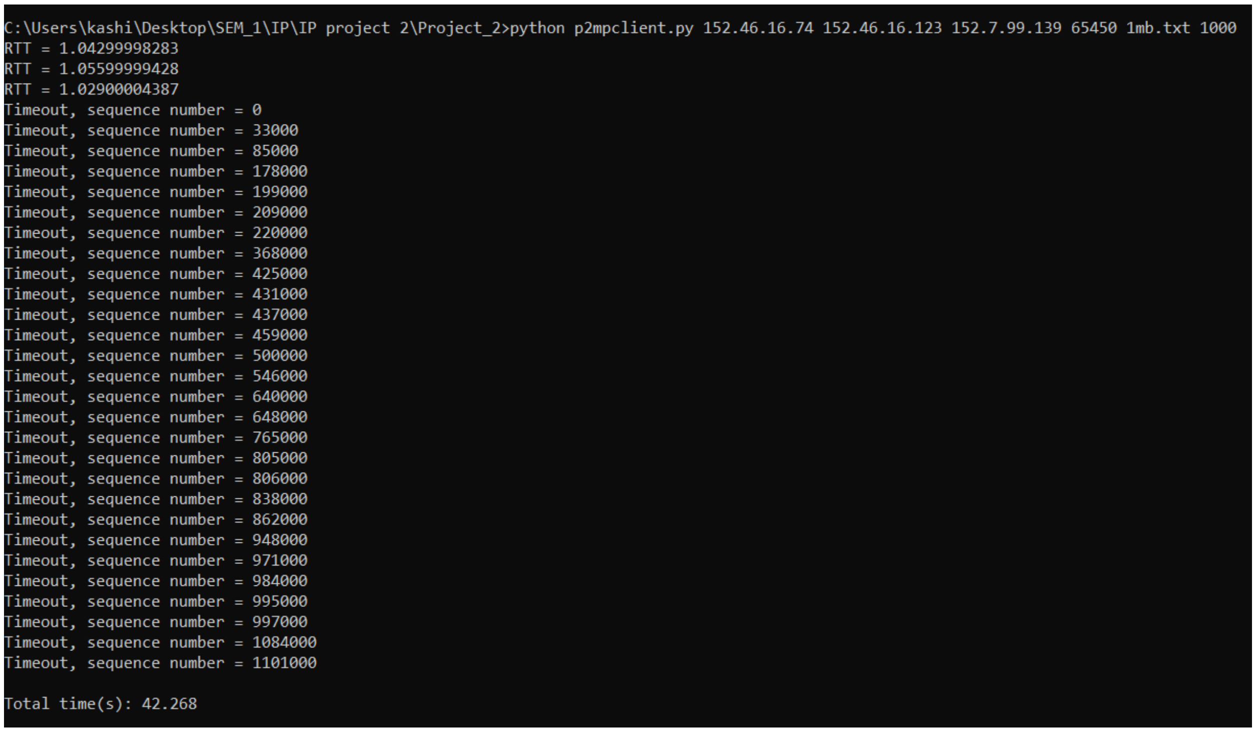
Fig 9.



```
Timeout, sequence number = 1122500
Timeout, sequence number = 1123000
Timeout, sequence number = 1126000
Timeout, sequence number = 1130000
Timeout, sequence number = 1135000
Timeout, sequence number = 1140000
Timeout, sequence number = 1140500

Total time(s): 336.757
```

Fig 10.

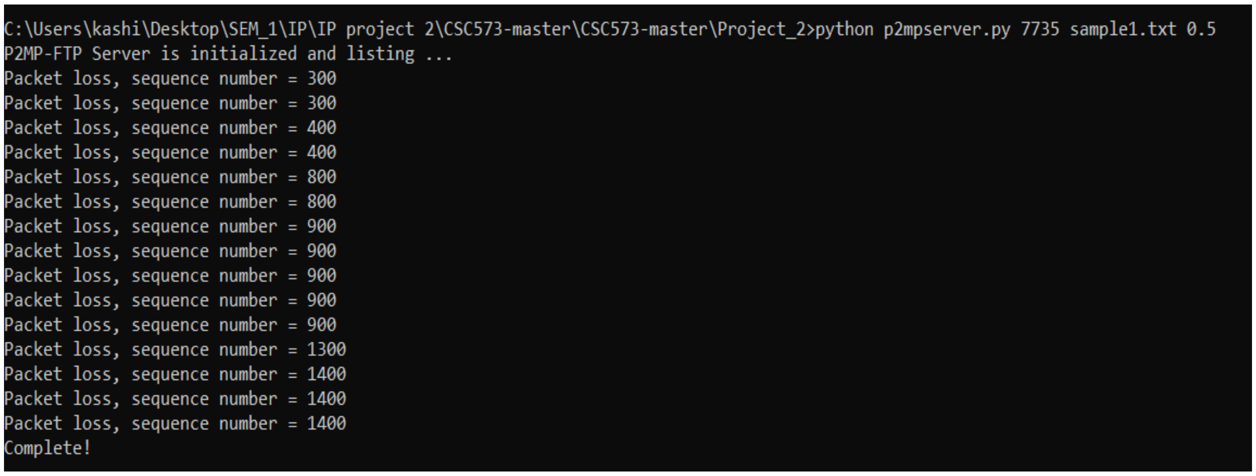
KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)



```
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\Project_2>python p2mpclient.py 152.46.16.74 152.46.16.123 152.7.99.139 65450 1mb.txt 1000
RTT = 1.04299998283
RTT = 1.05599999428
RTT = 1.02900004387
Timeout, sequence number = 0
Timeout, sequence number = 33000
Timeout, sequence number = 85000
Timeout, sequence number = 178000
Timeout, sequence number = 199000
Timeout, sequence number = 209000
Timeout, sequence number = 220000
Timeout, sequence number = 368000
Timeout, sequence number = 425000
Timeout, sequence number = 431000
Timeout, sequence number = 437000
Timeout, sequence number = 459000
Timeout, sequence number = 500000
Timeout, sequence number = 546000
Timeout, sequence number = 640000
Timeout, sequence number = 648000
Timeout, sequence number = 765000
Timeout, sequence number = 805000
Timeout, sequence number = 806000
Timeout, sequence number = 838000
Timeout, sequence number = 862000
Timeout, sequence number = 948000
Timeout, sequence number = 971000
Timeout, sequence number = 984000
Timeout, sequence number = 995000
Timeout, sequence number = 997000
Timeout, sequence number = 1084000
Timeout, sequence number = 1101000

Total time(s): 42.268
```

Fig 11.

The snapshots below show a sample of the Receiver or server output.



```
C:\Users\kashi\Desktop\SEM_1\IP\IP project 2\CSC573-master\CSC573-master\Project_2>python p2mpserver.py 7735 sample1.txt 0.5
P2MP-FTP Server is initialized and listing ...
Packet loss, sequence number = 300
Packet loss, sequence number = 300
Packet loss, sequence number = 400
Packet loss, sequence number = 400
Packet loss, sequence number = 800
Packet loss, sequence number = 800
Packet loss, sequence number = 900
Packet loss, sequence number = 900
Packet loss, sequence number = 900
Packet loss, sequence number = 900
Packet loss, sequence number = 900
Packet loss, sequence number = 1300
Packet loss, sequence number = 1400
Packet loss, sequence number = 1400
Packet loss, sequence number = 1400
Complete!
```

Fig 12.

KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)

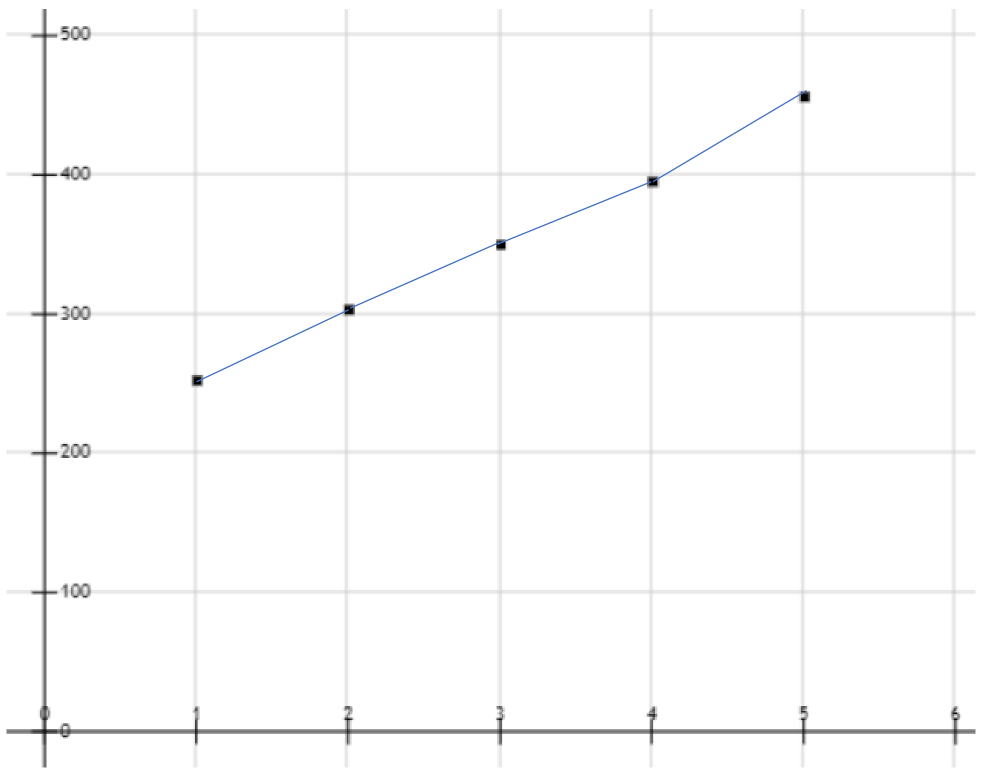
Task 1: Effect of Receiver Set Size n
For this first task, we set the MSS to 500 bytes and the loss probability p = 0.05. We ran the P2MP-FTP protocol to transfer a file of 2MB size, and vary the number of receivers n = 1, 2, 3, 4, 5. For each value of n, file transfer was ran 5 times, time of the data transfer (i.e., delay) was averaged over the five transmissions to arrive at the average delay. The below table lists the values generated by the test:

Receivers	MSS	Probability of error	Average delay(sec)
1	500	0.05	252.88
2	500	0.05	303.757
3	500	0.05	350.47
4	500	0.05	395.75
5	500	0.05	456.79

Table 1 – Task 1 data

GRAPH……………………

Average delay



Number of Receivers

Observation and Conclusion
As we can see from the graph, the time required to transfer the file increases almost linearly with increase in the number of receivers considering a constant value of MSS and P. When we used just 1 receiver we can clearly see that we had the lowest average delay value i.e ~253 seconds. With each additional receiver, the sender will incur an additional RTT which was also visible in the fig 11 attached above and the probability of retransmission also increases by a factor of 0.05 (Independent events). This finally increases the avg delay required in the transmission of each file.

Task 2: Effect of MSS
In this experiment, the number of receivers was fixed at n = 3 and the loss probability at p = 0.05. We ran the P2MP-FTP protocol to transfer a 2Mb file, and the MSS value was varied from 100 bytes to 1000 bytes by increasing every consecutive MSS value by 100 bytes. For each value of MSS, file transfer was repeated 5 times, and the average delay over five transactions was calculated. The below table lists the values generated during the task,

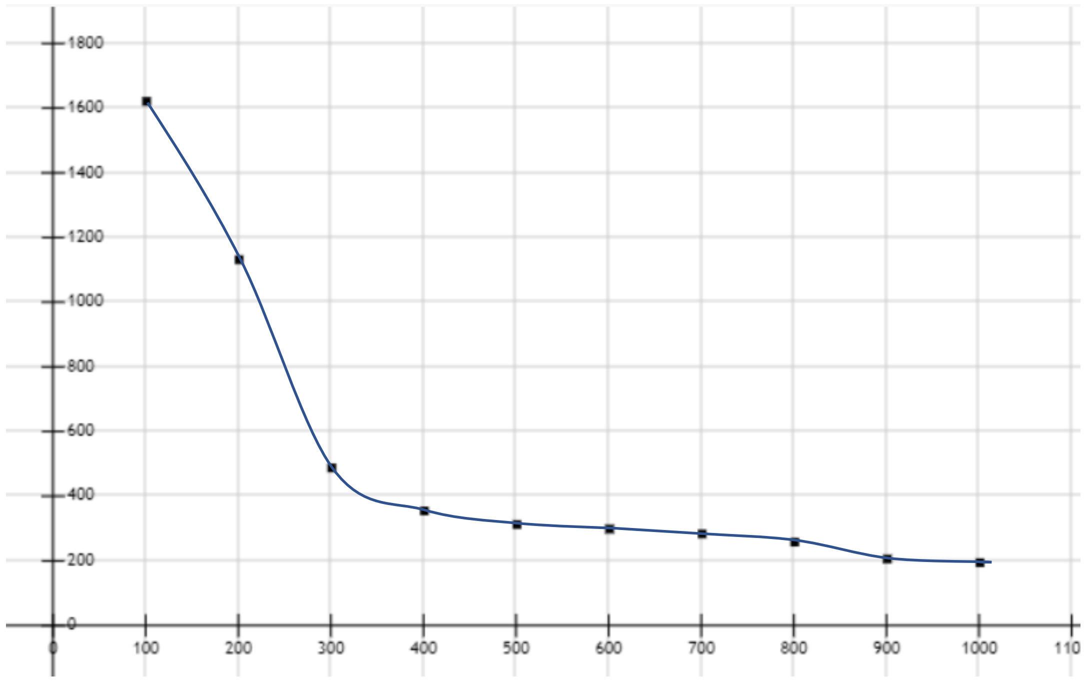
Receivers	MSS	Probability of error	Average delay(sec)
3	100	0.05	1623.71
3	200	0.05	1134.21
3	300	0.05	490.315
3	400	0.05	357.215
3	500	0.05	315.22
3	600	0.05	301.157
3	700	0.05	285.913
3	800	0.05	260.71
3	900	0.05	208.63
3	1000	0.05	197.07

Table 2 – Task 2 data

KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)

GRAPH…………………..

Average Delay



MSS

Observation and Conclusion

The delay for MSS=200 was nearly half that of delay and MSS=100 and substantially fell down to 490 for MSS = 300. The average delay drops dramatically, almost in thirds, with each increase in MSS value. But as the MSS value increases, we see marginal improvements in declination in the average delay time, this behavior can be explained by:

● When MSS is lower, we need a large number of segments to transfer a file of same size, this results in increased propagation delay and processing delay and as well as the delay in receiving ACK.

● Since Loss Probability, P, is a function of number of packets, we experience a large number of packet loss with lower MSS values. This adds additional delay due to Timeouts and retransmissions.

But this trend of increase in the speed of transmission i.e decrease in the average delay was not maintained after certain values of MSS and later on we can see that the speed of transmission is getting saturated.

So with lower size of the MSS, there is an increase in average delay and loss of packets also increases leading to high average delay and with larger MSS packets size the average delays decreases thus speed in transmission increases as observed in the graph above.

KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)

Task 3: Effect of Loss Probability, P
For this task, we set the MSS to 500 bytes and the number of receivers are fixed at n = 3. We tested the P2MP-FTP protocol transfer using the same 2Mb file, and varied the loss probability from p = 0.01 to p = 0.10 in increments of 0.01. For each value of p, file transmission was repeated 5 times, and computed the average delay over the five transfers. The below table lists the values generated during the task,

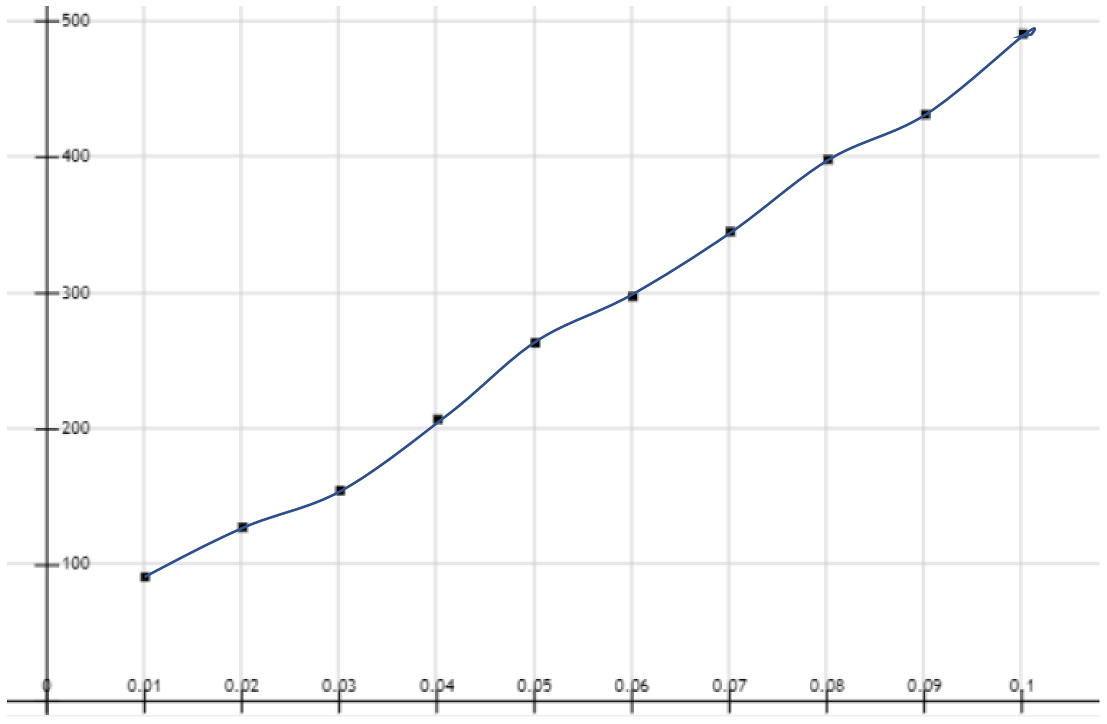
Receivers	MSS	Probability of error	Average delay(sec)
3	500	0.01	92.268
3	500	0.02	128.64
3	500	0.03	155.667
3	500	0.04	207.926
3	500	0.05	264.166
3	500	0.06	298.174
3	500	0.07	345.895
3	500	0.08	398.898
3	500	0.09	432.134
3	500	0.1	491.235

Table 3 – Task 3 data

KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)

GRAPH……………………

Average Delay



Probability of Packet Loss

Observation and Conclusion
From the graph we can clearly see a linear increase in delay with increase in loss probability. We have 3 receivers that are dropping packets independently of each other.
By design, the sender will not proceed unless and until everyone in the receiver set has acknowledged the reception of a segment. This mechanism will keep the other receivers waiting even if a single receiver drops a segment. Due to this behavior, the file transfer delay increases linearly with each step increase in loss probability. This is also clearer mathematically that a probability of random value selected below 0.01 is always lower that the value below 0.1. So automatically with increase in the value of p the packet loss increases and thus the time delay also increases.

KASHISH SINGH (ksingh9)
SATHWIK KALVAKUNTLA (skalvak)

FINAL Conclusion:

After the p2mp FTP transfer protocol tasks experiments were completed, we can conclude that,

1. From the task 1 set of experiments it was clearly observed that more the size of receiver set more will be the observed average delay. The average delay increases linearly with the number of receivers.

2. The observations from the set of Task 2 experiments we got a conclusive evidence that the large MSS specifications result in short average delays. The average delay decreases exponentially with increase in the MSS specifications.

3. The task 3 experiments verified that the large packet loss probability result in long average delays. The average delay increases exponentially with large deviations in packet loss probability and linearly with short deviations in the packet loss probability.