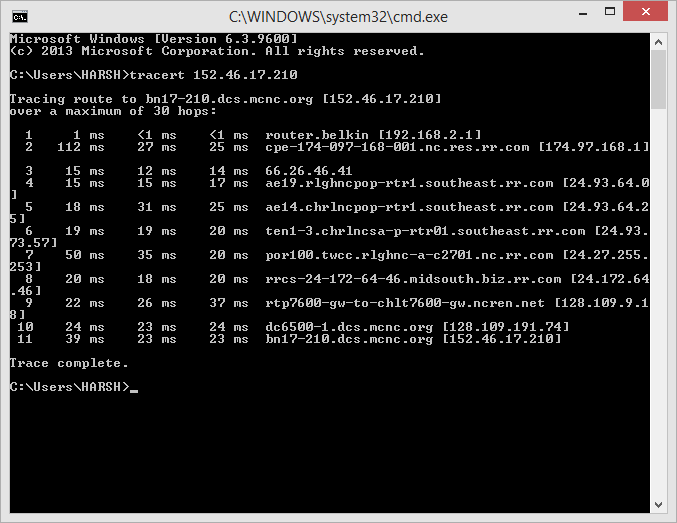
**PROJECT REPORT**

The following are the parameters used in the program for running P2MP-FTP protocol :

File Size : 1MB

RTT : 25\*4=100ms

An image of the traceroute from Client(our machines) to server(VCL machines) is shown as below:



1. **Task - 1:**

**MSS : 500 bytes**

**Loss Probability : 0.05**

**Fig-1 : Numbers of servers v/s Average delay**

|  |  |
| --- | --- |
| nO. of SERVERS | Average Delay (in secs) |
| 1 | **62.4** |
| 2 | **113.2** |
| 3 | **165.75** |
| 4 | **210.5** |
| 5 | **267.3** |

Table of Values

From the above graph, it can be concluded that as the number of receivers (Servers) increases, the average delay in transferring the file to all the receivers also increases linearly. This is because as per the design, the sender sends out the packet (of 1 MSS size) to all the receivers together. Only when it receives the ACK from each of these receivers does it send the next packet. There is a loss probability associated with each of the receivers (servers), that means that each server randomly drops a packet to simulate packet loss. Now as the number of receivers increase, the probability that the given packet is received with a drop goes on increasing. Each time any of the receiver drops a packet, the sender (client here) has to wait for 'timeout' duration before re- transmitting the packet only to that specific receiver that dropped the packet. Only when all receivers acknowledge the receipt of a particular packet does the client moves on to the transmission of next packet.

Because of this, with increase in number of server, probability of packet getting dropped in first attempt decreases, and this increases the overall cumulative delay for file transmission.

1. **Task - 2: Effect of MSS**

**No. of Receivers : 3**

**loss probability : 0.05**

**Fig-2 : MSS Size v/s Average delay**

SSSS

|  |  |
| --- | --- |
| MSS (in Bytes) | Average Delay (in secs) |
| 100 | **750.6** |
| 200 | **600** |
| 300 | **465** |
| 400 | **367** |
| 500 | **335** |
| 600 | **301** |
| 700 | **275** |
| 800 | **260** |
| 900 | **252** |
| 1000 | **241** |

Table of Values

From the above graph, it can be concluded that as the size of the MSS increases, the average delay in transferring the file to all the receivers decreases exponentially initially, but after certain point the average delay tends to be constant as MSS is increased.

This is as per expectation. MSS has a diminishing effect on the total receiving delay because since the loss probability and the number of servers are fixed, the amount of traffic dropped/lost requires re-transmission and this can be considered more or less fixed for all the test runs (although it is decided by random number but considering over large runs the number of drops come out to be more or less same). For low values of MSS, we have larger number of packets that need to be sent out. As the number of packets increases, the number of packets that get dropped will also increase and hence requiring retransmission. Since each loss of packet on any of the receiver requires the sender to re-transmit and then wait for ACK, this increases the total file transfer time. This delay is even more prevalent for small values of MSS as there would be large number of packets to be transmitted and hence larger number of losses. This means more re-transmissions and more time spent during timeout for packet loss detection.

1. **Task - 3: Effects of loss probability 'p'**

**No. of Receivers : 3**

**MSS Value : 500 bytes**

**Fig-3 : Packet Loss Probability v/s Average delay**

|  |  |
| --- | --- |
| Loss Probability | Average Delay (in secs) |
| 0.01 | 219.5 |
| 0.02 | 227.5 |
| 0.03 | 244.5 |
| 0.04 | 252.3 |
| 0.05 | 259 |
| 0.06 | 261 |
| 0.07 | 262.2 |
| 0.08 | 269 |
| 0.09 | 275.4 |
| 0.1 | 286.9 |

From the above graph, it can be concluded that as the loss probability of a packet increases, the average delay in transferring the file to all the receivers also increases.

This is because as the probability of loss increases all the receivers start dropping packets with greater probability. This results in increased time spent in waiting for ACKs from all the receiver, and causes time out at sender side and then re-transmission of same packets. This effect is not as prevalent as increasing the number of receivers because the number of transmissions required will be for lesser if the number of receivers are less. With increased number of receivers the total probability of a loss in the system is multiplied.

But nonetheless, with increase in the value of loss probability, the value of the total time required for file transfer does increase due to above stated reason.

**Conclusions**

From the graphs of Task-1, Task-2, and Task-3, it can be concluded that P2MP-FTP is not scalable. As even one of the parameter that we considered (MSS, Loss Probability or Number of receivers) increase the total file transfer time starts to jump. This is especially worse when the number of receivers increases. Hence, this protocol doesn't have good host scalability. Even with 5 receivers and small amount of packet probability of 0.05, the file transfer was taking roughly about 4 minutes. We cannot extend the same architecture to large number of receivers.

From our experiments we conclude that for efficient file transfers with multiple transfer, the loss probability should be extremely small and MSS should be large. In other words, the network should be reliable with very less packet loss in order to send very big files from one host to multiple hosts. Both of these criterion are difficult to meet in the real time network where the MSS is restricted to prevent excessive fragmentation and since the internet is highly unreliable its extremely difficult to provide the kind of reliability guarantees that the P2MP-FTP desires. Hence, it doesn't scale due to serious performance issues.