Point to Multi-point Reliable Data Transfer Protocol

**Unity ID: nsompal, nvenugo2**

11/20/2017

# IP Project2

***Objective:***

The objective of the project is to implement point to multi point reliable data transfer protocol using the stop and wait ARQ scheme. A simple application protocol has been defined to make sure that the Client and Servers follow precisely the specifications of the protocol to accomplish particular tasks. Client has been modelled to act as the sender and multicasts the file to multiple receivers(servers). UDP socket is created with reliable data transfer services using Socket Programming in Python.

***Description:***

In this project, a Client (acting as a sender) multicasts a file greater than 10 MB in size to multiple servers (acting as receivers) through UDP socket. The UDP socket provides a simple service protocol for file transfer, hence it is unreliable. So, reliable data transfer protocol features have been added over the simple service UDP protocol. The following features have been added to make the file transfer reliable:

* Timeout for the segments
* Retransmission in case of Timeout or Packet Loss
* Checksum
* Sequence numbers for segments
* Acknowledgements for the received segments
* Data/ ACK identifier fields in the header

***Client:***

The client transmits a single file to multiple servers. The Byte stream has the file contents broken into segments, each of size MSS (Maximum Segment Size), which is a command line input by the user. The Client transmits on the port 7735. It also listens for the ACKs on the same port. The socket created for transmission of segments is used to communicate with all the receivers from the same port. Each segment is appended with a header having the following fields:

* 32-bit sequence number
* 16-bit data identifier (0101010101010101)
* 16-bit checksum

Since it follows Stop and Wait ARQ protocol, it waits for the ACK from the receiver for each segment successfully received. If the ACK is not received for the specified Timeout value, the segment is retransmitted for that particular receiver. In this manner, the complete file is transmitted.

***Random Probabilistic Loss:***

Since, the file transfer happens over the existing Internet connection between the hosts, there is no possibility of packet loss. Hence, a random probabilistic loss generator function has been implemented. This generates a random probability(r) which is compared to the probability given by the user through command line input(p). The segments are dropped if r<p and hence enables to test the retransmission feature in case of packet loss.

***Receiver:***

More than one receiver can simultaneously receive packets from the client(multicast). The Servers listen on the port 7735. Once the segment is received (with r > p), it unpacks the segment and calculates the checksum for the data received. The computed checksum is compared with the received checksum in the segment. If both checksums match, then an ACK packet with an ACK identifier (1010101010101010) and sequence number of the received packet is sent back to the Client, indicating successful transmission.

***Test Environment:***

The test tasks have been carried out using a 11.4 MB file from a client to multiple servers connected over campus Wi-Fi and wired networks. The Timeout parameter has been set to 300 ms.

The Performance has been evaluated using the following tasks:

* Task1: Effect of Receiver Set Size on total delay
* Task2: Effect of MSS on Total delay
* Task3: Effect of Loss Probability on total delay

***Task1 : Effect of Receiver Set Size on total delay:***

The 11.4 MB file is transferred to n receivers with MSS = 500 bytes and Loss probability p = 0.05. n varies from 1 to 5 i.e., to 1 receiver and then to 2 receivers and so on. This is repeated till n is 5. Each experiment for a receiver is carried out 5 times and the average is calculated and plotted as per the values from the below table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***No of Receivers*** | **Run1** | **Run2** | **Run3** | **Run4** | **Run5** | **Average Delay** |
| *1* | 595 | 562 | 608 | 585 | 574 | 584.8 |
| *2* | 1146 | 1056 | 1187 | 1101 | 1097 | 1117.4 |
| *3* | 1414 | 1497 | 1465 | 1384 | 1402 | 1432.4 |
| *4* | 1670 | 1605 | 1520 | 1660 | 1586 | 1608.2 |
| *5* | 1950 | 1947 | 1994 | 1923 | 1986 | 1960 |

***Analysis of Curve:***

Graph plotted with the observed results is linearly increasing with the value of n . When the number of receivers increases the data transfer delay also increases. Since, single socket bound to the same port is used to communicate to all the receivers, the serving time decreases to a multiple of 1/n. An increase in the number of receivers leads to an increase of number of segments transferred and number of retransmissions.

***Task2 : Effect of MSS on total delay:***

The 11.4 MB file is transferred to 3 receivers with MSS varying from 100 to 1000bytes in steps of 100 and Loss probability p = 0.05. Each experiment for a receiver is carried out 5 times and the average is calculated and plotted as per the values from the below table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***MSS*** | **Run1** | **Run2** | **Run3** | **Run4** | **Run5** | **Average Delay** |
| *100* | 6527 | 6480 | 6522 | 6636 | 6419 | 6516.8 |
| *200* | 3541 | 3491 | 3420 | 3589 | 3411 | 3490.4 |
| *300* | 2545 | 2490 | 2430 | 2599 | 2398 | 2492.6 |
| *400* | 1823 | 1764 | 1701 | 1890 | 1879 | 1811.4 |
| *500* | 1420 | 1511 | 1440 | 1308 | 1376 | 1410 |
| *600* | 1252 | 1203 | 1247 | 1356 | 1185 | 1248.6 |
| *700* | 1089 | 1020 | 976 | 1145 | 1167 | 1079.4 |
| *800* | 1121 | 1168 | 1205 | 1091 | 1084 | 1133.8 |
| *900* | 806 | 887 | 919 | 828 | 783 | 844.6 |
| *1000* | 731 | 768 | 710 | 749 | 686 | 728.8 |

***Analysis of Curve:***

Graph plotted with the observed results decreases as the value of MSS increases. Since the number of bytes transferred at each transmission increase when MSS increases, the total number of segments transferred will decrease. Hence data transfer delay would require lesser number of RTTs. Also, the number of loss segments and retransmitted segments will be lesser when MSS increases.

***Task3 : Effect of Loss Probability on total delay:***

The 11.4 MB file is transferred to 3 receivers with MSS = 500 bytes and Loss probability p varying from 0.01 to 0.10 in steps of 0.01. Each experiment for a receiver is carried out 5 times and the average is calculated and plotted as per the values from the below table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Probability p*** | **Run1** | **Run2** | **Run3** | **Run4** | **Run5** | **Average Delay** |
| *0.01* | 664 | 604 | 593 | 670 | 709 | 648 |
| *0.02* | 809 | 895 | 872 | 786 | 792 | 830.8 |
| *0.03* | 953 | 908 | 982 | 949 | 875 | 933.4 |
| *0.04* | 1091 | 1078 | 1059 | 1176 | 992 | 1079.2 |
| *0.05* | 1479 | 1459 | 1491 | 1348 | 1329 | 1421.2 |
| *0.06* | 1617 | 1721 | 1643 | 1589 | 1519 | 1617.8 |
| *0.07* | 1897 | 1992 | 1840 | 1857 | 1753 | 1867.8 |
| *0.08* | 2145 | 2189 | 2064 | 1993 | 2348 | 2147.8 |
| *0.09* | 2525 | 2687 | 2554 | 2365 | 2484 | 2523 |
| *0.10* | 2778 | 2858 | 2677 | 2864 | 2911 | 2817.6 |

***Analysis of Curve:***

Graph plotted with the observed results increases as the value of Loss Probability (p) increases. Increased value of p leads to greater probability of packet loss at the receivers. Since the number of segments dropped at the receivers increase with the value of p, total number of retransmissions also increase. This leads to increase in delay.

***Conclusion:***

The receiver set size, MSS and Loss probability values affect the scalability of the P2MP – FTP system. For systems with a large number of receivers and/or huge loss values we can improve system performance by using large values of MSS. This facilitates better utilization of the network by means of lesser number of transfers. In contrast, for systems containing not many servers nominal performance can be obtained even when the MSS is small. Thus scalability of the systems can be obtained by the proper selection of these parameters.