Specifiction of My Transport Protocol(MTP)

Introduction

In this assignment, a reliable transport protocol over the UDP protocol has been implemented. This reliable transport protocol is called My Transport Protocol (MTP). MTP implements features listing below:

1. A threeway handshake for the connection establishment.
2. MTP Sender maintains a single-timer for timeout operation.
3. MTP Sender implements the simplified TCP sender and fast retransmit.
4. MTP Sender is able to deal with different maximum segment size (MSS).
5. The maximum number of un-acknowledged bytes that the MTP Sender can have is Maximum Window size (MWS).
8. MTP Sender implement a Packet Loss (PLD) Module.
9. MTP Sender uses a constant timeout.
10. MTP Receiver receives the data packet, and acknowledge it immediately.
10. Both MTP Sender and Receiver print a log file while the file transmission process.

The structure of my MTP header

The MTP header figure is like:

SYN	ACK		Sequence #	Acknowledgment #	Data Length	Payload
1 bit	1 bit	Other 6 bits	4 bytes	4 bytes	4bytes	Mss bytes

My MTP header have 4 field. First one is a one byte flag field, which has 8 bits. Each bit indicates one status. MY MTP only used 2 field which are SYN and ACK, and keep other 6 bits empty for extention purpose. Second one is a 4 bytes field used to indicate the sequence number. Third one is a 4 bytes field used to indicate the acknowledgement number. Fourth one is a 4 bytes field used to indicate the data length in the Payload field. Then the last mss bytes used to store the payload.

Mechanism description

Three-way handshake

In the beginning, the MTP sender send a SYN segment(SYN flag field set to 1) to MTP Receiver to establish a connection. After MTP receiver receives the SYN segment, it will immediately send a SYNACK segment(SYN and ACK flag field set to 1) back to MTP sender. When MTP Sender receives the SYNACK segment, it knows that the MTP Receiver is ready. Finally, it sends a ACK segment(ACK flag field set to 1) to MTP Receiver to indicate that the SYNACK segment has already been succeed received by MTP Sender.

File Transmission

The file takes 8 arguments which are as following:

1. RECEIVER_HOST_IP: the IP address of the host machine on which the MTP Receiver is running.
2. RECEIVER_PORT: the port number on which MTP Receiver is expecting a packet from the sender.
3. file.txt: the name of the text file that has to be transferred from sender to receiver
4. MWS: the maximum window size
5. MSS: Maximum Segment Size which is the maximum amount of data (in bytes) carried in each MTP segment.

6. timeout: the value of timeout in milli-seconds
7. pdrop: the probability that a MTP packet which is ready to be transmitted will be dropped.
8. seed: The seed for random number generator.

When the connection establishing, the MTP Sender reads the all contents of the file to the sending buffer. Then the file transmission begins. The MTP Sender prepares the sending window and starts the timer. If the sending window is not full filled, the MTP Sender take MSS bytes data from the sending buffer and attaches the header to make a MTP segment. Then the MTP segment will go through the PLD module. The PLD module is a method to drop the segment. It is done by comparing the pdrop with a random number that been generated by a random number generator. If the random number is larger than the pdrop, the segment will be send and vise versa.This is in order to simulate the real network environment, because the LAN environment on which this program will run can rarely drop a packet. While the MTP Sender sends the MTP segment, it wil also receive the ACK segment from the MTP Receiver. Before describe the how MTP Sender deal with the ACK segment, it is better to understand how MTP Receiver send the ACK segment.

When file transmission starts, the MTP sender prepares a received data buffer, open a receiver data socket and set the arrived bytes to zero. When MTP segment arrives, the sequence number of the MTP segment will be checked. If the number equals to arrived bytes, there could be two situations: one possible situation is the MTP segment is in ordered. Then the arrived bytes will be added by the length of the data which stores in the data length field of the MTP header. The other possible situation is the MTP segment is retransmitted by the MTP Sender. For this situation, after add the length of data to the arrived bytes, the received data buffer should also be checked. If there are in ordered segment stores in the buffer, the data length need to be added too. If not equals, it means the MTP segment is out ordered. Then the arrived bytes will be reminded. For example, say the arrived bytes now equals to 50, but the MTP segment with sequence number 50 was dropped. The following arrived MTP segments are with sequence 100, 150 and 200(data length is 50 bytes). For the three MTP segments, the ACK number in the corresponding MTP ACK segment is set to 50, 50 and 50 respectively. Then the MTP segment with sequnce number 50 arrives, this time, the ACK number is set to 250. The MTP Receiver will immediately send a ACK segment back to MTP Sender with the ACK flag field set to 1 and ACK number field set to the value of arrived bytes. Both of the in ordered segment and out ordered segment will be stored in the buffer.

After understanding how MTP Receiver set the ACK value in the MTP ACK segment, the way how the MTP sender deal with the ACK value can be discuessed. If the ACK value not equals to the acked_bytes recording by MTP Sender, the acked_bytes will be set to ACK value and the sending window will move ACK value minus acked_bytes bytes. If the ACK value equals to the acked_bytes, a counter will start to count the number of the duplicated ACKs, when the number equals to 3, the MTP Sender retransmits the first segment in the sending window and restarts the timer. That is the mechanism of fast retransmission.

When timer timout, the MTP Sender retransmits the first segment in the sending window and restarts the timer.

Answers of Quesions

(a) Use the following parameter setting: pdrop = 0.1, MWS = 500 bytes, MSS = 50 bytes, seed = 300. Explain how you determine a suitable value for timeout. Note that you may need to experiment with different timeout values to answer this question. Justify your answer.

First of all, randomly choose a enough small timeout value, such as 10 milli-seconds(ms). Then run the program. Open the MTP_SENDER_LOG file to check whether the retransmission of the dropped segment is done by fast retransmission method or timeout method. If it is done by timeout method, it can be determined that the timeout period is too short. Then increase the time by doubling the timeout period. Do the same process again and again until the retransmissions is basiclly done by fast retransmission method. Say the timeout value now is 40ms. Then try reduce 40ms by 5ms each time,

run the program, take note of the last timeout value which leads that the retransmissions is basiclly

done by fast retransmission method. Say the timeout value is 30ms now. Finally run the program several times with timeout value 30ms to confirm. The suitable timeout value is 30ms when the program running under the LAN environment in the CSE lab.

(b)With the timeout value that you have selected, run an experiment with your MTP programs transferring the file test1.txt (available on the assignment webpage). Show the sequence of MTP packets that are observed at the receiver. Run an additional experiment with pdrop = 0.3, transferring the same file (test1.txt). Discuss the resulting packet sequences of both experiments indicating where dropping occurred.
For the first experiment, after a MTP segment with sequence number 1300 arrived, the expected following MTP segment is with sequence number 1350, but the actual arrived one is with sequence number 1400. This indicates the packet with sequence number 1350 drop. The packet with sequence number 1500 can be determined as a dropped packet for the same reason. The sequence number of all packets are in order except the two dropped packet.
For the second experiment, the packets with sequence number 100, 250, 350, 850 and 1150 were dropped, this can be determined by the fact that these packet arrived at MTP Receiver out of order.

(c) Let Tcurrent represent the timeout value that you have chosen in part (a). Set pdrop = 0.1, MWS = 500 bytes, MSS = 50 bytes, seed = 300 and run three experiments with the following different timeout values
i. Tcurrent
ii. 4*Tcurrent
iii. Tcurrent/4
and transfer the file test2.txt using MTP. Show a table that indicates how many MTP packets were transmitted (this should include retransmissions) in total and how long the overall transfer took. Discuss the results.

Timeout period	MTP packets #	overall transfer time
Tcurrent	40	97.04ms
4*Tcurrent	40	92.25ms
Tcurrent/4	42	100.10ms

When divide the Tcurrent by 4, the timeout period is too short for the MTP Sender to handle. The reason of the extra 2 MTP segment been transmistted in the last case is that: MTP Sender retransmitted dropped segments because of timeout, however, it also receives 3 duplicated acknowledgment MTP segments at that time, so MTP Sender retransmittes them again.

Appendix

Sequence numbers for testing data: pdrop = 0.1, MWS = 500bytes, MSS = 50 bytes, seed = 300
Received fragment with seqence number 0 at 2012-06-01 12:21:480004 from mtp sender.
Received fragment with seqence number 50 at 2012-06-01 12:21:480008 from mtp sender.
Received fragment with seqence number 100 at 2012-06-01 12:21:480012 from mtp sender.
Received fragment with seqence number 150 at 2012-06-01 12:21:480015 from mtp sender.
Received fragment with seqence number 200 at 2012-06-01 12:21:480019 from mtp sender.
Received fragment with seqence number 250 at 2012-06-01 12:21:480023 from mtp sender.
Received fragment with seqence number 300 at 2012-06-01 12:21:480027 from mtp sender.
Received fragment with seqence number 350 at 2012-06-01 12:21:480031 from mtp sender.
Received fragment with seqence number 400 at 2012-06-01 12:21:480037 from mtp sender.
Received fragment with seqence number 450 at 2012-06-01 12:21:480041 from mtp sender.
Received fragment with seqence number 500 at 2012-06-01 12:21:480045 from mtp sender.
Received fragment with seqence number 550 at 2012-06-01 12:21:480049 from mtp sender.
Received fragment with seqence number 600 at 2012-06-01 12:21:480052 from mtp sender.
Received fragment with seqence number 650 at 2012-06-01 12:21:480057 from mtp sender.
Received fragment with seqence number 700 at 2012-06-01 12:21:480060 from mtp sender.
Received fragment with seqence number 750 at 2012-06-01 12:21:480064 from mtp sender.
Received fragment with seqence number 800 at 2012-06-01 12:21:480069 from mtp sender.
Received fragment with seqence number 850 at 2012-06-01 12:21:480073 from mtp sender.
Received fragment with seqence number 900 at 2012-06-01 12:21:480077 from mtp sender.
Received fragment with seqence number 950 at 2012-06-01 12:21:480081 from mtp sender.
Received fragment with seqence number 1000 at 2012-06-01 12:21:480085 from mtp sender.
Received fragment with seqence number 1050 at 2012-06-01 12:21:480089 from mtp sender.
Received fragment with seqence number 1100 at 2012-06-01 12:21:480093 from mtp sender.
Received fragment with seqence number 1150 at 2012-06-01 12:21:480205 from mtp sender.
Received fragment with seqence number 1200 at 2012-06-01 12:21:480216 from mtp sender.
Received fragment with seqence number 1250 at 2012-06-01 12:21:480220 from mtp sender.
Received fragment with seqence number 1300 at 2012-06-01 12:21:480223 from mtp sender.
Received fragment with seqence number 1400 at 2012-06-01 12:21:480227 from mtp sender.
Received fragment with seqence number 1450 at 2012-06-01 12:21:480229 from mtp sender.
Received fragment with seqence number 1550 at 2012-06-01 12:21:480233 from mtp sender.
Received fragment with seqence number 1350 at 2012-06-01 12:21:480236 from mtp sender.
Received fragment with seqence number 1500 at 2012-06-01 12:21:480283 from mtp sender.

Sequence numbers for testing data: pdrop = 0.3, MWS = 500bytes, MSS = 50 bytes, seed = 300
Received fragment with seqence number 0 at 2012-06-01 12:22:000415 from mtp sender.
Received fragment with seqence number 50 at 2012-06-01 12:22:000419 from mtp sender.
Received fragment with seqence number 150 at 2012-06-01 12:22:000423 from mtp sender.
Received fragment with seqence number 200 at 2012-06-01 12:22:000426 from mtp sender.
Received fragment with seqence number 300 at 2012-06-01 12:22:000429 from mtp sender.
Received fragment with seqence number 400 at 2012-06-01 12:22:000434 from mtp sender.
Received fragment with seqence number 450 at 2012-06-01 12:22:000440 from mtp sender.
Received fragment with seqence number 500 at 2012-06-01 12:22:000443 from mtp sender.
Received fragment with seqence number 550 at 2012-06-01 12:22:000501 from mtp sender.
Received fragment with seqence number 100 at 2012-06-01 12:22:000511 from mtp sender.
Received fragment with seqence number 600 at 2012-06-01 12:22:000517 from mtp sender.

Received fragment with seqence number 650 at 2012-06-01 12:22:000520 from mtp sender.
Received fragment with seqence number 700 at 2012-06-01 12:22:000523 from mtp sender.
Received fragment with seqence number 250 at 2012-06-01 12:22:000526 from mtp sender.
Received fragment with seqence number 750 at 2012-06-01 12:22:000532 from mtp sender.
Received fragment with seqence number 350 at 2012-06-01 12:22:000573 from mtp sender.
Received fragment with seqence number 850 at 2012-06-01 12:22:000585 from mtp sender.
Received fragment with seqence number 900 at 2012-06-01 12:22:000587 from mtp sender.
Received fragment with seqence number 950 at 2012-06-01 12:22:000590 from mtp sender.
Received fragment with seqence number 1000 at 2012-06-01 12:22:000592 from mtp sender.
Received fragment with seqence number 1050 at 2012-06-01 12:22:000595 from mtp sender.
Received fragment with seqence number 1100 at 2012-06-01 12:22:000598 from mtp sender.
Received fragment with seqence number 1200 at 2012-06-01 12:22:000600 from mtp sender.
Received fragment with seqence number 1250 at 2012-06-01 12:22:000603 from mtp sender.
Received fragment with seqence number 800 at 2012-06-01 12:22:000677 from mtp sender.
Received fragment with seqence number 1300 at 2012-06-01 12:22:000688 from mtp sender.
Received fragment with seqence number 1350 at 2012-06-01 12:22:000691 from mtp sender.
Received fragment with seqence number 1400 at 2012-06-01 12:22:000694 from mtp sender.
Received fragment with seqence number 1450 at 2012-06-01 12:22:000697 from mtp sender.
Received fragment with seqence number 1500 at 2012-06-01 12:22:000700 from mtp sender.
Received fragment with seqence number 1550 at 2012-06-01 12:22:000702 from mtp sender.
Received fragment with seqence number 1150 at 2012-06-01 12:22:000709 from mtp sender.