Surya Avinash Avala, z5096886

8. Report

In addition you should submit a small report, report.pdf (no more than 5 pages), plus appendix section (appendix need not be counted in the 5 page limit). Your report must contain the following:

1. A brief discussion of how you have implemented the STP protocol. Provide a list of features that you have successfully implemented. In case you have not been able to get certain features of STP working, you should also mention that in your report.

List of Features implemented:

1. Three-way handshake between Sender and Receiver.
2. Four-segment connection termination
3. Timer for timeouts on sender
   Sender waits for the timeout amount of time before retransmitting that data
4. Simplified TCP sender and receiver as per the text
   Assuming unidirectional data flow
5. Fast retransmission
   Sender retransmits data for the next segment an duplicate ack is received for a segment.
6. Immediate ACKs by receiver, cumulative ACK
   Receiver immediately sends an ACK if an expected packet is received. And if multiple expected packets are received it only sends the ACK for the last sequence number.
7. Receiver Buffers packets arriving in wrong order and reads from the buffer appropriately.
8. Byte stream oriented socket with sequence/acknowledgement numbers
9. MSS
10. MWS
11. PLD
12. Logging
13. Source port is picked by the OS randomly

Difference between traditional TCP and my socket

1 . ACK and SYN numbers are little different that the TCP
   For data packets, each packet is treated as ONE byte irrespective of actual number of bytes transmitted

2 . fast retransmission is sent after one duplicate acknowledgement
   in tcp traditionally sent after 3 dup acks

Code is structured as packet.py and mysocket.py for packet and mysocket classes and well documented. The progress can also be seen from the bitbucket repo (git@bitbucket.org:saavala2/network.git)

sender.py and receiver.py are very primitive programs using the my socket class, just the way we used tcp/udp sockets in python.

2. A detailed diagram of your STP header and a quick explanation of all fields (similar to the diagrams that we have used in the lectures to understand TCP/UDP headers).

STP header is a python dictionary with field names (["sport","dport","seq_nb","ack_nb","ACK","SYN","FIN","DATA"]) as keys and their appropriate values as values.

1. sport: Port number of the source
2. dport: Port number of the destination
3. seq_nb: Sequence number of the current packet (similar to tcp but with a minor difference as mentioned in the question)
4. ack_nb: Last acknowledgment number sent
5. ACK: Ack bit
6. SYN: Sync bit
7. FIN: Fin bit
8. DATA: Data bit indicating the presence of data/payload in the packet.

Example header would like follows:

Field	Value
sport	1234
dport	4321
seq_nb	90
ack_nb	500
ACK	0
SYN	0
FIN	0
DATA	1

3. For the standard version of the assignment, answer the following questions:

(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.

From my experiments a timeout value of 1sec (1000msec) is suitable.

No. of duplicate packets dramatically increase (in a order of 100 magnitude) to 4925 from 68 even if I go to 0.999sec(999msec).

And no matter how high I choose the timeout (even with 100000sec) no. of duplicate packets remain the same as in the case of timeout being 1sec.

So the most suitable timeout for this scenario is 1sec.

(b) 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:
12

i. Tcurrent
ii. 4 × Tcurrent
iii. Tcurrent/4

T value	**No. Of Packets transmitted(inc dup)**	**Transfer time (msec)**
Tcurrent (1000ms)	150	28.327
4*Tcurrent (4000ms)	150	30.919
Tcurrent/4 (250ms)	8882	1121.84

Tcurrent/4 timeout is too little that the timer on Sender goes off very soon and the sender will excessively retransmit the files.
Timeout values passes it threshold(time required for pack to reach receiver + time required for the ack to reach sender) at Tcurrent. So Tcurrent is optimal.