z5255891
Barry Zheng

COMP3331 Assignment Report

1. The PTP was implemented in python 3.7 in two files, receiver.py and sender.py. Limited testing in python 2.7 and python 3.7 showed that either could be used to run this program. For sender, I had two helper functions, one to calculate the time and one to read the file that needed to be sent. I also had two functions, one to start the handshake and the other to begin the four-segment connection termination. It sends a SYN packet to initiate the handshake and then starts to send over a copy of the data from the given file. After all the packets of data are sent, it begins the four-segment connection termination. For this part, the sender sends a FIN packet first, then waits for a FINACK packet and then finally sends an ACK. Then it begins to terminate the connection. There was a Sender.txt file which included the logs for the sender.py file during an experiment, and at the bottom had the necessary statistics.

   For receiver, I had one helper function which was to calculate time, and the same as the one in sender. I also had two functions, one to respond to the handshake and another which began the termination of the connection by responding to the four-segment connection termination. The receiver is listening for packets and will perform different functionalities depending what they are. After receiving all the data, it begins to terminate the connection. There was also a Receiver.txt file which included the logs for the receiver.py file during an experiment, and at the bottom had the necessary statistics.

   To my knowledge, I implemented all the necessary features from the specification such as retransmission, single timeout, the PL module and buffer storage. However, I do believe my time in my logs may possibly be wrong, however I only had limited time to do testing due to juggling 3 other assignments. For this I apologise.

2.

SYN
5048
ABCDEFGHIJK

   For the top row, this is the type of packet. In my PTP this is represented as a string and a Boolean state to check for the packet type. For the middle row, it represents the sequence number and can be seen in the logs as the middle column. This lets the receiver or sender know what packet is received or sent. The last row is the payload which is basically the data being transferred.

3. A) for the first experiment with pdrop of 0.1, I used a timeout time of 30, as the default timeout for UDP is 30 milliseconds. This resulted in these results in my receiver and sender logs:

```
################################################
Amount of Data Received: 32768 bytes
Number of Data Segments Received: 63
Number of duplicate segments received: 616
################################################
```

```
################################################
Amount of data transferred: 103272 bytes
Number of data segments sent: 2068
Number of packets dropped: 250
Number of retransmitted segments: 3439
Number of duplicate acknowledgements: 1234
################################################
```

For pdrop of 0.3 I got these results:

```
################################################
Amount of Data Received: 32768 bytes
Number of Data Segments Received: 27
Number of duplicate segments received: 639
################################################
```

```
################################################
Amount of data transferred: 134854 bytes
Number of data segments sent: 2699
Number of packets dropped: 1198
Number of retransmitted segments: 5474
Number of duplicate acknowledgements: 2180
################################################
```

As we can see from these results above, there seems to be a positive correlation between the probability to drop the datagram with the number of duplicate segments received. This also significantly slowed down the running of this program.