COMP3331 Assignment Report

Jianjun Jay Chen(z5261536)

1. A brief discussion of how you have implemented the PTP protocol.

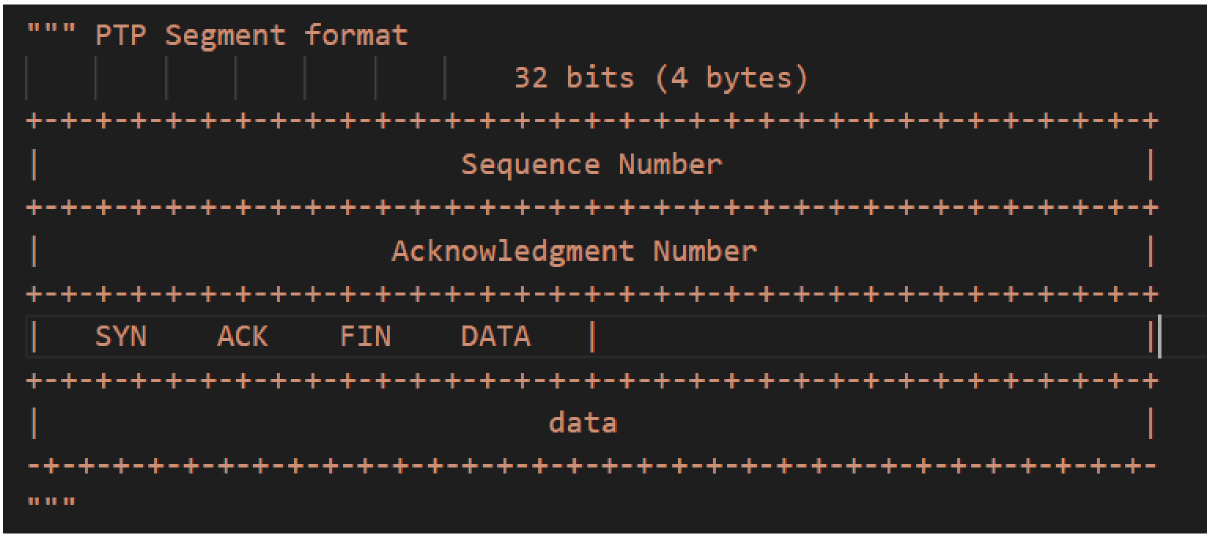
For the PTP protocol, I've completed the three-way handshake for the connection establishment by let the sender send a SYN type segment, and when the receiver received a SYN type segment, it will return a SYNACK type segment back to the sender, and then sender return an ack segment to the SYNACK segment, connection established.

Similarly, I've completed the connection termination, just like connection establishment. After the sender received all the data ack (all the data in the file has been received by the receiver), sender start sending the FIN segment, and when the receiver received the FIN type segment, it will also return a FINACK segment back to the sender, and then sender return an ack to the FINACK segment, connection termination complete.

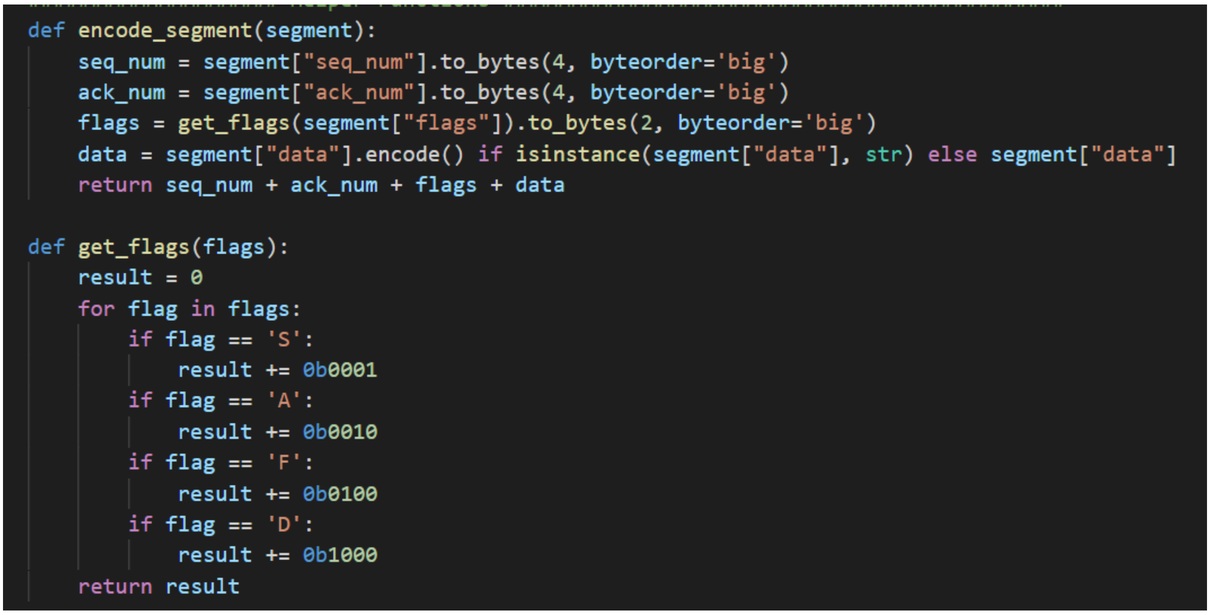
And for the sender reliable data transfer part. I have a read_file function to get all the data from FileToSend.txt and split them into MSS size. And then I could encapsulate these data with a header field by using encode_segment function and send these segments to the receiver. By using the Random and pdrop value, I developed the PL module to emulate packes loss on the internet. I've also used the socket.settimeout to maintain a single timer for timeout operation, generate_window and sliding_window function for the slide window features based on the MWS, dup_ack variable for counting duplicate ack received from the receiver. If dup_ack equals to 3, the fast retransmission would get triggered.

For the receiver, unlike sender which is like a Go-back-end protocol, PTP receiver has selective repeated mechanism. My receiver has a buffer for storing all received data segments(both in order and out-of-order) By using the buffer and get_max_ack function, we can buffer the correctly received but out-of-order segments and accomplish the error-recovery SN like mechanism.

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



```
""" PTP Segment format
|   |   |   |   |   |   |     32 bits (4 bytes)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Sequence Number                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   SYN    ACK    FIN    DATA   |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
"""
```



```python
def encode_segment(segment):
    seq_num = segment["seq_num"].to_bytes(4, byteorder='big')
    ack_num = segment["ack_num"].to_bytes(4, byteorder='big')
    flags = get_flags(segment["flags"]).to_bytes(2, byteorder='big')
    data = segment["data"].encode() if isinstance(segment["data"], str) else segment["data"]
    return seq_num + ack_num + flags + data

def get_flags(flags):
    result = 0
    for flag in flags:
        if flag == 'S':
            result += 0b0001
        if flag == 'A':
            result += 0b0010
        if flag == 'F':
            result += 0b0100
        if flag == 'D':
            result += 0b1000
    return result
```

So just like the TCP header, my PTP header has 32bits field for the sequence number of the segment, 32 bits filed for acknowledgement number, and 16bits for the flags (segment type) includes SYN, SYNACK, FIN, FINACK, DATA, ACK.

3. Answer the following questions:

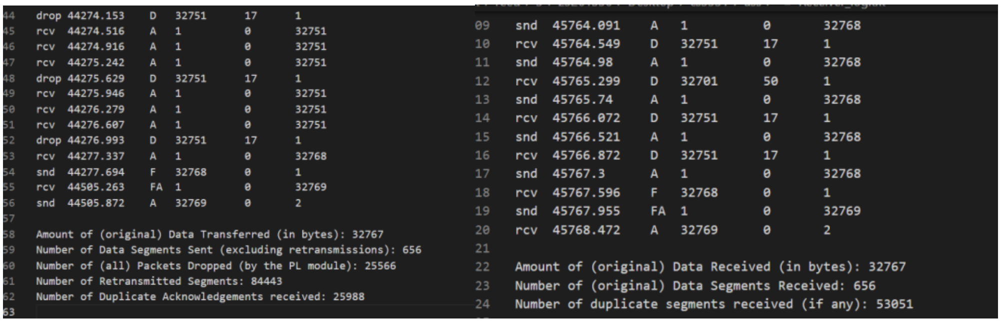
(a) Use the following parameter setting: pdrop = 0.1, MWS = 500 bytes, MSS = 50 bytes, seed = 300. Explain how you determine a suitable value for timeout. Justify your answer. With the timeout value that you have selected, run an experiment with your PTP programs transferring the file test1.txt. Show the sequence of PTP packets that are observed at the receiver. It is sufficient to just indicate the sequence numbers of the PTP packets that have arrived. You do not have to indicate the payload contained in the PTP packet. Run an additional experiment with pdrop = 0.3, transferring the same file (test1.txt). In your report, discuss the resulting packet sequences of both experiments indicating where dropping occurred. Also, in the appendix section show the packet sequences of all the experiments.

*** run by using python 3.7**

Command: python3 sender.py 127.0.0.1 5000 32KB.txt 500 50 1000 0.1 300

python3 receiver.py 5000 FileReceived.txt

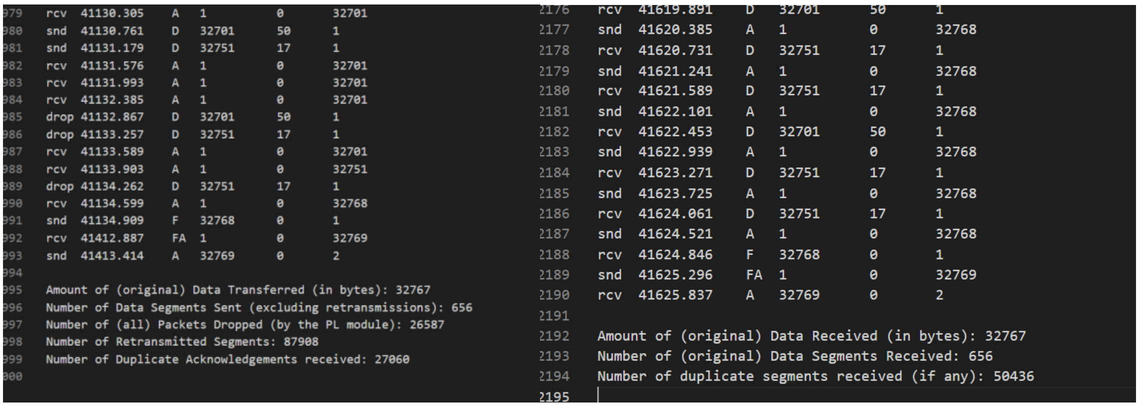
When I select timeout value 2000ms(2 seconds), the total transmission time is 45768.472ms, and the number of retransmitted segments is 84443.



```
drop 44274.153    D   32751      17      1
rcv  44274.516    A   1          0       32751
rcv  44274.916    A   1          0       32751
rcv  44275.242    A   1          0       32751
drop 44275.629    D   32751      17      1
rcv  44275.946    A   1          0       32751
rcv  44276.279    A   1          0       32751
rcv  44276.607    A   1          0       32751
drop 44276.993    D   32751      17      1
rcv  44277.337    A   1          0       32768
snd  44277.694    F   32768      0       1
rcv  44505.263    FA  1          0       32769
snd  44505.872    A   32769      0       2

Amount of (original) Data Transferred (in bytes): 32767
Number of Data Segments Sent (excluding retransmissions): 656
Number of (all) Packets Dropped (by the PL module): 25566
Number of Retransmitted Segments: 84443
Number of Duplicate Acknowledgements received: 25988
```

```
snd  45764.091    A   1          0       32768
rcv  45764.549    D   32751      17      1
snd  45764.98     A   1          0       32768
rcv  45765.299    D   32701      50      1
snd  45765.74     A   1          0       32768
rcv  45766.072    D   32751      17      1
snd  45766.521    A   1          0       32768
rcv  45766.872    D   32751      17      1
snd  45767.3      A   1          0       32768
rcv  45767.596    F   32768      0       1
snd  45767.955    FA  1          0       32769
rcv  45768.472    A   32769      0       2

Amount of (original) Data Received (in bytes): 32767
Number of (original) Data Segments Received: 656
Number of duplicate segments received (if any): 53051
```

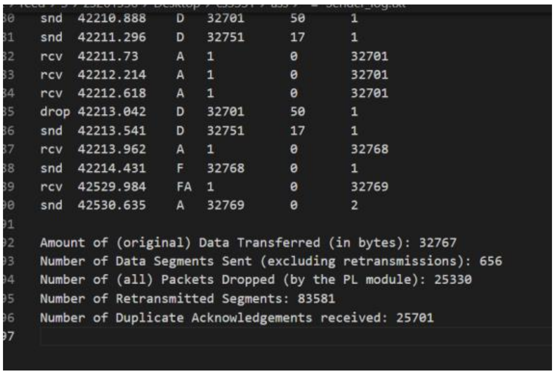
When I set timeout value 1000ms(1 seconds), the total transmission time is 41531.866ms, and the number of retransmitted segments is 87908.



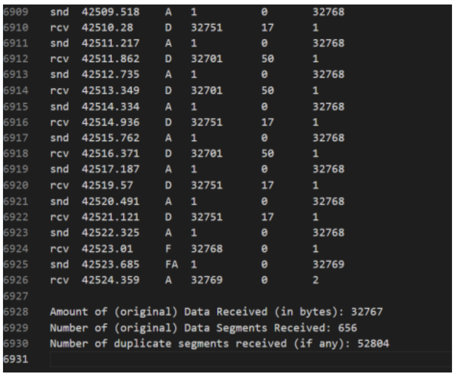
```
rcv  41130.305    A   1          0       32701
snd  41130.761    D   32701      50      1
snd  41131.179    D   32751      17      1
rcv  41131.576    A   1          0       32701
rcv  41131.993    A   1          0       32701
rcv  41132.385    A   1          0       32701
drop 41132.867    D   32701      50      1
drop 41133.257    D   32751      17      1
rcv  41133.589    A   1          0       32701
rcv  41133.903    A   1          0       32751
drop 41134.262    D   32751      17      1
rcv  41134.599    A   1          0       32768
snd  41134.909    F   32768      0       1
rcv  41412.887    FA  1          0       32769
snd  41413.414    A   32769      0       2

Amount of (original) Data Transferred (in bytes): 32767
Number of Data Segments Sent (excluding retransmissions): 656
Number of (all) Packets Dropped (by the PL module): 26587
Number of Retransmitted Segments: 87908
Number of Duplicate Acknowledgements received: 27060
```

```
rcv  41619.891    D   32701      50      1
snd  41620.385    A   1          0       32768
rcv  41620.731    D   32751      17      1
snd  41621.241    A   1          0       32768
rcv  41621.589    D   32751      17      1
snd  41622.101    A   1          0       32768
rcv  41622.453    D   32701      50      1
snd  41622.939    A   1          0       32768
rcv  41623.271    D   32751      17      1
snd  41623.725    A   1          0       32768
rcv  41624.061    D   32751      17      1
snd  41624.521    A   1          0       32768
rcv  41624.846    F   32768      0       1
snd  41625.296    FA  1          0       32769
rcv  41625.837    A   32769      0       2

Amount of (original) Data Received (in bytes): 32767
Number of (original) Data Segments Received: 656
Number of duplicate segments received (if any): 50436
```

When I set timeout value 300ms(0.3 seconds), the total transmission time is 42530.866ms, and the number of retransmitted segments is 83581.



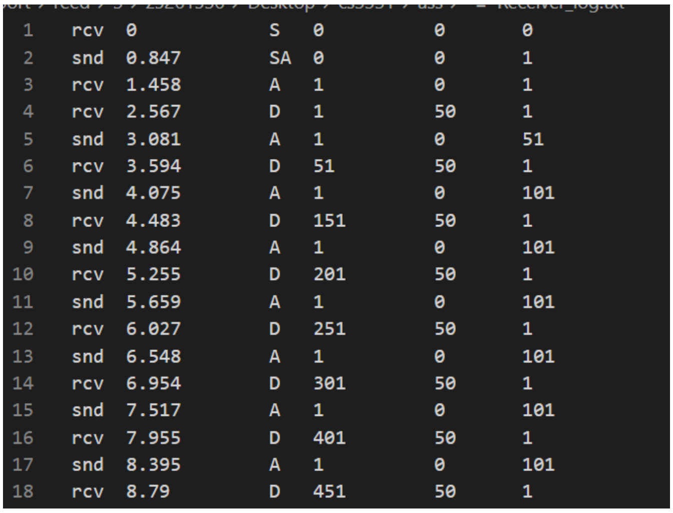
```
snd  42210.888    D   32701      50       1
snd  42211.296    D   32751      17       1
rcv  42211.73     A   1          0        32701
rcv  42212.214    A   1          0        32701
rcv  42212.618    A   1          0        32701
drop 42213.042    D   32701      50       1
snd  42213.541    D   32751      17       1
rcv  42213.962    A   1          0        32768
snd  42214.431    F   32768      0        1
rcv  42529.984    FA  1          0        32769
snd  42530.635    A   32769      0        2

Amount of (original) Data Transferred (in bytes): 32767
Number of Data Segments Sent (excluding retransmissions): 656
Number of (all) Packets Dropped (by the PL module): 25330
Number of Retransmitted Segments: 83581
Number of Duplicate Acknowledgements received: 25701
```



```
snd  42509.518    A   1          0        32768
rcv  42510.28     D   32751      17       1
snd  42511.217    A   1          0        32768
rcv  42511.862    D   32701      50       1
snd  42512.735    A   1          0        32768
rcv  42513.349    D   32701      50       1
snd  42514.334    A   1          0        32768
rcv  42514.936    D   32751      17       1
snd  42515.762    A   1          0        32768
rcv  42516.371    D   32701      50       1
snd  42517.187    A   1          0        32768
rcv  42519.57     D   32751      17       1
snd  42520.491    A   1          0        32768
rcv  42521.121    D   32751      17       1
snd  42522.325    A   1          0        32768
rcv  42523.01     F   32768      0        1
snd  42523.685    FA  1          0        32769
rcv  42524.359    A   32769      0        2

Amount of (original) Data Received (in bytes): 32767
Number of (original) Data Segments Received: 656
Number of duplicate segments received (if any): 52804
```

From the different result(transmission time, retransmitted segment) by using different timeout value, I realized if timeout value set too small, even the receiver could receive the data segments, the timer will resend the segments from the sender and cause more unnecessary duplicated segments being sent(data retransmission) to the receiver side. But if timeout value set too large, the sender will spend more time waiting on the ack segment from receiver even the sender might drop the packet already, in this case, sender will waste more time on waiting for the ack packet.

For example, from the receiver_log.txt, we could see after the receiver receive packet with sequence number 51 with data bytes 50, the receiver started sending ack 101 to the sender, but the sender kept dropping the sequence number 51 packets, so the receiver also kept sending ack101 to the sender. That’s one place where dropping occurred in the sender.

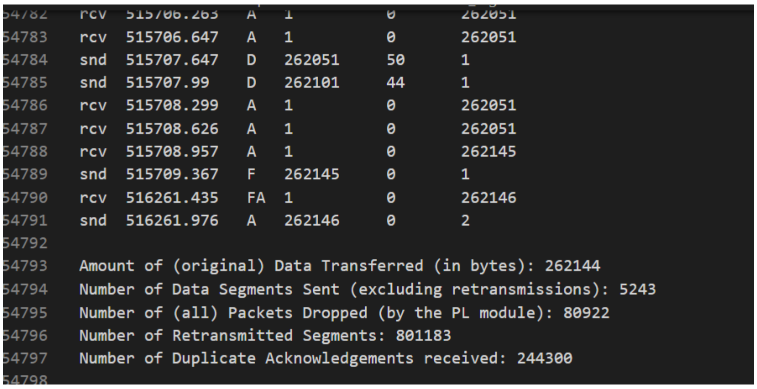


```
rcv  0            S   0          0        0
snd  0.847        SA  0          0        1
rcv  1.458        A   1          0        1
rcv  2.567        D   1          50       1
snd  3.081        A   1          0        51
rcv  3.594        D   51         50       1
snd  4.075        A   1          0        101
rcv  4.483        D   151        50       1
snd  4.864        A   1          0        101
rcv  5.255        D   201        50       1
snd  5.659        A   1          0        101
rcv  6.027        D   251        50       1
snd  6.548        A   1          0        101
rcv  6.954        D   301        50       1
snd  7.517        A   1          0        101
rcv  7.955        D   401        50       1
snd  8.395        A   1          0        101
rcv  8.79         D   451        50       1
```

(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:**

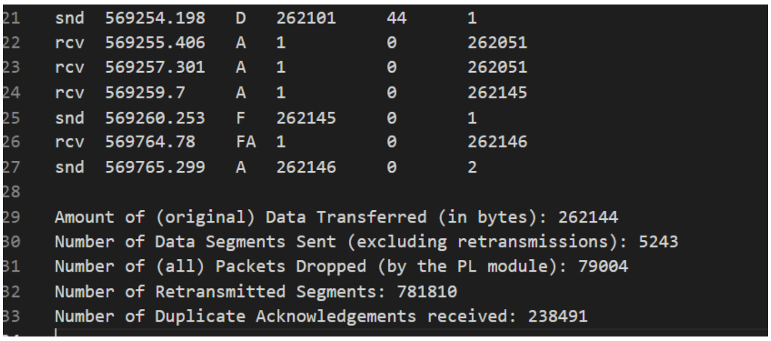
I’ve decided to choose 1000ms (1 second) as my timeout value

i. Tcurrent = 1S



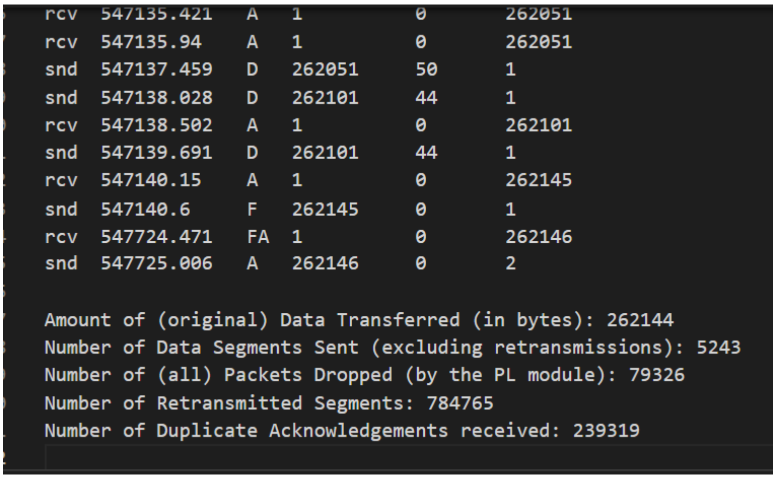
```
rcv  515706.263   A   1          0       262051
rcv  515706.647   A   1          0       262051
snd  515707.647   D   262051     50      1
snd  515707.99    D   262101     44      1
rcv  515708.299   A   1          0       262051
rcv  515708.626   A   1          0       262051
rcv  515708.957   A   1          0       262145
snd  515709.367   F   262145     0       1
rcv  516261.435   FA  1          0       262146
snd  516261.976   A   262146     0       2

Amount of (original) Data Transferred (in bytes): 262144
Number of Data Segments Sent (excluding retransmissions): 5243
Number of (all) Packets Dropped (by the PL module): 80922
Number of Retransmitted Segments: 801183
Number of Duplicate Acknowledgements received: 244300
```

ii. 4 × Tcurrent = 4S



```
snd  569254.198   D   262101     44      1
rcv  569255.406   A   1          0       262051
rcv  569257.301   A   1          0       262051
rcv  569259.7     A   1          0       262145
snd  569260.253   F   262145     0       1
rcv  569764.78    FA  1          0       262146
snd  569765.299   A   262146     0       2

Amount of (original) Data Transferred (in bytes): 262144
Number of Data Segments Sent (excluding retransmissions): 5243
Number of (all) Packets Dropped (by the PL module): 79004
Number of Retransmitted Segments: 781810
Number of Duplicate Acknowledgements received: 238491
```

iii. Tcurrent/4 = 0.25S



```
rcv  547135.421   A   1          0       262051
rcv  547135.94    A   1          0       262051
snd  547137.459   D   262051     50      1
snd  547138.028   D   262101     44      1
rcv  547138.502   A   1          0       262101
snd  547139.691   D   262101     44      1
rcv  547140.15    A   1          0       262145
snd  547140.6     F   262145     0       1
rcv  547724.471   FA  1          0       262146
snd  547725.006   A   262146     0       2

Amount of (original) Data Transferred (in bytes): 262144
Number of Data Segments Sent (excluding retransmissions): 5243
Number of (all) Packets Dropped (by the PL module): 79326
Number of Retransmitted Segments: 784765
Number of Duplicate Acknowledgements received: 239319
```

From the result, we can tell if the Tcurrent is large, it takes longer time for transmission. And if the Tcurrent is small, it has more number of retransmitted packets.