Question 1:

I achieved the standard version of the assignment including the following features:
1) three-way handshake
2) the four-segment connection termination
3) single-timer mainted by sender.py
4) selective repeat in section 3.5.4 of the text including simplified TCP sender and fast retransmit
5) receiver.py containing the features mentioned in Section 3.5.4 of the text
6) sequence and acknowledgement number in header
7) Maximum segment size
8) Maximum Window size
9) usage of only UDP socket
10) PLD module
11) constant timeout in sender.py

sender.py:
My first step is a funtion called SYN_state() controling the sending and receiving in three-way handshake process.
The second step is two thread runing simultaneously. One is receive_thread() in which there is a loop whenever receiving an in-order ack, add 1 to the sendbase of the window. Another is send_thread() in which there is a loop cheking rules of Selective Repeat and fast retransmition and also a single simer to decide whether to send a corespongding segment or not.
The last step is a funtion FIN_state() controling the four-segment connection termination. It starts when receiving the last ACK from the second step.

receiver.py:
My first step is a funtion called SYN_state() controling the sending and receiving in three-way handshake process.
The second is a funtion called TRANS_FIN_state() in which there is a loop to reply the ACK directly after the last in-order segment and a buffer to contain not-in-order segments. And when it receives a FIN segment, the funtion goes to the FIN state to deal with the four-segment cnnection termination.

Question 2:

My segement format is showed below. The header is a dict concluding four kinds of flags and sequence and acknowledgement number. Payload part is a string containing the data. And segment to be transmit is a list with the first element being header and second element being payload. Additionally, I use pickle module to encode and decode.

```
init_header = {'SP': -1 ,'DP': port,'SYN': False, 'ACK': False, 'FIN': False, 'DATA': False, 'seq': 0, 'ack': 0}
init_payload = ''
init_segment = [init_header, init_payload]
```

Source Port		Destination Port	
SYN_FLAG	ACK_FLAG	FIN_FLAG	DATA_FLAG
Sequence number			
Acknowledgement number			
data			

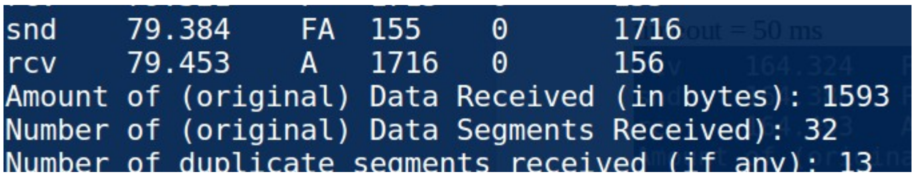
STP segment

Question 3:

(a)

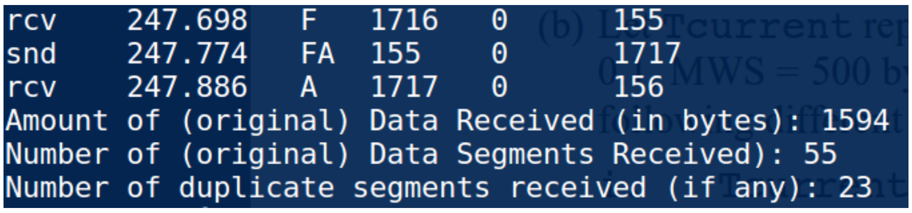
Experiments are showed below.

Timeout = 5 ms



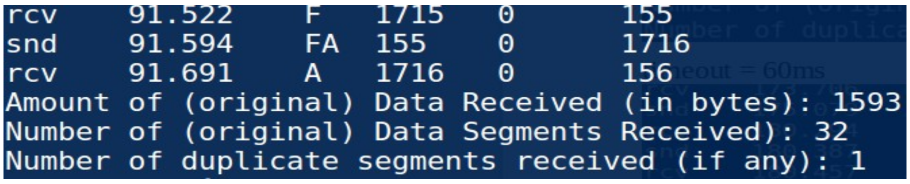
```
snd     79.384     FA  155     0        1716
rcv     79.453     A   1716    0        156
Amount of (original) Data Received (in bytes): 1593
Number of (original) Data Segments Received): 32
Number of duplicate segments received (if any): 13
```

timeout = 10 ms



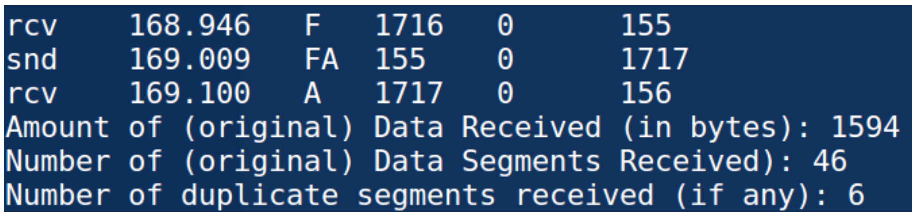
```
rcv     247.698    F   1716    0        155
snd     247.774    FA  155     0        1717
rcv     247.886    A   1717    0        156
Amount of (original) Data Received (in bytes): 1594
Number of (original) Data Segments Received): 55
Number of duplicate segments received (if any): 23
```

timeout = 20ms



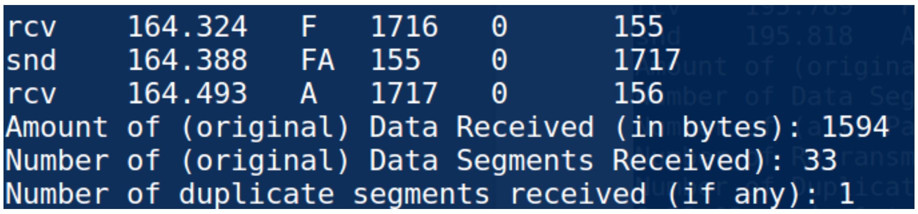
```
rcv     91.522     F   1715    0        155
snd     91.594     FA  155     0        1716
rcv     91.691     A   1716    0        156
Amount of (original) Data Received (in bytes): 1593
Number of (original) Data Segments Received): 32
Number of duplicate segments received (if any): 1
```

timeout = 40ms



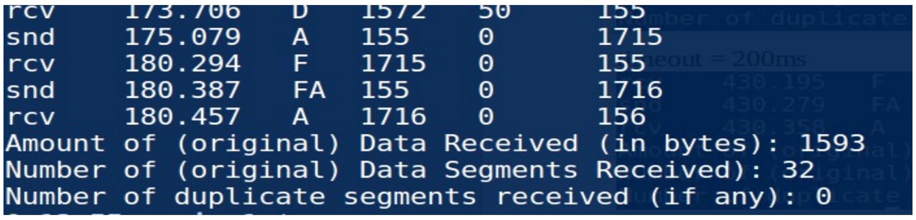
```
rcv     168.946    F   1716    0        155
snd     169.009    FA  155     0        1717
rcv     169.100    A   1717    0        156
Amount of (original) Data Received (in bytes): 1594
Number of (original) Data Segments Received): 46
Number of duplicate segments received (if any): 6
```

timeout = 50 ms



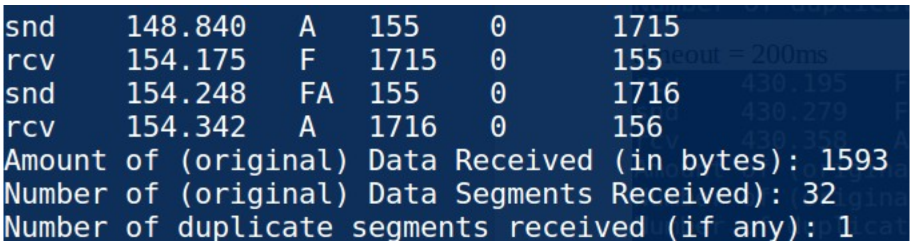
```
rcv     164.324    F   1716    0        155
snd     164.388    FA  155     0        1717
rcv     164.493    A   1717    0        156
Amount of (original) Data Received (in bytes): 1594
Number of (original) Data Segments Received): 33
Number of duplicate segments received (if any): 1
```

timeout = 60ms



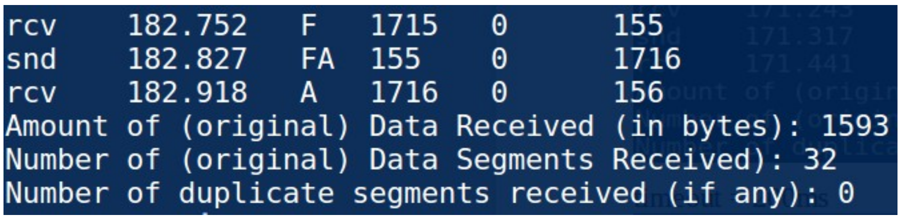
```
rcv     173.706    D   1572    50       155
snd     175.079    A   155     0        1715
rcv     180.294    F   1715    0        155
snd     180.387    FA  155     0        1716
rcv     180.457    A   1716    0        156
Amount of (original) Data Received (in bytes): 1593
Number of (original) Data Segments Received): 32
Number of duplicate segments received (if any): 0
```

timeout = 70ms



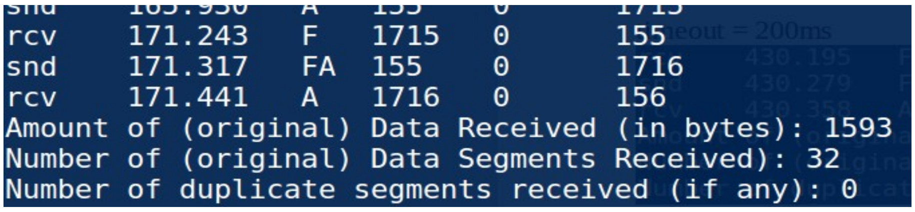
```
snd     148.840    A   155     0        1715
rcv     154.175    F   1715    0        155
snd     154.248    FA  155     0        1716
rcv     154.342    A   1716    0        156
Amount of (original) Data Received (in bytes): 1593
Number of (original) Data Segments Received): 32
Number of duplicate segments received (if any): 1
```

timeout = 80ms



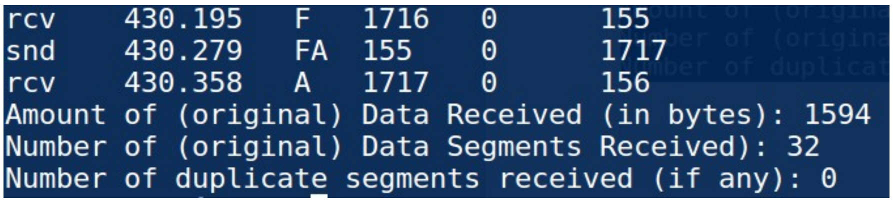
```
rcv     182.752    F    1715    0       155
snd     182.827    FA   155     0       1716
rcv     182.918    A    1716    0       156
Amount of (original) Data Received (in bytes): 1593
Number of (original) Data Segments Received): 32
Number of duplicate segments received (if any): 0
```

timeout = 100ms



```
rcv     171.243    F    1715    0       155
snd     171.317    FA   155     0       1716
rcv     171.441    A    1716    0       156
Amount of (original) Data Received (in bytes): 1593
Number of (original) Data Segments Received): 32
Number of duplicate segments received (if any): 0
```

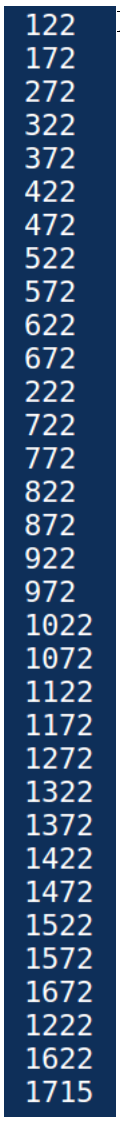
timeout = 200ms



```
rcv     430.195    F    1716    0       155
snd     430.279    FA   155     0       1717
rcv     430.358    A    1717    0       156
Amount of (original) Data Received (in bytes): 1594
Number of (original) Data Segments Received): 32
Number of duplicate segments received (if any): 0
```

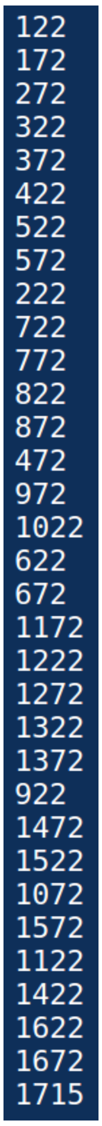
From experiments, it took one of the least times to transfer the same data when timeout = 20ms. And number of duplicate segments is acceptble Thus, 20ms is a suitable timeout value.

Dprob = 0.1:



```
122
172
272
322
372
422
472
522
572
622
672
222
722
772
822
872
922
972
1022
1072
1122
1172
1272
1322
1372
1422
1472
1522
1572
1672
1222
1622
1715
```

Dprob = 0.3:



```
122
172
272
322
372
422
522
572
222
722
772
822
872
472
972
1022
622
672
1172
1222
1272
1322
1372
922
1472
1522
1072
1572
1122
1422
1622
1672
1715
```

In dprop of 0.1, drop occourred in 172-272, 1172-1272, 1572-1672.
In dprop of 0.3, drop occourred in 172-272, 422-522, 872-972, 1022-1172, 1372-1472.

(b)

	20ms	80ms	5ms
Number of transmitted packets	41	40	48
Time overall tranfer takes(in ms)	80.832	142.596	49.239

Number of transmitted packets increases only when there are both a time-out packet and a fast retransmit packet occuring simultaneously, the same as dupilicate segments in receiver end. When the timeout is short enough, it is easy to trigger the timer resulting much more duplicate segments which will take more capicity in the link and vice versa. But when timeout is too long, it takes more time to trigger the timer as fast retransmition is not satisfied, which results more total transfer time.