<2019 Computer Network Homework>

Motivation

We divide the homework into two parts. First, you should understand the mechanism of TCP in detail including data transmission, flow control, delayed ACKs, and congestion control etc. Second, you have to implement TCP in application layer and call UDP to transmit TCP packets.

Rules

1. Run your program on Ubuntu 18.04 platform.

2. Do not copy homework from your classmates or senior, etc. If TAs find the situation, any participants will get a grade of ZERO.

3. You have to deeply understand what your program do because TAs will ask you the concept of your code.

4. If you have any question, you can send email or come to F-5008(High Speed Network Lab) to ask TAs but debugging.

5. You have to create Makefile to compile your program, and ensure your program can be compiled correctly.

6. You also need to submit a PDF that contains the picture of your program run's result in every step.

7. In each step, you can write a new program, respectively (but the program has to including the function of previous step).

8. The format of filename you upload should be "StudentID_Name.zip".

9.輸出格式僅供參考，實際輸出結果請依題目需求呈現。

Ex: B063040000_王小明.zip

Deadline

You should upload your homework to the Cyber University before 2019/06/20 23:59. If you do not submit your assignment on time, you will get a grade of ZERO.

Demo

The following figure shows the time you can come for demo.

Demo deadline: 2019/06/21 17:00

	Mon.	Tue.	Wed.	Thu.	Fri.
10:00 – 12:00	✓	✓	✓		✓
14:00 – 17:00		✓	✓		✓

Description

You have to obey the following schema:

The TCP segment structure

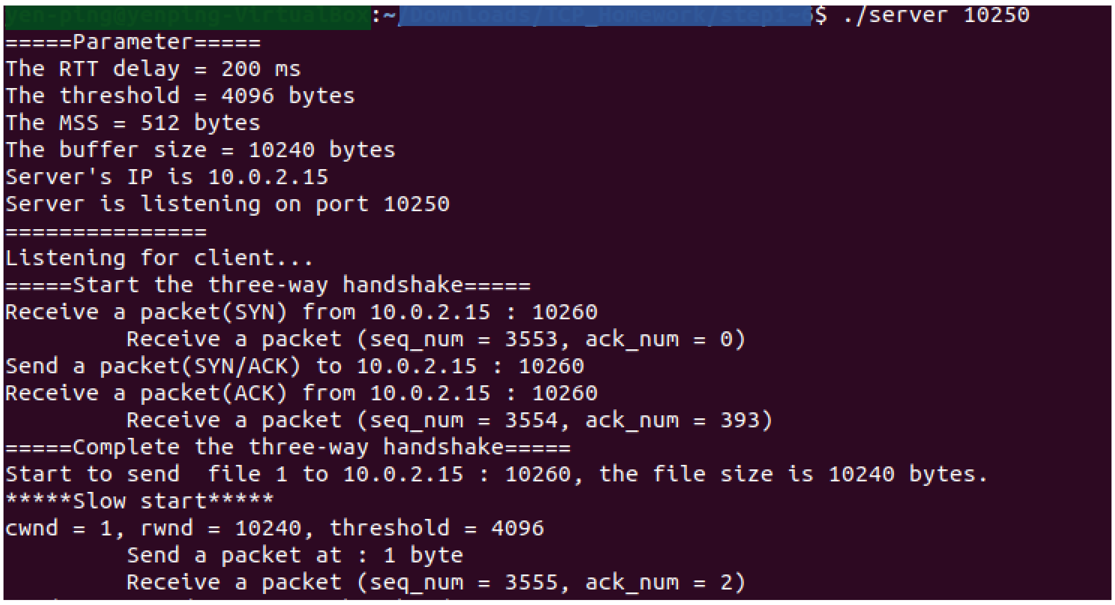
The initial sequence number should be set randomly (1~10000).

Step 1:

1. Set the parameters including RTT (150 ms), MSS (1024 bytes), threshold (65535 bytes) and the receiver’s buffer size (32KB), etc.

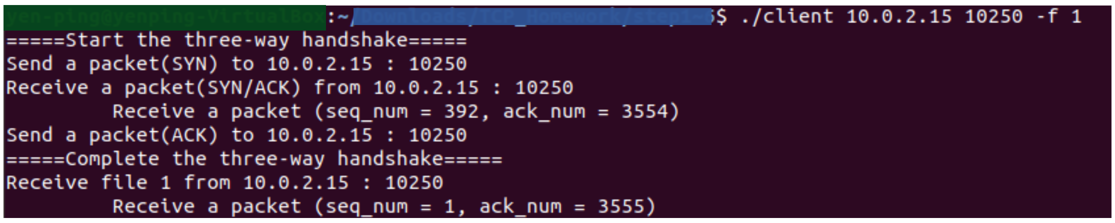
2. You have to transmit the video files in this step. A client could request single or multiple files in one command. The server should send the data to multiple clients in the same time. (You can use fork or thread.)
3. If you cannot send the video file, you can send a data created by yourself (ex. an 10240 bytes char array). (But you won’t get all score in the part if you do it in this way.)
4. You also have to implement the data transmission (You need to ensure the data that can transmit from server to client, and ACK packet transmit from client to server).
5. You have to print out which client the server is sending to and which file is sent in this step.

Server:



```
yen-ping@yenping-VirtualBox:~/Downloads/TCP_Homework/step1-6$ ./server 10250
=====Parameter=====
The RTT delay = 200 ms
The threshold = 4096 bytes
The MSS = 512 bytes
The buffer size = 10240 bytes
Server's IP is 10.0.2.15
Server is listening on port 10250
===============
Listening for client...
=====Start the three-way handshake=====
Receive a packet(SYN) from 10.0.2.15 : 10260
         Receive a packet (seq_num = 3553, ack_num = 0)
Send a packet(SYN/ACK) to 10.0.2.15 : 10260
Receive a packet(ACK) from 10.0.2.15 : 10260
         Receive a packet (seq_num = 3554, ack_num = 393)
=====Complete the three-way handshake=====
Start to send  file 1 to 10.0.2.15 : 10260, the file size is 10240 bytes.
*****Slow start*****
cwnd = 1, rwnd = 10240, threshold = 4096
         Send a packet at : 1 byte
         Receive a packet (seq_num = 3555, ack_num = 2)
```

Client:



```
[redacted]:~[redacted]$ ./client 10.0.2.15 10250 -f 1
=====Start the three-way handshake=====
Send a packet(SYN) to 10.0.2.15 : 10250
Receive a packet(SYN/ACK) from 10.0.2.15 : 10250
        Receive a packet (seq_num = 392, ack_num = 3554)
Send a packet(ACK) to 10.0.2.15 : 10250
=====Complete the three-way handshake=====
Receive file 1 from 10.0.2.15 : 10250
        Receive a packet (seq_num = 1, ack_num = 3555)
```

Step 2:

1. Including the previous step’s function.
2. You should randomly generate some packet loss under the rate of 0.01% and print the ACK number of loss packet.

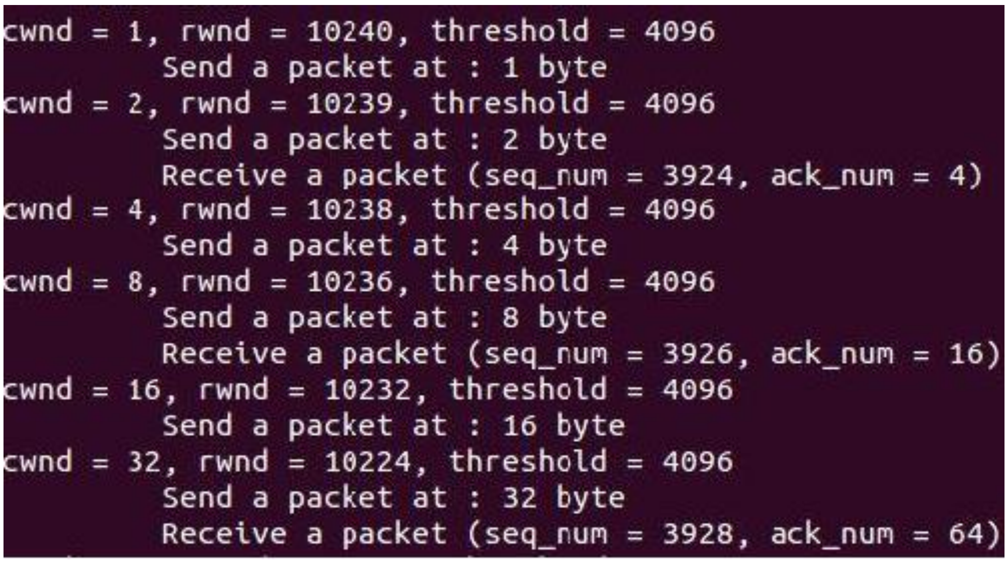
Step 3:

1. Including the previous step’s function.

2. Implement the delayed ACKs, you can wait up to 500ms for next packet, or delay for two packets, then send an ACK packet to server

3. You don’t have to print out which client the server is sending. (Or you can let only one client to connect to server.)

Server:

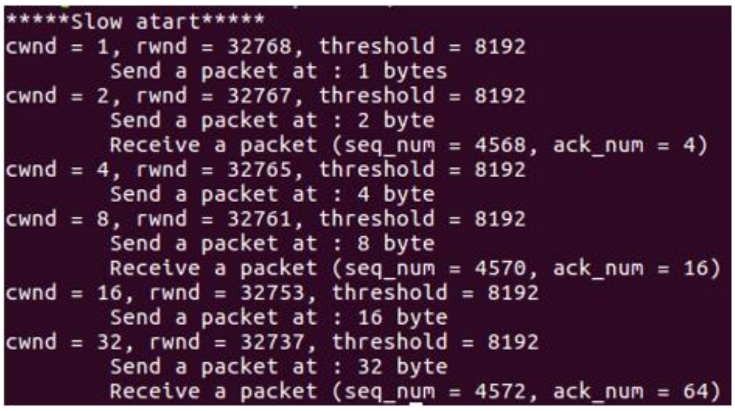


```
cwnd = 1, rwnd = 10240, threshold = 4096
         Send a packet at : 1 byte
cwnd = 2, rwnd = 10239, threshold = 4096
         Send a packet at : 2 byte
         Receive a packet (seq_num = 3924, ack_num = 4)
cwnd = 4, rwnd = 10238, threshold = 4096
         Send a packet at : 4 byte
cwnd = 8, rwnd = 10236, threshold = 4096
         Send a packet at : 8 byte
         Receive a packet (seq_num = 3926, ack_num = 16)
cwnd = 16, rwnd = 10232, threshold = 4096
         Send a packet at : 16 byte
cwnd = 32, rwnd = 10224, threshold = 4096
         Send a packet at : 32 byte
         Receive a packet (seq_num = 3928, ack_num = 64)
```

Step 4:

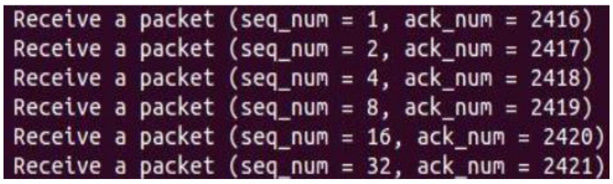
1. Including the previous step's function.
2. Implement the congestion control including slow start and congestion avoidance.
3. You need to reset the threshold to a lower value in order to enter the status of congestion avoidance.

Server (slow start):



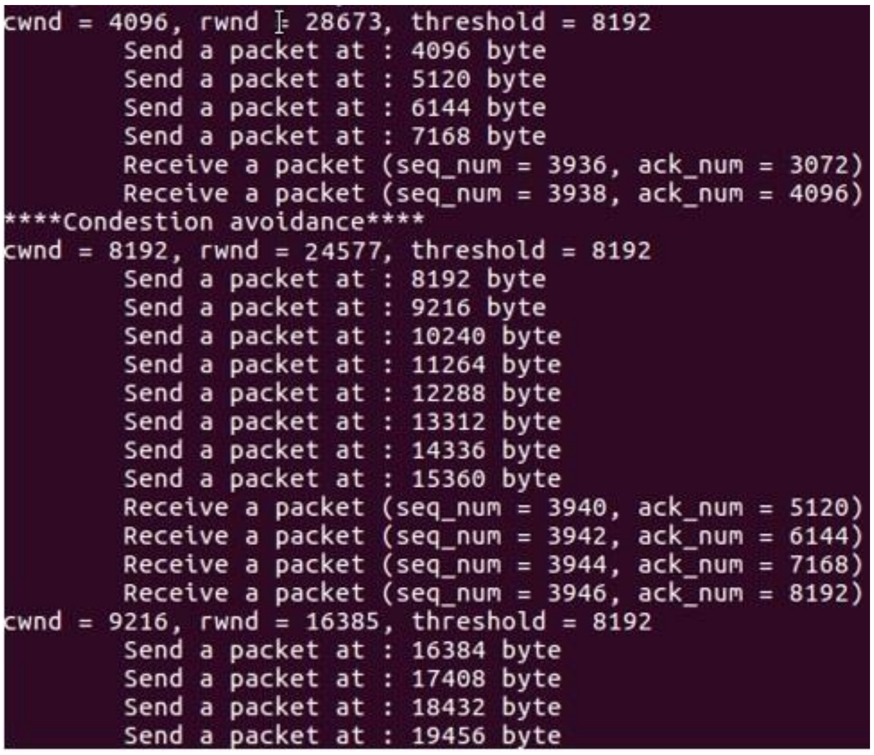
```
*****Slow atart*****
cwnd = 1, rwnd = 32768, threshold = 8192
        Send a packet at : 1 bytes
cwnd = 2, rwnd = 32767, threshold = 8192
        Send a packet at : 2 byte
        Receive a packet (seq_num = 4568, ack_num = 4)
cwnd = 4, rwnd = 32765, threshold = 8192
        Send a packet at : 4 byte
cwnd = 8, rwnd = 32761, threshold = 8192
        Send a packet at : 8 byte
        Receive a packet (seq_num = 4570, ack_num = 16)
cwnd = 16, rwnd = 32753, threshold = 8192
        Send a packet at : 16 byte
cwnd = 32, rwnd = 32737, threshold = 8192
        Send a packet at : 32 byte
        Receive a packet (seq_num = 4572, ack_num = 64)
```

Client (slow start):



```
Receive a packet (seq_num = 1, ack_num = 2416)
Receive a packet (seq_num = 2, ack_num = 2417)
Receive a packet (seq_num = 4, ack_num = 2418)
Receive a packet (seq_num = 8, ack_num = 2419)
Receive a packet (seq_num = 16, ack_num = 2420)
Receive a packet (seq_num = 32, ack_num = 2421)
```

Server (congestion avoidance):



```
cwnd = 4096, rwnd = 28673, threshold = 8192
        Send a packet at : 4096 byte
        Send a packet at : 5120 byte
        Send a packet at : 6144 byte
        Send a packet at : 7168 byte
        Receive a packet (seq_num = 3936, ack_num = 3072)
        Receive a packet (seq_num = 3938, ack_num = 4096)
****Condestion avoidance****
cwnd = 8192, rwnd = 24577, threshold = 8192
        Send a packet at : 8192 byte
        Send a packet at : 9216 byte
        Send a packet at : 10240 byte
        Send a packet at : 11264 byte
        Send a packet at : 12288 byte
        Send a packet at : 13312 byte
        Send a packet at : 14336 byte
        Send a packet at : 15360 byte
        Receive a packet (seq_num = 3940, ack_num = 5120)
        Receive a packet (seq_num = 3942, ack_num = 6144)
        Receive a packet (seq_num = 3944, ack_num = 7168)
        Receive a packet (seq_num = 3946, ack_num = 8192)
cwnd = 9216, rwnd = 16385, threshold = 8192
        Send a packet at : 16384 byte
        Send a packet at : 17408 byte
        Send a packet at : 18432 byte
        Send a packet at : 19456 byte
```

Step 5:

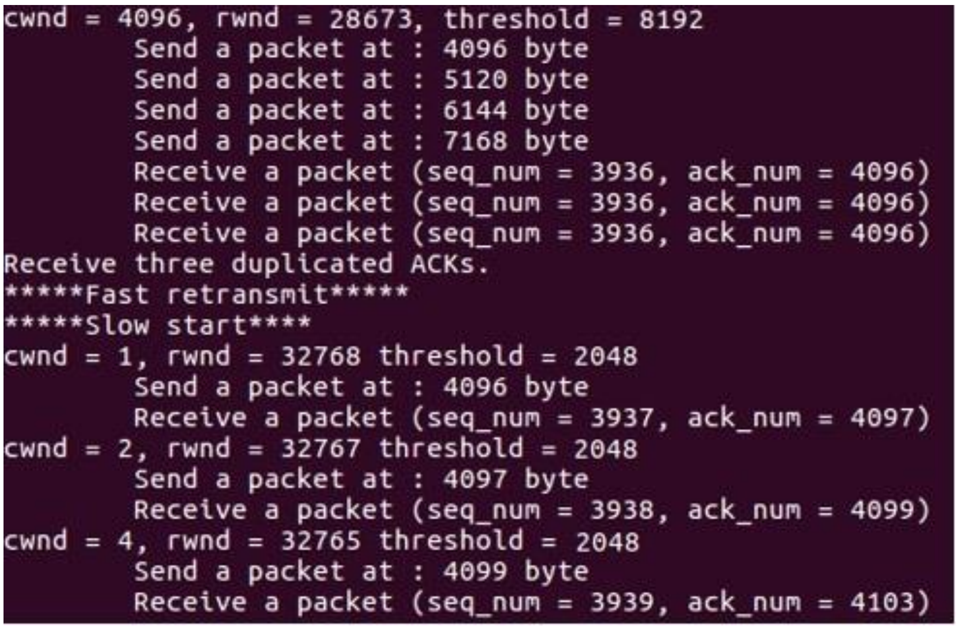
1. Including the previous step's function.

2. Implement the mechanism of fast retransmit. (Tahoe)

3. You need to create a packet loss at the packet which starts at 4096 byte to get duplicated ACKs, then the fast retransmit will be executed.

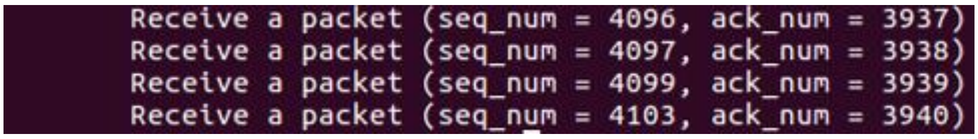
4. You can ignore the mechanism of delayed ACK to implement this step in order to check the receive packets.

Server:



```
cwnd = 4096, rwnd = 28673, threshold = 8192
        Send a packet at : 4096 byte
        Send a packet at : 5120 byte
        Send a packet at : 6144 byte
        Send a packet at : 7168 byte
        Receive a packet (seq_num = 3936, ack_num = 4096)
        Receive a packet (seq_num = 3936, ack_num = 4096)
        Receive a packet (seq_num = 3936, ack_num = 4096)
Receive three duplicated ACKs.
*****Fast retransmit*****
*****Slow start****
cwnd = 1, rwnd = 32768 threshold = 2048
        Send a packet at : 4096 byte
        Receive a packet (seq_num = 3937, ack_num = 4097)
cwnd = 2, rwnd = 32767 threshold = 2048
        Send a packet at : 4097 byte
        Receive a packet (seq_num = 3938, ack_num = 4099)
cwnd = 4, rwnd = 32765 threshold = 2048
        Send a packet at : 4099 byte
        Receive a packet (seq_num = 3939, ack_num = 4103)
```

Client:

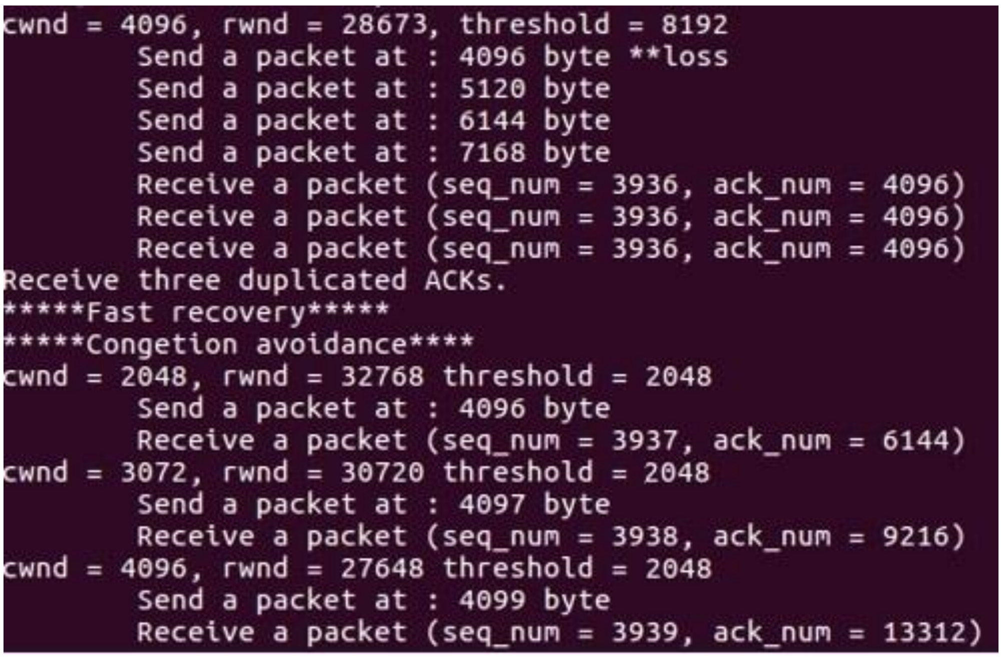


```
        Receive a packet (seq_num = 4096, ack_num = 3937)
        Receive a packet (seq_num = 4097, ack_num = 3938)
        Receive a packet (seq_num = 4099, ack_num = 3939)
        Receive a packet (seq_num = 4103, ack_num = 3940)
```

Step 6:

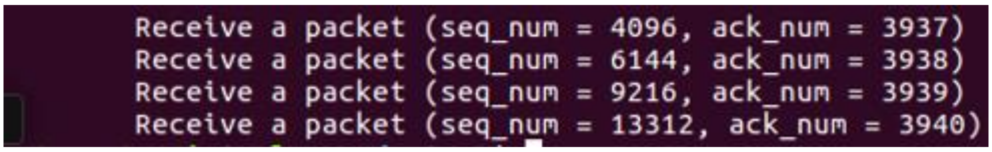
1. Including the previous step’s function.
2. Implement the mechanism of fast recovery. (TCP Reno)
3. You need to design a packet loss at byte 4096 to get duplicated ACKs, then the fast retransmit will execute, and enter the state of fast recovery.
4. You can ignore the mechanism of delayed ACK to implement this step in order to check the receive packets

Server:



```
cwnd = 4096, rwnd = 28673, threshold = 8192
        Send a packet at : 4096 byte **loss
        Send a packet at : 5120 byte
        Send a packet at : 6144 byte
        Send a packet at : 7168 byte
        Receive a packet (seq_num = 3936, ack_num = 4096)
        Receive a packet (seq_num = 3936, ack_num = 4096)
        Receive a packet (seq_num = 3936, ack_num = 4096)
Receive three duplicated ACKs.
*****Fast recovery*****
*****Congetion avoidance****
cwnd = 2048, rwnd = 32768 threshold = 2048
        Send a packet at : 4096 byte
        Receive a packet (seq_num = 3937, ack_num = 6144)
cwnd = 3072, rwnd = 30720 threshold = 2048
        Send a packet at : 4097 byte
        Receive a packet (seq_num = 3938, ack_num = 9216)
cwnd = 4096, rwnd = 27648 threshold = 2048
        Send a packet at : 4099 byte
        Receive a packet (seq_num = 3939, ack_num = 13312)
```

Client :

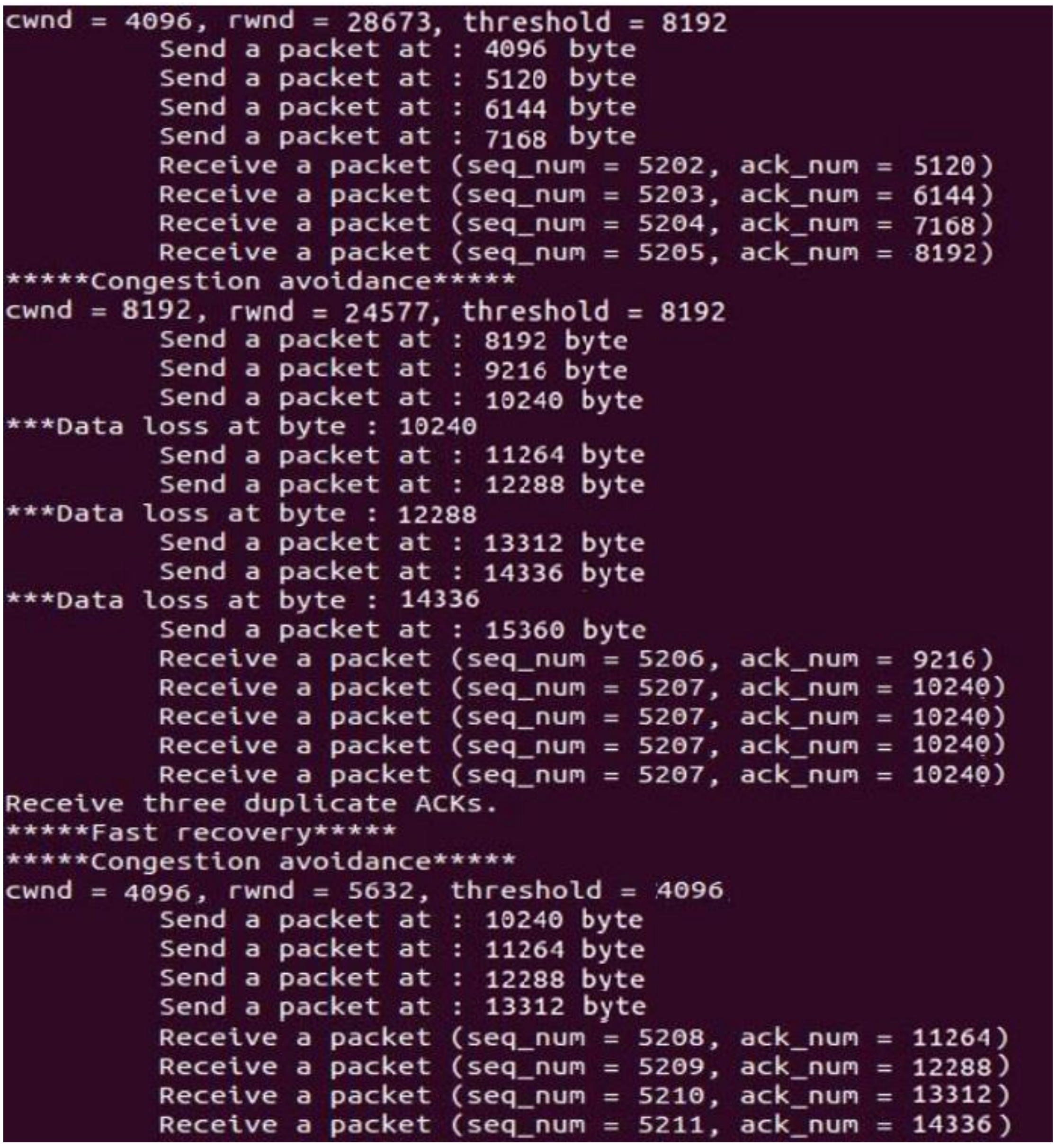


```
        Receive a packet (seq_num = 4096, ack_num = 3937)
        Receive a packet (seq_num = 6144, ack_num = 3938)
        Receive a packet (seq_num = 9216, ack_num = 3939)
        Receive a packet (seq_num = 13312, ack_num = 3940)
```

Step 7:

1. Including the previous step's function.
2. Implement the mechanism of TCP SACK, and using three blocks in this step.
3. You need to design a packet loss at byte 10240, 12288 and 14336 to create three SACK blocks.
4. You can ignore the mechanism of delayed ACK to implement this step in order to check the receive packets.

Server:



```
cwnd = 4096, rwnd = 28673, threshold = 8192
          Send a packet at : 4096 byte
          Send a packet at : 5120 byte
          Send a packet at : 6144 byte
          Send a packet at : 7168 byte
          Receive a packet (seq_num = 5202, ack_num = 5120)
          Receive a packet (seq_num = 5203, ack_num = 6144)
          Receive a packet (seq_num = 5204, ack_num = 7168)
          Receive a packet (seq_num = 5205, ack_num = 8192)
*****Congestion avoidance*****
cwnd = 8192, rwnd = 24577, threshold = 8192
          Send a packet at : 8192 byte
          Send a packet at : 9216 byte
          Send a packet at : 10240 byte
***Data loss at byte : 10240
          Send a packet at : 11264 byte
          Send a packet at : 12288 byte
***Data loss at byte : 12288
          Send a packet at : 13312 byte
          Send a packet at : 14336 byte
***Data loss at byte : 14336
          Send a packet at : 15360 byte
          Receive a packet (seq_num = 5206, ack_num = 9216)
          Receive a packet (seq_num = 5207, ack_num = 10240)
          Receive a packet (seq_num = 5207, ack_num = 10240)
          Receive a packet (seq_num = 5207, ack_num = 10240)
          Receive a packet (seq_num = 5207, ack_num = 10240)
Receive three duplicate ACKs.
*****Fast recovery*****
*****Congestion avoidance*****
cwnd = 4096, rwnd = 5632, threshold = 4096
          Send a packet at : 10240 byte
          Send a packet at : 11264 byte
          Send a packet at : 12288 byte
          Send a packet at : 13312 byte
          Receive a packet (seq_num = 5208, ack_num = 11264)
          Receive a packet (seq_num = 5209, ack_num = 12288)
          Receive a packet (seq_num = 5210, ack_num = 13312)
          Receive a packet (seq_num = 5211, ack_num = 14336)
```

Client: There is no strict output format, just show all packets in demo.

Step 8:

1. Including the previous step’s function.

2. Your output should demonstrate that the server can simultaneously receive request from 200 clients.