<2024 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, congestion control, etc. Second, you have to implement TCP in the application layer and call **UDP** to transmit TCP packets.

Rules

1. Run your program on **Ubuntu 20.04 or Ubuntu 22.04** platform.
2. Do not just copy and paste homework from your classmates or seniors, etc. If TAs find the situation, any participants will get **ZERO** on this homework.
3. If you have any questions except debugging, you can either send an email or drop by **EC5008** (High Speed Network Lab) to ask TAs .
4. You can use any programming language to do this homework.
5. You should create the **MAKEFILE** if your project has to be compiled before execution.
6. You should create a **README** file to show the usage of your project.
7. You also need to create a **PDF** that contains pictures of your program's output at every step.
8. **The output format in this PDF is for reference only**. Please present your results according to the requirements of each step. Ex: how to demonstrate the interactions between server and clients.
9. In each step, you can write a new program, respectively.
10. Pack your code, README, (MAKEFILE,) and pdf into a "**Homework.zip**".
11. Don't pack videos, pictures, and other unnecessary files into the zip file.
12. When you come for a demo, you can bring your computer to EC5008 to demonstrate your project, or you should email TAs to inform that you need our help.
13. You have to deeply understand your program because TAs will ask you the **concept** of your code.

Deadline

You should upload your homework to the **Cyber University** before **2024/06/14 23:59**.
If you do not submit your assignment on time, you will get **ZERO** on this homework.

Demo

The following figure shows the time you can come for a demo. You can mail TA for reservation.
Demo deadline: 2024/06/14 17:00.

	Mon.	Tue.	Wed.	Thu.	Fri.
10:00~12:00		✔	✔		✔
14:00~17:00		✔	✔		16:00~17:00

Description

You have to obey the following schema:

The TCP segment structure.

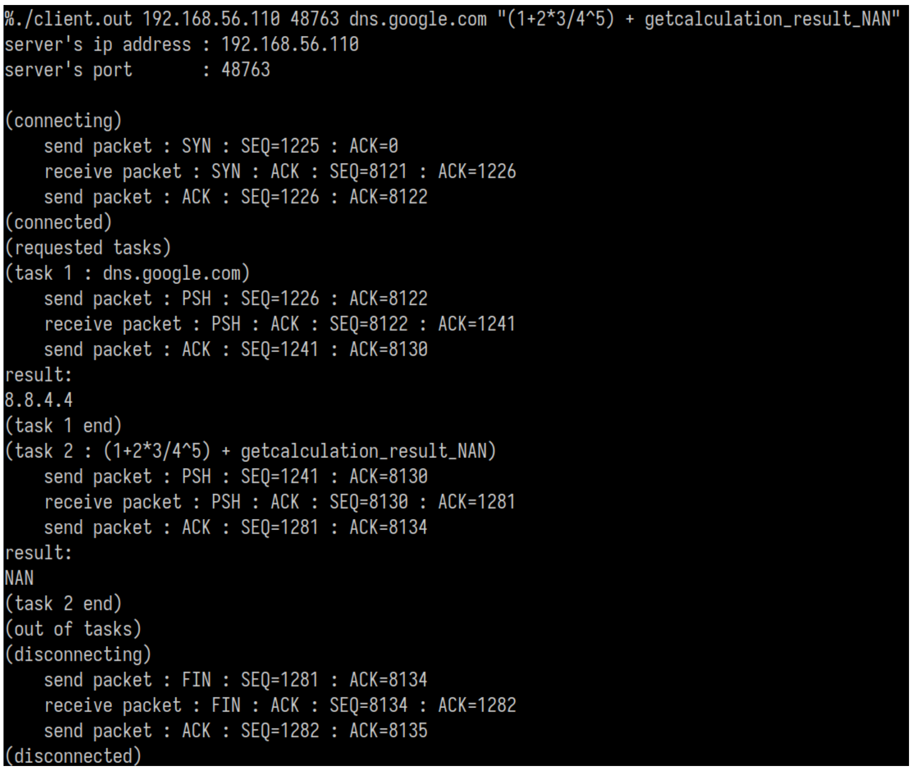
The initial sequence number should be set randomly (1~10000).

The program should work fine under a subnetwork.

Step 1:

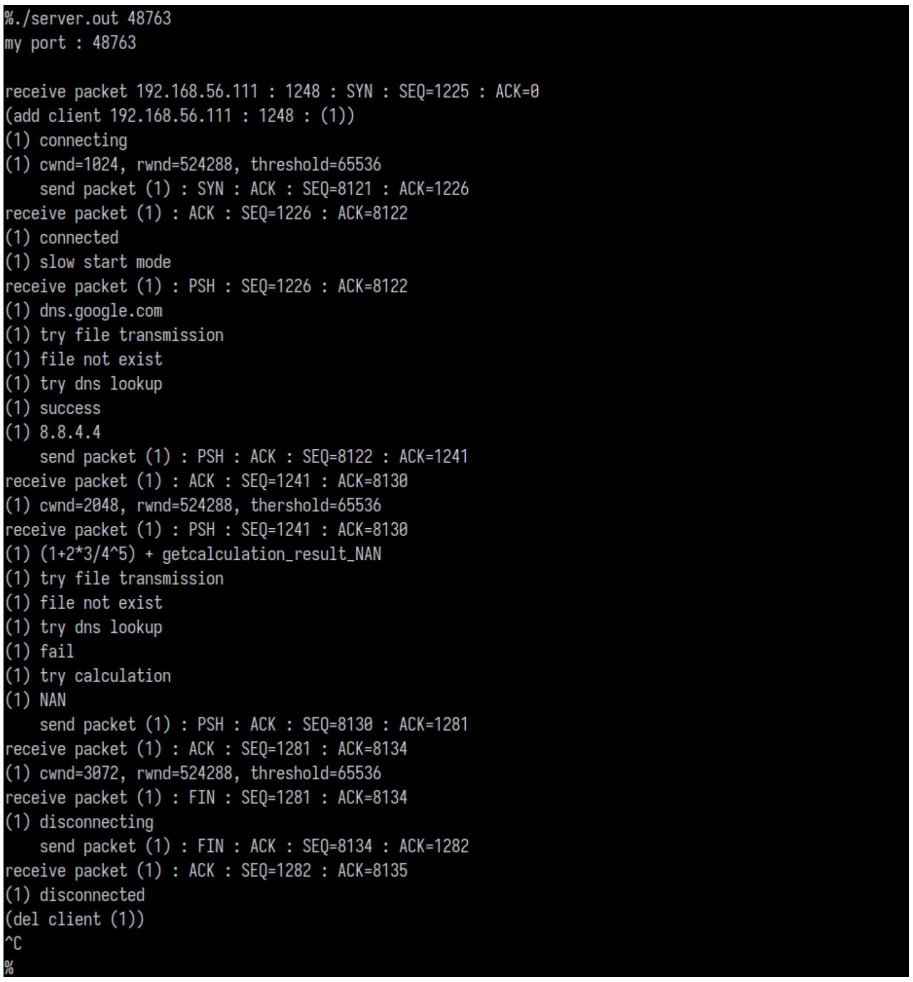
1. Set the parameters including initial RTT (30 ms), MSS (1 Kbytes), threshold (64 Kbytes), and the receiver's buffer size (512 Kbytes), etc.
2. You have to transmit files, perform mathematical calculations, and perform DNS functions in this step. A client could request a single job or multiple jobs in one run. The server should send the files, the results of DNS, and the results of mathematical calculations back to multiple clients at the same time. (You can use a fork or thread.)
3. In the DNS function, the client sends a domain name to the server. The server acts as a DNS proxy to get the DNS result of the request. Finally, the server sends the result to the client and the client should show the result on screen.
   (Ex: You type in "hsnl.cse.nsysu.edu.tw" and then you will get "140.117.172.136" as result.)
4. The mathematical calculations include add, subtract, multiply, divide, power, and square root.
5. You also have to implement the data transmission.
   (You need to ensure that the data are transmitted from the server to clients, and ACK packets are transmitted from clients to the server).
6. Note that the integrity of the transmitted files should be ensured. TAs will use hash function (sha256sum) to check this in demo.
7. You have to print out the status of the server and clients. For example, for the server, which client the server responses to, and for clients, which task is being served.

Client:



```
%./client.out 192.168.56.110 48763 dns.google.com "(1+2*3/4^5) + getcalculation_result_NAN"
server's ip address : 192.168.56.110
server's port       : 48763

(connecting)
    send packet : SYN : SEQ=1225 : ACK=0
    receive packet : SYN : ACK : SEQ=8121 : ACK=1226
    send packet : ACK : SEQ=1226 : ACK=8122
(connected)
(requested tasks)
(task 1 : dns.google.com)
    send packet : PSH : SEQ=1226 : ACK=8122
    receive packet : PSH : ACK : SEQ=8122 : ACK=1241
    send packet : ACK : SEQ=1241 : ACK=8130
result:
8.8.4.4
(task 1 end)
(task 2 : (1+2*3/4^5) + getcalculation_result_NAN)
    send packet : PSH : SEQ=1241 : ACK=8130
    receive packet : PSH : ACK : SEQ=8130 : ACK=1281
    send packet : ACK : SEQ=1281 : ACK=8134
result:
NAN
(task 2 end)
(out of tasks)
(disconnecting)
    send packet : FIN : SEQ=1281 : ACK=8134
    receive packet : FIN : ACK : SEQ=8134 : ACK=1282
    send packet : ACK : SEQ=1282 : ACK=8135
(disconnected)
```

Server:

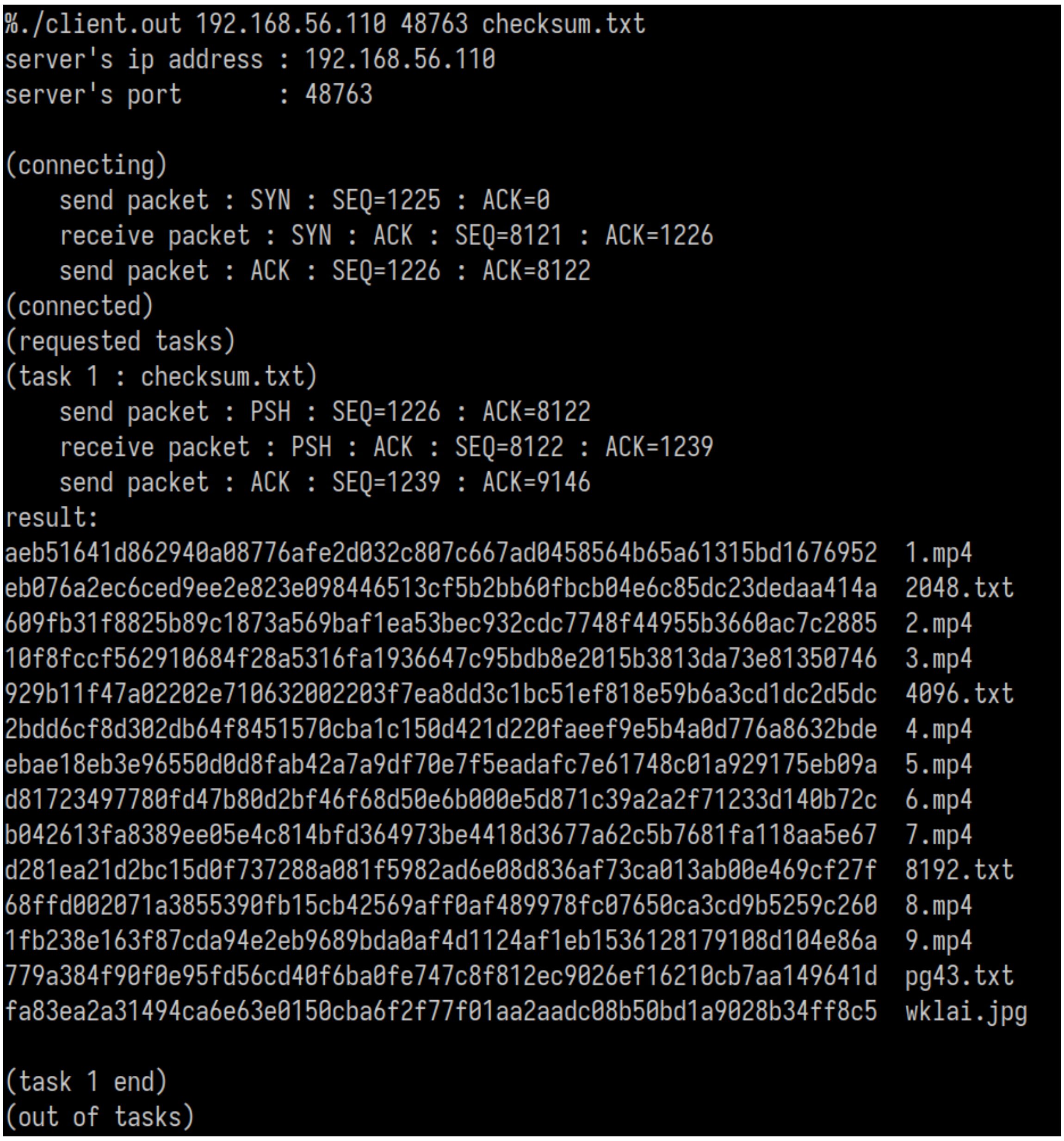


```
%./server.out 48763
my port : 48763

receive packet 192.168.56.111 : 1248 : SYN : SEQ=1225 : ACK=0
(add client 192.168.56.111 : 1248 : (1))
(1) connecting
(1) cwnd=1024, rwnd=524288, threshold=65536
    send packet (1) : SYN : ACK : SEQ=8121 : ACK=1226
receive packet (1) : ACK : SEQ=1226 : ACK=8122
(1) connected
(1) slow start mode
receive packet (1) : PSH : SEQ=1226 : ACK=8122
(1) dns.google.com
(1) try file transmission
(1) file not exist
(1) try dns lookup
(1) success
(1) 8.8.4.4
    send packet (1) : PSH : ACK : SEQ=8122 : ACK=1241
receive packet (1) : ACK : SEQ=1241 : ACK=8130
(1) cwnd=2048, rwnd=524288, thershold=65536
receive packet (1) : PSH : SEQ=1241 : ACK=8130
(1) (1+2*3/4^5) + getcalculation_result_NAN
(1) try file transmission
(1) file not exist
(1) try dns lookup
(1) fail
(1) try calculation
(1) NAN
    send packet (1) : PSH : ACK : SEQ=8130 : ACK=1281
receive packet (1) : ACK : SEQ=1281 : ACK=8134
(1) cwnd=3072, rwnd=524288, threshold=65536
receive packet (1) : FIN : SEQ=1281 : ACK=8134
(1) disconnecting
    send packet (1) : FIN : ACK : SEQ=8134 : ACK=1282
receive packet (1) : ACK : SEQ=1282 : ACK=8135
(1) disconnected
(del client (1))
^C
%
```

Step 2:

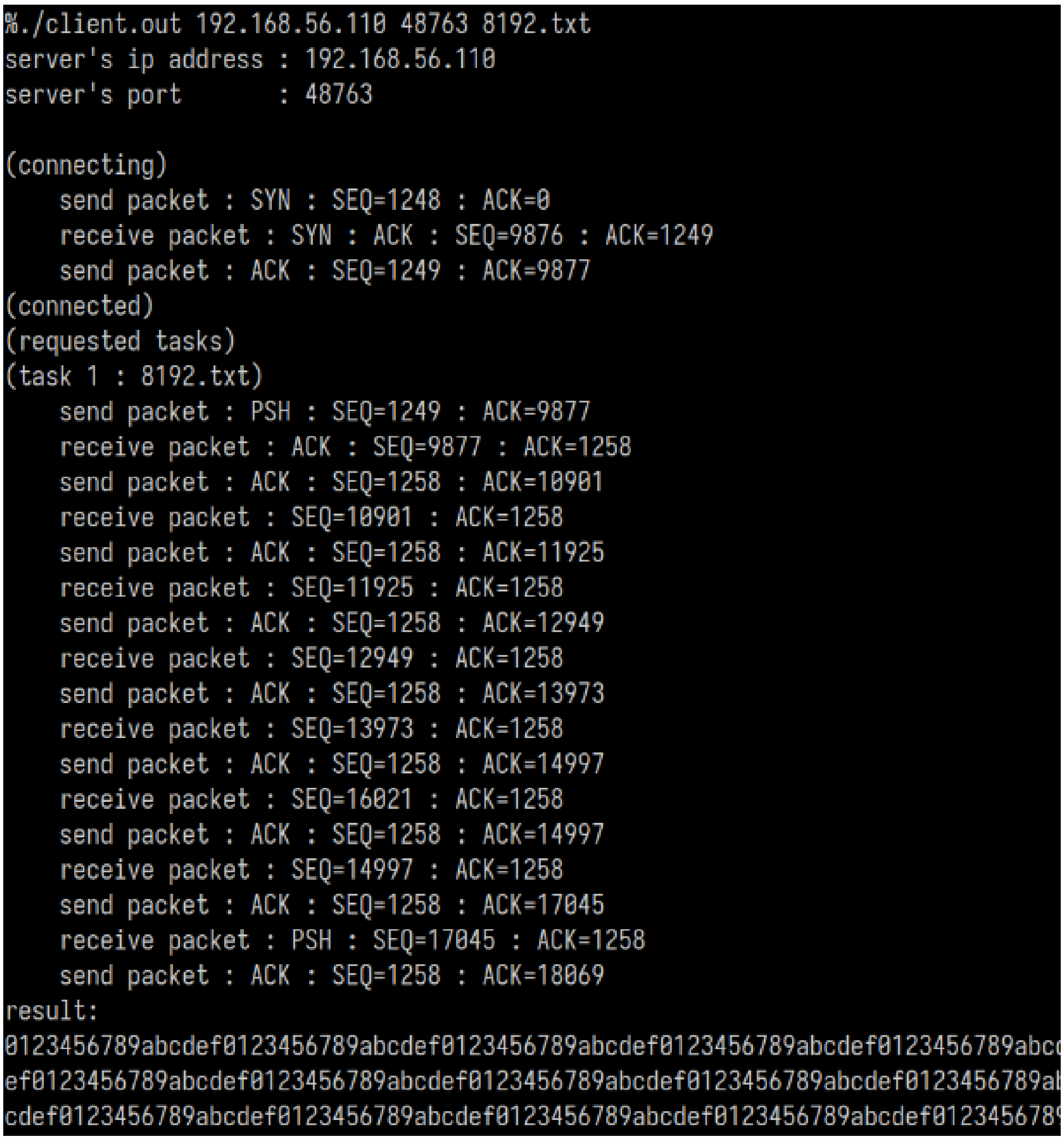
1. Including the previous step's function.
2. The clients are increased to two clients.
3. The server side has to implement some prevention mechanisms to handle simultaneous requests from clients.
4. The clients begin to request various services from the server and display results on the screen.

Client1:



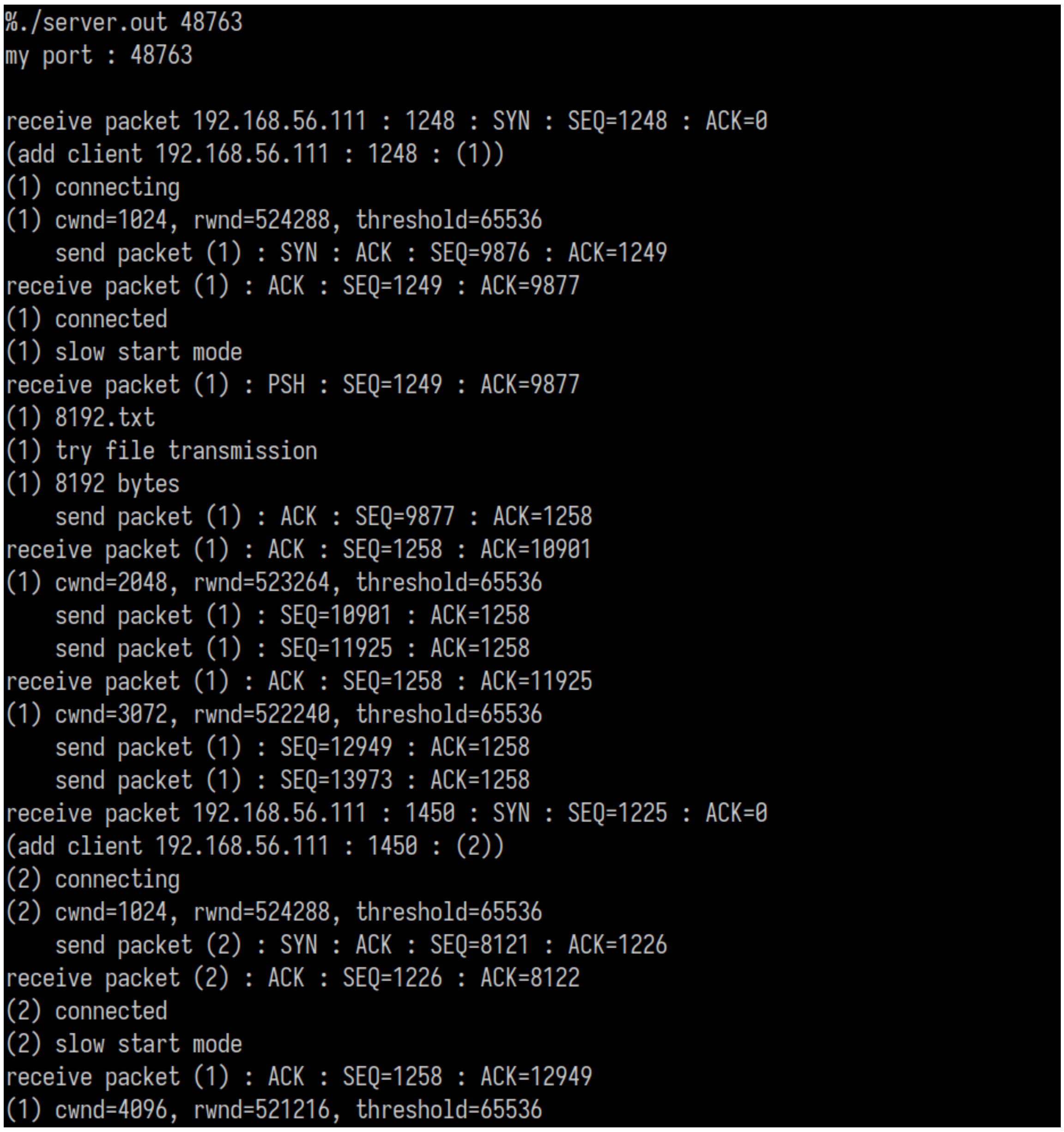
```
%./client.out 192.168.56.110 48763 checksum.txt
server's ip address : 192.168.56.110
server's port       : 48763

(connecting)
    send packet : SYN : SEQ=1225 : ACK=0
    receive packet : SYN : ACK : SEQ=8121 : ACK=1226
    send packet : ACK : SEQ=1226 : ACK=8122
(connected)
(requested tasks)
(task 1 : checksum.txt)
    send packet : PSH : SEQ=1226 : ACK=8122
    receive packet : PSH : ACK : SEQ=8122 : ACK=1239
    send packet : ACK : SEQ=1239 : ACK=9146
result:
aeb51641d862940a08776afe2d032c807c667ad0458564b65a61315bd1676952  1.mp4
eb076a2ec6ced9ee2e823e098446513cf5b2bb60fbcb04e6c85dc23dedaa414a  2048.txt
609fb31f8825b89c1873a569baf1ea53bec932cdc7748f44955b3660ac7c2885  2.mp4
10f8fccf562910684f28a5316fa1936647c95bdb8e2015b3813da73e81350746  3.mp4
929b11f47a02202e710632002203f7ea8dd3c1bc51ef818e59b6a3cd1dc2d5dc  4096.txt
2bdd6cf8d302db64f8451570cba1c150d421d220faeef9e5b4a0d776a8632bde  4.mp4
ebae18eb3e96550d0d8fab42a7a9df70e7f5eadafc7e61748c01a929175eb09a  5.mp4
d81723497780fd47b80d2bf46f68d50e6b000e5d871c39a2a2f71233d140b72c  6.mp4
b042613fa8389ee05e4c814bfd364973be4418d3677a62c5b7681fa118aa5e67  7.mp4
d281ea21d2bc15d0f737288a081f5982ad6e08d836af73ca013ab00e469cf27f  8192.txt
68ffd002071a3855390fb15cb42569aff0af489978fc07650ca3cd9b5259c260  8.mp4
1fb238e163f87cda94e2eb9689bda0af4d1124af1eb1536128179108d104e86a  9.mp4
779a384f90f0e95fd56cd40f6ba0fe747c8f812ec9026ef16210cb7aa149641d  pg43.txt
fa83ea2a31494ca6e63e0150cba6f2f77f01aa2aadc08b50bd1a9028b34ff8c5  wklai.jpg

(task 1 end)
(out of tasks)
```

Client2:

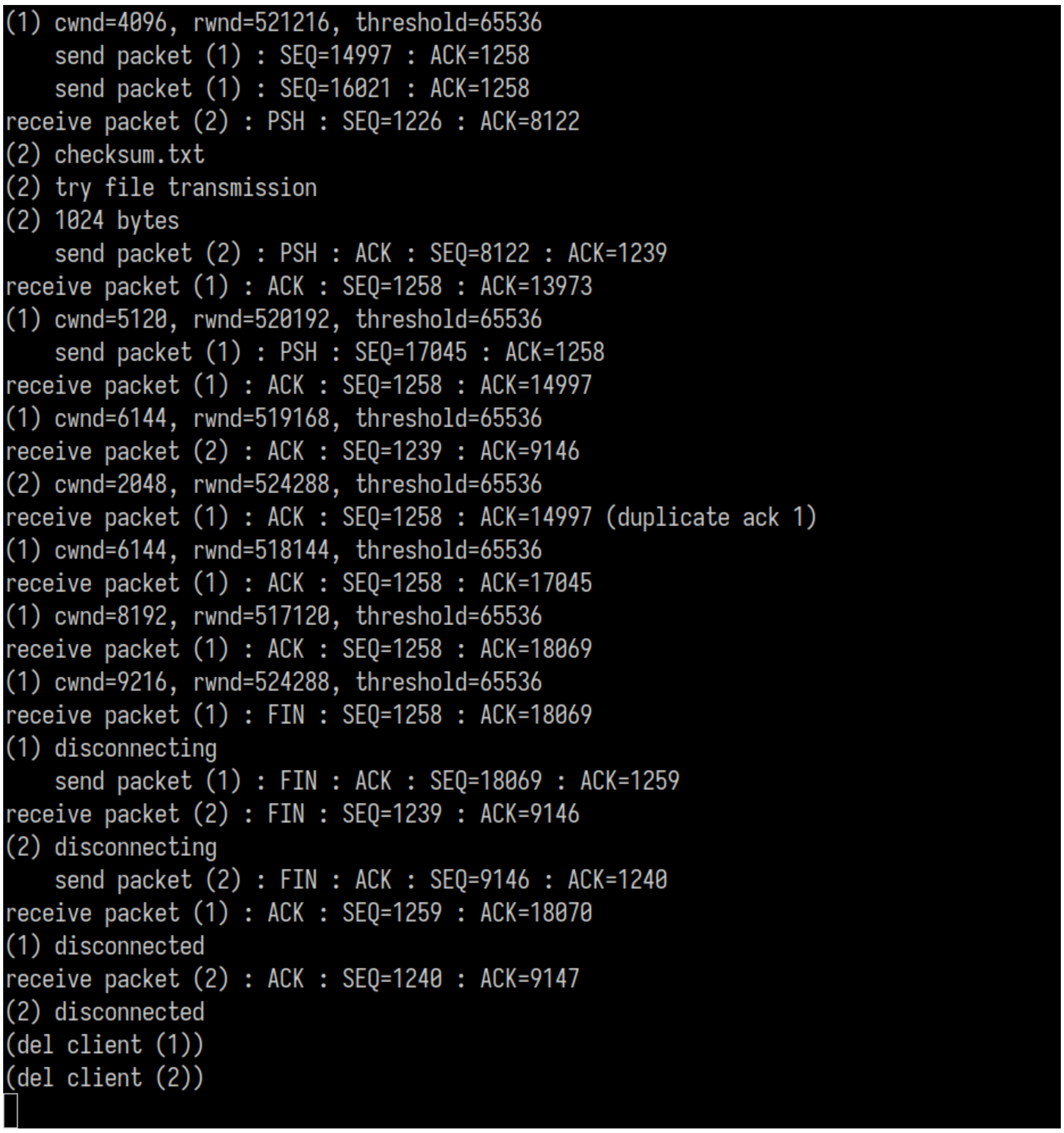


```
%./client.out 192.168.56.110 48763 8192.txt
server's ip address : 192.168.56.110
server's port       : 48763

(connecting)
    send packet : SYN : SEQ=1248 : ACK=0
    receive packet : SYN : ACK : SEQ=9876 : ACK=1249
    send packet : ACK : SEQ=1249 : ACK=9877
(connected)
(requested tasks)
(task 1 : 8192.txt)
    send packet : PSH : SEQ=1249 : ACK=9877
    receive packet : ACK : SEQ=9877 : ACK=1258
    send packet : ACK : SEQ=1258 : ACK=10901
    receive packet : SEQ=10901 : ACK=1258
    send packet : ACK : SEQ=1258 : ACK=11925
    receive packet : SEQ=11925 : ACK=1258
    send packet : ACK : SEQ=1258 : ACK=12949
    receive packet : SEQ=12949 : ACK=1258
    send packet : ACK : SEQ=1258 : ACK=13973
    receive packet : SEQ=13973 : ACK=1258
    send packet : ACK : SEQ=1258 : ACK=14997
    receive packet : SEQ=16021 : ACK=1258
    send packet : ACK : SEQ=1258 : ACK=14997
    receive packet : SEQ=14997 : ACK=1258
    send packet : ACK : SEQ=1258 : ACK=17045
    receive packet : PSH : SEQ=17045 : ACK=1258
    send packet : ACK : SEQ=1258 : ACK=18069
result:
0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789abc
ef0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789ab
cdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef012345678
```

Server:



```
%./server.out 48763
my port : 48763

receive packet 192.168.56.111 : 1248 : SYN : SEQ=1248 : ACK=0
(add client 192.168.56.111 : 1248 : (1))
(1) connecting
(1) cwnd=1024, rwnd=524288, threshold=65536
    send packet (1) : SYN : ACK : SEQ=9876 : ACK=1249
receive packet (1) : ACK : SEQ=1249 : ACK=9877
(1) connected
(1) slow start mode
receive packet (1) : PSH : SEQ=1249 : ACK=9877
(1) 8192.txt
(1) try file transmission
(1) 8192 bytes
    send packet (1) : ACK : SEQ=9877 : ACK=1258
receive packet (1) : ACK : SEQ=1258 : ACK=10901
(1) cwnd=2048, rwnd=523264, threshold=65536
    send packet (1) : SEQ=10901 : ACK=1258
    send packet (1) : SEQ=11925 : ACK=1258
receive packet (1) : ACK : SEQ=1258 : ACK=11925
(1) cwnd=3072, rwnd=522240, threshold=65536
    send packet (1) : SEQ=12949 : ACK=1258
    send packet (1) : SEQ=13973 : ACK=1258
receive packet 192.168.56.111 : 1450 : SYN : SEQ=1225 : ACK=0
(add client 192.168.56.111 : 1450 : (2))
(2) connecting
(2) cwnd=1024, rwnd=524288, threshold=65536
    send packet (2) : SYN : ACK : SEQ=8121 : ACK=1226
receive packet (2) : ACK : SEQ=1226 : ACK=8122
(2) connected
(2) slow start mode
receive packet (1) : ACK : SEQ=1258 : ACK=12949
(1) cwnd=4096, rwnd=521216, threshold=65536
```

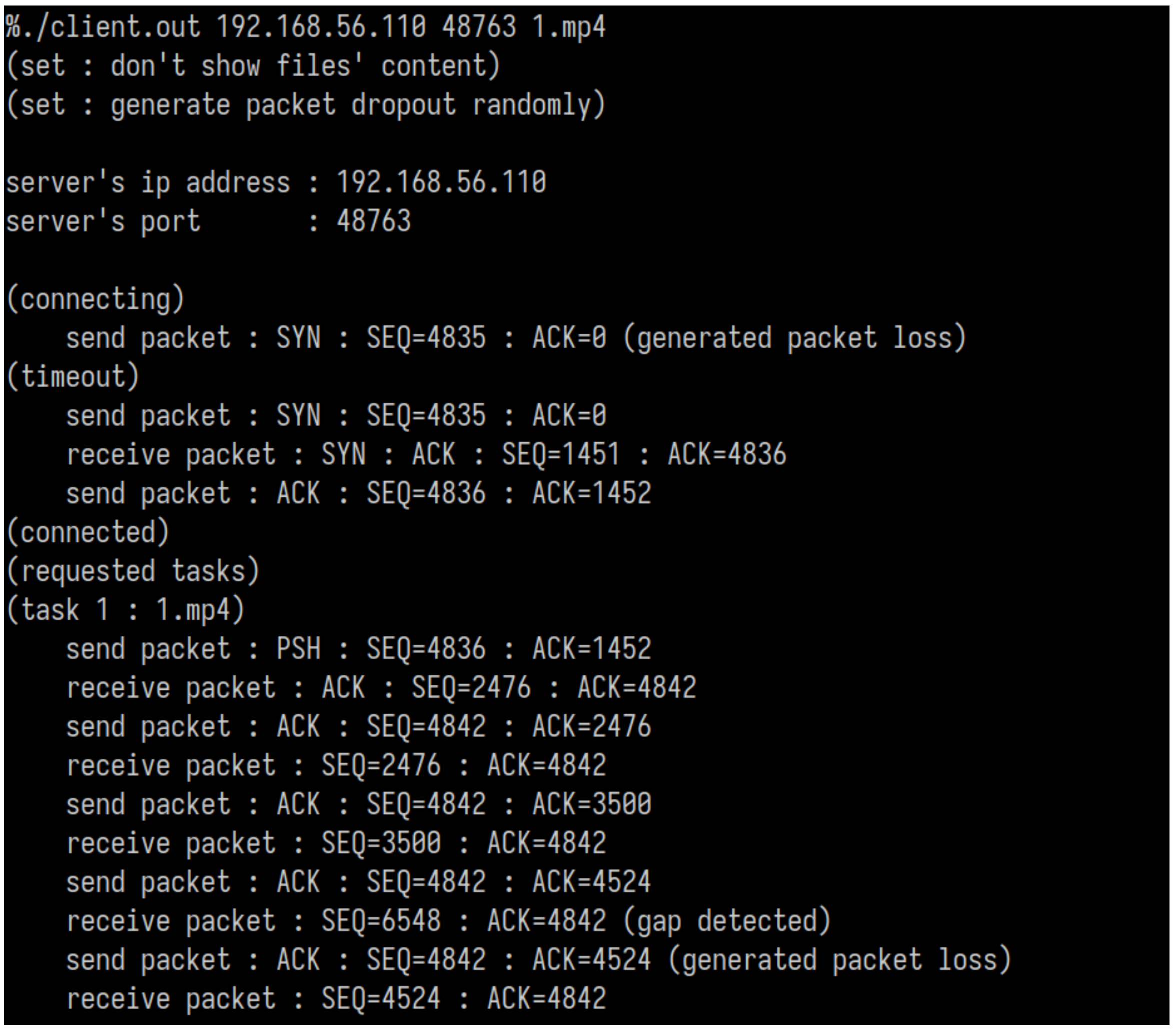


```
(1) cwnd=4096, rwnd=521216, threshold=65536
    send packet (1) : SEQ=14997 : ACK=1258
    send packet (1) : SEQ=16021 : ACK=1258
receive packet (2) : PSH : SEQ=1226 : ACK=8122
(2) checksum.txt
(2) try file transmission
(2) 1024 bytes
    send packet (2) : PSH : ACK : SEQ=8122 : ACK=1239
receive packet (1) : ACK : SEQ=1258 : ACK=13973
(1) cwnd=5120, rwnd=520192, threshold=65536
    send packet (1) : PSH : SEQ=17045 : ACK=1258
receive packet (1) : ACK : SEQ=1258 : ACK=14997
(1) cwnd=6144, rwnd=519168, threshold=65536
receive packet (2) : ACK : SEQ=1239 : ACK=9146
(2) cwnd=2048, rwnd=524288, threshold=65536
receive packet (1) : ACK : SEQ=1258 : ACK=14997 (duplicate ack 1)
(1) cwnd=6144, rwnd=518144, threshold=65536
receive packet (1) : ACK : SEQ=1258 : ACK=17045
(1) cwnd=8192, rwnd=517120, threshold=65536
receive packet (1) : ACK : SEQ=1258 : ACK=18069
(1) cwnd=9216, rwnd=524288, threshold=65536
receive packet (1) : FIN : SEQ=1258 : ACK=18069
(1) disconnecting
    send packet (1) : FIN : ACK : SEQ=18069 : ACK=1259
receive packet (2) : FIN : SEQ=1239 : ACK=9146
(2) disconnecting
    send packet (2) : FIN : ACK : SEQ=9146 : ACK=1240
receive packet (1) : ACK : SEQ=1259 : ACK=18070
(1) disconnected
receive packet (2) : ACK : SEQ=1240 : ACK=9147
(2) disconnected
(del client (1))
(del client (2))
```

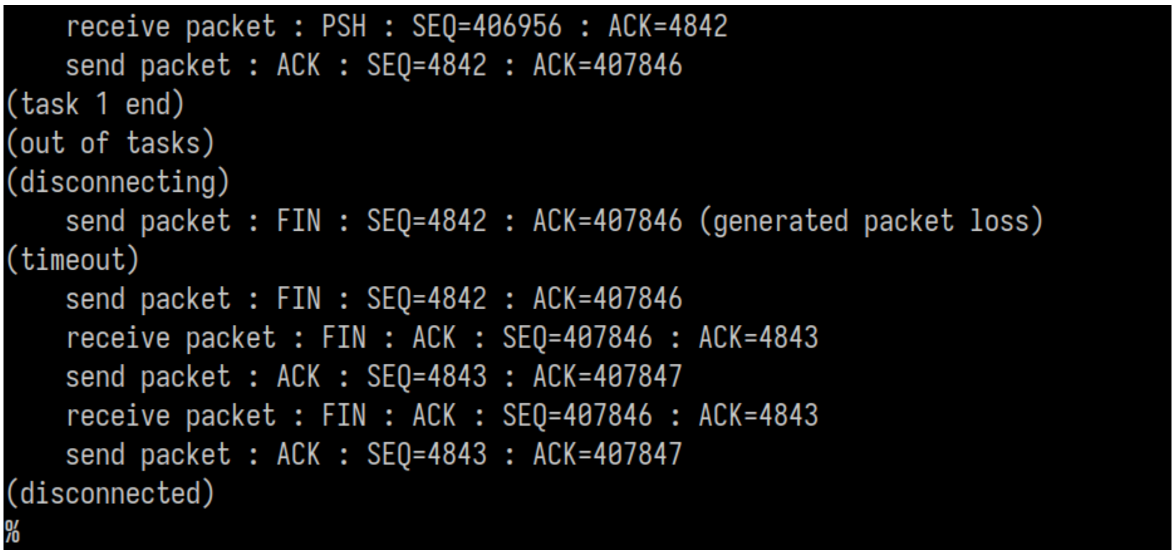
Step 3:

1. Including the previous steps' function.
2. You should randomly generate some packet losses under random distribution with a mean of 10^{-6} and print the ACK number of the lost packet.
3. Note that every packet includes SYN packets, ACK packets, and FIN-ACK packets can be chosen to drop out. Make sure that the timeout retransmission mechanism is implemented for all packets.

Client:

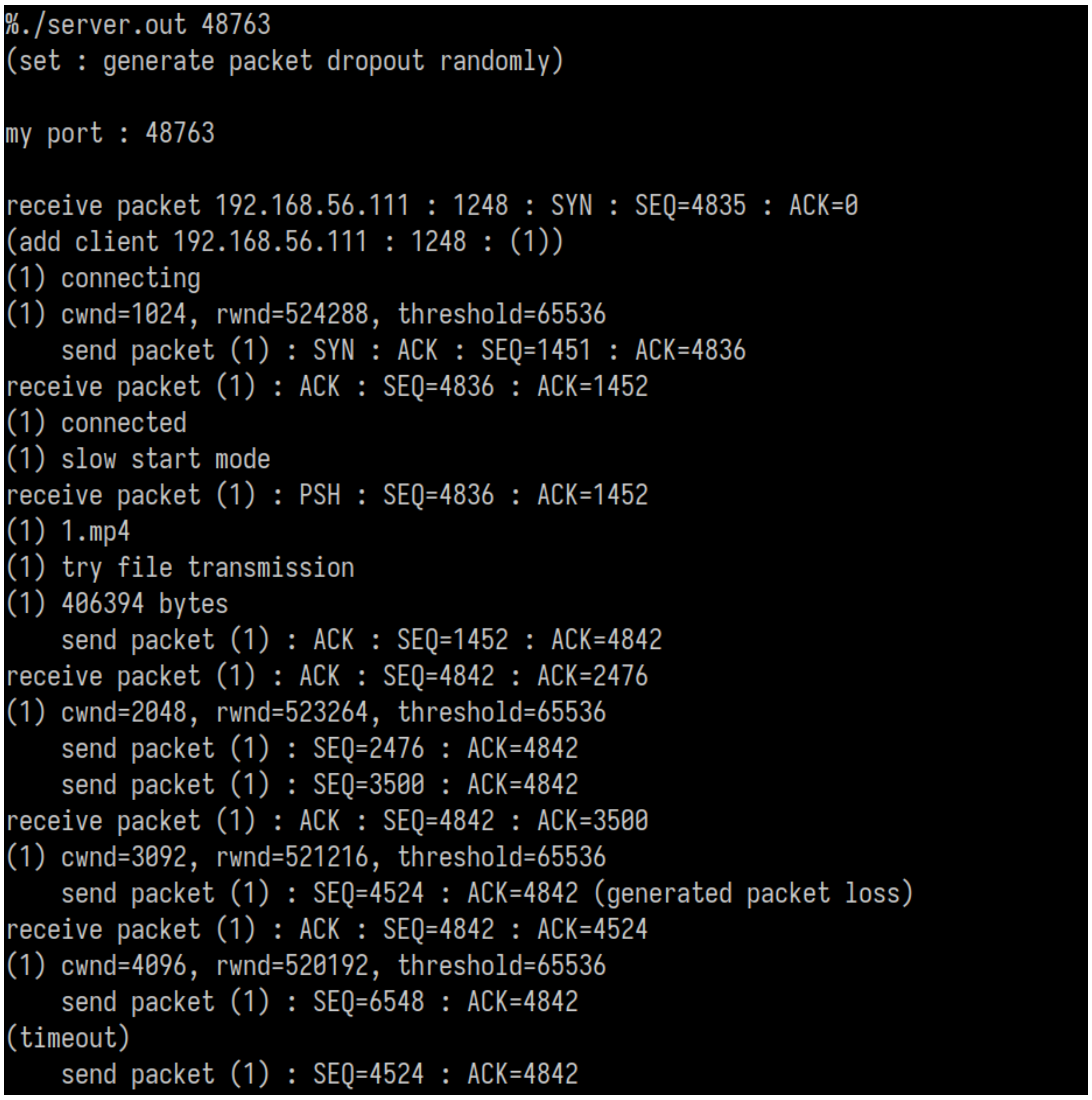


```
%./client.out 192.168.56.110 48763 1.mp4
(set : don't show files' content)
(set : generate packet dropout randomly)

server's ip address : 192.168.56.110
server's port       : 48763

(connecting)
    send packet : SYN : SEQ=4835 : ACK=0 (generated packet loss)
(timeout)
    send packet : SYN : SEQ=4835 : ACK=0
    receive packet : SYN : ACK : SEQ=1451 : ACK=4836
    send packet : ACK : SEQ=4836 : ACK=1452
(connected)
(requested tasks)
(task 1 : 1.mp4)
    send packet : PSH : SEQ=4836 : ACK=1452
    receive packet : ACK : SEQ=2476 : ACK=4842
    send packet : ACK : SEQ=4842 : ACK=2476
    receive packet : SEQ=2476 : ACK=4842
    send packet : ACK : SEQ=4842 : ACK=3500
    receive packet : SEQ=3500 : ACK=4842
    send packet : ACK : SEQ=4842 : ACK=4524
    receive packet : SEQ=6548 : ACK=4842 (gap detected)
    send packet : ACK : SEQ=4842 : ACK=4524 (generated packet loss)
    receive packet : SEQ=4524 : ACK=4842
```

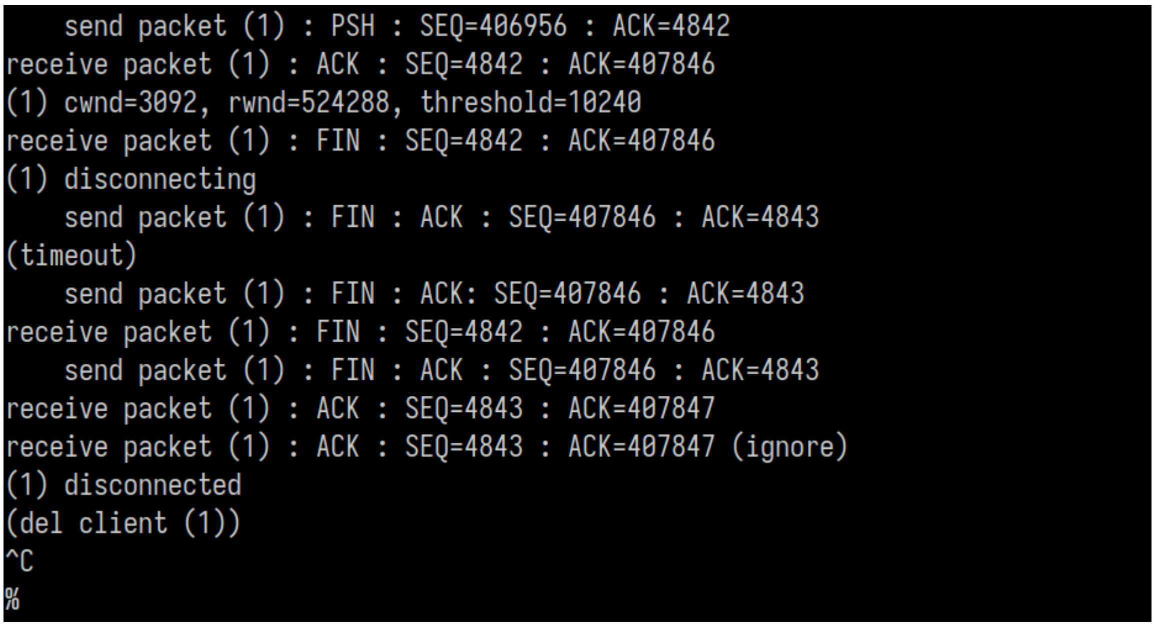


```
    receive packet : PSH : SEQ=406956 : ACK=4842
    send packet : ACK : SEQ=4842 : ACK=407846
(task 1 end)
(out of tasks)
(disconnecting)
    send packet : FIN : SEQ=4842 : ACK=407846 (generated packet loss)
(timeout)
    send packet : FIN : SEQ=4842 : ACK=407846
    receive packet : FIN : ACK : SEQ=407846 : ACK=4843
    send packet : ACK : SEQ=4843 : ACK=407847
    receive packet : FIN : ACK : SEQ=407846 : ACK=4843
    send packet : ACK : SEQ=4843 : ACK=407847
(disconnected)
%
```

Server:



```
%./server.out 48763
(set : generate packet dropout randomly)

my port : 48763

receive packet 192.168.56.111 : 1248 : SYN : SEQ=4835 : ACK=0
(add client 192.168.56.111 : 1248 : (1))
(1) connecting
(1) cwnd=1024, rwnd=524288, threshold=65536
    send packet (1) : SYN : ACK : SEQ=1451 : ACK=4836
receive packet (1) : ACK : SEQ=4836 : ACK=1452
(1) connected
(1) slow start mode
receive packet (1) : PSH : SEQ=4836 : ACK=1452
(1) 1.mp4
(1) try file transmission
(1) 406394 bytes
    send packet (1) : ACK : SEQ=1452 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=2476
(1) cwnd=2048, rwnd=523264, threshold=65536
    send packet (1) : SEQ=2476 : ACK=4842
    send packet (1) : SEQ=3500 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=3500
(1) cwnd=3092, rwnd=521216, threshold=65536
    send packet (1) : SEQ=4524 : ACK=4842 (generated packet loss)
receive packet (1) : ACK : SEQ=4842 : ACK=4524
(1) cwnd=4096, rwnd=520192, threshold=65536
    send packet (1) : SEQ=6548 : ACK=4842
(timeout)
    send packet (1) : SEQ=4524 : ACK=4842
```

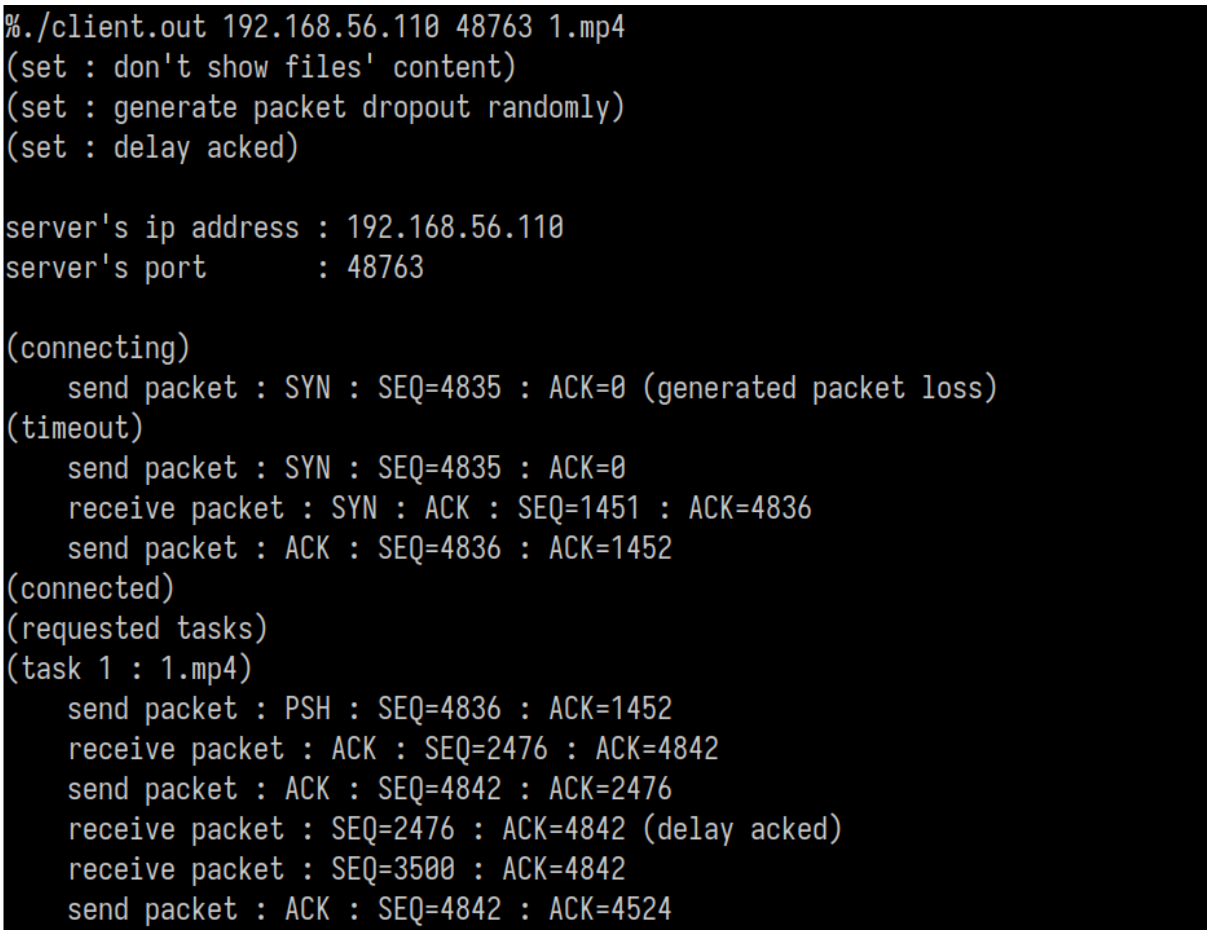


```
    send packet (1) : PSH : SEQ=406956 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=407846
(1) cwnd=3092, rwnd=524288, threshold=10240
receive packet (1) : FIN : SEQ=4842 : ACK=407846
(1) disconnecting
    send packet (1) : FIN : ACK : SEQ=407846 : ACK=4843
(timeout)
    send packet (1) : FIN : ACK: SEQ=407846 : ACK=4843
receive packet (1) : FIN : SEQ=4842 : ACK=407846
    send packet (1) : FIN : ACK : SEQ=407846 : ACK=4843
receive packet (1) : ACK : SEQ=4843 : ACK=407847
receive packet (1) : ACK : SEQ=4843 : ACK=407847 (ignore)
(1) disconnected
(del client (1))
^C
%
```

Step 4:

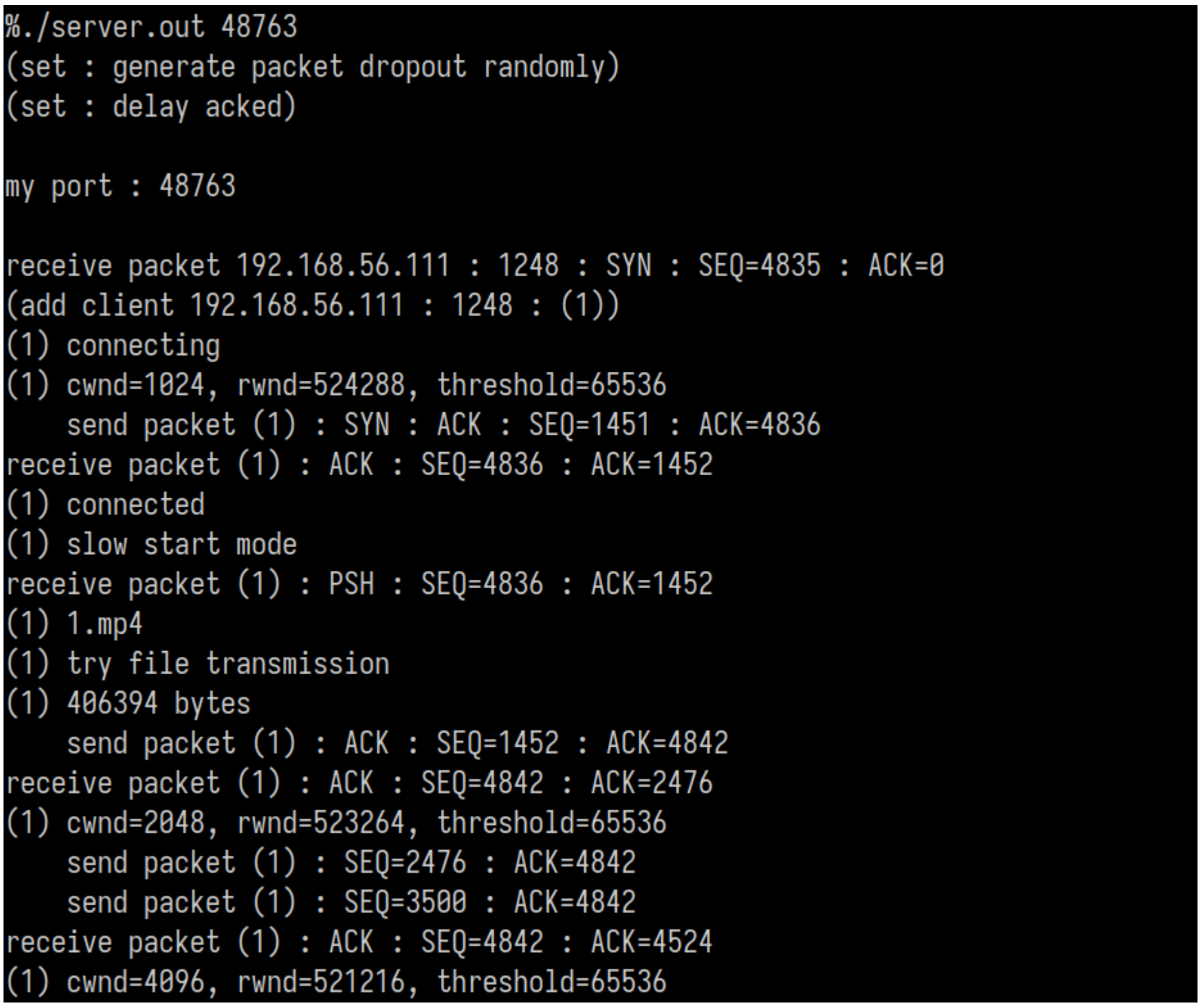
1. Including the previous steps' function.
2. Implement the delayed ACKs, you can wait up to 600ms for the next packet, or delay for two packets, then send an ACK packet to the server.
3. You don't have to print out which client the server is sending. (Or you can let only one client to connect to the server.)

Client:



```
%./client.out 192.168.56.110 48763 1.mp4
(set : don't show files' content)
(set : generate packet dropout randomly)
(set : delay acked)

server's ip address : 192.168.56.110
server's port       : 48763

(connecting)
    send packet : SYN : SEQ=4835 : ACK=0 (generated packet loss)
(timeout)
    send packet : SYN : SEQ=4835 : ACK=0
    receive packet : SYN : ACK : SEQ=1451 : ACK=4836
    send packet : ACK : SEQ=4836 : ACK=1452
(connected)
(requested tasks)
(task 1 : 1.mp4)
    send packet : PSH : SEQ=4836 : ACK=1452
    receive packet : ACK : SEQ=2476 : ACK=4842
    send packet : ACK : SEQ=4842 : ACK=2476
    receive packet : SEQ=2476 : ACK=4842 (delay acked)
    receive packet : SEQ=3500 : ACK=4842
    send packet : ACK : SEQ=4842 : ACK=4524
```

Server:

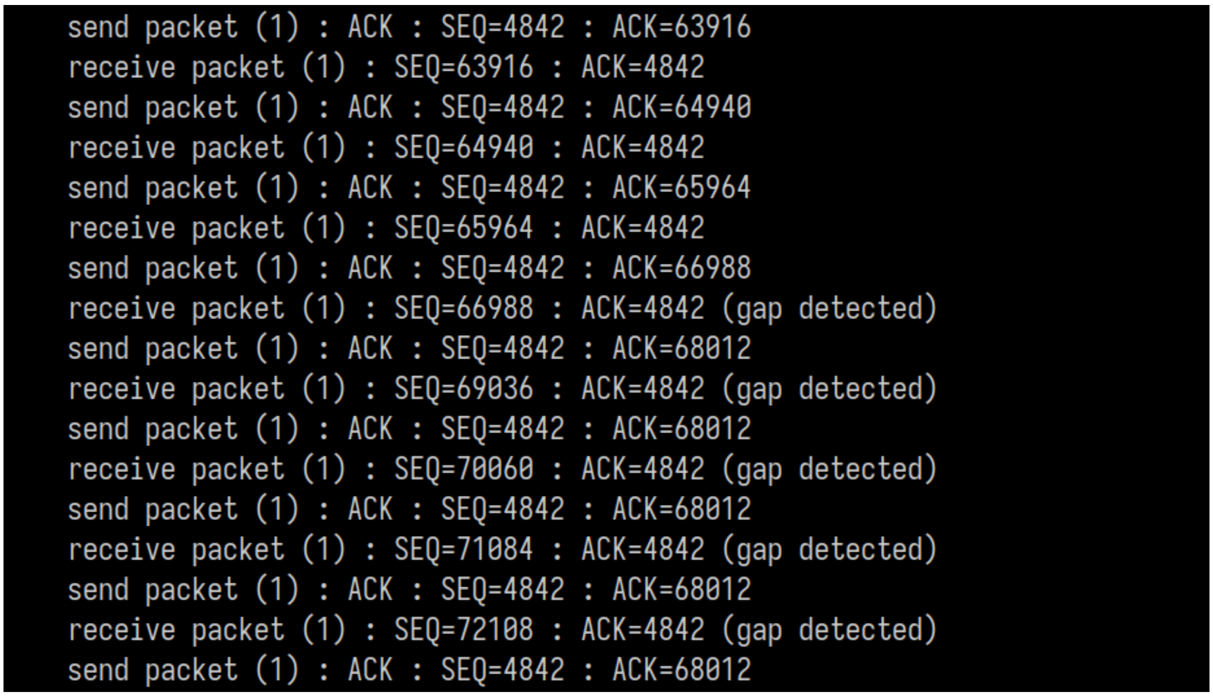


```
%./server.out 48763
(set : generate packet dropout randomly)
(set : delay acked)

my port : 48763

receive packet 192.168.56.111 : 1248 : SYN : SEQ=4835 : ACK=0
(add client 192.168.56.111 : 1248 : (1))
(1) connecting
(1) cwnd=1024, rwnd=524288, threshold=65536
    send packet (1) : SYN : ACK : SEQ=1451 : ACK=4836
receive packet (1) : ACK : SEQ=4836 : ACK=1452
(1) connected
(1) slow start mode
receive packet (1) : PSH : SEQ=4836 : ACK=1452
(1) 1.mp4
(1) try file transmission
(1) 406394 bytes
    send packet (1) : ACK : SEQ=1452 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=2476
(1) cwnd=2048, rwnd=523264, threshold=65536
    send packet (1) : SEQ=2476 : ACK=4842
    send packet (1) : SEQ=3500 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=4524
(1) cwnd=4096, rwnd=521216, threshold=65536
```

Step 5:

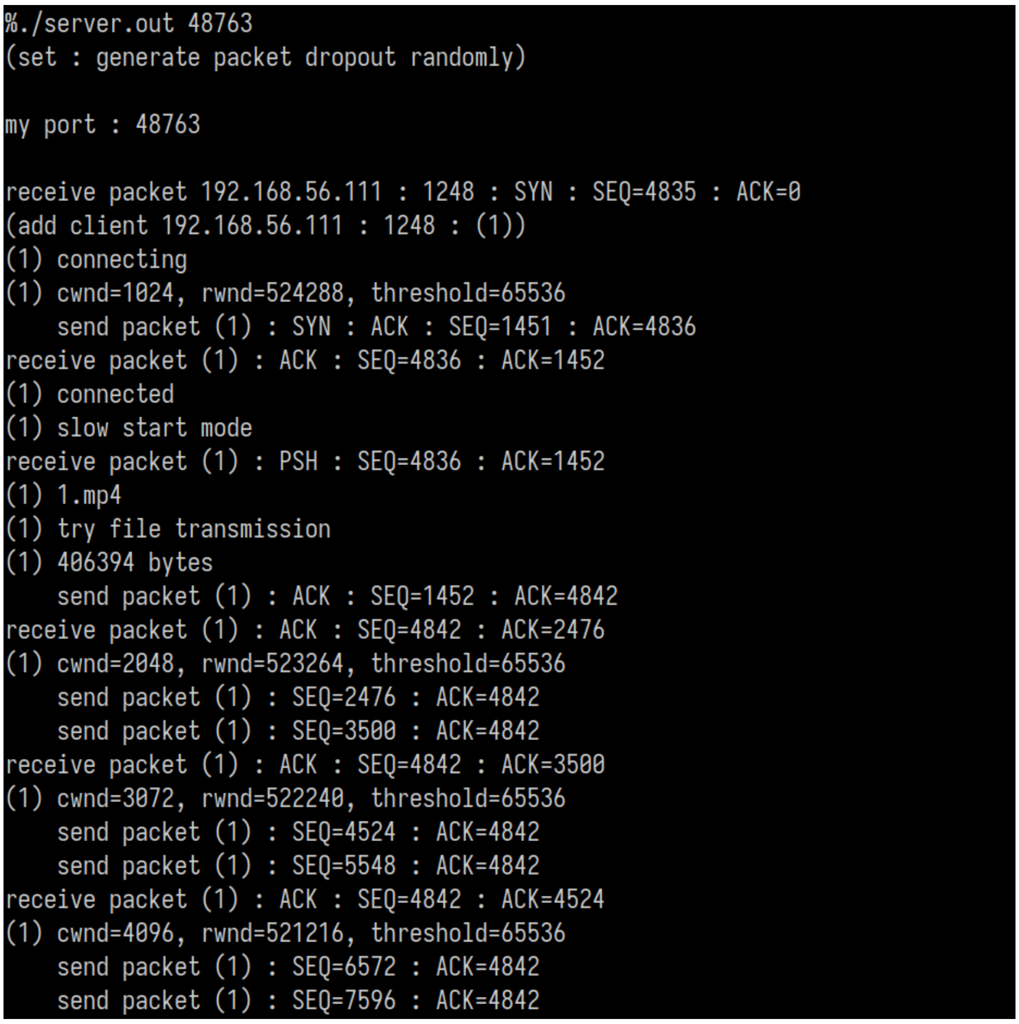
1. Including the previous steps’ function.
2. Implement congestion control including slow start and congestion avoidance.
3. You can ignore the mechanism of delayed ACK to implement this step in order to check the received packets.

Client:



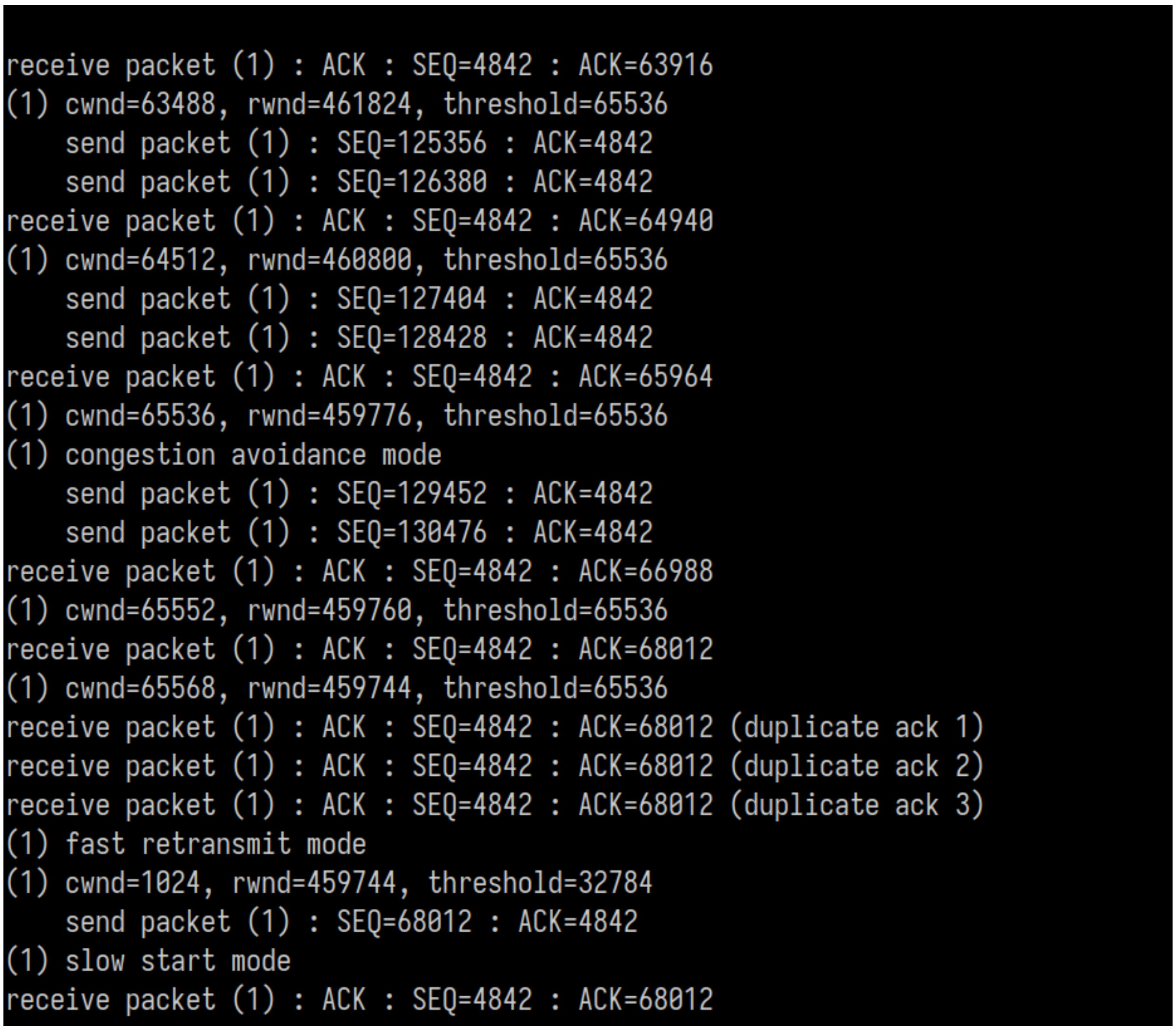
```
send packet (1) : ACK : SEQ=4842 : ACK=63916
receive packet (1) : SEQ=63916 : ACK=4842
send packet (1) : ACK : SEQ=4842 : ACK=64940
receive packet (1) : SEQ=64940 : ACK=4842
send packet (1) : ACK : SEQ=4842 : ACK=65964
receive packet (1) : SEQ=65964 : ACK=4842
send packet (1) : ACK : SEQ=4842 : ACK=66988
receive packet (1) : SEQ=66988 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=68012
receive packet (1) : SEQ=69036 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=68012
receive packet (1) : SEQ=70060 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=68012
receive packet (1) : SEQ=71084 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=68012
receive packet (1) : SEQ=72108 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=68012
```

Server (slow start):



```
%./server.out 48763
(set : generate packet dropout randomly)

my port : 48763

receive packet 192.168.56.111 : 1248 : SYN : SEQ=4835 : ACK=0
(add client 192.168.56.111 : 1248 : (1))
(1) connecting
(1) cwnd=1024, rwnd=524288, threshold=65536
    send packet (1) : SYN : ACK : SEQ=1451 : ACK=4836
receive packet (1) : ACK : SEQ=4836 : ACK=1452
(1) connected
(1) slow start mode
receive packet (1) : PSH : SEQ=4836 : ACK=1452
(1) 1.mp4
(1) try file transmission
(1) 406394 bytes
    send packet (1) : ACK : SEQ=1452 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=2476
(1) cwnd=2048, rwnd=523264, threshold=65536
    send packet (1) : SEQ=2476 : ACK=4842
    send packet (1) : SEQ=3500 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=3500
(1) cwnd=3072, rwnd=522240, threshold=65536
    send packet (1) : SEQ=4524 : ACK=4842
    send packet (1) : SEQ=5548 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=4524
(1) cwnd=4096, rwnd=521216, threshold=65536
    send packet (1) : SEQ=6572 : ACK=4842
    send packet (1) : SEQ=7596 : ACK=4842
```

Server (congestion avoidance):

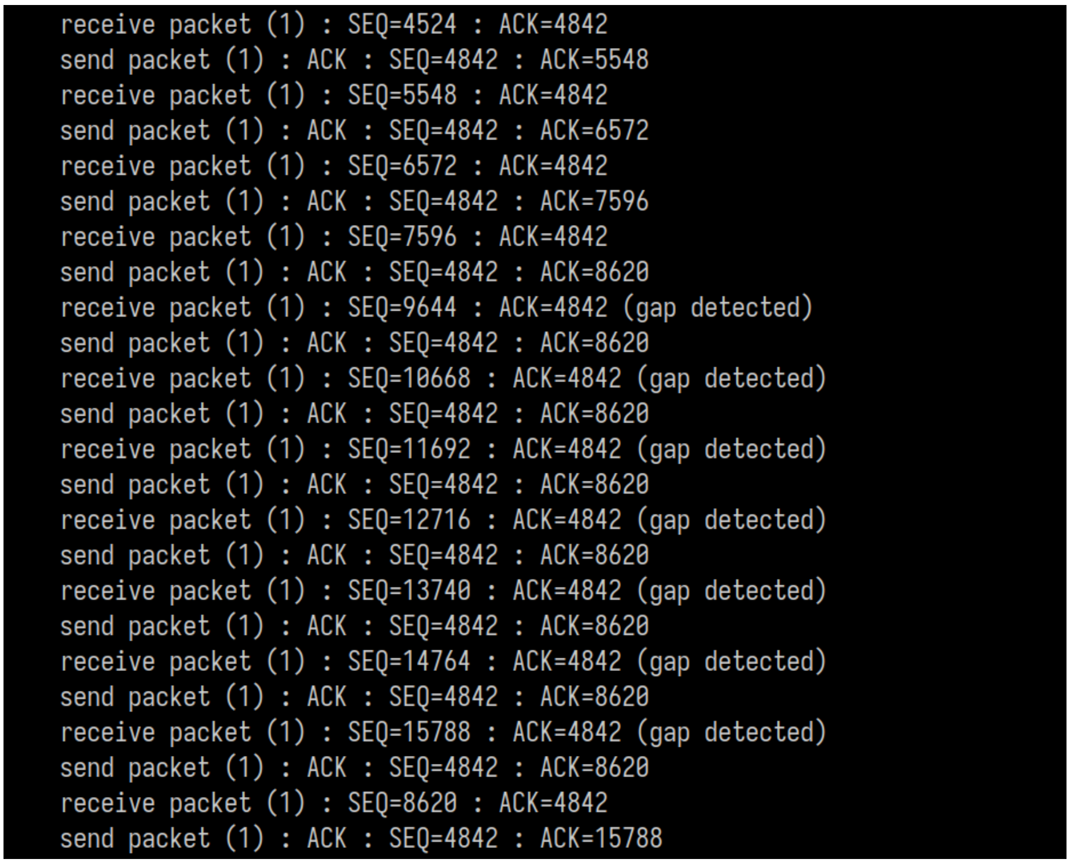


```
receive packet (1) : ACK : SEQ=4842 : ACK=63916
(1) cwnd=63488, rwnd=461824, threshold=65536
    send packet (1) : SEQ=125356 : ACK=4842
    send packet (1) : SEQ=126380 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=64940
(1) cwnd=64512, rwnd=460800, threshold=65536
    send packet (1) : SEQ=127404 : ACK=4842
    send packet (1) : SEQ=128428 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=65964
(1) cwnd=65536, rwnd=459776, threshold=65536
(1) congestion avoidance mode
    send packet (1) : SEQ=129452 : ACK=4842
    send packet (1) : SEQ=130476 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=66988
(1) cwnd=65552, rwnd=459760, threshold=65536
receive packet (1) : ACK : SEQ=4842 : ACK=68012
(1) cwnd=65568, rwnd=459744, threshold=65536
receive packet (1) : ACK : SEQ=4842 : ACK=68012 (duplicate ack 1)
receive packet (1) : ACK : SEQ=4842 : ACK=68012 (duplicate ack 2)
receive packet (1) : ACK : SEQ=4842 : ACK=68012 (duplicate ack 3)
(1) fast retransmit mode
(1) cwnd=1024, rwnd=459744, threshold=32784
    send packet (1) : SEQ=68012 : ACK=4842
(1) slow start mode
receive packet (1) : ACK : SEQ=4842 : ACK=68012
```

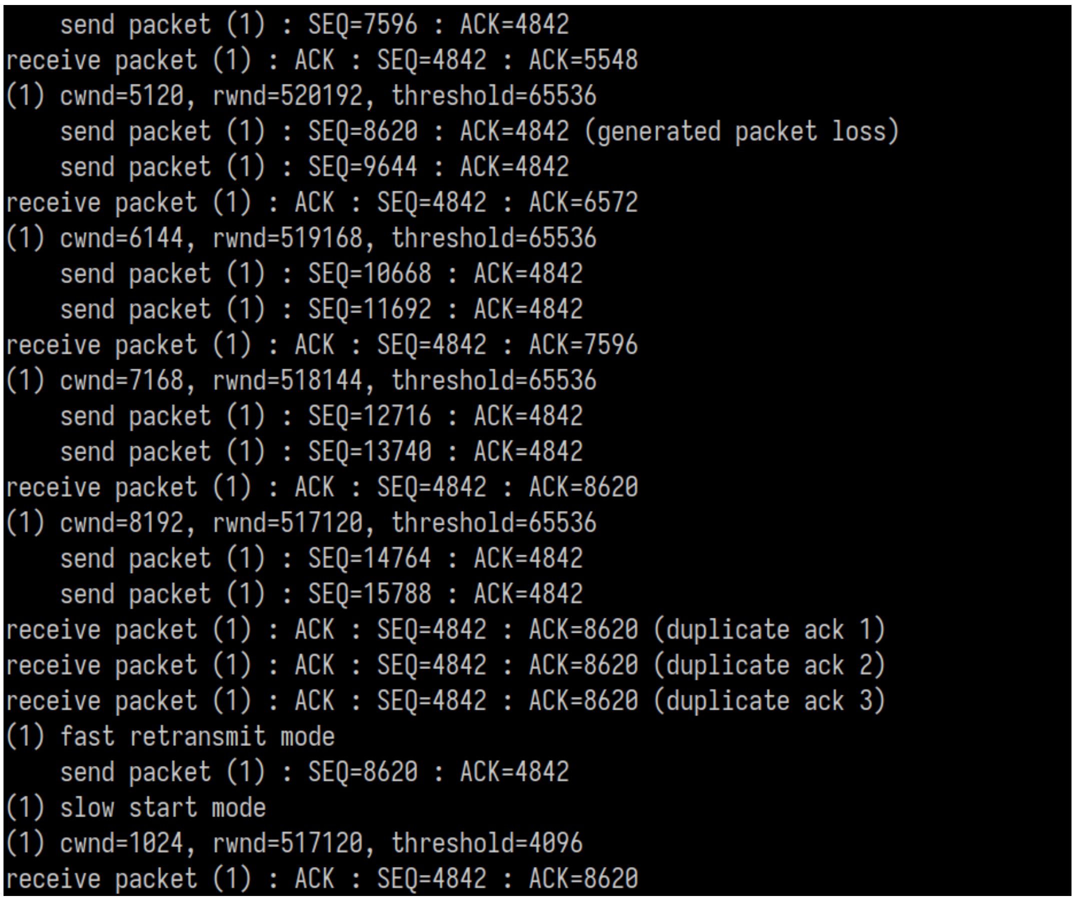
Step 6:

1. Including the previous steps' function.
2. Implement the mechanism of fast retransmit. (Tahoe)
3. You need to create a packet loss at the packet which starts at 8192 bytes 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 received packets.

Client:


```
    receive packet (1) : SEQ=4524 : ACK=4842
    send packet (1) : ACK : SEQ=4842 : ACK=5548
    receive packet (1) : SEQ=5548 : ACK=4842
    send packet (1) : ACK : SEQ=4842 : ACK=6572
    receive packet (1) : SEQ=6572 : ACK=4842
    send packet (1) : ACK : SEQ=4842 : ACK=7596
    receive packet (1) : SEQ=7596 : ACK=4842
    send packet (1) : ACK : SEQ=4842 : ACK=8620
    receive packet (1) : SEQ=9644 : ACK=4842 (gap detected)
    send packet (1) : ACK : SEQ=4842 : ACK=8620
    receive packet (1) : SEQ=10668 : ACK=4842 (gap detected)
    send packet (1) : ACK : SEQ=4842 : ACK=8620
    receive packet (1) : SEQ=11692 : ACK=4842 (gap detected)
    send packet (1) : ACK : SEQ=4842 : ACK=8620
    receive packet (1) : SEQ=12716 : ACK=4842 (gap detected)
    send packet (1) : ACK : SEQ=4842 : ACK=8620
    receive packet (1) : SEQ=13740 : ACK=4842 (gap detected)
    send packet (1) : ACK : SEQ=4842 : ACK=8620
    receive packet (1) : SEQ=14764 : ACK=4842 (gap detected)
    send packet (1) : ACK : SEQ=4842 : ACK=8620
    receive packet (1) : SEQ=15788 : ACK=4842 (gap detected)
    send packet (1) : ACK : SEQ=4842 : ACK=8620
    receive packet (1) : SEQ=8620 : ACK=4842
    send packet (1) : ACK : SEQ=4842 : ACK=15788
```

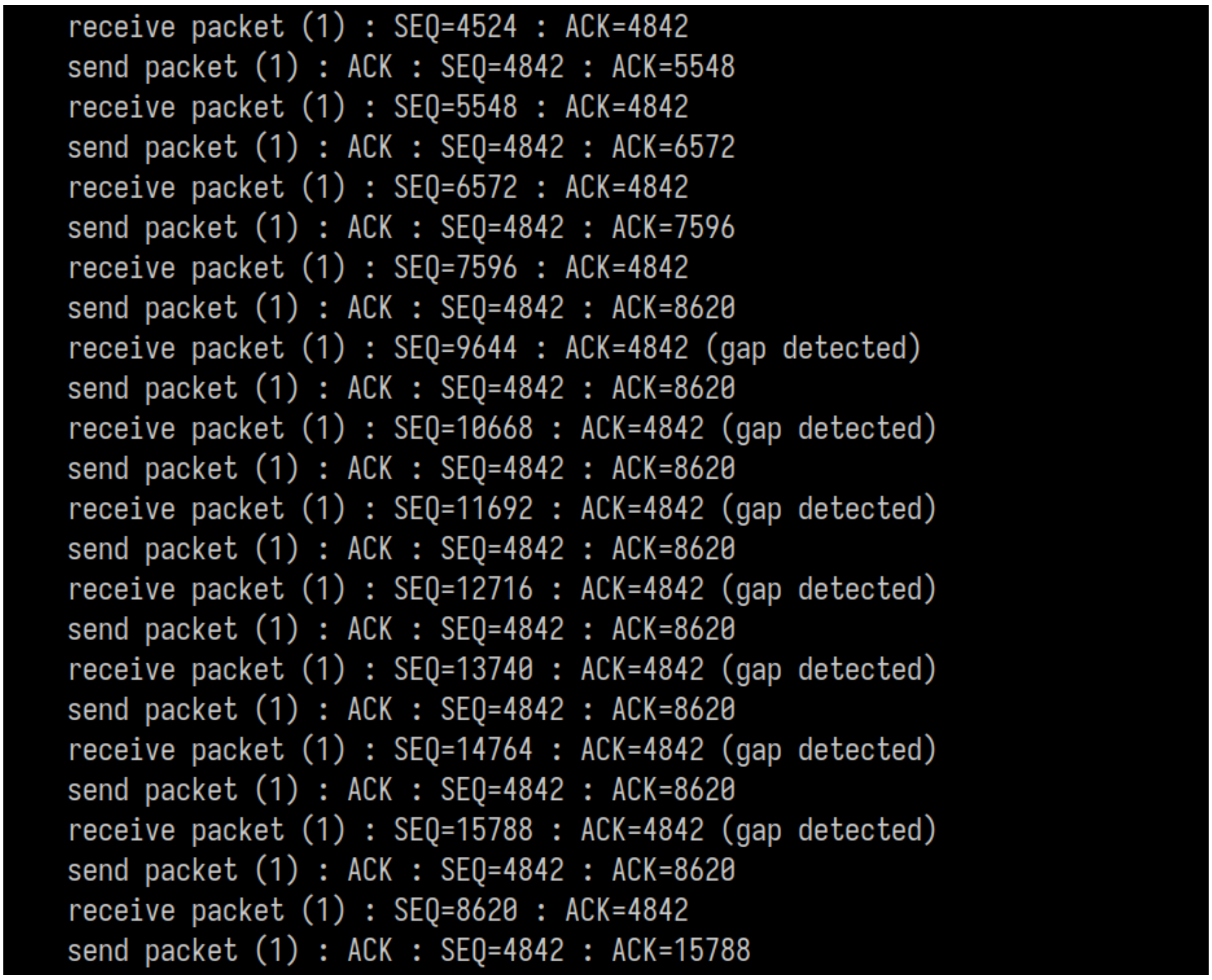
Server:


```
    send packet (1) : SEQ=7596 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=5548
(1) cwnd=5120, rwnd=520192, threshold=65536
    send packet (1) : SEQ=8620 : ACK=4842 (generated packet loss)
    send packet (1) : SEQ=9644 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=6572
(1) cwnd=6144, rwnd=519168, threshold=65536
    send packet (1) : SEQ=10668 : ACK=4842
    send packet (1) : SEQ=11692 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=7596
(1) cwnd=7168, rwnd=518144, threshold=65536
    send packet (1) : SEQ=12716 : ACK=4842
    send packet (1) : SEQ=13740 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=8620
(1) cwnd=8192, rwnd=517120, threshold=65536
    send packet (1) : SEQ=14764 : ACK=4842
    send packet (1) : SEQ=15788 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=8620 (duplicate ack 1)
receive packet (1) : ACK : SEQ=4842 : ACK=8620 (duplicate ack 2)
receive packet (1) : ACK : SEQ=4842 : ACK=8620 (duplicate ack 3)
(1) fast retransmit mode
    send packet (1) : SEQ=8620 : ACK=4842
(1) slow start mode
(1) cwnd=1024, rwnd=517120, threshold=4096
receive packet (1) : ACK : SEQ=4842 : ACK=8620
```

Step 7:

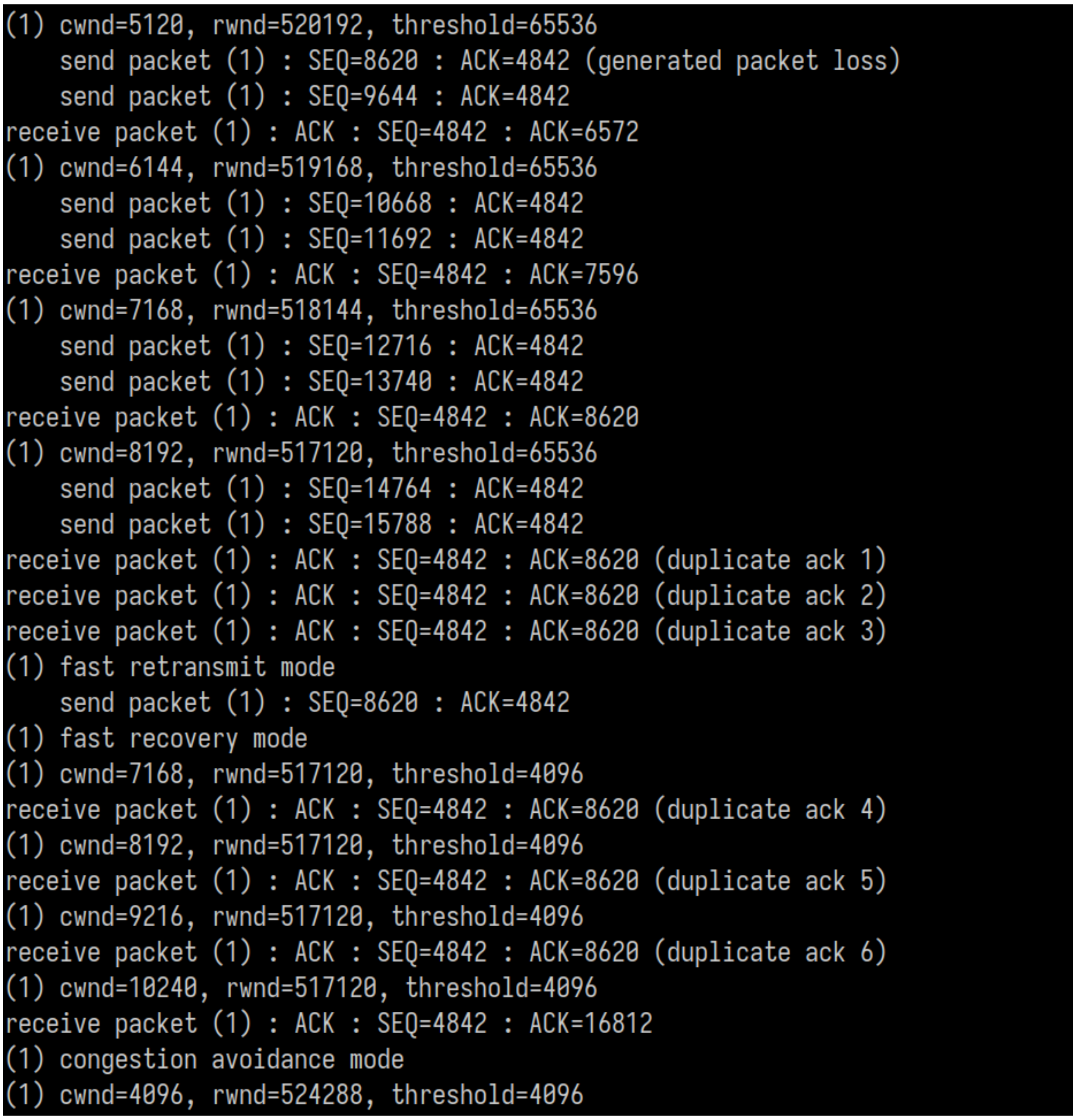
1. Including the previous steps' function.
2. Implement the mechanism of fast recovery. (TCP Reno)
3. You need to design a packet loss at byte 8192 to get duplicated ACKs, then the fast retransmit will be executed, and the state of fast recovery is entered.
4. You can ignore the mechanism of delayed ACK to implement this step in order to check the received packets.

Client:



```
receive packet (1) : SEQ=4524 : ACK=4842
send packet (1) : ACK : SEQ=4842 : ACK=5548
receive packet (1) : SEQ=5548 : ACK=4842
send packet (1) : ACK : SEQ=4842 : ACK=6572
receive packet (1) : SEQ=6572 : ACK=4842
send packet (1) : ACK : SEQ=4842 : ACK=7596
receive packet (1) : SEQ=7596 : ACK=4842
send packet (1) : ACK : SEQ=4842 : ACK=8620
receive packet (1) : SEQ=9644 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=8620
receive packet (1) : SEQ=10668 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=8620
receive packet (1) : SEQ=11692 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=8620
receive packet (1) : SEQ=12716 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=8620
receive packet (1) : SEQ=13740 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=8620
receive packet (1) : SEQ=14764 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=8620
receive packet (1) : SEQ=15788 : ACK=4842 (gap detected)
send packet (1) : ACK : SEQ=4842 : ACK=8620
receive packet (1) : SEQ=8620 : ACK=4842
send packet (1) : ACK : SEQ=4842 : ACK=15788
```

Server:



```
(1) cwnd=5120, rwnd=520192, threshold=65536
    send packet (1) : SEQ=8620 : ACK=4842 (generated packet loss)
    send packet (1) : SEQ=9644 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=6572
(1) cwnd=6144, rwnd=519168, threshold=65536
    send packet (1) : SEQ=10668 : ACK=4842
    send packet (1) : SEQ=11692 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=7596
(1) cwnd=7168, rwnd=518144, threshold=65536
    send packet (1) : SEQ=12716 : ACK=4842
    send packet (1) : SEQ=13740 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=8620
(1) cwnd=8192, rwnd=517120, threshold=65536
    send packet (1) : SEQ=14764 : ACK=4842
    send packet (1) : SEQ=15788 : ACK=4842
receive packet (1) : ACK : SEQ=4842 : ACK=8620 (duplicate ack 1)
receive packet (1) : ACK : SEQ=4842 : ACK=8620 (duplicate ack 2)
receive packet (1) : ACK : SEQ=4842 : ACK=8620 (duplicate ack 3)
(1) fast retransmit mode
    send packet (1) : SEQ=8620 : ACK=4842
(1) fast recovery mode
(1) cwnd=7168, rwnd=517120, threshold=4096
receive packet (1) : ACK : SEQ=4842 : ACK=8620 (duplicate ack 4)
(1) cwnd=8192, rwnd=517120, threshold=4096
receive packet (1) : ACK : SEQ=4842 : ACK=8620 (duplicate ack 5)
(1) cwnd=9216, rwnd=517120, threshold=4096
receive packet (1) : ACK : SEQ=4842 : ACK=8620 (duplicate ack 6)
(1) cwnd=10240, rwnd=517120, threshold=4096
receive packet (1) : ACK : SEQ=4842 : ACK=16812
(1) congestion avoidance mode
(1) cwnd=4096, rwnd=524288, threshold=4096
```

Step 8:

1. Including the previous steps' function.
2. Now increase the number of clients to 20. The server side has to implement some prevention mechanisms to handle simultaneous requests from clients.