<2018 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 contro etc. Second, you have to implement TCP in application layer and call UDP to transmit TCP packets.

**Rules**

1. Run your program on Ubuntu 16.04 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

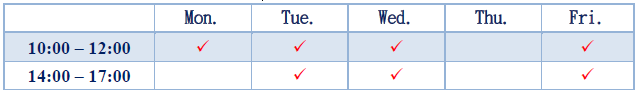
**Deadline**

You should upload your homework to the Cyber University before 2018/06/22 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: 2017/06/23 17:00



**Description**

You have to obey the following schema:

The TCP segment structure

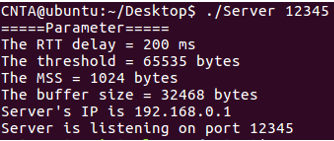
The initial sequence number should be set randomly (1~10000).

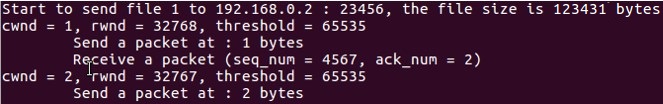
Step 1:

1. Set the parameters including RTT (200 ms), MSS (1024 bytes ), threshold (65535) and the receiver’s buffer size (32KB), etc.

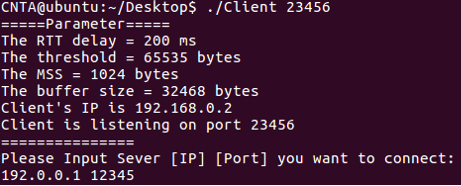
2. You also have to implement the data transmission (You need to ensure the data that can transmit from server to client, and ACK packet transmit from client to server).

Server:





Client:



Step 2:

1. Including the previous step’s function.

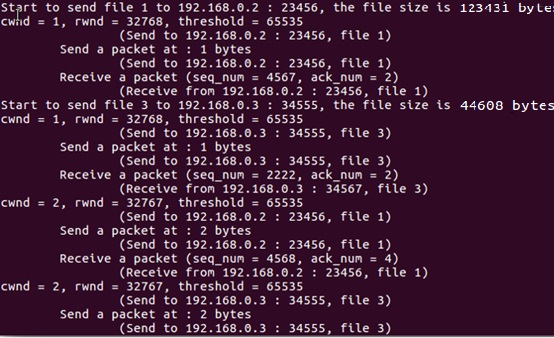
2.First, you have to transmit the video files in this step. A client should request multiple files in one packet. The server should send the data to multiple clients in the same time. (You can use fork or thread or select().)

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.)

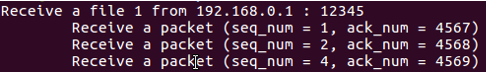
4. Second, you also have to implement the data transmission (You need to ensure the data that can transmit from server to client, and ACK packet transmit from client to server).

5. You have to print out which client the server is sending to and which file is sent in this step.

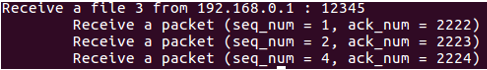
Server:



Client A:



Client B:



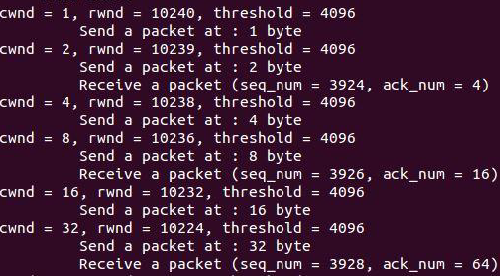
Step 3:

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:



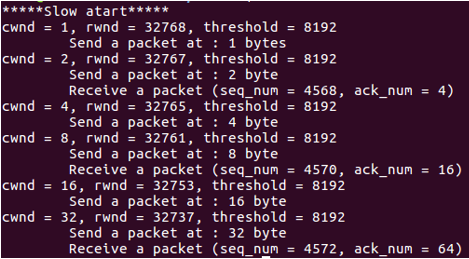
Step 4:

1. Including the previous step’s function.

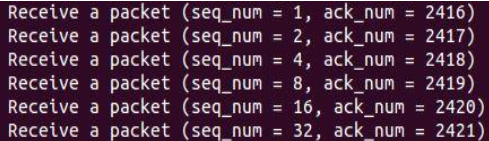
2. Implement the congestion control including slow start and congestion avoidance.

3. You need to reset the threshold as 8192 in order to enter the status of congestion avoidance.

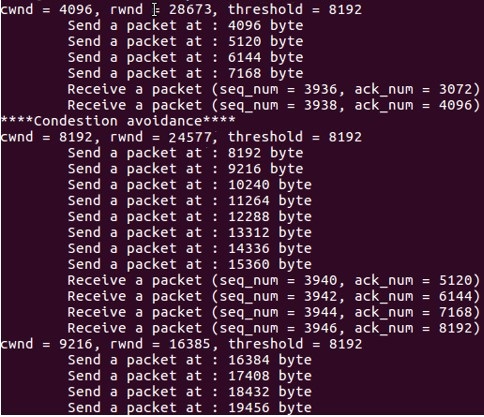
Server (slow start):



Client (slow start):



Server (congestion avoidance):



Step 5:

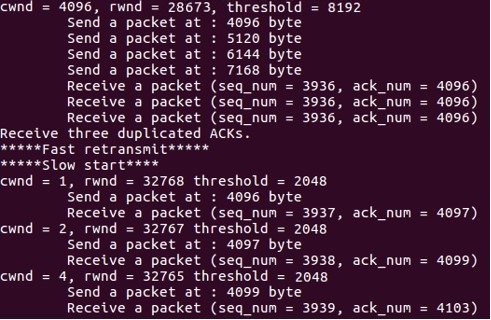
1. Including the previous step’s function.

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

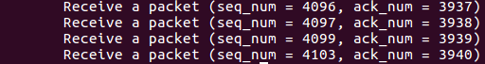
3. You need to design a packet loss at byte 4096 to get duplicated ACKs, then the fast retransmit will execute.

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

Server :



Client:



Step 6:

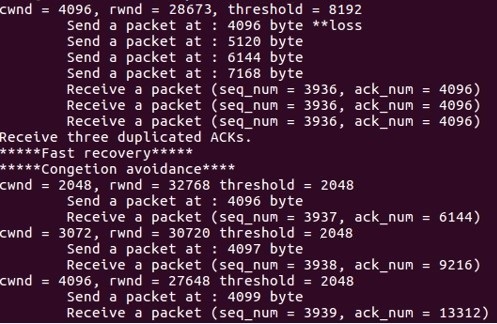
1. Including the previous step’s function.

2. Implement the mechanism of fast recovery. (TCP Reno)

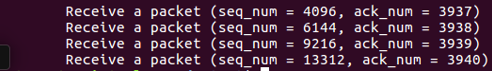
3. You need to design a packet loss at byte 2048 to get duplicated ACKs, then the fast retransmit will execute, and enter the state of fast recovery.

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

Server:



Client :



Step 7:

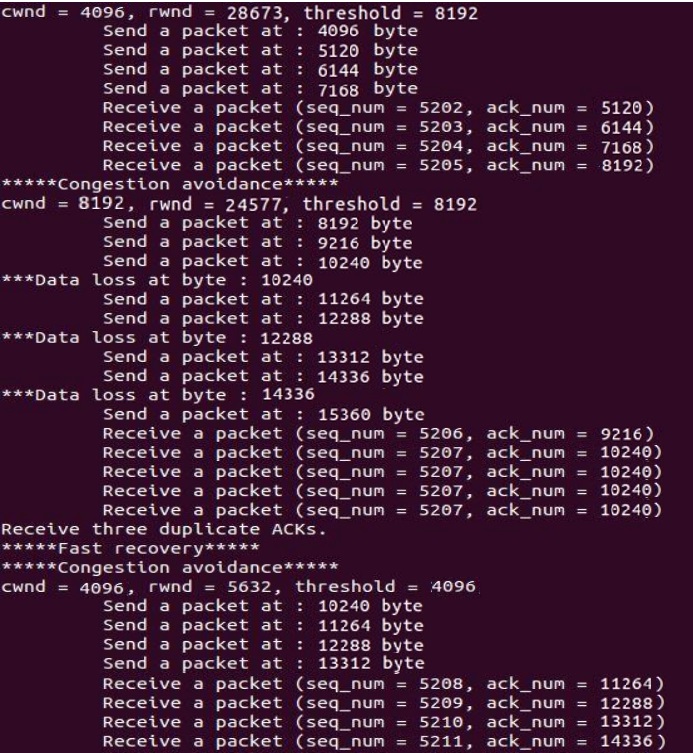
1. Including the previous step’s function.

2. Implement the mechanism of TCP SACK, and using three blocks in this step.

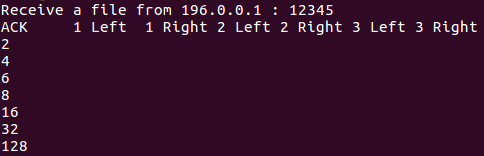
3. You need to design a packet loss at byte 10240, 12288 and 14336 to create three SACK blocks.

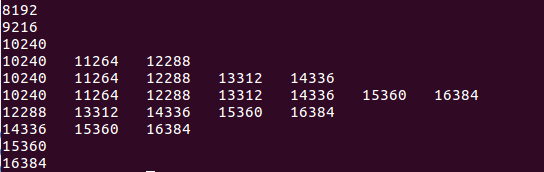
4. You can ignore the mechanism of delayed ACK to implement this step in order to check the receive packets.

Server :



Client (the output format of client is different, print each ACK packet) :





Step 8:

The link becomes unstable and there are 3% of loss rates. The loss rates are generated randomly. There are two connections in each direction. The direction from the server to each client has a maximum transmission rate of 200 kbps and RTT = 100ms, and the other direction from each client to the server has a maximum transmission rate of 50 kbps and RTT = 200ms. Only TCP SACK is simulated.

Step 9:

The link becomes unstable and there are 0.01% of loss rates. The loss rates are generated randomly. There are two connections in each direction. The direction from the server to each client has a maximum transmission rate of 300 kbps and RTT = 50ms, and the other direction from each client to the server has a maximum transmission rate of 50 kbps and RTT = 300ms. Only TCP SACK is simulated.