<2017 Computer Network Homework>

Motivation

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

Rules

1. Run your program on Ubuntu 14.04 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".
   Ex: B043040000_王小明.zip

Deadline

You should upload your homework to the Cyber University before 2017/06/20 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: 2017/06/23 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).
- You have to create data randomly using (0~9, a~z, A~Z), and the data size is 10240 bytes.

Step 1:

1. Set the parameters including RTT (200 ms), MSS (512 or 536 or 1500 bytes), threshold (65535) and the receiver's buffer size (10KB=10240 bytes), etc.
2. First, you should implement the three/four way handshake.
3. 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).
4. Final, add the mechanism of flow control (sender's congestion window and receiver's buffer window).

Node A side (parameter): init & connect to Node B

```
cpeter@ubuntu:~/Desktop/TCP_Homework_TA/step/TA/A$ ./Node 12345
=====Parameter=====
The RTT delay = 200 ms
The threshold = 65535 bytes
The MSS = 512 bytes
The buffer size = 10240 bytes
Server's IP is 192.168.0.1
Server is listening on port 12345
===============
Listening for Node...
(ClientFunction)Please Input Node [IP] [Port] you want to connec
192.168.0.2 23456
```

Other Node B

```
cpeter@ubuntu:~/Desktop/TCP_Homework_TA/step/TA/B$ ./Node 23456
=====Parameter=====
The RTT delay = 200 ms
The threshold = 65535 bytes
The MSS = 512 bytes
The buffer size = 10240 bytes
Server's IP is 192.168.0.2
Server is listening on port 23456
```

Node A side (three-way handshake):

```
=====Start the three-way handshake=====
Send a packet(SYN) to 192.168.0.2 : 23456
Receive a packet(SYN/ACK) from 192.168.0.2 : 23456
         Receive a packet (seq_num = 7421, ack_num = 8109)
Send a packet(ACK) to 192.168.0.2 : 23456
=====Complete the three-way handshake=====
```

Node B side (three-way handshake):

```
=====Start the three-way handshake=====
Receive a packet(SYN) from 192.168.0.1 : 10260
         Receive a packet (seq_num = 8108, ack_num = 0)
Send a packet(SYN/ACK) to 192.168.0.1 : 10260
Receive a packet(ACK) from 192.168.0.1 : 10260
         Receive a packet (seq_num = 8109, ack_num = 7422)
=====Complete the three-way handshake=====
```

Node A side (data transmission from Node B):

```
=====Complete the three-way handshake=====
Receive a file from 192.168.0.2 : 23456
         Receive a packet (seq_num = 1, ack_num = 8110)
         Receive a packet (seq_num = 2, ack_num = 8111)
         Receive a packet (seq_num = 4, ack_num = 8112)
```

Node B side (data transmission to Node A):

```
=====Complete the three-way handshake=====
Start to send the file, the file size is 10240 bytes.
*****Slow start*****
cwnd = 1, rwnd = 10240, threshold = 65535
         Send a packet at : 1 byte
         Receive a packet (seq_num = 8110, ack_num = 2)
cwnd = 2, rwnd = 10239, threshold = 65535
         Send a packet at : 2 byte
```

Node A side (four-way handshake):

```
=====Start the four-way handshake=====
Receive a packet(FIN) from 192.168.0.2 : 23456
        Receive a packet (seq_num = 9728, ack_num = 8138)
Send a packet(ACK) to 192.168.0.2 : 23456
Send a packet(FIN) to 192.168.0.2 : 23456
Receive a packet(ACK) from 192.168.0.2 : 23456
        Receive a packet (seq_num = 9729, ack_num = 8139)
=====Complete the four-way handshake=====
(ClientFunction)Please Input Node [IP] [Port] you want to connect to
```

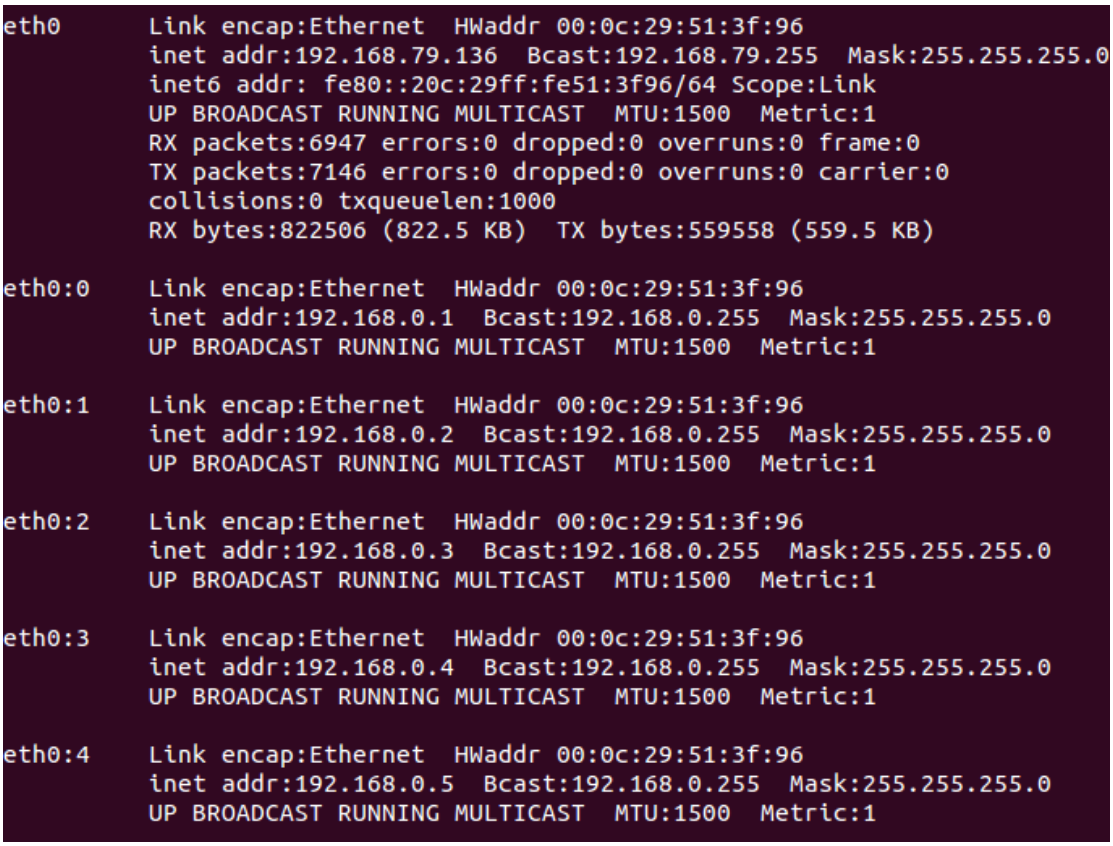
Node B side (four-way handshake):

```
=====Start the four-way handshake=====
Send a packet(FIN) to 192.168.0.1 : 10260
Receive a packet(ACK) from 192.168.0.1 : 10260
         Receive a packet (seq_num = 8138, ack_num = 9729)
Receive a packet(FIN) from 192.168.0.1 : 10260
         Receive a packet (seq_num = 8138, ack_num = 9729)
Send a packet(ACK) to 192.168.0.1 : 10260
=====Complete the four-way handshake=====
Listening for Node...
```

Step 2:

1. Including the previous step's function.
2. Implements the NAT function in client side or you can build a router as NAT.
3. The client has to transform the IP from one to another by mapping table after server transmit the packet to client.
4. You need to design a mapping table and create two virtual IPs at least (Ex: You can type the command "**ifconfig eth0:0 192.168.0.1 up**" in terminal to create a virtual IP).
5. Additional, you have to print the mapping table in the client side (or NAT router) and show the virtual IPs using command "ifconfig" in terminal.

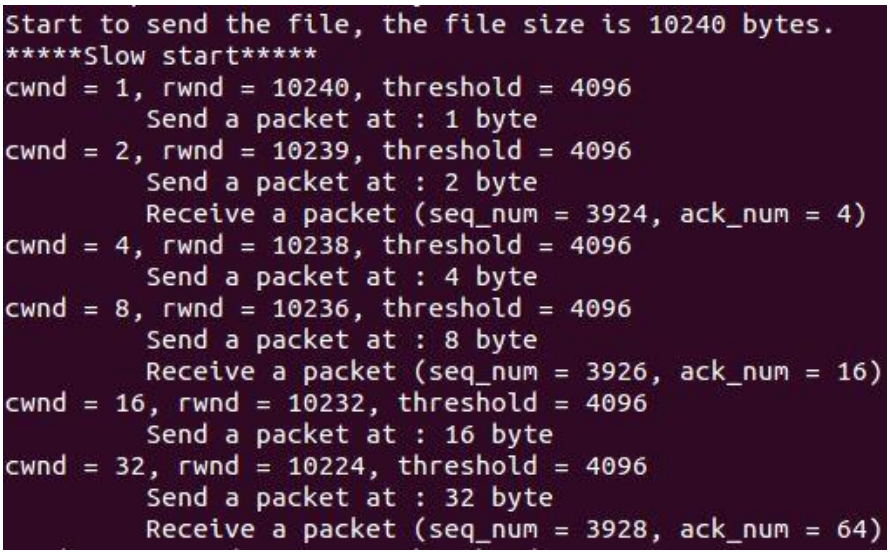
   Print the virtual IP's information (if need).



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.

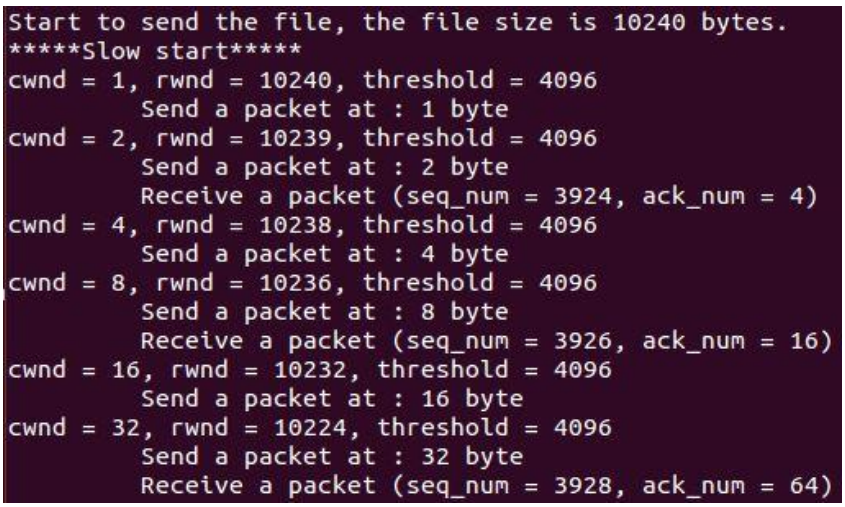
Node B side:



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 4096 in order to enter the status of congestion avoidance.

Server side (slow start):



Node A side (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)
```

Node B side (congestion avoidance):

```
cwnd = 2048, rwnd = 9216, threshold = 4096
        Send a packet at : 2048 byte
        Send a packet at : 2560 byte
        Send a packet at : 3072 byte
        Send a packet at : 3584 byte
        Receive a packet (seq_num = 3936, ack_num = 3072)
        Receive a packet (seq_num = 3938, ack_num = 4096)
*****Congestion avoidance*****
cwnd = 4096, rwnd = 8192, threshold = 4096
        Send a packet at : 4096 byte
        Send a packet at : 4608 byte
        Send a packet at : 5120 byte
        Send a packet at : 5632 byte
        Send a packet at : 6144 byte
        Send a packet at : 6656 byte
        Send a packet at : 7168 byte
        Send a packet at : 7680 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 = 4608, rwnd = 6144, threshold = 4096
        Send a packet at : 8192 byte
        Send a packet at : 8704 byte
        Send a packet at : 9216 byte
        Send a packet at : 9728 byte
        Receive a packet (seq_num = 3948, ack_num = 9216)
        Receive a packet (seq_num = 3950, ack_num = 10240)
The file transmission is finish.
```

Node A side (congestion avoidance):

```
Receive a packet (seq_num = 4096, ack_num = 2432)
Receive a packet (seq_num = 4608, ack_num = 2433)
Receive a packet (seq_num = 5120, ack_num = 2434)
Receive a packet (seq_num = 5632, ack_num = 2435)
Receive a packet (seq_num = 6144, ack_num = 2436)
Receive a packet (seq_num = 6656, ack_num = 2437)
Receive a packet (seq_num = 7168, ack_num = 2438)
Receive a packet (seq_num = 7680, ack_num = 2439)
Receive a packet (seq_num = 8192, ack_num = 2440)
Receive a packet (seq_num = 8704, ack_num = 2441)
Receive a packet (seq_num = 9216, ack_num = 2442)
Receive a packet (seq_num = 9728, ack_num = 2443)
Receive a packet (seq_num = 9728, ack_num = 2444)
```

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 2048 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.

Node B side:

```
cwnd = 2048, rwnd = 9216, threshold = 4096
        Send a packet at : 2048 byte
***Data loss at byte : 2048
        Send a packet at : 2560 byte
        Send a packet at : 3072 byte
        Send a packet at : 3584 byte
        Receive a packet (seq_num = 4984, ack_num = 2048)
        Receive a packet (seq_num = 4984, ack_num = 2048)
        Receive a packet (seq_num = 4984, ack_num = 2048)
Receive three duplicate ACKs.
*****Fast retransmit*****
*****Slow start*****
cwnd = 1, rwnd = 10240, threshold = 1024
        Send a packet at : 2048 byte
        Receive a packet (seq_num = 4985, ack_num = 2049)
cwnd = 2, rwnd = 10239, threshold = 1024
        Send a packet at : 2049 byte
        Receive a packet (seq_num = 4986, ack_num = 2051)
cwnd = 4, rwnd = 10238, threshold = 1024
        Send a packet at : 2051 byte
        Receive a packet (seq_num = 4987, ack_num = 2055)
cwnd = 8, rwnd = 10236, threshold = 1024
        Send a packet at : 2055 byte
        Receive a packet (seq_num = 4988, ack_num = 2063)
```

Node A side:

```
Receive a packet (seq_num = 2048, ack_num = 4985)
Receive a packet (seq_num = 2049, ack_num = 4986)
Receive a packet (seq_num = 2051, ack_num = 4987)
Receive a packet (seq_num = 2055, ack_num = 4988)
Receive a packet (seq_num = 2063, ack_num = 4989)
```

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.

Node B side:

```
cwnd = 2048, rwnd = 9216, threshold = 4096
        Send a packet at : 2048 byte
***Data loss at byte : 2048
        Send a packet at : 2560 byte
        Send a packet at : 3072 byte
        Send a packet at : 3584 byte
        Receive a packet (seq_num = 251, ack_num = 2048)
        Receive a packet (seq_num = 251, ack_num = 2048)
        Receive a packet (seq_num = 251, ack_num = 2048)
Receive three duplicate ACKs.
*****Fast recovery*****
*****Congestion avoidance*****
cwnd = 1024, rwnd = 10240, threshold = 1024
        Send a packet at : 2048 byte
        Send a packet at : 2560 byte
        Receive a packet (seq_num = 252, ack_num = 2560)
        Receive a packet (seq_num = 253, ack_num = 3072)
cwnd = 1536, rwnd = 9216, threshold = 1024
        Send a packet at : 3072 byte
        Send a packet at : 3584 byte
        Send a packet at : 4096 byte
        Receive a packet (seq_num = 254, ack_num = 3584)
        Receive a packet (seq_num = 255, ack_num = 4096)
        Receive a packet (seq_num = 256, ack_num = 4608)
```

Node A side:

```
Receive a packet (seq_num = 2048, ack_num = 252)
Receive a packet (seq_num = 2560, ack_num = 253)
Receive a packet (seq_num = 3072, ack_num = 254)
Receive a packet (seq_num = 3584, ack_num = 255)
Receive a packet (seq_num = 4096, ack_num = 256)
```

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 5120, 6144 and 7168 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.

Node B side:

```
cwnd = 2048, rwnd = 8704, threshold = 4096
         Send a packet at : 2048 byte
         Send a packet at : 2560 byte
         Send a packet at : 3072 byte
         Send a packet at : 3584 byte
         Receive a packet (seq_num = 5202, ack_num = 2560)
         Receive a packet (seq_num = 5203, ack_num = 3072)
         Receive a packet (seq_num = 5204, ack_num = 3584)
         Receive a packet (seq_num = 5205, ack_num = 4096)
*****Congestion avoidance*****
cwnd = 4096, rwnd = 6656, threshold = 4096
         Send a packet at : 4096 byte
         Send a packet at : 4608 byte
         Send a packet at : 5120 byte
***Data loss at byte : 5120
         Send a packet at : 5632 byte
         Send a packet at : 6144 byte
***Data loss at byte : 6144
         Send a packet at : 6656 byte
         Send a packet at : 7168 byte
***Data loss at byte : 7168
         Send a packet at : 7680 byte
         Receive a packet (seq_num = 5206, ack_num = 4608)
         Receive a packet (seq_num = 5207, ack_num = 5120)
         Receive a packet (seq_num = 5207, ack_num = 5120)
         Receive a packet (seq_num = 5207, ack_num = 5120)
         Receive a packet (seq_num = 5207, ack_num = 5120)
Receive three duplicate ACKs.
*****Fast recovery*****
*****Congestion avoidance*****
cwnd = 2560, rwnd = 5632, threshold = 2048
         Send a packet at : 5120 byte
         Send a packet at : 6144 byte
         Send a packet at : 7168 byte
         Send a packet at : 8192 byte
         Send a packet at : 8704 byte
         Receive a packet (seq_num = 5208, ack_num = 6144)
         Receive a packet (seq_num = 5209, ack_num = 7168)
         Receive a packet (seq_num = 5210, ack_num = 8192)
         Receive a packet (seq_num = 5211, ack_num = 8704)
         Receive a packet (seq_num = 5212, ack_num = 9216)
```

Node A side (the output format of client is different, print each ACK packet):

```
=====Complete the three-way handshake=====
Receive a file from 140.117.169.51 : 10250
ACK     1 Left  1 Right 2 Left  2 Right 3 Left  3 Right
2
4
8
16
32
64
128
256
512
1024
1536
2048
2560
3072
3584
4096
4608
5120
5120    5632    6144
5120    5632    6144    6656    7168
5120    5632    6144    6656    7168    7680    8192
6144    6656    7168    7680    8192
7168    7680    8192
8192
8704
9216
9728
```

Step 8:

1. Including the previous step's function.
2. Implement the TCP Reno, if the channel speed is lower than 95% of original speed, you need to discard the packet and retransmit the packet.
3. Implement multi-connections in each direction, and transmission rate limit
4. There is no strict output format, you just show your result in demo.