# <2023 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 22.04** platform.
2. Do not copy homework from your classmates or seniors, etc. If TAs find the situation, any participants will get a percent grade of **ZERO**.
3. You have to deeply understand what your program does because TAs will ask you the **concept** of your code.
4. If you have any questions, you can send an email or come to **EC5008** (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 pictures of your program's output results at every step.
7. In each step, you can write a new program, respectively (but the program has to include the function of the previous step).
8. The filename you upload should be "**StudentID_Name.zip**". Ex: B093040000_王小明.zip.
9. **輸出格式僅供參考，實際輸出結果請依題目需求呈現。**

---

## Deadline

You should upload your homework to the **Cyber University** before **2023/06/07 23:59**.
If you do not submit your assignment on time, you will get a percent grade of **ZERO**.

---

## Demo

The following figure shows the time you can come for a demo. You can mail TA for reservation.
**Demo deadline: 2023/06/09 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 be able to transmit a file under a subnetwork.

---

**Step 1:**

1. Set the parameters including RTT (20 ms), MSS (1 Kbytes), threshold (64 Kbytes), and the receiver's buffer size (512 Kbytes), etc.
2. You have to transmit the video files, perform mathematical calculations including power and square root, and perform DNS functions in this step. A client could request a single job or multiple jobs in one command. The server should send the video file, the results of DNS, and the results of mathematical calculations 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 sends the DNS packet with this domain name to the DNS server (ex: 8.8.8.8) and gets the result of the IP from the DNS server. Finally, the server sends the result to the client and the client should show the result on screen.
   (ex: You input "google.com" and then you will get the result as 172.217.160.110)
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. You have to print out the status of the server and clients. For example, for the server, which clients the server sends to and which files the server receives in this step.

Client:

```
harry@harry-VirtualBox:~/Desktop/123/Step123/Client$ ./client
Server's IP address: 127.0.0.1
Server's port: 1234
=====Start the three-way handshake=====
Send a packet(SYN) to 127.0.0.1 : 1234
        Send a packet at : 1 byte
Receive a packet(SYN/ACK) from 127.0.0.1 : 1235
        Receive a packet (seq_num = 8121, ack_num = 1226)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 1025 byte
=====Complete the three-way handshake=====
Enter command: dns google.com
Send a command.
        Send a packet at : 2049 byte
Finding IP address of google.com by 127.0.0.1 : 1235
        Receive a packet (seq_num = 9145, ack_num = 1227)
Result: 142.251.43.14
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 3073 byte
Finish finding
google.com
Enter command:
```

Server:

```
harry@harry-VirtualBox:~/Desktop/123/Step123/Server$ ./server
        Receive a packet (seq_num = 1225, ack_num = 0)
cwnd = 1, rwnd = 524288, threshold = 65536
        Send a packet at : 1 byte
        Receive a packet (seq_num = 1226, ack_num = 8122)
>>Waiting command<<
        Receive a packet (seq_num = 1226, ack_num = 8122)
dns google.com
==Start finding: google.com to 127.0.0.1 : 53273
cwnd = 2, rwnd = 524288, threshold = 65536
        Send a packet at : 1025 byte
        Receive a packet (seq_num = 1227, ack_num = 9146)
==Finish finding==
>>Waiting command<<
```

**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:

```
Server's IP address: 127.0.0.1
Server's port: 1234
=====Start the three-way handshake=====
Send a packet(SYN) to 127.0.0.1 : 1234
        Send a packet at : 1 byte
Receive a packet(SYN/ACK) from 127.0.0.1 : 1235
        Receive a packet (seq_num = 2844, ack_num = 2041)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 1025 byte
=====Complete the three-way handshake=====
Enter command: dns google.com
Send a command.
        Send a packet at : 2049 byte
Finding IP address of google.com by 127.0.0.1 : 1235
        Receive a packet (seq_num = 3868, ack_num = 2042)
Result: 142.251.42.238
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 3073 byte
Finish finding
google.com
Enter command:
```

Client2:

```
Server's IP address: 127.0.0.1
Server's port: 1234
=====Start the three-way handshake=====
Send a packet(SYN) to 127.0.0.1 : 1234
        Send a packet at : 1 byte
Receive a packet(SYN/ACK) from 127.0.0.1 : 1236
        Receive a packet (seq_num = 2844, ack_num = 470)
Send a packet(ACK) to 127.0.0.1 : 1236
        Send a packet at : 1025 byte
=====Complete the three-way handshake=====
Enter command: cal 5 add 13
Send a command.
        Send a packet at : 2049 byte
Caculating 5 add 13 by 127.0.0.1 : 1236
        Receive a packet (seq_num = 3868, ack_num = 471)
Result: 18
Send a packet(ACK) to 127.0.0.1 : 1236
        Send a packet at : 3073 byte
Finish Caculation
Enter command:
```

Server:

```
>>Waiting command<<
        Receive a packet (seq_num = 469, ack_num = 0)
cwnd = 1, rwnd = 524288, threshold = 65536
        Send a packet at : 1 byte
        Receive a packet (seq_num = 470, ack_num = 2845)
>>Waiting command<<
        Receive a packet (seq_num = 2041, ack_num = 2845)
dns google.com
==Start finding: google.com to 127.0.0.1 : 53029
cwnd = 2, rwnd = 524288, threshold = 65536
        Send a packet at : 1025 byte
        Receive a packet (seq_num = 2042, ack_num = 3869)
==Finish finding==
>>Waiting command<<
        Receive a packet (seq_num = 470, ack_num = 2845)
cal 5 add 13
==Start calculation to 127.0.0.1 : 44209
cwnd = 2, rwnd = 524288, threshold = 65536
        Send a packet at : 1025 byte
        Receive a packet (seq_num = 471, ack_num = 3869)
==Finish calculation==
>>Waiting command<<
```

**Step 3:**

1. Including the previous step's 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.
   Client: There is no additional output format, just show the losses in the demo.

Client:

```
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 364545 byte
        Receive a packet (seq_num = 364384, ack_num = 2634)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 365569 byte
Packet loss: ack => 2635
        Receive a packet (seq_num = 365408, ack_num = 2636)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 366593 byte
```

Server:

```
        Receive a packet (seq_num = 2630, ack_num = 360289)
        Send a packet at : 360449 byte
        Receive a packet (seq_num = 2631, ack_num = 361313)
        Send a packet at : 361473 byte
        Receive a packet (seq_num = 2632, ack_num = 362337)
        Send a packet at : 362497 byte
        Receive a packet (seq_num = 2633, ack_num = 363361)
        Send a packet at : 363521 byte
        Receive a packet (seq_num = 2634, ack_num = 364385)
        Send a packet at : 364545 byte
        Receive a packet (seq_num = 2636, ack_num = 365409)
```

**Step 4:**

1. Including the previous step's 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:

```
Enter command: get 1.mp4
Send a command.
        Send a packet at : 2049 byte
Receive video: 1.mp4 from 127.0.0.1 : 1235
        Receive a packet (seq_num = 1689, ack_num = 2862)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 3073 byte
        Receive a packet (seq_num = 2713, ack_num = 2863)
        Receive a packet (seq_num = 3737, ack_num = 2863)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 4097 byte
        Receive a packet (seq_num = 4761, ack_num = 2864)
        Receive a packet (seq_num = 5785, ack_num = 2864)
```

Server:

```
==Sending video: 1.mp4 to 127.0.0.1 : 34189
cwnd = 2, rwnd = 524288, threshold = 65536
        Send a packet at : 1025 byte
        Receive a packet (seq_num = 2862, ack_num = 1690)
        Send a packet at : 2049 byte
cwnd = 4, rwnd = 522240, threshold = 65536
        Send a packet at : 3073 byte
        Receive a packet (seq_num = 2863, ack_num = 3738)
        Send a packet at : 4097 byte
        Send a packet at : 5121 byte
        Receive a packet (seq_num = 2864, ack_num = 5786)
        Send a packet at : 6145 byte
```

**Step 5:**

1. Implement congestion control including slow start and congestion avoidance.
2. You need to reset the threshold to a lower value in order to enter the status of congestion avoidance.

Client:

```
Receive video: 1.mp4 from 127.0.0.1 : 1235
        Receive a packet (seq_num = 6696, ack_num = 8332)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 3073 byte
        Receive a packet (seq_num = 7720, ack_num = 8333)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 4097 byte
        Receive a packet (seq_num = 8744, ack_num = 8334)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 5121 byte
        Receive a packet (seq_num = 9768, ack_num = 8335)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 6145 byte
        Receive a packet (seq_num = 10792, ack_num = 8336)
```

Server (slow start):

```
*****slow start*****
cwnd = 2048, rwnd = 524288, threshold = 65536
        Send a packet at : 2048 byte
        Receive a packet (seq_num = 8332, ack_num = 6697)
cwnd = 2048, rwnd = 523264, threshold = 65536
        Send a packet at : 3072 byte
        Receive a packet (seq_num = 8333, ack_num = 7721)
cwnd = 4096, rwnd = 522240, threshold = 65536
        Send a packet at : 4096 byte
        Receive a packet (seq_num = 8334, ack_num = 8745)
cwnd = 4096, rwnd = 521216, threshold = 65536
        Send a packet at : 5120 byte
        Receive a packet (seq_num = 8335, ack_num = 9769)
cwnd = 4096, rwnd = 520192, threshold = 65536
        Send a packet at : 6144 byte
        Receive a packet (seq_num = 8336, ack_num = 10793)
cwnd = 4096, rwnd = 519168, threshold = 65536
        Send a packet at : 7168 byte
        Receive a packet (seq_num = 8337, ack_num = 11817)
```

Server (congestion avoidance):

```
cwnd = 32768, rwnd = 462848, threshold = 65536
        Send a packet at : 63488 byte
        Receive a packet (seq_num = 6497, ack_num = 64908)
cwnd = 32768, rwnd = 461824, threshold = 65536
        Send a packet at : 64512 byte
        Receive a packet (seq_num = 6498, ack_num = 65932)
*****Congestion avoidance*****
cwnd = 65536, rwnd = 460800, threshold = 65536
        Send a packet at : 65536 byte
        Receive a packet (seq_num = 6499, ack_num = 66956)
cwnd = 65536, rwnd = 459776, threshold = 65536
        Send a packet at : 66560 byte
        Receive a packet (seq_num = 6500, ack_num = 67980)
```

**Step 6:**

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 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:

```
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 12288 byte
        Receive a packet (seq_num = 14216, ack_num = 1503)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 13312 byte
        Receive a packet (seq_num = 15240, ack_num = 1504)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 14336 byte
        Receive a packet (seq_num = 16264, ack_num = 1505)
```

Server:

```
cwnd = 8192, rwnd = 516096, threshold = 65536
        Send a packet at : 10240 byte
        Receive a packet (seq_num = 1502, ack_num = 13193)
cwnd = 8192, rwnd = 515072, threshold = 65536
        Send a packet at : 11264 byte
        Receive a packet (seq_num = 1502, ack_num = 13193)
Receive duplicate ACKs.
*****Fast retransmit*****
*****Slow start*****
cwnd = 1024, rwnd = 524288, threshold = 4096
        Send a packet at : 8192 byte
        Receive a packet (seq_num = 1502, ack_num = 13193)
cwnd = 2048, rwnd = 523264, threshold = 4096
        Send a packet at : 9216 byte
        Receive a packet (seq_num = 1503, ack_num = 14217)
cwnd = 2048, rwnd = 522240, threshold = 4096
        Send a packet at : 10240 byte
        Receive a packet (seq_num = 1504, ack_num = 15241)
```

**Step 7:**

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 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:

```
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 7168 byte
        Receive a packet (seq_num = 6558, ack_num = 1965)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 8192 byte
        Receive a packet (seq_num = 7582, ack_num = 1966)
Send a packet(ACK) to 127.0.0.1 : 1235
        Send a packet at : 9216 byte
        Receive a packet (seq_num = 8606, ack_num = 1967)
```

Server:

```
cwnd = 4096, rwnd = 520192, threshold = 65536
        Send a packet at : 6144 byte
        Receive a packet (seq_num = 1965, ack_num = 6559)
cwnd = 4096, rwnd = 519168, threshold = 65536
        Send a packet at : 7168 byte
        Receive a packet (seq_num = 1965, ack_num = 6559)
Receive duplicate ACKs.
*****Fast recovery*****
*****Congestion avoidance*****
cwnd = 5120, rwnd = 524288, threshold = 2048
        Send a packet at : 4096 byte
        Receive a packet (seq_num = 1965, ack_num = 6559)
```

**Step 8:**

1. Including the previous step's 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.