<2021 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 the application layer and call UDP to transmit TCP packets.

Rules

1. Run your program on Ubuntu 20.04 platform.
2. Do not copy homework from your classmates or senior, 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 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's output results in 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: B063040000_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 2021/06/16 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: 2021/06/18 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 subnetwork.

Step 1:

1. Set the parameters including RTT (15 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 result of DNS and mathematical equations to multiple clients at the same time. (You can use fork or thread.)

3. In DNS function, the client send a domain name to the server. The server send the DNS packet with this domain name to DNS server (ex : 8.8.8.8), then get the result of IP from DNS server. Finally, the server send the result to client and client should show the result on screen.

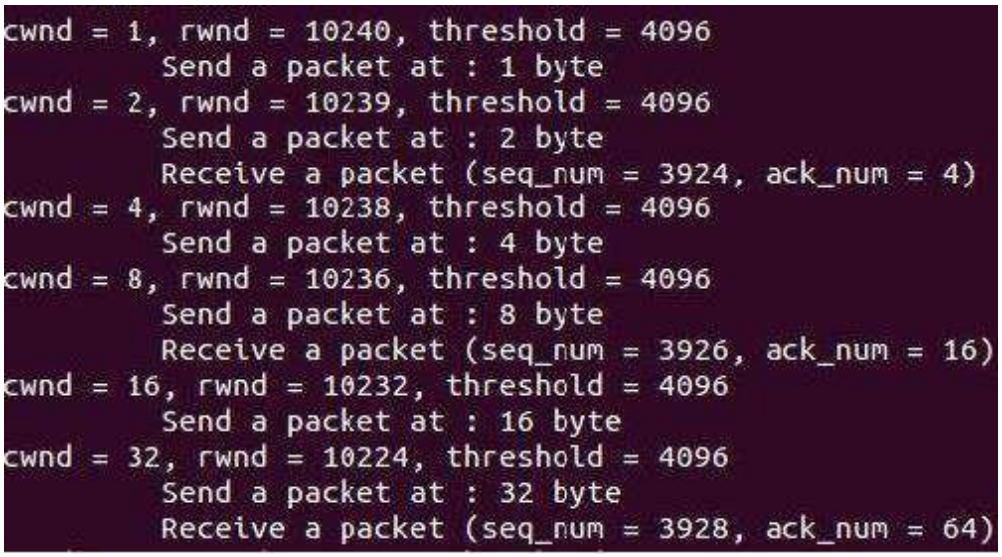
(ex: You input "google.com" and then you will get the result as 172.217.160.110)

4. The mathematical calculations include add, subtract, multiply, divide, power and square root.

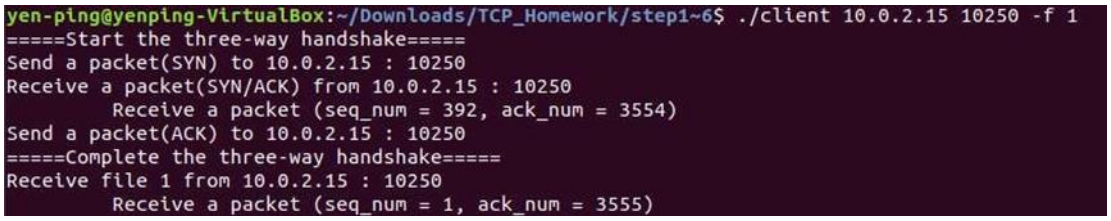
5. You also have to implement the data transmission (You need to ensure that the data are transmitted from the server to clients, and ACK packets are transmitted from clients to the server).

6. You have to print out the status of the server and clients. For example, for the server, which clients the server is sending to and which files the server receives in this step.

Server:



Client:



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.

   (You can show this step's result with step 3)

Step 3 :

1. Including the previous step's function.
2. The clients begin to request various services from the server and display results on the screen.

Step 4:

1. Including the previous step's function.

2. You should randomly generate some packet loss under random distribution with a mean of 10^{-6} and print the ACK number of loss packet.

Client: There is no additional output format, just show the lost in demo.

Step 5:

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 6:

1. Implement the 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.

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 7:

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 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 8:

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 9:

1. Including the previous step’s function.
2. Now increase the number of clients to 100. The server side has to implement someprevention mechanisms to handle simultaneous requests from clients.

Step 10:

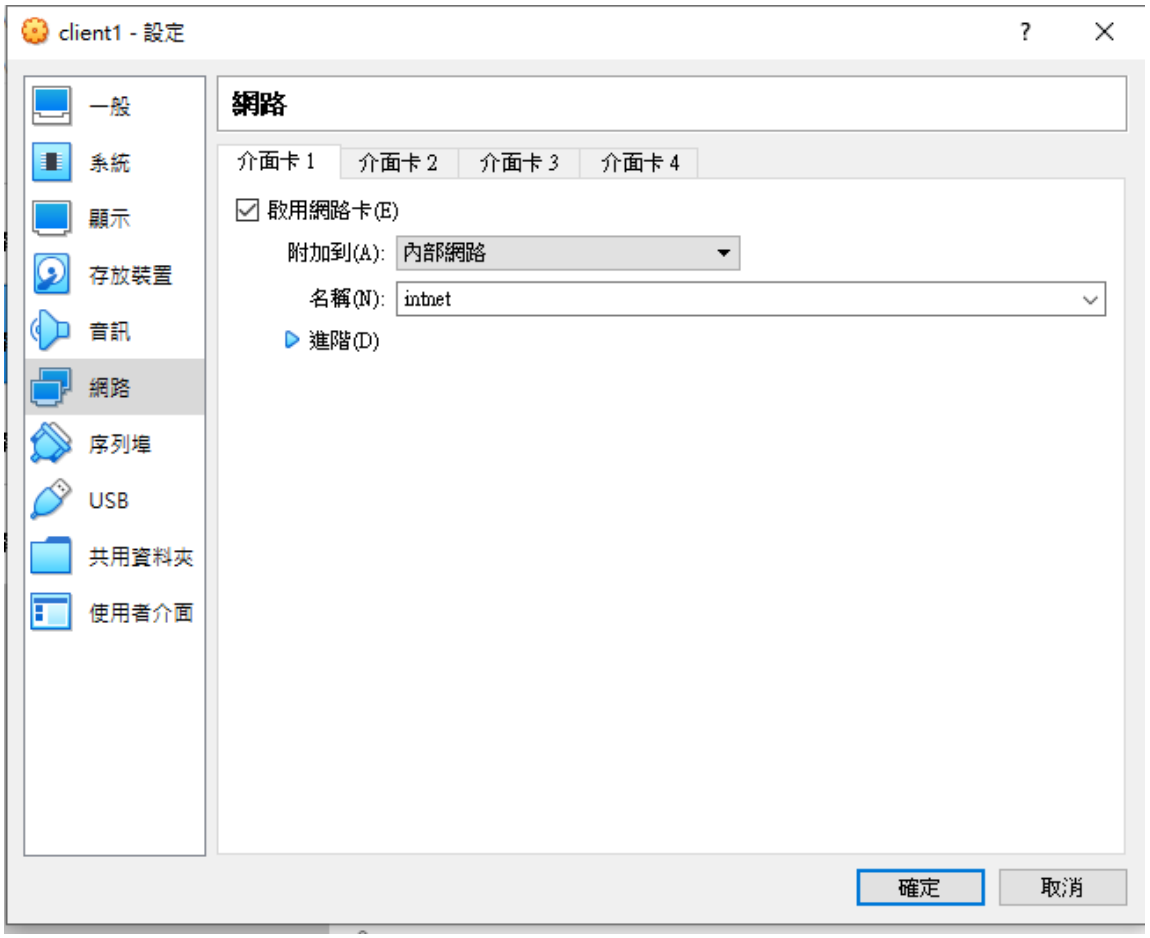
1. Including the previous step’s function.
2. Implement the functions of DHCP. Your server needs to allocate different IPs to each client.
3. I specify the number of clients of the DHCP subproblem to three clients to avoid heavy CPU loads.

Hint :

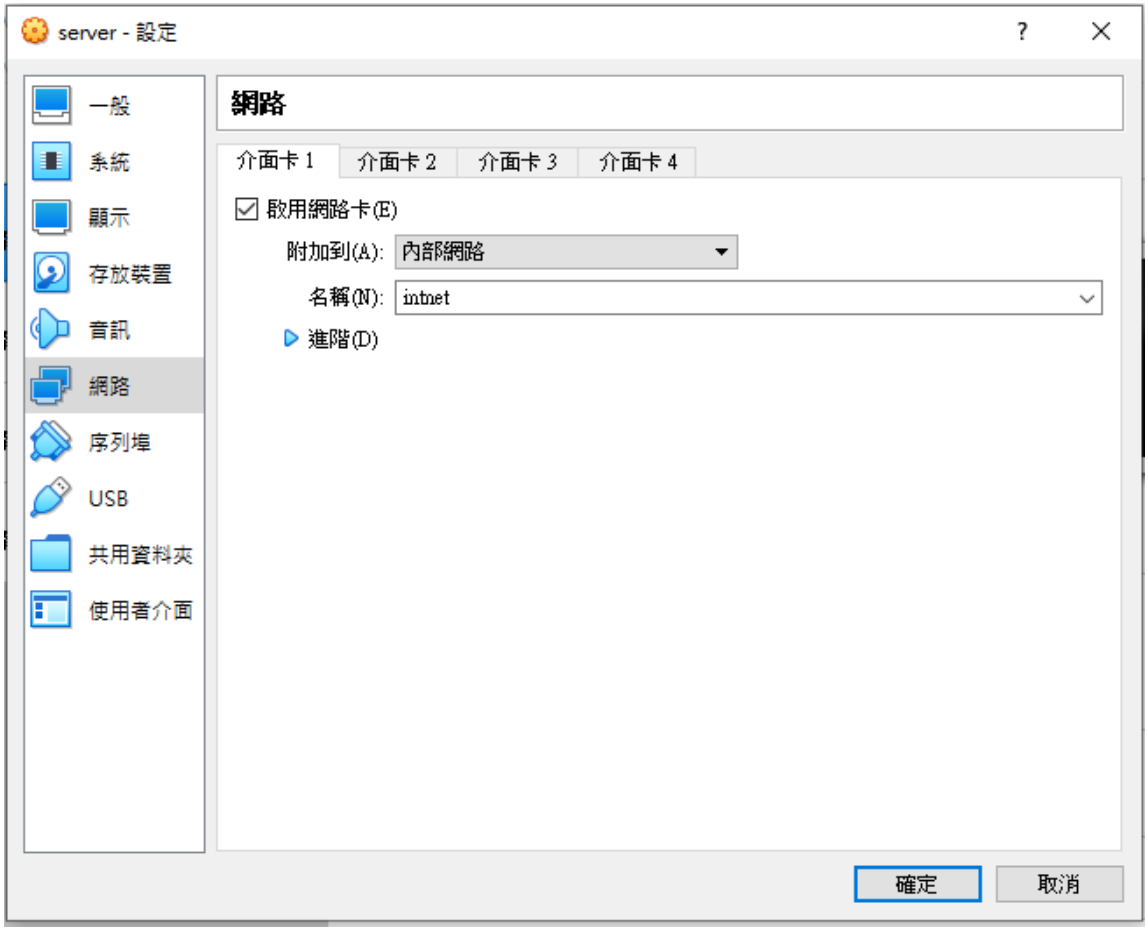
If you have no idea of how to start this step. I will show you a solution for using multiple VMs.

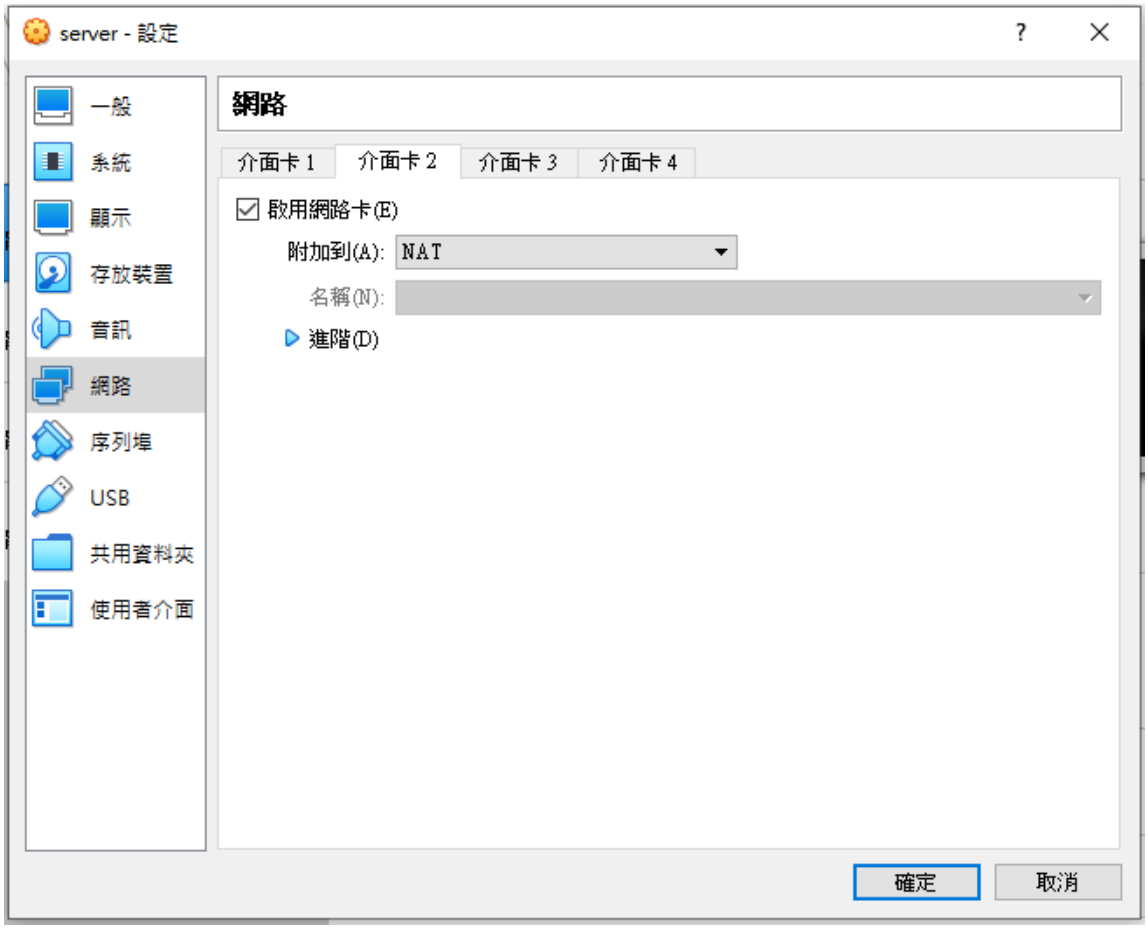
(This solution is using Virtual Box as an example which is for one client and one server.)

VM client set to Internal Network mode.

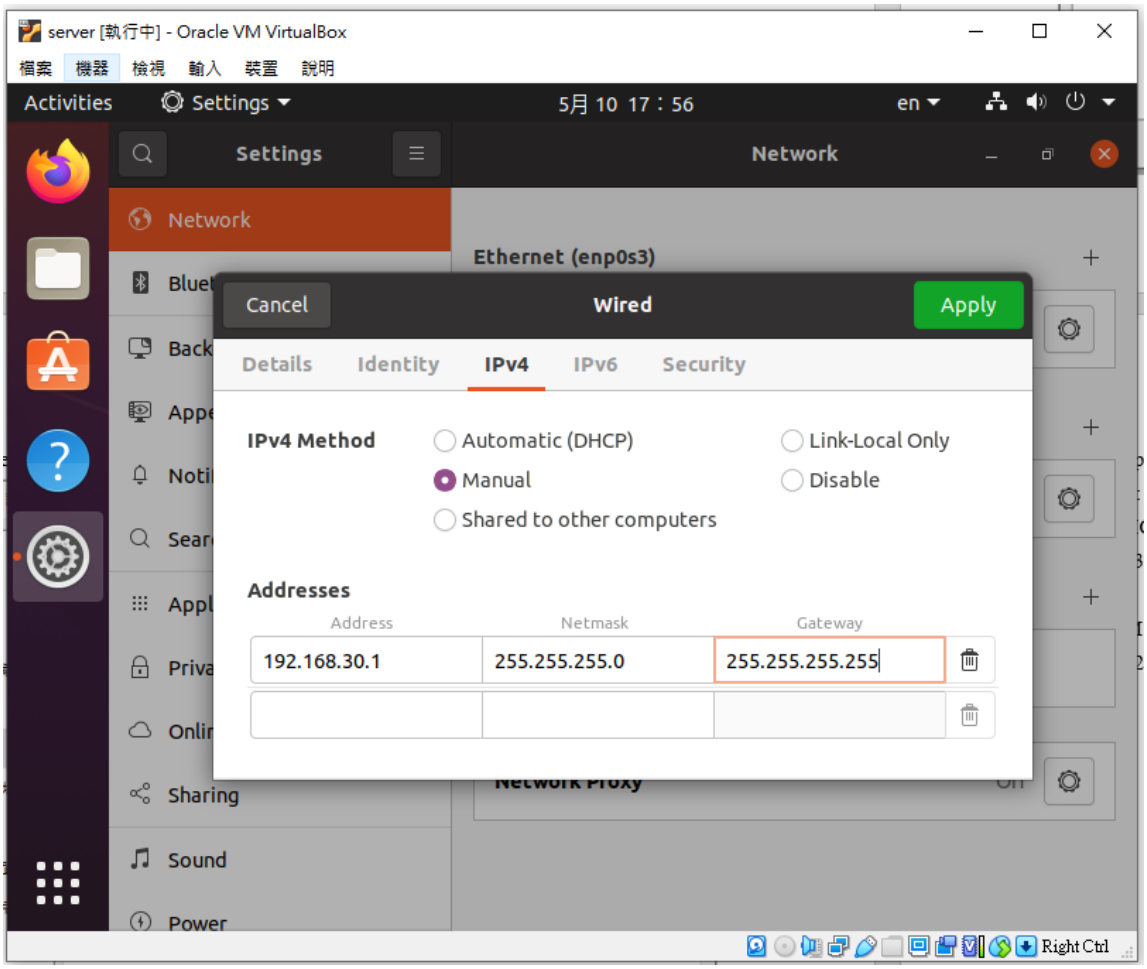


VM server set to Internal Network and NAT mode.

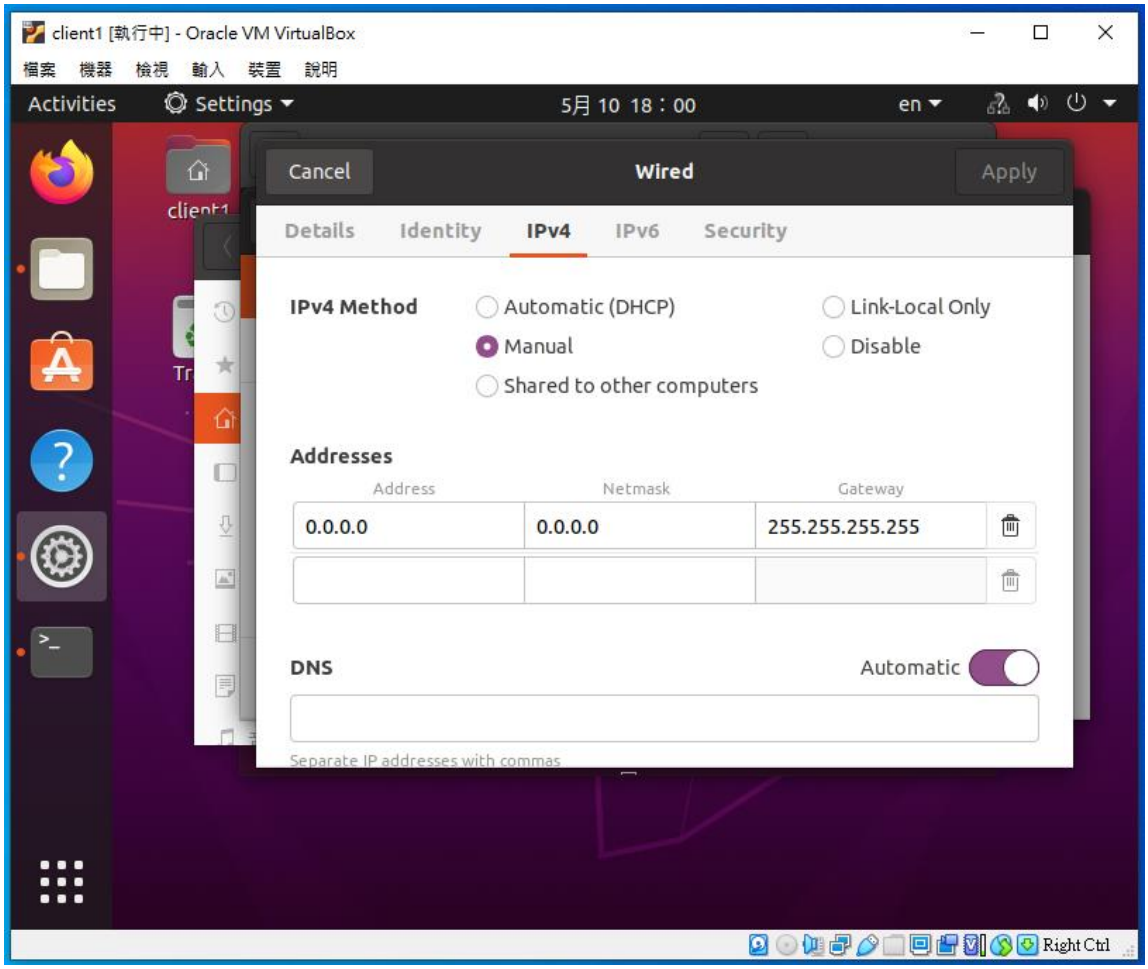




Set the private IP for the VM server. (ex : IP : 192.168.30.1 mask : 255.255.255.0 gateway : 255.255.255.255)



The VM client uses the setting as IP:0.0.0.0, mask: 0.0.0.0 and gateway : 255.255.255.255.



Now you can start to write the DHCP functions for the server and clients.
The client gets an IP from the server and uses that IP to connect to server.
The DHCP IP pool should be in the range of the subnet of the server. (ex : 192.168.30.0/24)