Shanghai Jiao Tong University

Computer Networks Lab3

Author: *LIU Qiaoan 520030910220*

Course: *Spring 2022, CS3611: Computer Networks*
Date: *April 24, 2022*

1 Build Topology.

I use Mininet to build the following topology, which contains 4 hosts.

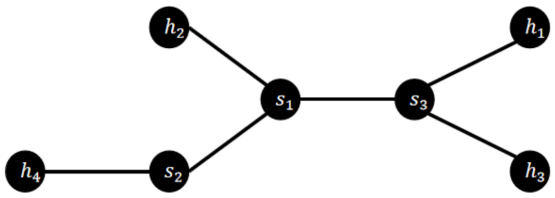

Figure 1: Topology built by Mininet

2 Chatting room with multiple users: client-server model with TCP protocol.

I write a MarkDown file to record my experience to do this task. I first got to understand the requirement of this task. We should build a chatting room with multiple users. Each user is both a client and a server. it not only need to be listening to other users, and it need to encapsulate its message and send TCP segment to others through sockets.

Then I studied the document from [1]. First, I concentrated at Chapter 5 and learned some basic functions used in socket programming. Then I studied Chapter 6 for client-server cases and Chapter 7 for some advanced functions such as `select()`.

My client.cpp file is based on the examples given on the website. First, we construct a socket `sockfd` to be listening to others (cf. server.c). Then I use a `while` loop, I define a socket file descripter set, named `readfds`, then add `sockfd` and a special `0` into this set. As we can see, this set is used to select a branch (cf. client.c and selectserver.c):

- When there is something to select now, if there is new connection from other client, we need to call `accept()` and tell it to get the pending connection. Then we call `recv()` to get a message and print the result in terminal.
- When there is something ti select now, if we input some characters and want to send them into a specific client, we first cut the input into three small pieces, then get the IP address of desitination by using a map. After calling `getaddrinfo()`, `socket()`, `connect()` and `send()`, we successfully send the message to desitination.

Then our Chatting room is completed, I start to run it in Ubuntu 20.04, which is contained in VMware Workstation Pro. I compile client.cpp by using: `g++ client.cpp -o client`. Then I start buildTopo.py to build a topology using Mininet, after open the terminal of each client by `xterm`, I run the executable files on terminals of hosts as `./client`. But when I implement it, I find that SOMETIMES a strange bug occurs (As an example in figure 2, when h1 receive message from h4, it can not get the IP address of h4):

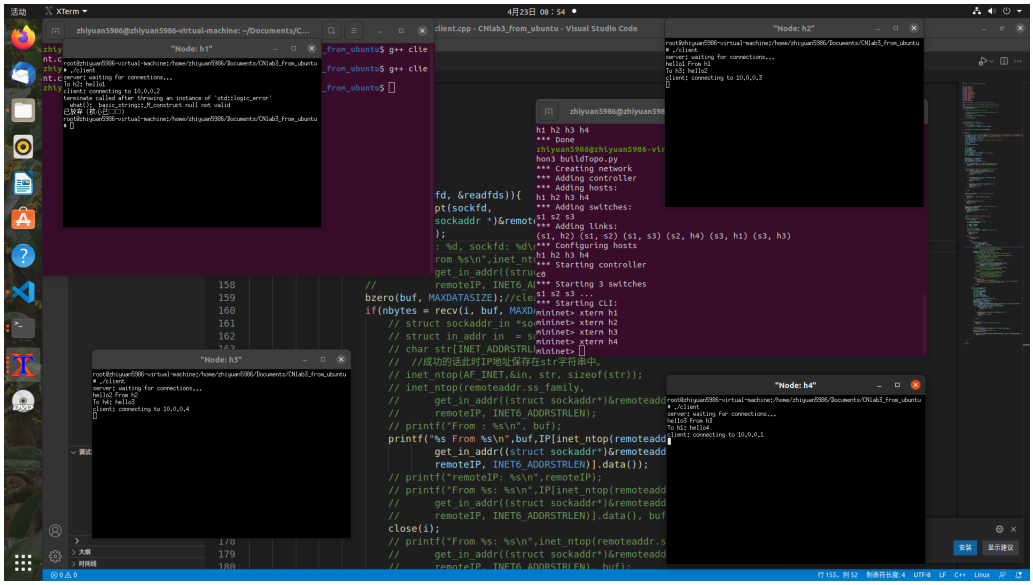

Figure 2: A bug caused by a NULL pointer of a string

After do some tests and search on the Internet, I find that this bug is caused by a NULL pointer of a string. The NULL pointer is exactly caused by the absence of correct IP address of sender.

But when I run this chatting room in my another Ubuntu 20.04, which is directly build in my computer and is not a virtual machine, I find that it runs correctly always!

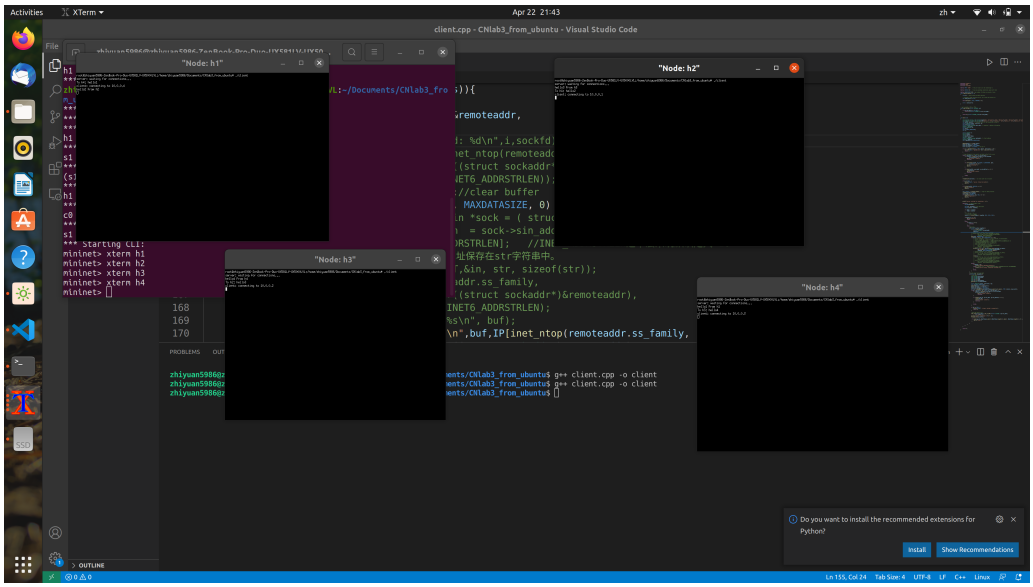

Figure 3: Result in Ubuntu 20.04 which is directly built in my computer

In my perspective, I think this bug is due to the stability of Mininet. When running on my VM, Mininet may sometimes occupy most of the memory and the performance may be unstable. (When I do lab2, my Ubuntu 18.04 in VirtualBox is broken due to Mininet since it occupied too much memory. So I make this assuption.)

3 Chatting room with multiple users: client-only model with UDP protocol

We first find the boardcast address of this net: 10.255.255.255

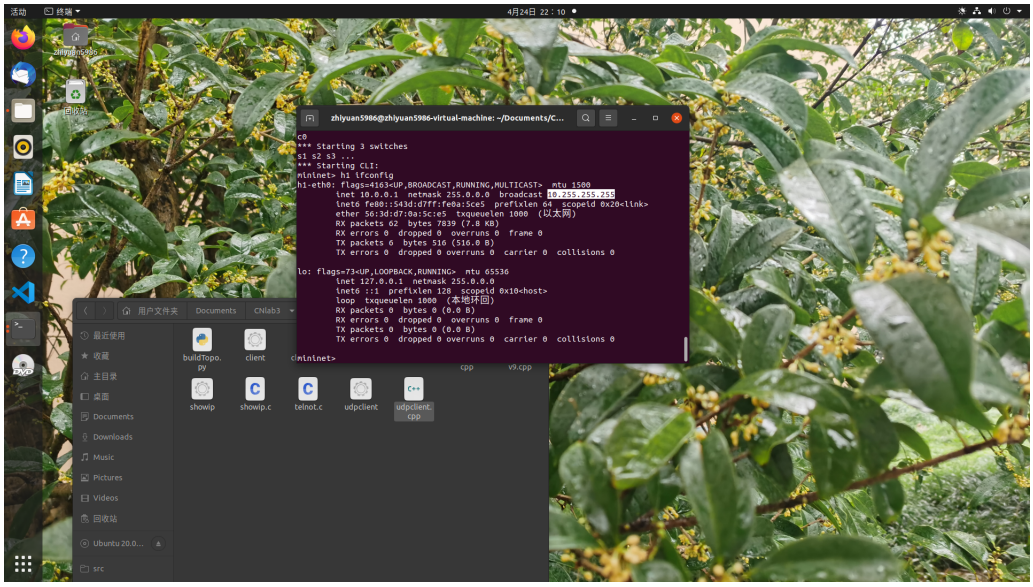

Figure 4: Boardcast address of this net

My understanding of this task is: Each client need to send message to boardcast address, and the system will forward this message to all other clients, but not the sender itself.

I studied the document from [1]. I concentrated at Section 7.7, and study how to get IP address of sender itself.

My udpclient.cpp file is based on the examples given on the website. First, we construct a socket `sockfd` to be listening to the boardcast address (cf. boardcaster.c). Then I find the IP address of myself [2], which can be used to determine whether to show the message in sender's terminal. (In the `for` loop, there is a if-statement: `if strcmp(rv_addr, local_addr)`. If the sender's IP is the same as mine, it means I'm exactly the sender, then I will not display it. Otherwise, I print the message. The key codes are in line 129 to 158 and 190.) Then I use a `for` loop, I define a socket file descripter set, named `readfds`, then add `sockfd` and a special `0` into this set. As we can see, this set is used to select a branch i.e. to send a message or to receive a message. (Similar to task 1, cf. client.c, boardcaster.c and selectserver.c):

Then our chatting room is completed, I start to run it in Ubuntu 20.04, which is contained in VMware Workstation Pro. I compile udpclient.cpp by using: `g++ udpclient.cpp -o udpclient`. Then I start buildTopo.py to build a topology using Mininet, after open the terminal of each client by `xterm`, I run the executable files on terminals of hosts as `./udpclient`. I implement it and get the result:

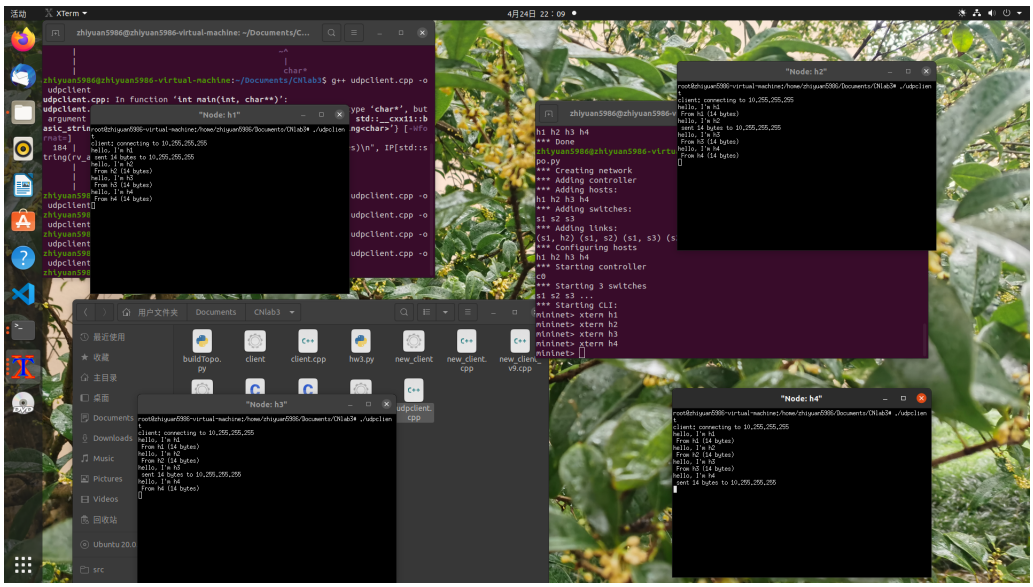

Figure 5: A useful chatting room with UDP protocol

As we can see, when a client send a message, other clients will receive this message.

4 Conclusion

This lab is quite interesting! I learned how to use socket programming to send and receive message on a net. I actually met many problems, and solving them is a chanllenging task! Thanks to Prof. Jin and TAs.

References

[1] https://beej.us/guide/bgnet/

[2] https://blog.csdn.net/zspzwal/article/details/51276929