Assignment 4

2016310932 배현웅

- Environment

```
mininet@mininet-VirtualBox:~$ python --version
Python 2.7.17
mininet@mininet-VirtualBox:~$ python3 --version
Python 3.6.9
```

```
mininet@mininet-VirtualBox:~$ cat /etc/issue
Ubuntu 18.04.5 LTS \n \l
```

사용한 라이브러리 : socket, struct, threading

- Explanation

  - Client

ID 를 입력 받고 클라이언트는 소켓 두개를 엽니다. 하나는 서버 통신, NAT 를 통한 네트워크 연결을 위한 소켓이고, 나머지 하나는 NAT 를 통해 통신하지 않는 private ip 와 port 로 통신하는 소켓을 동시에 thread 를 통해서 열어둡니다. 그 이후 서버에 packet 을 보내 등록을 요청하고, 현재 등록되어있는 클라이언트 정보를 받습니다. 그후 client 함수에서는 input 을 통해서 입력받는 명령인 @chat, @show_list 를 입력 받아 해당 명령을 수행합니다. @chat 수행 중, 같은 public ip 를 가지고 있는 클라이언트에게는 저장되어있는 private ip 를 통해서 패킷을 전송합니다. Public ip 로 연결하는 소켓은 서버로부터 새로운 클라이언트 등록/disappear/deregister 에 대한 broadcast 패킷이나 NAT 를 통한 chat 데이터를 수신합니다. (=recv_side 함수) Extended Goal 을 구현하기 위해 private ip 로 수신 받는 소켓을 열어서 입력 받는 함수 chat_recv 를 정의하였습니다. 그 소켓을 통해 패킷을 넘겨주는 상대방은 같은 NAT 상의 있는 클라이언트뿐 입니다.

Message format 을 이렇습니다.

보내는 패킷 : #num_instr + clientID + ₩n₩r + data

수신받는 패킷 : #num_instr + serverID + ₩n₩r + data

#num_instr	1	2	3	5	6
정보	register_response	broadcast_data	chat_data	broadcast_deregister	im_alive

Data 에는 clientID|chat_data 형태이고, 서버를 통해 오는 패킷은 ID|address|private_ip + ₩n 형태를 띄고 있습니다.

  - server

서버는 실행되자 마자 소켓을 열어 클라이언트로 새로운 등록정보를 받을 준비를 합니다. 서버에서는 3 개의 패킷 정보를 처리합니다. 새로운 등록을 요청하는 패킷, 클라이언트로부터 10 초마다 받는 'keep alive' purpose 패킷, deregister 를 요청하는 패킷입니다.

새로운 등록을 요청하는 패킷은 수신 받고 등록되어있는 다른 클라이언트에게 새로운 클라이언트 등록을 알리는 broadcast packet 을 보냅니다. 그 이후 패킷으로부터 받는 clientID, public address, private address 를 받아 dictionary

자료구조에 저장합니다.

client_list = dict() # ID 와 public addr 저장하는 dictionary

client_pri_ip = dict() # ID 와 private addr 저장하는 dictionary

client_time = dict() # ID : recent_access time – disappear action 을 구현하기 위해 Thread timer 객체를 저장하고 있는 dictionary 입니다. keep alive packet 에 의해 매번 초기화됩니다. 30 초동안 초기화되지 않을 시, disappear action 을 수행합니다.

저장한 이후 해당 클라이언트에게 현재 등록되어있는 클라이언트 정보를 넘겨줍니다.

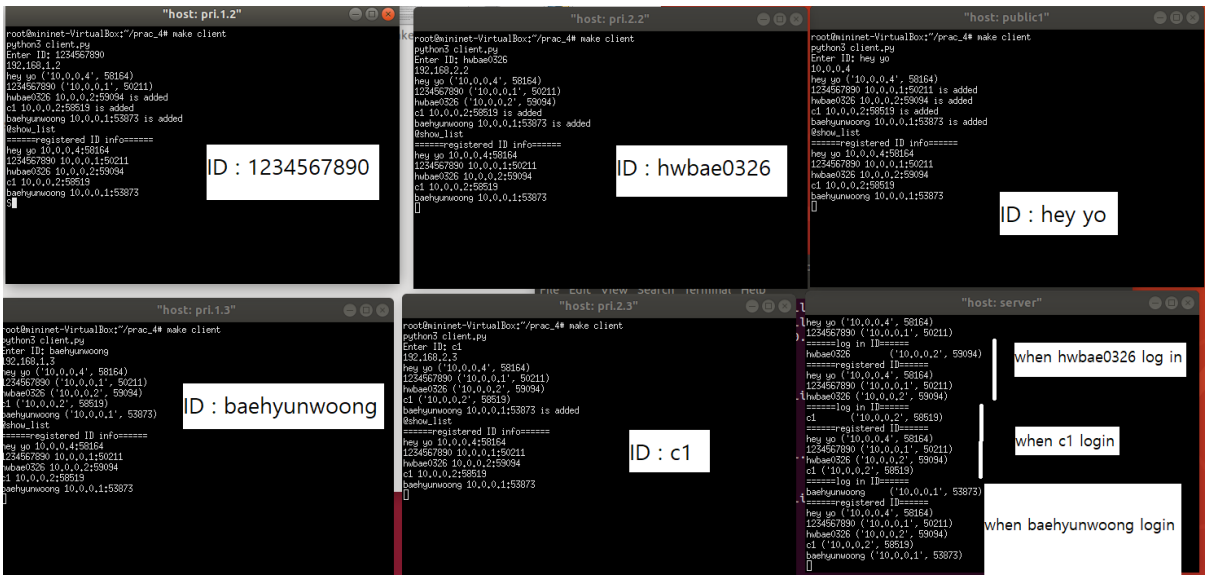
#num_instr + serverID + ₩n₩r + ID|address|private_ip + ₩n + ID|address|private_ip + ₩n + … 형태입니다.

'keep alive' purpose 패킷은 client_time 에서 해당 요청을 하는 ID 의 timer 객체를 초기화한 이후 다시 timer 을 시작합니다.

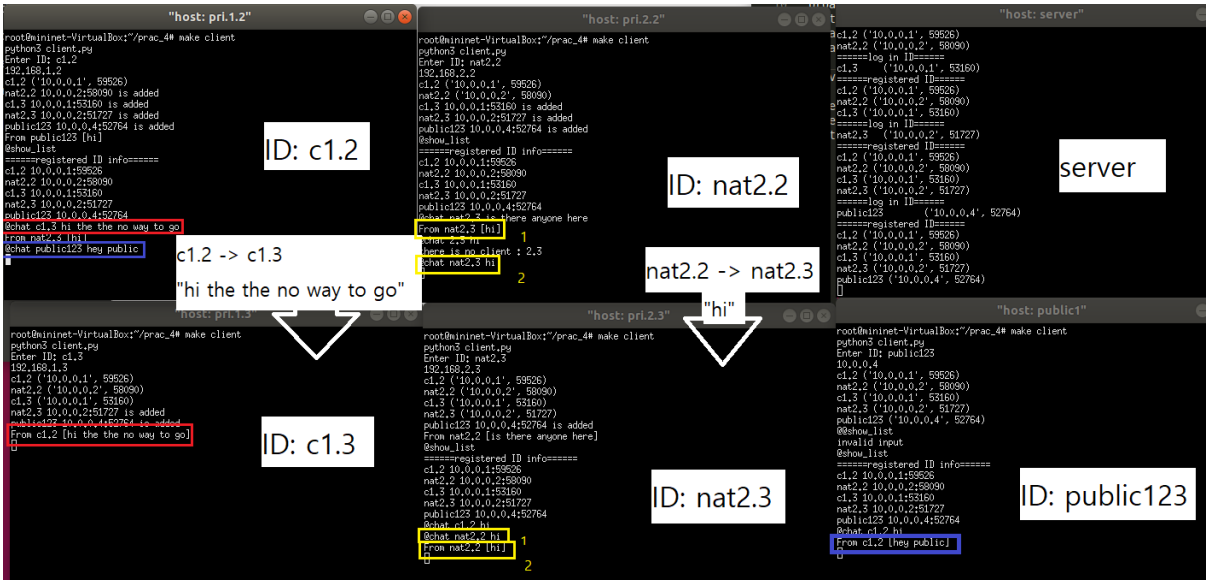
deregister 를 요청하는 패킷은 클라이언트에서 @exit 을 실행하는 받는 패킷입니다. 서버에서는 요청한 클라이언트의 데이터를 3 개의 dictionary 에서 모두 없앤 이후 남은 클라이언트에게 등록 취소를 알리기 위해 broadcast 합니다.

Disappear action : timout 함수에서 해당 action 을 수행합니다. 해당 clientID 정보를 삭제한 이후 클라이언트들에게 클라이언트정보를 삭제하라는 패킷을 broadcast 합니다.

- Evaluation
  - The client registers to the registration server.
  - The client and server can handle other client's registration and deregistration.

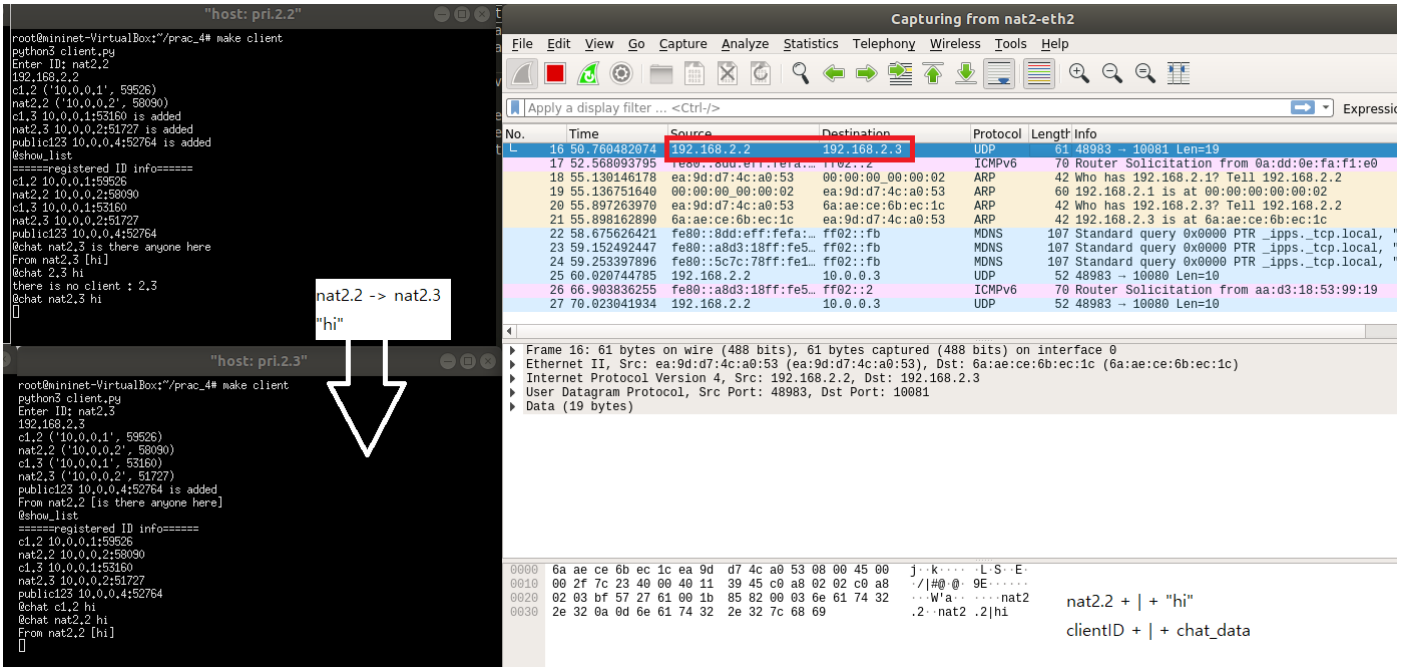


- The client can send, receive, and display a chat message.

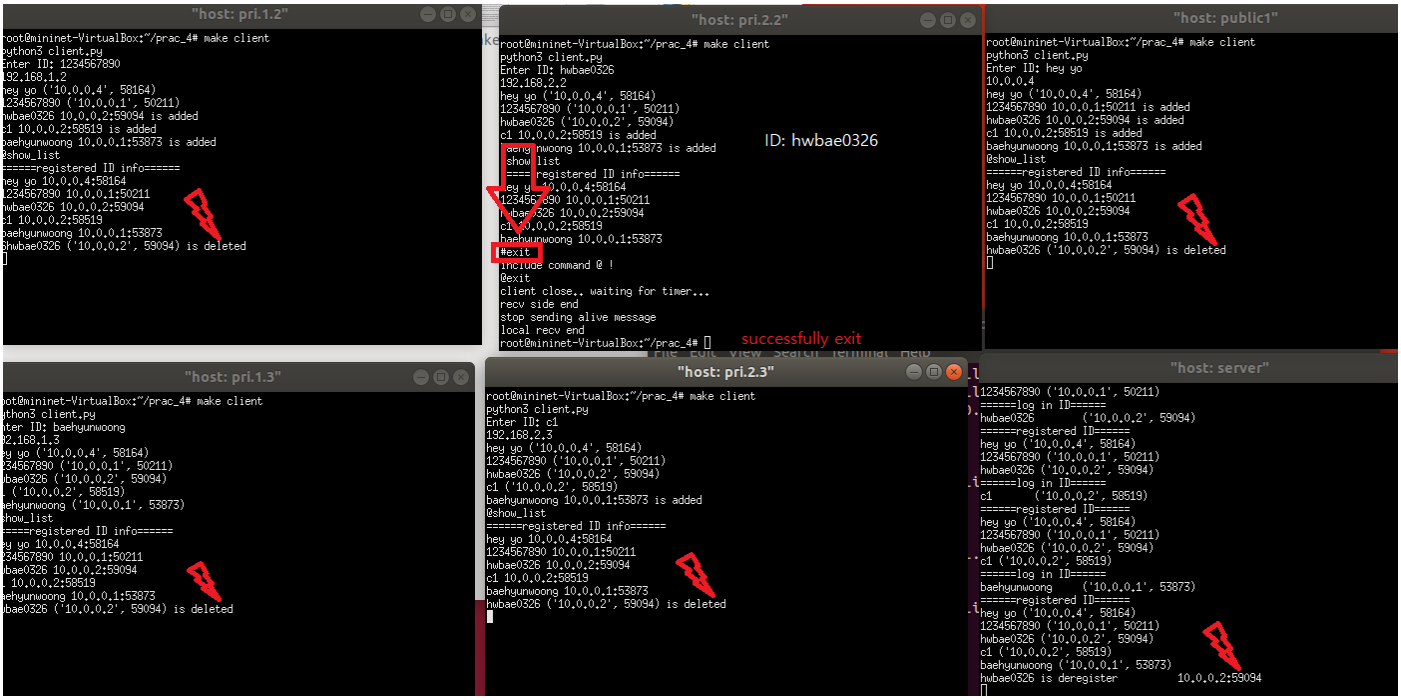


- Enable clients under the same NAT can communicate using the same private network prefix

  (without NAT assistance)



- For '@exit' command



- For client disappearance without sending a deregistration request.