Assignment4 : Solving NAT traversal problem

2016314364 박수헌

1. Development environment

운영체제 : virtual box를 사용해 제공된 mininet image를 설치

사용한 IDE : conda를 설치한 후 spyder를 이용해 개발

Python 버전 : Python 3.8.5

사용한 libraries : sys, socket, threading, pickle, time, subprocess

Main 함수 수정사항 : 과제 설명 ppt에는 처음 client가 실행되면 "Enter ID : "라는 input 문구가 표시되지만, skeleton code에는 존재하지 않았기 때문에 이를 추가해 주었다.

2. Data structures & algorithms

Global variable로 registers라는 list와 clients라는 set을 이용했고, 이 두 자료구조에 thread들이 동시에 접근해 잘못된 자료구조를 참조하지 않도록 threading.lock()의 variable인 lock을 선언했다.

<server.py>

ss라는 sender socket을 선언했다. argument로는 AF_INET, SOCK_DGRAM을 이용해 UDP 통신을 하는 데에 사용하였다. 총 두 개의 thread가 main thread에서 실행된다.

첫 번째 thread는 register_thread이다. 이 thread는 각 client들의 register를 수신 해 client들의 public IP, private IP를 등록한 후 이 정보를 각 client들에게 보내주는 역할을 한다. 우선 client socket으로부터 데이터를 받아 pickle을 통해 load한다. 이 때 받는 데이터의 자료구조는 dictionary 형태이다. 이 dictionary의 key 값들로는 ClientID, privateIP, mode 값이 있다. Mode는 1과 0이 존재한다.

Mode값이 1인 경우는 client가 server에게 등록을 위해 자신의 clientID와 함께 보낸 데이터일 수도 있고, 'keep alive'의 목적을 위한 request일 수도 있다. 따라

서 받은 데이터와 address를 이용해해 ClientID, clientAddr, rcvTime, privateIP의 키값을 갖는 temp라는 dictionary를 생성한다. 이 때 rcvTime은 time.time()을 이용해 받은 정확한 시간을 기록한다. 그리고 registers라는 list를 순회하며 이미 registers에 해당 clientAddr의 정보가 존재한다면 이는 "keep alive" 목적의 데이터이므로 해당 요소의 rcvTime만 현재의 시간으로 update해준다. 하지만 만약 registers list에 해당 clientAddr를 갖고 있는 요소가 존재하지 않는다면, 첫 등록을 위한 데이터이므로, (예시)client1 10.0.0.1:59494 라는 정보를 표시한다. 그리고 clients라는 set에 해당 client의 clientAddr를 추가한다. Clients set에 포함되어있는 address들은 이미 등록된 client들이므로, 이 clients set에 들어있는 addr들에게 registers list를 pickle을 통해 보내준다. 등록 정보를 최신화(broadcast)해주는 것이다.

Mode가 0인 경우는, deregister의 경우, 즉 client가 @exit을 입력한 경우이다. 이 때, 보낸 client는 당연히 이미 등록된 client일 것이기 때문에 registers list를 순회하며 해당 clientAddr를 가지고 있는 요소를 삭제하고 (예시)client2 is deregisters 10.0.0.2:53213 을 표시한다. Clients set에서 역시 해당 address를 삭제하고, update된 registers list를 clients set에 존재하는 address들에게 최신화(broadcast)해주기 위해 보내준다.

두 번째 thread는 client_timeout_thread이다. 이 thread는 말 그대로 timeout된, 즉 @exit을 입력하지 않고 종료된 client를 찾아 삭제하는 thread이다. 위의 thread에서 "keep alive" 요청이 오면 registers list에서 해당 clientID의 rcvTIme을 매번 최신화 해주었다. 따라서 해당 thread는 registers list를 순회하며, 각 요소들의 rcvTime값을 확인해 now=time.time()값과의 차이가 30 이상이라면, 즉 30초 이상 "keep alive" 데이터가 수신되지 않았다면, (예시)client1 is disappeared 10.0.0.1:59494 를 표시하고 registers list에서 해당 요소를 삭제하고, clients set에서 역시 삭제한다. 이 thread가 while 문을 통해 계속 실행되면 위의 register_thread가 lock을 acquire하지 못해 실행되지 못하므로 time.sleep(0.5)를 통해 0.5초의 간격을 준 뒤 다시 timeout check를 시작한다.

<client.py>

우선, subprocess 모듈을 사용해 'hostname -I'의 결과값인 privateIP를 privateIP

라는 global variable에 저장한다. client에는 총 3개의 thread가 존재한다.

첫 번째 thread는 register_send_thread이다. 이는 첫 등록 데이터와, "keep alive" 데이터를 송신하는 thread이다. key값으로 ClientID, mode, privateIP 내용을 담고 있는 temp라는 dictionary를 생성해 이를 sender에게 보낸다. 그리고 time.sleep(10)을 이용해 10초 간격으로 "keep alive" 데이터를 송신한다.

두 번째 thread는 register_receive_thread이다. 이 thread의 주된 목적은 바로 최신화된 client들의 정보를 갖고 있는 registers list를 수신하는 thread이다. 데이터가 수신되면 일단 type(data)를 통해 데이터가 list인지 dict인지 확인한다. list라면 이는 registers list라는 의미이고, dict라면 이는 chatting message라는 의미이다.

List일 경우에는 client.py의 global variable인 registers를 수신한 registers로 바꿔준다. 이 때 registers list를 순회하며 자신과 같은 clientID를 갖고 있는 요소를 찾아 그 요소에 담고 있는 clientAddr를 자신의 publicIP로 인식해 mypublicIP라는 변수에 담아 자신의 publicIP 주소를 알아낼 수 있다.

Dict일 경우에는 chatting message인 경우이다. 이 때에 수신받은 dictionary에는 'nat'이라는 key가 존재한다. 가능한 value으로는 'diff'와 'same'이 있다. 말 그대로 같은 NAT 아래에서 수신된 메시지인지, 다른 NAT 아래에서 수신된 메시지인지를 나타내는 정보이다. 'diff'라면 다른 NAT에서 왔으므로 publicIP를 이용해 왔다는 의미이다. 따라서 registers list를 순회하며 해당 clientAddr를 갖고 있는 clientID를 찾아 이를 표시해주면 된다. 'same'이라면, registers list를 순회하며 해당 privateIP를 갖고 있는 clientID를 찾아 이를 표시해주면 된다. 그 후 (예시)From client2 [hello]를 표시해준다.

마지막 thread는 instructions_thread이다. @show_list, @exit, @chat ~의 instruction만 input받을 수 있으며, 다른 문자열이 input된다면 'wrong instruction retype'이라는 문구가 표시된다.

@show_list라는 instruction이 input된다면 상시 update되는 registers list를 맞는 양식에 따라 표시해 준다.

@exit이라는 instruction이 input된다면 clientID와 mode(0)의 내용을 포함한 temp라는 dictionary를 만들어 sender에게 보내준다. 그럼 sender는 mode값을

보고 deregister할 것이다. 그리고 exit_flag를 True로 만들어준다. Exit_flag는 global variable로 main thread에서 while문을 통해 exit_flag가 true인지 false인지 계속 확인한다. 이 때, exit_flag가 true가 된다면 바로 전체 프로그램은 종료되게 된다. 모든 thread들의 daemon 속성을 True로 설정했기 때문에 main thread가 종료되는 즉시 모든 thread들이 종료된다.

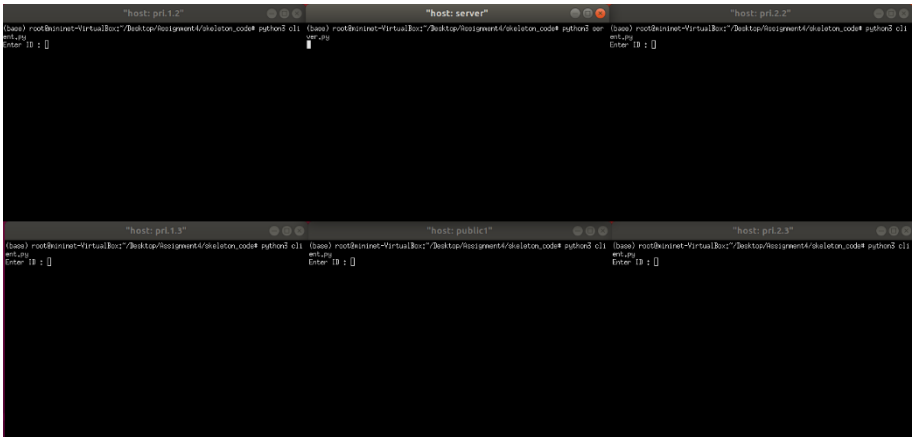
@chat ~ 이라는 instruction이 input된다면 input 문자열을 slicing해 목표 clientID와 message 부분으로 나눈다. 그리고 registers list를 순회하며, 목표 clientID가 같은 NAT 아래에 있다면 privateIP를 이용하고, 다른 NAT에 있다면 publicIP를 이용해 메시지를 송신한다.

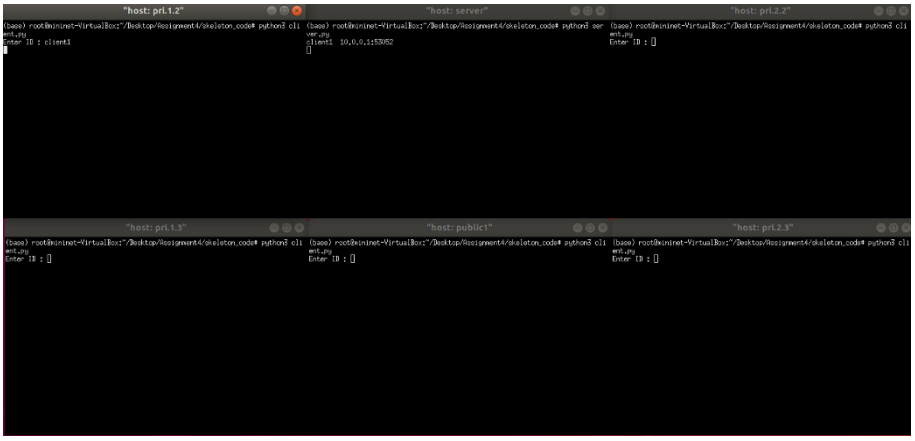
3. screen capture

실행 방법 : ./execute_mn.sh 를 입력한 후, server terminal에는 python3 sever.py 를 입력해 실행하고, 나머지 다섯개의 terminal들에는 python3 client.py를 입력해 실행한다.

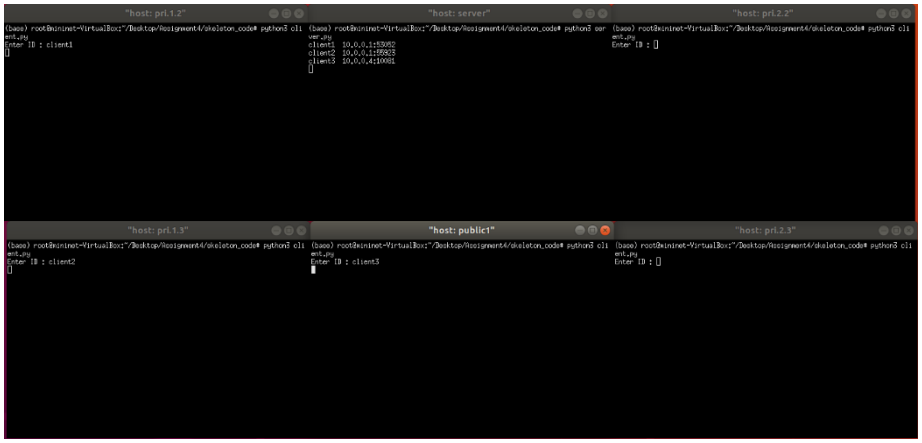
1) 20 points : client가 registration server에 등록한다.

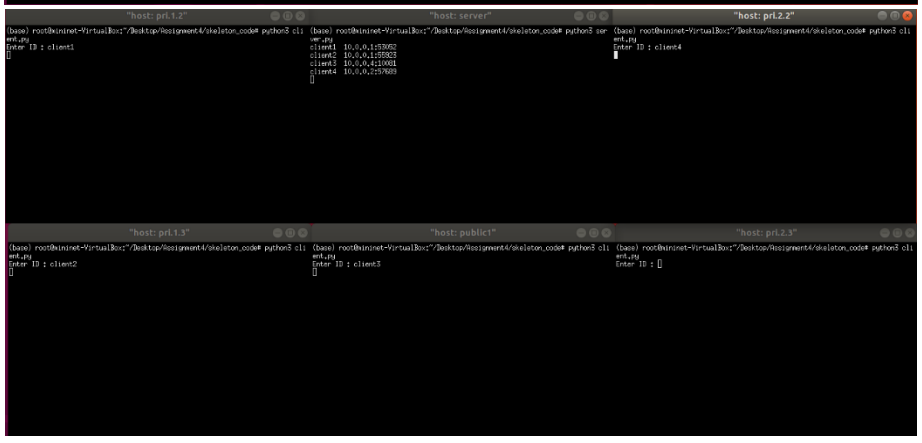
pri1.2, pri1.3, public1, pri2.2, pri2.3 순서대로 client1, client2, client3, client4, client5 의 clientID로 registration server에 register하는 화면이다.

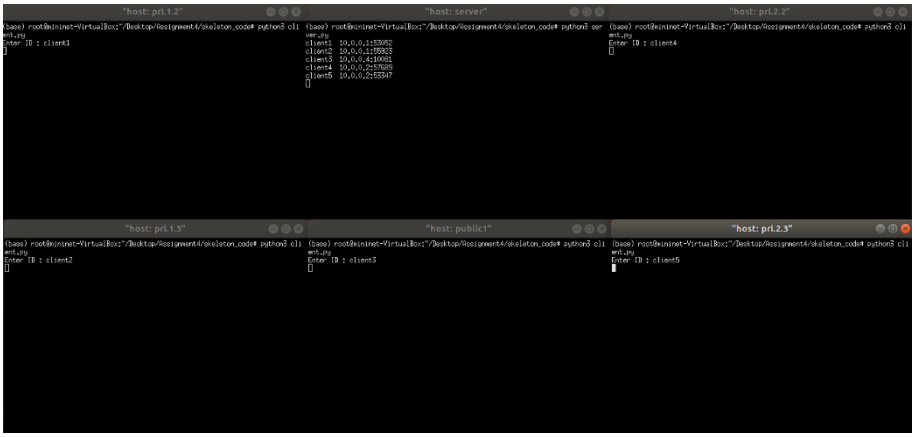



"host: pri.1.2"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : client1
"host: server"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 ser
ver.py
client1 10.0.0.1:53052
"host: pri.2.2"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : []
"host: pri.1.3"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : []
"host: public1"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : []
"host: pri.2.3"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : []


"host: pri.1.2"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : client1
"host: server"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 ser
ver.py
client1 10.0.0.1:53052
client2 10.0.0.1:55803
"host: pri.2.2"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : []
"host: pri.1.3"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : client2
"host: public1"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : []
"host: pri.2.3"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : []

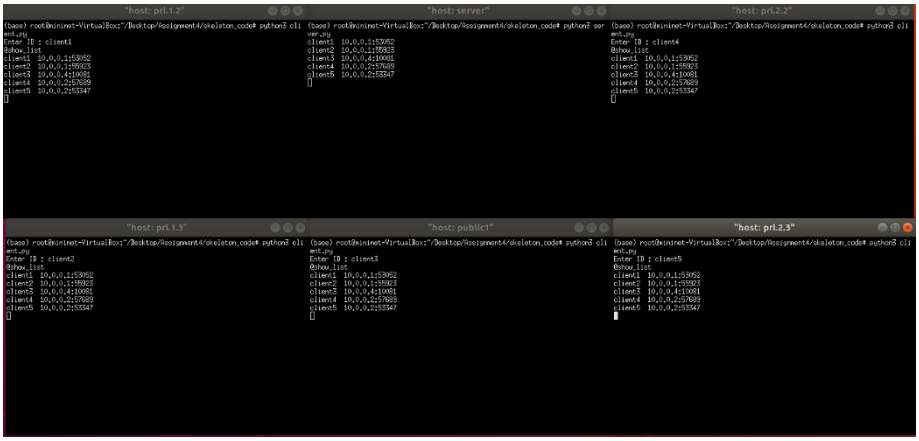

"host: pri.1.2"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : client1
"host: server"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 ser
ver.py
client1 10.0.0.1:53052
client2 10.0.0.1:55803
client3 10.0.0.4:10081
"host: pri.2.2"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : []
"host: pri.1.3"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : client2
"host: public1"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : client3
"host: pri.2.3"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : []


"host: pri.1.2"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : client1
"host: server"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 ser
ver.py
client1 10.0.0.1:53052
client2 10.0.0.1:55803
client3 10.0.0.4:10081
client4 10.0.0.7:57689
"host: pri.2.2"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : client4
"host: pri.1.3"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : client2
"host: public1"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : client3
"host: pri.2.3"
(base) root@mininet-VirtualBox:"/Desktop/Assignment4/skeleton_code# python3 cli
ent.py
Enter ID : []



2) 20 point : client와 server가 다른 client들의 registration와 deregistration을 관리한다. 서버가 새로운 registration와 deregistration 정보를 모든 client들에게 보내준다. Client가 해당 정보를 받으면 registration list를 최신화한다.

1)에서 순서대로 register했으므로 각 client마다 @show_list를 실행해 제대로 register된 client들의 정보를 받았는지, 또한 @show_list의 정상 작동 확인 화면이다.

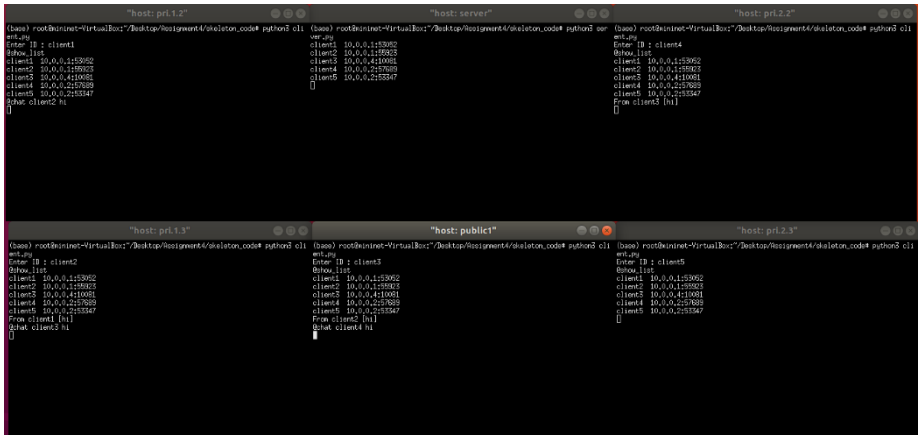


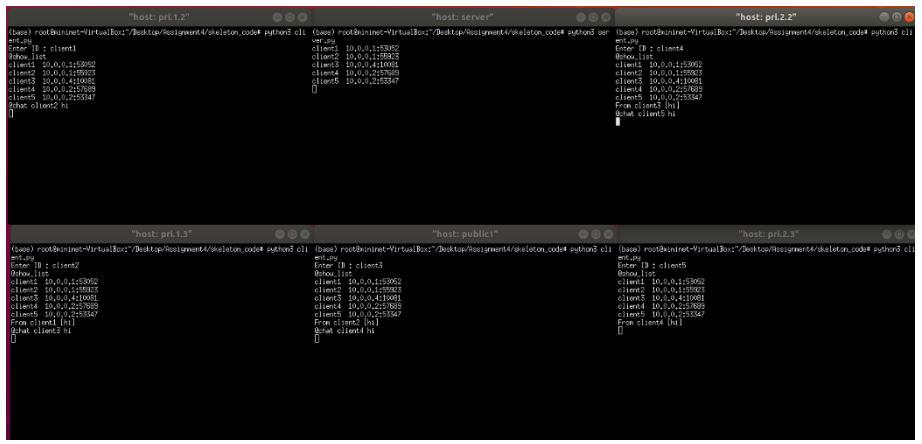
3) 20 points : client가 chat message를 보내고, 받고, 표시한다. Server의 relay 없이 chat은 직접적으로 전달된다.

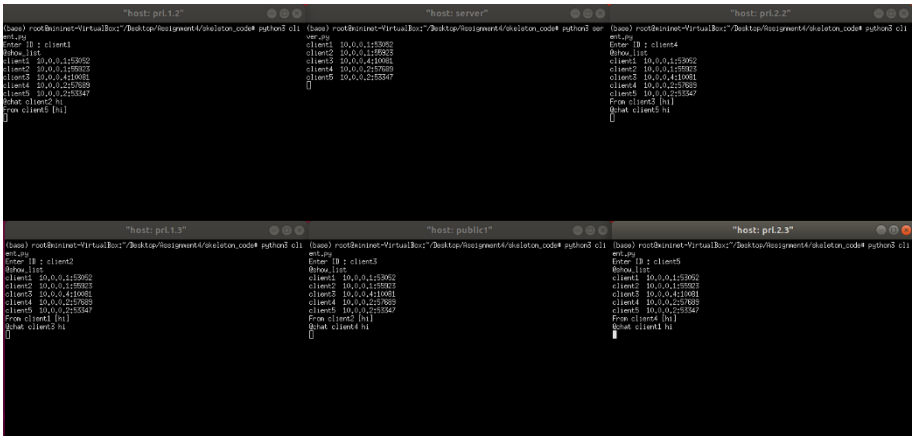
Client1 → client2 → client3 → client4 → client5 → client1 의 순서대로 hi라는 message를 보내 본 화면이다.


"host: pri.1.2"
"host: server"
"host: pri.2.2"
"host: pri.1.3"
"host: public1"
"host: pri.2.3"

"host: pri.1.2"
"host: server"
"host: pri.2.2"
"host: pri.1.3"
"host: public1"
"host: pri.2.3"


"host: pri.1.2"
"host: server"
"host: pri.2.2"
"host: pri.1.3"
"host: public1"
"host: pri.2.3"

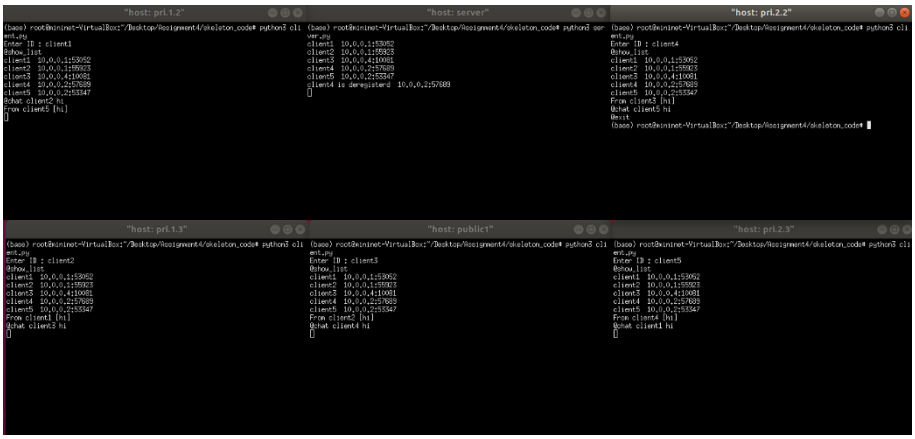

"host: pri.1.2"
"host: server"
"host: pri.2.2"
"host: pri.1.3"
"host: public1"
"host: pri.2.3"

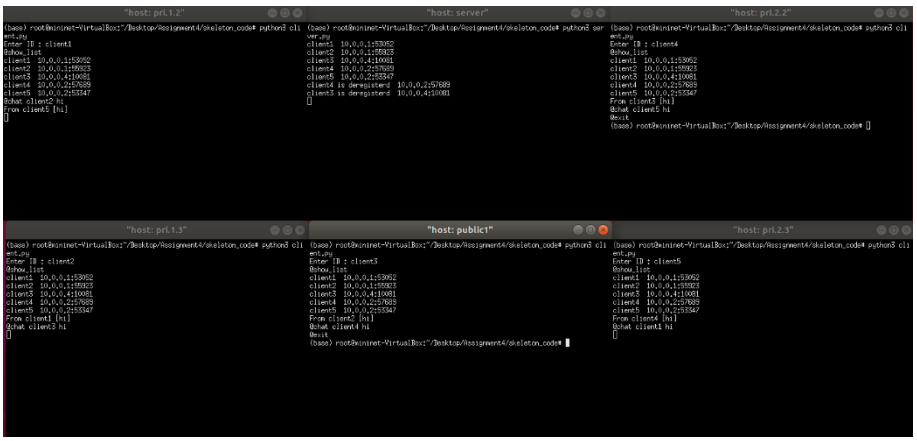


4) 10 points : 같은 NAT 아래에서는 같은 NAT의 보조 없이 같은 private network prefix를 사용해 통신하는 것은 위의 코드 설명 레포트를 통해 알 수 있다.

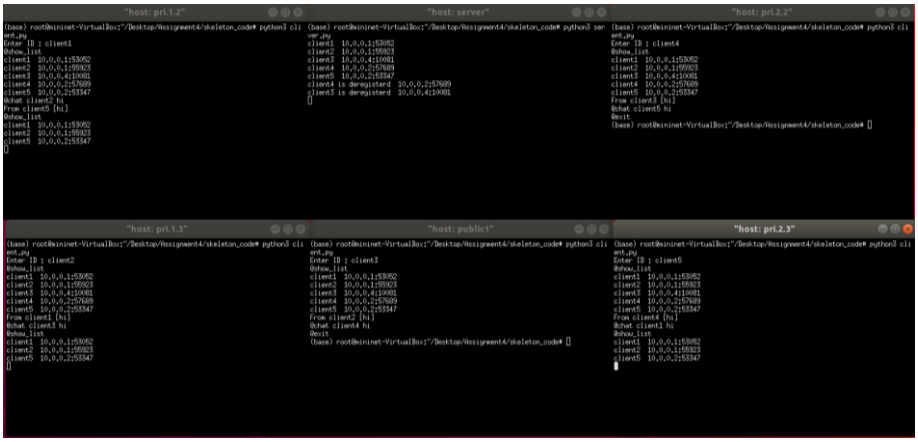
5) 10 points : @exit command. Client가 server에게 deregistration request를 보낸 후 종료된다. Server는 request를 표시하고 registration list에서 해당 client를 삭제한다.

순서대로 client4, client3이 deregister되는 화면이다.



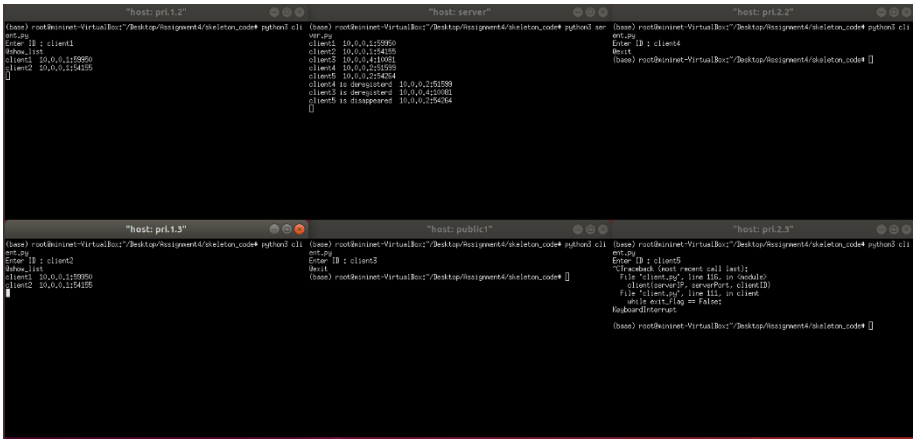


Registration list 정보가 제대로 update됐는지 확인하기 위해 남은 client들에서 @show_list를 실행한 화면이다.



6) 10 points : client가 deregistration request를 보내지 않고 종료되는 경우이다. Server는 이런 종료를 감지하고 표시한 후, 해당 client 정보를 registration list에서 삭제하고 broadcast한다.

Client5를 control+c를 이용해 강제 종료하고 일정 시간 이후 server에 disappeared 문구가 표시되는 것을 볼 수 있다. 그리고 나머지 client들에서 @show_list를 실행해 disappear된 client5가 registration list에서 삭제 되고 잘 broadcast됐는 지 확인하는 화면이다.


"host: pri.1.2"
(base) root@mininet-VirtualBox:~/Desktop/Assignment4/skeleton_code# python3 client.py
Enter ID : client1
@show_list
client1 10.0.0.1:59950
client2 10.0.0.1:54155
[]
"host: server"
(base) root@mininet-VirtualBox:~/Desktop/Assignment4/skeleton_code# python3 server.py
client1 10.0.0.1:59950
client2 10.0.0.1:54155
client3 10.0.0.4:10081
client4 10.0.0.2:51599
client5 10.0.0.2:54264
client4 is deregistered 10.0.0.2:51599
client3 is deregistered 10.0.0.4:10081
client5 is disappeared 10.0.0.2:54264
[]
"host: pri.2.2"
(base) root@mininet-VirtualBox:~/Desktop/Assignment4/skeleton_code# python3 client.py
Enter ID : client4
@exit
(base) root@mininet-VirtualBox:~/Desktop/Assignment4/skeleton_code# []
"host: pri.1.3"
(base) root@mininet-VirtualBox:~/Desktop/Assignment4/skeleton_code# python3 client.py
Enter ID : client2
@show_list
client1 10.0.0.1:59950
client2 10.0.0.1:54155
[]
"host: public1"
(base) root@mininet-VirtualBox:~/Desktop/Assignment4/skeleton_code# python3 client.py
Enter ID : client3
@exit
(base) root@mininet-VirtualBox:~/Desktop/Assignment4/skeleton_code# []
"host: pri.2.3"
(base) root@mininet-VirtualBox:~/Desktop/Assignment4/skeleton_code# python3 client.py
Enter ID : client5
^CTraceback (most recent call last):
File "client.py", line 116, in <module>
client(serverIP, serverPort, clientID)
File "client.py", line 111, in client
while exit_flag == False:
KeyboardInterrupt
(base) root@mininet-VirtualBox:~/Desktop/Assignment4/skeleton_code# []