

Introduction to Computer Network



과 목 :

Introduction to Computer Network

교 수 : 김 유 성

제 출 일 : 2020-12-02

전 공 : 의상학과 | 컴퓨터공학과

학 번 : 2015311438

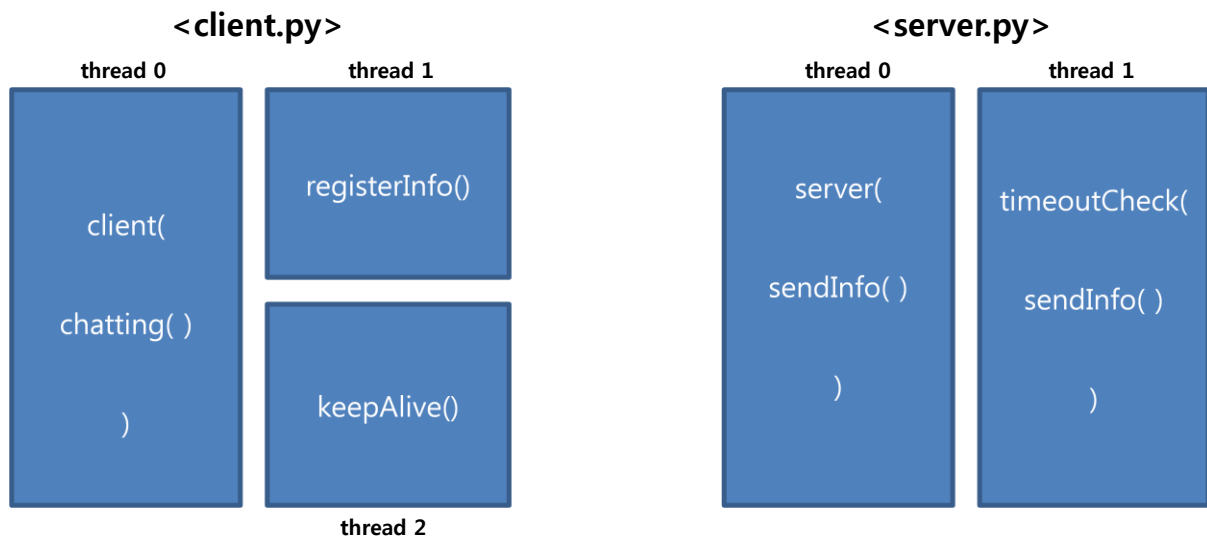
성 명 : 김 세 원

1. 개발환경

저는 올려주신 vm을 import 해서 사용했으며, 별도로 업그레이드 하거나 변동된 세팅은 없습니다. 테스트는 python3 커맨드를 이용하여 하였으며, python3 --version 은 3.6.9 임을 확인했습니다. 우분투 버전은 "18.04.05 LTS" 입니다.

사용한 라이브러리는 UDP 소켓을 생성하기 위한 socket과 timeout, registration 등의 작업은 threading 라이브러리를 import 하여 병렬처리 하였습니다. 이외에 시간 측정을 위한 time 라이브러리와 제가 작성한 코드의 일부인 packet 또한 import 하였습니다.

2. 알고리즘



< client.py >

client() – @show_list, @exit, @chat 의 커맨드를 수행하는 함수이며 서버에 해당 클라이언트의 ID 정보를 보냅니다.

chatting() – 서버를 거치지 않고, 서버로부터 받은 유저 리스트와 IP 를 통해 직접 메시지를 보냅니다.

registerInfo() – registration, deregistration 정보가 업데이트 되어 서버 측으로부터 수신될 때 마다 유저 정보를 업데이트 합니다.

keepAlive() – 10초 마다 서버로 패킷을 전송하여 disconnect 되지 않도록 합니다.

< server.py >

server() – 클라이언트 측으로부터 패킷을 받는 부분입니다. 처음 ID 등록, keep alive, exit 등의 요청을 받습니다. 패킷의 종류는 패킷의 첫 번째 요소인 req 의 값으로 판단합니다. 처음 등록은 0, exit은 1, alive는 3 으로 구분됩니다.

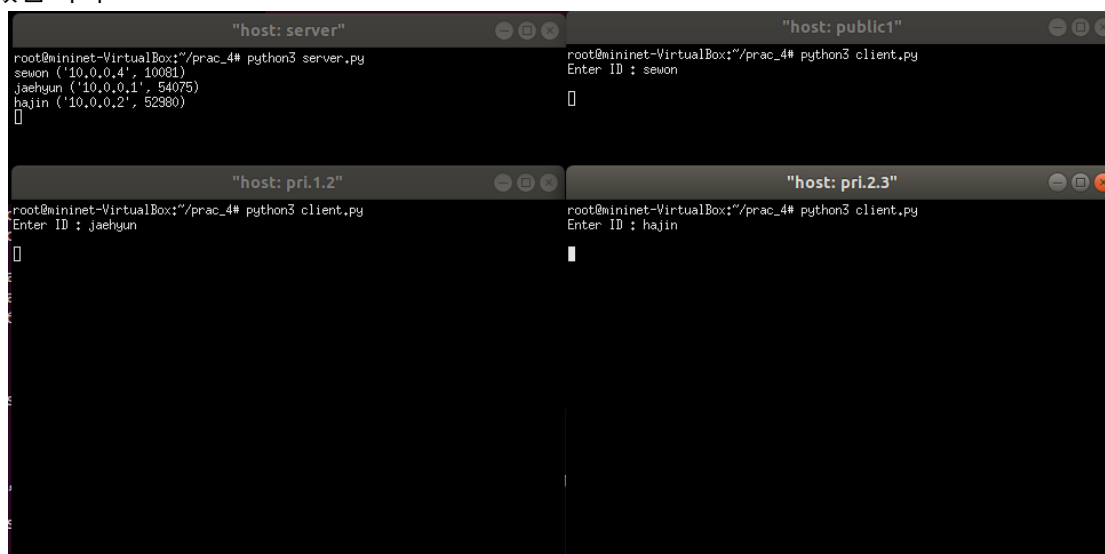
sendInfo() – 업데이트 된 유저 정보를 등록된 모든 IP 로 전송합니다. 이 때 유저 정보(id list)는 패킷 객체로 전달됩니다.

timeoutCheck() – 별도의 thread로 돌며 마지막으로 keep alive가 들어온 시간으로 disconnection 을 감지하고 유저 정보를 업데이트 합니다.

3. 구현

1) Registration

우선 저는 UDP socket을 이용하여 과제를 진행하였습니다. Server은 ('10.0.0.3', 10080) 에 bind 하였고, client host 들은 모두 ('', 10081) 로 바인드 하였습니다. NAT 뒤에 있는 private ip 의 경우에는 특별히 명시해 주지 않는 경우 해당 NAT의 IP와 각기 다른 포트를 통해 데이터를 주고 받았습니다. 저는 packet.py 파일을 따로 만들어 registration request를 서버 측에 보낼 때에 처음 id 등록 때의 req 변수 값을 '0' 으로 설정하였고, request 종류를 나타내는 bit (0 == 등록, 1 == exit, 2 == chatting mode, 3 == keep alive) 으로 사용했습니다. 패킷 클래스에는 request 변수 이외에도 userID, msg (채팅 메시지를 담습니다.), idList(서버측으로부터 업데이트 된 id 정보), externalIP(client 들의 externalIP 조회 결과), ipTable(private IP 와 public IP 가 다른 경우의 lookup table) 등의 정보가 있으며, 등록기능 뿐 아니라, 클라이언트 간의 채팅 메시지 전송, keep alive 요청 전송, deregistration 요청 등의 상황에 활용하였습니다.



```

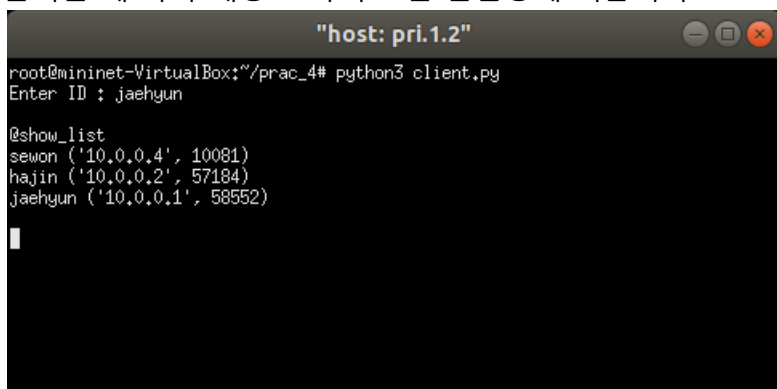
"host: server"
root@mininet-VirtualBox:~/prac_4# python3 server.py
sewon ('10.0.0.4', 10081)
jaehyun ('10.0.0.1', 54075)
hajin ('10.0.0.2', 52980)
[]

"host: public1"
root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : sewon
[]

"host: pri.1.2"
root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : jaehyun
[]

"host: pri.2.3"
root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : hajin
[]
```

위와 같이 등록 시에는 NAT 안의 private IP들도 public IP로 나타납니다. 서버 측에서는 매번 새로운 ID가 들어올 때 마다 해당 ID와 주소를 콘솔창에 띄웁니다.



```
"host: pri.1.2"
root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : jaehyun

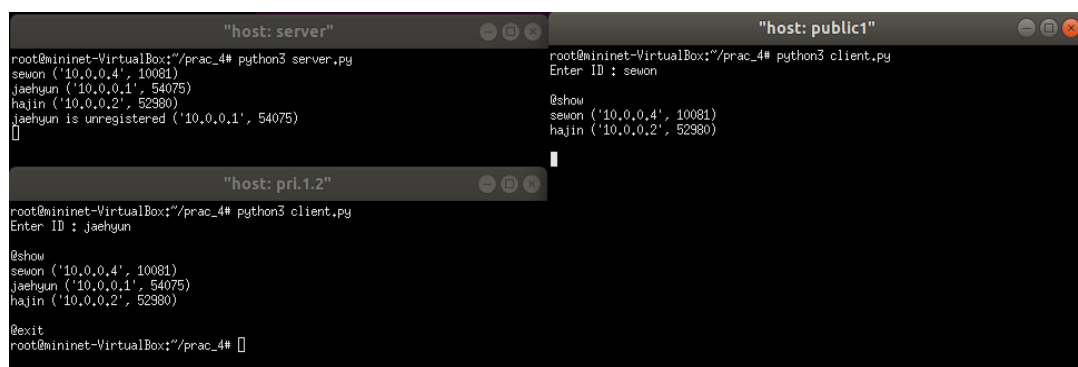
@show_list
sewon ('10.0.0.4', 10081)
hajin ('10.0.0.2', 57184)
jaehyun ('10.0.0.1', 58552)
```

client 콘솔에 '@show_list' 를 입력하면 등록되어 있는 유저 정보가 뜹니다 (원래 코드에서 @show 로 구현되어 있었는데 추후에 발견하여 모든 사진 자료를 업데이트 하지는 못했습니다. 코드는 @show_list를 입력 받을 시 리스트가 보이도록 수정하여 제출합니다). 새로운 유저가 등록을 할 때 마다 서버 측은 등록된 모든 유저 IP 에 업데이트 된 유저 정보 (idList) 를 전달하고 클라이언트 측에서도 해당 리스트를 업데이트 합니다.

2) Deregistration

@exit이나 ctrl-c 로 프로그램을 종료하여 keep alive 패킷이 보내지지 않은 경우 deregister 된 유저 정보를 broadcasting 했습니다. @exit 과 keep alive 기능의 차이점은 @exit 이 클라이언트 콘솔에 입력이 된 경우에는 deregistration info가 바로 업데이트 되고, 역시 업데이트된 유저 정보도 클라이언트 호스트 측으로 바로 broadcasting 된다는 점입니다.

2



```
"host: server"
root@mininet-VirtualBox:~/prac_4# python3 server.py
sewon ('10.0.0.4', 10081)
jaehyun ('10.0.0.1', 54075)
hajin ('10.0.0.2', 52380)
jaehyun is unregistered ('10.0.0.1', 54075)

"host: public1"
root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : sewon

@show
sewon ('10.0.0.4', 10081)
hajin ('10.0.0.2', 52380)

"host: pri.1.2"
root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : jaehyun

@show
sewon ('10.0.0.4', 10081)
jaehyun ('10.0.0.1', 54075)
hajin ('10.0.0.2', 52380)

@exit
root@mininet-VirtualBox:~/prac_4#
```

3

1

3) Chat message 전송

Client 가 서버로 부터 받는 정보는 유저 정보 (등록 업데이트) 뿐이며, 메시지를 전달할 때에

는 서버로 부터 받은 유저 정보를 통해 직접 다른 client 호스트로 메시지를 전달할 수 있습니다. 등록된 유저들은 모두 UDP 소켓이 열려있는 상황이기 때문에 ip 주소와 port 번호를 알면 직접 데이터를 전송할 수 있어 서버를 거치는 relay 과정이 불필요합니다.

```
"host: server"
root@mininet-VirtualBox:~/prac_4# python3 server.py
sewon ('10.0.0.4', 10081)
hajin ('10.0.0.2', 50734)
[]
```

서버에 'sewon', 'hajin' 두 ID가 등록된 상태입니다.

```
"host: public1" "host: pri.2.2"
root@mininet-VirtualBox:~/prac_4# python3 client.py root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : sewon Enter ID : hajin

From hajin [ how are you? ] @show
@chat hajin i am great! sewon ('10.0.0.4', 10081)
[] hajin ('10.0.0.2', 50734)

@chat sewon how are you?
From sewon [ i am great! ]
```

두 클라이언트가 메시지를 주고 받습니다.

4) 동일 NAT 안에서의 메시지 전송

저는 구글 IP를 이용하여 external IP address를 알아냈습니다. 하나의 소켓이 bind 와 connect 둘 다 처리할 수 없기 때문에 이를 위해 별도의 UDP 소켓을 열어 external IP 주소를 구하고 close 했습니다. 그리고 서버측에 초기에 보내는 패킷에 external IP 주소를 인코딩 하여 보내고 서버 측 소켓에서 받은 주소와 비교하여 다른 경우 (public IP 와 private IP 가 다른 경우) 에 별도의 private IP lookup table 을 만들어 매번 유저 등록 정보를 모든 client host 에게 보낼 때에 패킷에 함께 전송했습니다.

```
"host: pri.2.3" "host: pri.2.2"
root@mininet-VirtualBox:~/prac_4# python3 client.py root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : sewon Enter ID : hajin

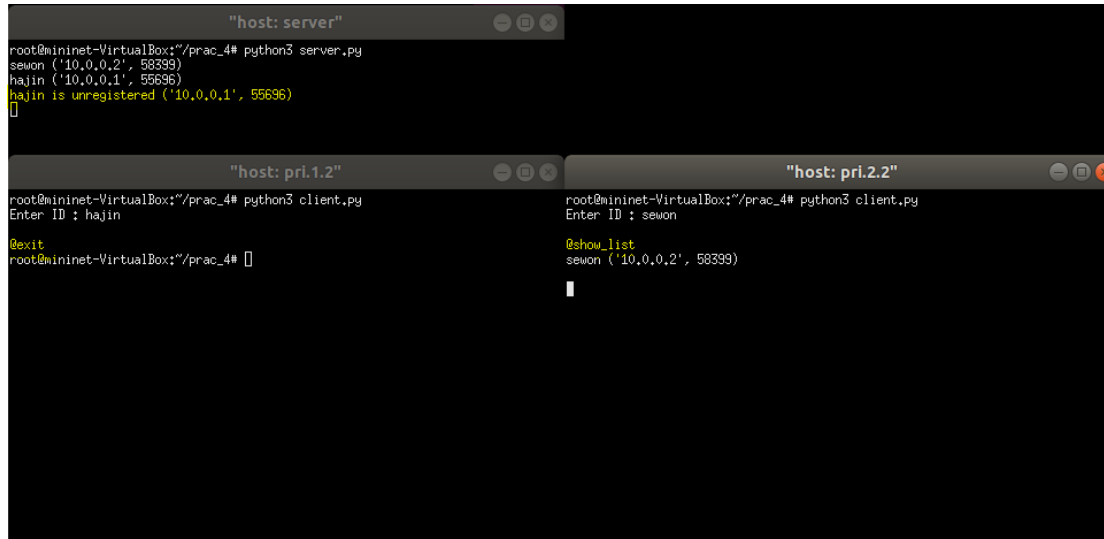
From hajin [ how are you? ] @show
Received from : ('192.168.2.2', 10081) sewon ('10.0.0.2', 54604)
@show hajin ('10.0.0.2', 50944)
sewon ('10.0.0.2', 54604)
hajin ('10.0.0.2', 50944)

@chat hajin im alright! @chat sewon how are you?
Sending to : ('192.168.2.2', 10081) Sending to : ('192.168.2.3', 10081)
From sewon [ im alright! ]
Received from : ('192.168.2.3', 10081)
[]
```

위와 같이 같은 NAT 상에 존재하는 client pri.2.2 와 pri.2.3 이 NAT의 도움 없이 직접 소통 하도록 구현했습니다. 이를 테스트 하기 위해 밑줄 친 부분을 임의로 추가해서 테스트 해보았습니다 (제출 코드에는 주석 처리 해놓았습니다). 이 때 ID 로 매핑되는 public IP를 조회해 보고, 자신의 public IP와 동일한 경우 private IP table에서 조회하여 직접 메시지를 전송하도록 구현했습니다. 위와 같이 동일 subnet 안에서는 (private IP, 10081) 로 전송이 가능합니다. 모든 client host 들은 본인 IP 와 포트 번호 10081 로 bind 했습니다.

5) @exit

클라이언트가 등록을 한 뒤 @exit을 입력할 시 패킷에 req 값을 1로 설정하고 서버 측에 보 exit(0) 합니다. 서버 측에서는 패킷을 받는 즉시 유저 정보에서 해당 IP 를 삭제하고 아직 등록된 모든 IP 들에 업데이트된 등록 정보를 전송합니다. 서버 측에서도 해당 ID 와 unregister 메시지, 주소를 콘솔창에 띄웁니다.



```
"host: server"
root@mininet-VirtualBox:~/prac_4# python3 server.py
sewon ('10.0.0.2', 58399)
hajin ('10.0.0.1', 55636)
hajin is unregistered ('10.0.0.1', 55636)
[]

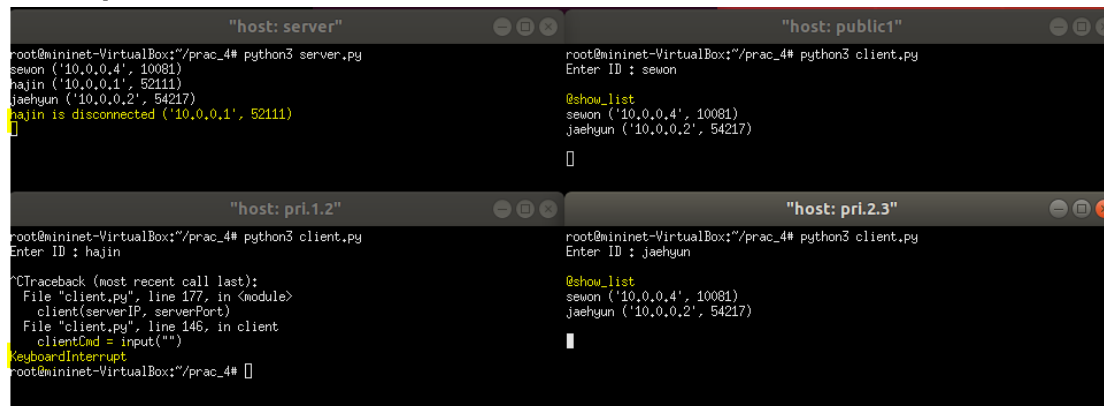
"host: pri.1.2"
root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : hajin

@exit
root@mininet-VirtualBox:~/prac_4# []

"host: pri.2.2"
root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : sewon

@show_list
sewon ('10.0.0.2', 58399)
[]
```

6) keep alive



```
"host: server"
root@mininet-VirtualBox:~/prac_4# python3 server.py
sewon ('10.0.0.4', 10081)
hajin ('10.0.0.1', 52111)
jaehyun ('10.0.0.2', 54217)
hajin is disconnected ('10.0.0.1', 52111)
[]

"host: public1"
root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : sewon

@show_list
sewon ('10.0.0.4', 10081)
jaehyun ('10.0.0.2', 54217)
[]

"host: pri.1.2"
root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : hajin

CTraceback (most recent call last):
  File "client.py", line 177, in <module>
    client(serverIP, serverPort)
  File "client.py", line 146, in client
    clientCmd = input("")
KeyboardInterrupt
root@mininet-VirtualBox:~/prac_4# []

"host: pri.2.3"
root@mininet-VirtualBox:~/prac_4# python3 client.py
Enter ID : jaehyun

@show_list
sewon ('10.0.0.4', 10081)
jaehyun ('10.0.0.2', 54217)
[]
```

처음 패킷을 보낸 이후 클라이언트에서는 패킷 내부 변수 req = 3 으로 하여 10초에 한 번 씩 패킷을 보냅니다. 서버에서는 @exit 요청을 보내지 않고 사라진 (timeout) 클라이언트를 별도의 thread 상에서 체크합니다. 30초 이상 keep alive 패킷을 보내지 않은 클라이언트는 유저 리스트에서 삭제하고 등록된 모든 유저들에게 업데이트된 정보를 전달합니다.

4. 프로그램 실행 방법

과제 파일은 세 개로 구성됩니다. 기본 client.py, server.py 와 packet.py 입니다. 세 파일 같은 경로에 둡니다.