

HW4 Report

Computer Network
2013313217 조상연

1. Development Environments

1. Python Version: Python 3.7 / Python 3.8
2. IDE: VS Code
3. OS: Mac OS X 10.15.3 / Linux 20.05

2. Getting Started

- 메인 서버에서 서버 먼저 실행

```
python server.py  
# python3 가 기본이 아니라 오류가 난다면  
python3 server.py
```

- NAT 속 VM이나 메인 서버에서 client 실행
- 그 후 Client ID, Server IP 차례로 입력

```
csy@csy:~$ python client.py  
# python3 가 기본이 아니라 오류가 난다면  
csy@csy:~$ python3 client.py  
Client ID>client1 #Client ID 입력  
Server IP>192.168.0.12 #Server IP 입력
```

3. Design

3.1 사용한 모듈

```
import os  
import time  
import threading  
import socket
```

3.2 전체 흐름 및 설계

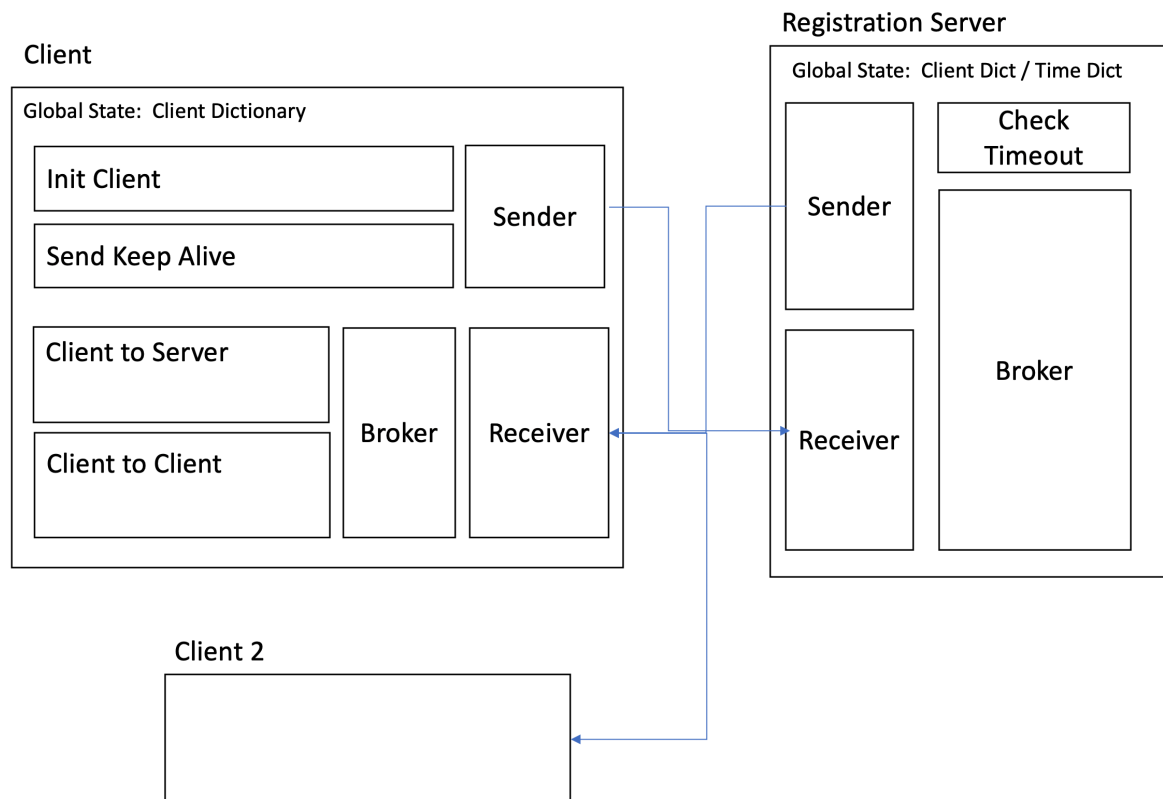


그림 1 전체 흐름도

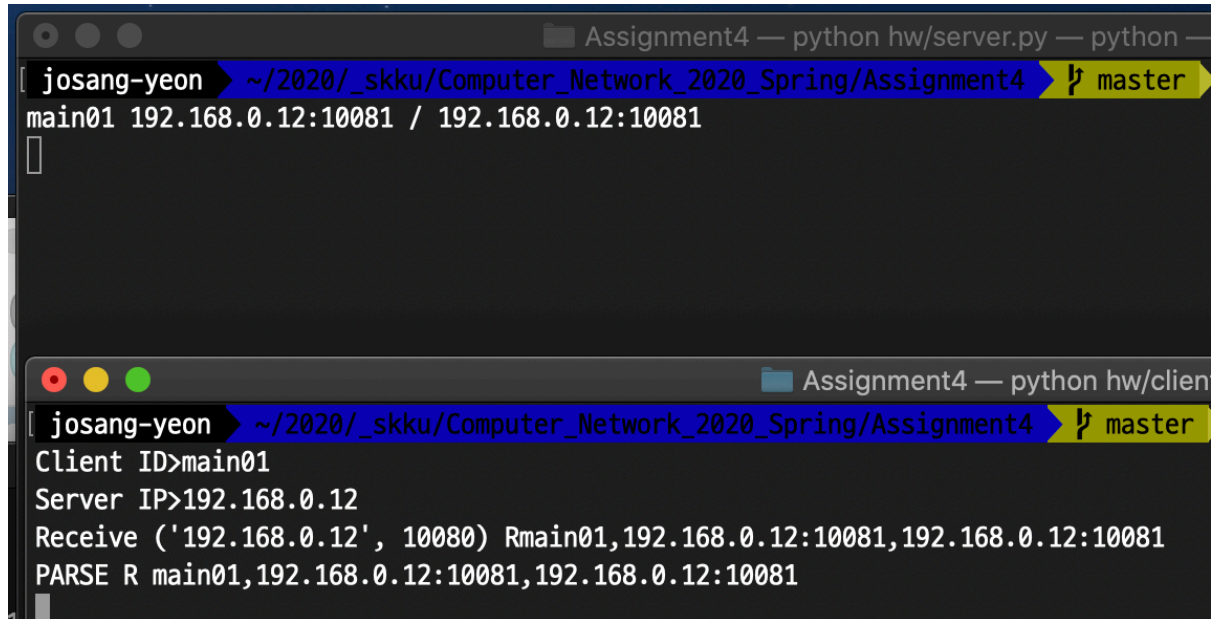
Server 는 우선 전체 Client 의 IP 주소와 최근 응답시간에 대한 Dictionary 를 생성하고 이를 지속적으로 관리한다. 그리고 Receiver 를 통해 들어오는 응답을 Broker 를 통해 분류하고 여기서 다시 sender 를 통해 전체 혹은 하나의 Client 로 전파한다. 또한 Timeout 을 주기적으로 검사하여 30 초가 넘어간 Client 는 전체 상태에서 삭제하고 이 역시 전체 Client 에 전파한다.

Client 는 처음 시작할 때 Private IP 와 Client ID 를 서버로 전송하고 전체 Client 의 정보를 전송 받는다. 전송 받은 상태를 지속적으로 관리하며 또한 10 초에 한번씩 Keep Alive 를 위한 패킷을 Server 측에 전송한다. 또한 다른 Thread 로 관리되는 Receiver 를 이용해 받은 Packet 의 맨 앞자리를 통해 어떤 데이터인지 판별한다. Broker 에선 이러한 헤더 정보 (R(등록), U(삭제))를 통해 payload 를 각 기능별 함수로 전달한다. 또한 client 에서 온 채팅 메시지의 경우 화면에 출력하도록 한다.

채팅을 전송할 땐 Client ID 의 Private IP 와 내 현재 Private IP 를 검사하여 같은 NAT 상인 것이 확인되면 해당 Private IP 로 전송하도록 하였다.

4. Screen Capture

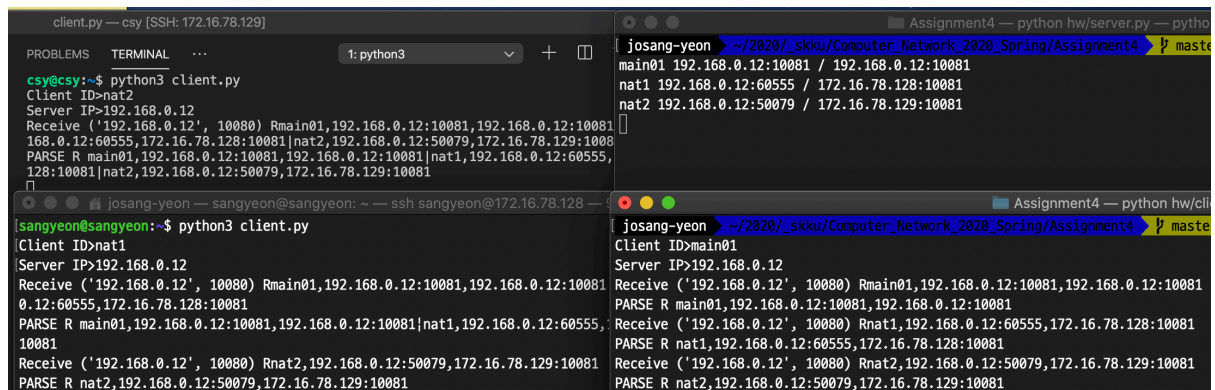
4.1 기본 등록 및 Show List



```
Assignment4 — python hw/server.py — python —
josang-yeon ~/2020/_skku/Computer_Network_2020_Spring/Assignment4 master
main01 192.168.0.12:10081 / 192.168.0.12:10081
[]

Assignment4 — python hw/client.py — python —
josang-yeon ~/2020/_skku/Computer_Network_2020_Spring/Assignment4 master
Client ID>main01
Server IP>192.168.0.12
Receive ('192.168.0.12', 10080) Rmain01,192.168.0.12:10081,192.168.0.12:10081
PARSE R main01,192.168.0.12:10081,192.168.0.12:10081
```

그림 2 Register 서버에 등록



```
client.py — csy [SSH: 172.16.78.129]
PROBLEMS TERMINAL ... 1: python3
csy@csy:~$ python3 client.py
Client ID>nat2
Server IP>192.168.0.12
Receive ('192.168.0.12', 10080) Rmain01,192.168.0.12:10081,192.168.0.12:10081
168.0.12:60555,172.16.78.128:10081|nat2,192.168.0.12:50079,172.16.78.129:10081
PARSE R main01,192.168.0.12:10081,192.168.0.12:10081|nat1,192.168.0.12:60555,
128:10081|nat2,192.168.0.12:50079,172.16.78.129:10081

josang-yeon sangyeon@sangyeon: ~ — ssh sangyeon@172.16.78.128 —
sangyeon@sangyeon:~$ python3 client.py
Client ID>nat1
Server IP>192.168.0.12
Receive ('192.168.0.12', 10080) Rmain01,192.168.0.12:10081,192.168.0.12:10081
0.12:60555,172.16.78.128:10081
PARSE R main01,192.168.0.12:10081,192.168.0.12:10081|nat1,192.168.0.12:60555,
10081
Receive ('192.168.0.12', 10080) Rnat2,192.168.0.12:50079,172.16.78.129:10081
PARSE R nat2,192.168.0.12:50079,172.16.78.129:10081

Assignment4 — python hw/server.py — python —
josang-yeon ~/2020/_skku/Computer_Network_2020_Spring/Assignment4 master
main01 192.168.0.12:10081 / 192.168.0.12:10081
nat1 192.168.0.12:60555 / 172.16.78.128:10081
nat2 192.168.0.12:50079 / 172.16.78.129:10081
[]

Assignment4 — python hw/client.py — python —
josang-yeon ~/2020/_skku/Computer_Network_2020_Spring/Assignment4 master
Client ID>main01
Server IP>192.168.0.12
Receive ('192.168.0.12', 10080) Rmain01,192.168.0.12:10081,192.168.0.12:10081
PARSE R main01,192.168.0.12:10081,192.168.0.12:10081
Receive ('192.168.0.12', 10080) Rnat1,192.168.0.12:60555,172.16.78.128:10081
PARSE R nat1,192.168.0.12:60555,172.16.78.128:10081
Receive ('192.168.0.12', 10080) Rnat2,192.168.0.12:50079,172.16.78.129:10081
PARSE R nat2,192.168.0.12:50079,172.16.78.129:10081
```

그림 3 여러 Client에서 서버에 접속한 모습

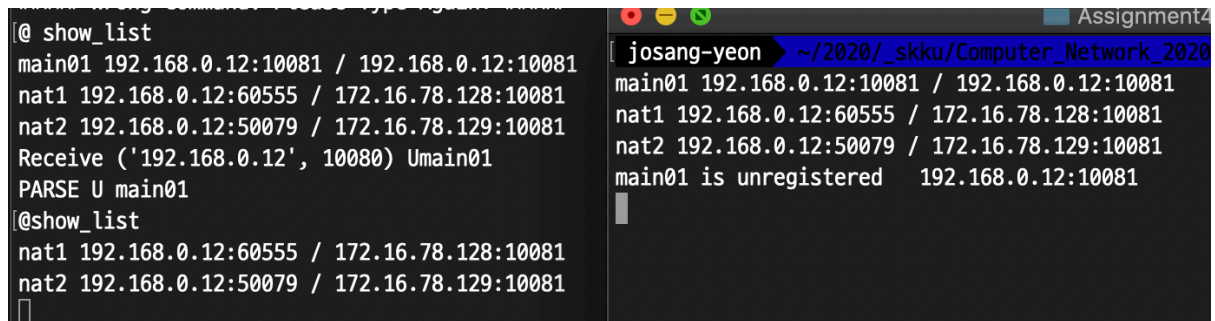
그림 3에서 우측 상단이 Server의 모습입니다. Main01, nat1, nat2 Client가 순차적으로 접속되었고 Public IP와 Private IP 모두를 전달 받은 모습이 보인다.

```
@ show_list
main01 192.168.0.12:10081 / 192.168.0.12:10081
nat1 192.168.0.12:60555 / 172.16.78.128:10081
nat2 192.168.0.12:50079 / 172.16.78.129:10081
```

그림 4 @show_list 구현

Show_list 를 통해 현재 접속한 모든 유저의 Public IP 와 Private IP 를 확인할 수 있습니다.

4.2 Exit & Timeout



The image shows two side-by-side terminal windows. The left window shows the output of the @show_list command, listing main01, nat1, and nat2 with their respective IP addresses. The right window shows the same output, but with an additional line: 'main01 is unregistered 192.168.0.12:10081', indicating that main01 has been removed from the list.

그림 5 main01 의 @Exit 에 따라 전파된 모습, @show_list

Main01 에서 Exit 를 실행한 후 서버에서 unregister 되고, 이를 모든 client 에 전파하였습니다. 전파 받은 client 에선 client list 에서 해당 exit 된 client 를 삭제합니다.

```
main02 192.168.0.12:10081 / 192.168.0.12:10081
main02 is Off-line 192.168.0.12:10081
main03 192.168.0.12:10081 / 192.168.0.12:10081
```

그림 6 main02 의 Time out

이번엔 Main02 로 접속한 후 강제 종료를 하였습니다. 이 경우 30 초 이후에 자동으로 off-line 상태로 인식하고 Unregister 합니다.

4.3 Chat & Same NAT

```
[@chat nat2 hi  
Chat) Send to 172.16.78.129:10081  
Receive ('172.16.78.129', 10081) nat1|hi
```

그림 7 같은 NAT 상의 NAT2 Client 로 전송, Private IP 로 전송한 화면

```
PARSE U main02  
Receive ('172.16.78.128', 10081) nat1|hi  
nat1> hi
```

그림 8 NAT2 에서 NAT1 의 메시지를 받은 화면

같은 NAT 내에선 Private IP 를 통해 전송한 모습입니다. 10081 Port 로 정상적으로 전송되었고 받았습니다. 172.16.78.129:10081 에서 172.16.78.128:10081 로 전송된 모습입니다. (logging 용, 실제 코드에선 삭제)

```
PARSE R main03, 192.168.0.12:10081, 192.168.0.12:10081  
[@chat main03 hihi outsider~  
Chat) Send to 192.168.0.12:10081  
[@show_list  
nat1 192.168.0.12:60555 / 172.16.78.128:10081  
nat2 192.168.0.12:50079 / 172.16.78.129:10081  
main03 192.168.0.12:10081 / 192.168.0.12:10081
```

그림 9 메인서버로 Public IP 를 통해 전송한 모습

메인 서버에 있는 main 03 으로 전송했을 땐 192.168.0.12 라는 Public IP 를 통해 전송하였습니다. 또한 위에서 Unregister 된 main01, main02 가 더이상 show_list 에 나타나지 않음을 알 수 있습니다.