ROW SOCKET 기반 로드밸런서 가이드 라인(CPP)

Server -> loadbalancer ->client OR loadbalancer ->Server ->client 실행순서를 따라주세요

server

```
kjs@kjs:~/iot/test/shared$ ./chat_serv 10000
```

```
kjs@kjs:~/iot/test/shared$ ./chat_serv 10001
```

```
kjs@kjs:~/iot/test/shared$ ./chat_serv 10002
```

loadbalancer

```
kjs@kjs:~/iot/test/cpp$ sudo ./loadbalancerLobin 172.18.136.132 172.18.136.132 100
00 172.18.136.132 10001 172.18.136.132 10002
server ip :172.18.136.132  port : 10000 operating
server ip :172.18.136.132  port : 10001 operating
server ip :172.18.136.132  port : 10002 operating
```

Client(nickname : test1 ->test6 순서대로 접속하고 대화를 입력)

->사진처럼 나오려면 각 client 연결 순서를 지켜야 하지만 client들 끼리는 연결 순서 상관 없이 모두 정상 작동 함

->라운드 로빈 방식에 의해 server1 : (test1, test4), server2: (test2, test5), server3: (test3, test6) 이런식으로 각 서버에 할당되어서 같은 서버에 할당된 클라이언트 들은 서로 대화가 공유됨

```
kjs@kjs:~/iot/test/shared$ ./chat_clnt Loadbalance
rIp 20000 test1
connect() error
kjs@kjs:~/iot/test/shared$ ./chat_clnt 172.18.136.
132 20000 test1
hello
str_len : 14
[test1] hello
```

```
kjs@kjs:~/iot/test/shared$ ./chat_clnt 172.18.136.
132 20000 test2
hi
str_len : 11
[test2] hi
```

```
kjs@kjs:~/iot/test/shared$ ./chat_clnt 172.18.136.
132 20000 test3
hi
str_len : 11
[test3] hi
```

```
kjs@kjs:~/iot/test/shared$ ./chat_clnt Loadbalance
rIp 20000 test1
connect() error
kjs@kjs:~/iot/test/shared$ ./chat_clnt 172.18.136.
132 20000 test1
hello
str_len : 14
[test1] hello
str_len : 17
[test4] hi test1
hello test1
str_len : 20
[test1] hello test1
```

```
kjs@kjs:~/iot/test/shared$ ./chat_clnt 172.18.136.
132 20000 test2
hi
str_len : 11
[test2] hi
str_len : 20
[test5] hello test2
hi test5
str_len : 17
[test2] hi test5
```

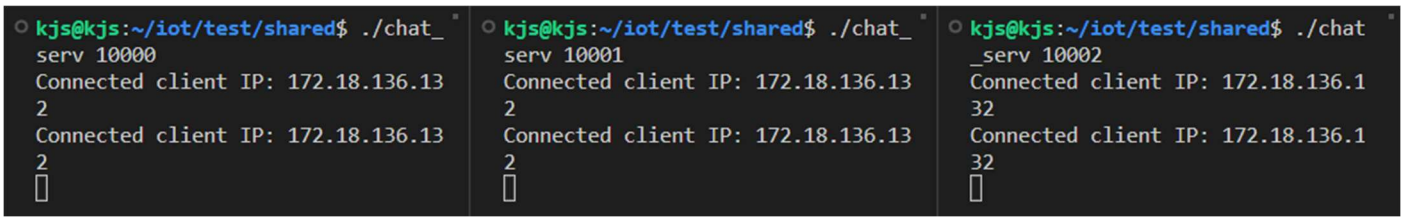
```
kjs@kjs:~/iot/test/shared$ ./chat_clnt 172.18.136.
132 20000 test3
hi
str_len : 11
[test3] hi
str_len : 20
[test6] hello test3
nice to meet test6
str_len : 27
[test3] nice to meet test6
```

```
./chat_clnt
kjs@kjs:~/iot/test/shared$ ./chat_clnt 172.18.136.
132 20000 test4
hi test1
str_len : 17
[test4] hi test1
str_len : 20
[test1] hello test1
```

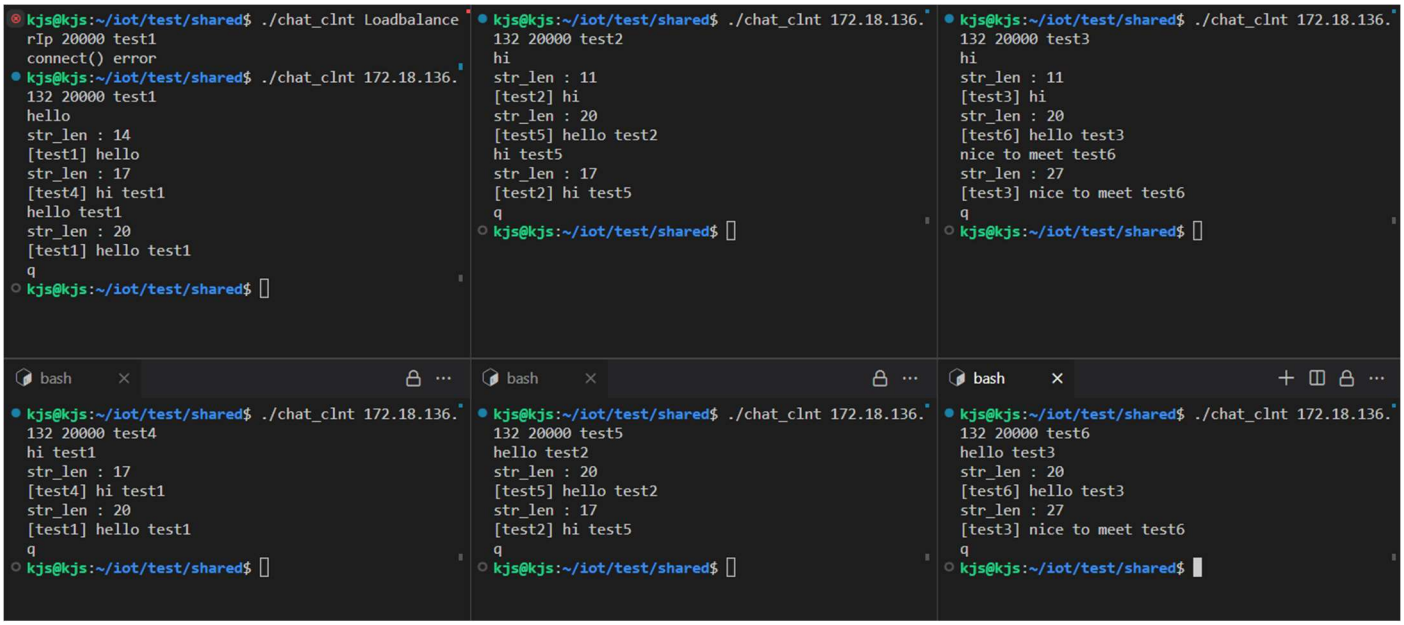
```
./chat_clnt
kjs@kjs:~/iot/test/shared$ ./chat_clnt 172.18.136.
132 20000 test5
hello test2
str_len : 20
[test5] hello test2
str_len : 17
[test2] hi test5
```

```
./chat_clnt
kjs@kjs:~/iot/test/shared$ ./chat_clnt 172.18.136.
132 20000 test6
hello test3
str_len : 20
[test6] hello test3
str_len : 27
[test3] nice to meet test6
```

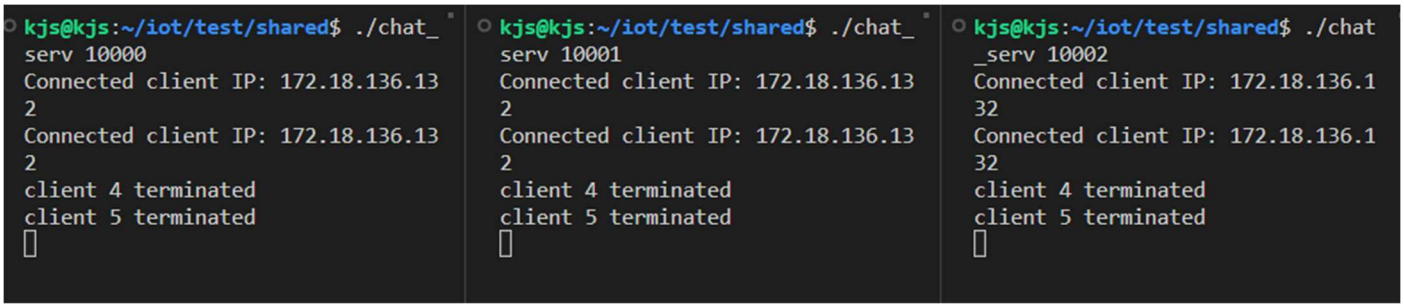
Server(위의 순서대로 접속됨을 보여줌 client 서버에 할당 방식은 라운드 로빈 방식)



Client(nickname : test1 ->test6 순서대로 q를 누르고 순서대로 client 종료)



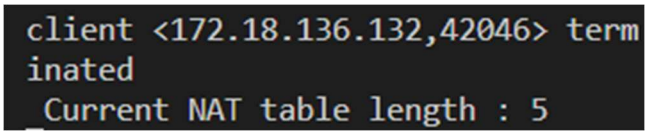
서버에서 각 연결된 client들이 정상 종료됨을 보여줌 ->server는 loadbalancer의 주소를 client로 인식

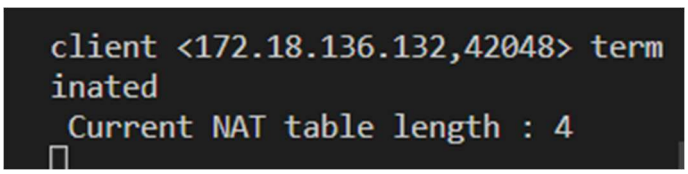


위의 순서대로 종료 할 때 마다 loadbalancer는 client가 종료됨을 인식하고 종료된 client에게 할당 됐던 NAT(Network address translation) table의 row를 삭제함

->터미널에 현재 남아있는 NAT table row 개수를 보여줌

row ex: (client Ip, port : client와 연결된 server Ip, port)





```
client <172.18.136.132,42064> term
inated
 Current NAT table length : 3
```

```
 client <172.18.136.132,42080> term
 inated
  Current NAT table length : 2
```

```
client <172.18.136.132,42096> term
inated
 Current NAT table length : 1
```

```
client <172.18.136.132,49268> term
inated
 Current NAT table length : 0
```