Project 2 - Usage & Test Cases

Introduction

This is an advanced version of project 1 which provides:

- High Available
- Eventually Consistancy

NOTE: Our implementation for delivering activity to client is synchronous, so that you may need to wait a period of time before you can actually receive an activity (default period is 1 second)

System User Guide

System Set Up

There are two JAR files in source code package, `ActivityStreamerClient.jar` and `ActivityStreamerServer.jar`.

Jar file usage:

Server startup

```
usage: ActivityStreamer.Server [-a <arg>] [-activity_check_interval <arg>]
       [-lh <arg>] [-lp <arg>] [-rh <arg>] [-rp <arg>] [-s <arg>]
       [-sync_interval <arg>] [-time_before_reconnect <arg>]
An ActivityStream Server for Unimelb COMP90015

 -a <arg>                          announce interval in milliseconds

 -lh <arg>                         local hostname
 -lp <arg>                         local port number
 -rh <arg>                         remote hostname
 -rp <arg>                         remote port number
 -s <arg>                          secret for the server to use
 -sync_interval <arg>              Provide the interval (in milliseconds,
                                   5000 by default) to sync data amoung
                                   servers.
 -time_before_reconnect <arg>      Provide the time (in milliseconds, 0 by
                                   default) to wait before reconnect if a
                                   server crashes, mainly for testing
                                   eventually consistancy
 -activity_check_interval <arg>    Provide the interval (in milliseconds,
                                   1000 by default) to check whether there
                                   is new activity coming in.
```

Client startup

```
usage: ActivityStreamer.Client [-rh <arg>] [-rp <arg>] [-s
       <arg>] [-u <arg>]
An ActivityStream Client for Unimelb COMP90015
 -rh <arg>   remote hostname
 -rp <arg>   remote port number
 -s  <arg>   secret for username, if not provided, run "register" process
 -u  <arg>   username, if not provided, login as "anonymous".
```

Test Scenario

Per projectspecification, this system is supposed to achieve following functions:

- High Availability: system can reconnect automaticallyafter network partition
- Clients can join (register/login) and leave (logout)the network at any time, Servers can join the network at any time
- Unique Register: a given username can only beregistered once over the server network
- Message ensure: a message sent by a client can reachall clients that are connected to the network at the time
- Message order: all activity messages sent by a clientare delivered in the same order at each receiving client
- Load balancing: clients are evenly distributed overthe servers

Our implementationfor delivering activity to clients is synchronous, so you may need to wait aperiod of time before you can actual receive an activity, default period is 1second.

Six scenarios havebeen designed for test case.

NOTE: All test screenshot shown below are using our new version client which is more useful for debug. If you want to test the last version client(client of project1) , change the command to user `ActivityStreamerClient-old.jar` as the client jar package. The old version also passed all these test cases.

High Available

Test Case

1. Start 4 servers

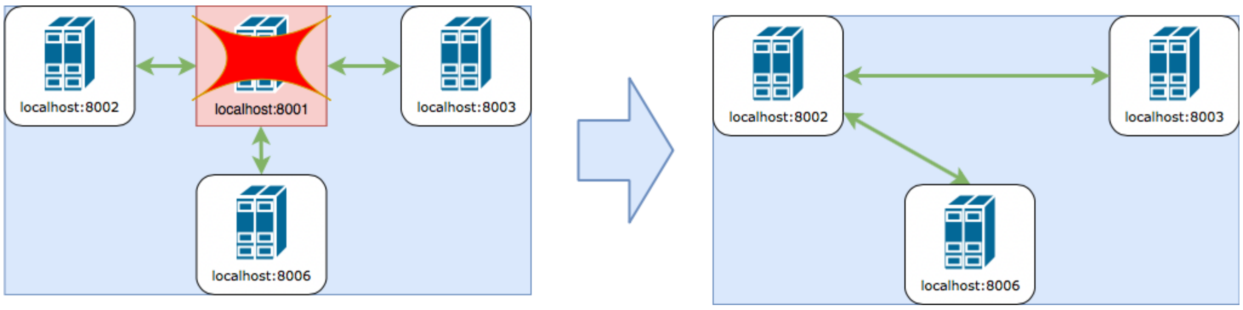
```
java -jar ActivityStreamerServer.jar -lh localhost -lp 8001 -s abc
java -jar ActivityStreamerServer.jar -lh localhost -lp 8002 -s abc -rh
localhost -rp 8001
java -jar ActivityStreamerServer.jar -lh localhost -lp 8003 -s abc -rh
localhost -rp 8001
java -jar ActivityStreamerServer.jar -lh localhost -lp 8006 -s abc -rh
localhost -rp 8001
```

2. Force quit server 8001

Click ***Close*** icon in _server UI_ or press ***CTRL+C*** in _command line_

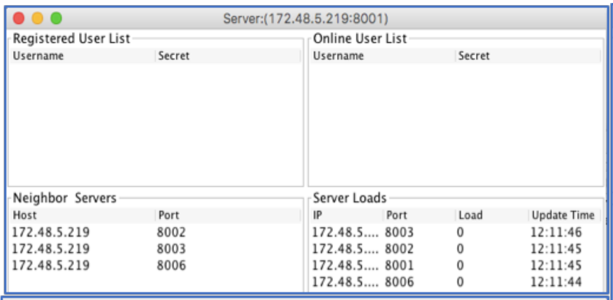
Expected Result

After that you will see server 8002, 8003, 8006 will automatically connected. The picture shows a successful situation (the one, 8002, that takes 8001's place may vary).



Screentshot:

After 4 servers were started:



Server:(172.48.5.219:8001)

Registered User List

Username	Secret

Online User List

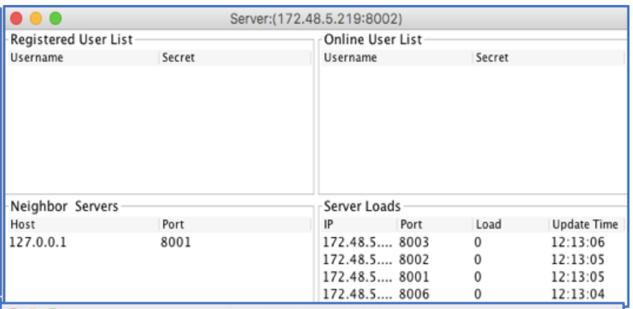
Username	Secret

Neighbor Servers

Host	Port
172.48.5.219	8002
172.48.5.219	8003
172.48.5.219	8006

Server Loads

IP	Port	Load	Update Time
172.48.5....	8003	0	12:11:46
172.48.5....	8002	0	12:11:45
172.48.5....	8001	0	12:11:45
172.48.5....	8006	0	12:11:44



Server:(172.48.5.219:8002)

Registered User List

Username	Secret

Online User List

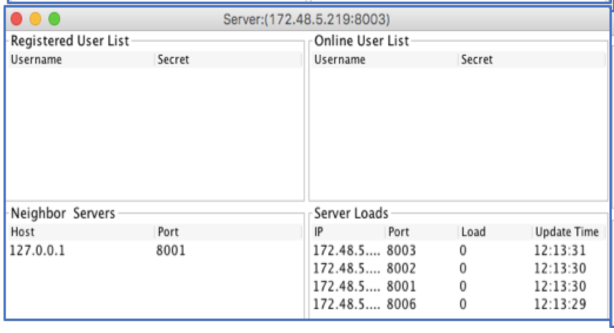
Username	Secret

Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
172.48.5....	8003	0	12:13:06
172.48.5....	8002	0	12:13:05
172.48.5....	8001	0	12:13:05
172.48.5....	8006	0	12:13:04



Server:(172.48.5.219:8003)

Registered User List

Username	Secret

Online User List

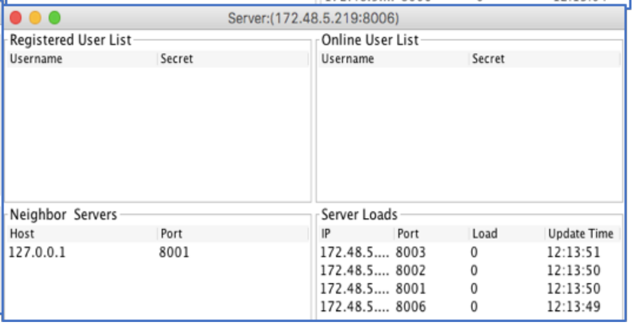
Username	Secret

Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
172.48.5....	8003	0	12:13:31
172.48.5....	8002	0	12:13:30
172.48.5....	8001	0	12:13:30
172.48.5....	8006	0	12:13:29



Server:(172.48.5.219:8006)

Registered User List

Username	Secret

Online User List

Username	Secret

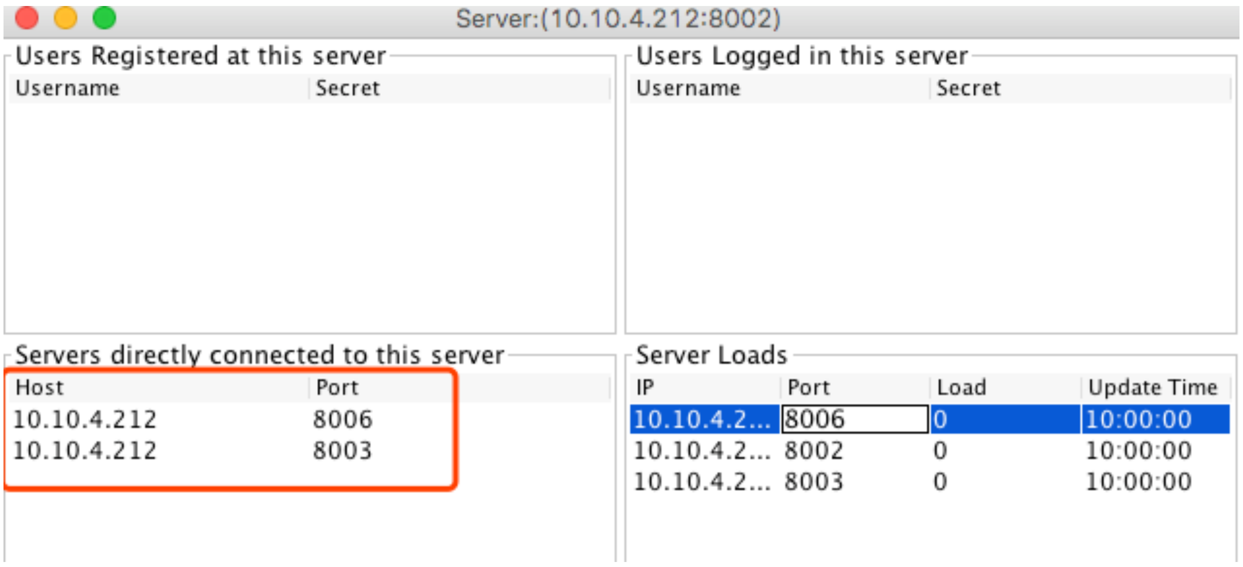
Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
172.48.5....	8003	0	12:13:51
172.48.5....	8002	0	12:13:50
172.48.5....	8001	0	12:13:50
172.48.5....	8006	0	12:13:49

After force quit server 8001:



Server:(10.10.4.212:8002)

Users Registered at this server

Username	Secret

Users Logged in this server

Username	Secret

Servers directly connected to this server

Host	Port
10.10.4.212	8006
10.10.4.212	8003

Server Loads

IP	Port	Load	Update Time
10.10.4.2...	8006	0	10:00:00
10.10.4.2...	8002	0	10:00:00
10.10.4.2...	8003	0	10:00:00

Testing Result:

As expected.

Client can join and leave any time

Test case

1. Start the very first server

```
java -jar ActivityStreamerServer.jar -lh localhost -lp 8001 -s abc
```

2. Register a user at 8001 and ***remember*** its secret

```
java -jar ActivityStreamerClient.jar -u user1 -rp 8001 -rh localhost
```

3. Quit client of last step (close GUI or press CTRL+C in terminal)
4. Start a new server and connect it to 8001

```
java -jar ActivityStreamerServer.jar -lh localhost -lp 8002 -s abc -rh
localhost -rp 8001
```

5. Login user1 at new server 8002 (replace `$secret` by actual secret)

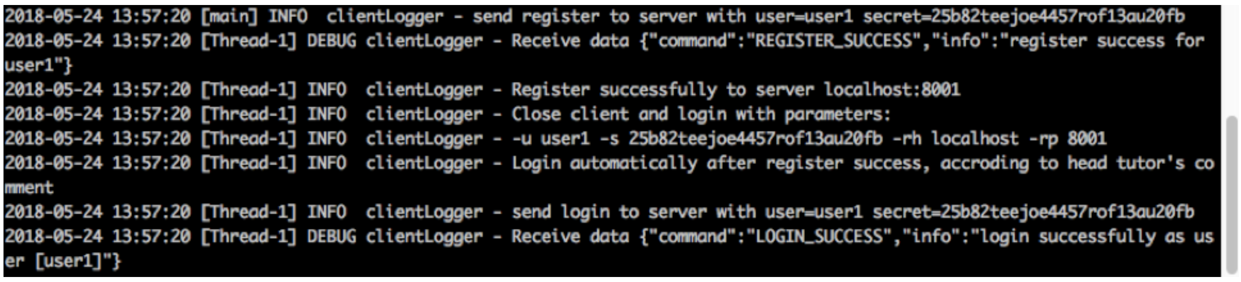
```
java -jar ActivityStreamerClient.jar -u user1 -s $secret -rp 8002 -rh
localhost
```

Expected Result

User1 should login on new server 8002 successfully, and all data of 8002 should be consistent with 8001.

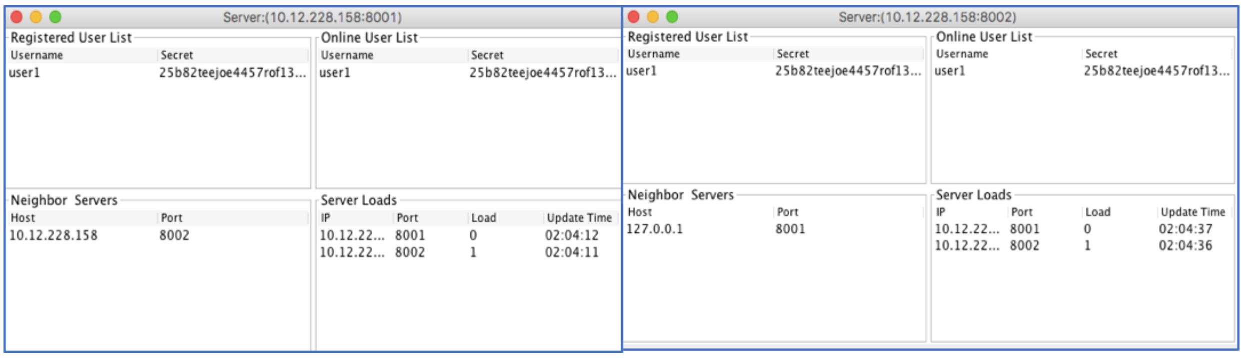
Screentshot

Register success and auto login with given secret



```
2018-05-24 13:57:20 [main] INFO  clientLogger - send register to server with user=user1 secret=25b82teejoe4457rof13au20fb
2018-05-24 13:57:20 [Thread-1] DEBUG clientLogger - Receive data {"command":"REGISTER_SUCCESS","info":"register success for
user1"}
2018-05-24 13:57:20 [Thread-1] INFO  clientLogger - Register successfully to server localhost:8001
2018-05-24 13:57:20 [Thread-1] INFO  clientLogger - Close client and login with parameters:
2018-05-24 13:57:20 [Thread-1] INFO  clientLogger - -u user1 -s 25b82teejoe4457rof13au20fb -rh localhost -rp 8001
2018-05-24 13:57:20 [Thread-1] INFO  clientLogger - Login automatically after register success, accroding to head tutor's co
mment
2018-05-24 13:57:20 [Thread-1] INFO  clientLogger - send login to server with user=user1 secret=25b82teejoe4457rof13au20fb
2018-05-24 13:57:20 [Thread-1] DEBUG clientLogger - Receive data {"command":"LOGIN_SUCCESS","info":"login successfully as us
er [user1]"}
```

User1 relogin on 8002 (user1 login successfully, 8001 and 8002 is consistent)



Server:(10.12.228.158:8001)

Registered User List

Username	Secret
user1	25b82teejoe4457rof13...

Online User List

Username	Secret
user1	25b82teejoe4457rof13...

Neighbor Servers

Host	Port
10.12.228.158	8002

Server Loads

IP	Port	Load	Update Time
10.12.22...	8001	0	02:04:12
10.12.22...	8002	1	02:04:11

Server:(10.12.228.158:8002)

Registered User List

Username	Secret
user1	25b82teejoe4457rof13...

Online User List

Username	Secret
user1	25b82teejoe4457rof13...

Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
10.12.22...	8001	0	02:04:37
10.12.22...	8002	1	02:04:36

Testing Result

Result as expected.

Server can join at any time

Test case

1. start the very first server

```
java -jar ActivityStreamerServer.jar -lh localhost -lp 8001 -s abc
```

2. register a user at this server and remember its secret.

```
java -jar ActivityStreamerClient.jar -u user1 -rp 8001 -rh localhost
```

3. Quit client of step 2
4. start a new server connecting to server 8001

```
java -jar ActivityStreamerServer.jar -lh localhost -lp 8002 -s abc -rh
localhost -rp 8001
```

5. Login user1 at the new server (8002) by replace `$secret` of below script

```
java -jar ActivityStreamerClient.jar -u user1 -s vl3et80v8mmn3ho6dm93hjqgg0
-rp 8002 -rh localhost
```

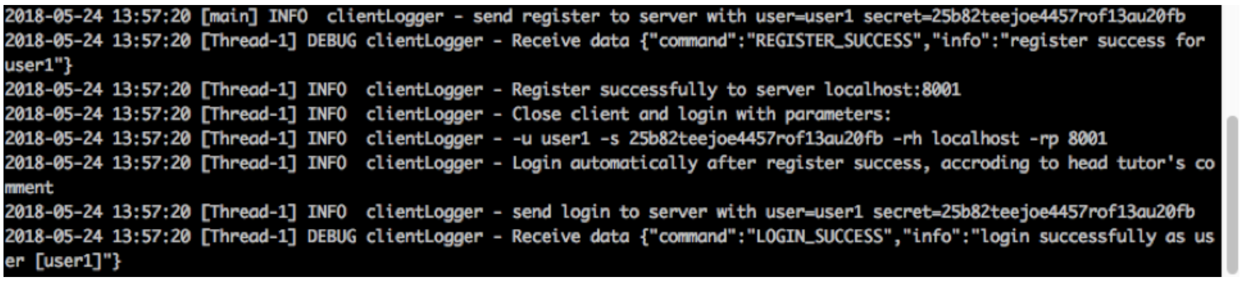
Expected Result

- user1 should login successfully at new server (8002) and all data of 8002 should be synced with 8001
- From test case Message ensure we can also see that:

  > user A is online at the time T, when a activity is sent by some other user B and A loses its connection it can receive this message.
  >
  > When user A reconnects to any server of this system, it can also receive this lost message.

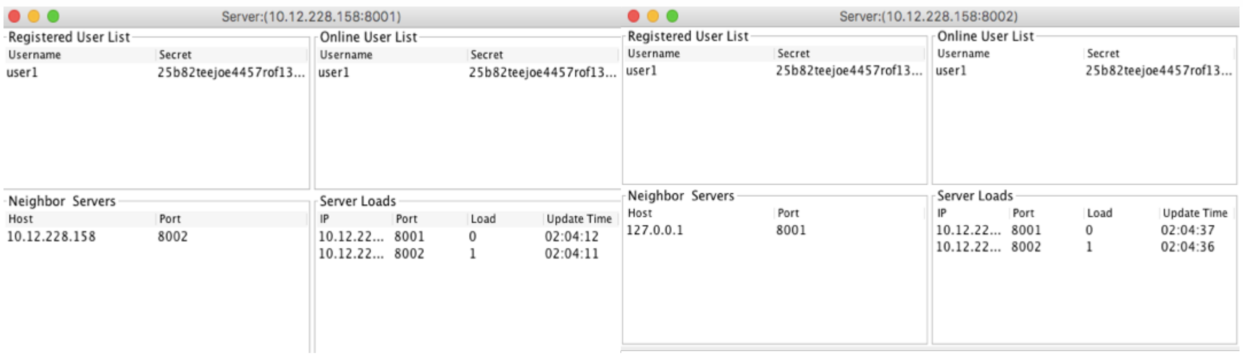
Screentshot:

Snapshot of register success and auto login with given secret



```
2018-05-24 13:57:20 [main] INFO  clientLogger - send register to server with user=user1 secret=25b82teejoe4457rof13au20fb
2018-05-24 13:57:20 [Thread-1] DEBUG clientLogger - Receive data {"command":"REGISTER_SUCCESS","info":"register success for
user1"}
2018-05-24 13:57:20 [Thread-1] INFO  clientLogger - Register successfully to server localhost:8001
2018-05-24 13:57:20 [Thread-1] INFO  clientLogger - Close client and login with parameters:
2018-05-24 13:57:20 [Thread-1] INFO  clientLogger - -u user1 -s 25b82teejoe4457rof13au20fb -rh localhost -rp 8001
2018-05-24 13:57:20 [Thread-1] INFO  clientLogger - Login automaticly after register success, accroding to head tutor's co
mment
2018-05-24 13:57:20 [Thread-1] INFO  clientLogger - send login to server with user=user1 secret=25b82teejoe4457rof13au20fb
2018-05-24 13:57:20 [Thread-1] DEBUG clientLogger - Receive data {"command":"LOGIN_SUCCESS","info":"login successfully as us
er [user1]"}
```

Snapshot of user1 relogin on 8002 (user1 login successfully, 8001 and 8002 is consistent)



Server:(10.12.228.158:8001)

Registered User List

Username	Secret
user1	25b82teejoe4457rof13...

Online User List

Username	Secret
user1	25b82teejoe4457rof13...

Neighbor Servers

Host	Port
10.12.228.158	8002

Server Loads

IP	Port	Load	Update Time
10.12.22...	8001	0	02:04:12
10.12.22...	8002	1	02:04:11

Server:(10.12.228.158:8002)

Registered User List

Username	Secret
user1	25b82teejoe4457rof13...

Online User List

Username	Secret
user1	25b82teejoe4457rof13...

Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
10.12.22...	8001	0	02:04:37
10.12.22...	8002	1	02:04:36

Testing Result

Result as expected.

Unique Register

Test case

1. start several servers, say 3

```
java -jar ActivityStreamerServer.jar -lh localhost -lp 8001 -s abc
java -jar ActivityStreamerServer.jar -lh localhost -lp 8002 -s abc -rh
localhost -rp 8001
java -jar ActivityStreamerServer.jar -lh localhost -lp 8002 -s abc -rh
localhost -rp 8001
```

1. register user1 at server 8001

```
java -jar ActivityStreamerClient.jar -u user1 -rp 8001 -rh localhost
```

1. try to register user1 at another server, say 8002

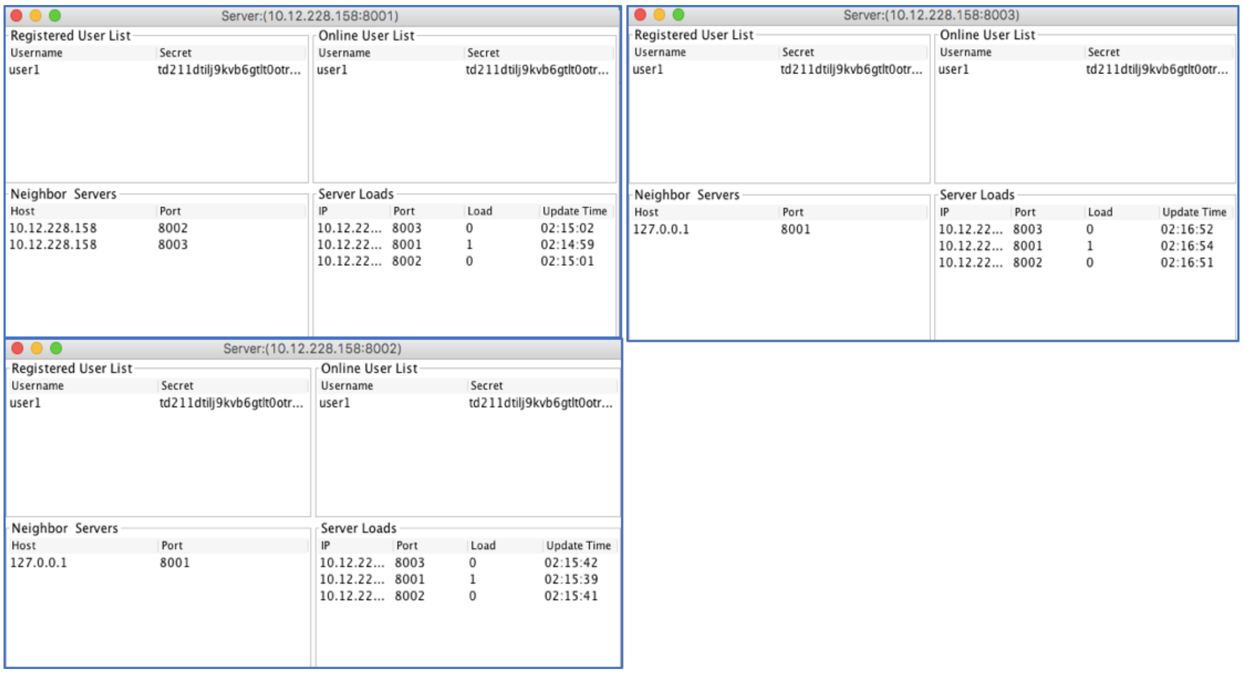
```
java -jar ActivityStreamerClient.jar -u user1 -rp 8002 -rh localhost
```

Expected Result

- the registration of step 3 (server 8002) will fail with error like "user already exists".

Screentshot

Snapshot of 3 servers’ GUI



Server:(10.12.228.158:8001)

Registered User List

Username	Secret
user1	td211dtilj9kvb6gtlt0otr...

Online User List

Username	Secret
user1	td211dtilj9kvb6gtlt0otr...

Neighbor Servers

Host	Port
10.12.228.158	8002
10.12.228.158	8003

Server Loads

IP	Port	Load	Update Time
10.12.22...	8003	0	02:15:02
10.12.22...	8001	1	02:14:59
10.12.22...	8002	0	02:15:01

Server:(10.12.228.158:8003)

Registered User List

Username	Secret
user1	td211dtilj9kvb6gtlt0otr...

Online User List

Username	Secret
user1	td211dtilj9kvb6gtlt0otr...

Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
10.12.22...	8003	0	02:16:52
10.12.22...	8001	1	02:16:54
10.12.22...	8002	0	02:16:51

Server:(10.12.228.158:8002)

Registered User List

Username	Secret
user1	td211dtilj9kvb6gtlt0otr...

Online User List

Username	Secret
user1	td211dtilj9kvb6gtlt0otr...

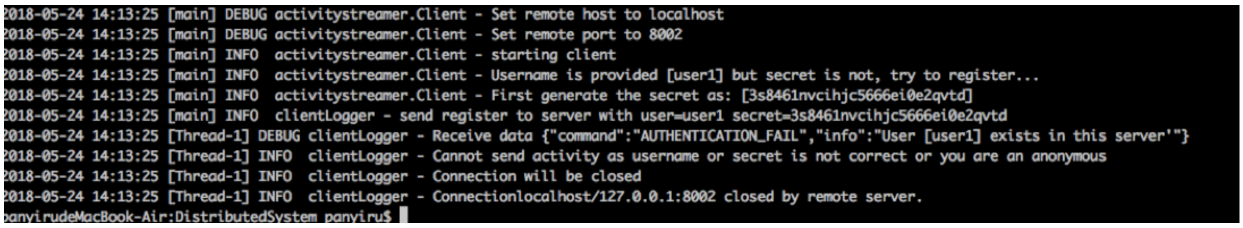
Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
10.12.22...	8003	0	02:15:42
10.12.22...	8001	1	02:15:39
10.12.22...	8002	0	02:15:41

Snapshot of error message (user1 already exists in server)



```
2018-05-24 14:13:25 [main] DEBUG activitystreamer.Client - Set remote host to localhost
2018-05-24 14:13:25 [main] DEBUG activitystreamer.Client - Set remote port to 8002
2018-05-24 14:13:25 [main] INFO  activitystreamer.Client - starting client
2018-05-24 14:13:25 [main] INFO  activitystreamer.Client - Username is provided [user1] but secret is not, try to register...
2018-05-24 14:13:25 [main] INFO  activitystreamer.Client - First generate the secret as: [3s8461nvcihjc5666ei0e2qvtd]
2018-05-24 14:13:25 [main] INFO  clientLogger - send register to server with user=user1 secret=3s8461nvcihjc5666ei0e2qvtd
2018-05-24 14:13:25 [Thread-1] DEBUG clientLogger - Receive data {"command":"AUTHENTICATION_FAIL","info":"User [user1] exists in this server'"}
2018-05-24 14:13:25 [Thread-1] INFO  clientLogger - Cannot send activity as username or secret is not correct or you are an anonymous
2018-05-24 14:13:25 [Thread-1] INFO  clientLogger - Connection will be closed
2018-05-24 14:13:25 [Thread-1] INFO  clientLogger - Connectionlocalhost/127.0.0.1:8002 closed by remote server.
panyirudeMacBook-Air:DistributedSystem panyiru$
```

Testing Result

Result as excepted.

Message ensure

Test case

In order to simulate message loss case, let us start servers with a parameter to ***delay*** the reconnection function.

1. Start 4 servers with `time_before_reconnect=10000 (10 seconds)`

```
# start the very first server, which will be terminated
java -jar ActivityStreamerServer.jar -lh localhost -lp 8001 -s abc
# start other servers
java -jar ActivityStreamerServer.jar -lh localhost -lp 8002 -s abc -rh
localhost -rp 8001 -time_before_reconnect 10000
java -jar ActivityStreamerServer.jar -lh localhost -lp 8003 -s abc -rh
localhost -rp 8001 -time_before_reconnect 10000
java -jar ActivityStreamerServer.jar -lh localhost -lp 8006 -s abc -rh
localhost -rp 8001 -time_before_reconnect 10000
```

2. Connect 3 clients to 3 different servers

Note: Please record the secret of user1 for future use

```
# Keept the secret after register successfully
java -jar ActivityStreamerClient.jar -u user1 -rp 8001 -rh localhost
# you can just run below 2 clients and no need to record their secretrs
java -jar ActivityStreamerClient.jar -u user2 -rp 8002 -rh localhost
java -jar ActivityStreamerClient.jar -u user3 -rp 8003 -rh localhost
```

1. Terminate server 8001 and send a message from user2 within 10 seconds

- Click ***Close*** icon in *server UI* or press ***CTRL+C*** in *command line* (user 1 will lose connection)
- Send message `{"a":1}` from user2.
- Wait for reconnection happens (10 seconds)

4. Reconnect user1 to any working server, let's say 8006

Replace `$secret` of below script with the secret from step 2.

```
java -jar ActivityStreamerClient.jar -u user1 -s $secret -rp 8006 -rh
localhost
```

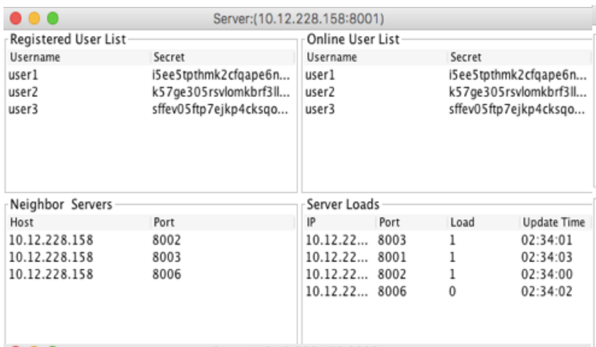
Expected Result

- user3 will receive the activity of user2 after reconection is done (about 10 seconds after disconnection)
- user1 will receive the activity of user2 after relogin to server 8006

> user A is online at the time T, when a activity is sent by some other user B and A loses its connection it can receive this message.
>
> When user A reconnects to any server of this system, it can also receive this lost message.

Screentshot

clients login on 8001,8002, 8003 respectively


Server:(10.12.228.158:8001)

Registered User List

Username	Secret
user1	i5ee5tpthmk2cfqape6n...
user2	k57ge305rsvlomkbrf3ll...
user3	sffev05ftp7ejkp4cksqo...

Online User List

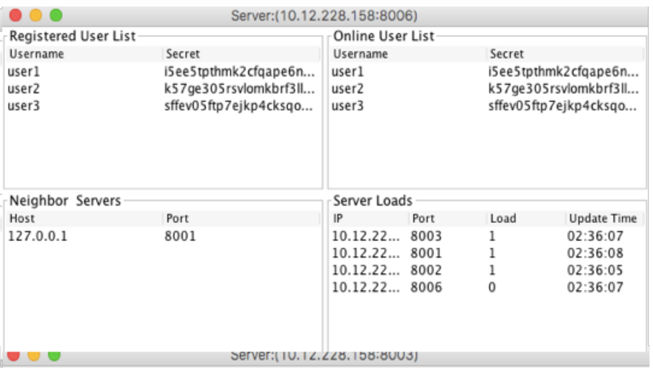
Username	Secret
user1	i5ee5tpthmk2cfqape6n...
user2	k57ge305rsvlomkbrf3ll...
user3	sffev05ftp7ejkp4cksqo...

Neighbor Servers

Host	Port
10.12.228.158	8002
10.12.228.158	8003
10.12.228.158	8006

Server Loads

IP	Port	Load	Update Time
10.12.22...	8003	1	02:34:01
10.12.22...	8001	1	02:34:03
10.12.22...	8002	1	02:34:00
10.12.22...	8006	0	02:34:02


Server:(10.12.228.158:8006)

Registered User List

Username	Secret
user1	i5ee5tpthmk2cfqape6n...
user2	k57ge305rsvlomkbrf3ll...
user3	sffev05ftp7ejkp4cksqo...

Online User List

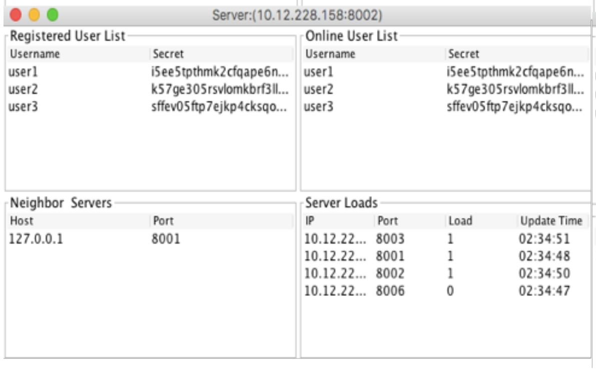
Username	Secret
user1	i5ee5tpthmk2cfqape6n...
user2	k57ge305rsvlomkbrf3ll...
user3	sffev05ftp7ejkp4cksqo...

Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
10.12.22...	8003	1	02:36:07
10.12.22...	8001	1	02:36:08
10.12.22...	8002	1	02:36:05
10.12.22...	8006	0	02:36:07


Server:(10.12.228.158:8002)

Registered User List

Username	Secret
user1	i5ee5tpthmk2cfqape6n...
user2	k57ge305rsvlomkbrf3ll...
user3	sffev05ftp7ejkp4cksqo...

Online User List

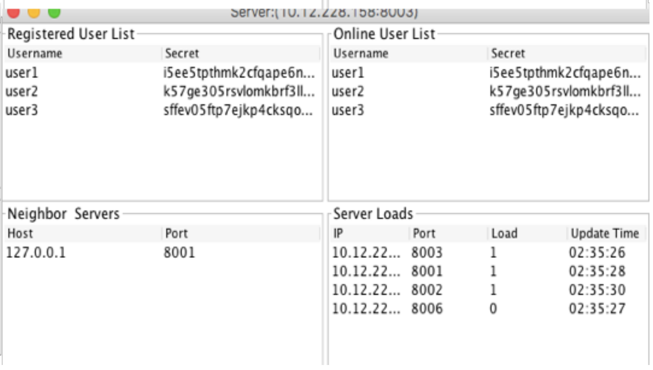
Username	Secret
user1	i5ee5tpthmk2cfqape6n...
user2	k57ge305rsvlomkbrf3ll...
user3	sffev05ftp7ejkp4cksqo...

Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
10.12.22...	8003	1	02:34:51
10.12.22...	8001	1	02:34:48
10.12.22...	8002	1	02:34:50
10.12.22...	8006	0	02:34:47


Server:(10.12.228.158:8003)

Registered User List

Username	Secret
user1	i5ee5tpthmk2cfqape6n...
user2	k57ge305rsvlomkbrf3ll...
user3	sffev05ftp7ejkp4cksqo...

Online User List

Username	Secret
user1	i5ee5tpthmk2cfqape6n...
user2	k57ge305rsvlomkbrf3ll...
user3	sffev05ftp7ejkp4cksqo...

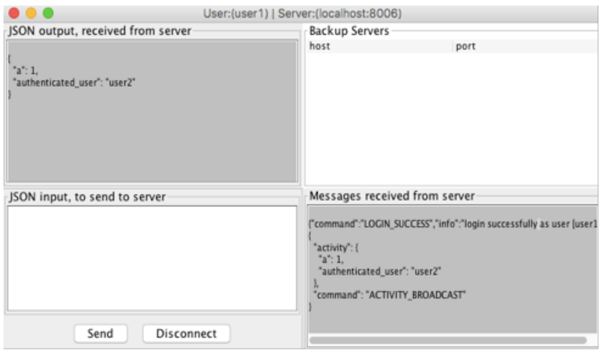
Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
10.12.22...	8003	1	02:35:26
10.12.22...	8001	1	02:35:28
10.12.22...	8002	1	02:35:30
10.12.22...	8006	0	02:35:27

clients after reconnection(user1, user2, user3 all received activity from user2)


User:(user1) | Server:(localhost:8006)

JSON output, received from server

```
{
  "a": 1,
  "authenticated_user": "user2"
}
```

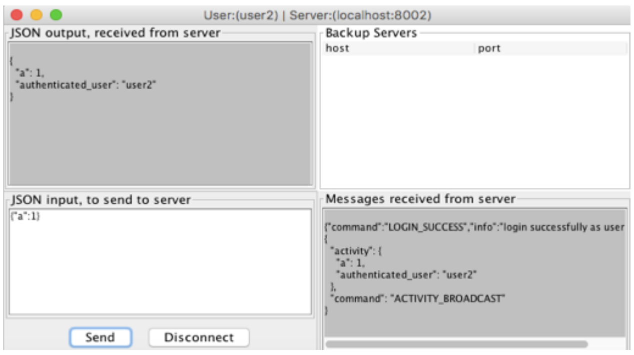
Backup Servers

host	port

JSON input, to send to server

Messages received from server

```
{"command":"LOGIN_SUCCESS","info":"login successfully as user [user1
{
  "activity": {
    "a": 1,
    "authenticated_user": "user2"
  },
  "command": "ACTIVITY_BROADCAST"
}
```


User:(user2) | Server:(localhost:8002)

JSON output, received from server

```
{
  "a": 1,
  "authenticated_user": "user2"
}
```

Backup Servers

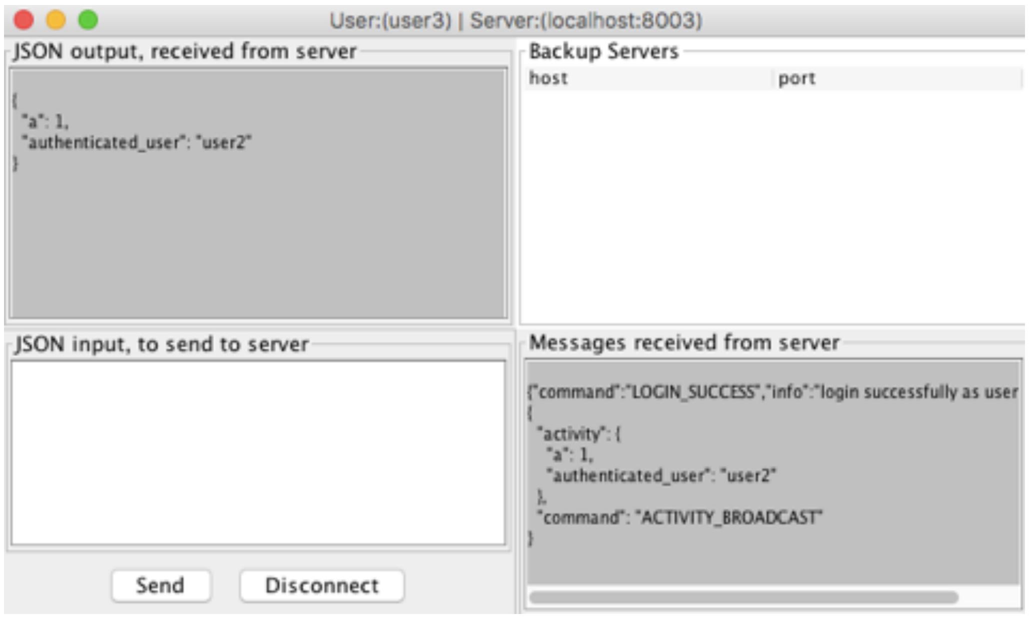
host	port

JSON input, to send to server

```
{"a":1}
```

Messages received from server

```
{"command":"LOGIN_SUCCESS","info":"login successfully as user
{
  "activity": {
    "a": 1,
    "authenticated_user": "user2"
  },
  "command": "ACTIVITY_BROADCAST"
}
```


User:(user3) | Server:(localhost:8003)

JSON output, received from server

```
{
  "a": 1,
  "authenticated_user": "user2"
}
```

Backup Servers

host	port

JSON input, to send to server

Messages received from server

```
{"command":"LOGIN_SUCCESS","info":"login successfully as user
{
  "activity": {
    "a": 1,
    "authenticated_user": "user2"
  },
  "command": "ACTIVITY_BROADCAST"
}
```

Testing Result

Result as excepted.

Message order

In order to simulate message disorder case, let us use a ***telnet session*** to simulate a ***server*** and make the order checking period a littler longer with `activity_check_interval=10000`. Fake messages will be broadcasted by the telnet server with a hooker ***"backTime"*** to set the send time of fake messages to be a time in the past.

> 'timeBack' field is a back door used for this kind of testing. If that field exists in an ActivityBroadcast message, then set the `sendTime` of this activity to `currentTimeInMillis() - timeBack`

Operations

1. Start 1 server with `activity_check_interval=10000 (10 seconds)`

```
java -jar ActivityStreamerServer.jar -activity_check_interval 10000 -lh
localhost -lp 8001 -s abc
```

2. Start a normal client connecting to server 1

```
java -jar ActivityStreamerClient.jar -u user1 -rp 8001 -rh localhost
```

3. Start a terminate and using telnet to simulate a client in following steps

- start telent session

```
telnet localhost 8001
```

- paste below string to authenticate this "server" with server 8001

```
{"command":"AUTHENTICATE","serverId":"serverId01","secret":"abc","host":"lo
calhost","port":8002}
```

- Broadcast 2 "fake" activities (**within 10 seconds**) by pasting below 2 string **separately(one by one)** into telnet session to simulate disordered message.

You can ignore the message telnet session receives. All of them are used by real server to sync data.

>

Message 1: a "fake" message that was sent 0 second ago

```
{"id":0,"activity":
{"message_num":2,"authenticated_user":"user2"},"isDelivered":false,"command
":"ACTIVITY_BROADCAST","timeBack":0}
```

Message 2: a "fake" message that was sent 10 seconds ago, which is early than preious one.

```
{"id":0,"activity":
{"message_num":1,"authenticated_user":"user2"},"isDelivered":false,"command
":"ACTIVITY_BROADCAST","timeBack":10000}
```

Expected Result

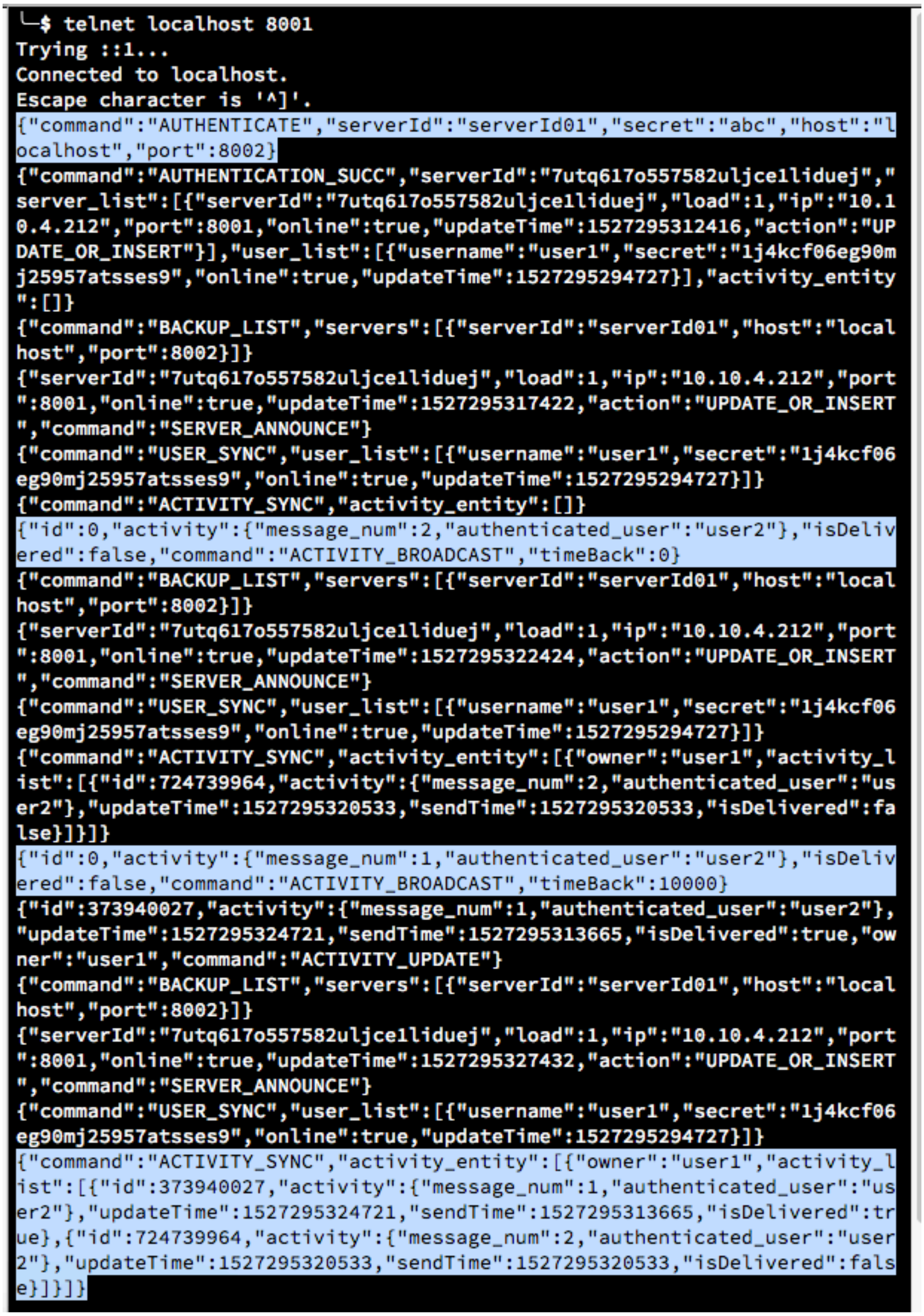
- After waiting ***10-20*** seconds, user1 (normal client with GUI) will receive 2 activities in order (message_num=1 first and then message_num=2) separately.

In real server, this order checking period can be relately shorter, like 0.5 or 1 second.

Screentshot

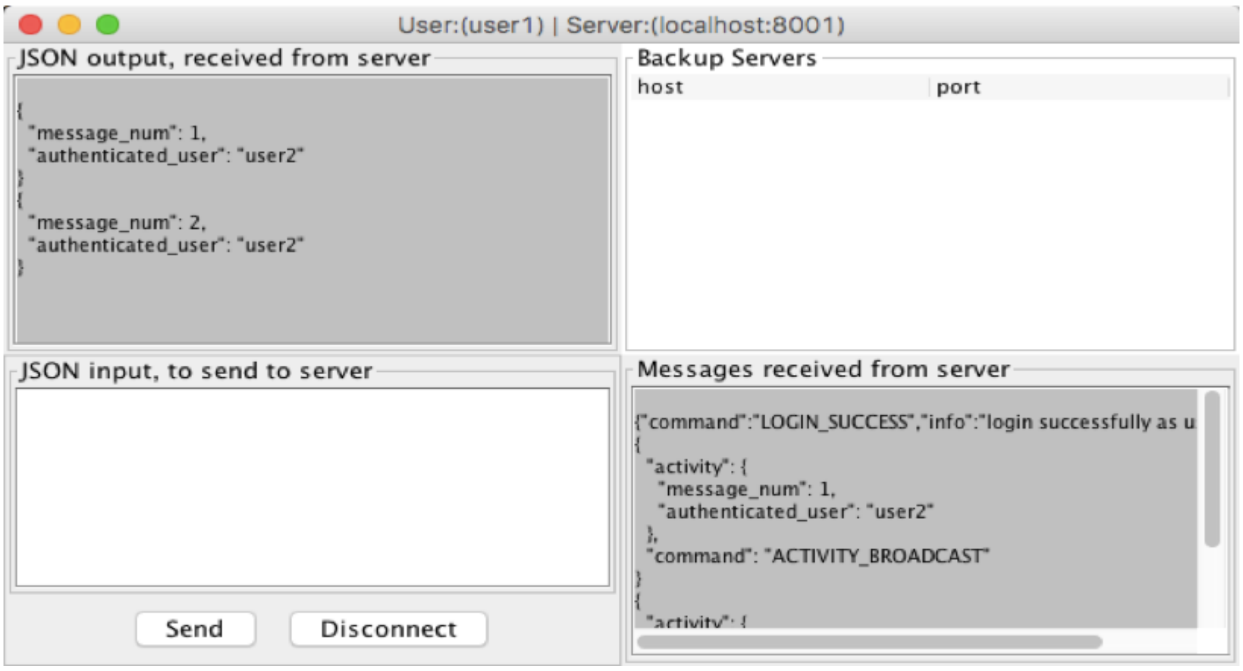
Telnet session input (in white, you can ignore other information, they are sync message from server)

The 3rd white block shows the ordered message list, in which the first is the one with message_num=1



```
└─$ telnet localhost 8001
Trying ::1...
Connected to localhost.
Escape character is '^]'.
{"command":"AUTHENTICATE","serverId":"serverId01","secret":"abc","host":"l
ocalhost","port":8002}
{"command":"AUTHENTICATION_SUCC","serverId":"7utq617o557582uljce1liduej","
server_list":[{"serverId":"7utq617o557582uljce1liduej","load":1,"ip":"10.1
0.4.212","port":8001,"online":true,"updateTime":1527295312416,"action":"UP
DATE_OR_INSERT"}],"user_list":[{"username":"user1","secret":"1j4kcf06eg90m
j25957atsses9","online":true,"updateTime":1527295294727}],"activity_entity
":[]}
{"command":"BACKUP_LIST","servers":[{"serverId":"serverId01","host":"local
host","port":8002}]}
{"serverId":"7utq617o557582uljce1liduej","load":1,"ip":"10.10.4.212","port
":8001,"online":true,"updateTime":1527295317422,"action":"UPDATE_OR_INSERT
","command":"SERVER_ANNOUNCE"}
{"command":"USER_SYNC","user_list":[{"username":"user1","secret":"1j4kcf06
eg90mj25957atsses9","online":true,"updateTime":1527295294727}]}
{"command":"ACTIVITY_SYNC","activity_entity":[]}
{"id":0,"activity":{"message_num":2,"authenticated_user":"user2"},"isDeliv
ered":false,"command":"ACTIVITY_BROADCAST","timeBack":0}
{"command":"BACKUP_LIST","servers":[{"serverId":"serverId01","host":"local
host","port":8002}]}
{"serverId":"7utq617o557582uljce1liduej","load":1,"ip":"10.10.4.212","port
":8001,"online":true,"updateTime":1527295322424,"action":"UPDATE_OR_INSERT
","command":"SERVER_ANNOUNCE"}
{"command":"USER_SYNC","user_list":[{"username":"user1","secret":"1j4kcf06
eg90mj25957atsses9","online":true,"updateTime":1527295294727}]}
{"command":"ACTIVITY_SYNC","activity_entity":[{"owner":"user1","activity_l
ist":[{"id":724739964,"activity":{"message_num":2,"authenticated_user":"us
er2"},"updateTime":1527295320533,"sendTime":1527295320533,"isDelivered":fa
lse}]}]}
{"id":0,"activity":{"message_num":1,"authenticated_user":"user2"},"isDeliv
ered":false,"command":"ACTIVITY_BROADCAST","timeBack":10000}
{"id":373940027,"activity":{"message_num":1,"authenticated_user":"user2"},
"updateTime":1527295324721,"sendTime":1527295313665,"isDelivered":true,"ow
ner":"user1","command":"ACTIVITY_UPDATE"}
{"command":"BACKUP_LIST","servers":[{"serverId":"serverId01","host":"local
host","port":8002}]}
{"serverId":"7utq617o557582uljce1liduej","load":1,"ip":"10.10.4.212","port
":8001,"online":true,"updateTime":1527295327432,"action":"UPDATE_OR_INSERT
","command":"SERVER_ANNOUNCE"}
{"command":"USER_SYNC","user_list":[{"username":"user1","secret":"1j4kcf06
eg90mj25957atsses9","online":true,"updateTime":1527295294727}]}
{"command":"ACTIVITY_SYNC","activity_entity":[{"owner":"user1","activity_l
ist":[{"id":373940027,"activity":{"message_num":1,"authenticated_user":"us
er2"},"updateTime":1527295324721,"sendTime":1527295313665,"isDelivered":tr
ue},{"id":724739964,"activity":{"message_num":2,"authenticated_user":"user
2"},"updateTime":1527295320533,"sendTime":1527295320533,"isDelivered":fals
e}]}]}
```

Messages user1 received (message_num1 is before message_num 2)


User:(user1) | Server:(localhost:8001)
JSON output, received from server
Backup Servers
host
port
JSON input, to send to server
Messages received from server
Send
Disconnect

```
{
 "message_num": 1,
 "authenticated_user": "user2"
}
{
 "message_num": 2,
 "authenticated_user": "user2"
}
```

```
{"command":"LOGIN_SUCCESS","info":"login successfully as u
{
 "activity": {
  "message_num": 1,
  "authenticated_user": "user2"
 },
 "command": "ACTIVITY_BROADCAST"
}
{
 "activity": {
```

Testing Result

Result as excepted.

Load balancing

Operations

1. start 2 servers

```
java -jar ActivityStreamerServer.jar -lh localhost -lp 8001 -s abc
java -jar ActivityStreamerServer.jar -lh localhost -lp 8002 -s abc -rh
localhost -rp 8001
```

2. Register and login 2 clients both to server 8001

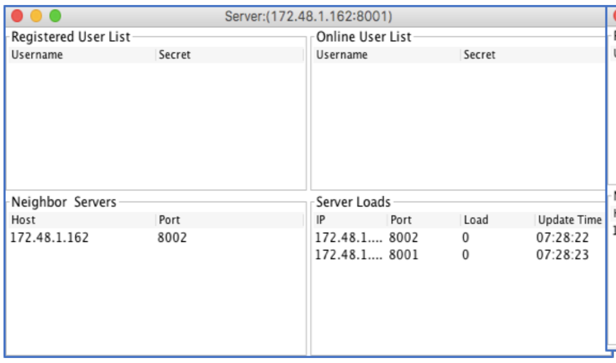
```
java -jar ActivityStreamerClient.jar -u user1 -rp 8001 -rh localhost
java -jar ActivityStreamerClient.jar -u user2 -rp 8001 -rh localhost
```

Expected Result

- user2 will be redirected to server 8002

Screentshot

Starting 2 servers


Server:(172.48.1.162:8001)

Registered User List

Username	Secret

Online User List

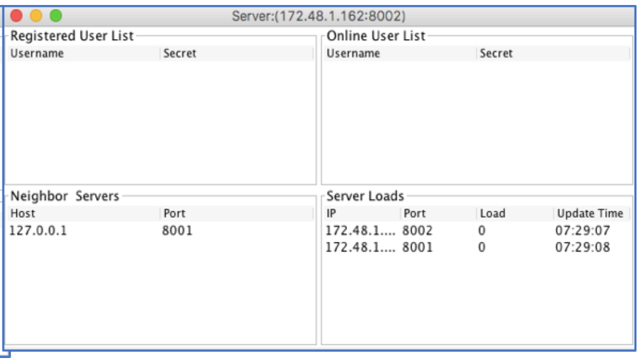
Username	Secret

Neighbor Servers

Host	Port
172.48.1.162	8002

Server Loads

IP	Port	Load	Update Time
172.48.1....	8002	0	07:28:22
172.48.1....	8001	0	07:28:23


Server:(172.48.1.162:8002)

Registered User List

Username	Secret

Online User List

Username	Secret

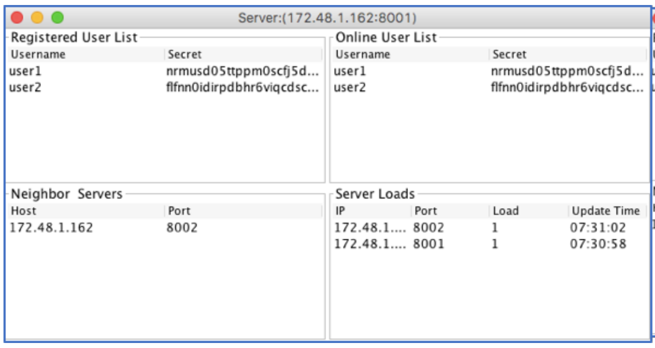
Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
172.48.1....	8002	0	07:29:07
172.48.1....	8001	0	07:29:08

After two clients login(load of each server has been changed to 1)


Server:(172.48.1.162:8001)

Registered User List

Username	Secret
user1	nrmusd05ttppm0scfj5d...
user2	flfnn0idirpdbhr6viqcdsc...

Online User List

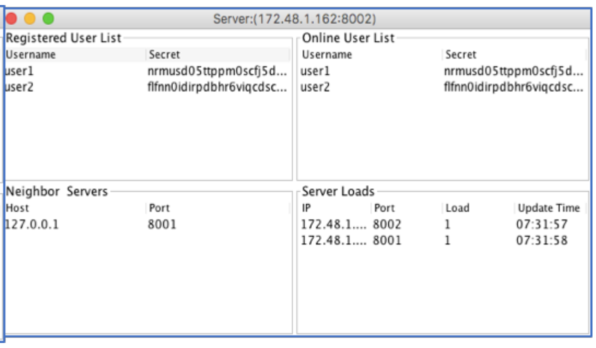
Username	Secret
user1	nrmusd05ttppm0scfj5d...
user2	flfnn0idirpdbhr6viqcdsc...

Neighbor Servers

Host	Port
172.48.1.162	8002

Server Loads

IP	Port	Load	Update Time
172.48.1....	8002	1	07:31:02
172.48.1....	8001	1	07:30:58


Server:(172.48.1.162:8002)

Registered User List

Username	Secret
user1	nrmusd05ttppm0scfj5d...
user2	flfnn0idirpdbhr6viqcdsc...

Online User List

Username	Secret
user1	nrmusd05ttppm0scfj5d...
user2	flfnn0idirpdbhr6viqcdsc...

Neighbor Servers

Host	Port
127.0.0.1	8001

Server Loads

IP	Port	Load	Update Time
172.48.1....	8002	1	07:31:57
172.48.1....	8001	1	07:31:58

Testing Result

Result as excepted.