Project 1 - Multi-server Network

Introduction

This is a multi-server communication network which allows

- Any number of servers join the system (if it has the correct secret of this system).
- Users register to this system with a unique username.
- Users login from **any server** within this network if he/she registered in this system (any server is ok) or he/she uses an **anonymous** user.
- Users send activities to the system and all other online users (include anonymous users) will receive this activities.

How to compile this code

All java source files are included in folder `src`, and use below command to package this project

(Note: You need `maven` to build/package this project)

```
cd src
./build.sh
```

This script will compile this project and create two jar files under `target` folder

```
└─$ cd target
└─$ ll ActivityStreamer*
-rw-r--r--  1 xxxx  staff   2.1M 27 Apr xx:xx ActivityStreamerClient.jar
-rw-r--r--  1 xxxx  staff   2.1M 27 Apr xx:xx ActivityStreamerServer.jar
```

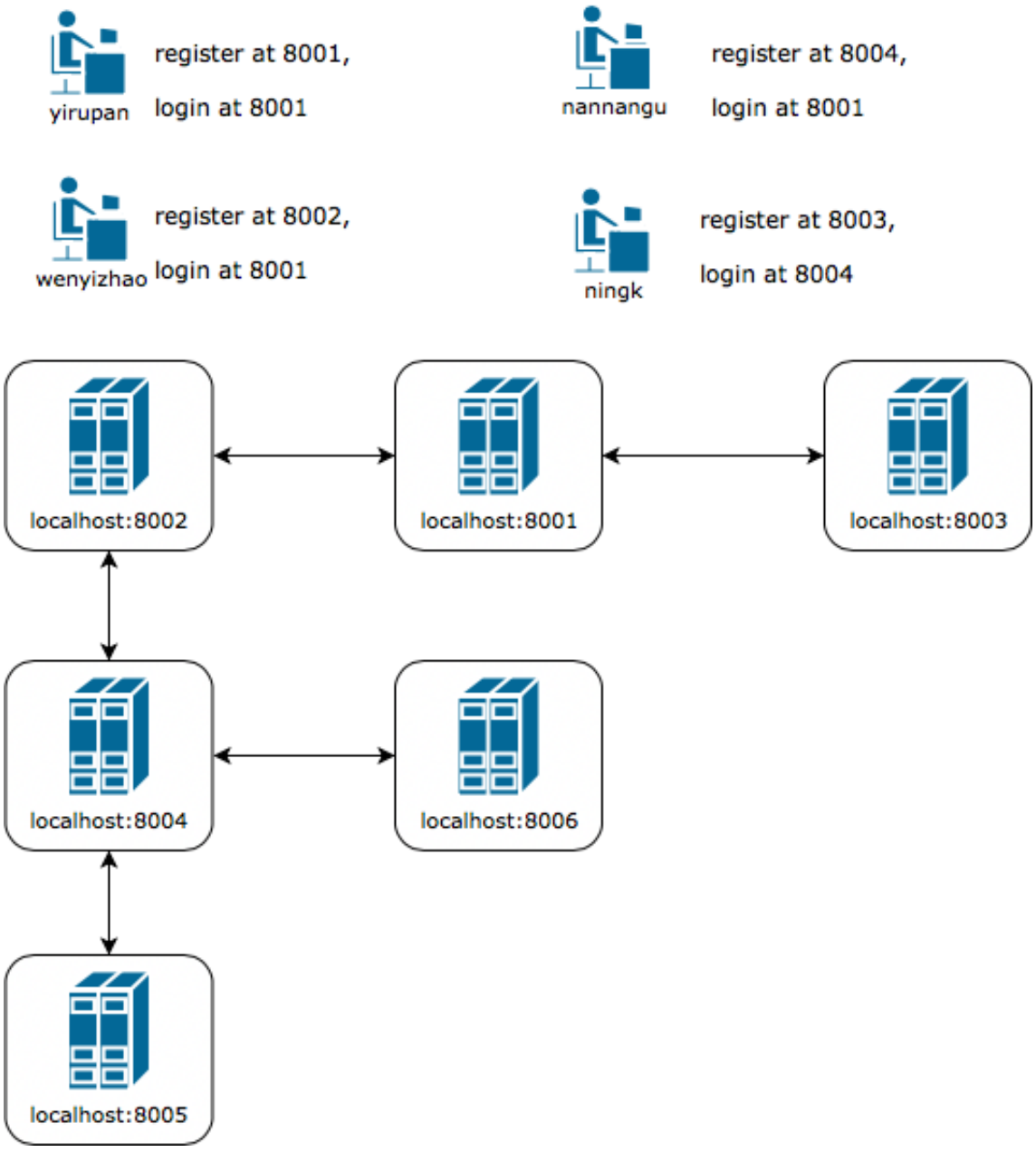
How to start the network using jar files

Shortcut

Use following script, a system as the picture below:

- 6 servers will be lanuched
- 4 users will register automatically
- 5 users will login in to this system, with a redirect case

```
cd $project_folder
sh auto_lanuch_deliver.sh
```



Server Setup

```
usage: java -jar ActivityStreamerServer [-a <arg>] [-lh <arg>] [-lp <arg>]
[-rh
       <arg>] [-rp <arg>] [-s <arg>]
An ActivityStream Server for Unimelb COMP90015

 -a  <arg>   activity interval in milliseconds;Optional, default value =
"5000"
 -lh <arg>   local hostname; Optional, default value="localhost"
 -lp <arg>   local port number; Optional, default value="3780"
 -rh <arg>   remote hostname; Optional for the very first server and
Mandantory for new coming servers.
 -rp <arg>   remote port number;OPtional, default value="3780"
 -s  <arg>    secret for the server to use; Optional, program will generate
one if is not provided.
```

Assume the secret is provided as `abc` and `8001` as the very first server port.

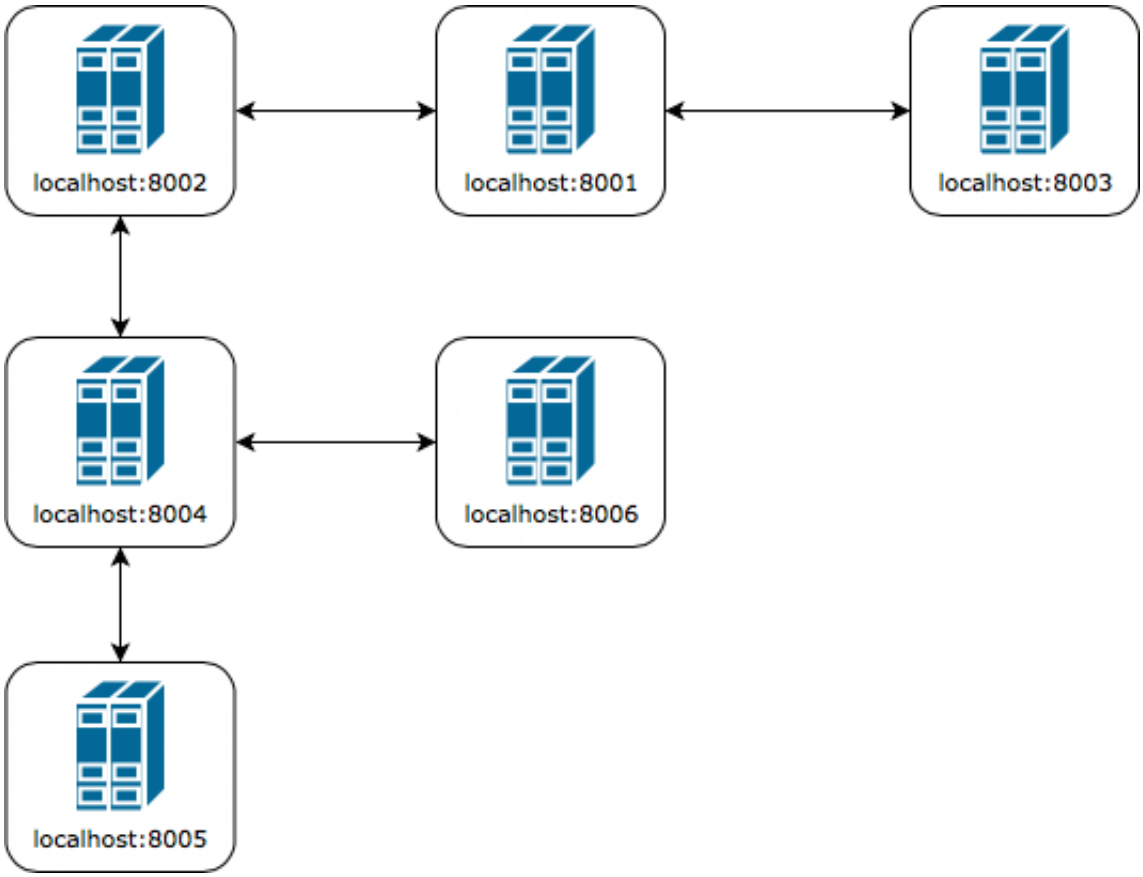
- Start the very first server

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

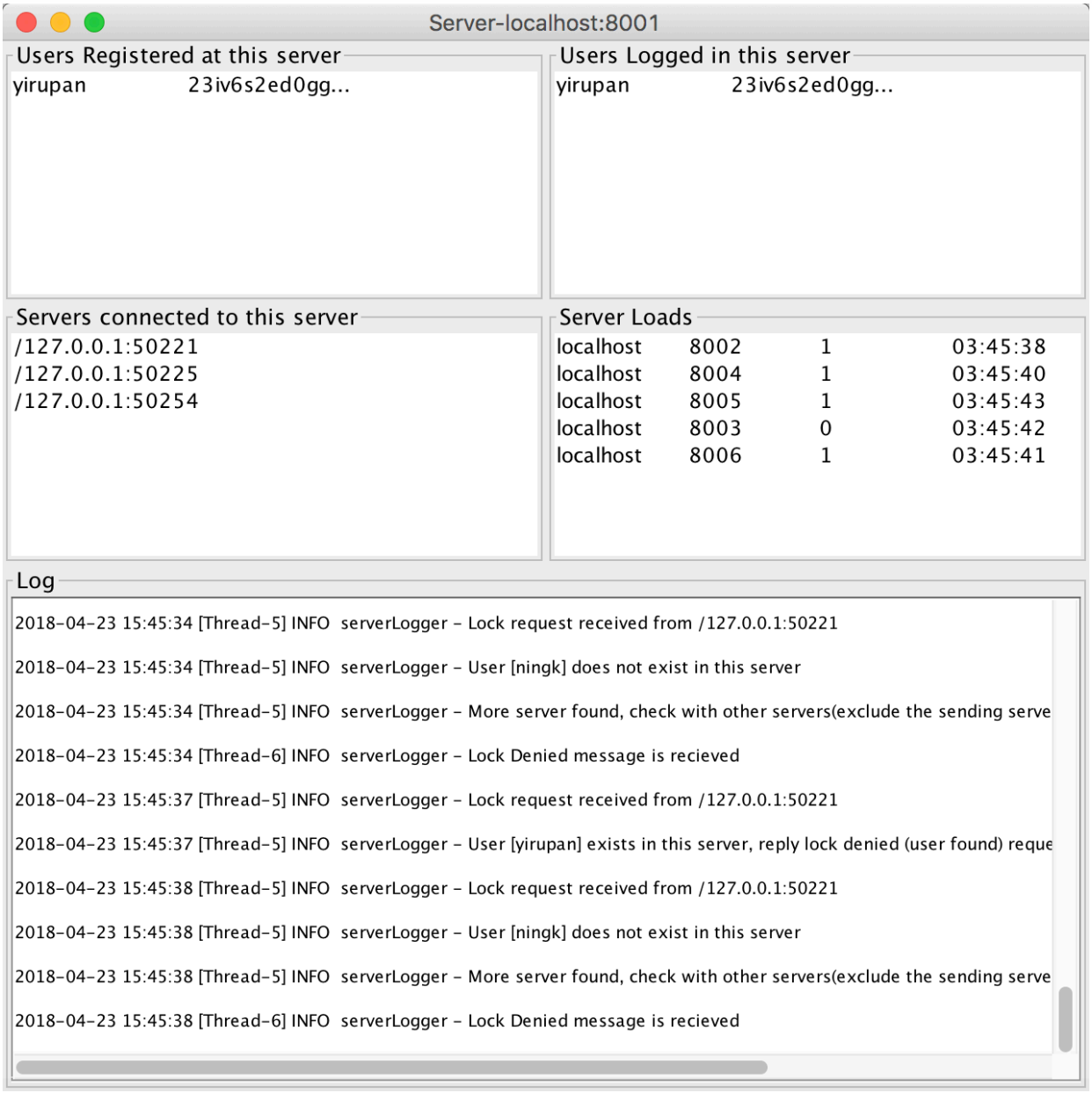
- New servers joining the system

```
# Connect to 8001 server with system secret
java -jar ActivityStreamerServer.jar -lh localhost -lp 8002 -s abc -rh
localhost -rp 8001

# Connect to 8001 server with system secret
java -jar ActivityStreamerServer.jar -lh localhost -lp 8003 -s abc -rh
localhost -rp 8001

# Connect to 8002 server with system secret
java -jar ActivityStreamerServer.jar -lh localhost -lp 8004 -s abc -rh
localhost -rp 8002

# Connect to 8004 server with system secret
java -jar ActivityStreamerServer.jar -lh localhost -lp 8005 -s abc -rh
localhost -rp 8004

# Connect to 8004 server with system secret
java -jar ActivityStreamerServer.jar -lh localhost -lp 8006 -s abc -rh
localhost -rp 8004
```

In this way, a network will be established.



For every server, a UI will show up to indicate the information of login users, registered users, existing connections and log.



Client Setup

```
usage: java -jar ActivityStreamerClient.jar [-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".
```

Assume servers are started as the structure described above.

- User register

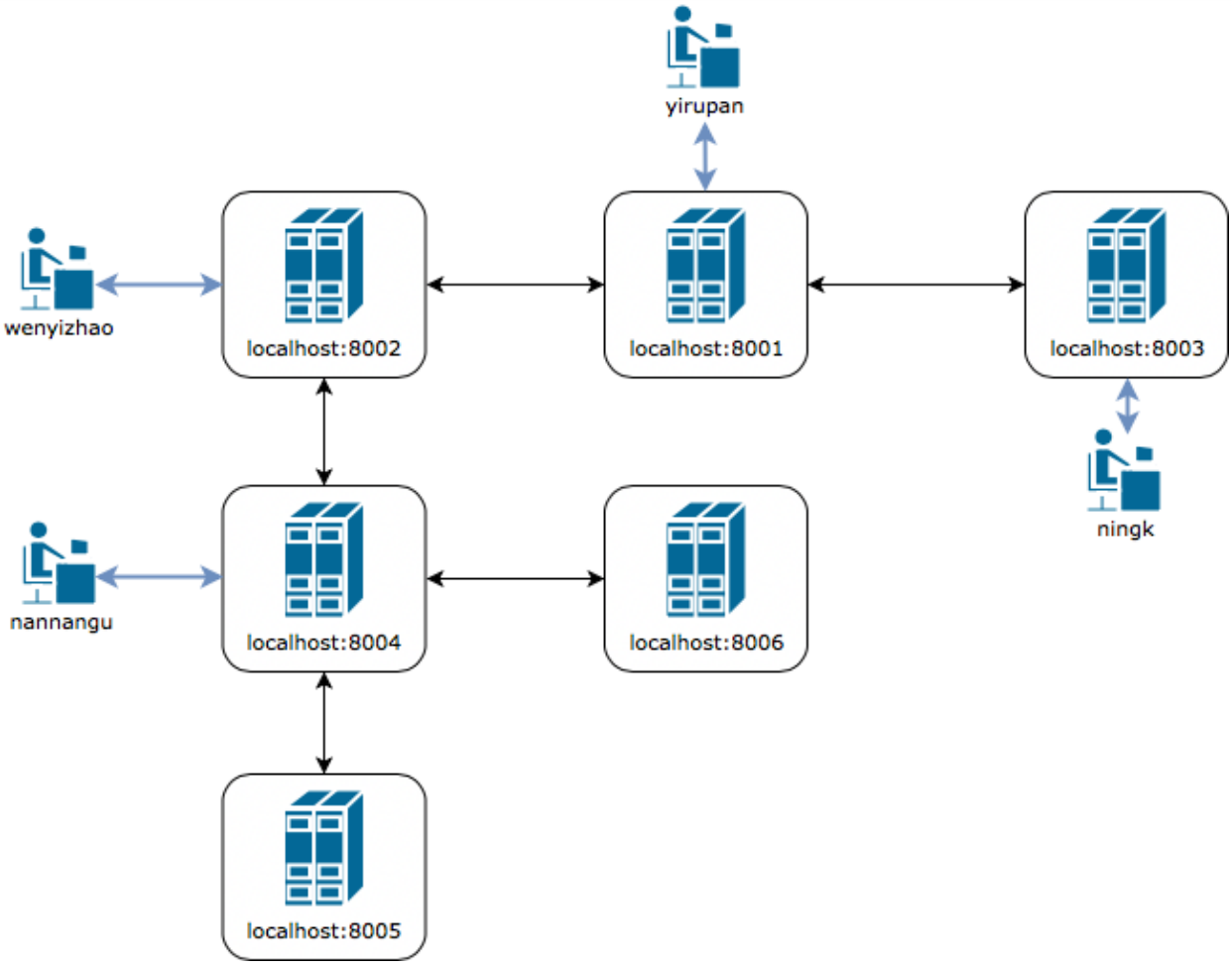
```
# Register user named 'ningk' at server 8001
java -jar ActivityStreamerClient.jar -u ningk -rp 8001 -rh localhost

# Register user named 'yirupan' at server 8002
java -jar ActivityStreamerClient.jar -u yirupan -rp 8002 -rh localhost

# Register user named 'nannangu' at server 8002
java -jar ActivityStreamerClient.jar -u nannangu -rp 8002 -rh localhost

# Register user named 'wenyizhao' at server 8005
java -jar ActivityStreamerClient.jar -u wenyizhao -rp 8005 -rh localhost
```

- User login

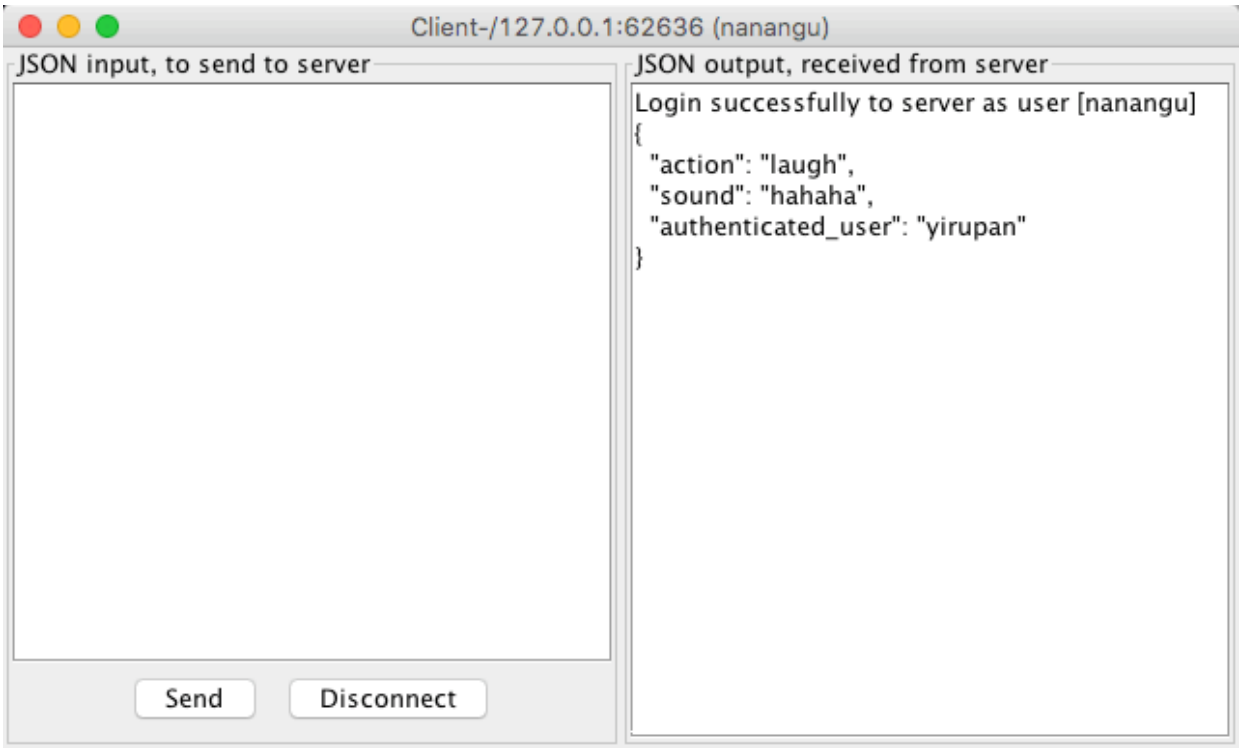
Note that users who are already registered can login from any server.

```
# Login user named 'ningk' at server 8003 (instead of 8001 which this id
registers at)
java -jar ActivityStreamerClient.jar -u ningk -rp 8003 -rh localhost -s
$secret1

# Login user named 'yirupan' at server 8001 (instead of 8002 which this id
registers at)
java -jar ActivityStreamerClient.jar -u yirupan -rp 8001 -rh localhost -s
$secret1

# Login user named 'nannangu' at server 8004 (instead of 8002 which this id
registers at)
java -jar ActivityStreamerClient.jar -u nannangu -rp 8004 -rh localhost -s
$secret1

# Login user named 'wenyizhao' at server 8002 (instead of 8005 which this
id registers at)
java -jar ActivityStreamerClient.jar -u wenyizhao -rp 8002 -rh localhost -s
$secret1
```

This login will make the network like this:



A UI will show up which allows user to send activity and receive message from server.



Client Sends Activities

Users can send activities through UI, just as what it shows, but must be **in JSON format**.

Contributors

Ning Kang

Nannan Gu

Yiru Pan

Wenyi Zhao

Copyright