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 start the network

Server Setup

```
usage: ActivityStreamer.Server [-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 Server-jar-with-dependencies.jar -lh localhost -lp 8001 -s abc
```

- New servers joining the system

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

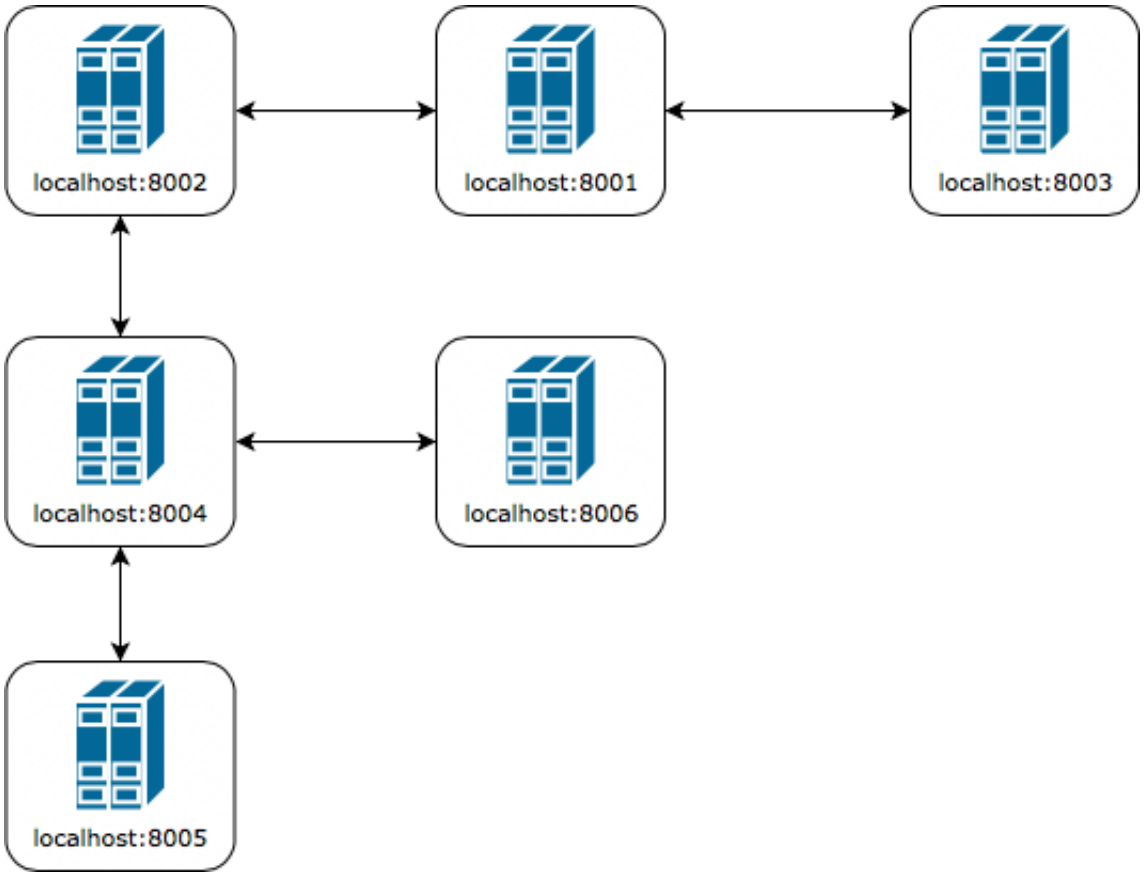
```
# Connect to 8001 server with system secret
java -jar Server-jar-with-dependencies.jar -lh localhost -lp 8003 -s abc -
rh localhost -rp 8001

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

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

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

In this way, a network will be established.



Client Setup

```
usage: ActivityStreamer.Client [-a] [-l] [-r] [-rh <arg>] [-rp <arg>] [-s
       <arg>] [-u <arg>]
An ActivityStream Client for Unimelb COMP90015

 -a          anonymous login
 -l or -r    user login or user register
 -rh <arg>   remote hostname
 -rp <arg>   remote port number
 -s  <arg>   secret for username
 -u  <arg>   username
```

Assume servers are started as the structure described above.

- User register

```
# Register user named 'ningk' at server 8001
java -jar Client-jar-with-dependencies.jar -r -u ningk -rp 8001 -rh
localhost -s secret1

# Register user named 'yirupan' at server 8002
java -jar Client-jar-with-dependencies.jar -r -u yirupan -rp 8002 -rh
localhost -s secret1

# Register user named 'nannangu' at server 8002
java -jar Client-jar-with-dependencies.jar -r -u nannangu -rp 8002 -rh
localhost -s secret1

# Register user named 'wenyizhao' at server 8005
java -jar Client-jar-with-dependencies.jar -r -u wenyizhao -rp 8005 -rh
localhost -s secret1
```

- User login

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

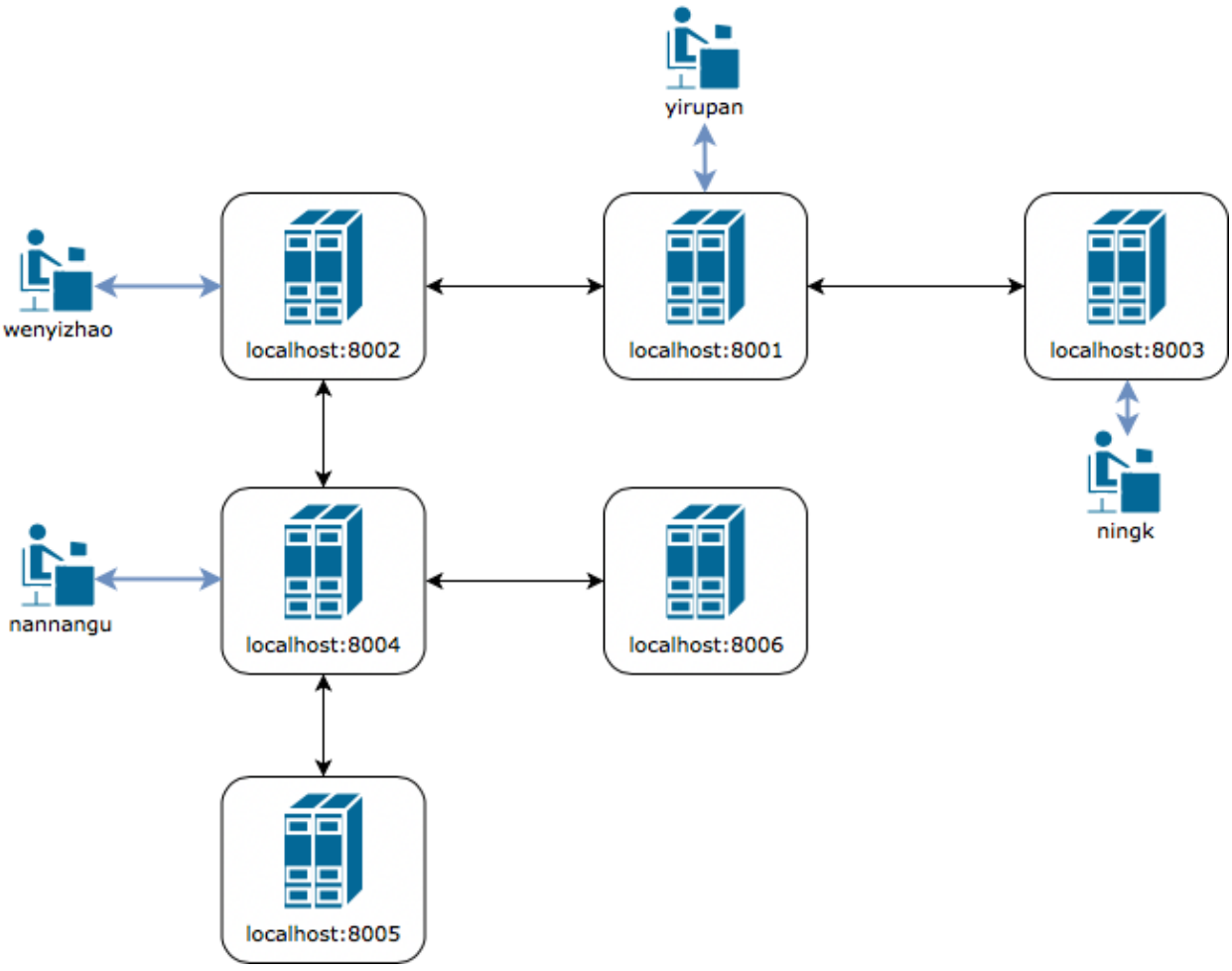
```
# Login user named 'ningk' at server 8003 (instead of 8001 which this id
registers at)
java -jar Client-jar-with-dependencies.jar -l -u ningk -rp 8003 -rh
localhost -s secret1

# Login user named 'yirupan' at server 8001 (instead of 8002 which this id
registers at)
java -jar Client-jar-with-dependencies.jar -l -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 Client-jar-with-dependencies.jar -l -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 Client-jar-with-dependencies.jar -l -u wenyizhao -rp 8002 -rh
localhost -s secret1
```

This login will make the network like this:



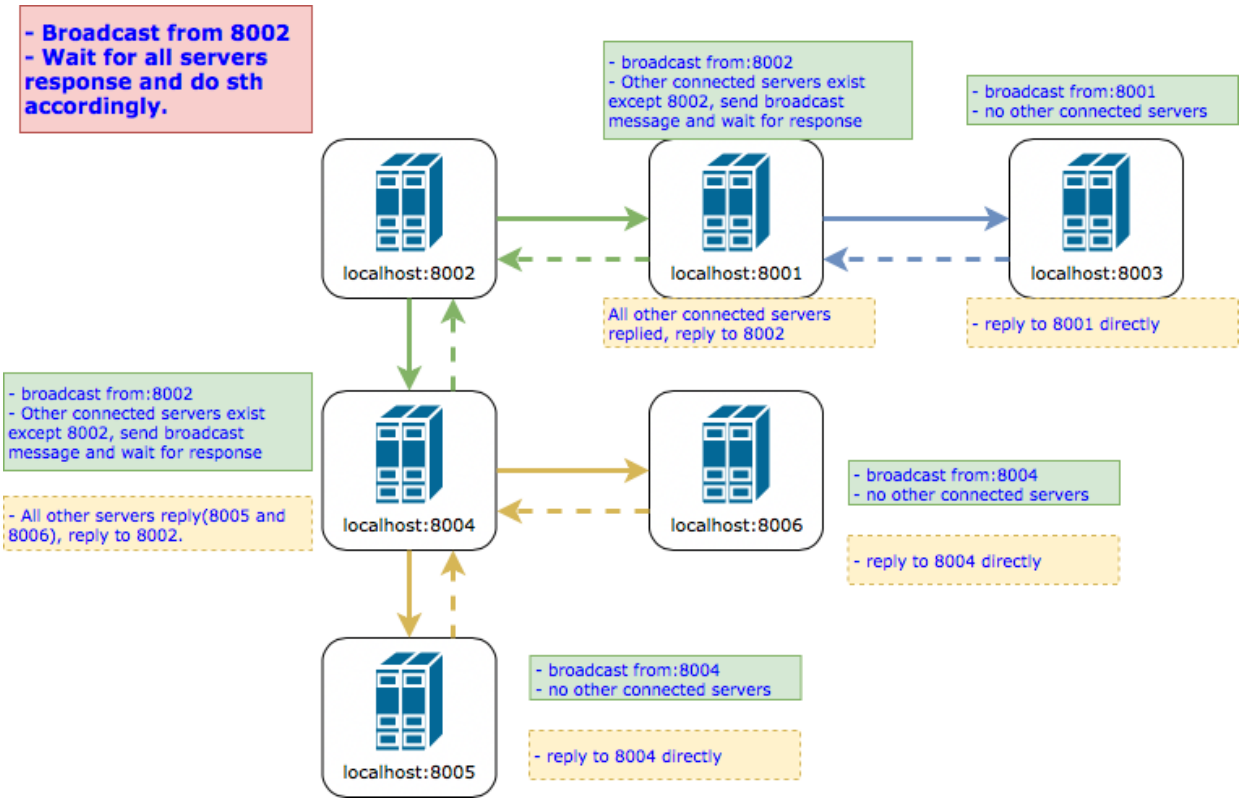
Client Sends Activities

Users can send activities through UI, just as what it shows.

How this system works

Server Broadcast

Most of the sync work is done by server boradcasting. As the network is a `tree-like` structure, every server should only broadcast message to other servers except the server who sends this message. For example (broadcast sending form `server 8002`)



Server Announce about client load

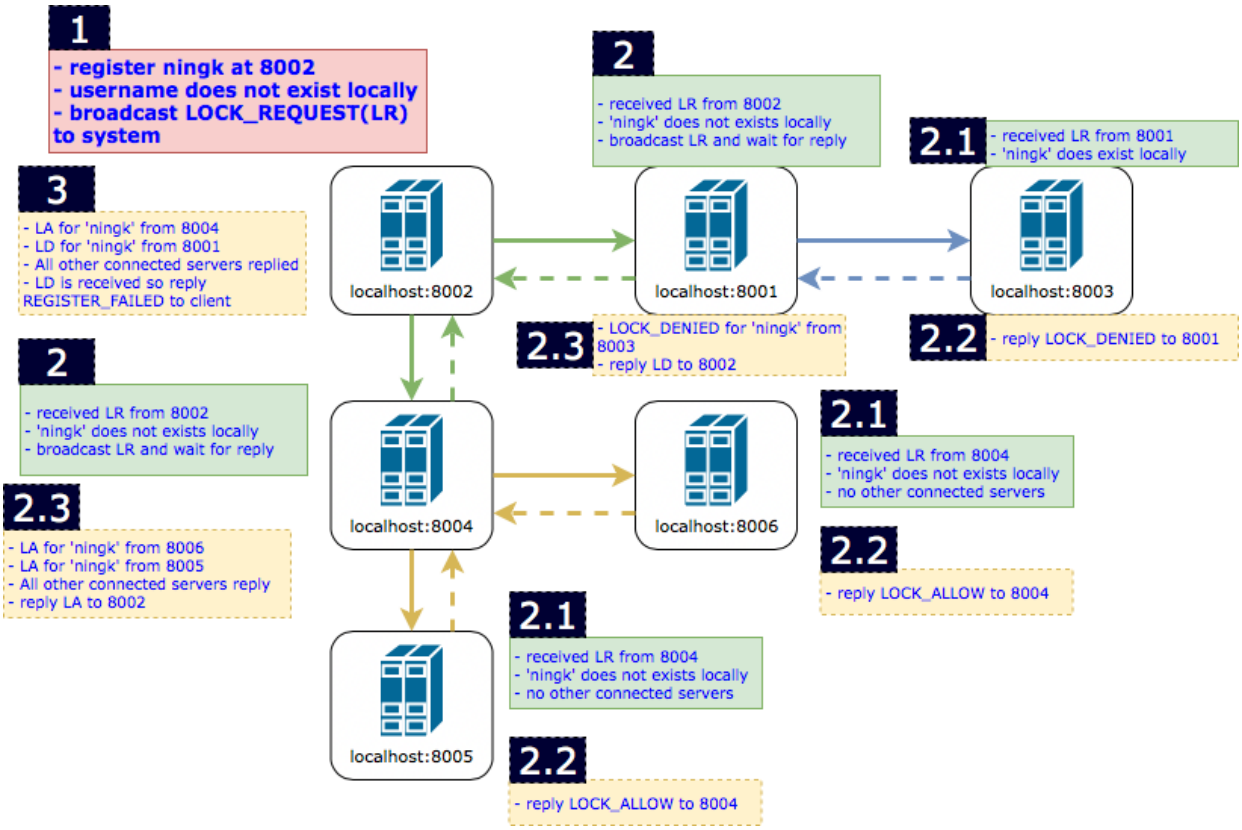
Every 5 seconds, every server will announce its client load via broadcast described above, which means every server will maintain a **table** which contains all client loads of all servers. **This is used for redirecting.**

Activity Broadcast

Activities sent by clients will be transformed by broadcast process described above.

Register Validation

In order to ensure the uniqueness of username, servers need to communicate with each other to ensure the username is not exists in any other servers. This is implemented by involving **LOCK** messages. Take an example from the previously built network, if someone registers username `ningk` from `server 8002`:



Note:

- A server will immediately reply LOCK_DENIED to the 'from' server once it receives one LOCK_DENIED from connected servers.
- A server only reply LOCK_ALLOW to the 'from' server after it receives LOCK_ALLOW from **ALL** connected servers.

Login Validation

The login process is almost the same with register process except it uses "ENQUIRY" messages. And the reply logic is just opposite.

Note:

- A server will immediately reply USER_FOUND to the 'from' server once it receives one USER_FOUND from connected servers.
- A server only reply USER_NOT_FOUND to the 'from' server after it receives USER_NOT_FOUND from **ALL** connected servers.

Login Redirection

Use the table that maintained by the announce process, new coming client connection may be redirected to another server which hold 2 or more clients than the server itself.

Contributors

Ning Kang

Nannan Gu

Yiru Pan

Wenyi Zhao

Copyright