COMP90015 Distributed Systems
Project 2 - Report
Spicy Hot Pot

Team Mentor
Yasmeen George

Team Members
- Duer Wang 824325
minghaow1@student.unimelb.edu.au
- Ivan Ken Weng Chee 736901
ichee@student.unimelb.edu.au
- Yue Yang 920577
yuey16@student.unimelb.edu.au
- Ziren Xiao 675485
zirenx@student.unimelb.edu.au

Introduction

This project is focused on improving the consistency and scalability of project 1 system with server failure models. We designed two different approaches to build a more robust system. Some changes were also made to the server protocol to incorporate the additional functionality. However, we nominate the tree server as our primary submission.

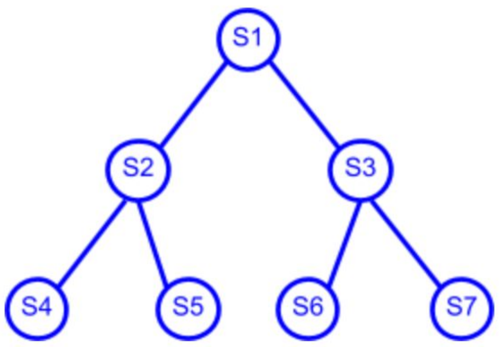


Figure: Tree Server Architecture

First one is built upon the existing system design from project 1 and simulates a hierarchical tree structure, with a root serve and subtrees of children with a parent server to link the communication between all connected clients.

In order to achieve the new aims of project 2, some main changes have been made based on our project 1:

To allow servers to join the system at any time, which means the new servers will be feeded information about all existing users. We have added such information to the protocol immediately after a new server is recognised as authenticated success.

To guarantee that an activity message sent by a client reaches all clients that are connected to the network at the time the message was sent, we have constructed a queue containing all the users currently connected to the system, and attached it with our broadcast message to servers, which allows other servers only to send the activity message to clients in the queue.

To maintain the load balance in our system as much as possible, for each server that receives new clients connection, instead of checking if there is any server connects to it that has 2 more / less load, we have adjusted the protocol to recognise the two servers that have largest load and smallest load, respectively, among its connected servers and itself. And then redirect clients from the server with largest load to smallest load if their load differ more than 2.

Regarding other requirements, we believe they have been addressed in our project 1.

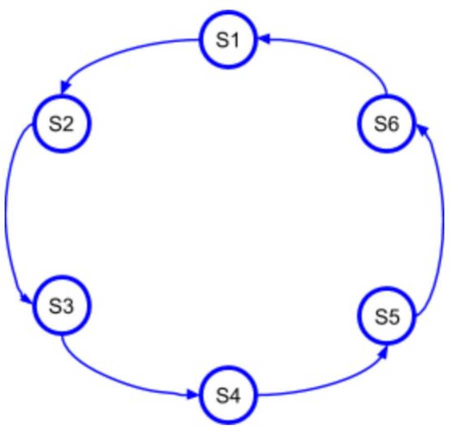


Figure: Ring Server Architecture

The other approach is a unidirectional ring architecture, whereby new servers join an existing server to extend the size of the ring. The ring system is essentially an improved variation of a worst case scenario of the tree structure, which would be a stub or linked-list, which is then wrapped around and joined at the ends ("Exploring Computer Network Topologies Like Bus, Ring and Star", 2018).

Description:

Protocol/ Terminology	Description
Anterior	Server directly in front of a server
Posterior	Server directly behind this server
Activity interval	How long a server waits before forwarding the orbit message. Interval of 0 would be the ideal performance
ORBIT	-Passed from a server to its anterior server -Only one exists in the system at a time -Contains -The id of the server who sent this -A list of servers, their hostname, port, and load -A list of messages to broadcast, in order -A list of new users with pending lock requests
AUTHENTICATION_SUCCESS	Responded to a server which authenticated successfully, thereby allowing it to fix the connection as its anterior server
DATA_SYNC	Sent to a new server in the system, which contains a list of existing users in JSON format, copied from anterior server
SERVER_REDIRECT	Sent to posterior once a new server joins, whereby the new server becomes the new posterior and old posterior is told to connect to the new server
LOCK_DENIED	The only message passed in reverse order, where once a server receives an ORBIT and checks users, the server sends this in the reverse direction for other servers to delete the new user

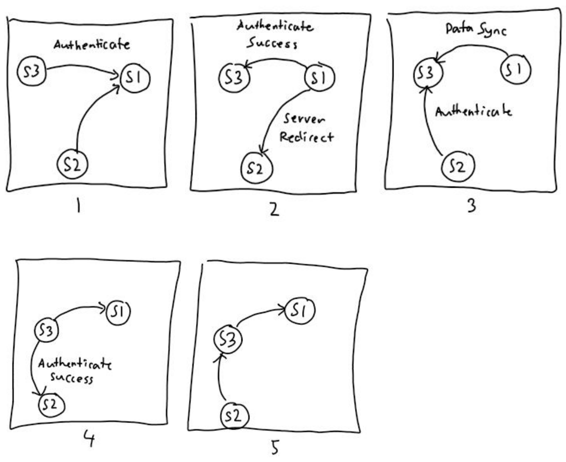


Figure: New server entering the ring

Server Failure Model

In this project, the revised failure model allows crashes to happen, including broken network connections. Based on this, there are certain modifications necessary for our system to handle these failures. There are many types of failures, such as Byzantine, Omission, Performance, Fail-stop, and Crash ("Failure modes in distributed systems", 2018). In our systems, we expect fail-stop failures where servers does not reply and stops responding. However, other servers are able to detect this.

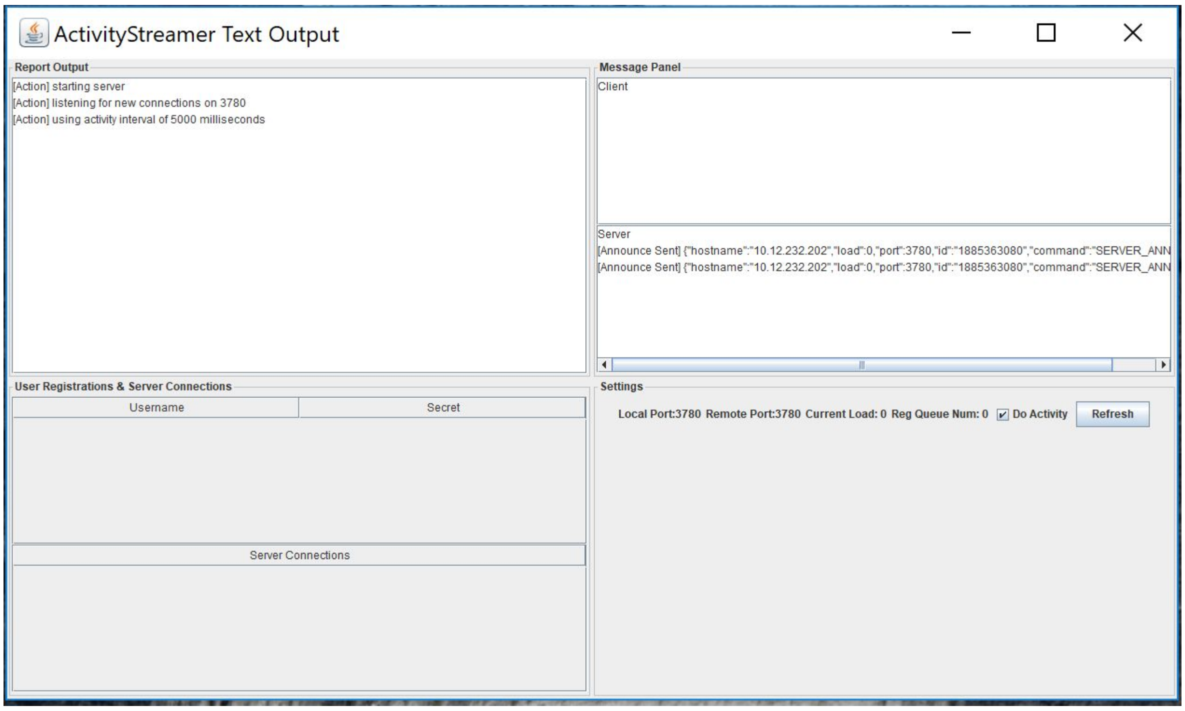
Tree structure approach:

Server Crash:
A new server can join the system at any living server node that has already existed in that system. If one of the server is crashed unexpectedly, both the parent node server and all children node servers will notice the abruptly terminates of communication. The child node server will redirect to the parent node of the crashed server and establish a new connection to reconstruct the system. However, if the parent is unreachable, the children servers will keep look up for an upper level server node until the new connection is built. One special case is that if the server root node crashed abruptly, the system will stop sending information or receiving information as a whole. Under that situation, all children nodes of the root server will send request for reconnection to the root server periodically. Once the root server is re-established, the child server will notice the server recovery and rebuild the connection with the server to restore the communication within the system.

Load balance for adding a new client:
Suppose we have 2 servers exist in the system waiting for client's connection. When the first two client login, both client will be connected to the root server. However, as the third client comes in, login redirect will be triggered and the root server will redirect the newest client to the other server since the difference of amount of connected client between two servers is larger than 2. Once a new client is login to the system, the root server will keep track the number of client existed on each connected servers node and reconstruct the system by redirect the client to achieve load balance.

Load balance for adding a new server:
Suppose we have one root server with three clients and the other child server with two clients in a system. If we add a new server in to the system by establish the connection with the child server of the root, the server node that the new server is connected to will first notice the unbalance of the number of clients among them. The parent server will redirect one of the clients to the new server. However, after this process, the root server, as the grandparent server of the new server, will now notice the unbalance between the servers (since the root server now has 3 clients, the child server has 1 and the new server has 1). The root server will redirect one of its clients to the child server in order to realize the load balance. After this redirection, all clients are evenly distributed among the system.

Step 1: Start a root server (3780):



ActivityStreamer Text Output

Report Output

```
[Action] starting server
[Action] listening for new connections on 3780
[Action] using activity interval of 5000 milliseconds
```

Message Panel

```
Client
```

```
Server
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANN
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANN
```

User Registrations & Server Connections

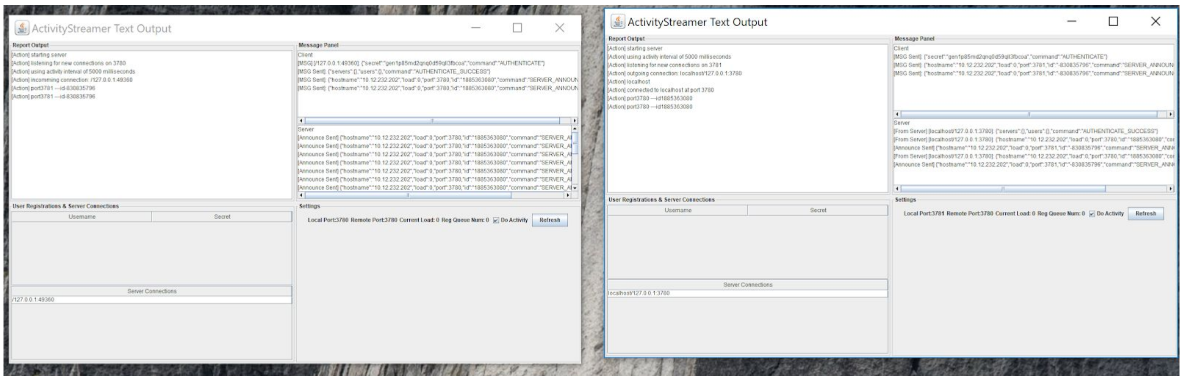
Username	Secret

Server Connections

Settings

Local Port:3780 Remote Port:3780 Current Load: 0 Reg Queue Num: 0 ☑ Do Activity

Step 2: add a child server (3781):



ActivityStreamer Text Output

Report Output

```
[Action] starting server
[Action] listening for new connections on 3780
[Action] using activity interval of 5000 milliseconds
[Action] incomming connection: /127.0.0.1:49360
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
```

Message Panel

```
Client
[MSG] [/127.0.0.1:49360]: {"secret":"gen1p85md2qnq0d59qll3fbcoa","command":"AUTHENTICATE"}
[MSG Sent]: {"servers":{},"users":{},"command":"AUTHENTICATE_SUCCESS"}
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNOUN
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNOUN
```

```
Server
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_AI
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_AI
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_AI
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_AI
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_AI
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_AI
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_AI
```

User Registrations & Server Connections

Username	Secret

Server Connections
/127.0.0.1:49360

Settings

Local Port:3780 Remote Port:3780 Current Load: 0 Reg Queue Num: 0 ☑ Do Activity

ActivityStreamer Text Output

Report Output

```
[Action] starting server
[Action] using activity interval of 5000 milliseconds
[Action] listening for new connections on 3781
[Action] outgoing connection: localhost/127.0.0.1:3780
[Action] localhost
[Action] connected to localhost at port 3780
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
```

Message Panel

```
Client
[MSG Sent]: {"secret":"gen1p85md2qnq0d59qll3fbcoa","command":"AUTHENTICATE"}
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOUN
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOUN
```

```
Server
[From Server] [localhost/127.0.0.1:3780]: {"servers":{},"users":{},"command":"AUTHENTICATE_SUCCESS"}
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","cor
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANN
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","cor
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANN
```

User Registrations & Server Connections

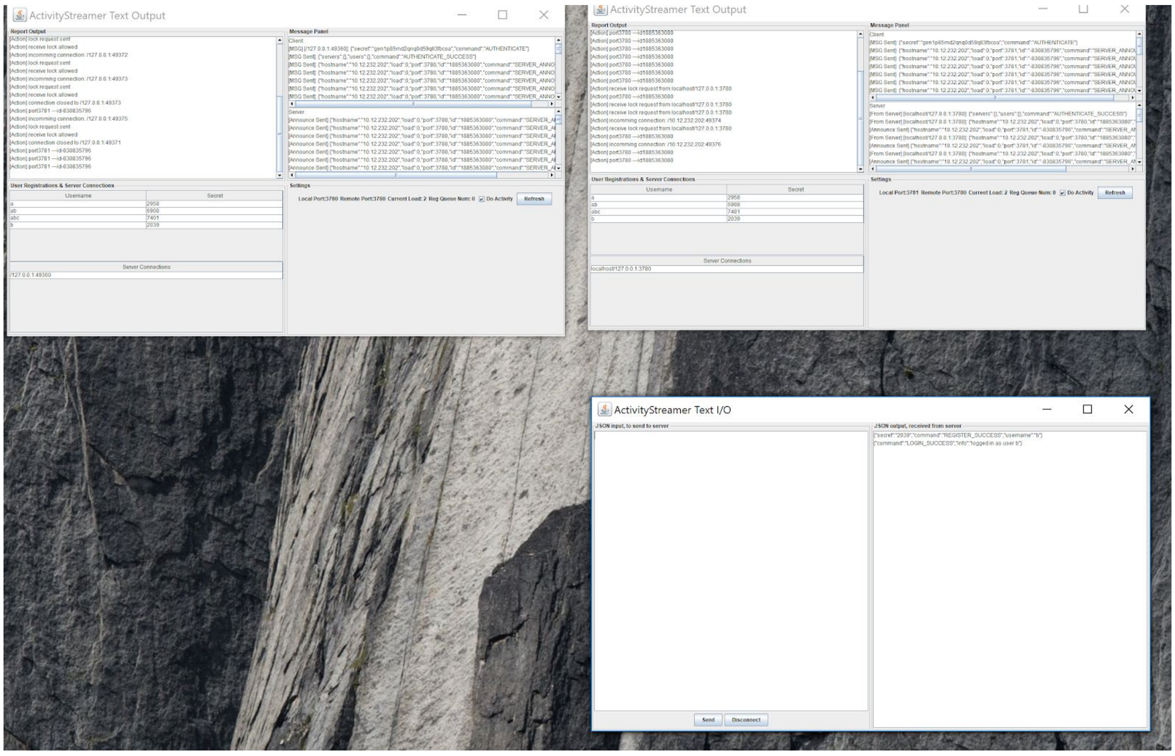
Username	Secret

Server Connections
localhost/127.0.0.1:3780

Settings

Local Port:3781 Remote Port:3780 Current Load: 0 Reg Queue Num: 0 ☑ Do Activity

Step 3: add four client to the root server, the client is evenly distributed in the system(each server has 2 clients) :



ActivityStreamer Text Output

```
Report Output
[Action] lock request sent
[Action] receive lock allowed
[Action] incomming connection: /127.0.0.1:49372
[Action] lock request sent
[Action] receive lock allowed
[Action] incomming connection: /127.0.0.1:49373
[Action] lock request sent
[Action] receive lock allowed
[Action] connection closed to /127.0.0.1:49373
[Action] port3781 —id-830835796
[Action] incomming connection: /127.0.0.1:49375
[Action] lock request sent
[Action] receive lock allowed
[Action] connection closed to /127.0.0.1:49371
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
```

```
Message Panel
Client
[MSG] [/127.0.0.1:49360]: {"secret":"gen1p85md2qnq0d59qll3fbcoa","command":"AUTHENTICATE"}
[MSG Sent]: {"servers":{},"users":{},"command":"AUTHENTICATE_SUCCESS"}
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
Server
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
```

User Registrations & Server Connections

Username	Secret
a	2958
ab	6908
abc	7401
b	2039

Server Connections
/127.0.0.1:49360

Settings: Local Port:3780 Remote Port:3780 Current Load: 2 Reg Queue Num: 0 [✓ Do Activity] Refresh

ActivityStreamer Text Output

```
Report Output
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] receive lock request from localhost/127.0.0.1:3780
[Action] port3780 —id1885363080
[Action] receive lock request from localhost/127.0.0.1:3780
[Action] receive lock request from localhost/127.0.0.1:3780
[Action] incomming connection: /10.12.232.202:49374
[Action] receive lock request from localhost/127.0.0.1:3780
[Action] port3780 —id1885363080
[Action] incomming connection: /10.12.232.202:49376
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
```

```
Message Panel
Client
[MSG Sent]: {"secret":"gen1p85md2qnq0d59qll3fbcoa","command":"AUTHENTICATE"}
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
Server
[From Server] [localhost/127.0.0.1:3780]: {"servers":{},"users":{},"command":"AUTHENTICATE_SUCCESS"}
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_AN
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_AN
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_AN
```

User Registrations & Server Connections

Username	Secret
a	2958
ab	6908
abc	7401
b	2039

Server Connections
localhost/127.0.0.1:3780

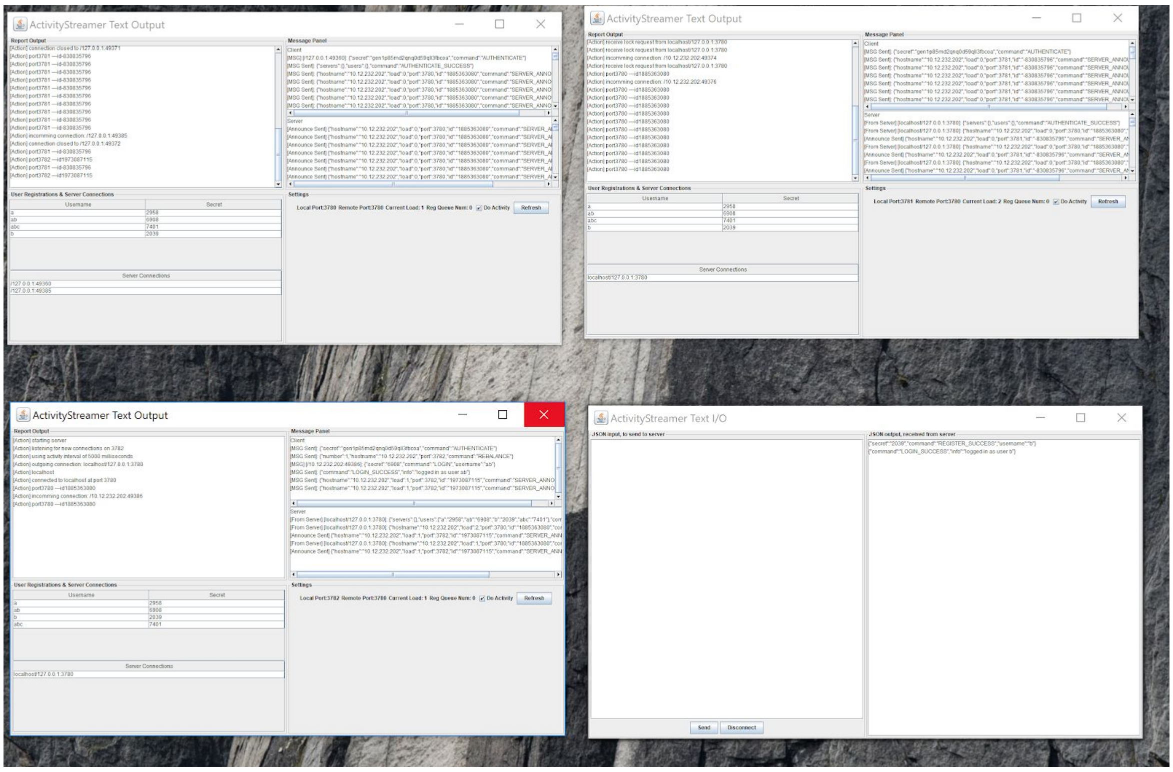
Settings: Local Port:3781 Remote Port:3780 Current Load: 2 Reg Queue Num: 0 [✓ Do Activity] Refresh

ActivityStreamer Text I/O

JSON input, to send to server

```
JSON output, received from server
{"secret":"2039","command":"REGISTER_SUCCESS","username":"b"}
{"command":"LOGIN_SUCCESS","info":"logged in as user b"}
```

Step 4: Add a new server (3782) to the system.The clients of child system will be redirect to the new server:



ActivityStreamer Text Output

```
Report Output
[Action] connection closed to /127.0.0.1:49371
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
[Action] port3781 —id-830835796
[Action] incomming connection: /127.0.0.1:49385
[Action] connection closed to /127.0.0.1:49372
[Action] port3781 —id-830835796
[Action] port3782 —id1973087115
[Action] port3781 —id-830835796
[Action] port3782 —id1973087115
```

```
Message Panel
Client
[MSG] [/127.0.0.1:49360]: {"secret":"gen1p85md2qnq0d59qll3fbcoa","command":"AUTHENTICATE"}
[MSG Sent]: {"servers":{},"users":{},"command":"AUTHENTICATE_SUCCESS"}
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
Server
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
```

User Registrations & Server Connections

Username	Secret
a	2958
ab	6908
abc	7401
b	2039

Server Connections
/127.0.0.1:49360
/127.0.0.1:49385

Settings: Local Port:3780 Remote Port:3780 Current Load: 1 Reg Queue Num: 0 [✓ Do Activity] Refresh

ActivityStreamer Text Output

```
Report Output
[Action] receive lock request from localhost/127.0.0.1:3780
[Action] receive lock request from localhost/127.0.0.1:3780
[Action] incomming connection: /10.12.232.202:49374
[Action] receive lock request from localhost/127.0.0.1:3780
[Action] port3780 —id1885363080
[Action] incomming connection: /10.12.232.202:49376
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
```

```
Message Panel
Client
[MSG Sent]: {"secret":"gen1p85md2qnq0d59qll3fbcoa","command":"AUTHENTICATE"}
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
Server
[From Server] [localhost/127.0.0.1:3780]: {"servers":{},"users":{},"command":"AUTHENTICATE_SUCCESS"}
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_AN
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_AN
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_AN
```

User Registrations & Server Connections

Username	Secret
a	2958
ab	6908
abc	7401
b	2039

Server Connections
localhost/127.0.0.1:3780

Settings: Local Port:3781 Remote Port:3780 Current Load: 2 Reg Queue Num: 0 [✓ Do Activity] Refresh

ActivityStreamer Text Output

```
Report Output
[Action] starting server
[Action] listening for new connections on 3782
[Action] using activity interval of 5000 milliseconds
[Action] outgoing connection: localhost/127.0.0.1:3780
[Action] localhost
[Action] connected to localhost at port 3780
[Action] port3780 —id1885363080
[Action] incomming connection: /10.12.232.202:49386
[Action] port3780 —id1885363080
```

```
Message Panel
Client
[MSG Sent]: {"secret":"gen1p85md2qnq0d59qll3fbcoa","command":"AUTHENTICATE"}
[MSG Sent]: {"number":1,"hostname":"10.12.232.202","port":3782,"command":"REBALANCE"}
[MSG] [/10.12.232.202:49386]: {"secret":"6908","command":"LOGIN","username":"ab"}
[MSG Sent]: {"command":"LOGIN_SUCCESS","info":"logged in as user ab"}
[MSG Sent]: {"hostname":"10.12.232.202","load":1,"port":3782,"id":"1973087115","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":1,"port":3782,"id":"1973087115","command":"SERVER_ANNO
Server
[From Server] [localhost/127.0.0.1:3780]: {"servers":{},"users":{"a":"2958","ab":"6908","b":"2039","abc":"7401"},"com
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":2,"port":3780,"id":"1885363080","co
[Announce Sent] {"hostname":"10.12.232.202","load":1,"port":3782,"id":"1973087115","command":"SERVER_ANN
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":1,"port":3780,"id":"1885363080","co
[Announce Sent] {"hostname":"10.12.232.202","load":1,"port":3782,"id":"1973087115","command":"SERVER_ANN
```

User Registrations & Server Connections

Username	Secret
a	2958
ab	6908
b	2039
abc	7401

Server Connections
localhost/127.0.0.1:3780

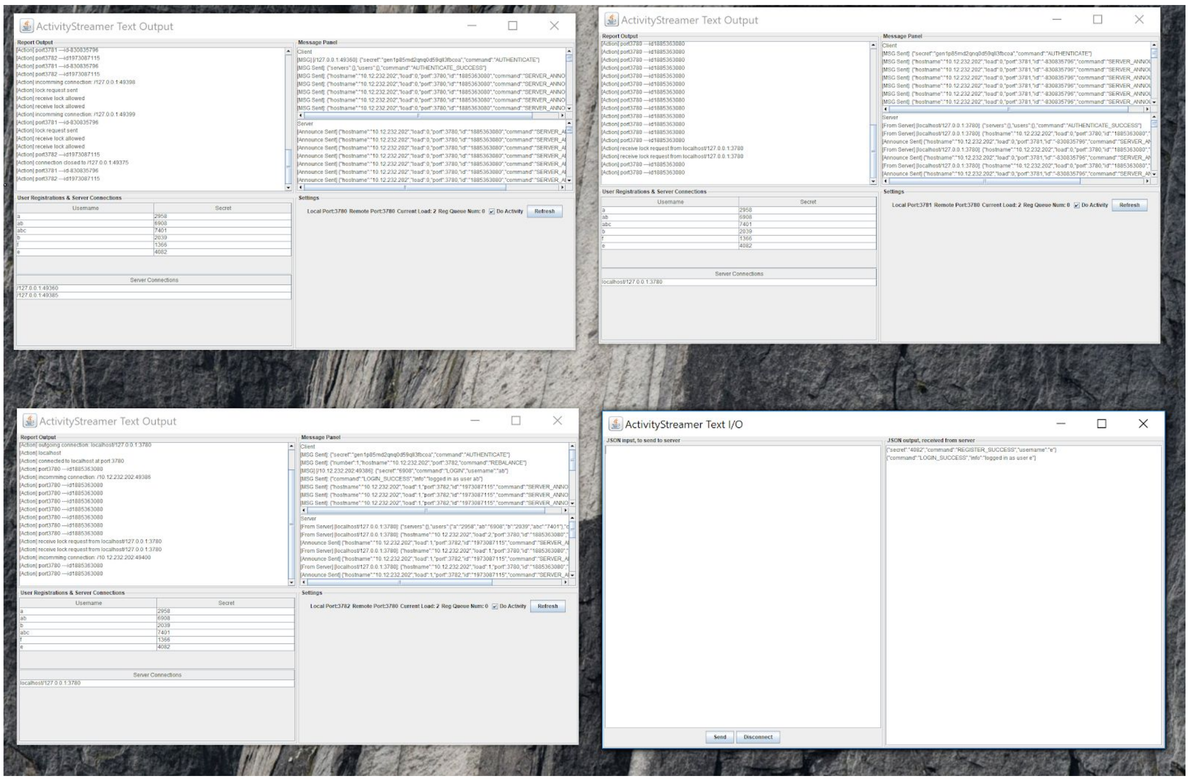
Settings: Local Port:3782 Remote Port:3780 Current Load: 1 Reg Queue Num: 0 [✓ Do Activity] Refresh

ActivityStreamer Text I/O

JSON input, to send to server

```
JSON output, received from server
{"secret":"2039","command":"REGISTER_SUCCESS","username":"b"}
{"command":"LOGIN_SUCCESS","info":"logged in as user b"}
```

Step5: Add two new clients.The clients are redirect and each server has 2 clients connections:



```
ActivityStreamer Text Output
Report Output
[Action] port3781 —id-830835796
[Action] port3782 —id1973087115
[Action] port3781 —id-830835796
[Action] port3782 —id1973087115
[Action] incomming connection: /127.0.0.1:49398
[Action] lock request sent
[Action] receive lock allowed
[Action] receive lock allowed
[Action] incomming connection: /127.0.0.1:49399
[Action] port3781 —id-830835796
[Action] lock request sent
[Action] receive lock allowed
[Action] receive lock allowed
[Action] port3782 —id1973087115
[Action] connection closed to /127.0.0.1:49375
[Action] port3781 —id-830835796
[Action] port3782 —id1973087115

Message Panel
Client
[MSG] [/127.0.0.1:49360]: {"secret":"gen1p85md2qnq0d59qll3fbcoa","command":"AUTHENTICATE"}
[MSG Sent]: {"servers":{},"users":{},"command":"AUTHENTICATE_SUCCESS"}
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_ANNO
Server
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","command":"SERVER_A

User Registrations & Server Connections
Username   Secret
a          2958
ab         6908
abc        7401
b          2039
f          1366
e          4082

Server Connections
/127.0.0.1:49360
/127.0.0.1:49385

Settings
Local Port:3780  Remote Port:3780  Current Load: 2  Reg Queue Num: 0  [✓] Do Activity  Refresh
```

```
ActivityStreamer Text Output
Report Output
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] receive lock request from localhost/127.0.0.1:3780
[Action] receive lock request from localhost/127.0.0.1:3780
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080

Message Panel
Client
[MSG Sent]: {"secret":"gen1p85md2qnq0d59qll3fbcoa","command":"AUTHENTICATE"}
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
[MSG Sent]: {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_ANNOU
Server
[From Server] [localhost/127.0.0.1:3780]: {"servers":{},"users":{},"command":"AUTHENTICATE_SUCCESS"}
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_AN
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_AN
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":0,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":0,"port":3781,"id":"-830835796","command":"SERVER_AN

User Registrations & Server Connections
Username   Secret
a          2958
ab         6908
abc        7401
b          2039
f          1366
e          4082

Server Connections
localhost/127.0.0.1:3780

Settings
Local Port:3781  Remote Port:3780  Current Load: 2  Reg Queue Num: 0  [✓] Do Activity  Refresh
```

```
ActivityStreamer Text Output
Report Output
[Action] outgoing connection: localhost/127.0.0.1:3780
[Action] localhost
[Action] connected to localhost at port 3780
[Action] port3780 —id1885363080
[Action] incomming connection: /10.12.232.202:49386
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080
[Action] receive lock request from localhost/127.0.0.1:3780
[Action] receive lock request from localhost/127.0.0.1:3780
[Action] incomming connection: /10.12.232.202:49400
[Action] port3780 —id1885363080
[Action] port3780 —id1885363080

Message Panel
Client
[MSG Sent]: {"secret":"gen1p85md2qnq0d59qll3fbcoa","command":"AUTHENTICATE"}
[MSG Sent]: {"number":1,"hostname":"10.12.232.202","port":3782,"command":"REBALANCE"}
[MSG] [/10.12.232.202:49386]: {"secret":"6908","command":"LOGIN","username":"ab"}
[MSG Sent]: {"command":"LOGIN_SUCCESS","info":"logged in as user ab"}
[MSG Sent]: {"hostname":"10.12.232.202","load":1,"port":3782,"id":"1973087115","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":1,"port":3782,"id":"1973087115","command":"SERVER_ANNO
[MSG Sent]: {"hostname":"10.12.232.202","load":1,"port":3782,"id":"1973087115","command":"SERVER_ANNO
Server
[From Server] [localhost/127.0.0.1:3780]: {"servers":{},"users":{"a":"2958","ab":"6908","b":"2039","abc":"7401"},"c
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":2,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":1,"port":3782,"id":"1973087115","command":"SERVER_A
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":1,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":1,"port":3782,"id":"1973087115","command":"SERVER_A
[From Server] [localhost/127.0.0.1:3780]: {"hostname":"10.12.232.202","load":1,"port":3780,"id":"1885363080","
[Announce Sent] {"hostname":"10.12.232.202","load":1,"port":3782,"id":"1973087115","command":"SERVER_A

User Registrations & Server Connections
Username   Secret
a          2958
ab         6908
b          2039
abc        7401
f          1366
e          4082

Server Connections
localhost/127.0.0.1:3780

Settings
Local Port:3782  Remote Port:3780  Current Load: 2  Reg Queue Num: 0  [✓] Do Activity  Refresh
```

```
ActivityStreamer Text I/O
JSON input, to send to server

JSON output, received from server
{"secret":"4082","command":"REGISTER_SUCCESS","username":"e"}
{"command":"LOGIN_SUCCESS","info":"logged in as user e"}

Send  Disconnect
```

Ring structure approach:

<u>Server Crash:</u>
When a server crashes, it leaves a gap in the ring, rendering the connection broken. Once its posterior server detects this, it will reference its anterior server (crashed) in the orbit server list and delete that entry, then attempt to connect to the next anterior, which would be the crashed server's anterior. Once a connection is successful, the posterior server will continue forwarding the orbit message to the new anterior, should it have received the latest orbit.

One advantage of this is that any server is allowed to crash, including the root server (first server). However, the system would not be able to recover should two or more servers crash at the same time, as their respective posteriors would not be able to synchronise which server crashed due to not communicating through the orbit, and hence would try to connect with the wrong server.

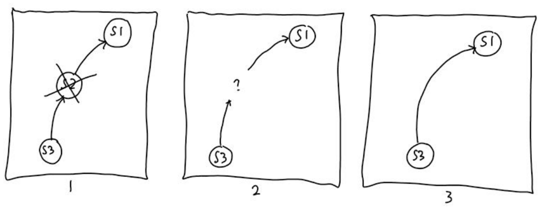


Figure: Ring server crash

<u>Load Balancing:</u>
If a new client connects to a server and registers successfully, and the server detects that another server has a client load count 2 less than its own, including the new client, it will redirect the client to that server. For new servers which have joined with 0 load, its anterior will push 1 client at a time to it, during each orbit. This will be followed by all servers which will slowly but eventually achieve load balance.

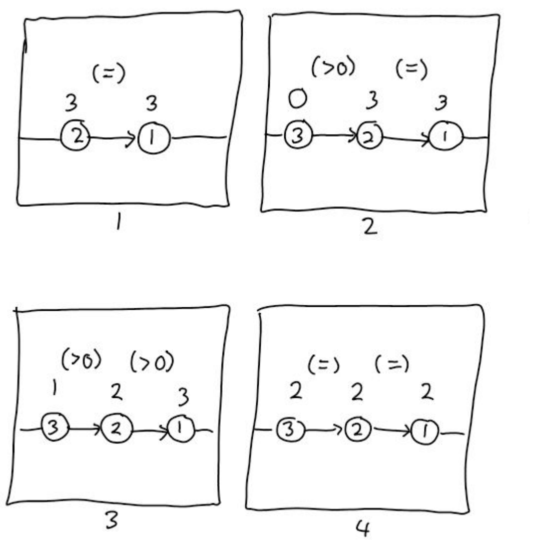


Figure: Ring server load balancing

Availability and Consistency

Based on the CAP theorem, we know that it is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees: consistency, availability and partition tolerance ("CAP theorem", 2018). Since no distributed system is absolutely safe from network failures, network partitioning should be allowed to exist in the system. This property implies that we need to decide the trade-off between the consistency and availability. Consistency guarantees that the client should receive the most recent message, as the availability is realized by hold the strong relationship between request and response (each request receives a response) without the simultaneous transfer. In this project, we should mainly achieve the availability request by allowing the login client to receive the message after he sends the request as soon as possible (may not the latest one). Alternatively speaking, in the context of our system is the amount of uptime the system provides for clients to be able to send and receive messages. On top of that, we implemented the load balance and reconnect and queuing structure to reach the goal of system Consistency. As the client is evenly distributed throughout the system, when a network failure happened, the system is easy to update the most recent message to the clients. Also, with queuing system, the client could get the message in the same order.

Concurrency Issues

Interesting concurrency issues have cropped up as a result of the new requirements and failure model. This section describes the issues and how our servers attempt to overcome them, if possible.

Message ordering:
The tree server has an in-built mechanism for ensuring the ordering of messages. The synchronized keyword in functions achieves this, as it handles multiple threads in a sequential order.

On the other hand, the ring system uses a queue to store all messages sent by connected clients before pushing them together with the orbit message as it passes through. The queue is appended to the master queue in the orbit.

Network partitioning:
In the tree server, when a server with children crashes, a partition occurs between the subtrees of its children and parent. While the system is able to fix the structure in the server crash scenario.

In the ring system however, the server which detects its anterior has crashed, will hold on to the existing orbit message and continue broadcasting only once reconnected. This means that during the reconnection, the system is in a paused state, sacrificing availability. Both our systems should be able to rebuild their structures as long as one server crashes at a time. If two or more go down simultaneously, they might not recover.

Scalability

<u>Server Location:</u>
Servers located physically far apart will cause a longer delay in connection, more evident in the ring system as the ring diameter will be as large as the distance between the two furthest servers. The tree structure allows faster transmission between subtrees sharing similar locality, but lock broadcasts would take just as much time.

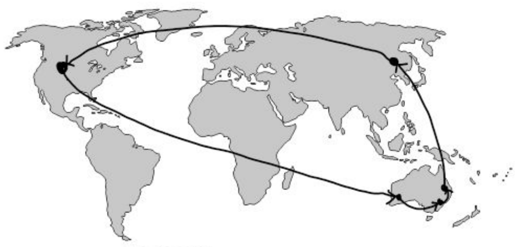


Figure: Ring server locality

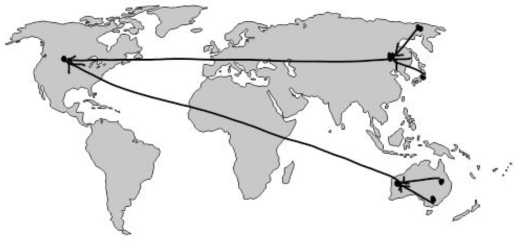


Figure: Tree server locality

<u>Server Load:</u>
When a new server joins the system after existing clients have joined, the system load would be imbalanced. Clients would be redirected to the new server to rebalance the system, but doing so may potentially be expensive, as well as sacrifice availability if messages do not reach clients while being redirected.

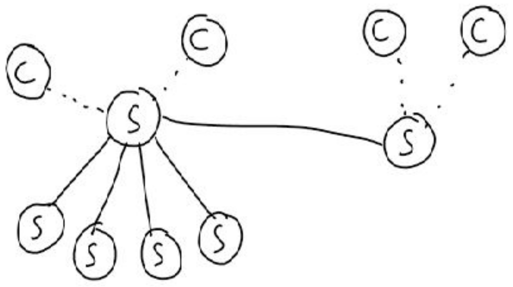


Figure: Servers with equal load in tree system

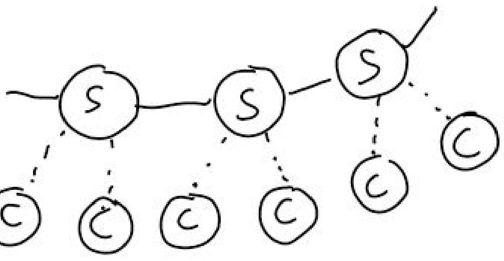


Figure: Servers with equal load in ring system

<u>Message complexity:</u>

Type	Action	Complexity
Tree	Registration	O(2n + 1)
	Broadcasting activity messages	O(n + m)
	Broadcasting server announce	O(n)
	Data sync	O(u)
Ring	Registration	O(n)
	Broadcasting activity messages	O(n + m)
	Broadcasting server announce	O(n)
	Data sync	O(u)

Transmision time:
Ring system becomes linearly slower as more servers are added, as the orbit message has to pass through additional servers. The tree structure only slows down with additional increase of depth.

Conclusion

In summary, we have chosen to design two systems in order to expose ourselves to understanding how different distributed systems function, which in turn gave us better insights as to how different architectures can offer interesting advantages and disadvantages. Both systems share a similarity in that they are completed from first principles, using basic socket communication. Their implementation differences are expressed below:

	Tree	Ring
Project management	Ant (Eclipse)	Maven
JSON Parsing	SimpleJSON	Google GSON
Multithreaded classes	Extends Thread	Implements Runnable
Debug Output	JFrame GUI	Apache Log4j Console
Source code	Extended from Project 1	Built from scratch

References

1
Failure modes in distributed systems. (2018). Retrieved from http://alvaro-videla.com/2013/12/failure-modes-in-distributed-systems.html

2
Exploring Computer Network Topologies Like Bus, Ring and Star. (2018). Retrieved from https://www.lifewire.com/computer-network-topology-817884

3
CAP theorem. (2018). Retrieved from https://en.wikipedia.org/wiki/CAP_theorem

Minutes

COMP90015 Distributed Systems

Spicy Hot Pot

30/04/2018 Project 2 Meeting 1

Meeting Minutes

Attendees

- Duer Wang
- Ivan Ken Weng Chee
- Yue Yang
- Ziren Xiao

Time

- 19:30 - 21:00

Venue

- EDS6,
  Old Engineering,
  The University of Melbourne,
  Parkville

Goals

- Familiarise ourselves with Project 2 specification
- General discussion of Project 1

Discussion

Time	Item	Who	Notes
30 min	Specification	Everyone	What we need to do
30 min	GitHub	Everyone	Created a GitHub repo
30 min	Architecture choice	Everyone	What architecture
10 min	Project Discussion	Everyone	How to start

Actions

- Read the specification
- Clone GitHub repository to our laptops
- Decided on centralised architecture

COMP90015 Distributed Systems

Spicy Hot Pot

07/05/2018 Project 2 Meeting 2

Meeting Minutes

Attendees

- Duer Wang
- Ivan Ken Weng Chee
- Yue Yang
- Ziren Xiao

Time

- 20:00 - 22:00

Venue

- 124,
  Old Engineering,
  The University of Melbourne,
  Parkville

Goals

- Decide on architecture
- Weigh pros and cons of decisions

Discussion

Time	Item	Who	Notes
1 hour	Discussion	Everyone	How to implement
30 min	Debate	Everyone	Pros and cons
30 min	Architecture choice	Everyone	Switch architecture

Actions

- Determined that centralised architecture had too many flaws and hard to fix
- Decided to go with extending tree architecture from Project 1
- Started thinking of alternative architectures like priority queue of servers

COMP90015 Distributed Systems

Spicy Hot Pot

14/05/2018 Project 2 Meeting 3

Meeting Minutes

Attendees

- Duer Wang
- Ivan Ken Weng Chee
- Yue Yang
- Ziren Xiao

Time

- 18:00 - 19:30

Venue

- Level 5,
  Arts West,
  The University of Melbourne,
  Parkville

Goals

- Rethink on our decisions
- Start coding
- Distribute tasks

Discussion

Time	Item	Who	Notes
1 hour	Coding	Everyone	Extending Project 1
30 min	Architecture choice	Everyone	New architecture?

Actions

- Started implementing extension on Project 1 code
- Decided to allocate 2 of us to work on a new Ring architecture as a backup
- Decided on doing and submitting two architectures, with 1 being the main

COMP90015 Distributed Systems

Spicy Hot Pot

21/05/2018 Project 2 Meeting 4

Meeting Minutes

Attendees

- Duer Wang
- Ivan Ken Weng Chee
- Yue Yang
- Ziren Xiao

Time

- 20:00 - 21:30

Venue

- Project Room 3,
  Eastern Resource Centre,
  The University of Melbourne,
  Parkville

Goals

- Code progress
- Start report writing
- Updates and discussion

Discussion

Time	Item	Who	Notes
30 min	Progress Discussion	Everyone	What we did so far
30 min	Start report skeleton	Everyone	Google docs link
20 min	Discuss ring system	Everyone	How to do
10 min	Start ring code	Everyone	Start coding

Actions

- Implemented server data synchronisation and initial server crashing handling on tree system
- Started implementing orbit message of ring architecture
- Started writing something in report

COMP90015 Distributed Systems

Spicy Hot Pot

25/05/2018 Project 2 Meeting 5

Meeting Minutes

Attendees

- Duer Wang
- Ivan Ken Weng Chee
- Yue Yang
- Ziren Xiao

Time

- 16:00 - 24:00

Venue

- Master of Computer Science Workroom,
  Doug McDonnell Building,
  The University of Melbourne,
  Parkville

Goals

- Code progress
- Report update
- Who submits

Discussion

Time	Item	Who	Notes
30 min	Progress Discussion	Everyone	What we did so far
30 min	Start report skeleton	Everyone	Google docs link
20 min	Discuss ring system	Everyone	How to do
10 min	Start ring code	Everyone	Start coding
6 hour 30 min	Coding	Everyone	Coding

Actions

- Finalised tree restructuring and other requirements of tree server
- Completed data sync, activity broadcast and lock requests for ring server
- Ziren will make submission

COMP90015 Distributed Systems

Spicy Hot Pot

26/05/2018 Project 2 Meeting 6

Meeting Minutes

Attendees

- Duer Wang
- Ivan Ken Weng Chee
- Yue Yang
- Ziren Xiao

Time

- 14:00 - 24:00

Venue

- Baillieu Library,
  The University of Melbourne,
  Parkville

Goals

- Code finalisation
- Report finalisation
- Discussion

Discussion

Time	Item	Who	Notes
30 min	Progress Discussion	Everyone	What we did so far
30 min	Finalise report	Everyone	Formatting
8 hour 30 min	Coding	Everyone	Finish coding
30 min	Finalise submission	Everyone	Package submission

Actions

- Finalised tree system
- Do whatever we can for ring system
- Finalised report
- Make submission