COMP 90015
Distributed System
Project 2 Report

Group Name:
1024
Group Members:
Mincheng Li (minchengl)
Email: minchengl@student.unimelb.edu.au
Wenzhou Wei (wenzhouw)
Email: wenzhouw@student.unimelb.edu.au
Xiaoming Zhang (xiaomingz)
Email: xiaomingz@student.unimelb.edu.au
Chaoran Zheng (chaoranz2)
Email: chaoranz2@student.unimelb.edu.au

1. Introduction

   This project aims to implement a server protocol and design a server based on the foundation of the project one. Referring to the requirements of this project, the system designed by the development team would allow sub-servers or clients to join or leave the system at any time. However, there also are several methods to prevent the failures caused by informal system problems or other problems. In order to ensure the system running smoothly, the team designed a standby plan for the master server.

2. The structure of the system

   In this system, there are one master server and one standby master server. Except these two servers, there would be several sub-servers named client servers connected to the master server. When different clients are connected to their servers, the master server would dynamically store registration information of clients and manage the messages which are unsuccessfully sent or received but other kinds of information or messages would be managed by the sub-servers.

3. Failure model

   3.1. The failure of master server

   As mentioned above, the system has a master server and a standby master server. When the master server works normally, it would manage the registration information and login information of all the clients, at the same time, it would also update these information to the standby master server. Therefore, when the master server could not work, the standby master server would become the master server. However, once the standby master server could not work, the system would crash down.

   When the master server crashed down and the system was changing to the standby master server, the system would have an operation to prevent the problems of this process. If the system started to change the master server, other sub-servers would start to timing. Once the system could not change the master server successfully in two minutes, the sub-servers would send the message of closing the servers to all the clients connected to them. At that time, the clients need to apply to the master server for login manually.

   3.2. The failure of sub-servers

   When a sub-server crashes down without sending any logout information to the current master server, the master server will capture such action and record it in its internal memory. The login record will exclude all the clients information connected to the crashed sub-server and mark them as offline status. As a result, the master server caches the original activity messages which have not been transmitted to the destination clients due to the disconnection. Hence, these messages are not lost but saved temporarily. When the crashed sub-server recovers from the abnormality, the activity messages stored in the caches will be released and sent to it and its corresponding connected clients progressively.

   Since the master server stores a pair array recording the unsent messages and the unique names of the clients, the next time it receives

a connected client information pair whose name is identical to any of the pair in the array, it means a previously crashed client recovers and reconnects again. As a result, the unsent messages in the pair will be sent to the newly connected client. However, this client may crash again during the process of the resending, so the pair in the array will be kept until a returning acknowledgement from a sub-server states that the newly recovered client has received the unsent messages eventually. Hence, even if a sub-server crashes and one of its previously connected client choose to reconnect to other available servers in the system, the unsent messages will be re-sent correctly.

3.3. The failure of clients

The login information of all the clients is stored and managed by the master server. When the clients crash or exit normally, the sub-servers will capture such action and send corresponding information to the master server. The master server then update the login information by changing those clients' status to offline accordingly. This can ensure that the status of all clients in the system are managed globally. It avoid the status of clients being stored wrong under situations like client exiting abnormally. This may happen if the login information were stored and managed by the sub-servers. The master server also manages the messages that should be received by the clients but aren't. This situation will happen when the clients exit abnormally. These messages will be sent to the corresponding client again when the next time it login the system. The system will keep sending these messages to corresponding client as long as it's online until it receives them. This prevent the messages never received if the client exit before receiving them.

4. Availability and Consistency

There are two methods used to ensure the consistency and the availability of the system. The first method is load balancing while the second one is uniformly managing registration information and the messages which could be not sent to some of sub-servers successfully.

In this system, load balancing means that the master server would keep the client numbers of different sub-servers in a state of equilibrium. When a client requires for logging in the system, the master server would check the status of each server and add the client to the sub-server which is in a unbalanced status. However, except the registration information and login information, all the other messages would be handled by the corresponding sub-servers.

There are some information and messages managed in the master server to keep the consistency of the system. The first type of information is the registration information and login information of all the clients. Although the clients are not directly connected to the master server, the master server still stores and manages these information. In the project one, the development team thought that each server should store the login information and registration information of all its clients, however, in such design, login inquiries and registration inquiries would be needed when there are some new clients. If login information and registration

information could be managed by the master server, when there are some new clients, the client only needs to apply to the master server. However, when the master server crashed down, the login information would not be synchronized to the standby master server and the standby master server needs to obtain the login status from each sub-server.

The second type of information is the messages which could be not sent to some of sub-servers successfully. When the clients or the sub-servers suddenly crash down, the master server would store the messages which could not be successfully sent to other servers or clients or received from other servers or clients. When the sub-servers or clients reconnect to the system, the messages managed by the master server would be re-sent or received. However, there is a similar problem as the login information. If the master server crashes down, these messages would not be synchronized to the standby master server.

5. Conclusion

The system tries to achieve a balance of availability and consistency simultaneously based on various failure models established previously. There is a standby master server implemented in case of the main master server fails. The number of clients connected to each of the servers in the system is kept balanced. A newly login client will be assigned to the server with least workloads. Moreover, the login information and unsent messages are totally kept and managed by the master server. In terms of the consistency, when a client reconnects to any servers in the system, it will receive the previous messages correctly due to the information kept in the master server. Compared with the system in Project I, the availability and consistency of it increases significantly.

APPENDIX

Meeting 1	
Date	May 5, 2018 2-6 pm
Venue	Eastern Resource Centre
Attendees	
. Kevin Zhang . Mincheng Li . Wenzhou Wei · Chaoran Zheng	
Objectives	
. To analysis and design the structure of the project 2 system and determine the division of labor	
Outcomes	
. Based on the code of project one and the requirements of project two, the team identified the tasks needed to be finished in the project 2. . The team identified the problems in the project one which were needed to be improved. . The team agreed on the determination of division of labor.	

Meeting 2	
Date	May 12, 2018 2-6 pm
Venue	Eastern Resource Centre
Attendees	
. Kevin Zhang . Mincheng Li . Wenzhou Wei · Chaoran Zheng	
Objectives	
. To analysis and improve the problems in the project one	
Outcomes	
. The team finished all the improvement of the project one. . The team finished	

the final design of the project two system.

Meeting 3	
Date	May 19, 2018 2-6 pm
Venue	Eastern Resource Centre
Attendees	
. Kevin Zhang . Mincheng Li . Wenzhou Wei · Chaoran Zheng	
Objectives	
. To analysis the problems in the coding of previous week and finish the coding on the basic functions of the project two system.	
Outcomes	
. The team solved the problems in the previous week. . The team finished the basic functions of the system.	

Meeting 4	
Date	May 26, 2018 2-6 pm
Venue	Eastern Resource Centre
Attendees	
. Kevin Zhang . Mincheng Li . Wenzhou Wei · Chaoran Zheng	
Objectives	
. To review all the code and write the document	
Outcomes	
. The team reviewed all the functions and finished the final test of the system. . The team finished the document writing.	