Report 4: Groupy

Chen, Zidi

October 1, 2020

1 Introduction

This assignment constructs a group membership service. Each node has a GUI with color. The node can decide to change the color and then multicast to all other nodes. Other nodes need to change the color according to the message. The color of all the nodes should be synchronized. The point for the project is to handle the newly joined nodes and crashed nodes.

2 Main problems and solutions

In this project, the code are almost given. The only problem is to understand what is the code doing. gui module is easy to understand. But how the message goes between worker module and gms module is a little bit hard to understand. After the discussion with classmates and reading the document for a lot of times, I managed to figure out what are these modules doing. Another important thing is that this project helps me to understand the application layer and group layer architecture.

3 Evaluation

3.1 First implementation

In the first implementation(gms1), there is no fault tolerance in the program.

Figure 1 shows the result of three workers. W1 is the leader. These three processes are synchronized in color.

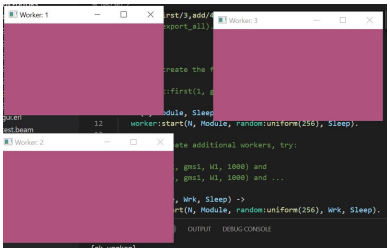

Figure 1: result of first implementation

When one team member crashes, there is no influence. But when the leader crashes, all the members stop changing color. This is because there is no leader in the group to multi-cast message and send message to gui module.

3.2 Second Implementation

In gms2 module, there is a error detector to monitoring the crash of leader. If the leader crashes, then a new leader will be generated by election process. In this way, the nodes can still be synchronized after the leader crashes. Figure 2 shows when worker 1(leader) crashes, the nodes are still displaying the same color.

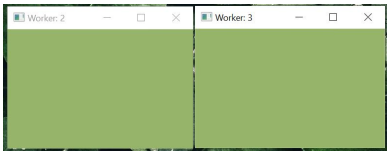

Figure 2: result of second implementation

When the crash simulation is added to the program, the nodes sometimes go out of synchronization. The leader sends the message from the first node in the list. Then, it might crash. In this case, some nodes may receive the message, but some nodes may not. This is the reason why the nodes sometimes are out of synchronization.

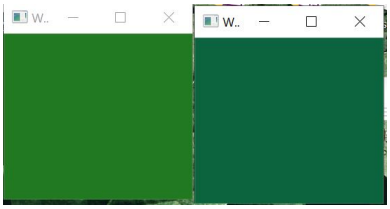

Figure 3: result of crash simulation in second implementation

3.3 Reliable multi-cast

In this implementation, the nodes have the sequence number of the message and the copy of the last message from the leader. As figure 4 shows, the workers are always synchronized even the leader crashes half way of broadcasting.

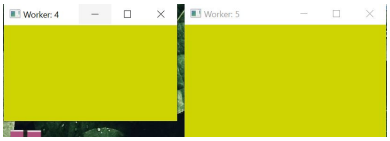

Figure 4: result of reliable multi-cast implementation

3.4 bonus task

I tried to solve the bonus task, but not very successful. I have implemented a simulation of message lost when leader is sending the message. Then, I think that if the slave receives a message whose sequence is larger than expected, this means I lost some messages and I need to ask the leader to resend them. But I am confused that the process of re-sending also need time. This short difference in time also cause being out of synchronization.

leader在resend的时候可以给所有人发，如果收到过这条消息，slave就忽视这条消息。这样可以同步。

4 Conclusion

In this assignment, I tried a group membership service based on the application layer and the group layer. I also got in touch with the graphic user interface in erlang. To get a reliable group service, I really need to take a lot into consideration.