Report 4: Homework Report groupy

Marcus Samuelsson

October 4, 2023

1 Introduction

This report covers the implementation of a Group Membership Service(**GMS**). This helps with running concurrent events, joining of new nodes and crashed of nodes in a distributed system. This is done with atomic *multicast* to handle multiple application layer processes. This is all shown through a GUI displaying a color, which should be synced between all of the nodes. When a node wants to change color the node need to contact a leader which will broadcast the new color to all other nodes. The topic of this report is the problems that may occur when setting up such a system and how to solve them. This is done through three steps shown in the evaluation.

2 Main problems and solutions

This task did not require a lot of code that was not already provided through the instructions which made the amount of issues few. The more difficult part was in module **gms3**, where issues occurred where the colors of the workers got out of sync when the leader was shut down. These issues steamed from copying the content from **gms2** and missing changes required for the new additions to the file. For the colors being out of sync the problem was that the **election/6** function did not broadcast correctly with the new addition of **N** for expected sequence number.

Another issue of the same cause was that it was not possible to start more then one worker. This issue was caused by missing changes in the **slave** function. The same issue as in the election function was found here which caused the problems. This issue was also present in module **gms2** for the function **slave**. The solution was adding *erlang:monitor/2* to the *init/4* function of the slave in the file.

3 Evaluation

In this task we created three different **GMS**. The first version with no election, the second with election but no sequence and the third with election and sequence. This is

all tested through a GUI provided for the task, which changes color depending on what the leader tells the slaves to be. If the colors match the program works. For each version 10 windows where opened for different workers with the first being the leader. Then the leader is closed to see how the program handles it.

3.1 First version with no election

In this version the program(module **gms1**) works with the colors changing simultaneously, but as there is no election the program fails when the leader is shut down. There is also no guarantee that the colors change in the correct order as the program does not contain sequencing as in Section 3.3.

3.2 First version with election

In this version of the program(module **gms2**) the colors change correctly again, but now when the leader is closed a new leader is appointed and the program keeps running. This was done through detecting the crash of the leader and then calling an election to appoint a new leader of the group. However because there is no sequencing the program will sometimes assign multiple leaders over different slaves, which causes the workers to show different colors.

3.3 First version with election and sequence

This version of the program(module **gms3**) is able to handle crashes and appointing a new leader for all nodes correctly. However on issue still persists where some slaves do not receive the messages and miss a color change causing them to be a different color for a little bit of time before receiving another message.

4 Conclusions

This assignment was easier then the previous as mentioned in Section 2. This task gave insite to how *Erlang* delivers messages in a FIFO order. Further more. it showed how important node synchronization is between nodes, the problems that can occur with it and some solutions on how to solve some of them. Due to time constraints **gms4** which would solve the issue with missing messages was not completed. However, a possible solution could be to make the leader wait for acknowledgements from each slave when sending a message, witch would cost in terms of resources and speed for the program but could potentially solve the issue.