Report 4: Groupy - A Group Membership Service

Michail Roussos, michailr@kth.se

October 4, 2023

1 Introduction

In this exercise we implemented a group membership service that provides atomic multicast. The nodes in the system we implemented multicast the state change so that the other nodes can change state with them, in order for all the nodes to perform the same state changes. A node that wishes to perform a state change must first multicast the change to the group so that all nodes can execute it. Since the multicast layer provides total order, all nodes will be synchronized.

2 Main problems and solutions

We had to implement the following modules:

1. The first module we had to implement was *gms1*. This was the simplest version were we do not deal with failures of nodes but only with adding new nodes to the system.

2. The second module we had to implement was *gms2*. In this module we will add some functionality to monitor the leader and elect a new leader when the old one fails.

3. The third module we had to implement was *gms3*. In this module we will try to make the multicast more reliable by keeping track of the messages and by keeping a counter/ID. So if the leader dies before sending a message to all the nodes then the new leader can resend it and if some nodes have received this message they know to ignore it because the have already processed a message with that ID. In the third module I also implemented the reliable broadcast functionality.Reliable broadcast means that if any alive nodes receive a message then all alive node should receive.

Each of the above nodes could act as a Leader or as a Worker in the system. When a Leader fails we can do en election to select a new Leader.

When a node wants to join the network he contacts the Leader and they can be added. Each node can send a message to the Leader and the Leader will multicast this message to the network.

3 Evaluation

As expected the first version works for a bit but as soon as the leader fails every thing falls apart.

In the second version as time passes new leaders are elected when the old ones fail, but the nodes are not kept in sync.

The third version works better and the nodes remain in sync for longer.

4 Conclusions

In this assignment we had the opportunity to further our Erlang skills, we saw how a multicast can be implemented and had the oportunity to understand somethings about the comunication between processes in Erlang.