Report 4: Groupy

Chen Sihan

October 1, 2020

1 Introduction

The goal of this project is to implement a multicast group communication application.

2 Main problems and solutions

2.1 Situation 1: no election

Firstly, I implement a basic application without election. As shown in the

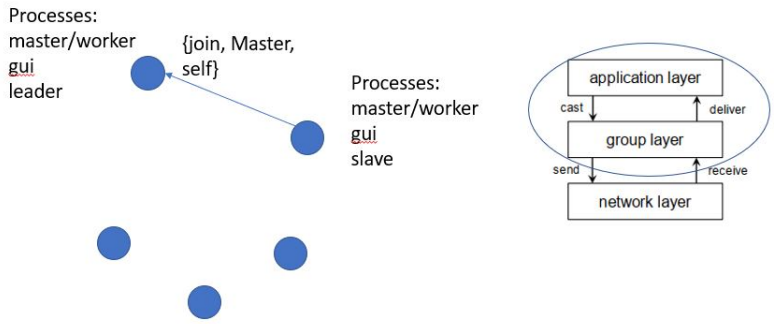

Figure 1: basic multicast/deliver

graph, each node represent a process in the application layer. Each application process will create a group process which plays the role of leader or slave. Firstly the main leader process should be created, and then the slaves processes should be created and join to the leader process. Here we maintain a peer list and the first element of it is the leader process.

To keep consistent, when every slave want to change the state(color), it will send message to the leader and the leader will broadcast to every slave and keep their states to be the same.

However, the disadvantage is that when the master crash, the slave will immediately lose connection with each other and they cannot keep consistent any longer.

2.2 Situation2: with election but no sequence

To relieve the situation 1, we need to implement the election process when the leader crash. Firstly we need to monitor the death of leader by Erlang monitoring mechanism. Each slave will monitor the leader and once it receive the 'DOWN' message, it will start the election process. The first node in the group(except the leader) will be the new leader and other slaves alive will monitor it. It will play the role of the new leader.

However, the disadvantage is that during B-multicast, the leader dies before it sends messages to all the slaves. Some slaves receive the messages but others do not. In that case, when a new leader is elected out, the nodes are all in different states. They will be inconsistent from then on.

2.3 Situation3: with election and sequence

In that case, we must make sure that the last message should be sent to all the nodes in the group (not part of the group). As shown in the above graph,

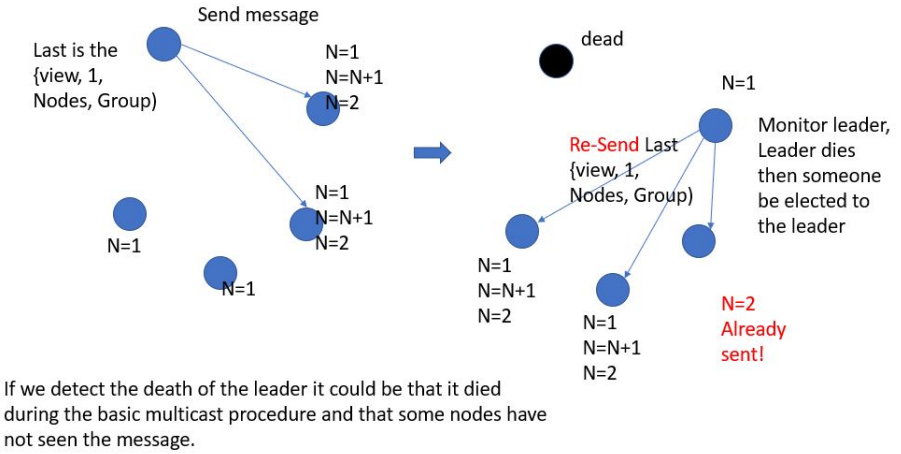

Figure 2: If we detect the death of a leader it could be that it died during the basic multicast procedure that some nodes have not seen the message

once the message is sent to one slave, a sequence number N on the slave side should be set to plus 1. If the leader send message to only two slaves in the group, suddenly the leader crash and the new leader must multicast this message again (re-send) to all the nodes. The original two nodes that has already received the message will not deliver the message because the

current number N is smaller than their number N. The other nodes that have not received the message will surely update their state.

In that way, the nodes keep consistent.

3 Bonus: Reliable multicast

The above solution cannot make sure that the messages are successfully sent to the nodes. We cannot deal with the situation when unstable net connection cause the lost of the messages.

Therefore we should implement a holdback queue at the leader side to ensure that the lost messages are backuped and once the slave restore, it should sequentially deliver the messages in its queue. The detailed explanation is in the code.