Report 4: A Group Membership Service

Luxiang Lin

October 1, 2024

1 Introduction

In this assignment, we focus on developing a group membership service to illustrate reliable multicasting. The key objective is to synchronize multiple application layer processes so they execute the same sequence of state changes while exchanging messages. This synchronization must be maintained despite the dynamic addition and removal of nodes within the network. The performance of the implement can be shown by the the visualization provided by GUI.

2 Main problems and solutions

To fulfill the requirements of this assignment, which involves developing a reliable group membership service, I have followed the instructions provided in the assignment description and separated the implementation into four steps.

2.1 The first implementation

The first implementation consists of basic functionality: nodes joining and message transmission.

2.2 Handling failure

We can learn little from the first implementation, as it is too simple and unlikely to encounter any errors. Therefore, following the instructions, I introduced a probability of leader crashes and implemented the election of a new leader. Once a leader crash is detected, a new leader is elected from the existing nodes. Despite these changes, the group could no longer stay synchronized because the second implementation cannot handle missing messages.

2.3 Reliable multicast

The third step is to handle the desynchronization caused by missing messages. To tackle this problem, I had each node keep its last message. Then, every time a leader crashes, the newly elected leader should broadcast its last message. Each slave node should update itself according to this message if it is newer than the one it kept. The result seems correct, as the whole group remains synchronized even when one of the nodes crashes.

2.4 Possibly lost messages

To fulfill the additional requirement of handling potential lost messages, I implemented an 'ack' message system. Every time the leader broadcasts a message or a view update message, it sets a timer and expects each node to respond with an 'ack' message within a specified time frame. If some nodes do not respond in time, the message sent to these nodes is considered lost. The leader will then resend the message until it receives an 'ack' message from the expected nodes.

3 Evaluation

3.1 gms1

In the first implementation, all the nodes are synchronized after execution, as the first implementation does not involve fault generation.

3.2 gms2

When leader crashes were introduced into the system, the group could no longer stay synchronized due to the lack of fault tolerance.

3.3 gms3

After solving the problem caused by missing messages, the group is able to remains synchronized even when one of the nodes crashes.

3.4 gms4

By applying an 'ack' message system, message transmission becomes even more reliable. The whole group stays synchronized regardless of whether a node crashes or messages are lost.

4 Conclusions

This assignment has offered me significant insights into managing synchronization challenges in real-world systems, even if on a smaller scale. I have gained a deeper understanding of reliable multicasting, its implementation, and the complexities involved in message passing within group membership service.