Report 4 - Groupy: Group Membership Service

Lorenzo Deflorian

September 27, 2025

1 Introduction

The main goal of the assignment was to implement a group membership service providing atomic multicasting in view syncrony, while also tolerating crash failures.

2 Main problems and solutions

2.1 Dancing together

Group communication in our implementation relies on a leader-based protocol. The leader is responsible for maintaining a consistent view of the group membership and multicasting messages to all members. When a peer wants to join, it sends a request to the leader, who then updates the group view and shares it with everyone. The view consists of two parts: the list of all **process identifiers in the group**, and the list of **process identifiers at the application layer**.

Peers that are not the leader act as slaves. They forward join requests and multicast instructions to the leader, and also relay messages from the leader to the application layer. This setup ensures that all group members have the same view and that messages are delivered reliably, even if some processes crash.

Using our first implementation gms1, we can see that the group communication works as expected and all the peers change the colors in sync. However if the leader crashes the whole system stops working.

2.2 Handling failures to keep on dancing

To address the issue of leader crashes, we use monitors to detect failures. Each peer monitors the leader, and if it detects that the leader has crashed, it initiates a leader election process to select a new leader from the remaining members. The new leader then takes over the responsibilities of maintaining the group view and multicasting messages.

With this system in place we can see that if the leader crashes, one of the slaves takes over and the system continues to function without interruption. The group members continue to change colors in sync, demonstrating that the group communication is resilient to leader failures.

The implementation of gms2 shows the presented improvements and the system can handle leader crashes effectively.

2.3 Showing that we cannot dance

While our gms2 implementation handles leader crashes effectively, it fails to guarantee true view synchrony when failures occur during message broadcasting. To demonstrate this limitation, we introduced random crashes in the broadcast function (gms2p5), where the leader might crash after sending a message to only some slaves.

This creates a critical problem: when the leader crashes mid-broadcast, some group members receive the message while others don't, leading to inconsistent states. Even after a new leader is elected, the group remains permanently out of sync because there's no mechanism to detect or recover from partial message delivery. This violates the reliability property of view synchrony, which requires that if any correct node delivers a message, all correct nodes must deliver it.

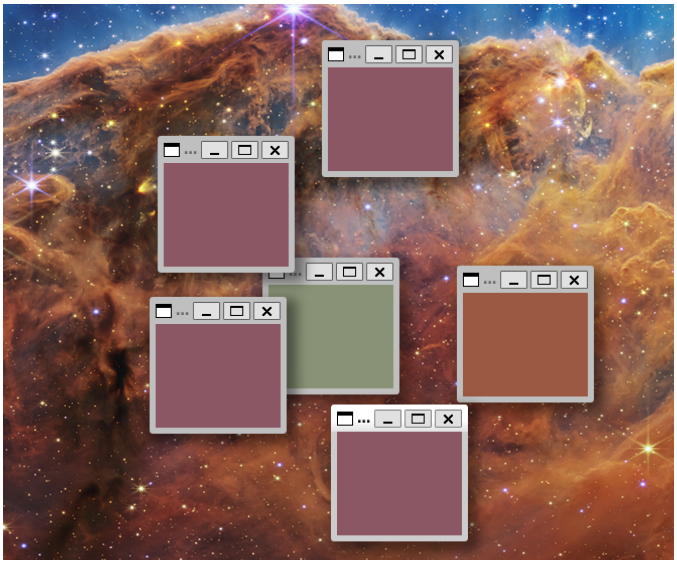

Figure 1: Peers out of sync after crash mid-broadcast.

3 Conclusions