University of St Andrews

Distributed Systems

CS4103

Ring-Based Distributed System

Author:
150008022

March 29, 2019

Goal

To demonstrate an understanding of leader election and mutual exclusion in distributed systems by developing a ring-based distributed social media application.

Contents

1 Initial Set-up

Java 8 was chosen for this project due to it's friendly socket API, and Maven [1] was used as the build tool. TDD was implemented with JUnit4 as the the test suite framework.

The project was started with the intention of implementing the bully algorithm as an extension, as it provided fault tolerance with low overhead, but also to provide the option to use the ring based algorithm for comparison.

1.1 Configuration

Since each node would be running on an isolated machine, the planned testing environment would involve using *ssh* to start the nodes remotely. Providing the configuration as command line arguments was considered simpler than other methods, and was implemented using the Apache Commons CLI library [2].

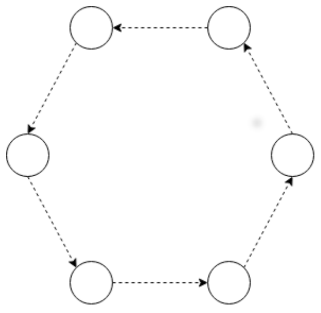
1.2 Communication Protocols

Due to different patterns of communication during recovery and regular message passing phases both TCP and UDP were used for this project, as shown in Figure 1.
TCP was used for communication **around** the ring:

1. Reliable communication avoids token being lost.
2. Keepalive functionality provides effective failure detection.
3. Connection is reused frequently between predecessors and successors, justifying handshake overhead.

UDP was used for communication **across** the ring:

1. Low communication overhead allows for messages to be sent to multiple nodes quickly, improving recovery time from node failure.
2. No session maintenance allows coordinator to handle more members in the ring.


(a) TCP Around Ring

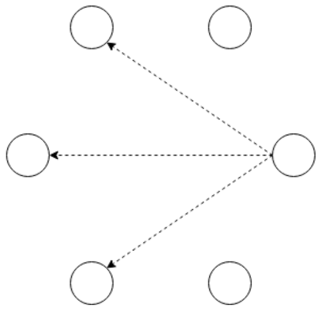

(b) UDP Across Ring

Figure 1: The different types of communication used

2 Initialization

On initialization, each node sends out join messages to the coordinator. The joining protocol is similiar to that described in [3], and is shown in Figure 2.

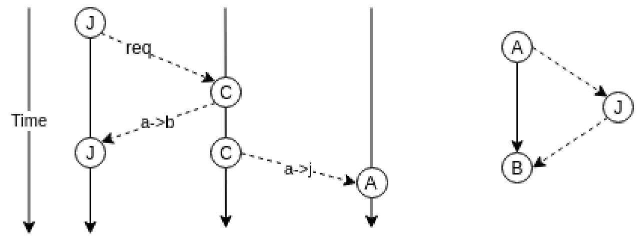


Figure 2: Joining protocol over time, with topology shown on right.

When node J wants to join the ring:

1. J sends join request to coordinator C.
2. C sends successor to J, telling it to connect to B.
3. C sends successor to A, telling it to connect to J.
4. A disconnects from B, B begins listening for new predecessor.
5. J connects to B, A connects to J.

Since TCP with keepalive is used around the ring, a node will notice the successor/predeccesor disconnect quickly.

3 Leader/Coordinator Election

4 Receive/Send Posts

References

[1] Apache. Apache maven project. `https://maven.apache.org/`.

[2] Apache Commons. Apache commons cli. `https://commons.apache.org/proper/commons-cli/`.

[3] D. Lee, A. Puri, P. Varaiya, R. Sengupta, R. Attias, and S. Tripakis. A wireless token ring protocol for ad-hoc networks. In *Proceedings, IEEE Aerospace Conference*, volume 3, pages 3–3, March 2002.