SLR203 - Chang and Roberts Leader Election Algorithm

Luca Erbí

January 27, 2024

1 Introduction

The Chang and Roberts algorithm is a ring-based coordinator election algorithm employed in distributed computing. The algorithm assumes processes arranged in a unidirectional ring with a communication channel between each process and its clockwise neighbor. This report discusses the testing process and results for the implementation of this algorithm using Akka in Java.

2 Testing Strategy

The testing strategy involves validating the correctness and performance of the Chang and Roberts algorithm. The following aspects were considered during testing:

1. **Correctness:**. Ensure that only one leader is elected and then confirm that the elected leader has the highest UID among the processes. Finally, Validate that the elected leader broadcasts the election result to all nodes in the ring.
2. **Performance:**. Assess the algorithm's efficiency in terms of message complexity evaluating the impact of varying the number of nodes in the ring on the algorithm's performance. (3N-1 sequential messages, worst case)

The test cases will involve:

1. . Simulating a scenario where a process notices a lack of leader and starts an election and then verifying that the algorithm correctly elects a leader and broadcasts the result.
2. . Testing the algorithm with topologies changes, such as a deleting nodes. Confirm that the algorithm adapts to various topologies without compromising correctness.
3. . Measuring the number of messages exchanged during the election in different scenarios to assessing the impact of network size and the correctness of the algorithm.

3 Results

3.1 Test 1

```
1     private static void scheduleElection(ActorRef node, ActorSystem system, int delay) {
2         system.scheduler().scheduleOnce(
3                 Duration.create(delay, TimeUnit.MILLISECONDS),
4                 () -> node.tell(new StartElectionMessage(), ActorRef.noSender()),
5                 system.dispatcher()
6         );
7     }
8     private static void scheduleStopAndSetNextNode(ActorRef node, ActorRef beforeNode, ActorRef
          nextNode, ActorSystem system, int delay) {
9         system.scheduler().scheduleOnce(
10                Duration.create(delay, TimeUnit.MILLISECONDS),
11                () -> {
12                    beforeNode.tell(new SetNextNodeMessage(nextNode), ActorRef.noSender());
13                    system.stop(node);
14                    System.out.println(node.path().name() + " being deleted");
15                },
16                system.dispatcher()
```

```
        );
    }
    // Simulate a process noticing a lack of leader and starting an election
    scheduleElection(node1, system, 500);
```

```
Node 1 is starting the election!
Node 2 received ElectionMessage from Node 1
Node 3 received ElectionMessage from Node 2
Node 4 received ElectionMessage from Node 3
Node 5 received ElectionMessage from Node 4
Node 6 received ElectionMessage from Node 5
Node 1 received ElectionMessage from Node 6
Node 2 received ElectionMessage from Node 6
Node 3 received ElectionMessage from Node 6
Node 4 received ElectionMessage from Node 6
Node 5 received ElectionMessage from Node 6
Node 6 received ElectionMessage from Node 6
Node 6 starting second phase and sending ElectedMessage to Node Node1
Node 1 received ElectedMessage from Node 6
Node 2 received ElectedMessage from Node 6
Node 3 received ElectedMessage from Node 6
Node 4 received ElectedMessage from Node 6
Node 5 received ElectedMessage from Node 6
Node 6 received ElectedMessage from Node 6
Node 6 is the leader!
```

In this test, we used N = 6 nodes, and run the algorithm as the worst scenario. As expected, the count of the messages is 3N-1 = 17 messages.

3.2 Test 2

```
    // Simulate three processes noticing a lack of leader and starting an election
    scheduleElection(node1, system, 1000);
    scheduleElection(node2, system, 1000);
    scheduleElection(node4, system, 1000);
```

```
Node 1 is starting the election!
Node 2 is starting the election!
Node 4 is starting the election!
Node 2 discarding ElectionMessage
Node 3 updating UID and forwarding ElectionMessage to Node Node4
Node 5 updating UID and forwarding ElectionMessage to Node Node6
Node 4 discarding ElectionMessage
Node 6 updating UID and forwarding ElectionMessage to Node Node1
Node 1 forwarding ElectionMessage to Node Node2
Node 2 forwarding ElectionMessage to Node Node3
Node 3 forwarding ElectionMessage to Node Node4
Node 4 forwarding ElectionMessage to Node Node5
Node 5 forwarding ElectionMessage to Node Node6
Node 6 starting second phase and sending ElectedMessage to Node Node1
Node 1 received ElectedMessage from Node 6
Node 2 received ElectedMessage from Node 6
Node 3 received ElectedMessage from Node 6
Node 4 received ElectedMessage from Node 6
Node 5 received ElectedMessage from Node 6
Node 6 received ElectedMessage from Node 6
Node 6 is the leader!
```

In this test, we used N = 6 nodes, and run the algorithm with 3 parallel election starting. Also with parallel request, the algorithm has been able to elect just one leader and the one with the biggest UID.

3.3 Test 3

```
    // Simulate two processes noticing a lack of leader and starting an election, then the
        biggest UID stop
    scheduleElection(node1, system, 1500);
    scheduleElection(node2, system, 1500);
    scheduleStopAndSetNextNode(node1, node6, node2, system, 1500);
```

```
Node 1 is starting the election!
Node 2 is starting the election!
Node 3 updating UID and forwarding ElectionMessage to Node Node4
Node 2 discarding ElectionMessage
Node 4 updating UID and forwarding ElectionMessage to Node Node5
Node1 being deleted
Node 5 updating UID and forwarding ElectionMessage to Node Node6
Node 6 updating UID and forwarding ElectionMessage to Node Node2
Node 2 forwarding ElectionMessage to Node Node3
Node 3 forwarding ElectionMessage to Node Node4
Node 4 forwarding ElectionMessage to Node Node5
Node 5 forwarding ElectionMessage to Node Node6
Node 6 starting second phase and sending ElectedMessage to Node Node2
Node 2 received ElectedMessage from Node 6
Node 3 received ElectedMessage from Node 6
Node 4 received ElectedMessage from Node 6
Node 5 received ElectedMessage from Node 6
Node 6 received ElectedMessage from Node 6
Node 6 is the leader!
```

In this test, we used N = 6 nodes, and run the algorithm with 2 parallel election starting and deleting node 1 in parallel. Also with deletion during the execution, the algorithm has been able to elect just one leader and the one with the biggest UID.

3.4 Test 4

```
    // Simulate two processes noticing a lack of leader and starting an election, then the
        biggest UID stops
    scheduleElection(node3, system, 2000);
    scheduleStopAndSetNextNode(node6, node5, node2, system, 2000);
```

```
Node6 being deleted
Node 3 is starting the election!
Node 4 updating UID and forwarding ElectionMessage to Node Node5
Node 5 updating UID and forwarding ElectionMessage to Node Node2
Node 2 forwarding ElectionMessage to Node Node3
Node 3 forwarding ElectionMessage to Node Node4
Node 4 forwarding ElectionMessage to Node Node5
Node 5 starting second phase and sending ElectedMessage to Node Node2
Node 2 received ElectedMessage from Node 5
Node 3 received ElectedMessage from Node 5
Node 4 received ElectedMessage from Node 5
Node 5 received ElectedMessage from Node 5
Node 5 is the leader!
Node 2 received new leader: Node 5
Node 3 received new leader: Node 5
Node 4 received new leader: Node 5
```

In this test, we used N = 4 nodes, and run the algorithm with 1 election starting and deleting node 6 in parallel Also with deletion during the execution, the algorithm has been able to elect one leader and the one with the biggest UID at that moment. To then communicate to everybody the new leader. In the end, if we count the number of messages they are 10 = 3N-2, because in this case the worst scenario would start by node 2, and with that we would get the 11 communications.

4 Conclusion

In summary, the implemented Chang and Roberts algorithm in Akka exhibits robustness, correctness, and scalability in different tests and scenario.