Simulation of Distributed Leader Election Algorithms

B.Tech VI Semester, Department of Information Technology And Business Informatics

PRATYAKSH SINGH IIB2020015

CHAHIT GAWRE IIT2020094

SHAILESH TIWARY IIB2020001

DEBANSHU SHEIKHR MALIK IIT2020013

Supervised by :-
DR. ANSHU ANAND

Abstract :-

This project presents a program for simulating the execution of distributed leader election algorithms in a ring-network. The application has been developed using the Python programming language and utilizes Python sockets, Python select, Distributed Systems Leader election algorithm, and Bully Leader election algorithm. The purpose of this project is to evaluate the performance of different leader election algorithms in a distributed system by measuring metrics such as the time taken to elect a leader, the number of messages sent and received, and the number of failures and recoveries.

Introduction :-

Distributed leader election algorithm is the most famous algorithm and most used in distributed systems to elect a leader across the nodes. The aim is to have a node selected which can coordinate the activities of the nodes in the distributed system. There are several different different leader election algorithms, each algorithm with their own pros and cons.
To simulate this distributed leader election algorithm, first you need to choose an algorithm, then Some common algorithms include Bully algorithm, Ring algorithm, and Chang and Roberts algorithm. Once you have selected an algorithm, after that create a simulation environment that includes a number of nodes, each running node an instance of the algorithm.

Every node in the simulation would need to be programmed to follow the rules of the algorithm, it should be ensured that including sending messages to other nodes and processing incoming messages. The simulation should also ensure mechanisms for nodes to join and leave the system, as well as for nodes to detect failures and recover from them.

In this algorithm to evaluate the performance of the algorithm to elect the leader, first we measure metrics as how much time it takes to elect the leader in the process, how many messages sent and received and if the how many times it failed and how much time it's taking for recoveries. Failure and recovery depends a lot on the algorithm which can sometimes cause varying network latencies or systems failure rate.

<u>Technology used</u> :-

In our project we have used many technologies for our implementation which are–

Sockets :- Socket programming is widely used in distributed systems, which is used to establish networks in distributed systems to communicate with different different nodes which are in the network. To create the socket, it contains or we have to pass the first parameter is AF_INET and the second one is SOCK_STREAM. We use AF_INET to the address-family ipv4 and The SOCK_STREAM means connection-oriented TCP protocol.

Select :- The ‘select’ module is Python dependencies which provides a speciality for multiple input output operations in a network. It selects the first message which will first get a response,which ones are ready for reading or writing.

Distributed System :- Distributed system provides the platform to a node to work collectively to achieve the same target. To simulate an election algorithm is the most famous algorithm and most used in distributed systems to elect a leader across the nodes. The aim is to have a node selected which can coordinate the activities of the nodes in the distributed system.

<u>Distributed Techniques</u> :-

We have used some selected algorithms to execute the program. There are many algorithms to simulate the execution of distributed leader election algorithms. The two we used are -

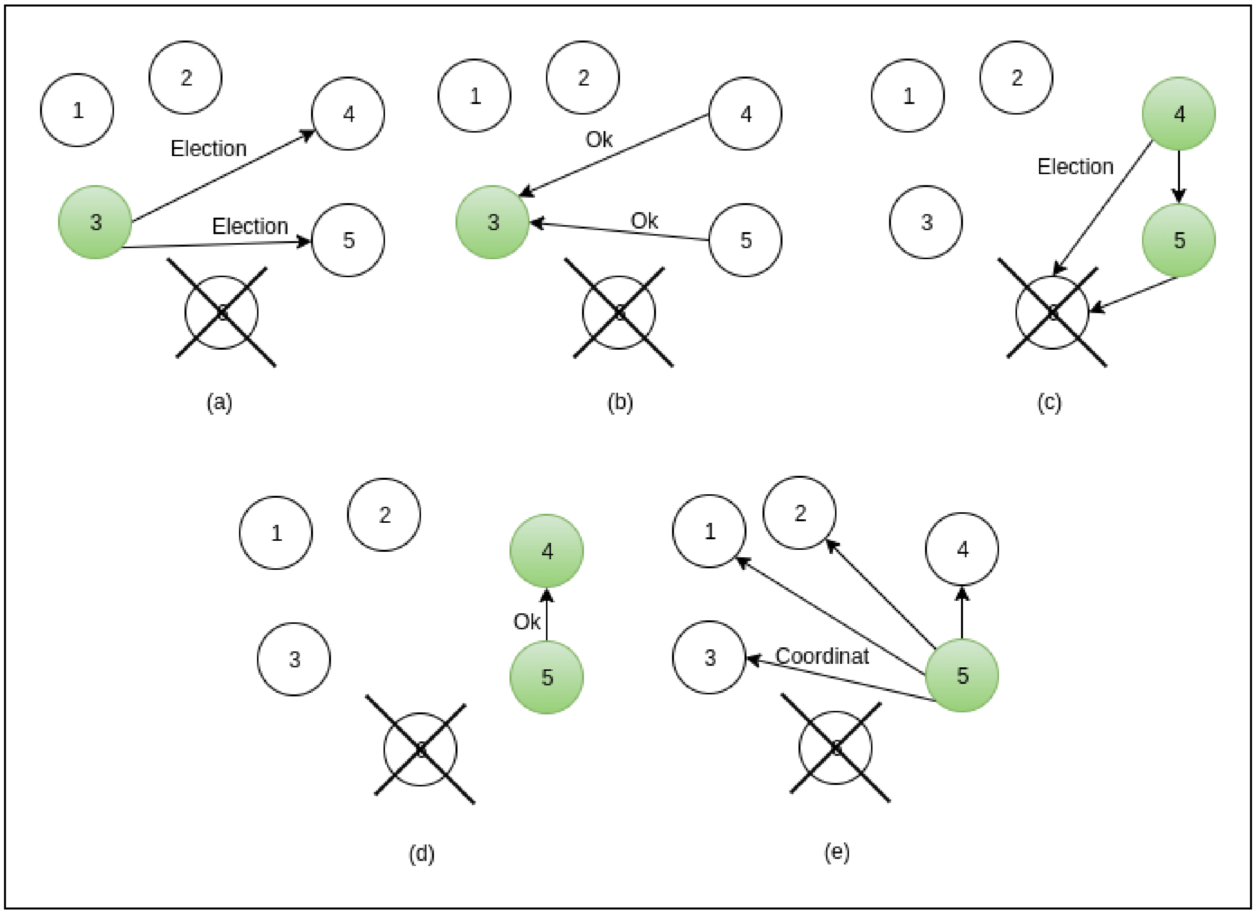
- Leader Election
- Bully Algorithm

Leader Election - Leader election provides a brief idea to elect the leader in a ring network. It elects the leader according to measurement of the activities of the nodes. Leader election algorithm is the simple idea of giving one thing (a process, host, thread, object, or human) in a distributed system some special ability. This ability is used to assign work as leader to the node. The ability to modify a piece of data, or even the responsibility of handling all requests in the system. This is the functionality of the ‘Leader Election Algorithm’ , which we have used in our implementation.

Bully Algorithm - In a distributed system, the bully algorithm is a brief technique to dynamically select a coordinator or leader from the network of collection of system or nodes. The process with the highest process ID number from amongst the non-failed processes is selected as the coordinator.

The algorithm follows the below message types for the process:

- **Election Message:** Sent to announce the election.
- **Answer (Alive) Message:** Responds to the Election message.
- **Coordinator (Victory) Message:** Sent by winner of the election to announce victory



Implementation Details

There are two types of nodes

- Genesis Node : A node that starts a cluster
- General Node : A general node is part of a cluster

The model follows the following architecture

- If a node is genesis then starts a cluster with that node.
- For all the other node the following process will be followed
    - Connect with the leader node and get all the peers in the network
    - Specify the id of the peer as the one provided by the leader
    - Connect with all the peers provided by the leader and then open its own server for connection.

The novelty of this method is that it uses only one thread to deal with all the peers. It uses select to deal with all the peers.
Whenever the leader node fails then a new node is selected based on the consensus algorithm provided. Currently the model uses the Bully Algorithm to decide the new leader but any other leadership election can be implemented. Following these standards

- The class should take the network as input.
- It should implement the following class
    - check_win() : Will be called every time there is a state change
    - process_message(Message): will be called every time there is an election message.

Paxos

Asynchronous leader elections can be conducted using the general consensus protocol known as Paxos. The Paxos algorithm's goal is to have network nodes agree on which node should be selected. Should take the helm. A node proposes to be the leader in this process, while other nodes in a network vote in favor of or against it. A node becomes the leader if the majority of other nodes vote for it. Otherwise, a node that failed to become the leader tries again. A node doesn't become the leader unless it receives a majority of the vote.

RAFT

Raft is a well-liked substitute for Paxos because it is simpler to figure out, execution, and use. Each Raft consensus node participates in the non-blocking method by maintaining track of its present "election term." Each node increases its copy of the term number when the leader election starts while also keeping an eye out for communications from other nodes. After a random interval if the node doesn't receive any messages, it becomes a candidate leader and solicits support from other nodes.