The American University in Cairo

School of Sciences and Engineering

Computer Science and Engineering

CSCE 4411 – Distributed Systems - Fall 2023

Project Milestone I: Distributed Election

Tamer Osman: 900192103

Mostafa Lotfy: 900192397

Salma Zaghloul: 900183256

Table of Contents

Introduction

The need for distributed election mechanisms arises in scenarios where multiple nodes collaborate to achieve common goals, such as managing resources, coordinating tasks, or maintaining system integrity. These nodes must rely on a set of algorithms and protocols to designate one node to be elected, thereby streamlining communication and decision-making processes within the network. The election of a node is not only essential for efficient system operation but also for ensuring resilience in the face of node failures or network partitions. These algorithms play a critical role in maintaining system stability, reliability, and efficiency. This report will present a comprehensive survey of various distributed election algorithms, shedding light on their characteristics, use cases, and trade-offs.

Ring Algorithm

There are several ring-based algorithms used in distributed computing and networking, each designed to solve specific problems or perform particular tasks within the context of a logical ring topology. The Chang and Roberts algorithm, introduced in 1979, stands as a seminal solution for the distributed leader election problem, specifically tailored for collections of processes organized in a logical ring topology. This algorithm offers a straightforward and elegant approach to electing a leader among interconnected processes or in general just electing a node to do a task or fail or anything. Each process initially lies in a dormant state until an election is triggered by a process. The election message circulates around the ring, and each process makes a local comparison of its unique identifier with the received message. The process with the highest identifier ultimately assumes the leadership role, setting itself as the leader and sending a victory message that traverses the ring. The simplicity of the algorithm, alignment with ring topologies, inherent distribution, scalability, and fault tolerance have cemented its status as a

foundational tool for scenarios where logical ring structures define the communication flow among processes. Whether applied in computer networks or distributed systems, the Chang and Roberts algorithm serves as a reliable mechanism for the orderly selection of leaders in a distributed and decentralized environment. While the ring-based algorithm is useful for understanding the properties of election algorithms in general, the fact that it tolerates no failures makes it of limited practical value. However, new mechanisms suggest different reliable fail-safe mechanisms, like traversing the ring in the other direction for instance [1].

Bully Algorithm

In this algorithm, each node within the system is assigned a different identification number upon its creation, serving as a priority indicator. The algorithm is based on a prioritization system based on the identification number such that the process with the highest priority is elected as the coordinator during elections. The process with the highest priority can declare itself as the coordinator or initiate an election by sending messages to higher-priority processes. If no response is received within a specified time, it becomes the coordinator. If a new coordinator is acknowledged, it continues; otherwise, it triggers another election. When a process with a higher priority is launched, it can assert itself as the coordinator, even if the current coordinator is still operational, which is why it's called the "bully" algorithm. This algorithm is resistant to single points of failure. It ensures graceful transitions of coordinatorship, and provides a means to notify all nodes of changes within the distributed system. However, an issue that may surface in this algorithm pertains to scenarios in which multiple nodes perceive themselves as having the highest priority and consequently attempt to assert themselves as coordinators simultaneously [1].

Invitation Election Algorithm

The Invitation Election algorithm operates as a leader election algorithm within an asynchronous system. In such asynchronous systems leader selection revolves around the concept of groups, where a group represents a collection of nodes that reach a consensus on a leader. This idea attempts to solve the problem of multiple nodes seeking to become a coordinator such that it permits multiple nodes aspiring to take on the role of coordinators to initiate group formation attempts with other nodes. Nodes within a group collectively assume responsibility for tasks within that group. Just like the case in Bully Algorithm, each node has a unique identifier called "priority," and they keep track of their status, the group they belong to, and who the leader of that group is [2]. Every now and then, if a node is not already a leader, it checks if the leader of its group is still alive by sending a message and waits for a response. If it doesn't get a response in time, it assumes something went wrong and takes actions to recover. A leader node, also, checks if other nodes are leaders. If it finds out that there are multiple leaders, it pauses for a period of time inversely proportional to its priority, helping to prevent multiple nodes from initiating elections concurrently, and then tries to merge the groups together. Other leaders receiving the invitation inform their group members and wait for a reply to join the new group. Eventually, only one group will exist, with a coordinator being the node with the highest priority. Ideally, all nodes would be members of the same group; however, high latency and network disruptions creating partitions make such a scenario unfeasible. Hence, multiple groups exist. This algorithm helps nodes in a network to work together and avoid problems when multiple nodes want to be in charge [3]. The Invitation Algorithm for leader election in distributed systems is advantageous for its scalability and fault tolerance, allowing multiple leaders and groups. However, it is complex to implement, and relies on frequent message

exchanges, potentially leading to latency and network overhead. Further, it offers relative consistency within individual groups, rather than ensuring global consistency. It might not work optimally in scenarios with persistent network partitions. Its suitability depends on the specific system's requirements and characteristics [2].

PALE: Time Bounded Practical Agile Leader Election

This algorithm terminates within a finite time frame, even in adverse node stability scenarios, assuming the presence of only one dependable node, all without depending on clock synchronization or a synchronizer layer. Its main principle is based on the **bully algorithm** in which more computationally capable nodes become the leaders. In other words, the node rank is assigned based on each node’s computational resources such as CPU speed and available memory. Interestingly, each node's IP address is used as a tiebreaker for the elections. The ranks are communicated in broadcast messages. The algorithm made a few assumptions; one of them was the bounded message delays and arbitrary message delivery order. This means that nodes operate in **asynchronous** rounds based on their internal timers where these rounds are not going to be identical in duration and offsets. Another Assumption is that nodes are not aware of any info about the neighboring nodes. Instead, it uses a volatile in-memory priority queue in the broadcasted messages that have the IDs and ranks. The algorithm tries to reduce the number of messages sent which increases the delay by only broadcasting messages from the nodes that consider themselves to be the best nodes in their broadcast domain. When the upper bound of the set message delay holds, the agreement and uniqueness of a leader are insured. There is a challenge which most agile environments face in terms of leader election which is the termination when two highly ranked nodes keep overtaking each other in the leading position. PALE solves this problem by increasing the ranks of non-jittering nodes whenever a failure is

detected in a leading node. Based on the increments the algorithm proves that the PALE algorithm can have a finite time termination given that only one node in the broadcast domain is up for the whole algorithm's execution [4].

Gossip Algorithms

Gossip-based leader election and consensus algorithms represent a fascinating category of distributed computing techniques inspired by the way information spreads organically, akin to social gossip. In these algorithms, nodes within a distributed network communicate with a limited set of randomly chosen neighbors, exchanging information about their local states, potential leaders, or proposed values. Over time, through these random interactions, a distributed consensus or a leader is elected. Gossip-based approaches are inherently decentralized, highly scalable, and resilient to network changes and failures. They find utility in various distributed systems, including peer-to-peer networks, distributed databases, and large-scale sensor networks, where they enable efficient leader election, data consistency, and fault detection. These algorithms are particularly well-suited for environments characterized by dynamic conditions and partial information, making them valuable tools for achieving consensus and leadership in complex, evolving networks [5].

Conclusion

In conclusion, Bully-based and Ring-based algorithms in their pure forms are highly impractical. Several limitations and weaknesses arise that are solved by using modified versions of those algorithms. Invitation algorithms solve the issue of several leaders electing themselves at the same time by forming groups, each having a single leader, where each leader sends invitations to other leaders to be part of the group. Furthermore, algorithms such as PALE decrease the number of messages sent by making the leaders broadcast a single message which is

a priority queue of node ranks and identities in the broadcasting domain. It also solves a problem where two leaders keep overtaking each other by incrementing lower-ranked non-jittery nodes. Finally, it has proven a finite time termination when there is at least one node that survived the whole program execution. An honorable mention goes to Gossip-based algorithms which have gained more research interest in the past period. These protocols boast a scalable, fault-tolerant, dynamic, and resilient election process. It has proven very appropriate for distributed systems due to its decentralized nature and ability to add nodes during the execution of the algorithm. As such, we will likely employ one or a mix of these algorithms, either PALE or Gossip, depending on the specific constraints that arise during implementation.

References

[1] G. F. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, "Chapter 15: Coordination and agreement," in Distributed systems: Concepts and design, Harlow, England: Addison-Wesley, 2012, pp. 641–646

[2] S. D. Stoller, "Leader election in Asynchronous Distributed Systems," *IEEE Transactions on Computers*, vol. 49, no. 3, pp. 283–284, 2000. doi:10.1109/12.841132

[3] Garcia-Molina, H. (1982). Elections in a distributed computing system. *IEEE transactions on Computers*, *31*(01), 48-59.

[4] B. Sidik, R. Puzis, P. Zilberman, and Y. Elovici, "Pale: Time bounded practical agile leader election," IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 2, pp. 470–485, 2020. doi:10.1109/tpds.2019.2933620

[5] P. Andrew and I. Antonios, "Asynchronous gossip-based Data Propagation Protocol," *2015 Fifth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, 2015. doi:10.1109/dictap.2015.7113162