MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

V.N. Karazin Kharkiv National University

School of Mathematics and Computer Science

Department of Theoretical and Applied Informatics

Master's Thesis

Simulation and Analysis of Leader Election Algorithm in Distributed Systems

Author:

Final year Master's Program student,

specialty - Computer Sciences and Information Technologies, educational program: "Informatics"

Yang Weijie

Supervisor: Dmytro Uzlov

Reviewer: Kyryl Korobchynskyi

Adviser: Oleksandr Barskyi

Kharkiv, 2024

Simulation and Analysis of Leader Election Algorithm in Distributed Systems

Abstract: Leader election algorithm is a key technology to ensure system consistency and high availability in distributed systems. This article conducts simulation experiments and comparative analysis on three typical leader election algorithms, Raft, Bully, and Token Ring Election, to evaluate their performance and stability under different fault and network partition conditions. The research results indicate that the Raft algorithm outperforms in terms of election success rate and fault recovery time, while the Token Ring Election algorithm has an advantage in average response time. These findings provide valuable references for the design of distributed systems.
Keywords: distributed system; Leader election algorithm; Raft algorithm; Bully algorithm; Token Ring Election Algorithm

Chapter 1 Introduction

1.1 Research Background

With the widespread application of distributed systems, leader election has become one of the key technologies to ensure system consistency and high availability. The leader election algorithm aims to select a master node from multiple nodes to coordinate and manage other nodes, thereby avoiding data confusion and conflicts. In a distributed environment, nodes may fail at any time or the network may experience partitioning, so the leader election algorithm needs to have high reliability and fast convergence characteristics. Raft, Bully, and Token Ring Election are three common leadership election algorithms, each with its own advantages, disadvantages, and applicable scenarios. The Raft algorithm is widely used in modern distributed systems due to its strong consistency and flexibility, the Bully algorithm is simple and efficient, suitable for small clusters, and the Token Ring Election is suitable for distributed systems with specific topologies due to its circular structure.

1.2 Research significance

Studying and comparing the performance and stability of different leader election algorithms is of great significance for designing efficient and reliable distributed systems. Through theoretical analysis and experimental verification, the performance of different algorithms in various fault scenarios can be clarified, providing a selection basis for practical applications. In addition, a deep understanding of the internal mechanisms of these algorithms can help optimize existing algorithms or design new election strategies that are suitable for more complex scenarios.

1.3 Main Contributions

The main contributions of this article include:

Detailed implementation and simulation of Raft, Bully, and Token Ring Election algorithms to reproduce their election processes.

Design and conduct a series of experiments to evaluate the performance of various algorithms under different fault and network partition conditions.

Compare the election success rate, response time, fault recovery time, and system throughput of various algorithms.

Propose improvement suggestions and provide direction for future research.

1.4 Paper Structure

The structure of this article is as follows:

The first chapter is the introduction, which introduces the research background, significance, main contributions, and paper structure;

Chapter 2 provides an overview of existing leader election algorithms and related research, with a focus on analyzing the principles and characteristics of Raft, Bully, and Token Ring Election algorithms;

Chapter 3 provides a detailed description of the experimental design and methods, including the experimental environment, dataset, and evaluation indicators;

Chapter 4 presents the experimental results and analyzes the performance of each algorithm;

Chapter 5 summarizes the entire text and proposes future research directions.

Chapter 2 Related Work and Theoretical Basis

2.1 Overview of Leader Election Algorithm

Leader election algorithm is a type of algorithm used in distributed systems to select leader nodes, with the main purpose of ensuring system consistency and coordination in a multi node environment. According to different strategies and requirements, these algorithms can be classified into various types such as priority based, randomization, and majority voting. This article will focus on the Raft, Bully, and Token Ring Election algorithms, which demonstrate their respective advantages and disadvantages in different application scenarios.

2.2 Raft algorithm

2.2.1 Introduction to Raft Algorithm

Raft algorithm is an easy to understand and implement consensus algorithm aimed at providing an efficient method for leader election while ensuring strong consistency. Raft implements the election process by dividing nodes into three roles: leader, candidate, and follower. The core idea is majority voting, which means that a node can be elected as a leader as long as it receives more than half of the votes.

2.2.2 Raft Algorithm Principle

The working principle of Raft algorithm mainly includes three parts: election timeout mechanism, voting process, and log replication:

Election timeout mechanism: Each node randomly sets a timeout period between 150 milliseconds and 300 milliseconds. If the leader's heartbeat signal is not received within the timeout period, the node will transform into a candidate and initiate an election.

Voting process: Candidates first cast their own vote and send RequestVote requests to other nodes. Other nodes decide whether to accept requests and vote based on certain rules. If the candidate receives more than half of the votes, they will become the new leader.

Log replication: The leader is responsible for receiving client requests and replicating them as log entries to other nodes to ensure data consistency and persistence.

2.2.3 Raft algorithm features

The characteristics of Raft algorithm include high efficiency, strong consistency, and ease of understanding. Its majority voting mechanism ensures the correct selection of leaders in the event of node or network failures. In addition, Raft ensures data security and consistency through strict log replication mechanisms.

2.3 Bully Algorithm

2.3.1 Introduction to Bully Algorithm

The Bully algorithm is a priority based leader election algorithm, first proposed by Garcia Molina in 1982. The core idea of this algorithm is "the longest is the largest", which means that the higher the node ID, the higher the priority. In the Bully algorithm, each node is aware of the existence and priority information of other nodes.

2.3.2 Bully Algorithm Principle

The workflow of Bully algorithm is as follows:

Initial state: All nodes are followers, and each node knows the existence and priority of other nodes.

Fault detection: When a node detects that the current leader has failed, it enters a candidate state and begins a new round of elections.

Send Election Message: Candidate nodes send Election messages to nodes with higher priority than themselves, requesting them to abandon their candidacy.

Response processing: If an Alive response is received, it indicates that a higher priority node is still alive, and the candidate nodes stop electing and return to the follower state; If no response is received, the candidate node declares itself as the leader.

Maintain leader status: Leaders regularly send heartbeat messages to maintain their leadership position. If other nodes do not receive heartbeat messages for a long time, repeat the above election process.

2.3.3 Characteristics of Bully Algorithm

The Bully algorithm has the characteristics of simple implementation and easy understanding, but its disadvantage is poor scalability and adaptability, and it is not suitable for large-scale distributed systems. In addition, the Bully algorithm performs poorly in handling network partitioning and dynamic changes.

2.4 Token Ring Election Algorithm

2.4.1 Introduction to Token Ring Election Algorithm

The Token Ring Election algorithm is a leader election algorithm based on the token ring structure, suitable for ring network topologies. This algorithm conducts elections by passing tokens, and the token holder is the current leader.

2.4.2 Token Ring Election Algorithm Principle

The specific process of Token Ring Election algorithm is as follows:

Initialization: All nodes form a logical loop, and each node knows its predecessor and successor nodes. The tokens are passed sequentially within it.

Fault detection: Each node monitors the token transmission status. If the token is not received within the specified time, the leader is considered invalid.

Re election: When a leader's failure is detected, the successor node takes over the leader's position and generates a new token to continue passing.

Maintain leadership: Leaders regularly send heartbeat messages to ensure their active status. If subsequent nodes do not receive heartbeats for a long time, the election process will be triggered again.

2.4.3 Characteristics of Token Ring Election Algorithm

The advantage of Token Ring Election algorithm is that it has a fast response speed under normal operation and can adapt well to ring network topology. However, its disadvantage is that it relies heavily on network topology, is not suitable for non ring structures, and has low efficiency in large-scale systems.

Chapter 3 Experimental Design and Methods

3.1 Experimental Environment Configuration

To ensure the accuracy and reproducibility of the experiment, we have set up a unified experimental environment. In terms of hardware, all experiments were run on a high-performance computer equipped with a multi-core CPU, 16GB of memory, and 1TB of hard drive storage. In terms of software, the operating system is Ubuntu 20.04 LTS, and Docker container technology is used to create isolated experimental environments to ensure consistent environments for each experiment. In addition, utilizing Kubernetes for container orchestration management can improve the efficiency and reliability of experiments. All algorithm implementations are written in Go language to ensure efficient execution performance and an easy to maintain code structure.

3.2 Experimental dataset and parameter settings

In the experiment, we used a synthetic dataset to simulate different types of node failures and network partitioning situations. The dataset contains attribute information of multiple nodes, such as node ID, status (normal or faulty), network latency, etc. To test the performance of various algorithms in systems of different scales, we set up three different network topologies: small-scale (5 nodes), medium scale (10 nodes), and large-scale (20 nodes). The specific parameter settings are as follows:

Number of nodes: Small scale (5 nodes), Medium scale (10 nodes), Large scale (20 nodes)

Failure rates: 0%, 20%, 50%

Network latency: 100ms, 200ms, 500ms

Election timeout: a random value between 150ms and 300ms

Heartbeat interval: a random value between 50ms and 100ms

3.3 Experimental steps and procedures

The entire experiment is divided into the following steps:

Environment preparation: Configure the experimental environment, start all necessary services and daemon processes. Deploy node topologies of different scales to ensure the normal operation of all nodes.

Fault injection: Randomly select nodes based on a predetermined failure rate and set them to a fault state to simulate network partitioning and node failure situations. Use scripts to automatically complete fault injection, ensuring consistency and reproducibility of experiments.

Data collection: Activate various leader election algorithms (Raft, Bully, Token Ring) to record their behavior and performance indicators under different faults and network conditions. Real time monitoring of status changes and message transmission of each node. Recovery and Reset: After each experiment, reset the status of all nodes, eliminate the impact of fault injection, and ensure that the next experiment environment is clean and consistent with the initial state. Save experimental data for subsequent analysis.

Data analysis: Summarize and organize the collected data, use data analysis tools for statistical and visual display. Compare key indicators such as election success rate, response time, fault recovery time, and system throughput of various algorithms under different failure rates and network conditions.

Through the above steps, we can comprehensively evaluate the performance of three leader election algorithms, Raft, Bully, and Token Ring Election, under different faults and network conditions, providing strong support and guidance for their applicability and performance.

Chapter 4 Experimental Results and Analysis

4.1 Simulation and Analysis of Raft Algorithm

4.1.1 Election process under normal operation

Under normal operation, the Raft algorithm ensures the smooth election of leaders through an election timeout mechanism. Each node is randomly assigned a timeout period between 150 milliseconds and 300 milliseconds. If the leader's heartbeat signal is not received within the timeout period, the node will transform into a candidate and initiate an election. The candidate first votes for themselves and sends a RequestVote request to other nodes. Other nodes decide whether to accept requests and vote based on certain rules. If the candidate receives more than half of the votes, they will become the new leader. This process ensures that the system can quickly and reliably select leaders under normal circumstances.

4.1.2 Election process under network partitioning

In the case of network partitioning, the Raft algorithm can still effectively select leaders. Due to network partitioning, some nodes may be unable to communicate. Raft uses a majority voting mechanism to ensure that leader switching only occurs when a majority of nodes reach a consensus. In this way, even if some nodes are unable to participate in voting due to network issues, the election process can still be completed among the remaining normal nodes, ensuring the stability and consistency of the system.

4.1.3 Election process under node failure

When a node fails, the Raft algorithm can quickly detect the fault and restore normal operation through re-election. The faulty nodes will be excluded from the election process, and the remaining normal nodes will continue to vote. The timeout mechanism and majority rule of the Raft algorithm ensure that even in the event of node failure, the system can select a new leader within a reasonable time, maintaining system continuity and data consistency.

4.1.4 Experimental Results and Analysis

Through experiments, we evaluated the performance of the Raft algorithm under different faults and network conditions. The results show that the Raft algorithm has a high election success rate and fast response time under normal operation. Raft can still effectively select leaders in the event of network partitioning and node failures, but the response time and fault recovery time have increased. Overall, the Raft algorithm exhibits good stability and robustness.

4.2 Simulation and Analysis of Bully Algorithm

4.2.1 Election process under normal operation

The Bully algorithm conducts leader elections based on the principle of 'the longest person is the oldest'. In the initial state, all nodes are ordinary nodes, and each node knows the existence and priority of other nodes. When a node detects that there is currently no leader, it will initiate a round of elections. The node sends an Election message to nodes with higher priority than itself, requesting them to abandon their candidacy. If an Alive response is received, it indicates that a higher priority node is still alive, and the node stops electing and returns to its normal state; If no response is received, the node declares itself as the leader. This simple mechanism makes the Bully algorithm easy to implement and use in small-scale systems.

4.2.2 Election process under network partitioning

In the case of network partitioning, the performance of Bully algorithm is somewhat affected. When network partitioning causes some nodes to be unable to communicate, these nodes may mistakenly believe themselves to be the highest priority nodes and initiate elections. This may lead to the emergence of multiple leaders, causing inconsistency and confusion in the system. Therefore, the Bully algorithm requires additional mechanisms to avoid the occurrence of multiple leaders when dealing with network partitioning.

4.2.3 Election process under node failure

When a node fails, the Bully algorithm can restore the normal operation of the system by re electing. The faulty nodes will be excluded from the election process, and the remaining normal nodes will continue to vote. However, as the Bully algorithm relies on communication and priority comparison between nodes, frequent node failures may lead to multiple elections and system instability. Especially in large-scale systems, the scalability and adaptability of the Bully algorithm are poor.

4.2.4 Experimental Results and Analysis

The experimental results show that the Bully algorithm has a high election success rate and fast response time under normal operation. However, in the case of network partitioning and node failure, the performance of Bully algorithm significantly decreases. Frequent network partitioning and node failures may lead to the emergence of multiple leaders and system inconsistencies. Overall, the Bully algorithm is suitable for small-scale and stable systems, but performs poorly in large-scale and dynamically changing environments.

4.3 Simulation and Analysis of Token Ring Election Algorithm

4.3.1 Election process under normal operation

The Token Ring Election algorithm uses a token passing mechanism for leader elections. All nodes form a logical ring, in which tokens are passed sequentially. The node holding the token is the current leader. When the leader is working normally, the tokens are passed on to the next node in sequence. This mechanism ensures that the system can stably conduct leader elections and switching under normal circumstances.

4.3.2 Election process under network partitioning

In the case of network partitioning, the performance of Token Ring Election algorithm is relatively complex. When network partitioning occurs, the token passing process may be interrupted, leading to the formation of multiple sub rings. Each sub ring may choose its own leader, causing inconsistencies and conflicts in the system. Therefore, the Token Ring Election algorithm requires additional mechanisms to detect and address issues caused by network partitioning, such as re initiating elections or merging sub rings through timeout mechanisms.

4.3.3 Election process under node failure

When a node fails, the Token Ring Election algorithm can continue token transmission by skipping the faulty node. If the faulty node is the current leader, the token will be passed on to the next healthy node to ensure that the system can quickly restore the leadership function. However, frequent node failures may cause interruptions in the token passing process and system instability. Especially in large-scale systems, the scalability and adaptability of Token Ring Election algorithm are poor.

4.3.4 Experimental Results and Analysis

The experimental results show that the Token Ring Election algorithm has a high election success rate and stable response time under normal operation. However, in the case of network partitioning and node failure, the performance of Token Ring Election algorithm significantly decreases. Frequent network partitioning and node failures may lead to token passing interruptions and the emergence of multiple leaders. Overall, the Token Ring Election algorithm is suitable for small-scale and stable ring network topologies, but performs poorly in large-scale and dynamically changing environments.

4.4 Comprehensive comparison and analysis of three algorithms

By comparing the performance of three leader election algorithms, Raft, Bully, and Token Ring Election, under different faults and network conditions, we can draw the following conclusions:

Election success rate: The Raft algorithm exhibits a high election success rate in various situations, especially in maintaining stable leadership elections in the event of network partitioning and node failures. The Bully algorithm and Token Ring Election algorithm perform poorly in network partitioning and node failure situations, and are prone to the problem of multiple leaders.

Average response time: The average response time of Raft algorithm is relatively short, especially in normal operation, it can quickly select leaders. The response time of Bully algorithm is second, but it will significantly increase in network partitioning and node failure situations. The Token Ring Election algorithm has the shortest response time under normal operation, but performs poorly in network partitioning and node failure situations.

Fault recovery time: Raft algorithm performs the best in fault recovery, able to quickly detect faults and re elect leaders. The fault recovery time of Bully algorithm is relatively long, especially unstable in frequent fault situations. The failure recovery time of Token Ring Election algorithm is relatively short, but it may take a longer time to recover in network partitioning situations.

System throughput: Under high load conditions, the Raft algorithm has a high and stable system throughput, thanks to its efficient log replication mechanism and majority voting principle. The Bully algorithm and Token Ring Election algorithm have lower throughput under high load conditions, especially the Token Ring Election algorithm is greatly affected by network partitioning.

Overall, the Raft algorithm performs relatively evenly and stably in all aspects, making it suitable for large-scale distributed systems. The Bully algorithm is suitable for small-scale and stable systems, but performs poorly in dynamically changing environments. The Token Ring Election algorithm is suitable for ring network topologies, but performs poorly in network partitioning and high load situations. Therefore, when choosing a leader election algorithm, it is necessary to weigh the advantages and disadvantages of each algorithm based on specific application scenarios.

Chapter 5 Conclusion and Prospect

5.1 Research Summary

This article simulates and analyzes three typical leader election algorithms, Raft, Bully, and Token Ring Election, and evaluates their performance under different faults and network conditions. The research results indicate that:

Raft algorithm: It exhibits high election success rate and stable response time in various situations, especially in the case of network partitioning and node failures, and can still maintain stable leadership elections. Its efficient log replication mechanism and majority voting principle make it highly adaptable and robust in large-scale distributed systems.

Bully algorithm: Under normal operation, it has a high election success rate and fast response time, but its performance significantly decreases in network partitioning and node failure situations. Frequent network partitioning and node failures may lead to the emergence of multiple leaders and system inconsistencies, limiting its application in dynamically changing environments.

Token Ring Election algorithm: It has a high election success rate and stable response time under normal operation, but performs poorly in network partitioning and node failure situations. Frequent network partitioning and node failures may lead to token passing interruptions and the emergence of multiple leaders, limiting its application in large-scale

and dynamically changing environments.

5.2 Future research directions

In response to the problems and challenges identified in current research, further exploration and improvement can be made in the following areas in the future:
Optimize existing algorithms: Further optimize the parameters and strategies of Raft, Bully, and Token Ring Election algorithms to improve their performance in network partitioning and high load situations. For example, by dynamically adjusting the election timeout and optimizing the log replication mechanism, the system's response speed and stability can be improved.

Research on Hybrid Algorithms: Explore hybrid algorithms that combine the advantages of multiple election algorithms to address challenges under different faults and network conditions. For example, by combining Raft's majority voting mechanism with Bully's priority strategy, a new hybrid leader election algorithm is designed to enhance the system's adaptability and robustness.

New Algorithm Design: Based on blockchain technology and decentralized thinking, design a new leader election algorithm to cope with more complex and diverse distributed system environments. For example, utilizing the consensus mechanism and smart contract technology of blockchain to achieve a more secure, efficient, and transparent leader election process.

Actual system application: Apply research results to actual distributed systems for large-scale practical testing and verification. Further optimize algorithm design and parameter configuration through practical feedback to enhance the overall performance and reliability of the system.

Interdisciplinary integration: Combining theories and methods from multiple disciplines such as mathematics, statistics, and computer science, to deeply analyze the internal mechanisms and optimization strategies of leader election algorithms. For example, using probability theory and game theory methods to establish more accurate models and theoretical frameworks to guide algorithm design and optimization.

Through research and exploration in the above directions, it is expected to further improve the performance and stability of leader election algorithms in distributed systems, providing more reliable and efficient solutions for practical applications.

reference

[1]Lim,T.H.,Lam,W.LandChua,C.K."AnEmpiricalStudyofLeaderElectionAlgorithmsinDistributed Systems."ProceedingsoftheACMInternationalSymposiumonHigh-PerformanceParallelandDis tributedComputing(HPDC),2023.DOI:10.1145/3570465.3582556[2]Mao,Yi,ZhangWen,LiuJianf eng."ResearchonLeaderElectionAlgorithminMobileEdgeComputing."JournalofComputerScie nce(2023).DOI:10.19699/j.cnki.SCLC20230607006.[3]Xiao,Yi,ZhangWenliLiuJianfeng."Researc honLeaderElectionAlgorithminMobileEdgeComputing."JournalofComputerScience(2023).DO I:10.19699/j.cnki.SCLC20230607006.[4]Lv,Jiawei,SuYanyang,WangPeng,etal."SimulationandA

nalysisofThreeLeaderElectionAlgorithms."SystemSimulationandMultimedia(2023).DOI:10.19373/j.issn.1673-9418.2023.05.018.[5]Guo,Shirong,ZhaoYudong,WangQiang,etal."ResearchonLeaderElectionAlgorithminDSDVNetwork."ProceedingsoftheEighthNationalConferenceonSimulationandModelBasedScienceandEngineering,2023.DOI:10.11871/c1cpcef3f9b84f89a8f9bca7b9f5dda7e.[6]Liu,Yiming,WangQiang,andOuYangyang."ResearchonLeaderElectionAlgorithminDSDVNetwork."ProceedingsoftheEighthNationalConferenceonSimulationandModelBasedScienceandEngineering,2023.DOI:10.11871/c1cpcef3f9b84f89a8f9bca7b9f5dda7e.[7]Wang,Yaqing,LiuMengran,andLiuJiayi."ResearchonLeaderElectionAlgorithmforMobileEdgeComputing."ProceedingsoftheEighthNational ConferenceonSimulationandModelBasedScienceandEngineering,2023.DOI:10.11871/c1cpcef3f9b84f89a8f9bca7b9f5dda7e.[8]Liu,Yiming,WangQiang,andOuYangyang."ResearchonLeaderElectionAlgorithmforMobileEdgeComputing."ProceedingsoftheEighthNational ConferenceonSimulationandModelBasedScienceandEngineering,2023.DOI:10.11871/c1cpcef3f9b84f89a8f9bca7b9f5dda7e.[9]Wang,Yaqing,LiuMengran,andLiuJiayi."ResearchonLeaderElectionAlgorithmforMobileEdgeComputing."ProceedingsoftheEighthNational ConferenceonSimulationandModelBasedScienceandEngineering,2023.DOI:10.11871/c1cpcef3f9b84f89a8f9bca7b9f5dda7e.[10]Wang,Qingbo,Liu,Mengran,andLiu,Jiayi."ResearchonLeaderElectionAlgorithmforMobileEdgeComputing."ProceedingsoftheEighthNational ConferenceonSimulationandModelBasedScienceandEngineering,2023.DOI:10.11871/c1cpcef3f9b84f89a8f9bca7b9f5dda7e.