MALAYSIAN JOURNAL OF SCIENCE
MJS
ISSN 1394-3065

Regular Issue

Malaysian Journal of Science 41 (3): 38-43 (October 2022) https://mjs.um.edu.my

A WAITING TIME-BASED BULLY ALGORITHM FOR LEADER NODE SELECTION IN DISTRIBUTED SYSTEMS

Md. Navid Bin Anwar[1a], Afroza Nahar[2a]*, Nashid Kamal Md.[3a], Mehedi Hasan Shuvo[4a]

Abstract: In distributed systems, a single node (referred to as a leader) coordinates all other nodes to ensure synchronization. If this node fails, another node in the system must adopt the role of leader. The classic bully algorithm suffers from some significant drawbacks, such as excessive message passing, a redundant number of election calls, and uncertainties over message delivery. The enhanced bully algorithm is one of the most recent improvements of this algorithm. However, this algorithm performs poorly in average- and worst-case scenarios. In this paper, a novel waiting time-based algorithm is proposed to improve the enhanced bully algorithm for electing a new leader during such critical scenarios. In this algorithm, if a single or multiple number of nodes discover that the leader has failed, it does not broadcast instantly. Rather, it waits for a certain period, and this waiting time is assigned to the nodes according to their load. After the timeout, the node sends its message and starts the election process. Moreover, it restricts nodes from unnecessary message passing and stops any redundant election calls. Accordingly, this algorithm detects the failure of the leader node more precisely and elects a new leader more quickly.

Keywords: *Distributed systems, bully election algorithm, electing coordinator, message passing*

1. Introduction

A distributed system is an accretion of isolated computers that engage simultaneously through a system to accomplish a complex task that is coordinated by message passing (Beaulah et al., 2013; Sathesh, 2015). To arrange the various errands in a distributed system, a leader is essential for synchronizing the whole system whenever necessary. To select the leader (or leader), several algorithms have been proposed, including distinctive, ring topology, bully election, Franklin's, Chang and Robert's, time slice, and variable speeds (Amit & Animesh, 2016; Balmukund et al., 2014; Rahman & Nahar, 2009). However, all these algorithms have their own shortcomings, including time complexity, message passing, redundancy, and large network traffic. This paper presents a reformation of the enhanced version of the bully algorithm by introducing the time allocation concept to the nodes. The remainder of the paper is arranged as follows: Section 2 contains a brief literature review, Section 3 presents a description of the proposed algorithm with legitimate examples, Section 4 reports the comparative results and outcomes, and Section 5 indicates the inferences of the present study.

Authors information:
[a]Department of Computer Science, American International University – Bangladesh (AIUB), Dhaka 1229, BANGLADESH. E-mail: navid.anwar@aiub.edu[1]; afroza@aiub.edu[2]; nashidkamaljitu7@gmail.com[3]; shuvo.rpm@gmail.com[4]
*Corresponding Author: afroza@aiub.edu

2. Methodology

In this section, the four major algorithms for electing the leader node in distributed systems are discussed.

2.1 Bully Algorithm by Garcia-Molina

In 1982, Garcia-Molina first introduced the bully algorithm (Garcia, 1982), which dynamically chooses a coordinator (or leader) by utilizing the process identification (ID) number. This algorithm is based on the following essential hypotheses:

- It is a synchronous method that utilizes a timeout instrument to monitor leader disappointment location, and each process has an exceptional number to allow them to be recognized.
- Each node acknowledges the node ID among all other nodes. No node knows which forms are currently up and which forms are down.
- The node with the highest node ID is chosen as a leader and is in accord with other active nodes.
- A failed node can rejoin the process after recuperation.

Whenever the coordinator node is down, an election process for selecting a new leader starts, and the node with the highest ID becomes the new leader. All active nodes receive this message, which entails extensive message

Received: December 4, 2020
Accepted: February 21, 2022
Published: October 31, 2022

DOI:https//doi.org/10.22452/mjs.vol41no3.5
Malaysian Journal of Science 41(3): 38-43 (October 2022)

passing and creates heavy network traffic (Garcia, 1982; Mamun et al., 2017).

From Figure 1, the leader election process can be described as follows:

a) As Node 2 detects that the leader node is down, it sends election messages to the higher Nodes 3 and 4.
b) In response, Nodes 3 and 4 send an OK message.
c) Nodes 3 and 4 send an election message to Node 5.
d) Node 4 sends an OK, although it will not receive any message from Node 5.
e) Node 4 will elect itself as leader and broadcast a leader message to each node in the network.

This algorithm has the following drawbacks:

- The most significant number of messages during the election is O (n2), regardless of how it is arranged. Whenever a node sees the leader node is down, another election is held. Subsequently, multiple elections could happen within this method at the same time, forcing substantial system traffic that could result in the system being overwhelmed.
- If the leader is running singularly or the connection between the process and the coordinator is damaged, some processes could fail to identify the leader and start an election, resulting in redundant elections.
- This algorithm does not guarantee message delivery. Consequently, multiple nodes could assign themselves as the leader simultaneously.

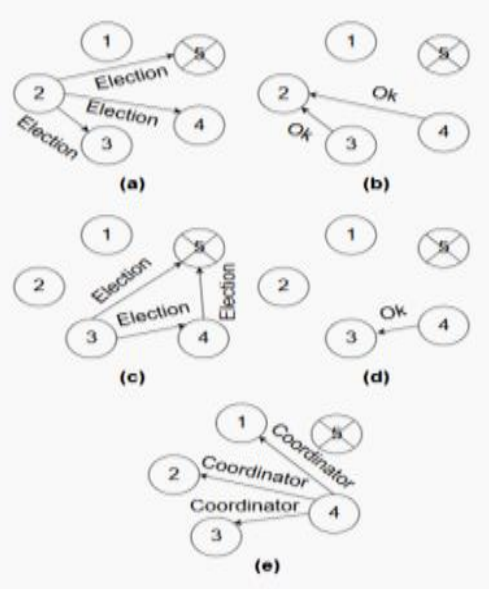

Figure 1. Leader election in the bully algorithm.

2.2 Modified Bully Algorithm by Quazi Ehsanul Kabir Mamun

The basic assumptions of this algorithm are rooted in the Bully algorithm, whereby the node with the highest ID number is selected as the leader (Mamun et al., 2017). However, this algorithm proposes a resolution in cases where any node (or multiple nodes) detects that the leader node does not respond.

- If any node identifies that the leader is unavailable or has not responded, an election is announced by sending a message to the nodes with higher IDs.
- In response, the node with the highest ID transmits an OK message and the node elects the node with the highest ID node after receiving the responses.
- After being elected, the highest node sends a leader message and broadcasts itself as the leader to all other existing nodes.

From Figure 2, the election process can be described as follows:

a) Node 2 detects that leader Node 5 is down.
b) An election message is then sent by Node 2 to the highest ID nodes (Nodes 3 and 4).
c) Upon receiving the election message, Nodes 3 and 4 send an OK message to Node 2.
d) Node 2 elects the highest ID node (i.e., Node 4), which then 4 broadcasts itself as the new leader to all other existing nodes.

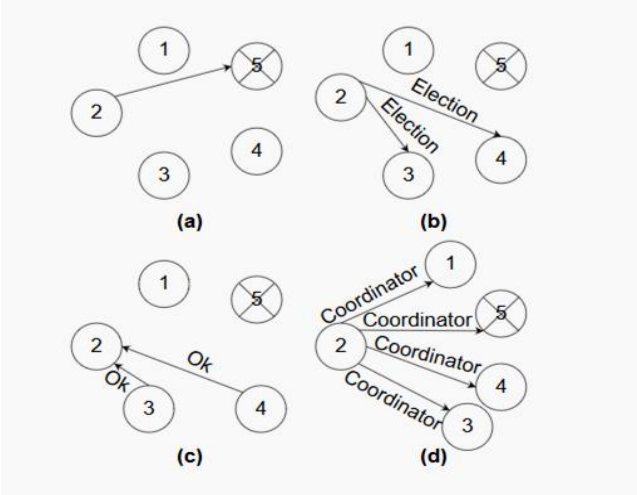

Figure 2. Leader election by Quazi Ehsanul Kabir Mamun (Mamun et al., 2017)

DOI:https//doi.org/10.22452/mjs.vol41no3.5
Malaysian Journal of Science 41(3): 38-43 (October 2022)

Although the modified bully algorithm reduces the total time of message passing and the complexity compared to the original bully algorithm, it displays some inadequacies:

- In some cases, the total number of messages passing increases because a node can receive more than one election message from its lower ID nodes, which increases network traffic.
- On recovery, the overall degree of message passing is also increased, causing a significant amount of network traffic.

2.3 Modified Bully Algorithm by Kordafshari et al.

Kordafshari et al. (2005) introduced a new algorithm in which the node containing the highest ID acts as the coordinator node. The author's also attempted to identify the ensuing circumstances if any node (or nodes) discovered that the leader node was down.

- When any node detects that the leader node is not responding, it instantaneously announces the start of an election and sends messages to all other existing nodes with higher IDs.
- In response, the higher ID nodes return an OK message, and the node selects the highest ID node among them after receiving the responses.
- After electing the highest node, it sends a GRANT message to the selected highest ID node.
- Upon receiving the GRANT message, the highest ID node then sends a new message to all other existing nodes as the new leader.
- If, after sending the message, the node does not receive either a response or an OK message from other existing nodes, it broadcasts itself as the new leader node and sends confirmation to all other active nodes.
- The algorithm will run again on recovery with the highest priority ID node.

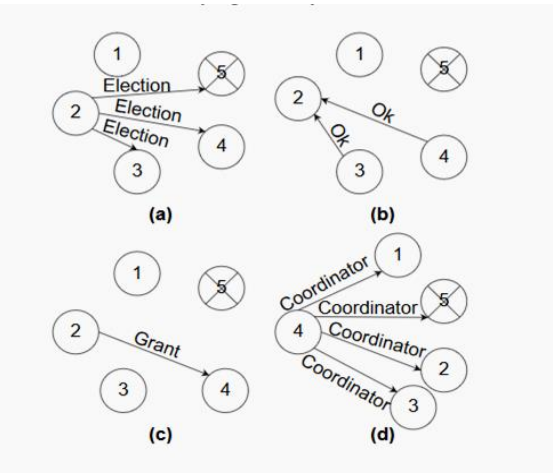

Figure 3. Leader Election by M.S. Kordafshari et al. (2005)

From Figure 3, the election process can be understood as follows:

a) Node 2 detects that the leader node (Node 5) is down. Therefore, it initiates an election and sends an election message to Nodes 3, 4, and 5.
b) An OK message is sent to all nodes as a response.
c) Upon receiving the replies, Node 2 sends a GRANT message to Node 4, as it contains the highest node ID.
d) When Node 4 receives the GRANT message, a leader message is broadcast to all other existing nodes.

This algorithm also suffers from the following deficiencies:

- If a node becomes down either while sending the election message or after receiving the priority number from higher ID nodes, the nodes will wait for a time of 3D for the leader message, where D is the average propagation delay. During this period, the nodes will recommence the algorithm if they do not receive any leader message, which is redundant (Kordafshari et al., 2005).
- The higher the number of nodes, the distinct precedent of the modified bully algorithm will remain at that moment in the system, which will cause repetitive election.
- The total amount of message passing, and network traffic will increase as a result of every redundant election.

2.3 Enhanced Bully Algorithm by Minhaj Khan, Neha Agarwal, and Jeeshan Ahmad Khan

This algorithm introduced the electing coordinator concept, which reduces the amount of unnecessary message passing and redundant election calls. When nodes detect that the coordinator has crashed, any of the following can happen:

- Only one node detects the crash.
- More than two nodes detect the crash.
- Every node detects that the leader is crashed.
- The node with the second-highest ID detects the crash.

This algorithm proposes that an election message should be sent by other nodes to the second-highest ID node. This node will then check whether the leader node is active. If the coordinator fails to respond again, then the second-highest ID node will elect itself as the new leader and send a notification to all other active nodes through a leader message (Minhaj et al., 2017). Three different variables are used to store the leader: the node ID, the ID of the leader that has crashed recently and the ID of the new leader (to reduce the total number of messages passing during the election).

DOI:https//doi.org/10.22452/mjs.vol41no3.5
Malaysian Journal of Science 41(3): 38-43 (October 2022)

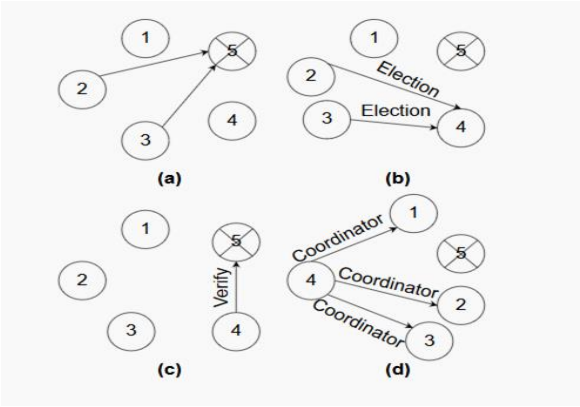

Figure 4. Leader election by Minhaj et al. (2017)

From Figure 4, the election process is clarified as follows:

a) The leader node (Node 5) has crashed, which is detected by Nodes 2 and 3.

b) Nodes 2 and 3 send an election message to the second higher node (Node 4).

c) After receiving the election message, Node 4 checks whether the leader has crashed.

d) When Node 4 detects that the leader node is down, it sends a message with the new leader ID and the recently crashed leader ID to all active nodes.

This algorithm has the following major drawbacks:

- When all nodes (n) detect at a certain time that the leader has crashed, this results in the system having a total number of messages of $3 \times (n - 2) + 1$ and a time complexity of O (n).
- If multiple nodes (n) detect that the leader is down, the number of messages passing for detecting the new leader will be $(2 \times x) + 1 + (n - 2)$ with a time complexity of O (n).

3. Results and Discussions

In this section, a waiting time-based bully algorithm is introduced based on the enhanced bully algorithm (Kordafshari, 2005). According to the enhanced bully algorithm, a large number of messages pass when multiple nodes detect that the leader is down. Therefore, in order to reduce the number of messages passing and improve efficiency, a waiting time is introduced in the proposed algorithm. It implies that when a node notices that the leader is down, it does not instantly broadcast. The node instead waits for a certain amount of time before sending its message. Depending on their load, nodes are given a waiting time. If the load is small, then the waiting time will be shorter, or vice versa. Therefore, during average- or worst-case scenarios, when multiple (or all) nodes detect that the leader node is down or crashed, only the node with the smallest load (shortest waiting time) will send election message to the second-highest ID node. As the waiting time of this node will be the shortest, the timeout will occur first, and the election process will begin soon. As explained previously, the second-highest process ID node will check the leader again by sending a message. If it also detects that the leader has failed, then it will declare itself the new leader by sending a leader message to all other active nodes. This process will prevent multiple nodes from sending election messages at the same time. Furthermore, it is very unlikely for multiple nodes to have the same waiting time, as this time depends on the load. Again, if the second-highest ID node is unavailable or is lost and cannot respond within the timeout, then the election requesting node will send the election request to the third highest process ID node. This process will continue after each timeout.

3.1 Algorithm

In the present algorithm, when N number of nodes detects that the leader node is down, the nodes respond according to their predefined waiting time. This waiting time is calculated and assigned to the nodes according to the following formula:

WaitingTime= propagation delay + verification time,
where the propagation delay describes the time, a packet takes to reach its destination from the source, the verification time defines the duration to check whether the leader is down or not, and w is a weight value defining the load on the node.

If there are five nodes in a distributed system and when the leader node is down, then the waiting time for the remaining four nodes can be assigned according to the following steps:

WaitingTime for Node 4 (highest process ID) = 0 ms (with minimal load)
WaitingTime for Node 3 (2nd highest process ID) = (2 × propagation delay + verification time)
WaitingTime for Node 2 (3rd highest process ID) = (2 × propagation delay + verification time) × 2
WaitingTime for Node 1 (3rd highest process ID) = (2 × propagation delay + verification time) × 3

After the waiting time has elapsed, the corresponding node sends an election message to the second-highest process ID node, after which the second-highest node checks the leader's status. If the leader node is down, it will broadcast a leader message to all active nodes, and the remaining nodes will not send any election messages; otherwise, it will discard the election message.

DOI:https//doi.org/10.22452/mjs.vol41no3.5
Malaysian Journal of Science 41(3): 38-43 (October 2022)

1. Initially the coordinator will assign the WaitingTime to each of the node according to the proposed algorithm.
// Node X_1, X_2 ..... X_n detect that the coordinator is down
2. A single node, having smallest WaitingTime, will send election message to the second highest process ID node, when waiting time is finished.
3. Second highest process ID node will CheckNode (scp_id is down or not)
4. If (scp_id node down)
5. scp_id = ncp_id
6. Broadcast coordinator message (ncp_id, rcp_id)
7. Cancel remaining node from sending election

Figure 5. Pseudo code for the scenario when all nodes detect that the coordinator node is down.

3.3 Example

The election process according to the proposed algorithm is shown in Figure. 6. The steps can be described as follows:

a) Leader/coordinator Node 5 is down.
b) Nodes 1, 2, and 3 detect leader failure at the same time.
c) Based on the waiting time span, Node 3 sends an election message to the second-highest process ID node (Node 4). In this case, Nodes 1 and 2 will not send an election message to Node 4, as their waiting time is greater.
d) Node 4 checks the coordinator again and finds that the leader is down.
e) Finally, Node 4 sends a leader message to all active nodes and broadcasts itself as the new leader node. Upon receiving the message, all other nodes (active) update their table and store Node 4 as the new leader.

According to the average- and worst-case scenarios of the enhanced bully algorithm (Kordafshari, 2005), when multiple nodes (p) or all nodes (n) detect leader failure, the total number of messages passing will be 2 × p + 1 + (n – 2) and 3 × (n – 2) + 1 (Kordafshari, 2005). However, in the proposed algorithm, when a multiple number (p) of nodes or all nodes (n) detect that the leader has failed, then the total number of messages passing between the nodes for electing the leader will be 2 × (n – 2) + p + 1 and 2 × (n – 2) + 1 + 1. However, the total number of messages passing within the nodes for electing the leader could be 3 × (n – 2) + 1, which is similar to Kordafshari (2005) when a timeout occurs for every election requesting node. It is very unlikely that this would happen in normal conditions and could only occur if the receiver (second-highest process ID node) was also down.

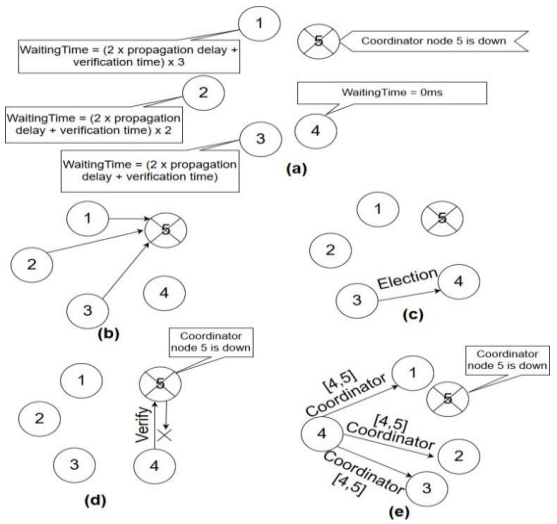

Figure 6. Steps to select a new leader node when multiple numbers of nodes find leader as down.

4. Comparison with Previous Algorithms

The performance of the proposed algorithm is compared with the bully algorithm, modified bully algorithm, and enhanced bully algorithm by counting the total number of messages passing during the scenario when multiple numbers of nodes detect the leader node is down. Table 1 indicates that the proposed waiting time-based bully algorithm exchanges the least number of messages. Hence, it detects the leader node failure earlier and can call the election quickly compared to other schemes. As a result, the efficiency of the proposed algorithm is better than other algorithms. This is due to the fact that the proposed algorithm restricts the nodes by assigning WaitingTime to send unnecessary redundant messages for verifying the leader node's failure while only one node, with a higher processing ID, will do this task first.

A network's propagation delay and verification time depend on the system conFigureuration. For this experiment, the following system conFigureuration was used: Intel i5-4210U CPU @ 1.70GHz (4 CPUs) ~2.4 GHz, 4 GB RAM, Windows 10 OS, and the network conFigureuration was Wireless LAN IEEE 802.11b/g/n with 3 Mbps bandwidth. The average network response time was 107.7 ms.

Table 1. Performance comparison of proposed algorithm with other algorithms

Total nodes in a network	Leader node Election Algorithms (number of messages)			
	Bully Algorithm	Modified Bully Algorithm	Enhanced Bully Algorithm	Proposed Algorithm
5	24	14	10	8
10	99	29	25	18
25	624	74	70	48
100	9999	299	295	198
150	22499	449	445	298

DOI:https//doi.org/10.22452/mjs.vol41no3.5
Malaysian Journal of Science 41(3): 38-43 (October 2022)

5. Conclusion

A novel waiting time-based bully algorithm to elect the leader node in a distributed system was proposed in this article. The algorithm solves the limitations of enhanced bully algorithms and improves the performance of the algorithm in terms of message passing. The waiting time method in the proposed algorithm restricted the nodes from unnecessary message passing, stopping redundant election calls. Therefore, the proposed algorithm helps to detect leader node failure more precisely and elect the new leader more swiftly.

6. Acknowledgement

Authors acknowledge American International University-Bangladesh (AIUB) for supporting this research.

7. References

Amit B., Animesh D. (2016). A Timer Based Leader Election Algorithm, IEEE Conference.

Balmukund M., Ninni S. & Ravideep S. (2014). Master-Slave Group Based Model For Co-Ordinator Selection, An Improvement of Bully Algorithm, International Conference on Parallel, Distributed and Grid Computing, Solan, 457-460,

Beaulah SP, Thriveni J, Venugopal, K., & Patnaik LM (2013). An improved leader election algorithm for distributed systems, International Journal of Next-Generation Networks (IJNGN). 5(1): 21-29.

Garcia-Molina H, (1982). Elections in distributed computing system. IEEE Transaction on Computer C-31:48-59.

Kordafshari MS, Gholipour M., Jahanshahi M. & Haghighat AT. (2005) Modified Bully Election Algorithm in Distributed Systems. WSEAS International Conference on Computers.

Kordafshari, MS. M. Gholipour, Jahanshahi M. & Haghighat AT. (2005). Two novel algorithms for electing coordinator in distributed systems based on bully algorithm, the fourth WSEAS International Conference on Software Engineering, Parallel and Distributed Systems.

Mamun QH, Masum SH & Mustfa MA. (2017). Modified bully algorithm for electing coordinator in distributed systems, in Proc. 3rd WSEAS International Conference on Software Engineering, Parallel and Distributed Systems :22-28.

Minhaj K., Neha Agarwal & Jeeshan AK. (2017). An Enhanced Bully Algorithm for Electing a Coordinator in Distributed Systems. International Journal on Recent and Innovation Trends in Computing and Communication 5(5): 1092-1097.

Rahman M. & Nahar A. (2009). Modified Bully Algorithm using Election Commission. MASAUM Journal of Computing 1(3): 439-446,

Sathesh BM. (2015). Optimized Bully Algorithm. International. Journal of Computer Application. 121(18): 0975 – 8887.

DOI:https//doi.org/10.22452/mjs.vol41no3.5
Malaysian Journal of Science 41(3): 38-43 (October 2022)