International Journal of Computing Sciences Research (ISSN print: 2546-0552; ISSN online: 2546-115X)
Vol. 8, pp. 3217-3234
doi: 10.25147/ijcsr.2017.001.1.215
https://stepacademic.net

Short Paper*

Modification of Traditional Bully Algorithm using Priority Queuing Technique Applied in CPU Memory Allocation

Richelle Rose R. Alcaide
Computer Science Department, Pamantasan ng Lungsod ng Maynila, Philippines
rrralcaide2020@plm.edu.ph

Saimon C. Rumol
Computer Science Department, Pamantasan ng Lungsod ng Maynila, Philippines
scrumol2020@plm.edu.ph
(corresponding author)

Vivien A. Agustin
Computer Science Department, Pamantasan ng Lungsod ng Maynila, Philippines
vaagustin@plm.edu.ph

Mark Christopher R. Blanco
Computer Science Department, Pamantasan ng Lungsod ng Maynila, Philippines
mcrblanco@plm.edu.ph

Jonathan C. Morano
Computer Science Department, Pamantasan ng Lungsod ng Maynila, Philippines
jcmorano@plm.edu.ph

Leisyl M. Mahusay
Computer Science Department, Pamantasan ng Lungsod ng Maynila, Philippines
lmocampo@plm.edu.ph

Jamillah S. Guialil
Computer Science Department, Pamantasan ng Lungsod ng Maynila, Philippines
jsguialil@plm.edu.ph

Date received: April 29, 2024
Date received in revised form: June 15, 2024; June 28, 2024
Date accepted: June 30, 2024

Recommended citation:

Alcaide, R. R. R., Rumol, S. C., Agustin, V. A., Blanco, M. C. R., Morano, J. C., Mahusay, L. M., & Guialil, J. S. (2024). Modification of traditional bully algorithm using priority queuing technique applied in CPU memory allocation. *International Journal of Computing Sciences Research*, *8*, 3217-3234. https://doi.org/10.25147/ijcsr.2017.001.1.215

**Special Issue on International Research Conference on Computer Engineering and Technology Education (IRCCETE). Guest Associate Editors: Dr. Roben A. Juanatas (National University-Manila) and Dr. Nelson C. Rodelas (University of East).*

Abstract

Purpose – This study aims to modify the Bully Algorithm, a leader-election algorithm, by introducing Priority Queuing to optimize its steps, and evaluate its efficiency based on message count, election time, and instances of communicating with inactive nodes.

Method – Priority Queuing will organize active nodes in descending order based on their active status, with the election message sent only to the highest-ranked node in the queue. The study intends to compare the performances of three variations of the Bully Algorithm (the Traditional Bully Algorithm, the latest enhancement, and the proposed modification) using a simulator that ensures the algorithms share the same data set.

Result – The findings show that the proposed modification trumps the latest enhancement only during an increased presence of inactive nodes in the distributed system. In return, the newest enhancement trumps the proposed modification when there is little to no presence of inactive nodes.

Conclusion – The proposed modification has successfully reduced the time consumed, communication costs, and the instance of data transmission with failed nodes compared to the traditional method. However, it is not completely better or worse than the latest enhancement.

Recommendation – While conducting the findings for the study, the researchers recommend looking into achieving the same objectives while also considering the reactivation of nodes during the election process. The researchers also recommend fine-tuning the timeout interval and exploring other strategies for enabling multiple nodes to initiate the election process.

Research Implications – This improved algorithm can efficiently coordinate resource management in cloud computing environments, facilitate data replication, and coordinate consensus mechanisms in blockchain networks. These enhancements

optimize coordination, fault tolerance, and scalability in distributed systems, ultimately improving performance and user experience.

Practical Implication – The findings of this study have several implications, one of which is enhanced failure tolerance in distributed systems. Moreover, the waiting time-based bully algorithm is an attractive solution for modern distributed systems due to its ability to quickly adapt to network dynamic changes without significant performance degradation.

Keywords – distributed system, leader-election, algorithm, bully algorithm, priority queuing

INTRODUCTION

A distributed system is made up of a variety of distinct procedures that are physically separated yet share messages (Lamport, 2019). A classic difficulty in distributed system applications is leader election. Some of the most widely known leader election algorithms are the Ring Algorithm and the Bully Algorithm. The Bully Algorithm is a straightforward procedure in which every active process is listed in the system and the one with the highest ID serves as the coordinator (Shenoy, 2022).

However, this algorithm is costly in terms of communication costs. Numan et al. (2018) stated that the algorithm has space to improve its performance by reducing the number of messages during election procedures. Similarly, Azzam et al. (2020) observed an issue in the Bully Algorithm regarding the time consumption of the Bully Algorithm and conducted a study using the leader-collaboration method to resolve this issue. Additionally, the algorithm requires that the failed node keeps receiving election messages from nodes with lower IDs.

To resolve these issues, the study aims to use the Priority Queuing technique to reduce the algorithm's communication cost by having only a single node initiate an election in any given situation. It also aims to reduce the time consumption of the algorithm to assign a single node to send and receive messages at any given time, and to minimize the instances of sending data to a failed node.

Improving the traditional Bully Algorithm contributes to the field by addressing the mentioned key challenges. Enhancing its communication cost enables the algorithm to scale to larger systems without sacrificing performance or efficiency. Modern applications demand real-time responsiveness, and enhancing the traditional Bully Algorithm's time consumption in leader election will be more relevant in time-sensitive applications. In crucial environments where node failures are common, ensuring uninterrupted operation and system reliability is a vital enhancement in leader election.

LITERATURE REVIEW

Traditional Bully Algorithm

Garcia Molina presented the Bully Algorithm in 1982. The node with the greatest ID serves as the coordinator in this method (Kanwal et al., 2021). The bully algorithm is more significant than the ring-based algorithm due to its fault tolerance, which the ring-based algorithm lacks (Wan, 2023). There are three different message types in the Bully algorithm: (1) Election, which initiates an election; (2) Answer, which acknowledges a message; and (3) Coordinator, which identifies a leader (Guo et al., 2020). The flowchart of the traditional Bully Algorithm is shown in Figure 1.

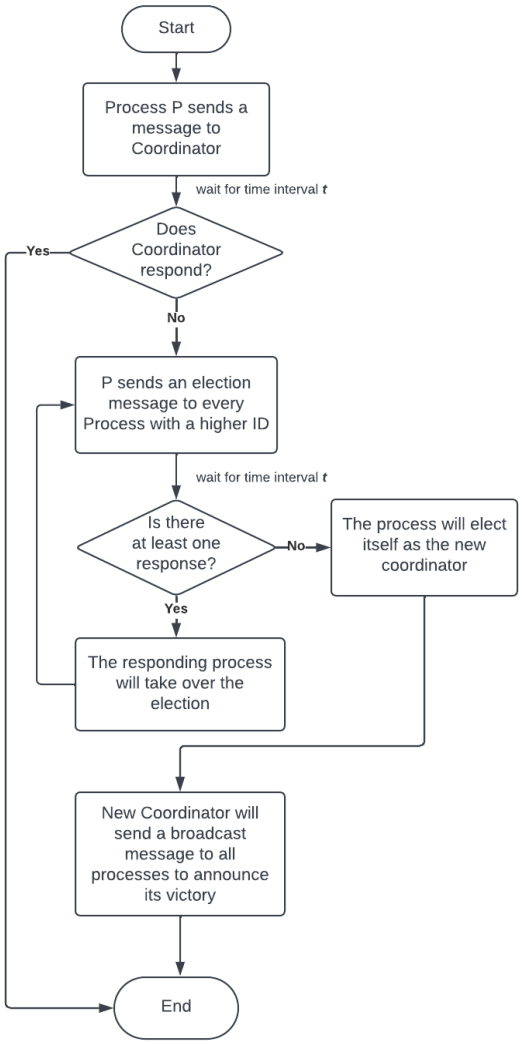


Figure 1. Flowchart of the Traditional Bully Algorithm

In a study by Madisetti and Panda (2021), the best-case scenario of this algorithm is when the second-highest ID is the one that notices the failure of the coordinator and immediately elects itself as the new Coordinator. In this scenario, the total number of

sent messages will be N-2 if there are N nodes since the kind of message that is sent is only to broadcast the victory of the self-elected node. Consecutively, the worst-case scenario would be when the process with the lowest ID detected the failure of the coordinator. In this instance, the total number of messages to be exchanged will be $N(N+1)^2$ or $O(n^2)$. The Traditional Bully Algorithm has the nodes, election time, and crashed leader node as input of the process and output is the new leader node. It assumes that the message delivery between processes is reliable and that each process is aware of its ID.

Waiting Time-Based Bully Algorithm

The Waiting-Time Based Bully Algorithm is the latest enhancement of the Bully Algorithm that was published in October 2022 by Anwar et al. and that suggests when a node detects that the leader is down, it does not broadcast the assumption immediately. Instead, the node waits a predetermined period before transmitting its message. When numerous (or all) nodes notice that the leader node is offline or crashed, only the node with the smallest load (shortest waiting time) will send an election message to the node with the second-highest ID. The following is the pseudocode of this enhancement:

1. Initially, the coordinator will assign the WaitingTime to each of the nodes according to the proposed algorithm.
2. e.g., Node X1, X2, X3, …, Xn detect the coordinator is down.
3. A single node, having the shortest WaitingTime, will send an election message to the second highest processID node, when waiting time is finished.
4. The second highest processID node will CheckNode (scp_id is down or not)
5. If(scp_id is down):
    a. scp_id = ncp_id
    b. Broadcast coordinator message (ncp_id, rcp_id)
    c. Cancel the remaining node from sending the election.

When a multiple number (p) of nodes or all nodes (n) discover that the leader has failed, the total amount of messages flowing between the nodes for electing the leader will be 2(n-2) + p + 1 and 2(n-2) + 1 + 1. This is extremely rare to occur under normal circumstances and could only occur if the receiver (second-highest process ID node) was also down. The performance of the Waiting-Time Based Bully Algorithm is observed following the traditional bully algorithm, the modified bully algorithm, and other enhancements of bully algorithms by comparing the total number of messages sent wherein multiple nodes have detected the failed coordinator.

As it is the latest enhancement, this version of the Bully Algorithm is the most efficient. However, the weakness of this version lies in an increased number of inactive nodes as election messages could be sent to more nodes if the next nodes with higher IDs are also inactive. More election messages could result in more time consumed and more instances of attempts at communicating with failed nodes.

Other Variations of Bully Algorithm

Modified Sandipan-Basu Bully Algorithm

Surolia and Bundele (2020) modified Sandipan-Basu's enhanced Bully Algorithm. In this version of the Bully Algorithm, the process table only examines node IDs that are greater than the node. As a result, the process table would require less storage space in each node's memory. In addition, the table overhead would be reduced. The flowchart for this algorithm is shown in the figure below:

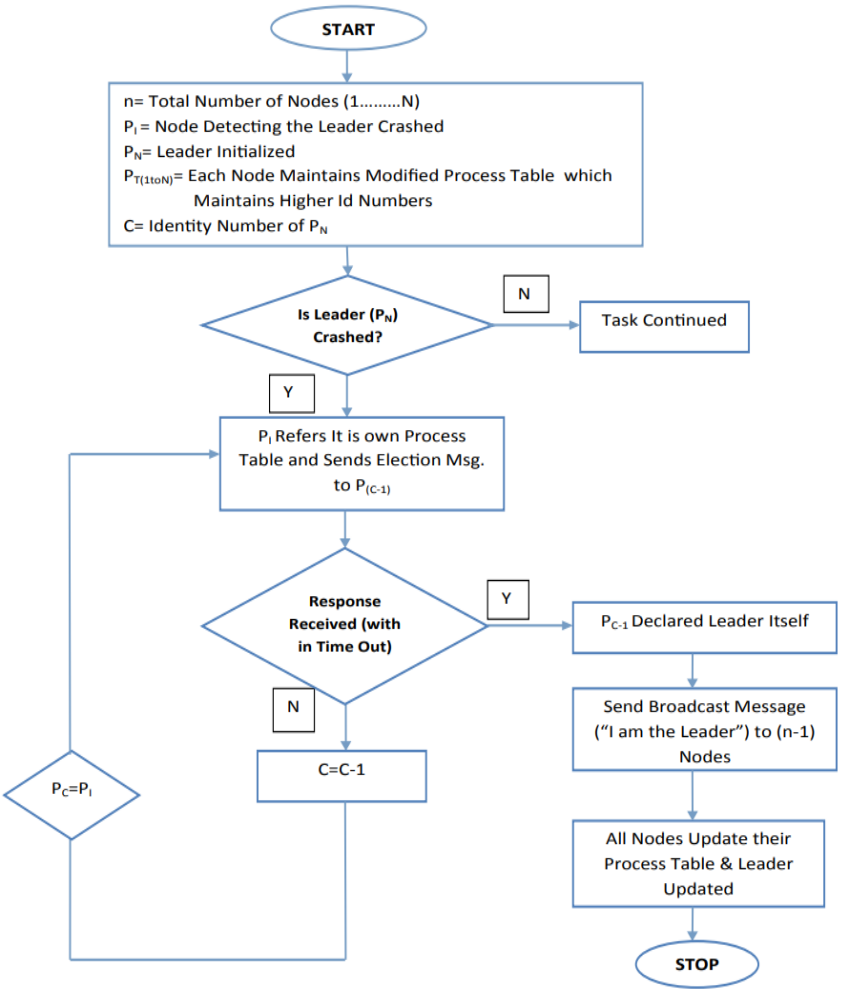


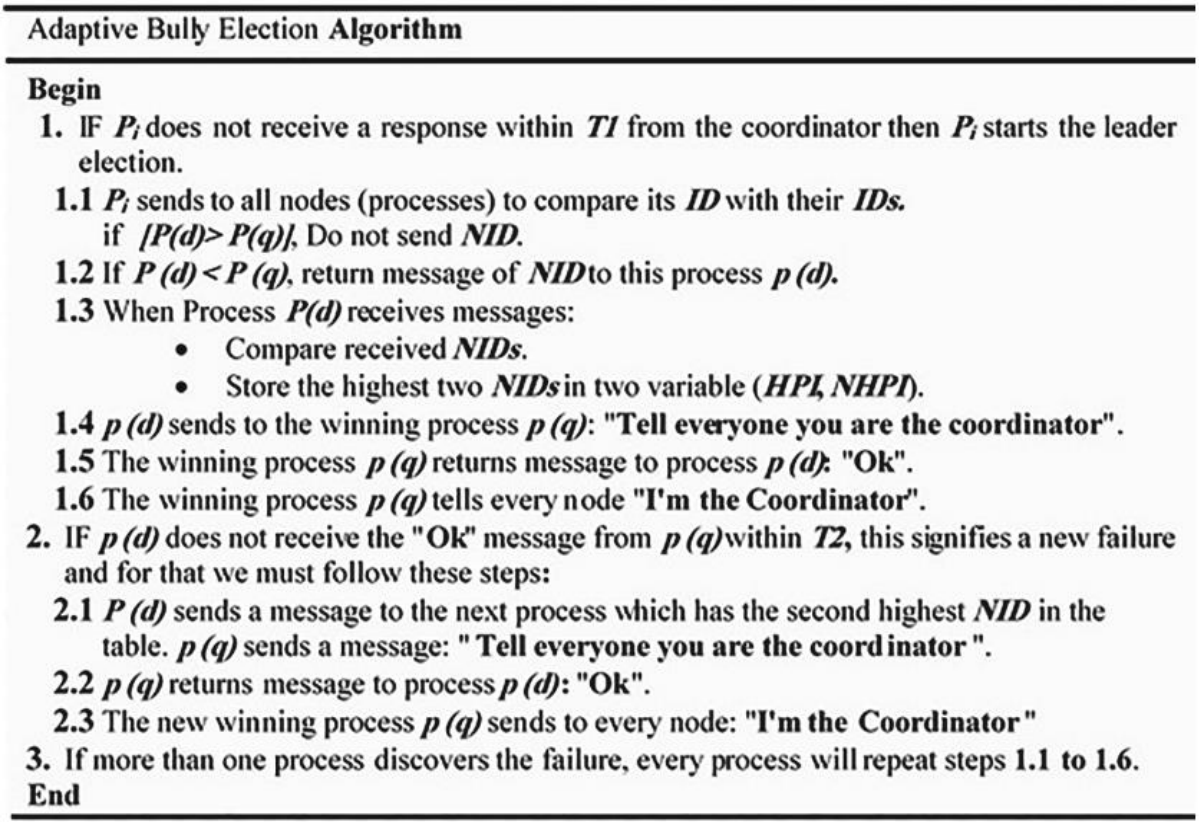
Figure 2. Flowchart of the Modified Sandipan-Basu Bully Algorithm

Once the modification of the Sandipan-Basu was completed, Surolia and Bundele compared their study with Sandipan-Basu's original enhancement of the Bully algorithm and the traditional version. Their study resulted in the modified Sandipan-Basu exceeding the performance of its base study, the Sandipan-Basu Bully Algorithm, and the traditional Bully Algorithm.

In similarity to the Waiting Time-Based Bully Algorithm, the weakness of this version lies in an increased number of inactive nodes as more election messages could be sent to inactive nodes resulting in more time consumed and more instances of attempts at communicating with failed nodes. Additionally, this method requires each node to store a modified process table in its memory which could lead to additional challenges if the distributed system allows the addition and/or deduction of nodes.

Adaptive Bully Algorithm

In the study made by Abdullah et al., (2019), an Adaptive Bully Algorithm is proposed to mitigate the quantity of messages and make the leader election procedure more flexible and secure. To facilitate the leader election process, the suggested technique is based on the Highest Process Identification (HPI) and the Next HPI (NHPI). Furthermore, if the candidate coordinator fails, the leader election is not repeated. The Pseudocode for the Adaptive Bully Algorithm is shown in Figure 3.



Adaptive Bully Election **Algorithm**

Begin

1. IF P_i does not receive a response within *T1* from the coordinator then P_i starts the leader election.

1.1 P_i sends to all nodes (processes) to compare its *ID* with their *IDs.* if *[P(d)> P(q)]*, Do not send *NID.*

1.2 If *P (d) < P (q)*, return message of *NID* to this process *p (d).*

1.3 When Process *P(d)* receives messages:

- Compare received *NIDs.*
- Store the highest two *NIDs* in two variable (*HPI, NHPI*).

1.4 *p (d)* sends to the winning process *p (q)*: **"Tell everyone you are the coordinator".**

1.5 The winning process *p (q)* returns message to process *p (d)*: **"Ok".**

1.6 The winning process *p (q)* tells every node **"I'm the Coordinator".**

2. IF *p (d)* does not receive the **"Ok"** message from *p (q)* within *T2*, this signifies a new failure and for that we must follow these steps:

2.1 *P (d)* sends a message to the next process which has the second highest *NID* in the table. *p (q)* sends a message: **" Tell everyone you are the coordinator ".**

2.2 *p (q)* returns message to process *p (d)*: **"Ok".**

2.3 The new winning process *p (q)* sends to every node: **"I'm the Coordinator "**

3. If more than one process discovers the failure, every process will repeat steps **1.1 to 1.6.**

End

Figure 3. Pseudocode for Adaptive Bully Algorithm

If problems occur during algorithm implementation for the candidate coordinator, this method leads to stopping an additional round of algorithm implementation when it fails in starting. To test their hypothesis, Abdullah et al., (2019) made a comparative analysis between the traditional Bully algorithm, a modified Bully algorithm, and their study - Adaptive Bully Algorithm. The Adaptive Bully Algorithm performed best in

delivering the least number of messages with the same number of nodes in comparison to the mentioned algorithms. Four variables (VE, NID, HPI, NHPI) successfully reduced the complexity of message passing from $O(n^2)$ to $O(n)$.

Much like the Waiting Time-Based Bully Algorithm and the Modified Sandipan-Basu Bully Algorithm, this version's main weakness is the inclusion of inactive nodes. Election messages will be sent from highest to lowest IDs without knowing the active status of these nodes. If there is a high number of inactive nodes during a leader-election there is a high probability that more election messages will be sent.

METHODOLOGY

The modification of the Traditional Bully Algorithm introduces Priority Queuing, which is a sorting technique that works similarly in a linear queue. A priority queue is an abstract data type that treats data components by their priority. The sequence in which items are removed from a queue is determined by their priorities; the item with the highest priority will be taken out first, and the one with the lowest priority will be taken out last (Simplilearn, 2023).

Priority Queuing is used as a solution to the three key issues of the Traditional Bully Algorithm due to its many uses, such as task scheduling, shortest path algorithms, event-driven simulations, Huffman coding, and heap sort. Additionally, they are employed in several computer science and engineering disciplines that call for the sorting and searching of data according to priority as well as in network routing methods (Tyagi, 2023).

By adding this technique to the algorithm, certain steps in the original algorithm will be revised to utilize the new information provided by the technique. An article by Baeldung (2023) stated that systems that manage many programs and their execution where programs are chosen to run depending on their priority, rely heavily on priority queues. They are crucial to networking systems like the Internet because they can aid in prioritizing vital data to ensure that it moves through the system more quickly. The flowchart for the modification of the traditional bully algorithm is shown in Figure 4.

The flowchart illustrates the highlighted modifications made to the traditional algorithm, emphasizing the implementation of Priority Queuing and the utilization of resulting data. While the flowchart appears to entail additional steps, these modifications ultimately streamline the algorithm, reducing the need for excessive looping and thereby minimizing overall workload and time consumption.

The process of leader election will only take place once the coordinator has been detected as failed, which will then be broadcast to the existing nodes with IDs higher than the elector. Once the broadcast has been acknowledged by the recipients, the elector will be able to take note of their IDs and their status. Priority Queuing will be

conducted according to the following conditions: the node can send a reply message that indicates that it is active, and these active nodes will be arranged in the queue from highest to lowest. The elector will then proceed to send an election message to the first entity in the queue. Once the status of the new leader and the previous leader has been confirmed, the leader election will conclude.

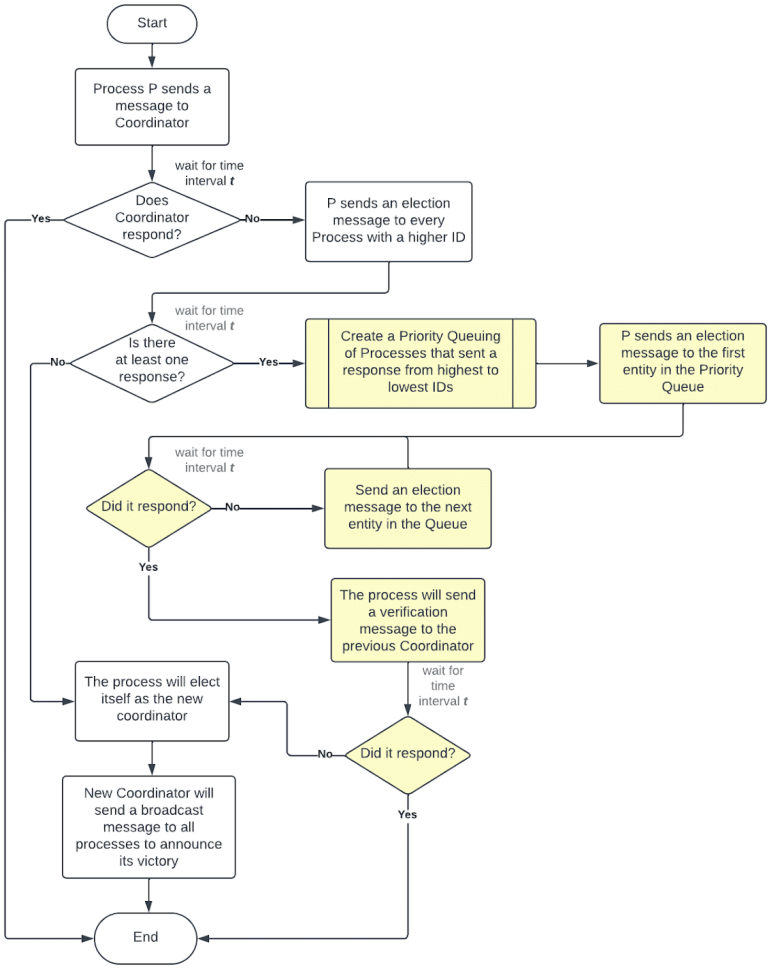


Figure 4. Flowchart of the Modification of Traditional Bully Algorithm

The study intends to compare the performances of three variations of the Bully Algorithm which are the Traditional Bully Algorithm, the Waiting Time-Based Bully Algorithm by Anwar et al., and the proposed Modification of Traditional Bully Algorithm. The simulation ensures that these algorithms share the same data set, as determining which nodes are inactive (if any) are purely randomized.

To determine if this method has succeeded in enhancing the traditional bully algorithm, the Key Performance Indicators (KPIs) will be the following:

- The total messages sent will indicate the exact communication cost of the algorithm with the given parameters (number of processes, time interval, number

of inactive nodes, and elector). In reducing the total number of messages sent, the algorithm will be able to contribute to the overall cost-effectiveness of distributed systems in employing the Bully Algorithm.

- The total time consumed will also be measured as it indicates how much time is needed to complete the leader-election process. The time consumed in milliseconds will be measured by getting the average when utilized by the following Central Processing Units (CPUs) that are available to the researchers: AMD Ryzen 5 PRO 4650G with Radeon Graphics 3.70 GHz, Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, and Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2.50 GHz. The main benefit of reducing the time consumed in completing the Bully Algorithm lies in achieving improved system responsiveness and minimized downtime.
- The total number of redundant messages is also measured, as both the number of messages sent and the time it takes to complete an election process centers around the abundance of redundant messages. Redundant messages in this study are the number of election messages that are received by the same node.
- Lastly, measuring the total instances of communicating with an inactive node will determine if the study has achieved reducing the risks involved in data transmission to a failed entity.

RESULTS

The standard scenario is when there are no inactive nodes, elector E is randomized, and the elected leader is the next highest node. In comparing the performance of the three variations of the Bully Algorithm, the total messages sent, total redundant messages, total time elapsed, and instances of communicating with a failed node are observed which is shown in Table 1.

Table 1. Standard Case of Bully Algorithm Comparative Analysis

No. of Processes	Total Messages			Total Redundant Messages			Total Time Elapsed (ms)			Instances of Communicating with Failed Node		
	T	W	P	T	W	P	T	W	P	T	W	P
10	20	28	17	6	0	0	3,065	1,532	1,521	5	1	2
50	600	147	97	275	0	0	13,325	1,534	1,523	25	1	2
100	2,450	298	197	1,176	0	0	26,061	1,549	1,542	50	1	2
500	62,250	1,498	997	30,875	0	0	128,474	1,551	1,544	250	1	2
1,000	249,500	2,998	1,997	124,250	0	0	267,946	1,786	1,750	500	1	2
5,000	6,247,500	14,998	10,000	3,121,250	0	0	1,324,560	1,798	1,754	2,500	1	2

where:
T = Traditional Bully Algorithm
W = Waiting Time-Based Bully Algorithm
P = Proposed Modification of Traditional Bully Algorithm

In two out of the four comparisons made for the standard case scenario, the Proposed Modification performs better than the Traditional and Waiting Time-Based; while the Waiting Time-Based is superior to the Proposed Modification regarding the instances of communicating with a failed node, and both Waiting Time-Based and Proposed Modification performing equally in terms of handling redundant messages.

The best-case scenario for both the Traditional Bully Algorithm and the Waiting Time-Based Bully Algorithm is when the next highest node starts the election and eventually elects itself as the new leader. This scenario entails that only the next highest node is active, and the elector is also the next highest node.

In this specific scenario, each version of the Bully Algorithm excels in one of the attributes used in comparison. In terms of the number of messages sent, the Traditional Bully Algorithm is the better choice. In terms of total time elapsed, the Proposed Modification outperforms the others. All three perform equally in terms of handling redundant messages for this scenario. Lastly, the Waiting Time-Based is best in terms of the least instances of communicating with a failed node (Table 2).

Table 2. Best Case Scenario of Traditional and Waiting Time-based Bully Algorithm Comparative Analysis

No. of Processes	Total Messages			Total Redundant Messages			Total Time Elapsed (ms)			Instances of Communicating with Failed Node		
	T	W	P	T	W	P	T	W	P	T	W	P
10	22	21	9	0	0	0	1,576	1,552	1,505	9	8	9
50	48	100	49	0	0	0	1,537	1,532	1,528	49	48	49
100	98	200	99	0	0	0	1,557	1,555	1,529	99	98	99
500	498	1,001	499	0	0	0	1,529	1,530	1,530	499	498	499
1,000	998	2,001	999	0	0	0	1,542	1,533	1,512	999	998	999
5,000	4,998	10,000	4,999	0	0	0	1,642	1,629	1,597	4,999	4,998	4,999

where:
T = Traditional Bully Algorithm
W = Waiting Time-Based Bully Algorithm
P = Proposed Modification of Traditional Bully Algorithm

The Worst-case situation for the Traditional Bully Algorithm would be if the lowest node started the election and had to communicate with all nodes between itself up to the highest node ID repeatedly. Table 3 compares the performance of the three versions of the Bully Algorithm in terms of total messages sent, total redundant messages, total time

elapsed, and instances of interacting with a failing node; where the number of inactive nodes is zero and the elector for each is also zero.

In two of the four comparisons made for the worst-case scenario of the Traditional Bully Algorithm, the Proposed Modification outperforms the Traditional and Waiting Time-Based; however, the Waiting Time-Based outperforms the Proposed Modification in reducing communication with a failed node, while both the Waiting Time-Based and Proposed Modification perform equally when handling redundant messages.

The Worst-Case Scenario for the Waiting Time-Based Bully Algorithm is when the lowest-ranking node starts the election and becomes the sole active node to take over as leader. With N-1 inactive nodes and zero as elector, Table 4 compares the performance of the three versions of the Bully Algorithm in terms of total messages sent, total redundant messages, total time elapsed, and instances of interacting with a failing node.

Table 3. Worst Case of Traditional Bully Algorithm Comparative Analysis

No. of Processes	Total Messages			Total Redundant Messages			Total Time Elapsed (ms)			Instances of Communicating with Failed Node		
	T	W	P	T	W	P	T	W	P	T	W	P
10	80	28	27	35	0	0	5,602	1,511	1,546	10	1	2
50	2,400	148	147	1,175	0	0	26,055	1,536	1,527	50	1	2
100	9,800	298	297	4,850	0	0	52,024	1,546	1,538	100	1	2
500	249,000	1,498	1,497	124,250	0	0	256,761	1,535	1,534	500	1	2
1,000	998,000	2,998	2,997	498,500	0	0	535,531	1,611	1,610	1,000	1	2
5,000	2,949,000	14,998	14,997	1,024,050	0	0	1,705,230	1,703	1,662	5,000	1	2

Table 4. Worst Case of Waiting Time-Based Bully Algorithm Comparative Analysis

No. of Processes	Total Messages			Total Redundant Messages			Total Time Elapsed (ms)			Instances of Communicating with Failed Node		
	T	W	P	T	W	P	T	W	P	T	W	P
10	17	21	17	4	0	0	5,587	1,532	1,518	17	9	9
50	96	101	76	25	0	0	26,071	1,536	1,534	97	48	25
100	196	201	139	62	0	0	51,482	1,539	1,537	197	98	38
500	996	1,001	968	133	0	0	256,416	1,531	1,532	997	498	278
1,000	1,996	2,001	1,706	295	0	0	509,303	1,612	1,649	1,997	998	705
5,000	9,996	10,001	6,534	3,467	0	0	2,559,710	1,695	1,607	9,997	4,998	1,533

In three out of four instances of comparison for this scenario, the Proposed Modification outperforms the other two versions. The only instance where the Proposed

Modification did not outperform is when it is equal to the Waiting Time-Based Bully Algorithm in terms of handling redundant messages.

DISCUSSION

The performance of a CPU's allocation is affected by its resources such as storage and the availability of nodes to perform tasks. One of the main goals of modern computer architecture is to process information efficiently inside the memory. By cutting down on the cost of time and communication between the memory and the processor, data processing may significantly reduce the systems' latency and use of resources (Ben-Hur et al., 2020).

In the standard case and worst-case scenario for Traditional, the proposed modification outperforms in nearly all aspects, with Waiting Time-Based relatively close in competition. For the best-case scenario, there is no clear outperforming variation. However, regarding the worst case for Waiting Time-Based, the proposed modification is the most efficient among the three.

In the Case Scenario of Traditional and Waiting Time-based Bully Algorithms for Comparative Analysis, Each version of comparison excels in different attributes. In terms of messages, the Traditional Bully Algorithm outperforms the three comparisons. In total time-elapsed the proposed modification is the best choice. However, in regarding the Waiting Time-Based is best in terms of the least instances of communicating with a failed node.

The Waiting Time-Based Bully Algorithm and the proposed modification have significantly reduced the KPIs from the Traditional Bully Algorithm. Both variations reduced the time and communication costs by eliminating redundant messages by sending a single election message to their respective version's next most eligible candidate for the new leader instead of sending an election message to all possible candidates.

The proposed modification only trumps over the Waiting Time-Based during an increased presence of inactive nodes in the distributed system as the Waiting Time-Based is still prone to sending an election message to node(s) that are inactive. In return, Waiting Time-Time Based trumps the proposed modification when there is little to no presence of inactive nodes. This results in the proposed modification to communicate with a failed node more than the Waiting time based on a single instance. However, the purpose of that additional instance is to ensure the inactivity of the previous leader node, which may restart the leader-election process if it is reactivated.

CONCLUSIONS AND RECOMMENDATIONS

This study's objective is to reduce the communication cost, the time consumed, and instances of communicating with a failed node from the Traditional Bully Algorithm. The proposed modification has achieved all of these in nearly all cases; however, it is not completely better nor worse than the latest enhancement, which is the Waiting Time-based Bully Algorithm by Anwar et al.

While conducting the results and discussion of the study, the researchers came up with the following recommendations for future studies:

The proposed modification has greatly lowered the number of messages involved in the leader election at the cost of not considering the reactivation of nodes after the Priority Queuing has been made. If there is a way to achieve the same objectives in addition to considering the reactivation of nodes with minimal communication to the inactive nodes will make the Bully Algorithm more flexible and adaptive. Furthermore, the study has also succeeded in lowering the time it takes to complete a leader election with Priority Queuing; wherein the most time-consuming task in the process is awaiting the time interval between each communication among all nodes. If there is a way to fine-tune the timeout interval per communication, it can improve the algorithm's responsiveness and fault tolerance. Lastly, it is also ideal to explore other strategies in enabling multiple nodes to initiate the election process simultaneously, while also minimizing the message cost of each election. This can expedite the leader election process, especially in larger distributed systems.

These recommendations aim to address various aspects of the Bully Algorithm that can further improve its adaptability, efficiency to security, and fault tolerance. These enhancement recommendations can be beneficial to system architects, designers, developers, and engineers.

IMPLICATIONS

In the fields of high-performance computing, artificial intelligence, big data analytics, machine learning, deep learning, signal processing, and bioengineering, parallel and distributed computing has become essential (Dhariwal, 2023). Achieving the study's goals will enhance the Bully Algorithm's performance, reliability, responsiveness, and fault tolerance for leader-election processes in distributed computing.

In cloud computing environments, where multiple virtual machines (VMs) are hosted across distributed servers, the improved Bully Algorithm could be used for leader election among VM instances. This ensures efficient resource coordination and management, enabling seamless scaling and load balancing across the cloud infrastructure.

In blockchain networks, where distributed nodes collaborate to maintain a decentralized ledger of transactions, the improved Bully Algorithm could be used for leader election among blockchain nodes to coordinate consensus mechanisms, block validation, and transaction processing. This enhances the security, scalability, and efficiency of blockchain networks.

In each of these scenarios, the improved Bully Algorithm plays a crucial role in facilitating efficient coordination, fault tolerance, and scalability in real-world distributed systems, contributing to improved system performance, reliability, and user experience.

ACKNOWLEDGEMENT

This research would not be successful without the assistance and help of those who inspired and encouraged the researchers to conduct the study. The researchers would like to express their appreciation to all the people and organizations that aided the researchers in conducting this study, the following:

To the thesis adviser for this study, for generously providing her time, invaluable guidance, and unwavering support. Her constructive criticisms greatly contributed to the refinement of the study and provided the researchers with unbiased feedback to aid in thinking outside the box.

To the panelists, who provided valuable feedback and suggestions during the defense of this research. Their insightful comments and recommendations have significantly enhanced the quality of the study.

To the coordinators for this thesis who determinedly pushed the researchers to complete the study. The coordinators have dedicated much of their time and efforts in supporting the researchers to provide the strength to overcome challenges and persevere until the end.

Finally, the researchers would like to acknowledge all the individuals who have contributed in any way, shape, or form to the success of this research. No matter the weight of support that has been provided, it is greatly appreciated.

FUNDING

The study did not receive funding from any institution.

DECLARATIONS

Conflict of Interest

The researchers declare no conflict of interest in this study.

Informed Consent

The study did not require informed consent as participants were not needed to conduct this study.

Ethics Approval

The study did not require ethics approval.

REFERENCES

Abdullah, M., Al-Kohali, I., & Othman, M. (2019). An adaptive bully algorithm for leader elections in distributed systems. In *Parallel Computing Technologies: 15th International Conference, PaCT 2019, Almaty, Kazakhstan, August 19–23, 2019, Proceedings 15* (pp. 373-384). Springer International Publishing.

Anwar, M. N. B., Nahar, A., Md, N. K., & Shuvo, M. H. (2022). A Waiting time-based bully algorithm for leader node selection in distributed systems. *Malaysian Journal of Science*, 38-43.

Azzam, A., Aboshama, A., Ali, R., & Fekry, M. (2020). *Leader Deputies Algorithm for leader election in distributed systems*. Retrieved from ResearchGate.

Baeldung. (2024, March 18). *Priority queue | Baeldung on Computer Science. Baeldung on Computer Science.* Retrieved from https://www.baeldung.com/cs/priority-queue

Ben-Hur, R., Ronen, R., Haj-Ali, A., Bhattacharjee, D., Eliahu, A., Peled, N., & Kvatinsky, S. (2020). SIMPLER MAGIC: Synthesis and mapping of In-Memory logic executed in a single row to improve throughput. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 39*(10), 2434–2447.

Dhariwal, N. (2023, December). Using Machine Learning Regression Model to Predict the Optimum Election Algorithm for Parallel and Distributed Computing Systems. In *2023 Third International Conference on Smart Technologies, Communication and Robotics (STCR)* (Vol. 1, pp. 1-5). IEEE.

Guo, H., Li, W., Nejad, M., & Shen, C. C. (2020). Proof-of-event recording system for autonomous vehicles: A blockchain-based solution. *IEEE Access*, *8*, 182776-182786.

Kanwal, S., Iqbal, Z., Irtaza, A., Ali, R., Siddique, K. (2021). A genetic-based leader election algorithm for IoT cloud data processing. *Computers, Materials & Continua, 68*(2), 2469-2486.

Lamport, L. (2019). Time, clocks, and the ordering of events in a distributed system. In *Concurrency: The works of Leslie Lamport* (pp. 179-196). https://doi.org/10.1145/3335772.3335934

Madisetti, V. K., & Panda, S. (2021). A dynamic leader election algorithm for decentralized networks. *Journal of Transportation Technologies, 11*(3), 404-411.

Numan, M., Subhan, F., Khan, W. Z., Assiri, B., & Armi, N. (2018, November). Well-organized bully leader election algorithm for distributed system. In *2018*

International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET) (pp. 5-10). IEEE.

Shenoy, P. (March 2022). CMPSCI 677 Operating Systems. UMass Amherst. Retrieved from: https://lass.cs.umass.edu/~shenoy/courses/spring22/lectures/Lec14_notes.pdf

Simplilearn. (2023). Priority queue in data structure: implementation, types, and more. Simplilearn.com. Retrieved from: https://www.simplilearn.com/tutorials/data-structure-tutorial/priority-queue-in-data-structure#:~:text=Priority%20queue%20in%20a%20data%20structure%20is%20an%20extension%20of,the%20element%20with%20lower%20priority.

Surolia, J., & Bundele, M. M. (2020). Design and analysis of modified bully algorithm for leader election in distributed system. In *International Conference on Artificial Intelligence: Advances and Applications 2019: Proceedings of ICAIAA 2019* (pp. 337-347). Springer Singapore.

Tyagi, G. (2023). *Priority Queue (Data Structure)*. Retrieved from https://www.codingninjas.com/studio/library/applications-of-priority-queue

Wan, Z. (2023). Fundamental Algorithms in Distributed Systems. *Journal Software, 18*(1), 44-54.

Author's Biography

Ms. Richelle Rose R. Alcaide, the primary author of this paper, is a student at Pamantasan ng Lungsod ng Maynila (University of the City of Manila), Philippines. She graduated from high school with honors at Manila Science High School and is in the process of completing her bachelor's degree in computer science. She aims to focus her research on enhancing algorithms and studies related to Machine Learning, Natural Language Processing, and Data Analytics.

Mr. Saimon C. Rumol, the primary author of this paper, is also a student at the Pamantasan Lungsod ng Maynila, He graduated from National Teacher's College with honors and is in the process of completing his Bachelor's Degree in Computer Science at the University of Pamantasan Lungsod ng Maynila.

Ms. Vivien A. Agustin is an Assistant Professor at Pamantasan ng Lungsod ng Maynila, serving as Program Chairperson of the Information Technology Department. With 22 years of experience, she previously worked as a professor and program coordinator at Universidad de Manila. She holds a Bachelor's degree in Information Technology from St. Paul University in Tuguegarao, Cagayan, and a Master's degree in Information Technology and Public Management Governance. Currently pursuing a Doctorate in Information Technology at La Consolacion University Philippines, she is actively involved in professional organizations such as PSITE-NCR and Institute of Industry and Academic Research Incorporated, while also contributing research publications to the global Information Technology field.

Mr. Mark Christopher R. Blanco is an Instructor 3 at Pamantasan ng Lungsod ng Maynila and is designated as the Chief of the Information and Communications Technology Office of the same University. Mr. Blanco is currently taking his Masters in Information Technology at Pamantasan ng Lungsod ng Maynila. Currently, Mr. Blanco focuses on research and studies related to Natural Language Processing, Machine Learning, Image Processing, Neural Networks, Algorithms, Data Analytics, and Data Mining.

Mr. Jonathan C. Morano is a full-time Instructor and Capstone Coordinator at Baliwag Polytechnic College, concurrently serving as a part-time Lecturer and Thesis Writing Coordinator at Pamantasan Ng Lungsod Ng Maynila. He holds a Bachelor of Science degree in Computer Science from the Technological University of the Philippines - Manila and is currently pursuing a Master of Science in Information Technology at La Consolacion University Philippines – Malolos Bulacan. His research interests encompass algorithm enhancement, with a focus on Beaufort Cipher, K-Nearest Neighbors (KNN), and Vigenère Cipher.

Ms. Leisyl M. Mahusay is a graduate of Master of Engineering Management major in Systems Management and currently taking up a Doctor of Philosophy in Computer Science (PHdCS) at Technological Institute of the Philippines (TIP), Manila. A Licensed Examination for Teacher (LET) passer. She graduated from the Pamantasan ng Lungsod ng Maynila (PLM) with a degree Bachelor of Science in Computer Science in 1993. Ms. Mahusay is a career-oriented woman. She works as a permanent IT Officer and a part-time faculty member at the College of Information System and Technology Management (CISTM. She became an Assistant Vice President of the Information and Communication Technology Office (ICTO). She dedicated her life to education. Over the course of her more than twenty (20) years as a full-time faculty member, she held many positions within the College of Engineering and Technology, including lead, chair, and college secretary.

Ms. Jamillah S. Guialil is a dedicated computer scientist and educator. She earned her Computer Science degree from Pamantasan ng Lungsod ng Maynila (PLM). Currently serving as a Computer Programmer II at PLM, Jamillah plays a vital role in developing and maintaining essential software systems for the university. Alongside her professional role, she serves as a part-time faculty member in the Computer Department, inspiring students and fostering a dynamic learning environment. Actively engaged in the tech community, Jamillah's dedication to lifelong learning reflects her unwavering passion for technology and education.