Ingénierie des Systèmes d'Information
Vol. 28, No. 4, August, 2023, pp. 843-851
Journal homepage: http://iieta.org/journals/isi

A Robust, Preference-Based Coordinator Election Algorithm for Distributed Systems

Shital Subhashchandra Supase[1], Jayshree Rahul Pansare[1,2*]

[1] Department of Computer Engineering, SCTR's Pune Institute of Computer Technology, S.P. Pune University, Pune 411043, India
[2] Department of Computer Engineering, M.E.S. College of Engineering, S.P. Pune University, Pune 411001, India

Corresponding Author Email: jayshree.pansare@mescoepune.org

https://doi.org/10.18280/isi.280405

Received: 17 May 2023
Revised: 6 August 2023
Accepted: 17 August 2023
Available online: 31 August 2023

Keywords:
distributed system, peer-to-peer system, Fault-Tolerant Coordinator Election Algorithm (FTCEA), preference-based voting algorithm

ABSTRACT

In peer-to-peer distributed systems, the selection of a reliable coordinator is a pivotal process, often vulnerable to node failure and communication link failure. Herein, we present an innovative Fault-Tolerant Coordinator Election Algorithm (FTCEA) designed to address these issues, specifically crafted to withstand node failures in peer-to-peer distributed systems. Our algorithm distinguishes itself by capitalizing on a unique preference-based method, which incorporates significant nodal attributes into the election process. This integration of nodal attributes contributes to the election of a durable and reliable coordinator, significantly enhancing the robustness of the system. A comprehensive analysis was conducted to measure FTCEA's communication complexity, execution time, and space complexity using a peer-to-peer distributed application. The results demonstrated that FTCEA successfully identifies a coordinator node with a communication cost of O(n) messages and a space complexity linear to the number of attributes, represented as O(n.m). Remarkably, FTCEA demonstrated an approximately 50.10% improvement in communication cost compared to the enhanced Bully algorithm, a widely utilized method in this domain. Moreover, FTCEA can maintain a linear storage cost of O(n), thereby significantly improving the computation cost. In summary, FTCEA offers a scalable and efficient solution for coordinator election in distributed systems, showing promising potential for practical applications in the field. The algorithm's unique design, robustness, and efficiency make it a valuable contribution to the advancement of peer-to-peer distributed systems.

1. INTRODUCTION

In the contemporary digital landscape, a growing number of applications are developed with a distributed design, primarily driven by the necessity to mitigate losses associated with the failure of a single, central entity inherent in centralized systems [1, 2]. Communication between entities, or nodes, in such distributed systems is facilitated through the application of message-passing techniques. At the heart of these systems lies a coordinator entity, a node elected via an election algorithm, which assumes a pivotal role in tasks such as process synchronization, group key distribution, and load balancing [3, 4]. Consequently, the election of a reliable coordinator emerges as a significant challenge within the distributed systems paradigm [5].

Coordinator election methodologies can be broadly classified into extrema-finding and preference-based methods. The former approach elects an entity possessing an extreme value of identity, while the latter considers the performance attributes of a node, electing it based on these attributes [2]. However, the reliability of communication channels used in message passing, a key feature of distributed systems, often stands compromised, making them susceptible to faults and failures [6, 7]. Communication channels employed in message passing, a characteristic feature of distributed systems, have been observed to lack reliability, thus raising their susceptibility to various faults and failures [8, 9]. Similarly, nodes within these systems are vulnerable to security attacks and failures [10, 11].

Despite the known vulnerabilities, prevailing coordinator election algorithms largely presuppose the reliability of entities, which leads to notable challenges [12, 13]. Faults and failures within the election algorithm can significantly impact both communication and computation costs [14, 15]. Classical election algorithms, typically utilizing the extrema-finding method, do not consider any nodal attributes representing the node's credibility [16, 17]. Yet, the consideration of such nodal attributes, such as battery percentage in mobile ad hoc networks, is critical, highlighting the necessity for their integration into the election process [18, 19].

The efficacy of a coordinator election algorithm within a distributed system depends heavily on the underlying network topology, which can take various forms: ring, bus, tree, mesh, torus, and complete network. This study focuses on a peer-to-peer network, where the number of links in the complete network is given by $(n^2-n)/2$, with 'n' being the total number of nodes in the system.

The following sections delve into the details of the study. Section 2 provides an overview of existing fault-tolerant coordinator election algorithms and a comparative analysis of their strengths and vulnerabilities. Section 3 introduces the novel Fault-Tolerant Coordinator Election Algorithm (FTCEA). Performance analysis of FTCEA is detailed in Section 4, while Section 5 concludes the study.

2. RELATED WORK

Features of coordinator election algorithms have been extensively surveyed, revealing a noticeable lack of focus on security and fault tolerance among researchers. It is paramount to facilitate secure coordinator election with inbuilt fault tolerance due to the inherent vulnerabilities and threats present in coordinator election algorithms [4, 20]. These algorithms play a crucial role in decision-making, synchronization, and task distribution within distributed systems. Existing algorithms generally assume reliability of the nodes in the system; however, in practical terms, these nodes are often susceptible to faults and failures [21, 22].

The reliability, or lack thereof, of the communication channels used for message passing in distributed systems can lead to failures of the links that connect nodes. Consequently, coordinator election algorithms remain vulnerable to faults, failures, and potential non-termination of the algorithm, compromising properties such as safety, liveness, and availability [20].

Coordinator election algorithm vulnerabilities can be classified into two categories: security vulnerabilities and failure vulnerabilities [20]. Security vulnerabilities, which may be accidentally triggered or intentionally exploited, can be exacerbated by the use of weak or incorrect cryptographic algorithms for system communication. This can give rise to attacks and system failures. Impersonation and voting log deletion attacks become feasible if standard cryptographic algorithms are not employed [23].

Security vulnerabilities have been addressed in previous work, where a Secure and Reliable Coordinator Election Algorithm (SRCEA) was proposed for secure coordinator node election [24]. SRCEA utilizes a preference-based coordinator election method and standard cryptographic algorithms with a stateful initialization vector for encryption and decryption. SRCEA aims to prevent impersonation, deny voting attacks, and ensure integrity via authenticated encryption.

Failure vulnerabilities may lead to non-termination of the election algorithm, necessitating the design of an election algorithm capable of tolerating or masking system failures during algorithm execution. The fault tolerance feature of the election algorithm ensures the safety, liveness, and robustness of the election algorithm.

Several researchers have proposed coordinator election algorithms that address the aforementioned issues. Murshed and Allen [2] proposed a coordinator election algorithm that tolerates node failures. Their enhanced Bully algorithm divides set nodes into two categories: candidate set and ordinary set. In the event of failure of the majority of the nodes, fault tolerance is ensured by adding election rounds.

Chang and Roberts [21] proposed a coordinator election algorithm for asynchronous systems using the extrema-finding method. A benchmark Bully election algorithm for asynchronous systems was also proposed by Garcia-Molina [1]. Neither of these algorithms tolerate faults that occur during the election process nor offer security features. Furthermore, they do not ensure the reliability of the elected coordinator.

Gallager et al. [3] and Bodlaender [22] also proposed coordinator election algorithms for asynchronous systems using the extrema-finding method. Gallager et al. use a spanning tree for coordinator node election, electing the root node of the spanning tree storing the group view as a coordinator when an existing coordinator fails. On the other hand, Bodlaender proposed an algorithm to elect the highest identity (ID) node as a coordinator. These algorithms also do not guarantee security or fault tolerance.

Vasudevan et al. [23] proposed a Secure Extrema-Finding Algorithm (SEFA) and a Secure Preference-Based Leader Election Algorithm (SPLEA). These algorithms are designed with encryption and hashing security mechanisms and implement confidentiality and integrity using public key infrastructure (PKI) and Message Digest 5 (MD5) algorithms, respectively.

The existing body of literature on coordinator election algorithms for distributed systems presents a spectrum of strategies, each with distinct features and inherent limitations. The core algorithmic complexity in these methods varies from quadratic in the worst-case to linear in the best-case scenarios. Notably, they provide some degree of protection against impersonation and modification attacks.

For instance, Basu [25] introduced an algorithm utilizing the extrema-finding method for asynchronous systems with ring topology. Though armed with a linear communication cost, it lacked any security features or fault tolerance measures. In a deviation from this approach, Sandipan proposed an innovative method where, in the best case, the election process is bypassed altogether. Here, the node with the next highest ID of the recently failed coordinator automatically assumes the coordinator role. Nevertheless, in the worst-case scenario, linear messages are required to complete the process. This method, however, overlooks nodal attributes or preferences during the election.

Jackson [26] proposed an election algorithm able to withstand omission failures and impersonation attacks. Despite these advantages, it still employs the extrema-finding method. Similarly, the algorithm proposed by Stephen has a linear communication cost with ensured safety properties. Daymude et al. [27] proposed an election algorithm for asynchronous networks with a linear communication cost, yet it did not address fault tolerance and security issues.

Rafailescu [6] introduced an algorithm with fault tolerance using a random roulette wheel selection method. This method elects the node that generates a random number in the minimum time as coordinator. Sidik et al. [28] proposed a coordinator election algorithm for ad-hoc networks ensuring termination, uniqueness, and agreement properties. However, Byrenheid et al. [29] proposed a leader election algorithm designed to resist attacks, specifically impersonation attacks.

Ritzkal et. al. [30] discussed the security vulnerability analysis for cloud server. The vulnerability analysis using two prominent security tools, Nmap and Nessus is executed and presented by authors. Secure key management in cloud environment using quantum cryptography is presented by Kranthi and Sanyasi [31]. The proposed technique enhances the execution of the encryption/decoding procedure, and bolsters a more anchored information transmission process utilizing less computational time.

In our prior work, an algorithm was proposed to elect a reliable coordinator, considering nodal attributes in the election voting process [24]. The Secure Reliable Coordinator Election Algorithm (SRCEA) is designed with confidentiality and integrity features and can tolerate node crash failures for less than the majority of nodes.

Amit et al. proposed the Fault-Resistant Leader Election (FRLLE) algorithm, which elects a reliable node that can effectively manage leadership roles [17]. This algorithm

satisfies safety, liveness, and termination properties, while also offering lower computation and communication costs.

To summarize, existing election algorithms aim to improve communication and computation costs. However, they typically assume the reliability of nodes and communication channels, which is not always the case in practical scenarios. Failures during the election process can drastically increase communication costs. Therefore, a fault-tolerant coordinator election algorithm is needed in distributed systems. This study proposes a novel Fault-Tolerant Coordinator Election Algorithm (FTCEA) that considers important nodal attributes for the candidate selection process, leading to the election of a reliable node as a coordinator (Table 1).

Table 1. Comparison of existing algorithms

Sr. No.	Coordinator Election Algorithm	Topology	System Type	Election Method	Communication Cost
1.	Change and Robert [21]	Unidirectional ring	Asynchronous	Extrema finding	$O(n^2)$
2.	Garcia-Molina [1]	Ring	Asynchronous	Extrema finding	$O(n^2)$
3.	Gallager et. al. [3]	Spanning tree	Asynchronous	Extrema finding	*O(5Nlog2N+ 2E)*
4.	Bodlaender [22]	Unidirectional ring	Asynchronous	Extrema finding	*O(N logN)*
5.	Vasudevan et al. [23]	Wireless ad-hoc	Partially synchronous	Preference-based	*O(n)*
6.	Basu [25]	Ring	Asynchronous	Extrema finding	*O(n)*
7.	Jackson [26]	Star	Asynchronous	Extrema finding	*O(n)*
8.	Daymude et al. [27]	Programmable matter	Asynchronous	Extrema finding	*O(n)*
9.	Rafailescu [6]	Wireless ad-hoc	Partially synchronous	Extrema finding	*O(n)*
10.	Sidik et al. [28]	Ad-hoc	Asynchronous	Extrema finding	*O(n)*
11.	Byrenheid et al. [29]	Peer to peer	Synchronous	Extrema finding	*O(n)*
12.	Biswas et al. [17]	Bidirectional ring	Synchronous	Extrema finding	$O(n^2)$
13.	FTCEA (Proposed algorithm)	Peer to peer	Partially synchronous	Preference-based	*O(n)*

Sr. No.	Coordinator Election Algorithm	Fault Tolerated	Coordinator Reliability	Properties Satisfied	Fault Tolerance During the Election Process
1.	Change and Robert [21]	None	None	None	None
2.	Garcia-Molina [1]	None	None	None	None
3.	Gallager et. al. [3]	None	None	None	None
4.	Bodlaender [22]	None	None	None	None
5.	Vasudevan et al. [23]	None	None	Uniqueness Liveness	None
6.	Basu [25]	None	None	None	None
7.	Jackson [26]	Omission failure	None	Safety	None
8.	Daymude et al. [27]	None	None	Termination	None
9.	Rafailescu [6]	Node crash	None	None	None
10.	Sidik et al. [28]	None	None	Termination Uniqueness Agreement	None
11.	Byrenheid et al. [29]	Failures due to an impersonation attack	None	None	None
12.	Biswas et al. [17]	Node crash recovery	A leader is elected based on the node's performance coefficient	Safety Liveness Termination	None
13.	FTCEA (Proposed algorithm)	Node crash and omission failure	The coordinator is chosen based on nodal attributes	Safety Liveness Termination	Yes (Time redundancy)

3. METHODOLOGY

In peer-to-peer distributed systems, each node is given a unique identifier (ID). Extrema-finding methods determine the coordinator node based on the minimum or maximum ID value, as explained in Section 2. On the other hand, preference-based methods rely on nodal attributes to assign preferences to nodes. However, algorithms do not consider nodal attributes in their election process. In this work, three nodal attributes are used for the election process. Practical implications of this work can be in applications such as mobile ad-hoc networks and peer-to-peer group communication applications. There is a need for a reliable entity election process in group key agreement protocols. FTCEA stores nodal attributes maintained by group view G_v including $\{P_1, P_2, P_3, \ldots, P_n\}$.

3.1 Candidate selection

FTCEA utilizes nodal attributes maintained by group view G_v including $\{P_1, P_2, P_3, \ldots, P_n\}$ in the system to select candidates, as demonstrated by Algorithm 1, Candidate_Selection, presented below [32]. The selection criteria for candidate nodes are to have the minimum values for all these attributes. To meet this condition, min-heap trees are used for nodal attribute storage.

Set P_{jts}, P_d, and P_{fc} store the nodal attributes are updated when a node joins or leaves G_v. At any given time, the node with the minimum attribute value can be accessed by retrieving the root of the corresponding min-heap tree.

Algorithm 1. Candidate_Selection (P_d, P_{jts}, P_{fc})
C=Null
P_{idmin}=root (P_d)
$P_{ijtsmin}$=root (P_{jts})
P_{ifcmin}=root (P_{fc})
C={P_{idmin}, $P_{ijtsmin}$, P_{ifcmin}}
End

Eq. (1) is employed to derive candidate set C

$$C = \exists\, i,j,k\ (Pi, Pj, Pk) | (Pi, Pj, Pk \in P)\ \&\ (Pi = Pdmin\ \&\ Pj = Pjtsmin\ \&\ Pk = Pfcmin) \quad (1)$$

When a single entity is eligible for elected as a coordinator then the voting process is not carried out. The node in set C declares itself a coordinator by sending a message IAC (I Am Coordinator). On receiving the IAC message, all member nodes verify the coordinator node and send M_v message.

3.2 Fault-tolerant coordinator election algorithm (FTCEA)

The coordinator election algorithm is executed using the message-passing protocol in the distributed system. Message passing protocol works with unreliable nodes communicating using an unreliable communication channel. Hence the nodes are vulnerable to failure. The hardware and software failures are inevitable. Hence there is a need of tolerating the faults during the election process. Some of the faults that occurred during the election process are omission faults, node crashes, or communication link failures. FTCEA is designed to tolerate node failures. FTCEA selects eligible candidates before starting the voting and election process. It uses the preference-based method for the election process. Nodal attributes joining timestamp, failure count, and distance are considered for the candidate selection process. The preferences of the vote are assigned to the candidate nodes. For example, the candidate set derived is C={P_2, P_{34}, P_{15}}. All nodes in the system send preferenced votes to these nodes now. The preferenced vote message sent by node P_3 is M_{vote} =P_3_timestamp_2_1_3. On receiving the vote messages, all candidates calculate the weighted votes. And the candidate receiving the maximum votes declares itself as a coordinator by sending an I Am Coordinator (IAC) message to all member nodes.

Algorithm 2. Vote (Candidate_Set C)
If (|C|=1) **then**
If (My_{ID}∈C) **then**
M_{iac}←P_c||IAC||timestamp
i←1
While (i≤n) Send M_{iac} to P_i
My_{ID}.CoordinatorStatus ← true
Else
Wait for δ time to receive a M_{iac} message from P_c
If (M_{iac} is not received in δ) **then**
C=Candidate_Selection (P_d, P_{jts}, P_{fc})
Vote (C)
Else
P_c.CoordinatorStatus ← true
Endif
Endif
Else
M_{vote} ← P_{sid} || P_{i1} || P_{i2} || [P_{i3}] || timestamp
i ← 1
While (i≤|C|) Send M_{vote} to P_i
Coordinator (C)
Endif
End

Algorithm 2 Vote is executed if the |C|>1 that is the number of candidates selected is two or three candidates. If there is a single node in set C then the voting messages are not sent and the candidate selected as a coordinator sends an IAC message to the rest of the nodes in the system. The node creates a vote message M_{vote} and sends to the candidate nodes. In case the non-candidate node do not receive a M_{iac} message within time then it can be inferred that the candidate node is not available or failed. In this case, the Vote algorithm is executed again with a new set of candidate nodes derived using the Candidate_Selection algorithm. Algorithm 3 is designed for finalizing the coordinator node by counting votes received. This algorithm is to be executed by the candidate nodes only. In case the nodes failed during the election process is more than the majority that is more than *n/2* then the algorithm round for candidate selection and voting process is to be repeated. Hence, FTCEA tolerates the faults that occurred during the election process by adding additional rounds. Figure 1 shows case-I where the coordinator election process is executed with no failures. All member nodes are synchronized during the election process. The maximum network propagation delay is used for completing the individual step of the election process. During the very first step, a set C is derived from existing member nodes. In the second step, the member nodes send preference votes to candidate nodes. The candidate node calculates the weighted sum of votes in the third step. In the fourth step, a candidate receiving maximum votes declares itself a coordinator by sending an IAC message. Other candidate nodes verify the coordinator node votes count and reply with the M_v messages to the coordinator node indicating it is successfully elected as a coordinator. The verification of the coordinator is the last step in the election algorithm. On receiving the verified message, the coordinator starts sending the IAA message periodically. The communication cost for the election process when there are no node failures during the election process is given by Eq. (2).

Algorithm 3. Coordinator (C)
If (My_{ID} ∈ C) **then**
If (Votes received≥n/2) **then**
While (i≤|C|)
C_{wvci} ← $\Sigma_{j=1to3}$j ∗ VCj
Max_{Vote} ←Max (C_{wvc1}, C_{wvc2}, C_{wvc3})
Else
Vote (C)
Coordinator (C)
Endif
Else
Wait to receive M_{iac}
If (M_{iac} not received) **then**
Candidate_Selection (P_d, P_{jts}, P_{fc})
Vote (C)
Coordinator (C)
Endif
Endif
End

An important factor affecting the cost of the election process is the number of nodal attributes considered. Eq. (2) depicts the communication cost is linear in the nodal attributes considered. The communication cost with only one nodal attribute is shown in Eq. (3).

$$Communication\ cost\ (Case\text{-}I) = m.n + (n-1) \quad (2)$$

$$Communication\ cost\ (Case\text{-}I) = n-1\ where,\ m=1 \quad (3)$$

Figure 2 shows case II where *f* ` nodes may fail during the election process when the failed nodes *f* ` ≤ *n/2* and none of the candidate nodes failed. The election process is carried out and

completed in this case without any additional election cost. The communication cost for case II is given by Eq. (4). The received votes are accepted and the coordinator node is elected based on majority votes.

$$\textit{Communication cost (Case II)} = (n - f').m + ((n - f') - 1) \quad (4)$$

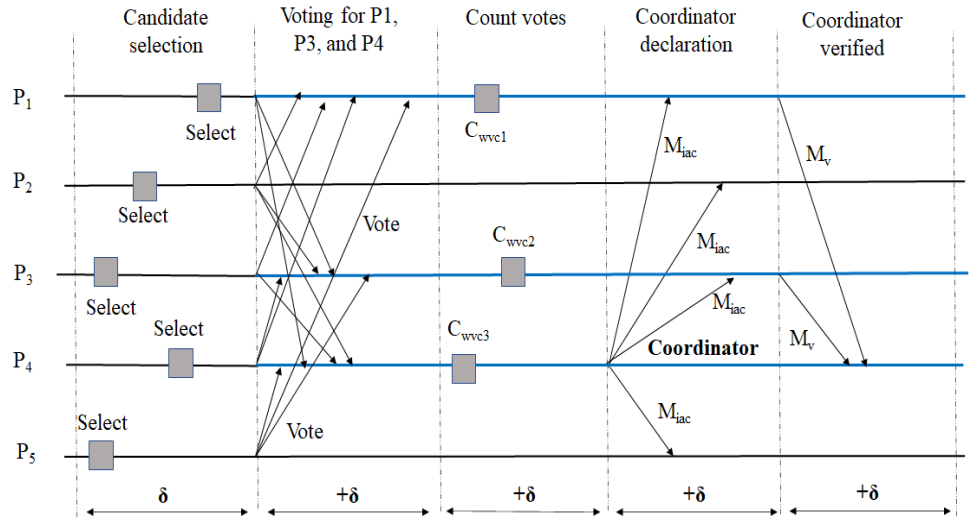


Figure 1. Case I: The coordinator election process with no failures

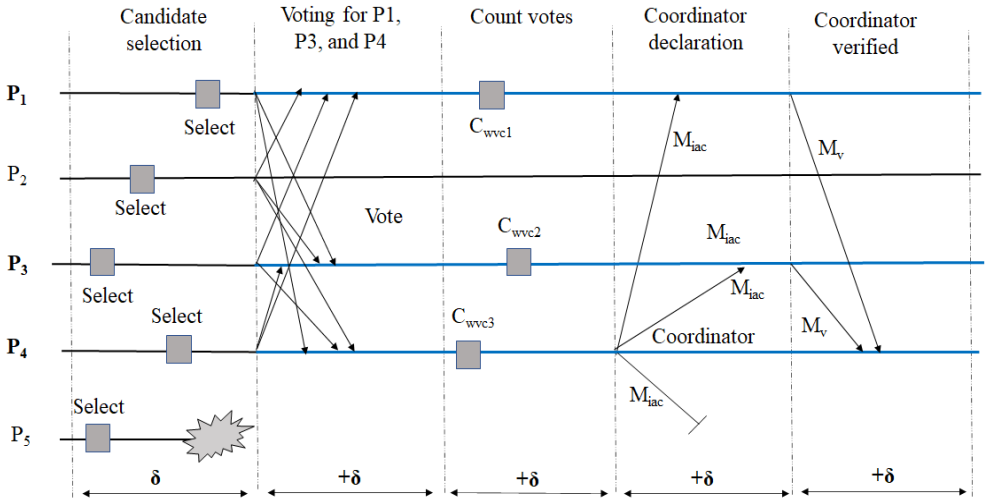


Figure 2. Case II: The coordinator election with $f' \leq n/2$ and *C is not included in f'*

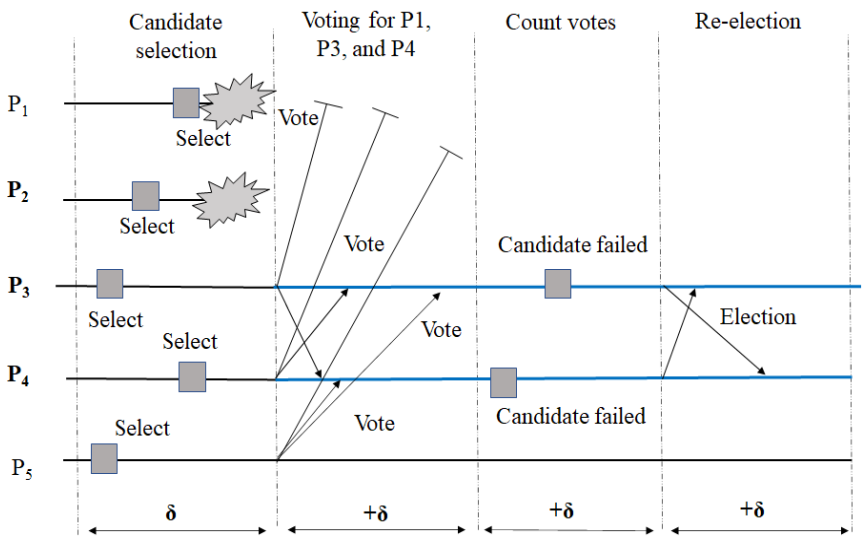


Figure 3. Case III: The coordinator election with failures less than the majority and candidates is included in *f'*

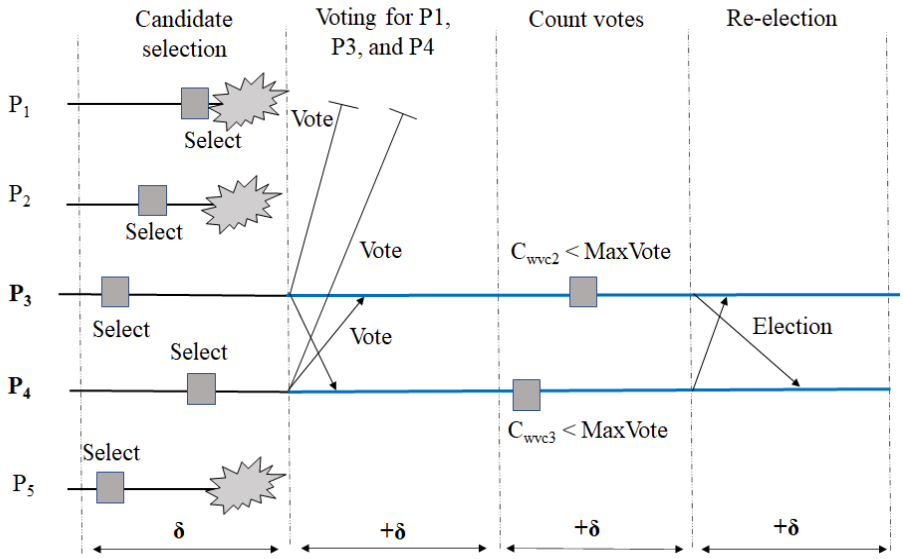


Figure 4. Case IV: The coordinator election with failures more than the majority

Figure 3 shows case III of the election process when less than the majority of the nodes failed and candidate node(s) failed. In case of candidate nodes failure, the election process starting from candidate selection is to executed again. This adds to the communication cost. The communication cost for case III, when candidate nodes are not failed, is given by Eq. (5) below. It shows that the communication cost is still linear if the nodes failed including the candidate nodes. In case III additional communication cost for updating the group, view is involved.

$$\text{Communication cost (Case III)} = 2.m.n + 4.n - 4(f\hat{}-1) \quad (5)$$

Figure 4 shows the election process with more than the majority of the nodes failed that is nodes failed $f\hat{}>n/2$. In this case, the majority vote cannot be calculated. Hence there is a need for a re-election process. The reelection is carried out by updating the group view for identifying the live nodes in the system and then conducting the voting process. As per the updated group view, the majority votes are calculated and a coordinator is elected in the system. The communication cost for case IV is given in Eq. (6) below.

$$\text{Communication cost (Case IV)} = 2.m(n - f\hat{}) + 3n - 3f - 1 \quad (6)$$

The communication for case IV which is $O(n)$ is also given by Eq. (6) as there is a need of updating the group view at first. This process of updating the group view includes sending $(n-f\hat{})$ messages and receiving the reply message from $(n-f\hat{})$ live nodes.

4. RESULTS AND DISCUSSION

The performance measures FTCEA and enhanced Bully algorithm [2] are analyzed. The crucial parameters analyzed are message complexity, time complexity, and space complexity. Communication cost in the election algorithm is the messages exchanged to execute the algorithm to completion. Communication cost has a significant impact on the election algorithm time as there is a delay involved in sending and receiving the messages. Similarly, the computation cost is the time to terminate to completion. The enhanced Bully algorithm is designed to tolerate the node's failure under similar system configurations. The majority of the research on fault-tolerant algorithms handles node failures by re-executing the election algorithms. Whereas, FTCEA is designed to perform the selected round for tolerating the faults. Storage cost represents the memory space required for executing the algorithm to completion. The storage space required for FTCEA is directly proportional to the number of attributes considered for selecting the eligible candidates.

4.1 Computation cost

As shown in Table 2, the computation cost of FTCEA is significantly less than the enhanced Bully algorithm [2]. A computation cost comparison for the best case and worst case is given here. The computation cost is analyzed for the system size of 20. That is the nodes in the system is 20. The communication delay in the system is analyzed to be 200 µsec. The best case of FTCEA includes the time to select a candidate set. The candidate set is created by retrieving the root node of min-heap storing the nodal attribute. In the best case, only one nodal attribute is considered for selecting a candidate for the election process. The worst-case scenario is where more than the majority of nodes fail and election rounds need to be repeated.

Table 2. Comparison of computation cost in the best case and worst case for $n=20$

Algorithm	Computation Cost
Enhanced Bully (Best case)	3800 µsec
Enhanced Bully (Worst case)	12204 µsec
FTCEA (Best case)	7 µsec
FTCEA (Worst case)	1728 µsec

4.2 Storage cost

The storage cost for FTCEA is compared with the Bully algorithm and enhanced Bully algorithm [2]. Table 3 shows that the storage cost is quadratic for the Bully algorithm and for FTCEA it is linear. The storage cost of FTCEA is proportional to the number of attributes. Three nodal attributes are considered during candidate selection in FTCEA. The

storage space for FTCEA is *3n* words. In case the nodal attributes are more than *n* then the storage cost of FTCEA becomes quadratic. As it is infeasible to consider more than 3 to 4 nodal attributes, the storage cost of FTCEA is linear.

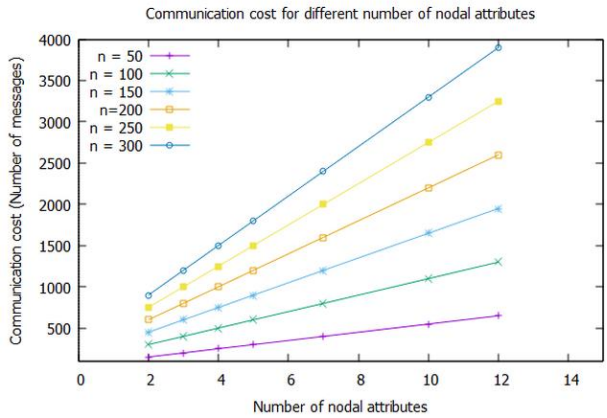
Table 3. Comparison of storage cost

Algorithm	Computation Cost
Enhanced Bully	*O(n)*
FTCEA	*O(n)*

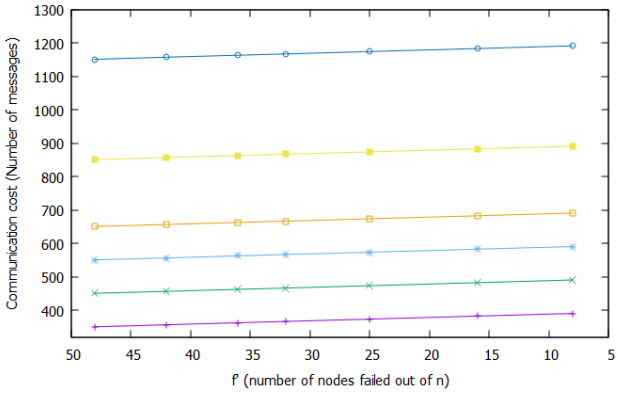
4.3 Communication cost

As shown in Figure 5 (a) the optimum number of nodal attributes found is two or three if the number of nodal attributes increases linearly the communication cost for the election process also increases. The communication cost is less than *2n* if the nodal attributes considered are less than or equal to 3. Figure 5 (a) compares the number of messages for the size of the system varying from *n=50* to *n=300* and a number of nodal attributes ranging from two to 12. It is observed that the computation cost is affected by the number of nodes and the nodal attributes.

Figure 5 (b) shows the communication cost for the nodal attributes and the nodes that failed during the execution of FTCEA. The figure shows the messages exchanged for the nodes failed in the system with *n=100* and failed nodes does not include the candidate nodes. The number of nodes assumed to be failed is chosen randomly and ranges from 8 to 48 nodes. It is observed that with a smaller number of nodal attributes and nodes failing, the communication cost is less. It increases linearly with additional attributes.

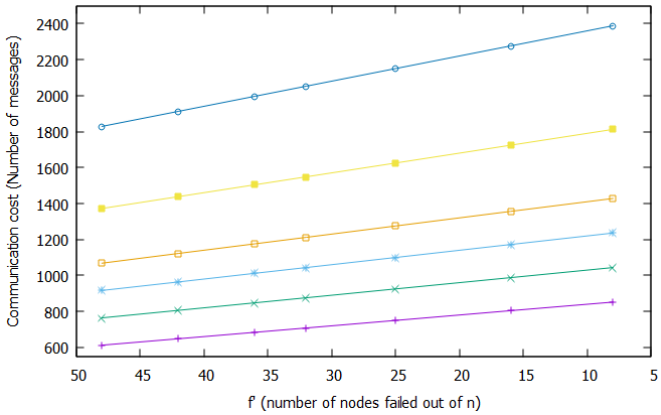


(a) Case-I: Different numbers of nodal attributes

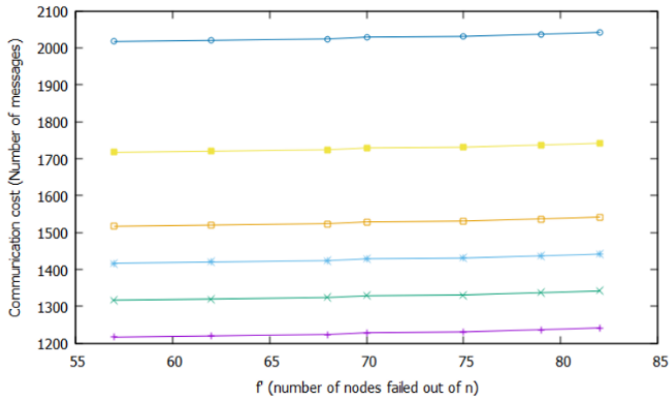


(b) Case-II: $f' < n/2$ *and* $C \notin f'$ and no failures

Figure 5. Comparison of election communication cost in cases I and II



(a) Case III: $f' < n/2$ *and* $C \epsilon f'$



(b) Case IV: $f' > n/2$

Figure 6. A comparison of communication costs in Case III and Case IV

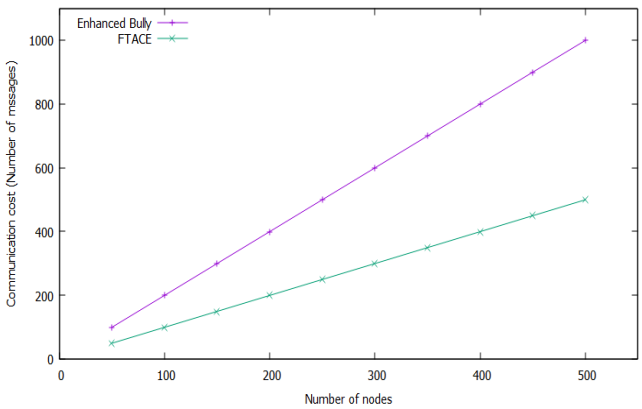


Figure 7. Comparison of FTCEA communication cost in the best case with enhanced Bully algorithm [2]

Figures 6 (a) and (b) show cases III and IV respectively. Case III shows the communication cost for different numbers of nodal attributes, *n=100* and failed nodes include the candidate nodes. In this case, if candidate nodes are failed then there is a need to select the candidate nodes again. The Candidate_Selection algorithm is to be executed and then the algorithm for Vote and Coordinator is to be executed. Thus, the communication cost of case III increases linearly concerning the number of nodes that failed. Case IV shows the communication cost for the election algorithm when the nodes failed is more than the majority of nodes. In this case, all of the algorithms need to be executed in the second round as there is a need to identify live nodes, derive a new candidate set, and

conduct an election also. Figure 7 shows the comparison of communication cost for FTCEA and enhanced Bully algorithm in the best case. The communication cost of the enhanced Bully algorithm and FTCEA in the best case is *O(n)*. The communication cost expression is derived for both of these algorithms and it is observed that the communication cost of the enhanced Bully algorithm and FTCEA is *2n-1* and *n-1* respectively. Hence the communication cost for FTCEA is improved by 50.10% on average.

5. CONCLUSION

A reliable election algorithm for a synchronous system is designed. FTCEA is designed using a preference-based election method. Crucial nodal attributes are used for the election voting process. The communication cost improvement of around 50.10% compared to the enhanced Bully algorithm is observed in the proposed algorithm. There is a significant improvement in the computation cost of FTCEA compared to the enhanced Bully Algorithm. It is ensuring the safety and liveness of properties. FTCEA is a scalable and efficient solution for coordinator election in distributed system applications like peer-to-peer group agreements and mobile ad hoc networks. FTCEA finds the most eligible entity with *O(n)* messages. The FTCEA is designed for a synchronous system with a known communication delay. In future work, it can be incorporated for asynchronous systems with no bounds on communication delay.

REFERENCES

[1] Garcia-Molina, H. (1982). Elections in a distributed computing system. IEEE Transactions on Computers, 31(1): 48-59. https://doi.org/10.1109/TC.1982.1675885

[2] Murshed, M.G., Allen, A.R. (2012). Enhanced bully algorithm for leader node election in synchronous distributed systems. Computers, 1(1): 3-23. https://doi.org/10.3390/computers1010003

[3] Gallager, R.G., Humblet, P.A., Spira, P.M. (1983). A distributed algorithm for minimum-weight spanning trees. ACM Transactions on Programming Languages and Systems (TOPLAS), 5(1): 66-77. https://doi.org/10.1145/357195.357200

[4] Chen, Z., Gul, O.M., Kantarci, B. (2023). Practical byzantine fault tolerance-based robustness for mobile crowdsensing. Distributed Ledger Technologies: Research and Practice, 2(2): 1-24. https://doi.org/10.1145/3580392

[5] Tel, G. (2000). Introduction to distributed algorithms. Cambridge University Press, 227-421. https://doi.org/10.1017/CBO9781139168724

[6] Rafailescu, M. (2017). Fault-tolerant leader election in distributed systems. International Journal of Computer Science and Information Technology, 9(1): 13-20. https://doi.org/10.5121/ijcsit.2017.9102

[7] Byrenheid, M., Strufe, T, Roos, S. (2020). Attack-resistant leader election in social overlay networks by leveraging local voting. Proceedings of the International Conference on Distributed Computing and Networking, Kolkata India, pp. 1-10. https://doi.org/10.1145/3369740.3369770

[8] Fernández-Campusano, C., Larrea, M., Cortiñas, R., Raynal, M. (2017). A distributed leader election algorithm in crash-recovery and omissive systems. Information Processing Letters, 118: 100-104. https://doi.org/10.1016/j.ipl.2016.10.007

[9] Zhang, H., Hao, R.X., Qin, X.W., Lin, C.K., Hsieh, S.Y. (2022). The high faulty tolerant capability of the alternating group graphs. IEEE Transactions on Parallel and Distributed Systems, 34(1): 225-233. https://doi.org/10.1109/TPDS.2022.3217415

[10] Li, Y., Qiao, L., Lv, Z. (2021). An optimized byzantine fault tolerance algorithm for consortium blockchain. Peer-to-Peer Networking and Applications, 14: 2826-2839. https://doi.org/10.1007/s12083-021-01103-8

[11] Jing, G., Zou, Y., Yu, D., Luo, C., Cheng, X. (2023). Efficient fault-tolerant consensus for collaborative services in edge computing. IEEE Transactions on Computers, 72(8): 1-12. https://doi.org/10.1109/TC.2023.3238138

[12] Freitas, L., Tonkikh, A., Bendoukha, A.A., Tucci-Piergiovanni, S., Sirdey, R., Stan, O., Kuznetsov, P. (2023). Homomorphic sortition–single secret leader election for PoS blockchains. Cryptography and Security. Cornell University, pp. 1-20. https://doi.org/10.48550/arXiv.2206.11519

[13] Jannes, K., Beni, E.H., Lagaisse, B., Joosen, W. (2023). BeauForT: Robust byzantine fault tolerance for client-centric mobile web applications. IEEE Transactions on Parallel and Distributed Systems, 34(4): 1241-1252. https://doi.org/10.1109/TPDS.2023.3241963

[14] Kutten, S., Pandurangan, G., Peleg, D., Robinson, P., Trehan, A. (2015). On the complexity of universal leader election. Journal of the ACM (JACM), 62(1): 1-27. https://doi.org/10.1145/2699440

[15] Al-Allaf, A.F., Farej, Z.K. (2023). Simulation-based fault-tolerant multiprocessors system. TELKOMNIKA (Telecommunication Computing Electronics and Control), 21(2): 354-363. http://doi.org/10.12928/telkomnika.v21i2.24253

[16] Zou, Y., Xu, M., Yu, J., Zhao, F., Cheng, X. (2022). Fault-tolerant consensus with NOMA in mobile networks. IEEE Wireless Communications, 29(3): 80-86. https://doi.org/10.1109/MWC.005.2100621

[17] Biswas, A., Tripathi, A.K., Aknine, S. (2021). Lea-TN: leader election algorithm considering node and link failures in a torus network. The Journal of Supercomputing, 77: 13292-13329. https://doi.org/10.1007/s11227-021-03803-7

[18] Mostéfaoui, A., Moumen, H., Raynal, M. (2018). Randomized k-set agreement in crash-prone and Byzantine asynchronous systems. Theoretical Computer Science, 709: 80-97. https://doi.org/10.1016/j.tcs.2017.03.018

[19] Mariani, L., Pezzè, M., Riganelli, O., Xin, R. (2020). Predicting failures in multi-tier distributed systems. Journal of Systems and Software, 161: 110464. https://doi.org/10.1016/j.jss.2019.110464

[20] Supase, S.S., Ingle, R.B. (2020). Are coordinator election algorithms in distributed systems vulnerable. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, pp. 1-5. https://doi.org/10.1109/ICCCNT49239.2020.9225327

[21] Chang, E., Roberts, R. (1979). An improved algorithm for decentralized extrema-finding in circular configurations of processes. Communications of the ACM, 22(5): 281-283. https://doi.org/10.1145/359104.359108
[22] Bodlaender, H.L. (1991). Some lower bound results for decentralized extrema-finding in rings of processors. Journal of Computer and System Sciences, 42(1): 97-118. https://doi.org/10.1016/0022-0000(91)90041-3
[23] Vasudevan, S., DeCleene, B., Kurose, J., Towsley, D. (2001). Secure leader election in wireless ad hoc networks. UMass Computer Science Technical Report, Washington, DC, USA, pp. 1–50. https://doi.org/10.1109/DISCEX.2003.1194890
[24] Supase, S.S., Ingle, R.B. (2021). A novel algorithm for secure and reliable coordinator election in distributed networks. International Journal of Advanced Technology and Engineering Exploration, 8(85): 1682-1694. https://doi.org/10.19101/IJATEE.2021.874588
[25] Basu, S. (2011). An efficient approach of election algorithm in distributed systems. Indian Journal of Computer Science and Engineering (IJCSE), 2(1): 16-21.
[26] Jackson, S.C. (2016). Models of leader elections and their applications. Ph.D. thesis, MISSOURI University of Science and Technology, Missouri.
[27] Daymude, J.J., Gmyr, R., Richa, A.W., Scheideler, C., Strothmann, T. (2017). Improved leader election for self-organizing programmable matter. In Algorithms for Sensor Systems: 13th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, Vienna, Austria, pp. 127-140. https://doi.org/10.1007/978-3-319-72751-6_10
[28] Sidik, B., Puzis, R., Zilberman, P., Elovici, Y. (2018). PALE: partially asynchronous agile leader election. arXiv preprint arXiv:1801.03734. https://doi.org/10.48550/arXiv.1801.03734
[29] Byrenheid, M., Strufe, T., Roos, S. (2020). Attack resistant leader election in social overlay networks by leveraging local voting. In Proceedings of the 21st International Conference on Distributed Computing and Networking, Kolkata India, pp. 1-10. https://doi.org/10.1145/3369740.3369770
[30] Ritzkal, R., Kodarsyah, Amalia, P.P., Mahmud, W., Hendrawan, A.H., Prakoso, B.A., Riawan, I. (2023). Security vulnerability analysis and recommendations for Open Media Vault cloud server on Raspberry Pi. Ingénierie des Systèmes d'Information, 28(3): 711-716. https://doi.org/10.18280/isi.280321
[31] Singamaneni K.K., Naidu P.S. (2018). Secure key management in cloud environment using quantum cryptography. Ingénierie des Systèmes d'Information, 23(5): 213-222. https://doi.org/10.3166/ISI.23.5.213-222
[32] Supase, S., Ingle, R. (2021). Method and system for election of a coordinator node in a distributed network. Indian Patent no. 360624. https://iprsearch.ipindia.gov.in/PatentSearch/PatentSearch/ViewDocuments.

NOMENCLATURE

FTCEA	Fault Tolerant Coordinator Election Algorithm
C	Set of candidate nodes i.e., C={P_{idmin}, $P_{ijtsmin}$, P_{ifcmin}}
G_v	Current group view (set of member nodes)
IAA	I Am Alive
IAC	I Am a Coordinator
n	Number of nodes
m	Number of nodal attributes
f'	Number of failed nodes
P	Set of member nodes {P_1, P_2, …, P_n}
P_d	Set of distance attribute {P_{1d}, P_{2d}, P_{3d}, …, P_{nd}} for G_v
P_{fc}	Set of failure count {P_{1fc}, P_{2fc}, P_{3fc}, …, P_{nfc}} for G_v
P_{jts}	Set of joining time stamp attribute {P_{1jts}, P_{2jts}, P_{3jts}, …, P_{njts}} for G_v
P_i	ID of i^{th} node
P_{id}	Distance of node to the center of network
P_{idmin}	ID of the node with minimum distance from the center of network among G_v
P_{ifc}	Failure count of P_i
P_{ifcmin}	ID of the node with minimum number of failures among G_v
P_{ijts}	Joining time stamp of P_i
$P_{ijtsmin}$	ID of the node joined the network at the earliest among G_v