Journal of Theoretical and Applied Information Technology
30th June 2023. Vol.101. No 12

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A NEW ALGORITHM FOR LEADER ELECTION IN DISTRIBUTED THREE-DIMENSIONAL HONEYCOMB MESH NETWORKS

YOUSEF ALRABA'NAH
Department of Software Engineering, Al-Ahliyya Amman University, Amman, Jordan
E-mail: y.alrabanah@ammanu.edu.jo

ABSTRACT

Leader failure is a fundamental issue in distributed systems. As multiple nodes (processes) work together to achieve a common task, coordination among them is requisite. Lack of a network leader leads to unstable and unreliable network. Moreover, if the leader crashes, a new else process should replace it as early as possible. This paper proposes a new leader election algorithm for leader failure in Three-Dimensional Honeycomb Networks. To simplify the election and to reduce number of exchanged messages, the algorithm breaks up the network into rings. The complexity analysis of the algorithm proves that the algorithm requires O(n) and $O(n^{1.3})$ messages for best and worst cases respectively, in $O(\sqrt[3]{n})$ time steps to elect a new leader.

Keywords: *Leader Election, Honeycomb Mesh, 3-DHM, Distributed Systems, Coordinator*

1. INTRODUCTION

With the ever-increasing advancement of technology, distributed systems are becoming more popular and pervasive. A distributed system is made up of several software components running on various independent computing nodes connected by a network and giving the impression to its users of being a single coherent system. Distributed systems play a significant role in information technology as more and more tasks involved are so large and complex that a single computer cannot handle them alone [1]. Distributed systems provide the abilities to share resources and information more easily and openly, which makes it easier to scale and improve performance, in addition to fault tolerance and transparency [2].

In distributed systems, a large task is divided into a number of potential subtasks and distributed among several system nodes in order to be accomplished quickly. To make communications in a distributed system simpler, one process can be chosen to coordinate and control the activities of all the other processes in the system. To do so, a Leader Election Algorithm (LEA) selects one of the current system nodes to be the leader or coordinator of a distributed system [3]. Usually, a criterion is used to determine who will be the leader, such as choosing the node with the highest identifier. This identifier denotes the node efficiency such as memory size, computational load, or processor speed [4].

The purpose of leader election is to grant special privileges to certain entities such as processes within a distributed system. These privileges might include assigning tasks, modifying data, or responding to system requests [5]. There is also a need for leaders in many fields such as load balancing [6], virtual traffic management [7], clock synchronization [8], task scheduling [9], key distribution and routing coordination [10]. Therefore, the leader node helps to build a consistent, stable, fault-tolerant, reliable, and efficient distributed system [11].

However, determining a single node as leader in a distributed system is a crucial issue that requires a convenient election algorithm. A leader election algorithm guides distributed system nodes to collectively agree that one node to act as leader, with as little interactions as possible [12]. Each node is typically given one of the three states: Leader, Follower, or Candidate [13]. Additionally, the leader should periodically send heartbeat or existence signal, this allows follower nodes to notify if a leader is unavailable or has failed and elect a new leader. Every node has a status variable indicating its situation. When a node is running normally and the leader in the system is active, the node status will be follower. In this case, just one node will have leader status. On the other hand,

candidate status implies that the node is currently participating in an election process, and therefore the leader has been failed. The election process concludes when it is selecting one node as a leader and announces it to others [14].

Many networks topologies have proposed in the literature such as ring, mesh, torus, tree, hypercube, and honeycomb. The network complexity is a primary factor to evaluate any topology, the lower network complexity the better [5]. Several criteria could be used to compute the complexity like diameter length, bisection width, degree, and the network cost [15]. Network cost denotes to the networks implementation cost and its performance and is obtained by product network's diameter length with the degree. The diameter is the shortest path across the network's two farthest nodes, while the degree is the maximum number of links a node has [16]. Referring to the network cost, diameter specifies the message transmission time, whereas the degree specifies the hardware cost. As well, the network is effective if it has few cost for a given number of nodes [17].

This paper proposes a new LEA for Three-Dimensional Honeycomb Mesh (3-DHM) networks. A 3-DHM network consists of multiple identical Honeycomb Mesh (HM) layers that are stacked vertically. According to [18] the network cost is approximately 20% better for 3-DHM networks than 3-D mesh networks. The leader of 3-DHM networks is subject to fail, which makes the network inconsistence, thus a LEA is required to select a new one.

The rest of this paper proceeds as follows: Section 2 gives an overview of the related works. Section 3 introduces 3-DHM networks and its properties. In Section 4, the proposed LEA is presented and explained. Section 5 provides the evaluation of the algorithm. Section 6 concludes the paper.

2. RELATED WORKS

LEAs have been extensively discussed in the literature. Many algorithms have been proposed for various networks topologies. However, it is impractical to apply the same algorithm on different topologies, since LEAs vary based on the algorithm nature, transmission media and the network size [4]. The most two famous LEAs are ring [19] and bully algorithms [20]. The ring algorithm is proposed by Le Lann in [19], the algorithm suggests the processes are logically arranged in the form of a ring. Any process notifies that the leader is crashed starts the election by sending a message to its next process in the ring. The message includes the process ID, and each process in the ring that receives the message adds its ID and forwards it to the next process. The election message continues until it reaches back to the process which initiates it. This process selects the process with the highest ID as the new leader for the network and announces it by circulating a leader message for all processes in the ring. The algorithm requires $O(n^2)$ messages.

On the other hand, bully algorithm [20] designed to solve leader election problem in complete networks. The algorithm considers that each process has a direct link to all other processes in the network, and therefore the processes can communicate directly to each other. The election starts once a process detects the leader failure, the process sends a message only to processes with higher IDs. When a process receives the message, it responds with acknowledgment message indicating that it will take over the election process by sending an election message to processes with higher IDs. Thereafter, only one process will not receive the acknowledgment message which will be elected as the leader for the network. The leader announces itself by sending a leader message to all other processes. Bully algorithm requires $O(n^2)$ messages. Considerable researches were proposed to improve ring algorithm in [1], [21] and [22], and bully algorithm in [23], [24] and [25].

The election problem for a tree network was solved in [26] through using heap structure. The algorithm elects a leader and sets it at the root, while other processes informed about the new leader by a leader message. The algorithm requires $O(n)$ messages.

Researchers in [27] proposed a partially agile leader election algorithm for asynchronous networks, which allows a boundless frequency of node failures and recoveries. The algorithm takes into consideration nodes leaving or joining to the network during the execution, in addition to the nodes jitter. The overall number of messages is $O(n)$.

Supase and Ingle [28] proposed a secure and reliable LEA for distributed networks. The algorithm carries out election using preference-based voting after identifying the eligible candidates. The algorithm secures communication channels among processes and tolerates attacks like denial of voting and impersonation. The algorithm elects a leader in $O(n)$ messages.

Authors in [4] proposed a LEA for honeycomb mesh networks. The algorithm consists of four

Journal of Theoretical and Applied Information Technology
30th June 2023. Vol.101. No 12

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

phases, starting by notifying the leader failure, and terminating by electing one process as a leader and informing others about it. The algorithm divides the network logically into a set of rings, where each ring elects a leader. afterwards, one leader with highest ID among rings leaders is elected as the network leader. The algorithm requires O(n) messages to be completed.

Refai and Alraba'nah [29] discussed hive networks and proposed a LEA to solve leader failure problem. The algorithm extends the LEA in [4] by adding an extra phase, where each layer in the hive nominates a leader. An election process is done among the layers nominating leaders to elect one as the whole network leader. The algorithm requires O(n) messages.

Processes in 3-DHM networks have to communicate with each other's. Due to the lack of leader, the communications will take a lot of time as well a huge number of messages. So we need to elect a leader for such network. To the best of author knowledge, no algorithms have been proposed yet to solve leader failure in 3-DHM networks.

3. 3-DHM NETWORKS

The 3-DHM networks are composed of multiple interconnected HM networks (which called layers) that are identical and connected by vertical links [18]. Authors in [30] first proposed the 3-DHM networks. Afterwards, the 3-DHM networks are discussed and reviewed by several researchers in [31] and [32]. The 3-DHM is obtained by vertically stacking a number of HM layers with a certain size [30], figure 1 shows a 3-DHM network topology.

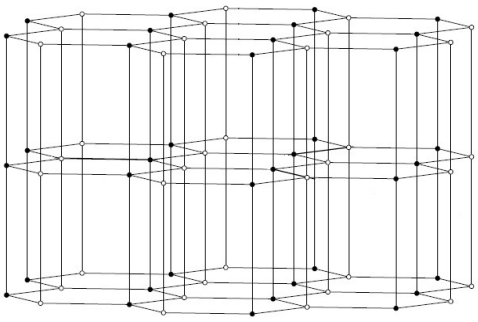

Figure 1: 3-DHM Network Topology

Mikanik [33] suggested that the number of HM layers that constructed the 3-DHM are determined by the size of the 3-DHM. For example, the 3-DHM of size 1 (3-DHM_1) consists of one layer of HM_1, and the 3-DHM2 consists of three HM2 layers, as shown in the figure 1. Thus the 3-DHM_t consists of 2t-1 layers of HM_t. Note that, the HM and 3-DHM are identical exactly when the size is 1, figure 2 shows a HM_2 network [33].

In order to address the nodes in 3-DHM, the coordinate system of HM is extended with an additional axis called V, and hence each node in 3-DHM is addressed using four integer coordinates (x,y,z,v), such that -t+1 ≤ x, y, z ≤ t and –t+1 ≤ v ≤ t-1 [33].

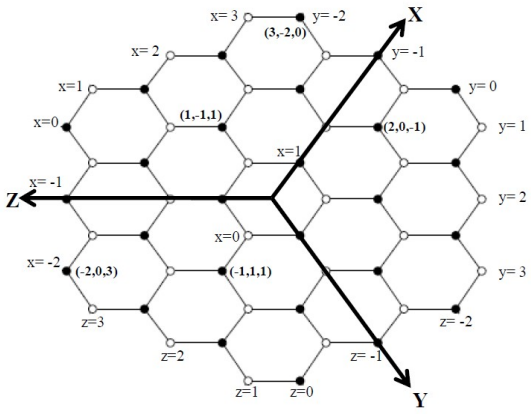

Figure 2: HM_2 Network

The V axis is used to determine the layer of the processing node, the coordinate system for the 3-DHM is shown in figure 3. The V axis starts from the middle layer, which has v=0, the first above layer has v=1, the next above layer has v=v+1. While the first down layer has v=-1, the next down layers has v=v-1. The rest layers are addresses in the same way [4], [18].

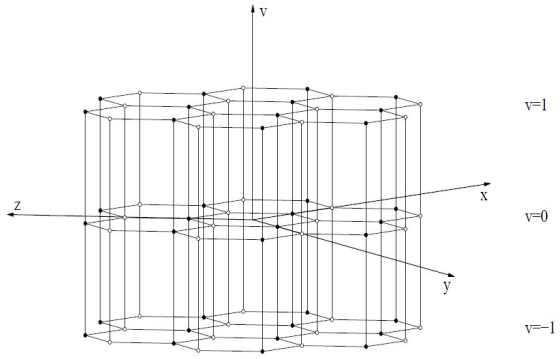

Figure 3: Coordinates of 3-DHM Network

As in HM, each node in the 3-DHM has either white color or black color [31]. The summation of coordinates (x,y,z) except coordinate v equals 2 for each white node and 1 for each black node. The adjacent nodes of a white node (x,y,z,v) are (x-1,y,z,v), (x,y-1,z,v), (x,y,z-1,v), (x,y,z,v-1), and (x,y,z,v+1), while the adjacent nodes for a black

node (x,y,z,v) are (x+1,y,z,v), (x,y+1,z,v), (x,y,z+1,v), (x,y,z,v+1), and (x,y,z,v-1) [31].

Two nodes in the network are connected by an edge if the distance between them is 1, the distance in 3-DHM between any pair of nodes p and p' is computed by equation (1) as the following [30]:

$$distance(p,p')= |x-x'|+|y-y'|+ z-z'|+|v-v'| \quad (1)$$

The overall number of nodes n in 3-DHM$_t$ is obtained by counting the nodes in each layer, this achieved by multiply the number of layers with number of nodes in each layer, the result is presented in equation (2) [30].

$$n(3\text{-}DHM_t)= 6t^2(2t-1) \quad (2)$$

The diameter of the 3-DHM is shorter than the diameter of HM for a given number of nodes, the 3-DHM$_t$ diameter is [33]:

$$diam(3\text{-}DHM_t) = 6t - 3 \quad (3)$$

The 3-DHMt network cost is computed using equation (4), the diameter of 3-DHM$_t$ is 6t-3, and the node degree is 5, so the network cost of 3-DHM$_t$ is [33]:

$$networkcost(3\text{-}DHM_t) = 5(6t-3) \quad (4)$$

Additional edges are presented in the 3-DHM to connect the layers, these edges are numbered with 4 as shown in figure 4, which are parallel to V axis [33].

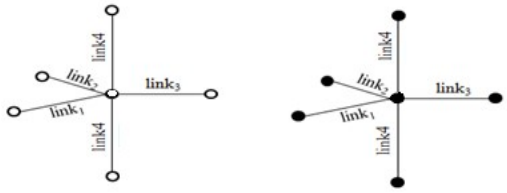

Figure 4: Link Numbers in 3-DHM Networks

The overall number of edges in 3-DHMt is calculated as in the following equation [33]:

$$e(3\text{-}DHMt) = 30t^3-27t^2+3t \quad (5)$$

4. THE PROPOSED ALGORITHM

The leader election is an important matter in the distributed systems, it preserves the system consistency by providing the solution for the failure of the system leader. The LEA starts at the time of leader failure detecting, and terminates when all alive nodes are aware of the new leader. The failure may be detected by one node, subset of nodes, or even by all nodes in the network [4].

4.1. The Algorithm Assumptions

The proposed algorithm assumes that:

- Network routers work all time.
- Network communication links are bidirectional.
- Links work all time without failure.
- Each network process has a priority value, which is ID. The process also has the following information: current leader ID and position, node state, node ring.
- Leader failure is detected by one or more processes if it doesn't receive any acknowledgments from the leader.
- The algorithm may be executed by one or more processes concurrently.
- The failed leader excluded from the current election process.

4.2. The Algorithm Phases

The proposed algorithm composes of five phases, phase one begins when at least one process detects the leader failure. The process initiates a notify message and sends it to other processes on links 1 and 2 to inform other processes in the same layer, as well on link 4 to inform other layers about leader failure. Each process informed about the leader failure starts phase two by sending election messages on its ring. This message elects a leader for each ring. Phase three elects the ring leader with highest ID as the layer leader. In phase four, each layer leader is compared with other layers' leaders to elect one of them as the network leader. Phase five involves sending leader messages to other processes to announce the new leader. Figure 5 shows the algorithm phases.

In phase one, the process that notifies the leader failure changes its state from follower to candidate and composes a notify message to inform processes in other rings in its layer and other layers to start the election in theirs rings. The notify message is sent across links 1, 2 and 4. The notify message is responsible for reporting leader failure and election path that shall be obeyed during the election in phase two. The election path denotes the link numbers and it could be (1, 3) and (3, 1). Another benefits of notify message is that, it ensures all informed processes will conduct the same election path.

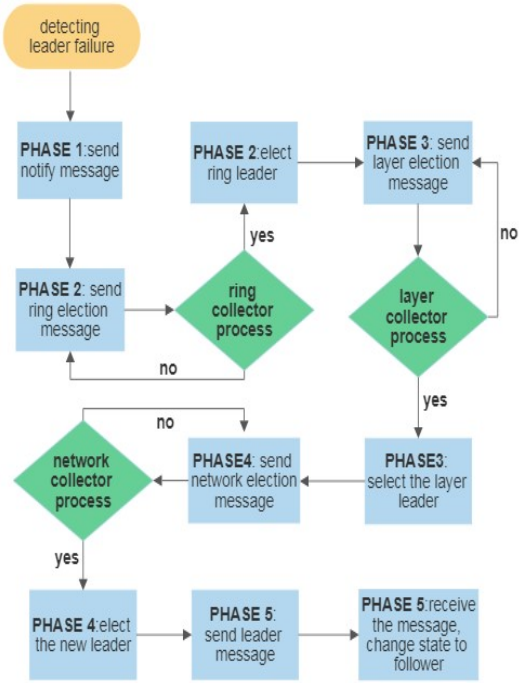

Figure 5: Algorithm Phases

Some processes that receive the notify message also change its state to candidate, pass the notify message to next process, and start phase two by sending election messages on link numbers extracted from the notify message. These processes are specified if the summation of its x and y coordinates equals to the notify message initiator. Note that, the notify message stops when it reaches to an already candidate node.

In phase two, candidate processes exchanged ring election messages to elect a leader for each ring. Each process receives the message compares its ID with the ID included in the message and stores the higher one again in the message. The ring election message continues until it reaches to a ring collector process which is the process with the position (x = ring number, y = -ring number + 1, z = 1, v) in each ring. This process collects the result of the election and elects the process with the highest ID as the leader for each ring. However, in this phase, there is some processes called permutation processes that positioned at the ring where z coordinate value is either t, 1, 0 and -1. The permutation process changes the election path to ensure that the message is conducting its ring path, figure 6 shows 3-DHM_2 rings in red and blue colors, and permutation processes.

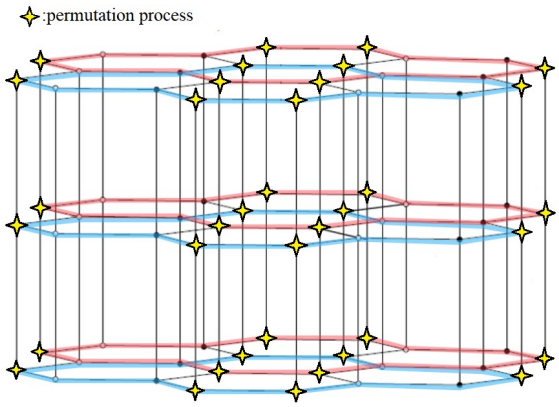

Figure 6: Rings and permutation Processes in 3-DHM_2

The ring collector process with position (x = t, y = -t+1, z = 1, v) in each layer starts phase three by composing a layer election message. The layer election message is exchanged in each layer to compare rings leaders and elect the one with highest ID a leader for that layer. The layer election message will use a path of (1, 2) links. The message continues until it reaches to the process with position (x = 1, y = 0, z = 1, v) which is the layer collector process.

Phase four is initiated by the layer collector processes with positions (x = 1, y = 0, z = 1, v = t-1) and (x = 1, y = 0, z = 1, v = 1-t), these processes create network election messages and pass them to other layer collector processes through links 1 and 4. The network election messages continues until it reaches the network collector process, which is the layer collector process with position (x = 1, y = 0, z = 1, v = 0). The layer collector process elects the layer leader with highest ID as the network leader.

In phase five, the network collector process creates a leader message and broadcasts it to all processes in the network. The leader message is sent via links 1, 2, 3 and 4 and contains the leader information such as its position and ID, the message is used to inform all processes about the new leader. Each process receives the message changes its state to follower and passes the message to its adjacents.

5. COMPLEXITY ANALYSIS

LEAs are evaluated using complexity analysis, where number of messages and time steps are calculated in Big-Oh for best case and worst case. In this section, the complexity analysis of the proposed algorithm is introduced.

5.4 Best Case Analysis

The algorithm requires O(n) messages in O($\sqrt[3]{n}$) times steps.

5.4.1 Number of Messages

Phase one: for one layer, number of messages is 2t [4], for all layers the algorithm needs 2t(2t-1). Moreover, the notify messages are sent vertically across links 4, total number of vertical links in the network of size t is $6t^2(2t-2)$, however, in best case number of vertical links is t(2t-2). So number of messages in phase one is $6t^2 – 4t$.

Phase two: the election here is done within each layer; no messages are sent vertically. Based on the analysis for honeycomb mesh in [4], number of messages in each ring in a layer is 8t-2. Each layer consists of t rings, consequently the number of messages in each layer is t(8t-2). For a network of size t, there is 2t-1 layers, so the number of messages in this message is $16t^3 – 12t^2 + 2t$.

Phase three: number of messages for one layer is 2t-2 [4], therefore, the algorithm requires $4t^2 – 4t + 2$ in this phase.

Phase four: each layer collector process sends a message vertically towards the network collector process, so the number of messages that is required is 2t-2.

Phase five: the network collector process sends a leader messages to all processes in the network. The number of messages is $30t^3 – 27t^2 + 3t – 1$.

By summing the number of messages in each phase, the result will be $46t^3 - 29t^2 – t – 1$. Therefore, the algorithm requires O(n) messages.

5.4.2 Number of Time Steps

Phase one: a layer requires 2t time steps [4]. To send a message for another layer, one time step is required. so the time steps required is 2t + (2t – 2) = 4t - 2.

Phase two: as the election will proceeds in each layer simultaneously, the algorithm requires 4t-1 time steps as in honeycomb mesh in [4].

Phase three: the algorithm requires 2t-2 time steps, as the process with position (x = t, y = -t+1, z = 1, v) sends a layer election message towards layer collector process.

Phase four: layer collector processes at first and last layers send a network election message towards the network collector process, the number of time steps in this phase is t-1.

Phase five: the algorithm sends leader messages in 2t + 1 time steps for one layer [4], the network collector process at the middle layer where v = 0 will send the messages also vertically for all layers. Therefore, the algorithm requires t - 1 + 2t + 1 which is 3t.

The total number of time steps in all phases is 13t – 6, and the algorithm requires O($\sqrt[3]{n}$) time steps.

5.5 Worst Case Analysis

The algorithm requires $O(n^{1.3})$ messages in O($\sqrt[3]{n}$) times steps.

5.5.1 Number of Messages

Phase One: in the worst case, all processes detect the leader failure. Each process will send notify messages across all links except link 3. So the number of messages is $30t^3-27t^2+3t – (3t^2-t)$, which is $30t^3 – 30t^2+4t$.

Phase two: all processes in each ring will send ring election message, number of rings in one layer is t. For all layers in the network the number of rings is $2t^2-t$. One ring requires $6t^2-4t$, all messages exchanged in this phase is $12t^4 – 12t^3 – 4t2$.

Phase three: as in the best case, the number of messages is $4t^2 – 4t + 2$.

Phase four: as in the best case, the number of messages is 2t-2.

Phase five: as in the best case, the number of messages is $30t^3 – 27t^2 + 3t – 1$.

By summing the number of messages in each phase, the result will be $12t^4 + 48t^3 - 57t^2 + 5t – 1$. Therefore, the algorithm requires $O(n^{1.3})$ messages.

5.5.2 Number of Time Steps

Phase one: all processes that detect the leader failure will send notify messages in one time step. The notify message is ignored by any candidate process, and in this case all processes are candidate.

Phase two: as in the best case, the algorithm requires 4t-1 time steps.

Phase three: as in the best case, the algorithm requires 2t-2 time steps.

Phase four: as in the best case, the algorithm requires t-1 time steps.

Phase five: as in the best case, the algorithm requires 3t time steps.

The total number of time steps in all phases is 10t – 3, and the algorithm requires O($\sqrt[3]{n}$) time steps.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6. CONCLUSION

In this paper, a new algorithm has been proposed for leader election among multiple distributed processes that are arranged in 3-DHM networks. The algorithm suggests dividing the network into a set of rings to elect a reliable and competent leader. such rings can utilize the network resources and improve the system performance by reducing the number of messages in communications. The results have been found in this study show that, the proposed algorithm has reasonable performance with ($\sqrt[3]{n}$) time steps and O(n) messages.

In the future, single and multiple links failure should be considered when the processes are communicating with each other. In addition, more investigations should be done on using the proposed rings in routings of 3-DHM networks.

REFERENCES:

[1] A. Biswas, A. K. Maurya, A. K. Tripathi, and S. Aknine, "Frlle: a failure rate and load-based leader election algorithm for a bidirectional ring in distributed systems," *The Journal of Supercomputing*, vol. 77, pp. 751-779, 2021.

[2] N. Naik, "Demystifying properties of distributed systems," *In 2021 IEEE International Symposium on Systems Engineering (ISSE)*, pp. 1-8, IEEE, 2021.

[3] N. Kadjouh, A. Bounceur, M. Bezoui, M. E. Khanouche, R. Euler, M. Hammoudeh, M., and F. Al-Turjman, "A dominating tree based leader election algorithm for smart cities IoT infrastructure," *Mobile Networks and Applications*, pp. 1-14, 2020.

[4] M. Refai, Y. Alraba'nah, M. Alauthman, A. Almomani, M. Al-kasassbeh, and M. Alweshah, "A Novel Leader Election Algorithm For Honeycomb Mesh Networks," *Journal of Theoretical and Applied Information Technology*, vol. 97, pp. 3783 -3795, 2019.

[5] R. Ganesan, X. M. Raajini, A. Nayyar, P. Sanjeevikumar, E. Hossain, and A. H. Ertas, "Bold: Bio-inspired optimized leader election for multiple drones," *Sensors*, vol. 20, no. 11, 2020.

[6] N. D. Nguyen, and T. Kim, "Balanced leader distribution algorithm in kubernetes clusters," *Sensors*, vol. 21, no. 3, 2021.

[7] P. Choudhary, R. K. Dwivedi, U. Singh, "Novel algorithm for leader election process in virtual traffic light protocol," *International Journal of Information Technology*, vol. 12, no. 1, pp. 113–117, 2020.

[8] M. Kara, A. Bounceur, M. Hammoudeh, H. Ngadi, and A. Laouid, "A Lightweight Leader Election Algorithm for IoT: Cloud Storage Use Case," *Research Square*, pp. 1-10, 2023.

[9] S. S. Safa'a, T. F. Mabrouk, and R. A. Tarabishi, "An improved energy-efficient head election protocol for clustering techniques of wireless sensor network," *Egyptian Informatics Journal*, vol. 22, no. 4, pp. 439-445, 2020.

[10] A. Biswas, A. K. Tripathi, and S. Aknine, "Lea-TN: leader election algorithm considering node and link failures in a torus network," *The Journal of Supercomputing*, vol. 77, pp. 13292-13329, 2021.

[11] J. Skrzypczak, F. Schintke, and T. Schütt, "RMWPaxos: fault-tolerant in-place consensus sequences," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 10, pp. 2392-2405, 2020.

[12] S. Kanwal, Z. Iqbal, A. Irtaza, R. Ali, and K. Siddique, "A genetic based leader election algorithm for IoT cloud data processing," *Comput. Mater. Contin*, vol. 68, pp. 2469-2486, 2021.

[13] J. Xu, W. Wang, Y. Zeng, Z. Yan, Z., and H. Li, "Raft-PLUS: Improving Raft by Multi-Policy Based Leader Election with Unprejudiced Sorting," *Symmetry*, vol. 14, no. 6, 2022.

[14] Y. Li, L. Qiao, and Z. Lv, "An optimized byzantine fault tolerance algorithm for consortium blockchain," *Peer-to-Peer Networking and Applications*, vol. 14, pp. 2826-2839, 2021.

[15] M. H. Rahman, M. Al-Naeem, M. N. Ali, and A. Sufian, "TFBN: A cost effective high performance hierarchical interconnection network," *Applied Sciences*, vol. 10, no. 22, 2020.

[16] F. Rad, M. Reshadi, and A. Khademzadeh, "A survey and taxonomy of congestion control mechanisms in wireless network on chip," J*ournal of Systems Architecture,* vol. 108, 2020.

[17] I. A. Alimi, R. K. Patel, O. Aboderin, A. M. Abdalla, R. A. Gbadamosi, N. J. Muga, and A. L. Teixeira, "Network-on-chip topologies: Potentials, technical challenges, recent advances and research direction," *Network-on-Chip-Architecture, Optimization, and Design Explorations*, 2021.

[18] I. SZCZEsniak, "The Hive Network and its Routing Algorithm," *Archive of Theoretical and Applied Informatics*, vol. 16, pp. 171-179, 2004.

[19] G. Le Lann, "Distributed Systems-Towards a Formal Approach," *in IFIP Congress*, 1977, pp. 155-160.

[20] H. Garcia-Molina, "Elections In A Distributed Computing System," *IEEE transactions on Computers*, vol. 100, pp. 48-59, 1982.

[21] C. Maurer, and C. Maurer, "Leader election algorithms," *Nonsequential and Distributed Programming with Go: Synchronization of Concurrent Processes: Communication-Cooperation-Competition*, pp. 381-395., 2021.

[22] K. Altisen, A. K. Datta, S. Devismes, A. Durand, and L. L. Larmore, "Election in unidirectional rings with homonyms," *Journal of Parallel and Distributed Computing*, vol. 146, pp. 79-95., 2020.

[23] J. Surolia, and M. M. Bundele, "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, 2020.

[24] M. Abdullah, I. Al-Kohali, and M. Othman, "An Adaptive Bully Algorithm for Leader Elections in Distributed Systems," In *Parallel Computing Technologies: 15th International Conference, Proceedings 15,* pp. 373-384, 2019.

[25] D. Mitra, A. Cortesi, and N. Chaki, "ALEA: an anonymous leader election algorithm for synchronous distributed systems," In *Progress in Image Processing, Pattern Recognition and Communication Systems: Proceedings of the Conference (CORES, IP&C, ACS),* pp. 46-58, 2022.

[26] M. Sepehri, and M. Goodarzi, "Leader election algorithm using heap structure," In *Proceedings of the 12th WSEAS international conference on Computers*, pp. 668-672, 2008.

[27] B. Sidik, R. Puzis, P. Zilberman, and Y. Elovici, "PALE: partially asynchronous agile leader election," *arXiv preprint arXiv:1801.03734*, 2018.

[28] S. S. Supase, and R. B. Ingle, "A novel algorithm for secure and reliable coordinator election in distributed networks," *International Journal of Advanced Technology and Engineering Exploration*, vol. 8, no. 85, 2018.

[29] M. Refai, and Y. Alraba'nah, "A Ring Based Leader Election Algorithm For Hive Networks," *Journal of Theoretical and Applied Information Technology*, vol 100, no. 6, pp. 1786 -1795, 2022.

[30] J. Carle, J. F. MYOUPO, and I. Stojmenovic, "Higher dimensional honeycomb networks," *Journal of Interconnection Networks*, vol. 2, no. 4, pp. 391-420, 2001.

[31] A. Yin, N. Chen, P. Liljeberg, and H. Tenhunen, "Comparison of mesh and honeycomb network-on-chip architectures," In *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1716-1720. 2012.

[32] N. Pietroni, M. Campen, A. Sheffer, G. Cherchi, D. Bommes, X. Gao, and M. Livesu, "Hex-mesh generation and processing: a survey," *ACM transactions on graphics*, vol. 42, no. 2, pp. 1-44, 2022.

[33] W. Mikanik, "Three-dimensional Variants of Honeycomb Networks," *Technical Report, Silesian University of Technology*, Gliwice, Poland, 1999.