Journal of Theoretical and Applied Information Technology
31st July 2019. Vol.97. No 14

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

A NOVEL LEADER ELECTION ALGORITHM FOR HONEYCOMB MESH NETWORKS

[1]MOHAMMED AL-REFAI, [2]YOUSEF ALRABA'NAH, [1]MOHAMMED ALAUTHMAN, [3]AMMAR ALMOMANI [4]MOUHAMMD AL-KASASSBEH AND [5]MOHAMMED ALWESHAH

[1]Department of Computer Science, Zarqa University, Zarqa, Jordan.
[2]Department of Software Engineering, Al-Ahliyya Amman University, Amman, Jordan.
[3]Department Information Technology, Al-Huson University College, Al-Balqa Applied University, Irbid, Jordan.
[4]King Hussein Faculty of Computing Sciences Princess Sumaya University for technology
[5]Prince Abdullah Ben Ghazi Faculty of Information Technology, Al-Balqa Applied University, Al Salt, Jordan

E-mail: [1]{refai, malauthman}@zu.edu.jo,[2]y.alrabanah@ammanu.edu.jo, [3]ammarnav6@bau.edu.jo
[4]m.alkasassbeh@psut.edu.jo, [5]weshah@bau.edu.jo

ABSTRACT

Leader election is an important issue in distributed systems and communication networks. Many protocols and algorithms that are running on distributed systems need a leader to ensure smooth execution; the leader has the responsibility to synchronize and coordinate the system processes. The absence of the leader makes the system inconsistent, and therefore unreliable. Such problem, however, can be solved by leader election algorithms. In this paper, we propose –for the first time- a new leader election algorithm to solve the leader failure in honeycomb mesh network. The honeycomb mesh network desired due to its low network cost, regularity, and scalability. The proposed algorithm aims to select exactly one node among all active nodes in the network to be a new leader, the node, which is elected as the new leader, has more priority over other nodes. The performance of the proposed algorithm is evaluated by computing the number of messages and time steps required to elect a new leader and complete the algorithm mission. The mathematical evaluation shows that the proposed algorithm requires O(n) messages in O($\sqrt{n}$) time steps in the best case to complete, as well as O($n^{1.5}$) messages in O($\sqrt{n}$) time steps in the worst case.

Keywords: *Leader Election, Distributed Systems, Honeycomb Mesh, Concurrency.*

1. INTRODUCTION

During last decades, distributed systems have used in a wide range of computing domains, and it became more and more crucial in evolving of various systems and applications, such as the web and business applications [1]. Distributed systems consist of multiple independent computers that cooperate with each other to perform common tasks that divided over it. Tasks are distributed on multiple computers to increase the computational speed of problems solving [2]. A group of processes that are communicating through various networks topologies in distributed systems requires one process to be a leader to coordinate and control their communications and actions [3]. Any process in a purely distributed system has to communicate with all other processes to take a certain action. The basic idea for reducing the communication complexity is to choose one process from the current alive processes to be a centralized process, which in turn manages all processes communications over the system [4].

Distributed systems need a coordinator to manage activities and actions. The coordinator is designated to perform particular roles in distributed systems such as managing group communications, reconstructing a lost token in a token ring network, lock server to decide which process can use a shared resource or enter a critical section, time server to synchronize the system processes, or a primary server to manage the replicated data and backups updating operations [5], [6].

The leader is prone to fail for some reasons, to continue the working and based on some criteria, another process should be elected as a substitution for the previous leader. Leader Election Algorithm (LEA) is the process of choosing a single node to be the coordinator or controller of some tasks that distributed over a number of computers (nodes), this coordinator acts as a centralized controller for that decentralized system [7], [8]. There are a wide variety of topologies that have been proposed and used in interconnection networks, such as a ring, mesh, hypercube, torus, tree, and honeycomb [9]. Preferred topologies are those which have less complexity than the others; multiple criteria can be used to evaluate any topology, some of these criteria are diameter length, degree, bisection width, cost, or bisection bandwidth [10], [11].

However, the network cost is considered as the main criterion to evaluate the topologies; the network cost refers to network performance and implementation cost of a topology, which can be defined as the product of network's diameter length and the node degree [12], as shown in the following equation.

$$\text{network\_cost} = \text{diameter} * \text{degree} \quad (1)$$

The diameter denotes to the maximum hop count from the set of network's minimal hops between any pair of nodes in the network, while the degree refers to the number of links that a node has. Nevertheless, nodes in a network may have different degrees, in this case, the degree determined by considering the maximum node degree. A compromising of the diameter and the degree should be taken into account in designing a network topology, where the diameter is related to the message transmission time, and the degree is related to the hardware cost [11]. A topology is said to be effective in terms of network cost if it has a small network cost for a given number of nodes [12].

This paper highlights the two-dimensional Honeycomb Mesh (HM) networks and proposes a new algorithm to elect a leader for it. Honeycomb mesh networks consist of a number hexagons, which is less complex than the mesh network, the complexity (also known as network cost) of the HM is approximately 40% less than the mesh network [11], [12]. The leader in HM networks may fail or crash, to keep the network proceedings, the system demands a LEA to take the initiative to elect a new one, which will be proposed in this paper. For example, let P is a group of N processes that connected over honeycomb mesh networks, P=(p1,p2,…, pN), we must run an algorithm to choose one process Pi, where i=(1,2, …, N) among these processes to be the leader. However, to the best of the author's knowledge, no previous studies have investigated leader election issue in HM networks yet; our research comes to solve this issue by proposing a new LEA that elects one process to become a new leader for the system that is distributed over the HM networks.

The rest of this paper proceeds as follows: Section 2 introduces the related works. In Section 3, we describe and discuss the HM networks and its properties. Section 4 presents the proposed algorithm and gives an illustrated example. Section 5 evaluates the performance of the proposed algorithm using a mathematical model. In Section 6, we conclude the paper and give directions for future works.

2. RELATED WORKS

Leader election is an inherited problem in distributed systems and has been extensively studied in the literature; several algorithms were proposed to solve the leader failure in various networks topologies. The same algorithm is hard to be applied to several systems since LEAs are varied based on algorithm nature, network topology, transmission media, and whether the network size is uniform (known) or non-uniform (not known) [13]. Le Lann in [14] proposed an algorithm to elect a leader in a ring network, the algorithm assumes the processes are logically ordered and organized in a ring, where each process has a communication link to the next process in the ring in a unidirectional way. When any process detects that the leader is not functioning, it initiates an election message that contains its ID and sends the message to next process in the ring. Each process receives the message puts its ID to the list in the message nominating itself a candidate to be elected as a leader. Finally, the message returns to the initiator process which started it; the initiator process selects the list member with highest ID as the new leader, and a new message known as leader message is circulated once again to announce the new leader and the members of the new ring. The message complexity of the ring algorithm is O(n2) messages. However, many enhancements introduced to the ring algorithm such as in [15], [16].

Bully algorithm was proposed in [17] to solve the leader election problem by considering the complete networks, the algorithm known as the bully algorithm. Unlike ring algorithm, the bully algorithm assumes that each process can communicate directly with all processes. A process P that observes the leader failure begins an election algorithm by sending election messages to all processes with IDs larger than its ID, if no process responds to the election message, process P wins the election and becomes the new leader. When a process receives an election message, it replies with an ok message to indicate it is alive and will take over the election algorithm. Afterwards, all processes give up except only one, which is the new leader, the leader announces itself by sending leader message to all other processes. The bully algorithm requires O(n^2) messages; several modifications were proposed to improve the bully algorithm in [18], [19], [20], [21].

Seperhi and Godarzi in [22] proposed a LEA to elect one process to be the new leader for a set of processes that connected by a tree network, this algorithm based on heap structure. The algorithm starts by constructing a tree T of size N processes and ends when a unique process which is greater than the others is elected as the new leader and stored at the root of the tree. Consequently, this process will take over the leader job and declare itself as the new leader by sending leader messages to the others, the message complexity of this algorithm is O(n).

Refai [23] proposed a new LEA to solve the leader failure in hypercube networks, even if the processes IDs are not distinguished, the algorithm requires O(n) messages. The election problem was also solved and examined in 2D torus networks [24] as well as 3D torus networks [13]. The researchers took into consideration the failure of a link during the election process; the link failure was overcome in both algorithms by sending the message on an alternative link. However, the proposed algorithms in 2D and 3D torus networks need O(n) messages to terminate.

Refai et al. [25] put forward an algorithm to solve the leader election problem in 2D honeycomb torus networks. The algorithm composed of three phases, where phase one starts at the detection of leader failure by at least one process, such process sends election messages to its neighbor's processes, the results of phase one are collected by a particular group of processes. These processes in phase two make a new election, and gather the results in one process. In phase three, the process that is aware of the new leader broadcasts a leader message to all processes in the network. However, The algorithm needs O($n^{1.5}$) messages.

A dynamic solution for 2D honeycomb torus networks was proposed in [26] to solve leader election problem in the presence of single link failure. The idea behind the solution is the same as in [13], [24].

3. HM NETWORKS VS MESH NETWORKS

The HM networks topologies are widely studied in the literature and have attracted many research interests due to its high regularity, and scalability [27]. Despite it was firstly reported in 1997 [12] where the author proposed an addressing scheme for the nodes and introduced some of the HM topological properties, HM topologies have been recently showing increasing concerns by many researchers, it further examined and analyzed in [11], [27], [28], [29], [30], [31], [32], [33]. The HM networks vastly used in computer graphics, cellular phone base station, image processing, and wireless sensor networks[28]. An HM topology consists of a number of hexagons; one hexagon forms an HM of size one, which denoted as HM1. The HM of size two (HM2) obtained by surrounding the boundary edges of HM1 with six hexagons. In such manner, HM of size t (HMt) constructed by attaching one hexagon to each edge on the outside boundary of HMt-1[11]. Figure 1 shows HM network with different sizes.

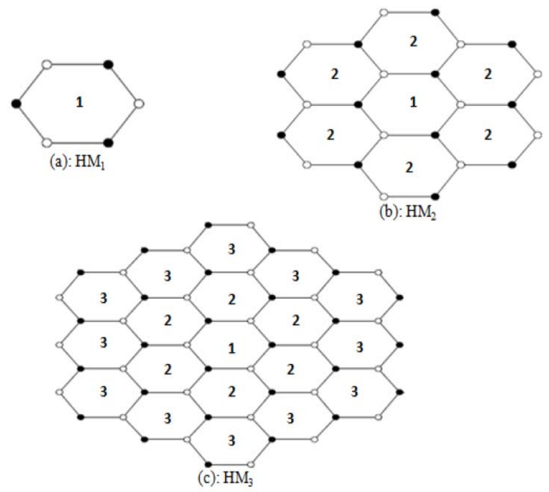

Figure 1: Honeycomb Mesh With Different Sizes

The coordinate system for HM based on X, Y, and Z axes, those axes begin from the center of the topology and split it logically into three regions, see Figure 2. The nodes that connected by any edge that is a parallel to an axis say X share the same coordinates values except the x coordinate value, in other words, a certain coordinate value will change only by following an edge parallel to its corresponding axis [33]. Nodes in HM of size t (HMt) are addressed using three coordinates (x, y, z), such that $-t+1 \leq x, y, z \leq t$. Starting from the honeycomb center, x coordinate value equals one at the first node that meets the x-axis, for the next node in the same movement direction x=x+1, which is two. In the reverse direction, the first node that meets the X axis has x value 0, for the next node x=x-1, which is -1, and so on. The address is designated in the same approach for y and z coordinates as in x coordinate [29]. A zigzag chain formed by crossing the nodes that have the same coordinate (x, y, or z) in a particular direction, for example, consider the z coordinate, all nodes that have the same z-coordinate value will form a zigzag chain. In figure 2, there are six chains with respect to z coordinate, that are z={-2,-1,0,1,2,3}. The zigzag chains for x and y coordinates are obtained in the same way [11].

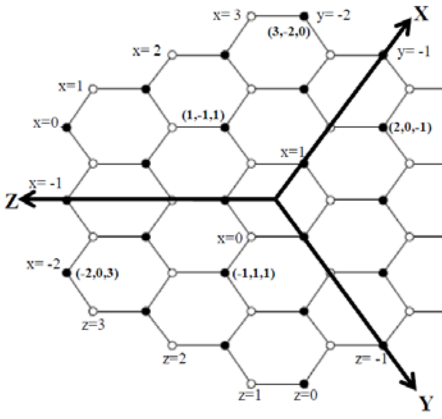

Figure 2: Coordinate System Of HM Network

The HM is a bipartite graph, each node has either white color or black color, which will be called a white node or a black node, the white node joins only to black nodes, and vice versa. Thus, each edge (link) in the network connects a white node with a black node. The summation of coordinates in each white node equals 2, whereas the coordinate's summation in each black node is 1. The adjacent nodes for any node (x , y, z) are determined as following, for a white node the adjacent nodes are (x-1 , y, z), (x, y-1, z), and (x, y, z-1), and for a black node the adjacent nodes are (x+1,y,z), (x,y+1,z), and (x,y,z+1) [29].

Links are bidirectional and numbered from 1 to 3, as illustrated in Figure 3, where a link that is parallel to X-axis is called link1, link2 is parallel to Y-axis, and link3 is parallel to Z-axis, the number of links in HM is 9t2 – 3t [28]. The distance between any two nodes p and p' is |x-x'| + |y-y'| + |z-z'|, two nodes are connected by an edge if and only if the distance is 1, and this occurs when the nodes differ in exactly one coordinate [11], [12].

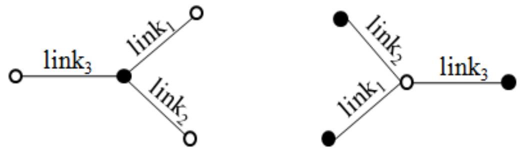

Figure 3: Links names in HM network

The number of nodes (n) in HMt is $6t^2$, the diameter is 4t-1, and the node degree is 3 [12], therefore, the network cost of the HMt is 3(4t-1). However, the network cost can be written in terms of n, by substitute t with $\sqrt{n/6}$, so the network cost is $12\sqrt{n/6}-3$, where $t = \sqrt{n/6}$ [11]. Comparing the HM with mesh connected computers; where the diameter is 2, and the node degree is 6, the HM has 25% smaller degree and 18.5% smaller diameter than the mesh connected computers with the same number of nodes. The network cost of mesh connected computers is 8, therefore the network cost of HM is approximately 40% lower than the cost of the mesh connected computer [12]. As consequence, the HM has less network cost than mesh connected computer network, and our objective in this research is to adapt HM networks and propose LEA for it, as will be discussed in the next Section.

4. THE PROPOSED ALGORITHM

4.1 Research Assumption

The research study assumes the following:

- Routers should run all the time.
- All communication links are bidirectional.
- Each node has a unique ID, which indicates its relative importance, the ID is computed by an internal function, and the ID value never changes during the algorithm execution.
- The leader failure is detected when the timeout exceeds without acknowledgement, the nodes which detect this failure start the election algorithm.
- The failed leader is excluded from the current election process by degrading its ID to 0.
- No link failure occurs during the election process.
- More than one node can run the election process concurrently; this takes place when a subset of nodes detects the failure.

Each node has the following variables:

a) Node ID: Unique ID.
b) Node position: Position of the node.
c) Leader ID and leader position.
d) Phase: Election phase and step.
e) Node ring: A value that determines to which ring that the node will belong.
f) Node state: Normal, candidate, leader.

If the failed leader comes back again, it sends a message to an adjacent node, checking the ID of the current leader, if it has higher ID than the current leader ID, an election process will be initiated by this node. This procedure is also done when a new node joins the network to substitute a crashed node.

4.2 Definitions

In this section, we introduce the essential terms that will be mentioned in the algorithm discussion; these terms help in understanding the algorithm.

1. *Node State*: while the algorithm is in progress, each node in the network can be in one of the following three states:
   - Normal: the network is normal, and this node does not participate in any election process.
   - Candidate: there is a leader failure, and this node participates in the current election process.
   - Leader: exactly one node has the leader state in a stable network, this state lost when the leader crashes.
2. *Position:* The position of a node determined according to its coordinates (x,y,z).
3. *Node Ring:* The authors suggest that the HM network consists of t equalized size overlapping rings, where t is the honeycomb size, i.e. number of rings = t. The ring number is determined depending on the node position. The ring number is in the range (t, t-1, …, 1), see Figure 4 which represents HM4 with four overlapping rings.
4. *Path:* The path represents the track that the election message will follow, this path composed of two links numbers, known as the path sequence, for example, 23.
5. *Ignore Node*: This node satisfies the condition of ignoring the inform message, the inform message is ignored either by candidate nodes or by a node where another node in its ring is informed with the leader failure and has the same election paths of the inform message initiator.

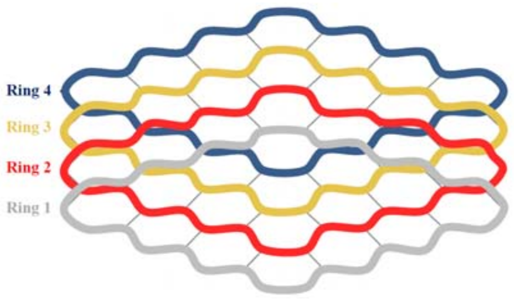

Figure 4: Rings In HM4

6. *Alteration Nodes:* Each node in this group alters the path of the received message by replacing a link number in the path sequence with another one. These nodes represent the zigzag chains, where z coordinate value is t, 1-t, 1, or 0, the summation of node coordinates is regarded in addition to the z value to replace the links in the path, the alteration nodes in HM3 are shown in Figure 5. For example, if any node with z=1-t receives election message with path sequence 31, it will alter the path to 23, or if it receives the message with path sequence 31, it will modify the path to 23. The main benefit of alteration nodes is to enforce the election message to go through a circular trajectory (ring).

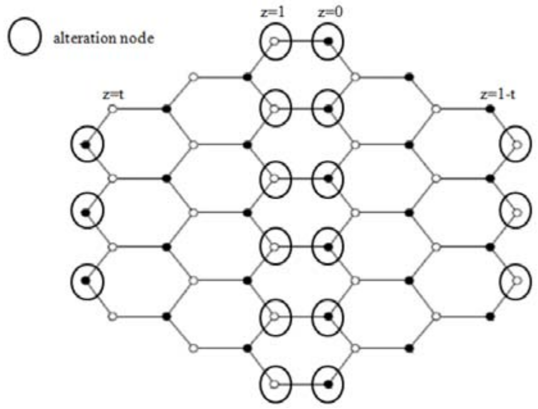

Figure 5: Alteration Nodes In HM3

7. *Messages* Types: The types of message that included in the proposed algorithm are:
   - Inform Message: A message that informs other nodes about leader failure composed of (initiator position, phase, step, ignore_node, path_1, path_2).
   - Election Message: A message that is composed of (initiator position, phase, step, path, ring_number, higher ID, the position of

higher ID). The election message goes through a predetermined sequence of links (path), and within the initiator's ring.

- Ring Result Message: similar to election message, but used in phase 3 to compare the nominee's leaders of all rings in the network.
- Leader Message: A message that contains the leader ID and position.

8. *Step:* The step is increased by one in the inform message and the election message in each time the message forwarded to the next node.
9. *Ring Accumulator Node:* A node with position (ring_number, - ring_number + 1, 1) in each ring is used to gather the result of election process. Each ring in the network will nominate a leader; this leader information is gathered by the ring accumulator node corresponding to that ring.
10. *Network Accumulator Node:* The node with position (1,0,1) in the network is used to select only one node as the new leader for the whole network. The node gathers the rings nominee leaders and selects the nominee leader with highest ID as the new leader. Afterwards, this node broadcasts leader messages to all nodes in the network announcing the new leader.

4.3 The Algorithm Phases

The proposed algorithm consists of several phases; phase1 involves detecting the leader failure and informing other nodes about the failure. Phase2 starts the election process to elect a leader for each ring in the network. In phase3, the algorithm concerns with selecting the new leader of the network. Finally, phase4 broadcasts leader messages to all nodes in the network. However, the Figure 6 represents main algorithm phases.

Phase1: The algorithm begins when a node detects the leader failure. The node changes its state to the candidate, checks its ring number, determines the election paths and inform message ignore condition based on its position, and sends two inform messages to t-1 nodes across links 1 and 2, the inform message guarantees that, all informed nodes will have the same election paths, Figure 7 summarizes phase1.

The informed node has the following properties: The same summation of the message initiator's x and y coordinates, and the same z-coordinate value of the message initiator.

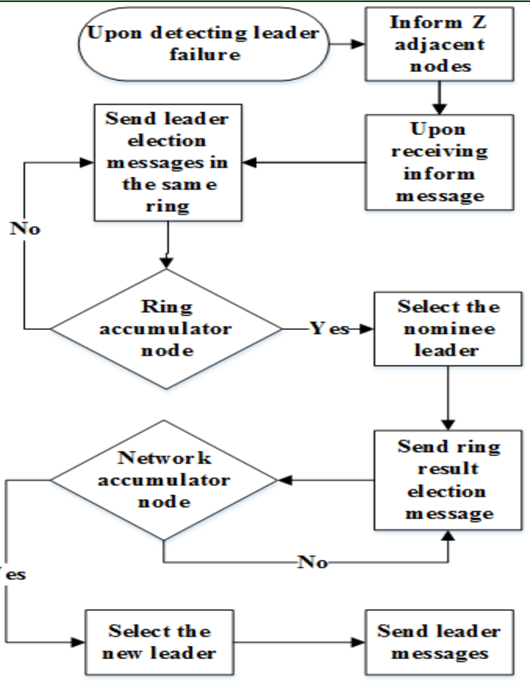
Upon detecting leader failure
Inform Z adjacent nodes
Upon receiving inform message
Send leader election messages in the same ring
No
Ring accumulator node
Yes
Select the nominee leader
Send ring result election message
Network accumulator node
No
Yes
Select the new leader
Send leader messages

Figure 6: Algorithm phases

Each candidate node starts phase2 by extracting the paths from the inform message, chooses its ID as higher ID and sends election messages to other nodes within its ring across the first link in each election path. Phase1 finishes when one inform message reaches an ignore node, and the another inform message reaches a node that does not have a link to next node in the z coordinate direction. Note that, if the state of the node is a candidate, the inform message will be immediately ignored.

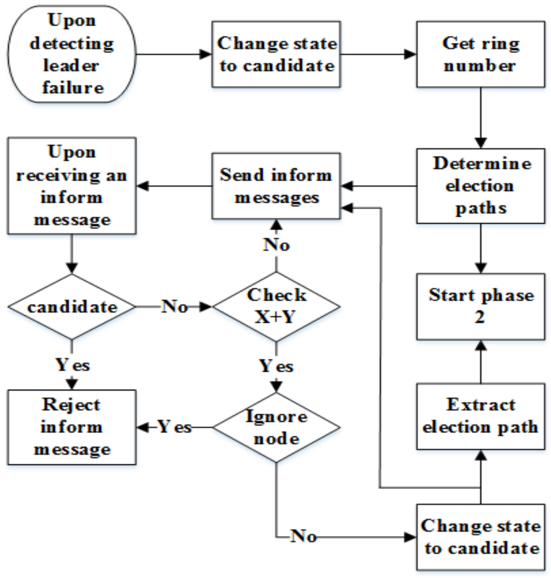

Figure 7: Phase1

Phase2: Each candidate node in phase one starts the election process by sending election messages in two different directions to all nodes within its ring. Every node that received the election message compares the received ID with its ID, selects the higher ID and passes the election message to next node after changing the path. The ring accumulator nodes which have positions (ring_number, -ring_number + 1, 1) gather the election result of each ring in the network; these nodes have the nominees leaders of all rings in the HM network, as illustrated in Figure 8.

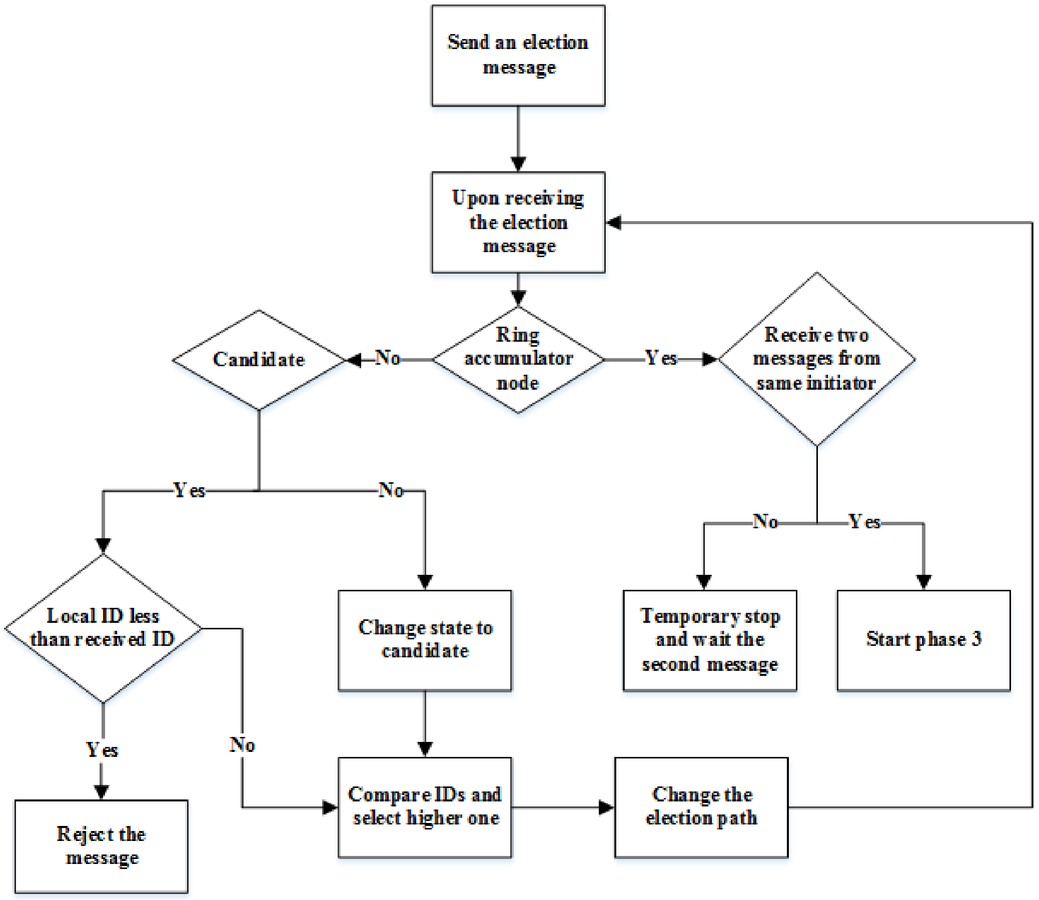
Send an election message
Upon receiving the election message
Ring accumulator node
No
Yes
Candidate
Receive two messages from same initiator
Yes
No
No
Yes
Local ID less than received ID
Change state to candidate
Temporary stop and wait the second message
Start phase 3
Yes
No
Reject the message
Compare IDs and select higher one
Change the election path

Figure 8: Phase2

During phase 2 each node receives the election message will check its ring number and the alteration node condition. The nodes that have the same ring number of the election message initiator are authorized to compare the received ID with its local ID, the nodes that differ than the ring number of the message initiator will pass the election message without comparing the IDs. The alteration node condition is checked to decide whether the path numbers will be only swapped, replaced without swapping, or replaced and swapped.

If the ring accumulator node that has the position (t,-t+1,1) receives two election messages belong to the same initiator, it selects the higher ID and starts phase 3 by sending ring result message across a path with sequence 12 to compare its nominee leader with all nominees leaders in the network.

Phase3: Figure 9 shows phase3, where the node with position (t, -t+1, 1) starts phase 3 by sending a ring result message across path: 12. As in phase1, the message sent to t-1 nodes, each node that has the same summation of message initiators' x, and y coordinates, will compare its local ID with received ID and select higher one, then it passes the message to next node.

If the phase of the receiving node does not equal to the phase of the incoming message, the message will suspend until the phase of the node becomes 3.This process continues until the message reaches the network accumulator node (1, 0,1), this node selects the node with highest ID to be the leader of the whole network and starts phase 4 by broadcasts leader messages to all nodes in the network.

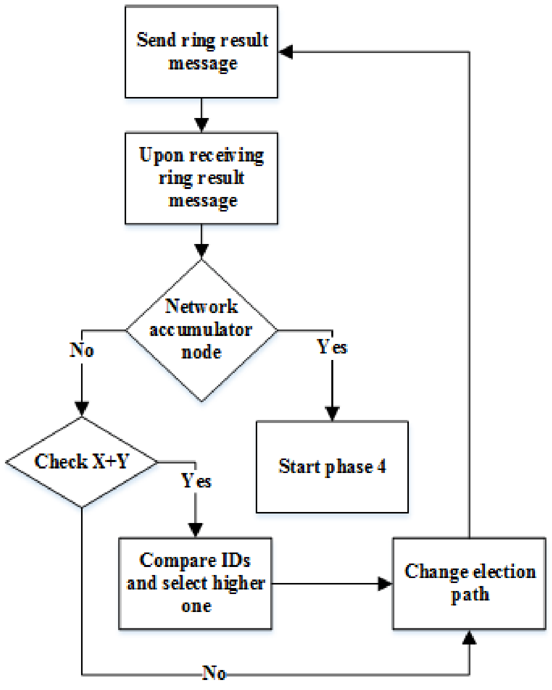

Figure 9: Phase3

Phase4: The network accumulator node (1, 0, 1), is aware of the new leader ID and position. It broadcasts the leader message to other nodes on all available links 1, 2 and3; every node receives the message will change its state to normal, update the leader information depending on the leader message, and pass the message to other adjacent nodes, see Figure 10.

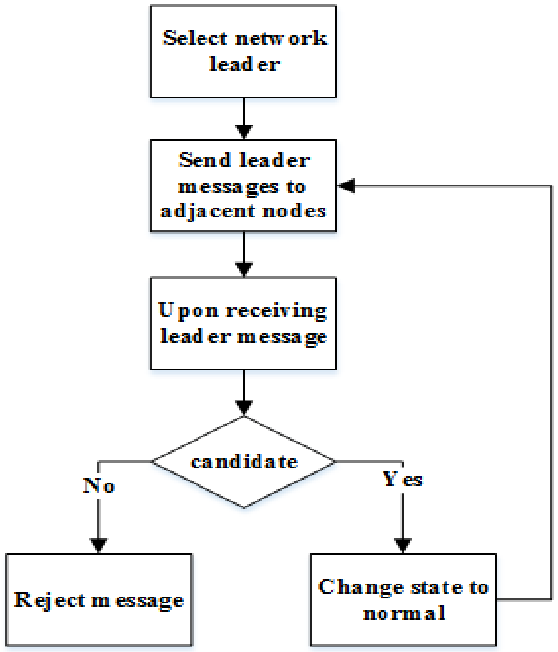

Figure 10: Phase4

4.4 Practical Example

To provide more understanding of the algorithm, consider the following example, where the honeycomb size (t) is 3, and the number of nodes is 54. Assume node A detects the leader failure, as shown in Figure 11 (a).Node A starts the algorithm by changing its state to the candidate and prepares inform messages to inform two other nodes which have the same x, and y coordinates summation in the same z coordinate. The inform message is sent on link1 and link 2, but link2 is not available so that the node A will send one inform message on link1 to node D. The inform message is as following ((-2,3,0), 1, 1, (---,0,---),13, 31).

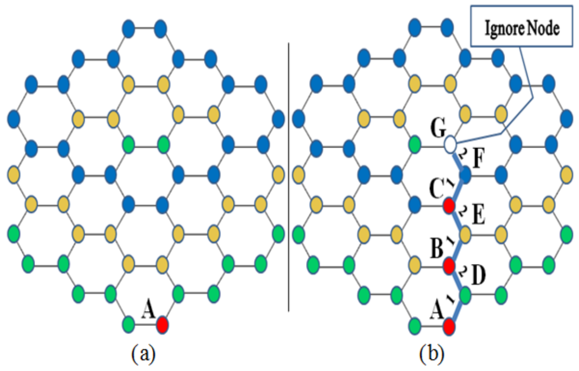

Figure 11: An Example For Illustrating Phase 1

Node A also starts phase2 and prepares two election messages. The first message ((-2,3,0), 2, 1, 13, 2, higher ID, (-2,3,0)) is sent on link1, while the second message is sent on link3, the second message has the same information of the first message, except for fourth value which is the path 31.When node D receives the inform message, it compares its x and y summation with the message initiator's x and y coordinates. The summation is different, so D will pass the inform message on link2 to node B. Node B has the same summation of the message initiators' x and y, the node B changes its state to candidate, checks its ring number, extracts the election paths from the inform message, and sends two election messages as node A did. Node B passes the inform message on link1 to next node, the Figure 11 (b) shows phase 1.

However, the inform message continues until it reaches node G. The node G has the same summation of x and y coordinates of the message initiator, it satisfies the ignoring node condition, where its y value is 0. The inform message will be discarded and deleted, here phase1 finishes, see Figure 11 (b).The nodes A, B, and C started phase2 by sending election messages to the next nodes on the first link in the two election paths, as in Figure 12 (a). The election messages go through a path as in Figure 12 (b), (c) and (d) until reaching the ring accumulator nodes with positions (ring_number, -ring_number + 1, 1) in each ring, these nodes are P, Q, and R in our example.

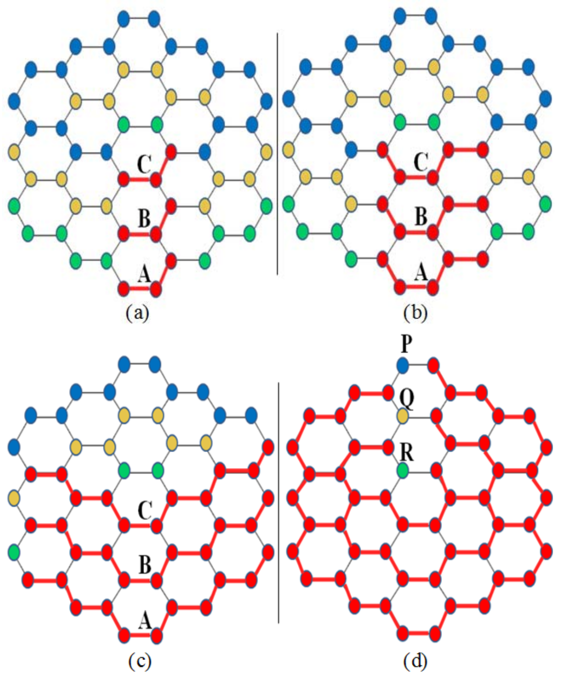

Figure 12: Phase 2 In The Example

When the node P which has position (t, -t+1, 1) receives two election messages belong to the same initiator from links 1 and 3, it selects the highest ID in its ring (its ring nominee leader) and starts phase 3 by sending ring result message to node Q as in Figure 13 (a). Node Q compares its ring nominee leader with the nominee leader of P's ring and selects the higher nominee leader. Node Q sends ring result message to node R to select the leader of the whole network, as shown in Figure 13 (b).

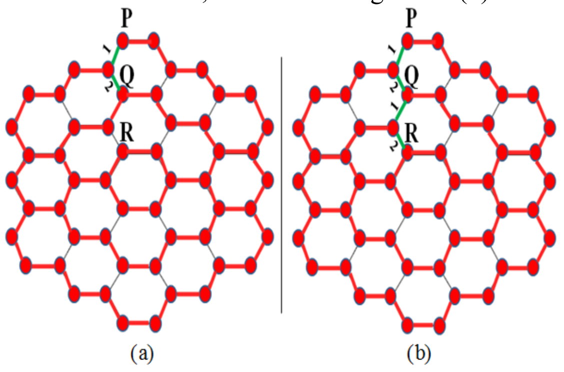

Figure 13: Phase 3 In The Example

The network accumulator node R starts phase 4 by broadcasting leader messages to all other nodes in the network to announce the new leader, Figure 14 (a), and (b) show phase 4.

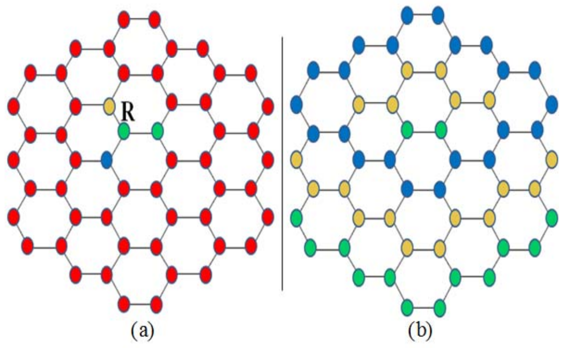

Figure 14: Phase 4 In The Example

5. PERFORMANCE EVALUATION

Leader election algorithms evaluated by computing number of time steps and messages needed to bring the system again to a stable state. The number of time steps computed by counting the steps that have been required to finish each phase, and therefore to complete the algorithm, while the number of messages computed by counting the messages that have been exchanged among all nodes in the network to elect a new leader [13]. However, in this section we present a mathematical analysis for our proposed LEA, the analysis includes two cases: best case, where one node detects the leader failure, and the worst case where all nodes detect the leader failure.

5.1 Best Case

The algorithm requires O(n) messages to elect a new leader in O($\sqrt{n}$) time steps.

5.1.1 Proof of message complexity

The overall number of messages needed by the algorithm is the summation of messages number in each phase, as follows:

Phase1: The node which detects the leader failure will send inform messages to t-1 nodes that have the same summation of x, and y coordinates in the same z coordinate, where t represents the size of the honeycomb. The node first sends at most two inform messages in two directions to two adjacent nodes; those adjacent have a different summation of x, and y, so each node that receives the inform message sends one inform message to next node, this process continues until t-1 nodes informed with the leader failure. To inform one node with leader failure we need two messages, but also we need two additional messages to reach the ignore node, so the total number of inform messages in phase one is 2(t-1) + 2= 2t, an additional message may be used depending on the node position.

Phase2: The candidate nodes in phase1 send election messages equal to its ring size, the ring size in HMt is 8t-2. The total number of election messages in t rings in this phase equals t(8t-2).

Phase3: The ring accumulator node that has coordinates (t, -t+1, 1) starts phase three by sending a ring result message to t-1 nodes that have the same summation of its x, and y to compare the leader ID of its ring with the leader ID in t-1 rings. As in phase1, we need two messages to reach the first node, for t-1 nodes we need 2t-2, in this phase, there is no ignore node. Therefore, we do not need two additional messages. The number of messages that is required until the message reached the network accumulator node (1, 0, 1) in this phase is 2t-2.

Phase4: The network accumulator node (1,0,1) broadcasts leader message to all adjacent nodes, it sends three messages. Each node receives the message will send messages across all links except the link it received the message from, each node has degree three will send two messages, while each node with degree two will send one message. The number of nodes with degree two is 6t, and the number of nodes with degree three is 6t2-6t, the first node sends one additional message, so this phase requires a number of messages equal $6t+2(6t^2-6t)+1=12t^2-6t+1$.

The total number of messages (number_bst(messages)) that are used by the proposed algorithm is computed by adding the number of messages in each phase, as following:

$$\text{number\_bst(messages)} = 20t^2-4t-1+(1) \qquad (2)$$

By replacing t with $\sqrt{n/6}$, the result shows that the algorithm requires O(n) messages.

5.1.2 Proof of time complexity

The overall number of time steps computed by adding a number of time steps in each phase, as follows:

Phase1: The node that detects the failure sends two messages in two directions in one step, next node sends one message in one step, this process continues until t-1 nodes informed with the failure. To deliver the election message to one node we need two timesteps, for t-1 nodes we need 2t-2 time steps, two additional time steps required to deliver the message to ignore node. Depending on the node position, some nodes require fewer time steps, the number of time steps in this phase is at least t time steps, and at most 2t time steps, we consider the node that requires the largest time steps, so the algorithm needs at most 2t-2+2= 2t time steps in phase1.

Phase2: In phase 2, the number of time steps depends on the position of the node that detects the failure. In the first case, the node at the middle of a ring detects the failure, if such node detects the failure, then the number of time steps equals 4t-1.The node sends two election messages in two directions; each message needs 4t-1 steps to reach the ring accumulator node (ring_number, -ring_number+1, 1).In the second case, the ring accumulator node (ring_number, -ring_number+1, 1) detects the failure; it needs 8t-2 time steps to complete the ring.

Phase3: the ring accumulator node (t, -t+1, 1) sends a ring result message to t-1 nodes, to deliver the message to next node we need two timesteps, the message is stopped by the network accumulator node (1, 0, 1), for t-1 nodes the algorithm requires 2t-2 time steps.

Phase4: In HM1, we need three steps to broadcast the leader message to all nodes in the network, in HM2 we need five steps, and in HM3 the number of steps needed is 7. It is easy to observe that HMt needs 2t+1 messages to finish this phase.

The total number of time steps regarding the first case in best case shown in (3)

$$\text{number\_bst(timesteps)} = 10t - 2 \qquad (3)$$

While the total number of time steps regarding the second case in best case shown in (4)

$$\text{number\_bst(timesteps)} = 14t - 3 \qquad (4)$$

The results in (3) and (4) verifies that $O(\sqrt{n})$ timesteps required to complete the algorithm.

5.2 Worst Case

The algorithm requires $O(n^{1.5})$ messages to elect a new leader in $O(\sqrt{n})$ time steps.

5.2.1 Proof of message complexity

Phase1: All the nodes in the network detect the leader failure at the same time. The nodes begin the algorithm by sending at most two inform messages in theirs' z coordinates; the inform message will be ignored immediately since all nodes in the network are candidates. There are 4t nodes that have one link in the same z coordinate value; these nodes will send 4t messages, and other nodes $6t^2$ - 4t will send $2(6t^2 - 4t)$ messages, the number of messages in this phase is $4t+2(6t^2-4t)$.

Phase2: The worst case happens when the ID's of all nodes in a ring are ordered, the first node sends two messages, the second node sends three messages, the third node sends four messages, the node 8t-3 sends 8t-2 messages, while The node 8t-2 sends 8t-2 messages. Consequently, the number of the total message in one ring is as following:

$$(\sum_{k=1}^{8t-2}(k+1))-1 \qquad (5)$$

For t rings, the number of messages is computed by multiplying t with (5), as represented by (6)

$$t((\sum_{k=1}^{8t-2}(k+1))-1) \qquad (6)$$

Phase3: As in the best case, needs 2t-2 messages.

Phase4: requires $12t^2$-6t +1 messages to finish, as in the best case.

The total messages number needed by the algorithm in worst case is the summation of the number of messages in each phase, the number of messages is shown in the (7).

$$\text{number\_wrst(messages)}=12t^2 - 4t + t((\sum_{k=1}^{8t-2}(k+1))-1) + 2t-2 + 12t^2-6t+1 \qquad (7)$$

By substituting t with $\sqrt{n/6}$, the algorithm needs $O(n^{1.5})$ messages.

5.2.2 Proof of time complexity

Phase1: In this case, all nodes detect the failure simultaneously; each node sets its state to the candidate and sends two inform messages to its adjacent in same z coordinate. Therefore, the time steps needed in this phase is one, since candidate nodes will ignore the inform message.

Phase2: When all nodes detect the failure, only the election message with highest ID will complete the ring, the ring accumulator node with position (ring_number,-ring_number +1, 1), needs 8t-2 steps to complete the ring.

Phase3: this phase needs 2t-2 time steps to finish, as in the best case.

Phase4: phase4 needs 2t+1 time steps.

Consequently, the total number of timesteps is represented by the following equation:

$$\text{number\_wrst(timesteps)} = 12t - 2 \qquad (8)$$

The result in (8) proofs that, the algorithm requires $O(\sqrt{n})$ timesteps.

6. CONCLUSION

The research paper proposes, for the first time, a new LEA to solve the leader failure problem in HM networks. The performance of the proposed algorithm is evaluated by calculating the number of required messages and time steps to complete the algorithm. The evaluation included two cases, best case and worst case. In the best case, leader failure is detected by only one node, whereas in the worst case, the failure detected by all nodes in the network. The algorithm requires O(n) messages to elect a new leader in $O(\sqrt{n})$ time steps in the best case. While in the worst case, it requires $O(n^{1.5})$ message in $O(\sqrt{n})$ time steps, where n represents the number of nodes in the network.

There are some challenges in our proposed algorithm that remains for further investigations, such as designing fault tolerant algorithms. In future works, the proposed algorithm for HM can be improved to deal with one link and multi-link failure in the network; the algorithm can also be extended to solve the leader election in three-dimensional HM.

REFERENCES:

[1] T. Vergnaud, L. Pautet, and F. Kordon, "Using the AADL to describe distributed applications from middleware to software components," in *International Conference on Reliable Software Technologies*, 2005, pp. 67-78.

[2] M. Refai, "Leader Election Algorithms in Torus and Hypercube Networks Comparisons and Survey," *International Journal of Computer Science and Mobile Computing (IJCSMC),* vol. 4, pp. 102-111, 2015.

[3] A. S. Tanenbaum and M. Van Steen, *Distributed systems*: Prentice-Hall, 2007.

[4] M. EffatParvar, N. Yazdani, M. EffatParvar, A. Dadlani, and A. Khonsari, "Improved algorithms for leader election in distributed systems," in *Computer engineering and technology (ICCET), 2010 2nd International Conference on*, 2010, pp. V2-6-V2-10.

[5] M. Mirakhorli, A. A. Sharifloo, and M. Abbaspour, "A novel method for leader election algorithm," in *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on*, 2007, pp. 452-456.

[6] P. BeaulahSoundarabai, J. Thriveni, K. Venugopal, and L. Patnaik, "AN IMPROVED LEADER ELECTION ALGORITHM FOR DISTRIBUTED SYSTEMS," *International Journal of Next-Generation Networks,* vol. 5, p. 21, 2013.

[7] M. Vojdani and Y. Taj, "DR: Divided Ring Leader Election Algorithm," in *International Conference on Algorithms and Architectures for Parallel Processing*, 2009, pp. 90-99.

[8] S. A. Hussain, N. A. Khan, A. Sadiq, and F. Ahmad, "Simulation, modeling and analysis of master node election algorithm based on signal strength for VANETs through Colored Petri nets," *Neural Computing and Applications,* pp. 1-17.

[9] A. Qatawneh, "Embedding Hex-Cells into Tree-Hypercube Networks," *International Journal of Computer Science Issues (IJCSI),* vol. 10, May 2013 2013.

[10] M. Rahman, Y. Inoguchi, F. A. Faisal, and M. K. Kundu, "Symmetric and folded tori connected torus network," *Journal of Networks,* vol. 6, pp. 26-35, 2011.

[11] A. W. Yin, T. C. Xu, P. Liljeberg, and H. Tenhunen, "Explorations of honeycomb topologies for network-on-chip," in *Network and Parallel Computing, 2009. NPC'09. Sixth IFIP International Conference on*, 2009, pp. 73-79.

[12] I. Stojmenovic, "Honeycomb networks: Topological properties and communication algorithms," *IEEE Transactions on Parallel and Distributed Systems,* vol. 8, pp. 1036-1042, 1997.

[13] M. Refai, I. Aloqaily, and A. Alhamori, "Leader Election Algorithm in 3D Torus Networks with the Presence of One Link Failure," *World of Computer Science and Information Technology Journal (WCSIT),* vol. 2, pp. 90-97, 2012.

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

[15] E. Chang and R. Roberts, "An improved algorithm for decentralized extrema-finding in circular configurations of processes," *Communications of the ACM,* vol. 22, pp. 281-283, 1979.

[16] D. S. Hirschberg and J. B. Sinclair, "Decentralized extrema-finding in circular configurations of processors," *Communications of the ACM,* vol. 23, pp. 627-628, 1980.

[17] H. Garcia-Molina, "Elections in a distributed computing system," *IEEE transactions on Computers,* vol. 100, pp. 48-59, 1982.

[18] S. Basu, "An efficient approach of election algorithm in distributed systems," *Indian Journal of Computer Science and Engineering (IJCSE),* vol. 2, pp. 16-21, 2011.

[19] M. Gholipour, M. Kordafshari, M. Jahanshahi, and A. Rahmani, "A new approach for election algorithm in distributed systems," in *Communication Theory, Reliability, and Quality of Service, 2009. CTRQ'09. Second International Conference on*, 2009, pp. 70-74.

[20] Q. E. K. Mamun, S. M. Masum, and M. A. R. Mustafa, "Modified bully algorithm for electing coordinator in distributed systems," *WSEAS Transactions on Computers,* vol. 3, pp. 948-953, 2004.

[21] M. B. Yassein, N. Alslaity, and A. Alwidian, "An Efficient Overhead-Aware Leader Election Algorithm for Distributed Systems," *International Journal of Computer Applications,* vol. 49, 2012.

[22] M. Seperhi and M. Godarzi, "Leader election algorithm using heap structure," in *12th WSEAS Int. l Conf. on Computers*, 2008, pp. 23-25.

[23] M. Refai, "A new leader Election Algorithm in Hypercube Networks," in *Symposium Proceedings Volume II Computer Science & Engineering and Electrical & Electronics Engineering, European University of Lefke, North Cyprus, PP*, 2006.

[24] M. Refai, A. Sharieh, and F. Alshammari, "Leader Election Algorithm in 2D Torus Networks with the Presence of One Link Failure," *The International Arab Journal of Information Technology (IAJIT),* vol. 7, pp. 105-114, 2010.

[25] K. A. Mohammed Refai, Ibrahim Aloqaily, "Leader Election Algorithm in the Honeycomb Torus Networks," *Journal of Engineering Technology,* vol. 5, p. 21, January 2016.

[26] M. Refai, K. Alkheder, Y. Al-Raba'nah, and M. Alauthman, "Dynamic Solution for Leader Failure in the Honeycomb Torus Network," *Journal of Engineering Technology (ISSN: 0747-9964),* vol. 6, pp. 97-112, 2017.

[27] 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)*, 2012, pp. 1716-1720.

[28] S. Hayat and M. Imran, "Computation of topological indices of certain networks," *Applied Mathematics*

and Computation, vol. 240, pp. 213-228, 2014.

[29] J.-H. Seo, H. Sim, D.-H. Park, J.-W. Park, and Y.-S. Lee, "One-to-one embedding between honeycomb mesh and Petersen-torus networks," *Sensors,* vol. 11, pp. 1959-1971, 2011.

[30] V. C. Sharmila and A. George, "A Survey on Area Planning for Heterogeneous Networks," *International Journal on Applications of Graph Theory in Wireless Ad Hoc Networks and Sensor Networks,* vol. 5, p. 11, 2013.

[31] D. Xu, J. Fan, X. Jia, S. Zhang, and X. Wang, "Hamiltonian properties of honeycomb meshes," *Information Sciences,* vol. 240, pp. 184-190, 2013.

[32] X. Yang, Y. Y. Tang, and J. Cao, "Embedding torus in hexagonal honeycomb torus," *IET Computers & Digital Techniques,* vol. 2, pp. 86-93, 2008.

[33] W. Zhang, X. Yang, Y. Liang, and Z. Huang, "Routing algorithms in honeycomb meshes," *International Journal of Parallel, Emergent and Distributed Systems,* vol. 24, pp. 367-382, 2009.