Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112

ORIGINAL ARTICLE

Dynamic Solution for Leader Failure in the Honeycomb Torus Network

Mohammed Refai[1], Khalid Alkheder[2], Yousef Al-Raba'nah[3] and Mohammad Alauthman[4]

[1,2,3,4]Department of Computer Science, Faculty of Information Technology, Zarqa University, Jordan

Abstract:
Leader election algorithm (LEA) is the process of electing a new leader (coordinator) to substitute the failed leader in the distributed systems that depend on the central control. The LEA is a distributed algorithm over all nodes; it begins when one node or more detect the leader failure and terminates when one node is electing to be the new leader. This paper, proposes a dynamic LEA to solve the leader failure in honeycomb torus networks with the presence of intermittent link failure. The proposed algorithm uses various detours to pass over link failure to complete LEA. The proposed algorithm enhanced a previous algorithm that solved leader failure in honeycomb torus networks without link failure. When the process terminated, the network returned to a stable state with one node as a leader and the rest of nodes aware of the new leader. The paper evaluated the performance of the proposed algorithm by calculating time steps and the number of messages needed to elect a new leader and found that the number of messages equal to $O(n^{1.5})$ within $O(\sqrt{N})$ time steps.

Keywords: Leader election algorithm, Honeycomb torus, intermittent link failure, dynamic algorithm.

1. Introduction

After publishing the first algorithm to solve leader failure in Honeycomb Torus network in [18], this paper comes to enhance that algorithm by solving the leader failure problem despite the presence of intermittent link failure. The new algorithm can react to links disconnection status during execution time, so it can be called dynamic algorithm.

The LEA aims to elect a node as the centralised controller (coordinator) for distributed systems that require one node to manage the activities of the system ([4], [5], [7]). LEA is a program distributed over all nodes or processes; it begins when one node or more detect the leader failure, and terminates when all processes aware of

ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112
ORIGINAL ARTICLE

the newly elected leader. The function of LEA is to move the system from an initial state, where all nodes are in the same computation state (state with no leader), to a new state in which one of these nodes is a leader and all other nodes aware of this leader ([15], [21]).

The honeycomb architecture applied to different applications such as in wireless networks, where the base stations arranged in a honeycomb network [20], the representation of benzenoid hydrocarbons in chemistry, applications in computer graphics and applications in image processing [11].The honeycomb torus obtained by adding wrap around edges to the honeycomb mesh to achieve edge and vertex symmetry.

The honeycomb topology is one of the principal topologies that need an algorithm to solve the leader failure problem with the presence of link failure. In this paper, we will suggest a new dynamic LEA to solve this problem. We will provide the mathematical analysis for the time steps and a number of messages needed to complete the algorithm. The proposed algorithm will take in consideration if one node detects the failure simple case, or any subset of nodes up to all nodes in the worst case.

This paper organised as follows. Section two discusses related work. Section three presents the honeycomb torus model structure and properties. Section four explains the proposed algorithm. Section five evaluates the performance of the algorithm. In section six conclusion and future work are presented.

2. Related Works

The LEAs started with the ring and complete topology to solve the leader failure when occurs ([2] and [21]). After that, the researchers studied meshes, hypercube and tree topologies ([16],[17]). To date, these topologies and wireless networks still studied. Different election algorithms solve the leader failure such as bully algorithm and ring algorithm. Also, the LEA applied in wireless networks. The researchers proposed several leader election algorithms in different network topologies ([1], [2],[16], [17], [18], and [19]). The LEAs are varying according to the type of topology, node identity [7], communication mechanism used and transmission media ([3],[6], [12], [13] and [22]).

Molina in [14] presented an algorithm to address the failure of leader for a complete network. When a process detects the failure event, it sends an election message with the higher number of all processes. If one node has a number greater

than the received number it becomes the leader, otherwise the node that detects the failure event becomes the leader. This algorithm called the Bully Algorithm.

Another LEA is proposed to solve the leader failure in asynchronous complete (fully-connected) distributed networks [1]. The communication channels between reliable processors may fail intermittently before or during the execution of the algorithm, and it cannot be detected because the network is asynchronous. If (N) is the number of processors in the network and (F) is the maximum number of faulty channels incident on each processor, and $F \leq (n-1)/2$, then the number of messages needed to elect the new leader is $O(n^2+nF^2)$.

In [9] the researcher proposed a new algorithm that is more efficient than a bully algorithm, by reducing the number of redundant election messages and time complexity. The proposed algorithm requires, in the best case (n-1) messages, while it requires 2(n-1) messages in the worst case.

There are different LEAs proposed for 2D torus topology, 3D torus topology and hypercube topology with the presence of one link failure ([16] , [17] and [18]). In [19] they presented leader election algorithm for a group of nodes connected by a tree where the algorithm based on heap structure.

In [8] Researchers aims to minimise the number of election messages. A new strategy proposed for the election algorithms where it based on choosing a leader and candidate nodes as alternatives to the leader node.

Researchers in [18] (required for this paper) proposed a LEA to solve leader failure in honeycomb torus.

3. Honeycomb Torus

The honeycomb torus network is different from honeycomb mesh in adding wrap connections around edges to achieve edge and vertex symmetry. Figure (1) shows a 3d honeycomb torus network. In honeycomb torus network, interconnection topology is a graph with $N = 6D^2$ vertices, where D is the dimension of the honeycomb torus network. The number of edges between all nodes is $9D^2$ edges [20].

Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112

ORIGINAL ARTICLE

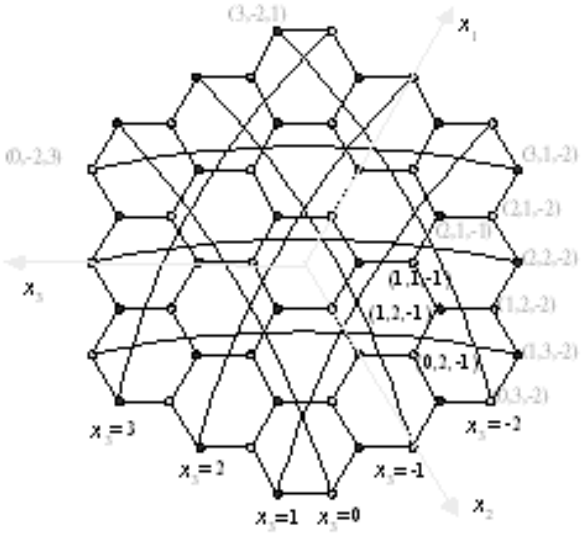


Fig. 1 3D Honeycomb Torus Network [18].

The honeycomb torus network with level (D), contains D-rings parallel for each axis, for example in 3D honeycomb torus network, there are three rings parallel to X1, and three rings parallel to X2, and three rings parallel to X3. as shown in figure (1), where the black nodes represent the end of ring for each axis.

A node can send or receive messages simultaneously and use all communication ports, where the node can send the message to all of its neighbours simultaneously.

The network uses two types of nodes, according to the summation of coordinates X1, X2 and X3 as shown in figure (2) where the summation of white and black nodes is equal to one and two respectively.

Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112
ORIGINAL ARTICLE

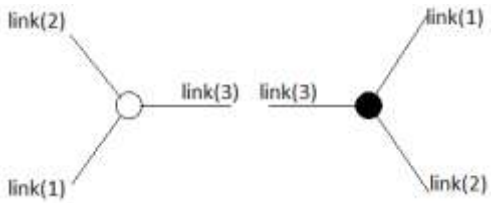


Fig. 2 Node links [18].

In this research paper, we use three level honeycomb torus network multicomputer model, that composed of N nodes, labelled 0, 1, 2… N-1.

This paper assumes the following:

- Bidirectional communication links.
- Routers should work all the time even with faulty node because the fault is in leader properties.
- Leader node could fail due to different reasons which lead to reach the threshold leader failure value.
- Nodes can detect this failure when the timeout exceed without acknowledgement. A node that detects this failure starts the election algorithm.
- Each node calculates a weight that defines its relative importance. This weight represented by distinguishing identification (ID) for each node. In LEA node compares its ID with the weight of other nodes that it has received and propagated the maximum weight.
- When the leader node crashes, its ID degrades to 0. So, it cannot win the election.

More assumptions and node variables use in this paper described in [18].

4. Proposed solution

This section describes how LEA in honeycomb behaves when a link failure occurs through executing time. Before that it presents briefly the main algorithm which found in details in [18]. The algorithm was composed of three phases, as follows:

a) Phase One: the node that detects leader failure (initiator) checks its position and sends an election messages toward link1, if it located on one of the rings of X1-axis, link2 if it is located on one of the rings of X2-axis and link3 if it located

on one of the rings of X3-axis and the node becomes in candidate state. The election message contains different paths according to the coordinates of the initiator. The node that receives the election message reads the paths of the message and compares the received ID with local ID to select the higher ID, after that the node becomes in candidate state and checks its position, the node sends election messages to other rings depends on its position except the link it received the election message from it. If the node is already in candidate state and the received ID higher than candidate ID it passes the election message according to its path without checking its position. This process will continue until the nodes that located on the end of rings receives two election messages for the same node or receive one election message where the number of steps in the message is 4D. The results will be in specific nodes that explained in [18].

b) Phase Two: when the node (1-D,0, D) finish phase one it sends election message through link 2 and compares between the candidate ID of the nodes that are located on the end of rings and have finished phase one.

c) Phase Three: when the election message arrives at the last node in phase two the new leader becomes known to the last node in phase two, after that it broadcasts the information about the new leader to all nodes by sending a leader message.

The previous algorithm was asynchronous static, proposed to solve leader failure only. In this algorithm, a dynamic synchronous way is used. When a node receives an Election message, it sends an acknowledgement (ACK) message toward the node that sent the election message to guarantee the message arrival to destination.

Each node in phase one sends an election message; it waits for a period for the ACK message from the next node to guarantee that the message received without any problem. If it did not receive an ACK message, it prepares link failure message which contains the election message itself and sequence of links to arrive the election message to the next node in the ring. For example, figure (3) shows that the link (3) failed and the message cannot reach the next node due to this failure. So, the "link failure" message will be initiated and sent according to certain sequence to arrive at the next node in the ring of X3 to continue the election process. The algorithm deals with the link failure in phase one as shown in Table 1.

Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112
ORIGINAL ARTICLE

Table (1): The detours in phase one

Det #	Link failure	Detour Routing
1	Link1 and x1 is od	The link-failure message follows the sequence 23123 one time
2	Link1 and x1 is even	The link-failure message follows the sequence 32132 one time
3	Link2 and x2 is odd	The link-failure message follows the sequence 31231 one time
4	Link2 and x2 is even	The link-failure message follows th sequence13213one time
5	Link3 and x3 is odd	The link-failure message follows the sequence 12312 one time
6	Link3 and x3 is even	The link-failure message follows the sequence 21321 one time

For the previous example in figure (3) the node (a) does not receive an Ack message from node (b). The coordinates of the node (a) are (0,1,1), and x3 is odd, so the node (a) sends the link failure message according to the sequence (12312) to arrive it to the node (b) to continue the election process.

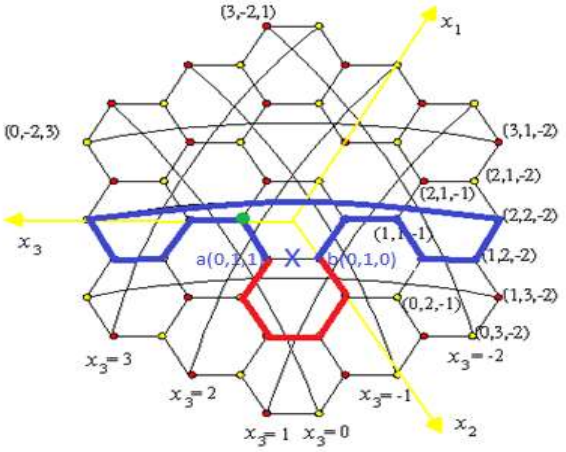


Fig.3 Link failure in phase one

Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112
ORIGINAL ARTICLE

In phase two any node that sends an election message, it also waits for a period to receive the ACK message from the next node. If it did not receive the ACK message, it sends failure message according to the Table (2).

Table (2): The detours in phase two

Det	Link failure	Detour Routing
1	Link1	The link-failure message follows the sequence 23123 one time
2	Link2	The link-failure message follows the sequence 31231 one time
3	Link3	The link-failure message follows the sequence 12312 one time

Figure (4) shows an example of link failure in phase two where node (a) sends an election message to node (b) with the sequence (12) but it did not receive an ACK message from node (b) due to the link (2) failure, node (a) will send link failure message according to the sequence (31231) to arrive the election message to node (b) to continue the election process.

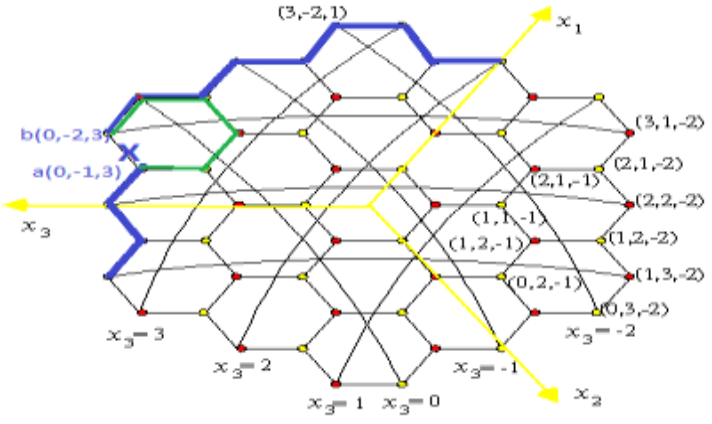


Fig.4 Link failure in phase two

Phase three solves the link failure between any two nodes, so if the candidate node did not receive the leader message from one link because it failed, it receives the leader message from the other links.

Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112
ORIGINAL ARTICLE

5. Performance evaluation

In this section, we evaluate the algorithm by calculating the number of time steps and messages in the simple case when one node only detects the leader failure or worst case when all nodes detect the leader failure.

5.1 Simple case

In this case, one node detects the leader failure. Number of messages and time steps that required completing LEA is in the following subsections.

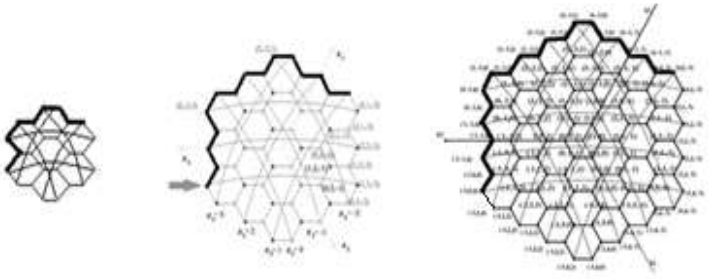
5.1.1 Number of messages:

Phase one: The initiator that detects the failure sends at most three messages (message 1, message 2 and message 3), if it located on three rings, and each node, when receives the election message, sends at most two messages because it does not send election message toward the link it received the message from. So, the number of messages (NOM) needed for this phase given in equation (1) where (N) is the number of nodes in the honeycomb torus network.

$$NOM(1) = 2N+1 \qquad (1)$$

Phase two: Figure (5) shows the path of election message in phase two for different honeycomb torus networks (2D, 3D and4D), the number of election messages needed to complete phase two in level two is (8) messages where the number of nodes in the network is (24) nodes, while in level three it needs (14) messages where the number of nodes in the network is (54) nodes, Also the number of election messages in level four is (20) messages where the number of nodes in the network is (96) nodes. So, for any honeycomb torus network the number of messages required for phase two is calculated by the following equation:

$$NOM(2) = 6\sqrt{N/6} - 4 \qquad (2)$$



(a) (b) (c)

Fig. 5 Phase two for 2D, 3D and 4D honeycomb torus network

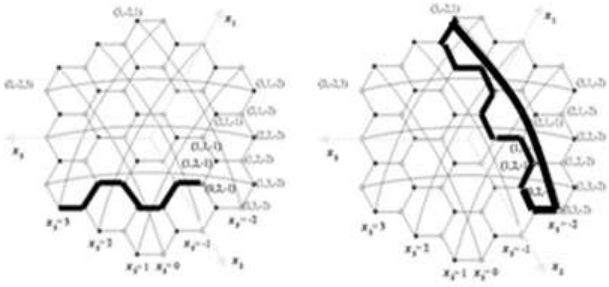
Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112
ORIGINAL ARTICLE

Phase three: The first node in phase three sends three leader messages toward link 1, 2 and 3. When the other nodes receive the leader message from one link, it resends two leader messages toward the other two links. This means that all nodes will send 2N messages except the first node in phase three where it will send three messages. So, the total number of messages needed to complete phase three is the same as equation (1).
The algorithm requires five messages to tolerate the link failure between two nodes. So, the total number of messages in all phases NOM (a) is:

$$NOM\ (a) = (2N+1) + (6\times \sqrt{N/6} - 4) + (2N+1) + 5 = O\ (N) \qquad (3)$$

5.1.2 Number of time steps:

Phase one: the number of time steps depends on the position of initiator that starts the election process if the initiator locates on three rings it needs only $4\sqrt{N/6}$ Time steps to accomplish phase one because all nodes execute the LEA in parallel, but if the imitator locates on one or two rings the number of time steps will change. Figure(6) shows the maximum number of time steps needed to complete phase one when the imitator locates on one ring when a node detects the failure; it needs $(4\sqrt{N/6}- 5)$ time steps to send the election message to the nearest node in the last ring (0,2,-1) as shown in figure (6-a) .the node (0,2,-1) needs at maximum $(4\sqrt{N/6})$ time steps to complete phase one as illustrated in figure (6-b).



(a) (b)

Fig.6 Maximum number of time steps in phase one

The total number of time steps (TS) needed in phase one is:

$$TS\ (1) = (4 \times \sqrt{N/6}-5) + (4 \times \sqrt{N/6}) \qquad (4)$$

Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112
ORIGINAL ARTICLE

Phase two: The number of time steps is equal to the number of messages in phase two. So the number of time steps needed to complete phase two is the same as equation (5).
Phase three: The number of time steps is equal to the number of time steps in the ring divided by 2 because of the nature of the topology where there are wraps around the edges, for example in 3D honeycomb torus the last node in phase two needs (6) time steps to broadcast the leader message to all nodes as shown in figure (7). So the number of time steps needed to complete phase three in the honeycomb torus network is:

$$TS(3) = 2\sqrt{N/6} \qquad (5)$$

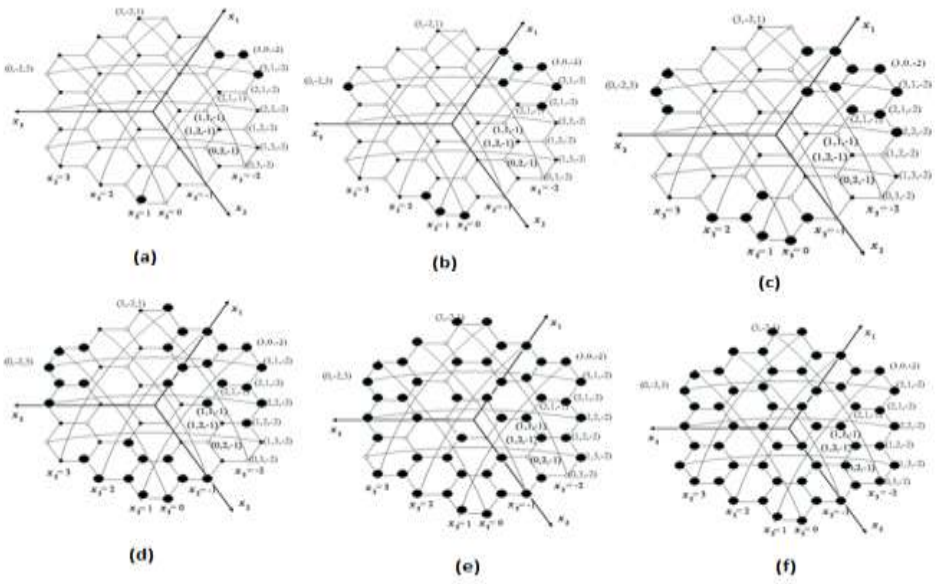


Fig.7 Phase three in 3D honeycomb torus network where it needs (6) time steps for broadcasting leader message.

The algorithm needs five steps to tolerate the link failure between two nodes. So, the total number of time steps TS (a) in all phases is:

$$TS\,(a) = ((4\times \sqrt{\frac{N}{6}}-5)+ (4\times\sqrt{\frac{N}{6}}))+(6\times \sqrt{\frac{N}{6}}-4) +(2\times \sqrt{\frac{N}{6}}) + 5=O(\sqrt{N}) \qquad (6)$$

Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112
ORIGINAL ARTICLE

5.2 Worst case

In this case, subset of nodes up to N-1 nodes detects the leader failure. Number of messages and time steps that required completing LEA is in the following subsections.

5.2.1 Number of messages

Phase one: the number of messages depends on the ID numbers of the nodes when the message faces a node with higher ID, the node will pass it to the next nodes, but if message faces a node with lower ID, the message will stop. To calculate the number of messages in the worst case, we will assume the identifiers of nodes arranged in ascending order as shown in figure (8).

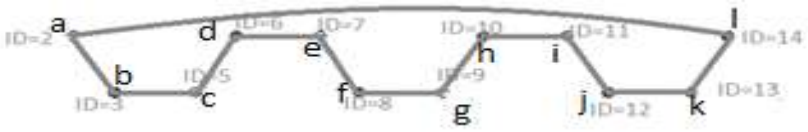


Fig.8 Arranging the identifiers in the ring

In this case, when all nodes detect the leader failure, it sends election message in two directions, when the node receives election message it compares the received ID with its ID. If the received ID higher than local ID, the node will pass the message to other nodes, otherwise the message will be neglected. In figure (8) the node (a) sends message to node (b), but node (b) will neglect the message because it has lower ID than the local ID for node (b). Also during this time node (b) will send a message to the node (c) and node (c) will neglect the message for the same reason. This process will be the same for the nodes (d, e, f, g, h, I, j and k). Node (l) will send a message to the node (a) which represents the end of the ring, and it will stop. The algorithm needs (4D) to do these procedures.

Moreover, in each ring when the node sends election message toward the other direction where each node has lower ID than the received ID, the node will pass the message to the next node. In step one node (b) sends message to node (a), node (c) sends message to node(b),node (d) sends message to node(c), node (e) sends message to node(d), node (f) sends message to node(e), node (g) sends message to node(f), node (h) sends message to node(g), node (i) sends message to node(h), node (j) sends message to node(i), node (k) sends message to node(j) and node (l) sends message to node(k). All messages will arrive at node (a) because each message will face a node which has lower ID, this means it requires one message from node (b) to node (a),two messages from node (c) to node (a),three messages from node (d) to node (a),four messages from node (e) to node (a) and so on. So, the total number of messages for each ring given in equation (7):

Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112
ORIGINAL ARTICLE

$1+2+3+4+\ldots+4D=\sum_0^{4D} i$ (7)

The number of rings in the honeycomb torus is 3D. So, the total number of messages for all rings in phase one is:

NOM (1) = $(4D+\sum_0^{4D} i) \times (3D) = O(D3)$ (8)

Phase two: The number of messages for phase two is the same as the number of messages in a simple case which is calculated by the equation (2).

Phase three: The number of messages for phase two is same as the number of messages in a simple case which is calculated by the equation (4).

The algorithm needs five messages to tolerate the link failure between two nodes. So, the total number of messages in all phases is:

NOM (a) = $((4D +\sum_0^{4D} i) \times (3D) + (6\times\sqrt{\frac{N}{6}} -4) + (2N+1)$ (9)

After substituting $D = \sqrt{N/6}$ in equation (9) the total number of messages in phase one is O ($N^{1.5}$).

5.2.2 Number of time steps:

Phase one: when all nodes detect the leader failure, each ring needs the same number of time steps to complete this phase which is equal the number of nodes in each ring. So, the number of time steps required for phase one is calculated by the following equation:

$$TS\ (1) = 4\sqrt{\frac{N}{6}} \qquad (10)$$

Phase two: The number of time steps is equal to the number of time steps in phase two of the simple case, so the number of time steps needed to complete phase two is the same as equation (2).

Phase three: The number of time steps is equal to the number of steps in the ring divided by two because the nature of topology there are wraps around the edges as explained before. So the number of time steps needed to complete phase three is equals to the number of time steps in phase three of the simple case.

The algorithm needs five steps to tolerate the link failure between two nodes. So, the total number of time steps TS (a) in all phases is:

$$TS\ (a) = (4\times\sqrt{\frac{N}{6}}) + (6\times\sqrt{\frac{N}{6}} - 4) + (2\times\sqrt{\frac{N}{6}}) + 5 = \quad O(\sqrt{N}) \qquad (11)$$

ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Journal of Engineering Technology (ISSN: 0747-9964)
Volume 6, Issue 1, Jan, 2017, PP.97-112
ORIGINAL ARTICLE

6. Conclusions

In this paper, we proposed a dynamic solution for leader failure in the honeycomb torus network to solve the leader failure with the presence of intermittent link failure. This article enhanced a previous algorithm in [18] by adding steps to elect a new leader despite the presence of link failure. The main idea behind this paper is to use a detour to pass over the intermittent link failure during the execution of the algorithm. Number of messages needed to complete the algorithm is less than or equal to O ($N^{1.5}$) within $O(\sqrt{N})$time steps. Comparing the result of time steps and number of messages with [18], we find that both algorithms have the same complexity numbers, but in details this algorithm needs more messages and time steps, because it needs ACK and Detour messages over the first algorithm.

This work can improve to deal with many links failure in the future, and researchers think to study leader election algorithms in other topologies like hive and hyper mesh networks.

References

1. Abu-Amara, H. and Lokre, J.(1994), Election in Asynchronous Complete Networks with Intermittent Link Failures, IEEE Transactions on Computers, Vol. 34 No. 7, pp. 778-788.
2. Akbar B., and Effatparvar M. and Effatparvar M. (2006) Bully Election Algorithm Improvement with New Methods and Fault Tolerant Mechanism, Symposium Proceedings Volume II Computer Science & Engineering and Electrical & Electronics Engineering, PP 501-506.
3. Chang C.-C. and Tsai J. (2012), An Efficient Leader Election Algorithm of Performance-Related Characteristics for Dynamic Networks, International Conference on Smart Grid Systems (ICSGS), vol.45.pp 31-37.
4. Coulouris G, Dollimore J, Kindberg T and Blair G (2012) DISTRIBUTED SYSTEMS Concepts and Design. Addison –Wesley.
5. Devillers M., Griffioen D., Romijn J. and Vaandrager F. (2004), Verification of Leader Election Protocol, Formal Method Applied to IEEE 1394, Springer International Journal on Software Tools for Technology Transfer(STTT).
6. Ghaffari M and Haeupler B (2013) Near Optimal Leader Election in Multi-Hop Radio Networks. Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'13).

7. Ghaffari M., Lynch N., and Sastry S. (2011), Leader Election Using Loneliness Detection, Distributed Computing, 25(6): 427-450, Special issue for DISC.
8. Gholipour M., Kordafshari, M.S., Jahanshahi M. and Rahmani A. M(2009). A New Approach For Election Algorithm in Distributed Systems. Second International Conference on Comm0unication Theory, Reliability, and Quality of Service. IEEE .
9. Ingram R., RadevaT.,Shields P., Viqar S., Walter E. J. and Welch L. J. (2013), A Leader Election Algorithm for Dynamic Networks with Causal Clocks, Distributed Computing, 26(2):75-97.
10. Kumar V, Grama A, Gupta A and Karypis G (2003) Introduction to Parallel Computing. The Benjamin/Cumminy Publishing Company.
11. Liu K. Y., Zhang H.B., Z. and Li L.H. (2010), Fundamental Algorithms of Computer Graphics and the Needs for New Display Materials, International Journal of the Society of Materials Engineering for Resources, Vol.17, No.2.
12. Lynch N., Radeva T. and Sastry S. (2012), Asynchronous Leader Election and MIS Using Abstract MAC Layer, Proceedings of FOMC 2012 (8th ACM International Workshop on the Foundations of Mobile Computing).
13. Marckert J. F. (2005), Quasi-Optimal Leader Election Algorithms in Radio Network with Log-Logarithmic Awake Time Slots, INRIA,pp.97-100.
14. Molina G. (1982), Election in Distributed Computing System, IEEE Transactions on Computers.
15. Radeva T. (2013), Properties of Link Reversal Algorithms for Routing and Leader Election, Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.
16. Refai M. and Ajlouni N. (2006), A New Leader Election Algorithm in Hypercube Networks, Symposium Proceedings Volume II Computer Science and Engineering and Electrical & Electronics Engineering.
17. Refai M. and Sharieh A. and Alshammari F. (2010), Leader Election Algorithm in 2D Torus Networks, The International Arab Journal of Information Technology, Vol. 7, No.2.
18. Refai, M., Alkhader K., and Aloqaily I. , Leader Election Algorithm in the Honeycomb Torus Networks, Journal of Engineering Technology JoET, Vol. 5, Issue 1,2016, pp. 72-92.
19. Sepehri, M. and Goodarzi, M. (2008), Leader Election Algorithm Using Heap Structure, 12th WSEAS International Conference on Computers.

20. Stojmenovic I. (1997), HoneycombNetworks: Topological Properties and Communication Algorithms, IEEE Transactions on Parallel and Distributed Systems, vol. 8, No.10.
21. Tanenbaum S. A. and Steen V. M. (2007), DISTRIBUTED SYSTEMS principles and paradigm, SecondEdition, Pearson.
22. Vasudevan S., DeCleene B., Immerman N., Kurose J. and Towsley D. (2003), Leader Election Algorithms for Wireless Ad Hoc Networks, In Proc. Of IEEE DISCEX III.