World of Computer Science and Information Technology Journal (WCSIT)
ISSN: 2221-0741
Vol. 4, No. 3, 31-37, 2014

Dynamic Solutions for Leader Failure in 3D Torus Networks

Mohammed Refai
Al Zarqa University, College of Science and Information Technology

Abstract— Leader election is a very important algorithm in wired and wireless networks. It is used to solve the single point failure in distributed systems when one process which called leader is responsible to coordinate and manage the whole network. The leader election algorithms (LEA's) solve the instability problem in the network, which caused by leader failure. The research work reported here is concerned with building and designing a dynamic leader election algorithm, to contribute in solving leader crash problem in three dimensional torus networks. The algorithm solves leader failure despite the existing of intermittent links failure. Algorithm performance was evaluated by calculating the number of messages and time steps overall the algorithm. In a network of N nodes connected by a three dimensional torus network, the performance is evaluated, when leader failure is detected by a (N-1) node and algorithm faces F link failure. The number of messages is O(N+F) in $O(\sqrt[3]{N}+F)$ time steps.

Keywords- Dynamic Leader Election; Intermittent Links Failure; Time Complexity; 3D Torus Networks.

I. INTRODUCTION

After publishing leader election algorithm in 3d torus with the presence of one link failure in [19], this paper comes with more dynamic algorithm which can solve the leader failure with many links failure. Leader election, the task of nodes agreeing on the election of a single node in a network, is one of the most fundamental solutions for leader failure problems in distributed computing. It is the ultimate way to break symmetries in an initially unknown system [14].

The LEA aims to find a new leader identified by some characteristics from all other nodes. When the algorithm is terminated, the network is returned to a stable state with one node as leader, and all the other nodes aware of this leader [26]

Distributed systems are used to increase the computational speed of problem solving. These systems use a number of computers which cooperate with each other to execute tasks. The control of distributed algorithms requires one node to act as a controller (leader) in centralized control [25]. If the leader fails for any reason, a new leader should be automatically elected to keep the network working. The LEA's solves this problem by substituting the failed leader by a new deserved leader.

Election process is a program distributed over all nodes. It starts when one or more nodes discover that the leader has failed.

This research is funded by the Deanship of Research and Graduate Studies in Zarqa University /Jordan

It terminates when the remaining nodes know who the new leader is.

LEAs are widely used in centralized systems to solve single point failure problem [3]. For example, in Client-Server, the LEAs are used when the server fails, and the system needs to transfer the leadership to another station. The LEAs are also used in token ring. When the node that has the token fails, the system should select a new node to have the token [1].

In a dynamic network, communication channels go up and down frequently. Causes for such communication volatility range from the changing position of nodes in mobile networks to failure and repair of point-to-point links in wired networks [6].

In distributed systems, there are many network topologies like hypercube, meshes, torus, ring, bus…etc. These topologies may be either hardware processors, or software processes embedded over other hardware topology ([4], [12]). This study will focus on the 3D torus topology where one node works as a leader. This paper proposes a dynamic new LEA to solve leader failure in 3D torus network automatically. Also it guarantees to solve the leader failure problem despite of existing of links failure.

The election algorithms start when the leader failure is detected by one node at simple case, or subset of nodes reached to (N-1) at the worst case. It terminates when the new leader is elected and all other nodes become aware of the new leader [5].

This paper is organized as follows. Section 2 presents related work. Section 3 describes the 3D torus model structure and properties. Section 4 presents the new algorithm in different ways. Mathematical proof for the time steps and message complexity is presented in section 5. Section 6 will conclude the results and suggest future works.

II. RELATED WORKS

Leader election algorithms have been studied by a number of researchers ([1], [2], [3], [5], [6], [9], [10], [11], [12], [13], [16], [17], [19], [21], [22], [24], [25], [26], and [27]). In these studies, the researchers presented different methods to deal with the leader election algorithms. In distributed systems, a major problem is the leader failure and the relevant leader election algorithm. The leader election problem has been studied extensively in various models in distributed computing, including

- Static and dynamic networks. In this research we focus on LE algorithms for dynamic networks where communication channels go up and down frequently. Causes for intermittent communication failure in point-to-point links in wired networks [26].
- Node Identity (ID) (unique identity vs. anonymous ID) (Distinguished vs not distinguished) [27].
- Topology Type (ring, tree, complete graph, meshes, torus, hypercube …etc) ([7], [8], and [20]).
- Communication mechanism used (synchronous vs. asynchronous). [17]
- Transmission media (wired vs. wireless or radio) ([10],[23], [24])
- Some of the previous work dealt with the link failure ([21], [22]).

The leader election solution was first thought of at the end of the seventies and eighties in previous century, it was started by the ring and complete networks ([2], [15]). In the nineties meshes, hypercube and tree were studied. To date, these topologies and wireless networks are still being studied.

This section will look over some previous work in election algorithms and focus on the most relevant researches.

In ([19-21]) refai and etl proposed leader election algorithms in torus and hypercube. The presence of one link failure was solved in these papers.

In [25], Singh G. proposed a protocol for leader election tolerant to intermittent link failure in the complete graph network. He assumes that up to N/2 – 1 links incident on each node may fail. So, up to N2/4 – N/2 links overall the system may fail.

In [16] Molina-G. Presented an algorithm to solve the leader failure for a complete network. It was one of the first five leader election algorithm and it is called Bully algorithm. This algorithm was improved in [2] with new method.

In [23] paper presents a comparative analysis of various leader election algorithms and a new leader election algorithm in MANET in analytical way which considers factors such as node's position, time complexity, message complexity, battery life and security.

In ([14] ,[15]) they present two algorithms that solve deterministic and probabilistic leader election in strong collision detection systems with time costs of O(log u) and O(min(log u; log log n + log(1/€))), respectively, where € is the error probability.

Most of the previous researchers depended on mathematical proof to verify their algorithms. They used the big O notation to obtain the complexity of the number of messages and time steps, which represent the domain factors of the algorithm complexity. Other researchers used simulation to validate their algorithms.

III. MODEL PROPERTIES RESEARCH ASSUMPTIONS

The 3D torus network is similar to 3D mesh, except in the connection between the first and the last nodes (boundaries) in each dimension. These connections make all nodes connected with six neighbors (Left, Right, Front, Back, Up and Down) to present more flexible topology. Figure-1 shows three dimensional torus networks (3, 3, and 3). For research analysis, we use this model with the following properties:

1. 3D torus is a multicomputers consist of N nodes that can be labeled as 0, 1, 2… N-1.
2. The nodes physically form an X * Y * Z, (rows) * (columns) * (Depth), Three-Dimensional torus.
3. A node can send or receive simultaneously to and from the same or different nodes.

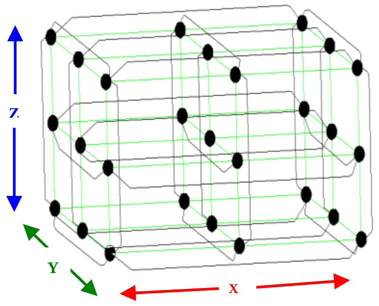

Figure. 1: (3 X 3X 3) Torus Networks

4. The network uses XYZ-routing: a message is routed within a row to the column that contains the destination node and subsequently routed within the column then in depth.
5. Leader failure can occurs any time. This failure may be discovered by one node in simple case, or concurrently by more than one node reached at worst case to N-1 nodes.
6. The proposed algorithm solves leader failure even when there are neighbor's links failures.
7. Each node is connected neighbors by six links as in Figure-2, which Shows node links.

3D torus is one of the most common networks for multicomputers due to their desirable properties, such as ease of implementation and ability to reduce message latency (Jehad et al., 2003). Three-dimensional torus interconnection networks have been used in recent research and commercial distributed memory parallel computers. Examples of such multicomputers are the IBM BlueGene/L (IBM Blue Gene Team, 2008) the

Cray T3D (Kessler and Schwarzmeier, 1993) the Cray XT3. An important advantages of the 3D torus are its lower diameter and higher bisection width, symmetry and regularity (William et al., 2007.

TABLE 1: LINK FAILURE SOLUTION BY DETOURS

Det #	Direction	Detour Routing
1	+ Z	+X(RIGHT),+Z(UP),-X(LEFT)
2	+y	+X(RIGHT),+y(FORWORD),-X(LEFT)
3	+x	+Y(FORWORD), +X (RIGHT),-Y(BACKWORD)
4	- Z	+X(RIGHT),-Z(UP),-X(LEFT)
5	-y	+X(RIGHT),-y(FORWORD),-X(LEFT)
6	-x	+Y(FORWORD), -X (RIGHT),-Y(BACKWORD)

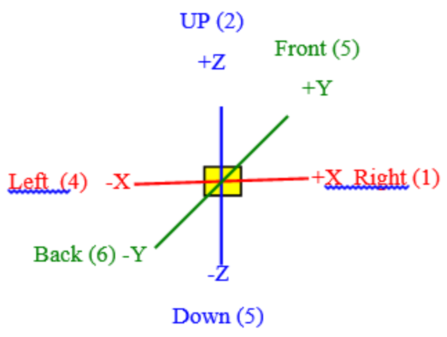

Figure 2. Node Links

This Research assumes the following:

1. Routers should work all the time even with fault node because the fault is in leader properties.
2. All communication links are bidirectional.
3. Leader node could fail due to different reasons which lead to lose of the leadership property. Other nodes can detect this failure, when the time out exceed without acknowledgement. Nodes detect this failure start the election algorithm.
4. Wid: For a given process on processor node i there is set of attributes such as storage capacity, CPU speed, battery age, ram speed. W_{id} is the weight value which will be computed from these attributes. To every node in the network, we will use this value to compare W_{id} for every node with others to elect the leader to be the one with the highest value.
5. Many intermittent links failure are recoverable.
6. Each node has the following variables:

- W_{id}: A unique value for the election process.
- Position: The label indicates its position.
- Leader ID, Leader position.
- Phase and step.
- State: leader or normal or candidate.

IV. DYNAMIC LEA

This section presents the proposed algorithm in phases. Each phase composed of many steps. Before describing the algorithm, the definition of the following variables will assist in understand:

Node State: during the execution of the algorithm the node state will be in one of the following states:

- Normal: the network is normal and no leader failure is detected by this node.
- Candidate: there is a failure and the election process is in progress inside this node.
- Leader: only one node must have this state in a stable network, while this state is absence after leader failure.

The new algorithm is also composed of five phases as in the in [19]. It is proposed to deal with the presence of many intermittent links failure during its execution so it is called dynamic. The main idea to solve this problem is by using more detours in all directions to pass the messages over the links failure. Algorithm phases are as follow:

Phase-1: The algorithm starts by a node(s) that detects leader failures in any location. This node changes its state to candidate. It sends (failure messages) through X axes right (+X) and through Y axes forward (+Y), to inform all neighbors in the same 2D torus about the leader failure.

Any node receives leader failure message makes the following:

- Node state changes to candidate.
- Passes the failure message to the opposite direction through the opposite links depending on the direction from which it has received the message.
- Start phase two: selects its Wid as greater Wid , and send election message through links (z axis up direction). The election message is composed of (message type, Phase, Step, Greater Wid, and Position of the Greater Wid, Message initiator). If the state is candidate, the received message is ignored.

The main contribution in this research is to solve the probability of links failure in all phases. The idea to achieve this goal is to make sender wait for acknowledgement message from receiver, then after time out it uses the detour way to bypass the message to the target node. To choose the suitable detour algorithm uses table 1. Detour routing depends on the direction of the missed message as it shown in table 1. Phase one guarantees that all nodes in the 2D level which have the node(s), that detects the failure, are informed about the leader failure. Each node from this level starts phase 2 by sending an election message in the column Z.

Phase -2: Nodes in candidate state continue election process by sending election message to the neighbor up on the +Z axes. If the message is received successfully, receiver sends acknowledgment messages to the sender and continues to send leader election messages up to the next neighbor. Any node which receives the election message compares its W_{id} with the receiving W_{id}, and continues with the greater Wid. When the election message reaches the node that it starts the process, it sends the result of election to the first node in the column. Eventually this phase puts the results of phase two in the first node of each column with (Z=0) or (x, y, 0). To solve the probability of links failure in phase two, as in phase one, this algorithm uses detours way, to bypass the message to the target node. This way is applied even if the second link failure in the detour itself. Detour routing depends on the direction of the missed message as it shown in table 1. The links failure in the second phase is explained as follow:

1- if the node that detected link failure in the link (up)(+z), it sends link failure message using detour number 1 from table 1 which use the following path +X(RIGHT),+Z(UP),-X(LEFT). If the node detect a link failure for the second time in link +X it sends link failure message using detour 3 +Y (FORWORD), +X (RIGHT),-Y (BACKWORD) and so on for any consequence links failure. By this way the algorithm continue until the message reach its target figure-3.

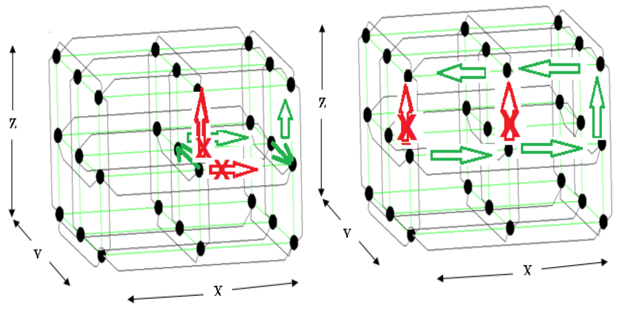

Figure 3: phase two with links failure

Phase-3: Nodes in the 2D torus Z = 0 and Y = 0, or (x, 0, 0) start the election in the Y axes, to obtain the result in one row, Y = 0, Z=0 row, which is (x, 0, 0). If the message is received successfully, the receiver will send acknowledgment message to the sender and continues to send leader election messages to the next neighbor in the direction +Y. This process continues until the message return to the initiator candidate node. The role of these nodes is to wait for phase 4 except node (0, 0, 0) it starts phase 4.

To solve the probability of Links failure in +Y, there will be an alternative path +X (RIGHT), +Y (FORWORD), -X (LEFT). The node that detect a link failure will send a link failure message to node in direction +X, if sender node receive acknowledgment message, it will continue in the alternative path (+X, +Y,-X) in the direction of +Y, else, it detects that there is a link failure in the direction of +Y. To solve this problem in this phase, it sends a link-failure message through link to the right on the X-axis, and continue the new alternative path (+X, +Y,-X) to inform the node in the +Y direction as in figure- 4.

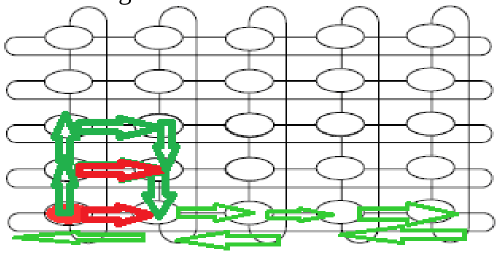

Figure 4: with links failure +x, +x

If the second Link failure is in the direction -X, there will be alternate path (+X, +Y,-X). The general idea is by using table 1 to select the detour depends on the direction of the message figure 5 and 6 explain other two cases.

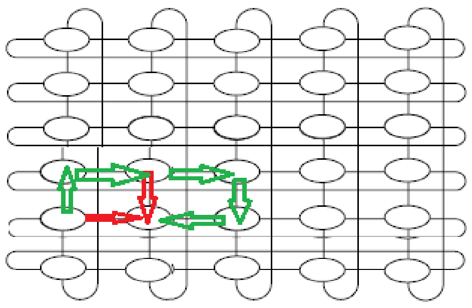

Figure 5: two Links failure in directions +X and -Y

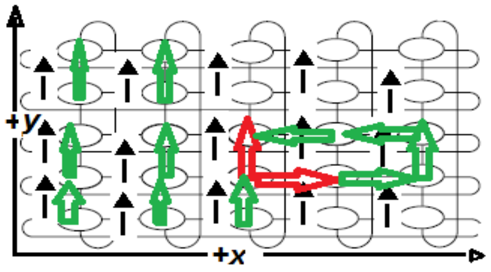

Figure 6: phase-3 two link failure

Phase-4: Node (0, 0, 0) start phase 4 by sending election message to its neighbor in X-axis, to do the election in one row to obtain the result in one node X=0, Y=0, Z=0.figure 7 show the steps in phase 4.

Note: to avoid the probability of links failure in phase four the algorithm uses detours way as in table 1 for any link failure.

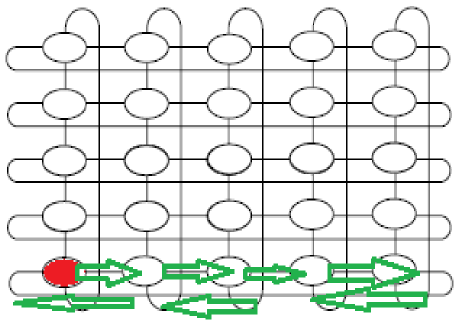

Figure 7 phase 4 election

Phase-5: At the end of phase four, only one node is aware of the new leader information node (0, 0, 0). This node broadcasts leader message in three steps: the first step is to send the leader message in two directions (+,-) X to inform all nodes in the X axis of the new leader information. In the second step each node finish the first step send the leader message in Y axes in two direction (+, -) Y to inform the firs 2D torus about the new leader. Each node in this 2D torus send leader message in Z axes to complete the leader message broadcast. Any node aware of the new leader in phase five ignores any new message about election algorithm. To deal with links failure our algorithm uses detour way as in table 1 to bypass the links failure as discussed in previous phases.

V. Performance Evaluation

Performance evaluation is carried out by computing the number of messages and time steps. The analyses process is carried out for two cases. The first case is the simple case, when the failure is detected by one node. While the second case, is when the leader failure is detected by subset of nodes which can reach all nodes in the worst case.

Simple Case

Number of Messages: Theorem (1): assume that we have N number of nodes in three dimensional torus networks, and F number of intermittent link failures. Then, leader election algorithm needs O (N+F) messages to complete.

Proof: Number of messages is computed for each phase. Then, add the results to get the total number overall the algorithm, proof is for all cases, as follow:

Simple Case: Phase One: Each node in the 2D torus receives one message and sent one acknowledgement. So, the number of messages needed to complete phase one is

$$2(X\,Y) \qquad (1)$$

Phase Two: all nodes in this phase receive election messages and send acknowledgement messages. So, the number of messages needed to complete phase one is

$$[\sum_{i=0}^{Z-1} 2(X*Y)] + 2(X*Y) = 2XYZ + 2XY \qquad (2)$$

Phase Three: Nodes (X, y, 0) needs Y election messages through link 4 to start the phase, and waits for acknowledgement. Eventually, the result reaches to nodes in row labeled (0, y, 0) after this number of messages:

$$\sum_{i=0}^{X-1} 2Y = 2\,XY. \qquad (3)$$

Phase Four: Node (0, 0, Z) starts leader election in Z axes, each node, along Z axes, sends election message and receives acknowledgement. The result reaches to node (0, 0, 0) after this number of messages:

$$\sum_{i=0}^{Z-1} 2 = 2\,Z \qquad (4)$$

Phase Five: Each node in the 3D torus receives one message and sent one acknowledgement to complete leader message broadcast. So, the number of messages needed to complete phase five is:

$$2(X\,YZ) \qquad (5)$$

To cover the link failure in all phases algorithm propose that F represent the number of links failed during the execution of the algorithm, and each link failure need 6 messages to bypass the message (3 information and three acknowledgement messages). So, the total number of messages overall the algorithm is computed by add messages (6 * F) messages to equations (1 to 6) as in Equation 6:

$$2\,XY + 2\,XYZ + 2\,XY + 2\,XY + 2\,Z + 2(X\,YZ) + 6\,F \quad (6)$$

When X=Y =Z = $\sqrt[3]{N}$ then XYZ =N, so the total messages by using N is expressed in Formula 8:

$$4N + 6\sqrt[3]{N^2} + 2\sqrt[3]{N} + 6F = O(N+F) \text{ messages} \qquad (7)$$

Worst Case:

Phase One: all nodes detect the leader failure simultaneously. To start the algorithm each node receives one message and send acknowledgement message. Phase one is finished after one step because all nodes state transform to candidate. The number of messages is equal:

2(XYZ) messages (8)

Phase Two: All nodes start phase three simultaneously by sending election messages through link 2. All nodes also send acknowledgement messages. There for step1 needs 2XYZ messages. Algorithm needs 2XY for each step from 2 to Z. To send the result to the first 2D algorithm needs 2XY. The number of messages needed in this phase is in formula 11:

$$2XYZ + \sum_{I=2}^{Z} 2XY + 2XY = 4XYZ + 2XY \qquad (9)$$

Phases (3, 4 and 5) are the same as in the simple case so, , the total number of messages overall the algorithm in the worst case is computed by add messages in formulas (9,8,7, 3, 4, 5) besides 6*F messages to cover the link failure as in formula 12:

2XYZ + 4XYZ +2XY + 2 XY +2 Z + 2X YZ +6 F = 8XYZ + 4XY + 2Z + 6F (10)

When X= Y = Z = $\sqrt[3]{N}$ the previous equation equals:

8XYZ + 4XY + 2Z + 6F

$$8N + 4\sqrt[3]{N^2} + 2\sqrt[3]{N} + 6F = O\,(N + F) \text{ messages} \quad (11)$$

Time Steps: Theorem (2): Assume that we have N number of nodes in three dimensional torus network. Then, leader election algorithm needs $O\sqrt[3]{N}$ time steps to complete

Proof: Number of time steps is computed for each phase. Then add these numbers to get the total number of time steps overall the algorithm. We apply the computations at the simple case and then at the worst case as follow:

Simple Case:

Phase One

Step 1: One node detects leader failure and sends Leader-failure message through X and Y axis.

Number of time steps is equal to X + Y (12)

Phase Two: In step one all candidate nodes send election messages to the upper neighbors through links labeled 2 (Up).

Step 2 to step Z: nodes receive the election messages make the comparison and pass election messages up with the greater

ID. After Z -1 steps the result of the column leader is found in phase three initiator node. These nodes need another step to send column results to nodes with coordinators (x, y, 0). So the algorithm needs (Z+1) steps to complete phase 2 as in Equation 6:

1+Z-1+1 = Z+1 (13)

Phase Three: Nodes with coordinators (x, Y, 0) start election process in step one by sending the greater ID through link 6 (back). This process continues as follow:

Step 2 to step Y: Any node receive the election message, makes the comparison and sends election message with greater ID to the back neighbor. Phase four is terminated when nodes (x, 0, 0) receive the election message from link 3 (front). This phase needs:

Y steps (14)

Phase Four: Node (X, 0, 0) starts election process in step one by sending the greater ID through link 4 (left). This process continues as follow:

Step 2 to step X: Any node receive the election message, makes the comparison and sends election message with greater ID to the left neighbor. Phase four is terminated when node (0, 0, 0) receive the election message from link 1 (right). This process needs X steps.

To tolerate the probability of the presence of one link failure in phase 3, 4 and 5 the algorithm needs 3 steps as explained in the algorithm description. So the total steps for this phase: X+ 3 steps (15)

Phase Five: Since node (0, 0, 0) it broadcast

X+ Y+ Z steps (16)

The total time steps overall the algorithm in simple case is the summation of time steps in (12 to 16) is:

X + Y+ Z+1+ Y +X+ 3+ X+ Y+ Z +3*F (17)

When X= Y = Z = $\sqrt[3]{N}$, the number of time steps can be expressed as in Equation 21:

$$8\sqrt[3]{N} + 4 + 6F = O(\sqrt[3]{N} + F) \quad (18)$$

In the worst case when all nodes detect the leader failure simultaneously, the time steps will be as follow.

Phase one: all nodes start the algorithm by sending leader-failure message. All nodes state become candidate after one time step. Therefore, phase one needs one time step to complete.

Phase Two: in step one, all nodes start phase two. In step two, one node in each column continues the election, while all other nodes in the same column stop the process. So, number of time steps in this phase is equal Z, and need one step for column result message. Thus the total for phases 1, 2 is:

Z+2 steps (19)

Phases (3, 4 and 5) are the same as in the simple case. The total time steps overall the algorithm in worst case

Z+2 +Y+X+3+X+ Y+ Z +3*F Time steps (20)

When X= Y = Z = $\sqrt[3]{N}$, the number of time steps can be expressed as

$$6\sqrt[3]{N} + 5 + 3F = O\sqrt[3]{N} + F \text{ steps} \quad (21)$$

VI. CONCLUSION

In this work, a leader election algorithm in 3D torus network is proposed and analyzed.

Our proposed algorithm consists of five phases. Phase one is initiated when one or more nodes detects leader failure. This node(s) informs other nodes in the same 2D about leader failure to change its state to candidate. In phase two, nodes aware of leader failure start election process throughout their columns. In phase three, another election is applied on the 2D torus to obtain the new leader information in one row. In Phase four leader election is applied to this row to get new leader information in one node. Last phase, broadcasts one to all is applied to disseminate the new leader information to all nodes. Proposed algorithm considered the probability of links failure in all phases.

Algorithm performance was evaluated by calculating the number of messages and time steps overall the algorithm. In a network of N nodes connected by a three dimensional torus network (X, Y, Z), the performance is evaluated in simple case, when leader failure is detected by one node and in the worst case, when leader failure is detected by (N-1) nodes. For all cases the number of messages is O (N + F) in $O(\sqrt[3]{N} + F)$ steps.

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, July 1994, pp. 778-788.
2. Akbar B., and Effatparvar Mohammed., and Effatparvar Mahdi, (2006), Bully Election Algorithm Improvement with New Methods and Fault Tolerant Mechanism, Symposium Proceedings Volume II Computer Science & Engineering and Electrical & Electronics Engineering, European University of Lefke, North Cyprus, PP 501-506.
3. Antonoiu, G. and Srimani, K.(1996)A Self-Stabilizing Leader Election Algorithm for Tree Graphs, Journal of Parallel and Distributed Computing, 34, Article No. 0059, 1996, pp. 227-232.
4. Coulouris G., Dollimore J., and Kindberg T. , (2005), Distributed Systems Concept and Design, Fourth Edition, Addison-Wesley, USA.
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 Tecknology Transfer(STTT), December 2004.
6. Dolev S., Israeli A. and Moran S., (1997), Uniform Dynamic Self-Stabilizing Leader Election, IEEE Transaction on Parallel and Distributed Systems, VOL 8,NO.4, April .PP 424-440.
7. Flocchini, P. and Mans, B. (1996).Optimal Elections in Labeled Hypercube, Journal of Parallel and Distributed Computing 33, Article No. 0026, pp. 76-83.
8. Fredrickson, N., and Lynch , N.(1987).Election a Leader in Asynchronous Ring, Journal of the ACM, Vol.34, PP. 98-115.
9. IBM Blue Gene Team. Overview of the IBM Blue Gene/P project. IBM Journal of Research and Development, 52(1/2), 2008.
10. Jean-Franqois Marckert (2005), Quasi-Optimal Leader Election Algorithms in Radio Network with Log-Logarithmic Awake Time Slots, F.chyzak(ed.),INRIA,pp.97-100.

11. Junguk L. and Geneva G., (1996), A Distributed Election Protocol for Unreliable Networks, Journal of Parallel and Distributed Computing, 35, PP 35-42.
12. Kumar V. , Grama A. , Gupta A. and Karypis G. (2003).Introduction to Parallel Computing, The Benjamin/Cumminy Publishing Company, Inc,Redwood City, California.
13. Levitin A., (2003), Introduction to The Design and Analysis of Algorithms, Addison Wesley Company, USA.
14. Mohsen Ghaffari and Bernhard Haeupler. Near Optimal Leader Election in Multi-Hop Radio Networks. Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'13), New Orleans, Louisiana, January, 2013.
15. Mohsen Ghaffari, Nancy Lynch, and Srikanth Sastry. Leader Election Using Loneliness Detection. Distributed Computing, 25(6): 427-450, 2012. Special issue for DISC 2011.
16. Molina G, H., (1982).Elections in A Distributed Computing systems, IEEE Transactions on Computers, Vol. 31 Jan 1982, pp. 48-59.
17. Nancy Lynch, Tsvetomira Radeva, and Srikanth Sastry. Asynchronous Leader Election and MIS Using Abstract MAC Layer. Proceedings of FOMC 2012 (8th ACM International Workshop on the Foundations of Mobile Computing), Madeira, Portugal, July 2012.
18. Rebecca Ingram, Tsvetomira Radeva, Patrick Shields, Saira Viqar, Jennifer. E. Walter, and Jennifer L. Welch. A Leader Election Algorithm for Dynamic Networks with Causal Clocks. Distributed Computing, 26(2):75-97, 2013.
19. Refai M. , Oqily I. , Alhamori A. , Leader Election Algorithm in 3D Torus Networks with the Presence of One Link Failure, World of Computer Science and Information Technology Journal (WCSIT), ISSN: 2221-0741, Vol. 2, No. 3, 90-97, 2012.
20. Refai, M. and Ajlouni, N., A new leader Election Algorithm in Hypercube Networks, Symposium Proceedings Volume II Computer Science & Engineering and Electrical & Electronics Engineering, European University of Lefke, North Cyprus, PP 497-501, 2006.
21. Refai, M., Shari'ah, A., Alshammari, F. (2010), Leader Election Algorithm in 2D Torus with the Presence of One Link Failure, IAJIT, Vol. 7, No. 2, April 2010 .
22. Singh G., (1996). Leader Election in the Presence of Link Failures, IEEE Transactions on Parallel and Distributed Systems, VOL 7,No 3,March.
23. Smita Bhoir and Amarsinh Vidhate, A Modified Leader Election Algorithm for MANET, International Journal on Computer Science and Engineering (IJCSE), Vol. 5 No. 02 Feb 2013.
24. Sudarshan V., DeCleene B., Immerman N., Kurose J. and Towsley D. Leader Election Algorithms for Wireless Ad Hoc Networks. In Proc. Of IEEE DISCEX III, 2003.
25. Tanenbaum, A., (2002). Distributed Systems, Prentice-Hall International, Inc, New Jersey.
26. Tsvetomira Radeva. Properties of Link Reversal Algorithms for Routing and Leader Election. Masters thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, June 2013.
27. Yamshita M. and Kammeda T.,(1999), Leader Election Problem on Networks in which Processor Identity Numbers are not Distinct, IEEE Transactions on Parallel and Distributed Systems, VOL 10,No 9,September

AUTHORS PROFILE

Dr. Mohammed Al Refai received his PHD in computer science(CS) from Amman Arab University for Graduated studies, Jordan, 2/2007, M.S degree in CS from Alalbayet university, Jordan, 3/2002. He received his undergraduate studies in CS from mutah university, Jordan, 6/1992. He is currently work as chairman of the Software Engineering Department in Zarqa university in Jordan. His main research interests include many aspects in parallel and distributed systems, Simulation and Data Mining.