RESEARCH ARTICLE OPEN ACCESS

Dynamic Leader Election Algorithm in 2D Torus Network with Multi Links Failure

Dr. Mohammed Al Refai
PHD, Chairman,
Department of Software Engineering
Zarqa University in Jordan

ABSTRACT
Leader election algorithms (LEAs) solve the instability problem in the network, which caused by leader failure. Dynamic LEAs solve some problems that appear during algorithm execution. One of the major problems affect the completion of the algorithm is intermittent or complete links failure. This research concerns in building and designing a dynamic leader election algorithm, to contribute in solving leader failure problem in two dimensional torus network despite the existing of complete or intermittent links failure. . In two dimensional torus network with connected N nodes and F failed links, the proposed algorithm needs O (N+F) messages in time steps to complete the election process.
Keywords:- Dynamic Algorithm, Leader Failure, intermittent Links Failure, concurrency, Time Complexity, 2D Torus

I. INTRODUCTION

This paper presents a dynamic distributed algorithm for electing a leader in 2D torus network. The algorithm concerns with the presense of many intermittent or complete links failure. It improves previous algorithms in [19], [21] that focused on elect the leader with the presense of just one link failure.

Centralized control in distributed systems requires one node to act as a controller (leader) [25].leader is responsible to coordinate all communication between nodes in the network. 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 node.

LEA aims to find a new node identified by some characteristics from all other nodes to be the new leader to substitute the failed leader. 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]. LEA, 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 problem in distributed computing. It is the ultimate way to break symmetries in an initially unknown system [14].

Election process is a program distributed over all nodes. It starts when one or more nodes discover that the leader has failed. It terminates when the remaining nodes know who the new leader is [20].

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 and wired networks to failed [6].

In distributed systems, there are many network topologies like hypercube, meshes, torus, hypermesh, honeycomb, tree, ring, bus…etc. These topologies may be either hardware processors, or software processes embedded over other hardware topology ([4], [12]). This paper proposes a dynamic new LEA to solve leader failure in 2D torus network automatically. Also it guarantees to solve the leader failure problem despite of existing of links failure. It solves the election problem when, the leader failure is detected by one node at simple case, or subset of nodes reached to (N-1) at the worst case.

This paper starts with presenting related work section 2. Section 3 describes the 2D torus model structure and properties. Section 4 describes proposed algorithm. Mathematical proof for algorithm performance is in section 5. Section 6 will conclude the results and suggest future works.

II. RELATED WORKS

The problem of leader failure has received vast amount of attention under various network topologies . This problem becomes considerably by a number of researchers in ([1], [2], [3], [5], [6], [9],[10], [11], [12], [13], [16], [17], [19], [21], [22], [24], [25], [26], and [27]). LEA has been studied extensively in various models in distributed computing, including

- Static and dynamic. In dynamic solution the properities of the network is changed during algorithm execution [26]. This research focuses on dynamic networks where communication channels go up and down frequently. Causes for intermittent communication failure in point-to-point links in wired networks.

- Node Identity (W_{id}) (unique identity vs. anonymous W_{id}) (Distinguished vs not distinguished) [27]. In distinguished

leader wight W_{id} may be identical. This research proposes distinguished $W_{id.}$

- Topology Type (ring, tree, complete graph, meshes, torus, hypercube …etc) ([7], [8], and [20]).
- Communication mechanism used (synchronous vs. asynchronous) [17]. Proposed algorithm used both types.
- 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 $N^2/4 – N/2$ 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 2D torus network is similar to 2D mesh, except in the connection between the first and the last nodes (boundaries) in each dimension. These connections make all nodes connected with four neighbors (Left, Right, Up and Down) to present more flexible topology. Figure-1 shows two dimensional torus networks (4,4). For research analysis, we use this model with the following properties:

1. 2D 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 (rows) * (columns) , Two-Dimensional torus.
3. A node can send or receive messages simultaneously to and from its neighbors.

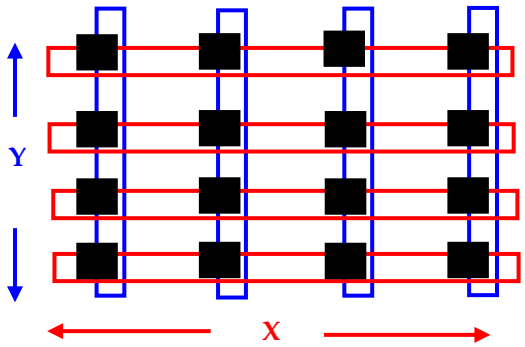

Figure 1: (4X 4) 2D Torus Networks

4. The network uses XY-routing: a message is routed within a row to the column that contains the destination node.
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.

Table 1: Link failure solution by detours

Det #	Direc-	Detour Routing
1	+y	+X(RIGHT),+Y(UP),-X(LEFT)
2	+x	+Y(UP), +X (RIGHT),-Y(DOWN)
3	-y	+X(RIGHT),-Y(FORWORD),-X(LEFT)
4	-x	+Y(UP), -X (LEFT),-Y(DOWN)

6. The proposed algorithm solves leader failure even when there are links failures.

7. Each node is connected to its neighbors by four links as in Figure-2, which Shows node links.

2D torus is one of the most common networks for multi-computers due to their desirable properties, such as ease of implementation and ability to reduce message latency (Jehad et al., 2003). 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 2D torus are its Lower diameter and higher bisection width, symmetry and regularity (William et al., 2007.

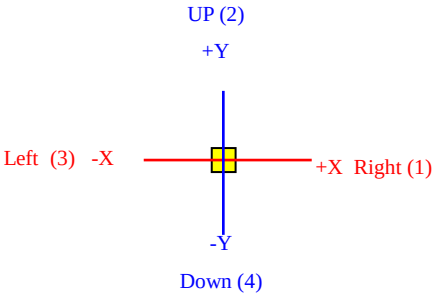

Fig. 2 Node Links

This Research assumes the following:

1. 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.
2. W_{id}: 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.
3. Routers should work all the time even with fault node because the fault is in leader properties.
4. All communication links are bidirectional.
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. PROPOSED DYNAMIC LEA

This section presents the proposed algorithm which composed of foure phases as in our previous LEA with one link failure in [21]. 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 during leader election algorithm execution.

The new algorithm is proposed dynamic solution to deal with the presence of many intermittent links failure during its execution. 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 left (-X), to inform all neighbors in the same X axes about the leader failure.

Any node receives leader failure message makes the following:

- Change its state to candidate.
- Passes the failurs
-
- e 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 W_{id} as greater W_{id}, and send election message through links (Y axis up direction). The election message is composed of (message type, Phase, Step, Greater W_{id}, and Position of the Greater W_{id} , 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 same X axis which have the node(s), that detected leader failure, are in-

formed about this failure. Each node from this level starts phase 2 by sending an election message in the column Y. figure 3 shows that the message faces link failure in +x direction so it use the detoure from table to rout +y, +x, -y to reach the next node. Probem increased when the message faces onother link failure in detoure itself in –y direction. So it solves it again as in figure 3.

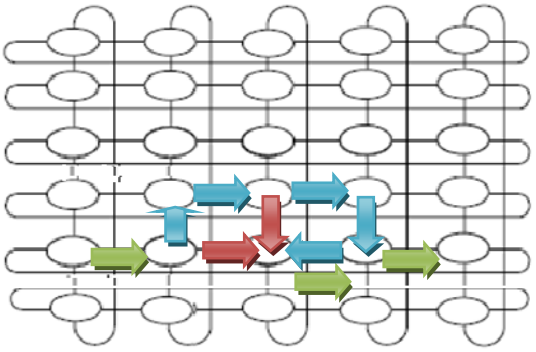

Figure 3 Phase 1 with two links failure

Phase -2: Nodes in candidate state continue election process by sending election message to the neighbor up on the +Y 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 W_{id}. 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 (Y=0) or nodes with cardinality (x, 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 appears 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)(+Y), it sends link failure message using detour number 1 from table 1 which use the following path +X(RIGHT),+Y(UP),-X(LEFT). If the node detect a link failure for the second time in link +Y it sends link failure message using detour 1 for second time +X(RIGHT),+Y(UP),-X(LEFT). The problem increases if the message face link failure for the third time in –x direction it uses detoure 4 +Y(UP), -X (LEFT),-Y(DOWN) and so on for any consequence links failure. By this way the algorithm continue until the message reach its target figure-4

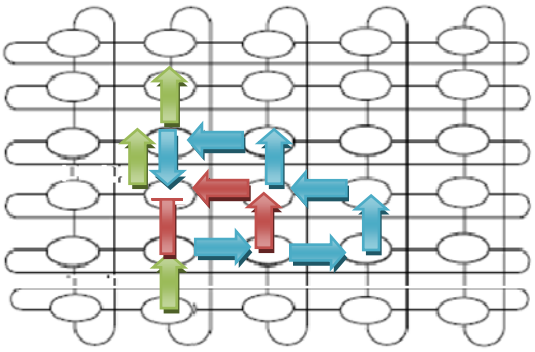

Figure 4 Phase 2 with three links failure

Phase-3 : Node (0 , 0) start phase 3 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. figure 5 shows the steps in phase 3.

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

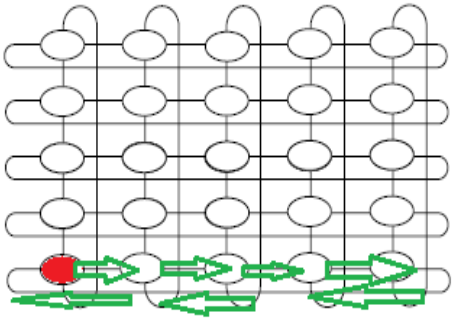

Figure 5 phase 3 election steps

Phase-4:

At the end of phase three, only one node is aware of the new leader information node (0, 0, 0). This node broadcasts leader message in three stages : the first stage 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 stage each node finish the first stage send the leader message in Y axes in two direction (+,-)Y to inform all 2D torus about the new leader. Any node aware of the new leader in phase four 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.

1. Simple Case

Number of Messages: Theorem(1) : *assume that we have N number of nodes in two 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 line that contains the node that detect the faikure, receives one message and sent one acknowledgement. So, the number of messages needed to complete phase one is

2X (1)

Phase Two: all nodes in this phase receive election messages and send acknowledgement messages and the algorithm need 2X messages to infom the results to the first row. So, the number of messages needed to complete phase two is

2 XY + 2X (2)

Phase Three: Nodes (X, 0) needs **X** election messages and X acknowledgement messages so the number of messages is:

2X (3)

Phase Four: Each node in the 2D torus receives one message and sent one acknowledgement to complete leader message broadcast . So, the number of messages needed to complete phase four is:

2X Y (4)

To cover the links failure in all phases algorithm propose that F represent the number of links failed during the execution of the algorithm, and each link failure needs 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 4) as in Equation 5:

2 X +2 XY+2X+ 2 X + 2 XY +6 F= 6X + 4XY + 6F (5)

When X=Y = $\sqrt[2]{N}$ then XY =N , so the total messages by using N is expressed in Formula 6:

4N+ 6 $\sqrt[2]{N}$ **+ 6F= O(N+F)** messages (6)

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

2(XY) messages (7).

Phase Two: All nodes start phase two simultaneously by sending election messages through link 2 (+Y). All nodes also send acknowledgement messages. There for step1 needs 2XY messages. In Step 2 the number of messages is 2X * Y/2 and so on to the last step which needs 2X messages.. To send the result to the first row algorithm needs **2**X. The number of messages needed in this phase is in formula 8:

$$= \mathbf{2X(2Y-1) + 2XY=} \quad 2X\left(\sum_{i=0}^{\frac{Y}{2}} \frac{Y}{2^i}\right) + 2XY$$

6XY-2X (8)

Phases (3, 4) 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 (8,7,4, 3) besides 6*F messages to cover the link failure as in formula 9 :

2XY + 6XY-2X +2 X + 2 XY +6 F = 10**XY + 6F** (9)

When X= Y = $\sqrt[2]{N}$ the previous equation equals:

10N + 6F = O(N + F) messages (10)

Time Steps: Theorem (2): *Assume that we have N number of nodes in two dimensional torus network. Then, leader election algorithm needs* $O\sqrt[2]{N}$ *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 axis in two directions +x and -x. Number of time steps is equal to

X /2 (11)

Phase Two: In step one all candidate nodes send election messages to neighbors in direction + y through links labeled 2 (Up).

Step 2 to step Y: nodes receive the election messages make the comparison and pass election messages up with the greater ID. After Y -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, 0). So the algorithm needs **(Y+1)** steps to complete phase 2

as in Equation 12:

1+Y-1+1 = Y+1 (12)

Phase Three: Nodes with coordinators (x, 0) start election process in step one by sending the greater ID through link 1 (right). 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 back neighbor. Phase three is terminated when nodes (x, 0) receive the election message from link 3 (left). This phase needs:

X steps (13)

Phase Four: after phase 3, node(0,0) has the informations about the new leader. This node starts broadcasting this result to all nodes. Broadcast as described in algorithm description needs

X/2+ Y/2 steps (14)

To solve the problem of links failure each link failure needs 3 time steps to by pass the message so the numbe of time steps is = 3 * F

The total time steps overall the algorithm in simple case is the summation of time steps in (11 to 14) is :

X/2 + Y+ 1 +X+ X/2+ Y/2 + 3*F (15)

When X= Y = $\sqrt[2]{N}$, the number of time steps can be expressed as in Equation 16:

$$\frac{7}{2}\sqrt[2]{N}+1+3F=O(\sqrt[2]{N}+F) \quad (16)$$

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 becomes candidate after one time step. Therefore, phase one needs **one** time step to complete.

Phase Two: the number of time steps in this phase is equal Y, and need one step for column result message. Thus the total for phases 1,2 is:

Y+2 steps (17)

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

Y+2 + X+ X/2+ Y/2 +3*F **Time steps** (18)

When X= Y = $\sqrt[2]{N}$, the number of time steps can be expressed as

$3\sqrt[2]{N}+2+3F = O\sqrt[2]{N}+F$ **steps** (19)

VI. CONCLUSION

This work presents improved algorithm to elect new leader despiteof the presense of many links failure in 2D torus network.

Proposed algorithm starts by informing about leader failure then make elelction process and finaly broadcast the result 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 2D torus network The number of messages is O(N + F) in $O(\sqrt[2]{N}+F)$ 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 Paral-

lel 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, and 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.

ACKNOWLEDGMENT

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

Author 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