Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 1, January 2015, pg.102 – 111

RESEARCH ARTICLE

Leader Election Algorithms in Torus and Hypercube Networks Comparisons and Survey

Mohammed Refai

SE Department, Zarqa University, Jordan

refai@zu.edu.jo

Abstract— Leader failure is one of the most fundamental problems in distributed systems. This problem can be solved by Leader Election Algorithms (LEAs). These algorithms move the system from an initial state where all the nodes are in the same computation state into a new state where only one node is distinguished computationally (called leader) and all other nodes aware of this leader. This research aims to make survey for leader election algorithms in the most popular network topologies hypercube and torus networks. Comparisons will be made between algorithms in these networks using fixed number of nodes.

Keywords— Hypercube, Torus, Leader Election, Distributed Systems, Link failure

I. INTRODUCTION

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). If the leader crashes or fails for any reason, a new leader must be automatically elected to keep the network working. The LEA's solves this problem by substituting the failed leader by a new deserved one [23].

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 aware of the new leader [3]. LEAs are widely used in centralized systems to solve single point failure problem.

In distributed systems, there are many network topologies like hypercube, meshes, torus, ring, hyper mesh, butterfly…etc. These topologies may be either hardware processors, or software processes embedded over other hardware topology [13]. After solving leader failure problem in 2D torus, 3d torus and hypercube networks with the presence of links failure, this study come to Survey LEAs in these networks, and make a Comparisons between them [16-20].

3D torus and hypercube are most popular multicomputers networks, due to their desirable properties. Some of these properties contain ease of implementation, ability to reduce message latency, lower diameter, higher bisection width, symmetry and regularity [13, 23]. 3D torus interconnection networks have been used in recent research and commercial distributed memory parallel computers. Examples of such multicomputers are the IBM Blue Gene/L , the Cray T3D , Cray XT3 [11].

The organization of this paper starts by reviewing previous work in the next Section. Section 3 describes the hypercube model and torus model structures and properties. Section 4 reviews LEAs in each model. Comparisons and analyses are presented in section 5 and eventually main ideas are conclude.

II. RELATED WORK

LEAs were started by the ring and complete networks at the end of the seventies [3]. In the nineties meshes, hypercube and tree were studied([1],[3],[8],[28]). To date, these topologies and wireless networks are still being studied ([14], [20]). LEAs have been studied in different perspectives as follows: The nature of the algorithms (Dynamic vs. Static) ([19][6]) ,Node Identity (ID) (unique identity vs. anonymous ID) (Distinguished vs not distinguished) [18] , Topology Type (ring, tree, complete graph, meshes, torus, hypercube …etc) [10], Communication mechanism used (synchronous vs. asynchronous) [5] , transmission media (wired vs. wireless or radio). Some of the previous work dealt with the link failure ([15], [9], [22]). This section will look over some previous work in election algorithms and focus on hypercube and torus researches which are the most relevant to research reported here.

In (Molina-G., 1982) :This study presented an algorithm to solve the link failure for complete network [14], (Villadanjosand etl,2005) improved LEA in complete networks[24]. Bully algorithm was improved in [2]. In (Gerard, 1993): he proposed an election algorithm for oriented hypercube, where each edge is assumed to be labelled with its dimension in the hypercube. The algorithm exchanges O(N) messages and uses O(log2N) time steps to solve the problem and in (Flocchini and Mans, 1996) In this literature the election problem in hypercube networks was studied, by using two models with sense of direction, the dimensional and the distance models. The proposed algorithm needs $\theta(\log 3N)$ time steps using $\Omega(N)$ messages [8]. Intermittent Link Failures was studded in (Abu-Amara and Loker, 1994) [1].

In (Flaviu Cristian, and Christof Fetzer, 2008) research propose a formal definition for the timed asynchronous distributed system model. They present extensive measurements of actual message and process scheduling delays and hardware clock drifts. These measurements confirm that this model adequately describes current distributed systems such as a network of workstations. They also give an explanation of why practically needed services, such as consensus or leader election, which are not implementable in the time-free model, are implementable in the timed asynchronous system model [8].

In (Singh G., 1996) he proposed a protocol for leader election tolerant to intermittent link failure in the complete graph network [21]. In (Navneet M., Jennifer L., Welch, Nitin V., 2001) they presented two new leader election algorithms for mobile ad-hoc networks [15] In (Sudarshan and others, 2003). They proposed two asynchronous distributed system in which the various rounds of election proceed in a lock-step fashion [22].

III. MODELS PROPERTIES

This section describes the prosperities of hypercube and 3D torus to show the main differences between them. A hypercube Hn, of order n, is defined to be regular symmetric graph G = (V,E) Where V is the set of 2n vertices = N, each representing a distinct n-bit binary number and E is the set of symmetric edges such that two nodes are connected by an edge if the Hamming distance between the two nodes is 1 i.e., the number of positions where the bits differ in the binary labels of the two nodes is 1. For example, in H3, the node 010 is connected to three nodes 110, 000 and 011 [25].

The diameter and radius of the hypercube equal (Log N). The shortest bath between any two nodes is less than or equal (Log N). This bath can be founded using Exclusive Or (EXOR) operation between the source label and destination label. The Hypercube graphs have an elegant recursive structure. To construct a labelled (d+1) dimensional hypercube, take two d-dimensional hypercube and extend all labels in the first one with a 0 and all labels in the second a 1. For each process in the first one adds an edge (of direction d) to connect it with the associated node in the second hypercube [13]. Figure-1 shows the 3-dimension hypercube with the labels for its nodes.

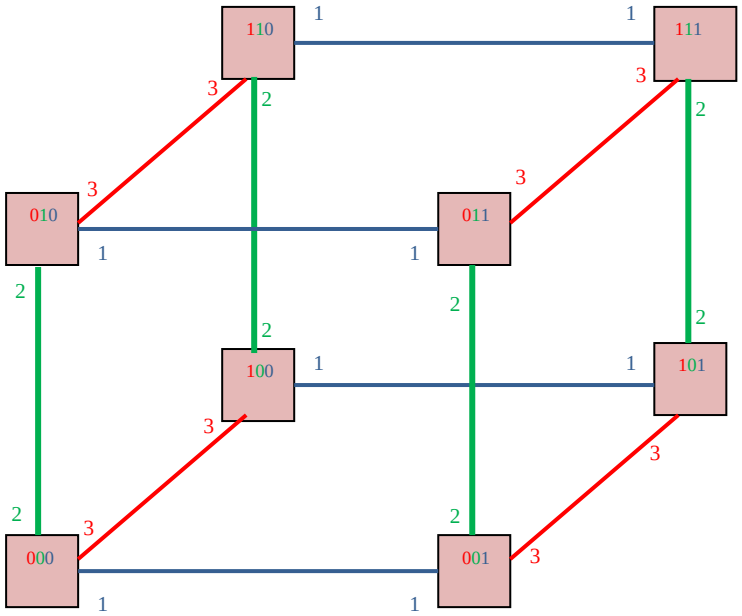

Fig-1 3 Dimension Hypercube

The 3D torus network is similar to 3D mesh with wraparound links between boundaries in each dimension [4]. These connections make all nodes connected with six neighbours (Left, Right, Front, Back, Up and Down) to present more flexible topology[7]. Figure-2 shows three dimensional torus networks (3, 3, 3). For research analysis, we use this model with the following properties:

1. 3D torus is a multicomputer consists of N nodes that can be labelled 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.

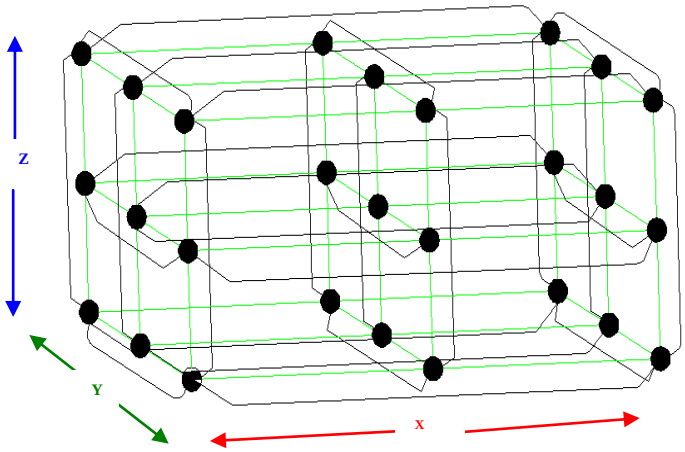

Figure 2: (3 X 3X 3) Torus Networks

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 neighbour's links failures.
7. Each node is connected neighbours by six links as in Figure-2, which Shows node links.

3D torus is one of the most common networks for multicomputer due to their desirable properties, such as ease of implementation and ability to reduce message latency [10]. 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 [11] the Cray T3D [12]. Important advantages of the 3D torus are its Lower diameter and higher bisection width, symmetry and regularity ([26],[27]).

LEAs used in this research assumed the following Assumptions for both topologies:

- One intermittent link failure is recoverable.
- Routers working all the time even at the fault process.
- All communication links are bidirectional.
- Leader process could fail due to different reasons which lead to lose of the leadership property. Other processes can detect this failure, when the time out exceed without acknowledgement. Nodes detect this failure start the election algorithm.
- Each node calculates a weight that defines its relative importance to be the new leader. This weight is represented by identification Number (ID) for each node. This algorithm considers the state when the IDs for two or more processes are same.
- Each process is in one of the following States: leader or normal or candidate.

IV. SURVEY

This section presents a survey for our leader election algorithms in hypercube and torus networks. We will review three algorithms for each topology, and then a comparison will be made between leader election algorithm in torus and hypercube networks with 64 nodes for each of them.

1- Leader Election Algorithm in Hypercube: The first algorithm was proposed by (refai and ajloni) [16] to present a distributed deadlock-free solution for leader failure problem in hypercube networks contention and synchronization issues were considered. The algorithm consists of three phases as follow:

Phase One: This phase starts when one or more nodes detect the leader failure. These nodes start the election algorithm:

Step1: send an election message to the node that differs in the right most bit. Election message composed of (phase, step, winner ID, and winner position) through link 1.

Step2: the sender and receiver in the previous step send an election message to the two associated nodes that differ in the second bit through link 2.

Step3: the senders and receivers from the previous step send an election message to the nodes that differ in the third bit through link 3.

Step Log N: N/2 processes (the senders and receivers from the Log N –1 step) send an election message to the nodes that differ in the Log N left bit through link Log N.

Phase one ends after Log N steps, reducing the participant nodes to N/2. The leader id and its position for the whole hypercube becomes within (Log N –1) dimension hypercube.

Phase Two: The second phase uses the reduction all-to-one communication operation to guide the result towards the process that has the address $X_1 0 \ldots 0_d$. (X means 0 or 1) As follows:

Step1: nodes with the second left bit = $X_1 1 X_3 \ldots X_d$ send election message to nodes with the second left bit = 0 ($X_1 0 X_3 \ldots X_d$) through link Log N - step.

Step2: the receivers in the previous step and third bits in its labels = 1 ($X_1 X_2 1 \ldots X_d$) send an election message to the nodes differ in this bit ($X_1 X_2 0 \ldots X_d$) through log N - step.

Step (Log N – 1): the receiver in the previous step with Log N bit in its label equals ($X_1 0_2 0_3 \ldots 1_d$) , sends an election message to the process that differs in the right most bit ($X_1 0_2 0_3 \ldots 0_d$) through log N - step.

After the end of phase 2 the last node ($X_1 0_2 0_3 \ldots 0_d$) is the node that has all the new leader.

Phase3: In this phase the node ($X_1 0_2 0_3 \ldots 0_d$) broadcasts a message containing the result of the election using one-to-all broadcast operation, the broadcasted message (leader message) contains the new leader ID.

The first algorithm needed O(N) messages to elect a new leader in O(log N) time steps when N represent the number of nodes.

This algorithm needs the following messages and time steps to complete leader election process:

Total : (N-1) + (N/2 –1) + (N –1) = 5N/2 –3 = O(N) messages

Total : N(Log N) + (N/2 –1) + (N –1) = O (N log N) time steps

2- Dynamic Solution for Leader Failure in Hypercube Network: The second algorithm was proposed by Refai and Oqily in [19] to consider any changes in the processes and links states during the execution of leader election algorithms. It proposed a dynamic algorithm to solve this problem despite these changes. It solved leader failure problem when the ID number is not distinguished and with the probability of presence of intermittent or complete link failure. The proposed algorithm uses O(N) messages to elect a new leader in O(log N) time steps.

As in the previous algorithm this one consists of three phases. Phase one is started, when one or more nodes detect leader failure. After log N time steps, phase one reduces the count of participated nodes in the election algorithm to N/2 nodes aware of the election process. In second phase the algorithm uses the reduction all-to-one communication operation to have the result in one node with address (X10203…0d) (X means 1 or 0). Finally, in the third phase node (X10203…0d) broadcasts the leader message to all nodes in the network. During each step in phases one and two, received ID is compared with the local ID and Greater ID is passed to the next step. **When the two IDs are equal; summation of ones in processes label is compared and selects the greater to complete with. Detail description for all phases is as follow:**

Phase One: This phase starts when one or more nodes detect the leader failure. Each node detect the failure change its state to candidate and initiate the election:

Step1: Send an election message to the node differs in the right most bit through link 1. Election message composed of (phase, step, winner ID, and winner position). The election message changes the state of receiving node from normal state to candidate state.

Step2: The sender and receiver in the previous step (nodes in the candidate state) send an election messages to the two associated nodes differ in the second right most bit through link 2.

Step r (r >2) : The senders from the previous step send an election messages to the nodes differ in the r-1 left bit through the link r-1. The receiver nodes send an authentication message (election message) through link r-2 to insure that the election message reached the node or not due to the link failure. After sending the authentication message the node wait for acknowledgement or time out to send election in step r.

Nodes that receive the authentication message may be received the election message in advance, so the authentication message is ignored after get the greater ID. In the second case when the authentication message is received by a node in normal state. To tolerate the probability of link failure the node sends the election message through link (r-1). In both cases the nodes that receive the authentication message send an acknowledgement to the sender.

Nodes that received the authentication message send the election messages through link r + 1 to continue the algorithm. Surely any node send any election or authentication message must use the greater ID it knows.

Step Log N : N/2 processes (the senders and receivers from the Log N –1 step) send an election messages to the nodes differ in the Log N left bit through log N link. End the phase 1.

During the execution of phase one, if the receiver is aware of the failure and is in progress with its own initiated election step, it will complete the greater step and terminate the smaller one. Each node receives the election message, it compare its own ID with received ID then complete the next step with the greater ID. **If the received ID equal the local ID, algorithm select the ID with greater label bits to complete with.** After this phase leader ID is included in N/2 nodes one half of the hypercube. These nodes will continue to phase 2.

Phase Two: second phase uses the reduction all-to-one operation. We add some alterations to tolerate the link failure. Reduction operation applied to N/2 nodes hypercube that was resulted from phase 1, to have the result in one process that have the address $X_1 0_2 \ldots 0_d$. , "X means 0 or 1 depends on which half to complete" As follows:
Step1 To Step Log N –3 : nodes that reach phase 2 and with the (step + 1) and (step + 2) left most bit =1 ($X_1 11 X_4 \ldots X_{d)}$ exchange the greater id with the nodes with the same properties except the (step + 2) left most bit = 0 through the link (log N – step –1). The participants in the exchange needs acknowledgement or wait until time out to progress. After the exchange these nodes send election message to nodes with the second left bit = 0 ($X_1 0 X_3 \ldots X_d$) through link Log N – step. These steps conclude the result inside a two dimensional hypercube in spite of the probability of the link failure.

Step Log N -2: the receivers in the previous step and with the two right bits (10, 01) send election to node with the two right bits(11).

Step Log N-1: After comparison and get the greater ID node (11) send the winner to nodes(10,01).

Step Log N: In last step nodes(10,01) send election message to the node ($X_1 0_2 0_3 \ldots 0_d$).

After the end of phase 2 the last node ($X_1 0_2 0_3 \ldots 0_d$) is the only node knows about the leader ID.

Phase3: In this phase the node ($X_1 0_2 0_3 \ldots 0_d$) broadcasts the result to all nodes considering the link failure. Leader message contains new leader ID and position. The broadcast will be as follows:
Step 1: node ($X_1 0_2 0_3 \ldots 0_d$) send leader message through link (step) to node ($X_1 0_2 0_3 \ldots 1_d$).
Step 2: the two nodes in phase 1 send the message through link (step).
Step 3 to Log N: Each node receive the leader message send it through the link step. To consider the link failure for this phase each node receives the leader message send another message through link step – 2 to check if the message reaches the other side. Node that receives the extra message may know about the broadcast so it ignores the extra message. If the node doesn't know about the message it sends the extra message through link step -1. The last step isn't needed after the step Log N.

This second algorithm needs the following messages and time steps to complete leader election process:

Total : 3N(Log N) + (N –4) + (2N –1) = O (N log N) Messages
3 log N–3 + 3Log N -6 + Log N + 2 = 7LogN -7 Steps = O (Log N) Steps

3- Leader Election Algorithm in 2d Torus Networks with the Presence of One Link Failure: the third algorithm was proposed by (Refai and etl) in [17] and concerned with leader election algorithm in two dimensional torus networks. The algorithm solved this problem despite the existence of one link failure. In a network of N nodes connected by two dimensional torus networks, the algorithm used O (N) messages to elect a new leader in $O(\sqrt{N})$ time steps.

The algorithm uses X and Y to represent the dimensions and x and y to represent node position.

- **Phases**: The proposed algorithm is composed of four phases, as follows:
  1- **Phase One:** The node that detects leader failure informs the failure event to its row.
  2- **Phase Two**: The nodes in candidate states do election process within each column to obtain the result in the first row.
  3- **Phase Three:** Nodes in the first row make the election within the first row to obtain the result in one node.
  4- **Phase Four:** The node that aware of the new leader in phase three, broadcast the new leader to all nodes.

Phase One: The algorithm starts by node(s) that detects leader failure. This node sends failure messages through link 1 (right) and 3 (left) to inform its row about leader failure. A failure message informs the receivers about leader failure. To avoid the probability of link failure in this phase, the failure message is sent in two directions.

Phase Two: The candidate nodes send election messages through links 2. Any node which receives the election message compares its ID with the received ID in order to continue with the greater. This process ends when the initiator position receives the same message. After the column election, the result is sent to the first node of each column.

This phase faces two problems: concurrent initialization and link failure. To deal with the first problem, any candidate node which receives the election message ignores the message. If there is no link failure, the result for the column is found in the node that completed the ring. This node sends the result to the node labelled (x,0).

To solve the second problem, the node that sends the election message waits for acknowledgment. If the node doesn't receive this message after time out, it detects that there is a link failure. The role of the node that detects the link failure is to send link-failure message through link 3. The node which receives this message forwards it through link 2, and then left to pass the failure link.

Phase Three: When the node which is labelled (0, 0) (the most left node in the first row) finishes phase 2, it starts phase 3 by sending election message through links 1 and waits for acknowledgment. Any node which receives a phase three election message from the left sends an acknowledgment message. Then it compares IDs and sends a phase three election message through link 1. If the node doesn't receive the acknowledgment message after time out, it detects there is link failure. To solve this problem in this phase, this node sends a link-failure message through link 2, then link 1 and down to pass the link failure. Phase three is terminated when phase three election message is received by node (0,0). This node starts phase four by broadcasting the result to all nodes in phase four.

Phase Four: After phase three, one node aware of the new leader information. This node broadcasts the result. The initiators of the leader message, within the row in row broadcast and within the columns in column broadcast send the leader message in two directions. This is to recover the probability of one link failure.
The third algorithm needs the following messages and time steps to complete leader election process:
$7N+4N^{1/2}+5 = O(N)$ messages

$$\frac{3}{2}\sqrt{N}+\frac{3}{2}\sqrt{N}+7=O(\sqrt{N}) \quad \text{Steps}$$

4- Dynamic Solutions for Leader Failure in 3D Torus Networks: The fourth algorithm is proposed by Refai [20]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. When leader failure is detected by a (N-1) node and algorithm faces F links failure . The number of messages is O(N+F) in $O(\sqrt[3]{N}+F)$ time steps. 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 guarantee 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 neighbour 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 neighbour. 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 (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

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.

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.

Phase-4 : Node (0 , 0, 0) start phase 4 by sending election message to its neighbour 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.

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.

This fourth algorithm needs the following messages and time steps to complete leader election process:

Total : $8N + 4\sqrt[3]{N^2} + 2\sqrt[3]{N} + 6F = O(N + F)$ messages

$6\sqrt[3]{N} + 5 + 3F = O\sqrt[3]{N} + F$ steps

V. COMPARISONS

In this section a comparison will be made by applying each algorithm in 64 nodes in each network. Then compare the number of messages and time steps. This section use the following equations to compare:

First algorithm:

$(N-1) + (N/2 -1) + (N -1) = 5N/2 -3 = O(N)$ messages

$N(\text{Log } N) + (N/2 -1) + (N -1) = O(N \log N)$ time steps

Second algorithm:

$3N(\text{Log } N) + (N -4) + (2N -1) = O(N \log N)$ Messages

$3 \log N - 3 + 3\text{Log } N - 6 + \text{Log } N + 2 = 7\text{Log } N - 7$ Steps $= O(\text{Log } N)$ Steps

Third algorithm:

$7N+4N^{1/2}+5 = O(N)$ messages

$\frac{3}{2}\sqrt{N} + \frac{3}{2}\sqrt{N} + 7 = O(\sqrt{N})$ Steps

Fourth algorithm:

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

$6\sqrt[3]{N} + 5 + 3F = O\sqrt[3]{N} + F$ steps

TABLE I
NUMBER OF MESSAGES WHEN N = 64

Algorithm	Messages	Steps
First	157	478
Second	13120	35
Third	485	31
Fourth	590	32

We can see in table 1 that the first algorithm needs long time to complete the election algorithm, to get less messages. On the contrary the second algorithm needs big number of messages with short time to complete, because the messages overhead that need to deal with links failure in hypercube.

In the third and fourth algorithms which solved the leader problem in 2d and 3d torus networks. The Figures are closed to each other with a simple preference for the third algorithm. Figure 3 shows explain the differences among the four algorithms.

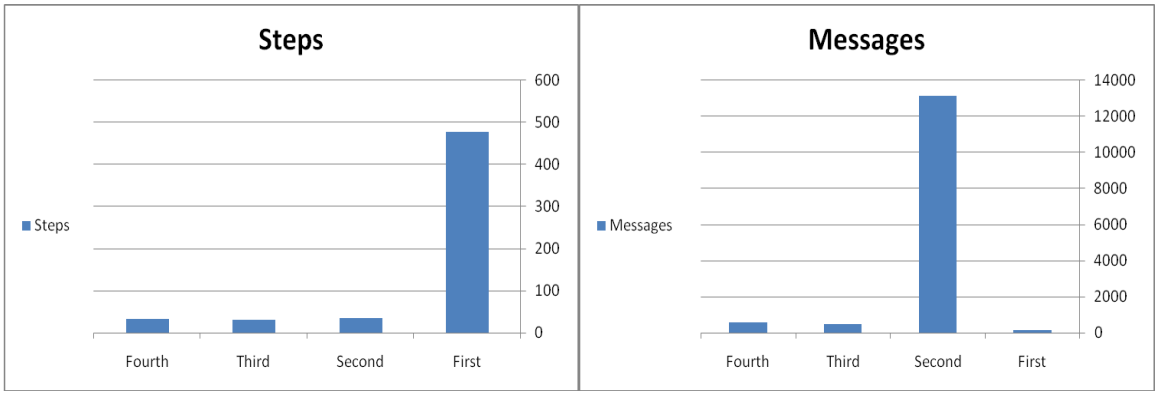

Fig 3 Messages and time steps comparisons when N = 64 nodes

VI. CONCLUSIONS

This research gives a review and survey for Leader election algorithm in hypercube and torus networks with analyses and comparison. Four algorithms were used to achieve this objective. The first algorithm related to deal with leader failure in hypercube and the second one is also related to hypercube but it was concerned with link failure. Third algorithm related to 2D torus network with link failure and the last one concerned with 3D torus networks. Last section makes a comparison between leader algorithms in these networks using 64 nodes for each of them. The third algorithm that used for 2D torus is the most effective among all of them.

ACKNOWLEDGEMENT

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

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] B. and Laxmikant V. Kal ,Scalable Topology Aware Object Mapping for Large Supercomputers, PHD Dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign, 2009.

[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] Duato, J. Yalamanchili, S. and Ni, L. , (1997) Interconnection Networks an Engineering Approach, IEEE Computer Society, The Institute of Electronic Engineers, Inc, Los Alamitos, California.

[8] Flocchini, P. and Mans, B. (1996).Optimal Elections in Labeled Hypercube, Journal of Parallel and Distributed Computing 33, Article No. 0026, pp. 76-83.

[9] 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.

[10] Jehad Al-Sadi, Khaled Day and Mohamed Ould-Khaoua, A Fault-Tolerant Routing Algorithm for 3-D Torus Interconnection Networks, *The International Arab Journal of Information Technology, Vol. 1, No. 0, July 2003*.

[11] IBM Blue Gene Team. Overview of the IBM Blue Gene/P project. IBM Journal of Research and Development, 52(1/2), 2008.

[12] Kessler, R., and Schwarzmeier, J. "CRAY T3D: A New Dimension for Cray Research," Proc. COMPCON, pp.176-182, 1993.

[13] Kumar V. , Grama A. , Gupta A. and Karypis G. (2003).Introduction to Parallel Computing, The Benjamin/Cumminy Publishing Company, Inc, Redwood City, California.

[14] Molina G, H., (1982).Elections in A Distributed Computing systems, IEEE Transactions on Computers, Vol. 31 Jan 1982, pp. 48-59.

[15] Navneet M., Jennifer L., Welch, Nitin V., (2001), Leader Election Algorithms for Mobile Ad Hoc Networks, by NSF grant CCR-9972235.

[16] Refai, M., Ajluni N., (2006), New Leader Election Algorithm in Hypercubes with Minimum time steps and number of messages, European University of Lefke, 4th FAE International Symposium, Gemikonag, TRNC , 30 November- 1 December.

[17] Refai, M., Shari'ah, A., Alshammari, F. (2010), Leader Election Algorithm in 2D Torus with the Presence of One Link Failure, The International Arab Journal of Information Technology, Vol. 7, No. 2, April 2010 .

[18] Refai, M., Leader Election Algorithm In Hypercube network When The Id Number Is Not Distinguished, The 2nd International Conference on Information & Communication Systems, May 22-24, 2011, Irbid, Jordan.

[19] Refai, M. and AL-Oqily I., (2011) Dynamic Solution For Leader Failure In Hypercube Network, ISTEC-2011, International Science & Technology Conference, Istanbul University, Turkey.

[20] Refai M. , Dynamic Solutions for Leader Failure in 3D Torus Networks:, World of Computer Science and Information Technology Journal (WCSIT), ISSN: 2221-0741, Vol. 2, No. 3, 90-97, 2012.

[21] Singh G., (1996). Leader Election in the Presence of Link Failures, IEEE Transactions on Parallel and Distributed Systems, VOL 7,No 3,March.

[22] 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.

[23] Tanenbaum, A., (2002). Distributed Systems, Prentice-Hall International, Inc, New Jersey.

[24] Villadanjos J., Cordoba,F. Farina, M. and Prieto, Efficient leader election in complete networks, Proceedings of the 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing IEEE, 2005.

[25] Wei Shi and Pradip K Srimani, Leader Election in Hyper-Butterfly Graphs, IFIP International Federation for Information Processing NPC 2004 beijing, China, October 2004, pp 292-299.

[26] William K., Nellson d., and Ryan S., 2001, Drawing Graph on the Torus [Online]. Available at http://bkocay.cs.umanitoba.ca/g&g/articles/Torus.pdf , (verified 15 Mar. 2007).

[27] William K., and Winnipeg M., 2001, Embidings of Small Grapg on the Torus [Online]. Available at: http://bkocay.cs.umanitoba.ca/g&g/articles/Embeddings.pdf, (verified 16 Mar. 2007).

[28] Yamshita M. and Kammeda,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.

