World of Computer Science and Information Technology Journal (WCSIT)
ISSN: 2221-0741
Vol. 2, No. 3, 90-97, 2012

Leader Election Algorithm in 3D Torus Networks with the Presence of One Link Failure

Mohammed Refai
Department of SE
Faculty of Science and Information Technology
Zarqa University, Jordan

Ibrahim AL-Oqily
Department of CS, Faculty of
Prince Al-Hussein Bin Abdullah II for Information Technology
Hashemite University, Jordan

Abed Alhamori
Faculty of Information Technology
Albalqa University, Jordan

Abstract — Leader election is the process of choosing a leader for symmetry breaking where each node in the network eventually decides whether it is a leader or not. This paper proposes a new leader election algorithm to solve the problem of leader failure in three dimensional torus networks. The proposed algorithm solves the election problem despite the existent of link failure. In a network of N nodes connected by three dimensional torus network, the new algorithm needs O(N) messages to elect a new leader in $O\sqrt[3]{N}$ time steps. These results are valid for two cases: the simple case where the leader failure is detected by one node, and the worst case where the failure is discovered by N-1 nodes.

Keywords- Concurrency; Leader Election; Link Failure; leader failure; 3D Torus Networks.

I. Introduction

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 [3, 4, 8]. LEAs solve the instability problem in the network, which is caused by leader failure. LEAs aim to elect one node to be a new leader. The new leader is identified by some characteristics that other network nodes do not have. When the algorithm is terminated, the network is returned to normal state with one node as a leader, while all other nodes aware of this leader.

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 (or 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 elected one [4, 33, 34].

Election process is a program distributed over all nodes. It starts when the leader failure is detected either by one node, or by a subset of nodes up to N-1 (where N is the number of network nodes) in the worst case. It terminates when the new leader is elected and all other nodes become aware of the new leader.

LEAs are widely used in centralized systems to solve single point failure problem [5]. 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 [3].

In distributed systems, there are many network topologies like hypercube, meshes, torus, ring, bus…etc [33]. These topologies may be either hardware processors, or software processes embedded over other hardware topology [9, 18, 16, 2]. in our previous work [25] leader failure problem in 2D torus networks with the presence of one link failure was presented and solved, this paper focuses on the 3D torus topology where one node should serve as a leader. This paper proposes a new election algorithm to solve leader failure in 3D torus network automatically. Also it guarantees to solve the leader failure problem despite of the existing of one link failure.

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 concludes the results and suggests future works.

II. Related Work

Leader election algorithms have been extensively studied [3, 4, 5, 7, 8, 11, 12, 13, 14, 16, 20, 21, 23, 25, 26, 28, 29, 30, 31, 39]. 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 proposed election algorithms vary based on the following:

- The nature of the algorithms (Dynamic vs. Static) [8, 13, 25, 1, 26].
- Node Identity (ID) (unique identity vs. anonymous ID, Distinguished vs. not distinguished) [39].
- Topology type (ring, tree, complete graph, meshes, torus, hypercube …etc) [3, 10, 25, 26, 35].
- Communication mechanism used (synchronous vs. asynchronous) [25, 26].
- Transmission media (wired vs. wireless or radio) [14].
- The presence of failures such as link failure [3, 25].

The leader election solution was first thought of at the end of the seventies, it was started by the ring and complete networks [3, 21, 30]. In the nineties meshes, hypercube and trees were studied. To date, these topologies and wireless networks are still being studied [14, 20].

In [25], we proposed LEA to solve leader failure problem in 2D torus networks with the presence of one link failure. In a network of N nodes connected by two dimensional torus network, the algorithm uses O(N) messages to elect a new leader in $O\sqrt[2]{N}$ time steps.

In [29], Singh proposed a protocol for leader election tolerant to intermittent link failure in the complete graph network. The message complexity of the protocol was $O(N^2)$.

In [21] Molina Presented an algorithm to solve the leader failure for a complete network. That algorithm is one of the earliest LEAs and it was called Bully algorithm.

In [35] Authors presented a novel algorithm that reduces the information necessary to elect a leader compared with other leader election algorithms for complete networks.

In [12] Fredrickson and Lynch assumed that the processes are physically or logically ordered, So that each process knows who its successor is. The election message is built when any process detects that the leader is not functioning. The process sends messages containing its number to the successor. If the successor is down, the sender will skip over it and go on to the next number along the ring. During each step the sender adds its own number to a list in the message. Eventually, the message gets back to the process that started it. That process recognizes this event when it receives an incoming message containing its own process number. At that point, the process sends a leader message to inform all the processes about the new leader.

In [36] Authors proposed an election algorithm for the oriented hyper butterfly networks with $O(N\log N)$ messages.

In [13] Gerard, proposed an election algorithm for oriented hypercube, where each edge is assumed to be labeled with its dimension in the hypercube. When N represents the size of the cube, the algorithm exchanges O(N) messages and uses $O(\text{Log}_2 N)$ time steps to solve the problem.

Abu-Amara and Loker [3], considered the problem of, fault tolerant, leader election in asynchronous complete (fully connected) distributed networks.

In [8] the election problem in hypercube networks was studied, by using two models: sense of direction with dimensional and sense of direction with distance models. The proposed algorithm needs $O(\log^3 N)$ time steps using O(N) messages. Self-stabilizing algorithm for leader election in a tree graph was proposed in [5]. In [22] Authors presented two new leader election algorithms for mobile ad-hoc networks.

In [32] Sudarshan et al, proposed two cheat-proof election algorithms: Secure Extreme Finding Algorithm (SEFA), and Secure Preference-based Leader Election Algorithm (SPLEA). Both algorithms assume asynchronous distributed system in which the various rounds of election proceed in a lock-step fashion. Mathematical proof is usually used to verify the algorithms' correctness. And the big O notation is used as a mean to obtain the complexity [19] of the number of messages and time steps, which identifies the domain factors of the algorithm complexity [10, 12]. In certain cases simulation was used to validate the algorithms' correctness and to measure their cost [28].

To the best of our knowledge none of the presented algorithms in literature have been proposed to solve the leader failure election in a 3D torus mesh.

III. Model Descriptions and Properties

Three-dimensional 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 [15]. 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 [40], the Cray T3D [17], the Cray XT3. An important advantage of the 3D torus over the 2D torus is its lower diameter and higher bisection width, which means that it can achieve reductions in communication delays given the same number of processors.

In 3D Torus network, interconnection topology is a torus graph with N = X * Y * Z nodes (where X is the number of nodes in the X dimension, and Y is the number of nodes in the Y dimension, and Z is the number of nodes in the Z dimension of the torus network). 3D torus network is similar to 3D mesh, except in 3D torus there are connections 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 [37, 38]. Figure-1 shows three dimensional torus network (9, 4 ,4).

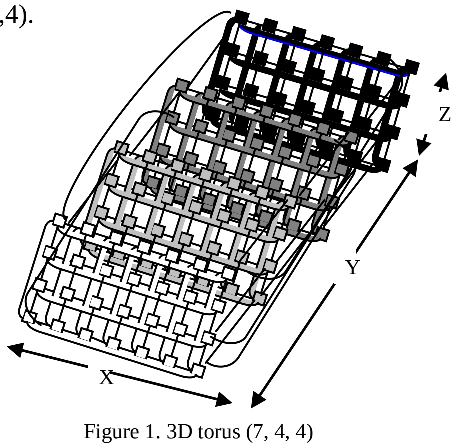

Figure 1. 3D torus (7, 4, 4)

In this paper, we use the following model:

1. The multicomputers consist of N nodes, labeled 0, 1, 2… N-1.
2. The nodes physically form an X * Y * Z, (rows) * (columns) * (Depth), three-dimensional torus.
3. Communication is allowed with one node only at a time. Multi–cast is not implemented in the hardware.
4. A node can send or receive simultaneously to and from the same or different nodes.
5. 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.
6. Each node is connected to neighbors by six links as shown in Figure 2.

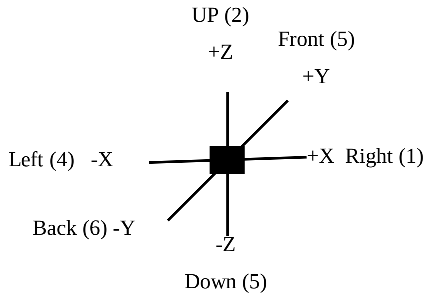

Figure-2 Node Links

A torus network has advantages that make it one of the preferable topologies. Torus is an attractive Structure for parallel processing due to its symmetry and regularity [37]. Diameter of the Torus is X + Y + Z. Node labels as (x,y,z) and use X-Y-Z routing techniques. The number of links is 3*(X*Y*Z). In fact, it has been shown to be a very versatile and robust. 3D torus is capable of executing several efficient parallel algorithms and it is a suitable architecture for design tightly coupled systems in both parallel and distributed systems.

This Research assumes the following:

1. All communication links are bidirectional.
2. Routers should work all the time even with fault node because the fault is in leader properties.
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 that detect this failure start the election algorithm.
4. To LEA, each node calculates a weight that defines its relative importance. Then, compares it with the weight of other nodes that it has received and propagate the maximum weight. This weight is represented by distinguish identification (ID) for each node.
5. Link failure during algorithm execution may occur.
6. Leader failure may be detected by a subset of nodes (concurrent failure). This case becomes complicated when the failure is detected by N-1 nodes (worst case).
7. Each node has the following variables:
    - ID: A unique value for the election process.
    - Position: The label of its position.
    - Leader ID, Leader position.
    - Phase (to store the current phase during the algorithm execution) and step (to store the step number in this phase).
    - State: leader or normal or candidate.

IV. PROPOSED SOLUTION

This section describes and 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 the algorithm:

Node State: during the execution of the algorithm each node 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 algorithm is composed of five phases briefed as follows:

In phase one, node(s) that detects the leader failure informs all nodes in 2D torus X, Y in the same Z level about the failure. These nodes will then change their states from normal to candidate. In phase two, nodes in candidate state start the election process by sending an election message in the Z axes to obtain the election result for each column in the 2D torus with Z = 0 (X,Y,0). In phase three, nodes in the first 2D torus Z = 0 make the election in the Y axes for Y coordinate to obtain the result in one row, in Y = 0, Z=0 row, which is (X,0,0). In phase four, nodes on the X axis make the election in one row to obtain the result in one node X=0, Y=0, Z=0. In the last phase, nodes that are aware of the new leader information, from phase 4, broadcast this information to all nodes. The following paragraphs present the details of each phase:

Phase One: The algorithm starts by node(s) that detects leader failure. This node changes its state to candidate. It sends failure messages through X axes (+X, -X) and through Y axes (+Y, -Y) to inform all neighbors in the same 2D torus (same Z level) about the failure. Each node receives failure message do the following:

- Node state is changed to candidate.
- Pass the failure message to the opposite direction through the opposite links depending on the direction it has received the message.
- Starts phase two: selects its ID as greater ID, and sends election message through links +Z. The election message is composed of (message type, phase, step, greater ID, and position of the greater ID, message initiator). If the state is candidate, the received message is ignored because candidate nodes had been informed by the leader failure.

Note: to avoid the probability of link failure in phase one, the proposed algorithm sends failure messages in two directions (left and right)(-X,+X) in X axis and (front and back) (-Y,+Y) in Y axis. Phase one guarantee that all nodes in the 2D level that has the node(s), which detects the failure, are informed about the leader failure. Each node from this level directly starts phase 2 by sending an election message in the column Z.

Phase Two: Candidate nodes that have been informed about leader failure in phase one, send an election messages through Z - axis (Up). Any node receives the election message compare it's ID with the received ID, and continues with the greater ID. When the election message reaches the node that started the process, it sends the result of the election to the first node in the column. The proposed algorithm puts the results of phase two in the first node of each column with (Z=0) i.e. (X,Y,0). This phase faces two problems, concurrent initialization of phase 2 in the same (0,0,Z) and link failure. To deal with the first problem: any candidate node receives an election message, if its coordinators position is greater than the initiator of the message coordinators position; it will ignore the message because it has started phase two so there is no need to repeat the process. If there is no link failure, the result for the column is found in the node that complete the ring. This node sends the result to the node labeled (X, Y, 0).

To solve the link failure problem, the node that sends an election message, waits for acknowledgment. If the node doesn't receive this message after a 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 the link to its left then the node that receives the link-failure message from the right forwards it through (up) and then left to bypass the failure. To complete the process the algorithm, sends the result to node labeled (X,Y,0). After the completion of phase two, 2D torus with Z=0 has the result so the election inside this part will take place in phase three.

Phase Three: Nodes, that finished phase two with X position equal 0, (0,Y,0) start phase three by sending election messages through X axes (right, or +X). Any node receives the election message compare its ID with the received ID, it continue with the greater ID. If sender doesn't receive acknowledgment message after time out, it assumes that there is a link failure. To solve this problem, it sends link-failure message through link Z-axis (up) then link X- axis(right) and forward it down to bypass the link failure. The proposed algorithm conclude the results of phase three in the row with (X,0,0) coordinates.

Phase Four: After finishing phase three the new leader ID is available in row labeled (X,0,0). When node (0,0,0) finish phase three, it starts phase four by sending an election message through X-axis(right). Any node that receives phase four election message from left direction, it sends an acknowledgment message. It compares IDs and sends phase four election message to the right of X. If the sender doesn't receive the acknowledgment message after time out, it detects there is a link failure. To solve this problem in this phase, it sends link-failure message through link Z-axis (up) then link X- axis (right) and forward it down to bypass the link failure. Phase four is terminated when the phase four election message is received by node (0, 0, 0). This node starts phase five by broadcasting the result to all nodes.

Phase Five: At the end of phase four, only one node is aware of the new leader information. This node broadcasts the result as follows:

X-axis broadcast: node (0,0,0) sends leader message in two directions through links -X and +X (left, right) to inform all nodes in the x axis of the new leader information even if there is a link failure.

Y-axis broadcast: In the same way, node (0,0,0) and its X axis nodes send the leader message to nodes in front and back (+Y,-Y) to inform the Y axis of the new leader information.

Z-axis broadcast: all nodes in the 2D (X, Y, 0) send the leader message through links Up and down (+Z,-Z) to inform Z axis of the new leader information. Each node that receives the Z axis broadcast changes its state to normal and change their contents information according to the leader message (Any node aware of the new leader in phase five ignores any new message about election algorithm).

In phase five initiators of the leader message, within the X, Y, Z axis send the leader message in two directions, to tolerate the probability of link failure.

In the following a pseudo code for the proposed algorithm is presented. A number of assumptions and variables have to be assigned as follows:

- Each node has the following variables:
  1. Local ID: the node ID that participate in the election process.
  2. Local Pos: The node position.
- The algorithm uses five types of messages:

1. Election: Composed of: Phase (1 to 5), step, ID(the winner ID), Pos (the winner position), initiator of the message.
2. Leader: contains the new leader (ID and position).
3. Link-Failure: Similar to the election message, except the type to pass the link failure.
4. Column-result: is used in phase two to inform column result to the first row.
5. Failure: is used to inform about the leader failure detection.

- Each node is in one of four states:
  1. Normal: when the node is unaware of any failure and the network is stable.
  2. Candidate: when the node is aware of the failure and the node is participating in the election process.
  3. Leader: one node must have this state in a stable network.
  4. Failure: when the node lose the leader character. Figure 4 shows the pseudo code for the proposed algorithm.

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.  Case state = Normal
      Upon detecting failure
          {
            State = Candidate
             Phase = 1
             Send inform message in links (1,3,4 and 6)
             Phase =2
             Send election message on link (2)
              Wait for Ack message
          }
      Upon receive inform message from link L
          {
        Pass Inform message on Link (L+2) mod 6 +1.
             State = Candidate
              Phase = 2
             Step = 1
             Send election message on link (2)
           Wait for Ack message
           }
    Upon receiving election message from link 5
            {
                State = Candidate phase = 2
                Compare received ID with Local-ID and select the Winner-ID
                Step = Step +1
               Send election message on link (2)
             Wait for Ack message
            }

    2- Case state = Candidate

    Upon Receiving Election message
     If (Phase = 2)
        {
             Compare received ID with Local-ID and select the Winner-ID
             If Step = Z (Network height) then
              Send Height-Leader message to node(x,y,0) (node same column
             and first_2D torus)
             Else
                  {
                Step = Step+1
                 Send election message on link (2)
                Wait for Ack message
               Send ackt message on link (5)
                }
            If time out without receive Ack message from link l then
             {
                Send link-failure message on link L+1
             }
        }
    Upon receiving link-failure message
            If the message was received from link 5, then forward it on link 1
            If the message was received from link 4, then forward it on link 5
             If the message was received from link 2, then
                {
                  Compare received ID with Local-ID and select the Winner-ID
                        If Step =Z (Network height) then
             Send Height-Leader message to node(x,y,0) (node same column
             and first_2D torus)
                   Else
                    {
                   Step = Step+1
                  Send election message on link (2)
                    }

    Upon Receiving Column-Leader message

     {
    Phase = 3,
     If  z = 0 and x = X
     Select greater ID
    step =0
               Send election message on link 4

        Wait for Ack message
        }

    If Phase = 3 then
     {
       Up on receive election message from link 1 ( right)
         select the greater-ID; increment step
         If Step = X (Network Length) then
                Phase = 4; step = 0;
                If (x=0,z=0, y=Y;Send election message on link 6(–y) backward

         Else
                Send election message on link 4 (left)
                 Wait for Ack message
                   If time out without receive Ack message from link L then
                {
    Use denture to pass the message as in phase 2
                  }
       If (Phase = 4)
      Up on receive election message from link 3 (+ z)
         select the greater-ID; increment step
            If Step = z (Network Depth) then
              Phase = 5;step = 0;
              If (x=0,z=0, y=0;
               State = normal; Send Leader message on link (1,4)
           If Step < z (Network Depth) then
              Increment step; send election message on link 3
             Wait for ack message

           If time out without receive Ack message from link L then
              {
          Use denture to pass the message as in phase 2
                  }
     Upon Receiving Leader message
        If the state is normal or leader disregard the message
         If the message from link 1 or 4 then
          {
            Pass the message to inverse link
           State = normal
         Send Leader message on links 2 and 5
          }
            If the message from link 2 or 5
             {
            State = normal
                Pass the message to inverse link
              Send Leader message on links 3 and 6

                  }
            If the message from link 3 or 6
              State = normal
               Pass the message to inverse link

    End the algorithm
```

In the following sub-section number of messages and time steps are computed for all cases.

- Number of Messages

Theorem (1) : *assume that we have N number of nodes in three dimensional torus network. Then, leader election algorithm needs O (N) messages to complete.*

Proof:

Number of messages is computed for each phase. Then, add the results to get the total number used by the algorithm, proof are for simple case and worst case, as follow:

Simple Case: In phase one each node receives one message except the last two nodes, which receive one extra message for each of them, when they send the last message before receiving from inverse direction. So, the number of messages needed to complete phase one is

$$X + 2 \qquad (1)$$

Phase Two: As in phase one the inform messages are sent from nodes in line (x, 0, 0) through Y axis to inform nodes in (x, y, 0) so the number of messages will be:

$$X * (Y+2) \qquad (2)$$

Phase Three: X*Y candidate nodes start the election by sending election messages through links labeled 2. In each step from one to Z the algorithm needs X*Y messages. Same number of messages is required for acknowledgments. X*Y messages are needed to inform the first 2D torus about columns-result. This formulated as in the following formula:

$$[\sum_{i=0}^{Z-1} 2(X*Y)] + 2(X*Y) = 2XYZ + 2XY \qquad (3)$$

Phase Four: 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 the number of messages is:

$$\sum_{i=0}^{X-1} 2Y = 2\ XY. \qquad (4)$$

Phase Five: 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 the number of messages is:

$$\sum_{i=0}^{Z-1} 2 = 2\ Z \qquad (5)$$

Phase Six: Node (0, 0, 0) starts row broadcast by sending leader messages through links 1 and 4. As shown in phase 1 and 2, X+2 and X *(Y+2) messages are needed for row and depth and XY *(Z+2) for columns. Number of messages needed to inform the leader message to all nodes is:

$$(X+2) + X * (Y+2) + XY * (Z+2)$$
$$= XYZ + 3XY + 3X + 2 \qquad (6)$$

To cover the link failure in phases (three, four, five), the algorithm needs three messages. So, the total number of messages used by the algorithm is computed by adding messages in equations (1 to 6) and as shown in Equation 7:

$$X + 2 + X * (Y+2) + 2(XYZ) + 2\ XY + 2\ Z + XYZ + 3XY + 3X + 2 + 3 = 3XYZ + 6XY + 6X + 2Z + 7 \qquad (7)$$

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

$$3N + 6\sqrt[3]{N^2} + 8\sqrt[3]{N} + 7 = O(N) \text{ messages} \qquad (8)$$

Worst Case in phase One all nodes detect the leader failure simultaneously. To start the algorithm each node sends two messages in links 1 and 4. Phase one is finished after one step because all nodes state transform to candidate. The number of messages is equal

$$2(XYZ) \text{ messages} \qquad (9)$$

Phase Two: all nodes start phase two simultaneously so each node sends two messages in links 3 and 6. Phase two is finished after one step because all nodes informed about leader failure. The number of messages is:

$$2XYZ \qquad (10)$$

Phase Three: All nodes start phase three simultaneously by sending election messages through link 2. All nodes also send acknowledgement messages. Therefore, 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:

$$2XYZ + \sum_{I=2}^{Z} 2XY + 2XY = 4XYZ + 2XY \qquad (11)$$

Phases (4, 5 and 6) are the same as in the simple case so, the total number of messages used by the algorithm in the worst case is computed by adding messages in formulas (11,10,11, 4, 5, 6) in addition to 9 messages to cover the link failure and as shown in formula 12:

$$2XYZ + 2XYZ + 4XYZ + 2XY + 2Z + XYZ + 3XY + 3X + 2 + 9$$
$$= 9XYZ + 5XY + 3X + 2Z + 11 \qquad (12)$$

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

$$9N + 5\sqrt[3]{N^2} + 5\sqrt[3]{N} + 11 = O(N) \text{ messages} \qquad (13)$$

The results in 8 and 13 proof theorem (1)

- Time Steps

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

Proof:

The needed time steps by the algorithm are the summation of time steps for each phase. In the following we apply the computations to the simple case and then to the worst case as.

Simple case: in phase one 1tep 1, one node detects leader failure and sends leader-failure message to the right and left neighbors. Steps 2 to X/2 + 1. In each step, nodes that receive leader-failure messages forward the message through the inverse link, and sends acknowledgement message. Number of time steps is equal to:

$$X/2 + 1 \qquad (14)$$

Phase Two: Same way as in phase one but through Y rows, so phase two needs

$$Y/2 + 1 \text{ steps} \qquad (15)$$

Phase Three: 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:

$$1 + Z - 1 + 1 = Z + 1 \qquad (16)$$

Phase Four: 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 (17)

Phase Five: 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 (18)

Phase Six: Since node(0,0) has the new leader information and it sends this information in two directions (left and right), the row broadcasting is terminate after X/2 steps and extra step may occur if X is odd. Same idea for depth broadcasting (Y/2+1) steps and column broadcasting (Z/2+1) steps. So, the total number for this phase is:

X/2+1+ Y/2+1+ Z/2+1 steps (19)

The total time steps needed by the algorithm in simple case are the summation of time steps in (14 to 18) is as in Equation 20:

X/2 + 1+Y/2+1+ Z+1+Y +X+3 +X/2+1+ Y/2 + 1 + Z/2 +1 =
2X+2Y+ (3 *Z) /2 +9 Time steps (20)

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

$$\tfrac{11}{2}\sqrt[3]{N} + 7 = O(\sqrt[3]{N}) \qquad (21)$$

Worst case: 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: after one step phase two is terminated and all nodes start phase 3. Therefore, phase two also needs one time step to complete.

Phase Three: 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, the 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 and 3 is:

Z+3 steps (22)

Phases (4, 5 and 6) are the same as in the simple case. The total time steps needed by the algorithm in the worst case are the summation of equations (22, 17, 18, and 19) and as follow:

1+ 1+Z+1 +Y+X + 3 +X/2+1 + Y/2 + 1+ Z/2+1 =
3/2X + 3/2Y + 3/2Z +9 Time steps (23)

When X= Y = Z = $\sqrt[3]{N}$, the number of time steps can be expressed as in Equation 15:

$$\tfrac{9}{2}\sqrt[3]{N} + 9 = O\sqrt[3]{N} \text{ steps} \qquad (24)$$

The results in 21 and 24 proof theorem (2)

VI. Conclusion

In this paper, a leader election algorithm for a 3D torus network is proposed. Algorithm performance was evaluated by calculating the number of messages and the overall time steps needed by 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) in $O\sqrt[3]{N}$ steps.

References

[1] Abdelouahid Derhab and Nadjib Badache,A Self-Stabilizing Leader Election Algorithm inHighly Dynamic Ad Hoc Mobile Networks, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 19, NO. 7, JULY 2008.

[2] Abhinav 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.

[3] 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.

[4] 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.

[5] 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.

[6] Coulouris G., Dollimore J., and Kindberg T. , (2005), Distributed Systems Concept and Design, Fourth Edition, Addison-Wesley, USA.

[7] 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.

[8] 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.

[9] 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.

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

[11] Foster I.(1994).Designing and Building Parallel Programs, Addison-Wesley Publishing Company, USA.

[12] Fredrickson, N., and Lynch , N.(1987).Election a Leader in Asynchronous Ring, Journal of the ACM, Vol.34, PP. 98-115.
[13] Gerard ,T.,(1993). Linear Election for Oriented Hypercube, Technical Report TR-RUU-CS-93-39, Department of computer Science, Utrecht University, The Netherlands.
[14] 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.
[15] 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.
[16] Junguk L. and Geneva G., (1996), A Distributed Election Protocol for Unreliable Networks, Journal of Parallel and Distributed Computing, 35, PP 35-42.
[17] Kessler, R., and Schwarzmeier, J. “CRAY T3D: A New Dimension for Cray Research,” Proc. COMPCON, pp.176-182, 1993.
[18] Kumar V. , Grama A. , Gupta A. and Karypis G. (2003).Introduction to Parallel Computing, The Benjamin/Cumminy Publishing Company, Inc, Redwood City, California.
[19] Levitin A., (2003), Introduction to The Design and Analysis of Algorithms, Addison Wesley Company, USA.
[20] Miroslav K., and Wojciech R., 2004, Adversary Immune Leader Election in Ad Hoc Radio Networks [Online]. Available at
cs.huji.ac.il/labs/.../adhoc/kutylowski_2003adversdaryimmuneleader.pdf, (verified 2 Mar. 2007).
[21] Molina G, H., (1982).Elections in A Distributed Computing systems, IEEE Transactions on Computers, Vol. 31 Jan 1982, pp. 48-59.
[22] Navneet M., Jennifer L., Welch, Nitin V., (2001), Leader Election Algorithms for Mobile Ad Hoc Networks, by NSF grant CCR-9972235.
[23] Ostrovsky, R., Rajagoplan, S., and Vazirani, U.,(1994), Simple and Efficient Leader Election in the Full Information Model. In Proceedings of the Twenty-Sixth Annual ACM Syposium on Theory of Computing.
[24] Power H.., (1999), Algorithms and Application in Parallel Computing, WIT Press/Computational Mechanics Publications, USA.
[25] Refai M., Sharieh A. and Alshammari F., Leader Election Algorithm in 2D Torus Networks with the presence of one link failure, The International Arab Journal of Information Technology, Vol. 7, No. 2, April 2010.
[26] 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.
[27] Richard E. and Kumarss N.,(2004), Foundations of Algorithms Using Java PseudoCode, Jones and Bartlett Publishers,Canada.
[28] Russell, A., Saks, M., and Zuckerman, D.,(1999) Lower Bounds For Leader Election And Collective Coin-Flipping In The Perfect Information Model. In Proceedings of the Symposium on the Theory of Computing (STOC).
[29] Singh G., (1996). Leader Election in the Presence of Link Failures, IEEE Transactions on Parallel and Distributed Systems, VOL 7,No 3,March.
[30] Singh, G., (1991), Efficient Distributed Algorithms for Leader Election in Complete Networks, 11th IEEE Int. Conf. on Distributed Computing Systems, PP 472-479.
[31] Singh G., (1997), Efficient Leader Election Using Sense of Direction, Department of Computing and Information Sciences, Kansas State University, Manhatten, KS66506.
[32] 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.
[33] Tanenbaum, A., (2002). Distributed Systems, Prentice-Hall International, Inc, New Jersey.
[34] Tanenbaum, A., (1995). Distributed Operating Systems, Prentice-Hall
International, Inc, New Jersey.
[35] 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.
[36] 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.
[37] 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).
[38] 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).
[39] 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.
[40] IBM Blue Gene Team. Overview of the IBM Blue Gene/P project. IBM Journal of Research and Development, 52(1/2), 2008.