FAULT TOLERANT ROUTING ALGORITHM IN OCTAGON-CELL INTERCONNECTED NETWORK FOR HORIZONTAL MOVING MESSAGES

Sanjukta Mohanty[1] and Prafulla Ku. Behera[2]

[1]North Orissa University, Sriram Chandra Vihar, Takatpur, Baripada, Odisha, India
[2]Utkal University, Vani Vihar, Bhubaneswar, Odisha, India

ABSTRACT

Octagon-Cell interconnected network can be viewed as an undirected graph, in which vertices and edges can be compared with processors and bidirectional communication links respectively between the processing elements. It has attractive features like small diameter and better bisection width and constant node degree. It is analyzed to arrive at fault-tolerant communication. A fault tolerant communication scheme for this network is described in this paper. Here an efficient routing scheme has been described which routes the horizontal moving messages from source node to the destination node in presence of faulty nodes / link failure along the path. In this algorithm when a message is received by an intermediate node, it will consider itself a new source node.

KEYWORDS

Interconnected networks, Octagon-cell, Routing Algorithm, Network services

1. INTRODUCTION AND RELATED WORK

In an interconnection network, a fault tolerance scheme means the ability to continue operating in presence of faulty nodes / link failures [3]-[14]. If the number of interconnected processors rises, the probability of having faulty nodes increases and for successful transmission it is very much essential to find another fault free path [15], [22]-[24].

In parallel processing systems it is very necessary to select optimal paths for efficient inter process communication. In this system, if each and every processor has the status of all processors then an optimal routing can be possible. In a system it may be possible for each component suffers from hardware or software problem. If the system can't handle the faulty problem, that is unreliable, inefficient [15]-[21].

A fault tolerant scheme has been proposed in hexagonal arrays [25]. It has been described that the routing scheme makes the reconstructed array transparent to the various algorithms utilizing the hexagonal array.

A mesh embedded interconnected hypercube network has been analyzed to arrive at fault tolerant communication. An efficient routing algorithm has been proposed that can route a message from

DOI: 10.5121/ijcnc.2017.9103

a source node to the destination in presence of fault free or single/multiple faulty nodes in mesh embedded hypercube interconnection networks [3].

A fault tolerant routing algorithm has been described for star interconnection network in the presence of faults [26]. A new fault tolerance algorithm has been described in hex-cell interconnection network and the algorithm guarantees the delivery of messages even with the presence of component failure [15].

In our paper a routing scheme has been introduced for interconnected processors, which communicates messages in a faulty octagon-cell. We have described the optimal routing algorithm for octagon-cell interconnected network along with its attractive features in [1]. Here we have derived the fault tolerant algorithm for horizontal moving messages in the network. An octagon-cell interconnected network has many attractive features such as constant node degree, desirable diameter, bisection width [1].

2. OCTAGON-CELL NETWORK TOPOLOGY

An octagon-cell has eight nodes. It has d levels numbered from 1 to d with depth d. Level 1 represents one octagon-cell. Level 2 represents eight octagon-cells surrounding the octagon-cell at level 1. Level 3 represents sixteen octagon-cells surrounding the eight octagon-cells at level 2 and so on [1].

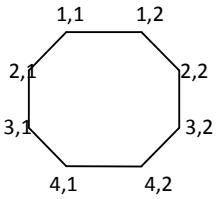

Figure 1: Addressing nodes in Octagon-Cell with level-1
(X, Y represents line no-X with node no-Y)

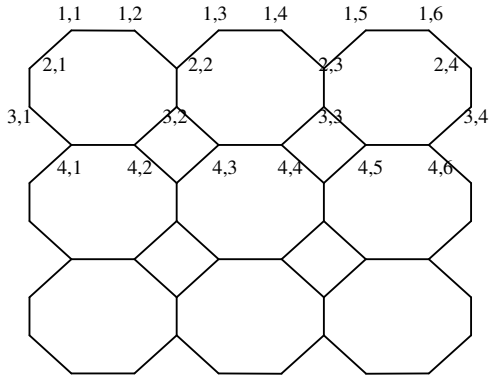

Figure 2: Addressing nodes in Octagon-Cell with level- 2
(X, Y represents line no-X with node no-Y)

We have described the optimal routing algorithm for octagon-cell network in [1]. Due to its recursive structure routing can be done easily. The level numbering scheme is used in the above algorithm. Each node in octagon-cell is identified by a pair (X, Y), Where X denotes the line number in which the node exists and Y denotes serial number of the node in that line. A node with the address (1, 1) is the first node in first line. A node with the address (1, 2) is the second node in first line and so on.

There are four cases for optimal routing for horizontal moving messages. In this paper we have derived the fault tolerant scheme for the following cases.

Case: 1 Fault tolerant routing algorithm for horizontal move for lines m mod 3 = 1 [Move from left to right, if ($X_s = X_d$ && $Y_s < Y_d$)]
Case: 2 Fault tolerant routing algorithm for horizontal move for lines m mod 3 = 1 [Move from right to left, if ($X_s = X_d$ && $Y_s > Y_d$)]
Case: 3 Fault tolerant routing algorithm for horizontal move for lines m mod 3 ≠ 1 [Move from left to right, if ($X_s = X_d$ && $Y_s < Y_d$)]
Case: 4 Fault tolerant routing algorithm for horizontal move for lines m mod 3 ≠ 1 [Move from right to left, if ($X_s = X_d$ && $Y_s > Y_d$)]

3. FAULT TOLERANT ROUTING ALGORITHM

3.1. Description of Model

We have derived the fault tolerance routing algorithm for octagon-cell interconnected network for horizontal moving messages. Here it is assumed that each node has information about its three consecutive nodes on the original optimal path in which it could have gone if there won't be any faulty nodes, link failures and dead end state. Each node say 'A' checks its three consecutive nodes and links simultaneously along its original path. If any error occurs in one of three consecutive links or nodes, then the algorithm will work with respect to the address of source node at 'A'.

When a signal passes from a source node to a destination, it is very much essential to find a path of non-faulty nodes. For this purpose, each node can store the information about its three consecutive nodes and links along the original path. There are three possible cases for a node in an octagon-cell network. That is:

a) There are fault free nodes / links along the original path. This is called normal state.
b) If any faulty node or link occurs in the original path, this situation is called faulty state. This situation can be handled by the nodes along the path, because each node in the path has the status of its three consecutive nodes and links. So the original path will be changed by using the algorithm.
c) One link has two nodes. If two nodes of a link are faulty, then this situation is called dead end state.

NOTE:

1. Since we have derived the fault tolerant scheme for the presence of faulty node or faulty link along the optimal path, so in all cases the link failure conditions have not been mentioned, because the algorithm for link failure case is equivalent to the node failure along the optimal path. That is if the link $(x_s, y_s) \rightarrow (x_s^*, y_s^*)$ is failed along the optimal path, then this situation is equivalent to the left node failure of this link. If the right node is failed, then this is equivalent to the link failed connecting to (x_s^*, y_s^*) and (x_s^{**}, y_s^{**}).

2. The symbol "d" represents depth of the network and the word we have used in our algorithm "w.r.t" represents "with respect to".

3.2. Description of the Fault Tolerant Scheme

When a message is to be sent from a source node to a destination node, the algorithm first finds the optimal path [1]. The message moves on that optimal path. In that path each and every node has the status about three consecutive nodes / links. If 'B' be the faulty node and 'A' be another fault free node on that path, then before reaching at 'B', the message first reaches at 'A'. 'A' has the status about next three consecutive nodes / links. 'B' is the node amongst the three consecutive nodes. So at that situation, the message suddenly goes away from that original optimum path [1] and finds another fault free path with respect to node 'A'. Else dead end may occur for which the message fails to reach at the destination. Let the optimal path be A→B→C→D→E→F, where the source node is 'A' and the destination node is 'F'. If 'F' is faulty then dead end occurs. If 'B' or 'C' or 'D' is faulty, then the algorithm will work with respect to the node 'A'. If 'E' is faulty, then the algorithm will work with respect to the node 'B'. If the optimal path be A→B→C→D, where the source and destination nodes are 'A' and 'D' respectively and if faulty node is 'B' or 'C', then in this case the algorithm will work w.r.t the source node 'A'.

3.3. Fault Tolerant Algorithm

The fault tolerant algorithm is based on the optimal routing scheme in octagon-cell network [1]. Here four main cases have been derived for fault tolerant scheme.

Case-1 Fault tolerant routing algorithm for horizontal move for lines m,
Where m mod 3 = 1 [move from left to right, if ($X_s = X_d$ && $Y_s < Y_d$)]

If ($X_s < [(d*5) + (d-2)]$)

Sub case: 1-A1

If [(the original source node y_s is odd) && [faulty node at (x_s, y_s+1) || (faulty node occurs in line x_s+1) || at (x_s, y_s+2)) with respect to original source node || Failure of any of 3 consecutive links]]

Step-1 Move Vertical Top to Bottom till $(x_s = x_s+3, y_s)$

Step-2 If y_d is even

Move Horizontal Right till $(y_s = y_d)$

Move Vertical Top till $(x_s = x_d)$

Else if y_d is odd

Move Horizontal Right till $(y_s = y_d - 1)$

Move $(x_s - 1, y_s/2 + 1, x_d, y_d)$

Move $(x_s - 1, y_s, x_d, y_d)$

Move (x_s- 1, 2y_s-1, x_d, y_d)

Sub case: 1-B1

Else if (Original source node y_s is even) && (faulty node is any of 3 consecutive neighbors of original source node)

Step-1 Move Horizontal Left till (y_s = y_s-1) && Go to Step-1 and Step-2 of Sub case 1-A1

Sub case: 1-C1

Else if (faulty node = x_s+1 w.r.t the node (x_s, y_s), where (x_s,y_s) is any node on the path other than the original source node) && (faulty node is not the neighbor of (x_d, y_d))

Step-1 Move Vertical to Bottom till (x_s = x_s +1)

Step-2 Move (x_s+1, 2y_s – 1, x_d, y_d)

Step-3 Go to Step-2 of Sub case 1-A1

Sub case 1-D1

Else if faulty node (x_s = x_d && y_s is odd or even w.r.t their respective nodes)

Step-1 Go to Sub case-1-A1

Sub case 1-E1

Else if (y_d is even) && (faulty node is neighbor of (x_d, y_d) i.e. at the line x_d) && (faulty node is w.r.t the node which is other than the original source node)

Step-1 Move (x_s+1, y_s, x_d, y_d)

Step-2 Move Vertical Bottom till (x_s = x_s+3)

Step-3 Move Horizontal Right till (y_s = y_d)

Step-4 Move Vertical Top till (x_s = x_s+3)

Sub case 1-F1

Else if (y_d is odd) && (faulty node is neighbor of y_d i.e. at the line x_d+1) && (faulty node is w.r.t the node which is other than the original source node)

Step-1 Move (x_s+1, y_s, x_d, y_d)

Step-2 Move (x_s+1, 2y_s-1, x_d, y_d)

Step-3 Move Horizontal Right till (y_s = y_d+1)

Step-4 Move Vertical Top till (x_s = x_s-3)

Step-5 Move Horizontal Left till (y_s = y_s-1)

Else if (X_s = [(d*5) + (d-2)])

Sub case: 1-A2

If (the original source node y_s is odd) && [faulty node at (x_s, y_s+1) || faulty node occurs in line (x_s-1) || at (x_s, y_s+2) with respect to original source node) || Failure of any of 3 consecutive links]

Step-1 Move Vertical Top till (x_s = x_s-3, y_s)

Step-2 If y_d is even

Move Horizontal Right till ($y_s = y_d$)
Move Vertical Bottom till ($x_s = x_d$)
Else if y_d is odd
Move Horizontal Right till ($y_s = y_d - 1$)
Move ($x_s + 1$, $y_s/2 + 1$, x_d, y_d)
Move ($x_s + 1$, y_s, x_d, y_d)
Move ($x_s + 1$, $2y_s - 1$, x_d, y_d)

Sub case 1-B2

Else if (Original source node y_s is even) && (faulty node is any of 3 consecutive neighbors of original source node)
Step-1 Move Horizontal Left till ($y_s = y_s - 1$)
Step-2 Go to Step -2 of Sub case 1-A2

Sub case: 1-C2

Else if (faulty node = $x_d - 1$ w.r.t the node (x_s, y_s), where (x_s, y_s) is any node on the path other than the original source node) && (faulty node is not the neighbor of (x_d, y_d))
Step-1 Move ($x_s - 1$, y_s, x_d, y_d)
Step-2 Move ($x_s - 1$, $2y_s - 1$, x_d, y_d)
Step-3 Go to Step-2 of Sub case 1-A2

Sub case 1-D2

Else if faulty node ($x_s = x_d$ && y_s is odd or even w.r.t their respective nodes)
Step-1 Go to Sub case-1-A2

Sub case 1-E2

Else if (y_d is even) && (faulty node is neighbor of (x_d, y_d) i.e. at the line x_d) && (faulty node is w.r.t the node which is other than the original source node)

Step-1 Move Vertical Top till ($x_s = x_s - 3$)
Step-2 Move Horizontal Right till ($y_s = y_d$)
Step-3 Move Vertical Bottom till ($x_s = x_s + 3$)

Sub case 1-F2

Else if (y_d is odd) && (faulty node is neighbor of y_d i.e. at the line $x_d - 1$) && (faulty node is w.r.t the node which is other than the original source node)
Step-1 Move ($x_s + 1$, y_s, x_d, y_d)
Step-2 Move ($x_s + 1$, $2y_s - 1$, x_d, y_d)
Step-3 Move Horizontal Right till ($y_s = y_d + 1$)
Step-4 Move Vertical Bottom till ($x_s = x_s + 3$)
Step-5 Move Horizontal Left till ($y_s = y_s - 1$)

NOTE: We have explained the optimal routing schemes in all directions in [1]. So the Horizontal moves and Vertical moves follow the algorithms in [1].

Example: Let $(X_s, Y_s) = (1, 3)$, $(X_d, Y_d) = (1, 8)$ and the faulty node at (1,4).

Using the algorithm of Horizontal moves [1], We have the optimal path to reach the destination is : (1,3) →(1,4) →(2,3) →(1,5) →(1,6) → (2,4) →(1,7) →(1,8). The shortest path length is 7. Now using the above algorithm we have the following fault free path.
(1,3) →(2,2) →(3,2) →(4,3) →(4,4)→(5,3) →(4,5) →(4,6) → (5,4) →(4,7) →(4,8)→(3,5) →(2,5) →(1,8). The path length is 13.

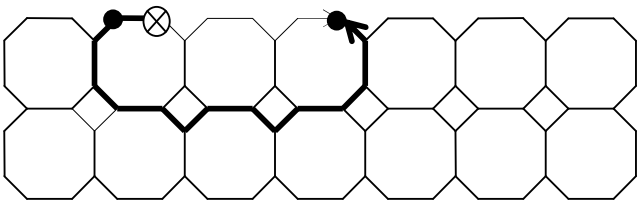

Figure 3: [we have drawn only first 7 lines of the Octagon-Cell network of depth 4, which shows the fault free path]

Case: 2 Fault tolerant routing algorithm for horizontal move for lines m, Where m mod 3 = 1 [Move from right to left, if ($X_s = X_d$ && $Y_s > Y_d$)]
If ($X_s < [(d*5) + (d-2)]$)

Sub case 2-A1

If [(the original source node y_s is even) && ((faulty node at (x_s,y_s-1) || faulty node occurs in line x_s+1|| at (x_s, y_s-2)) with respect to original source node || failure of any of 3 consecutive links]
Step-1 Move Vertical Bottom till (x_s= x_s +3)
Step-2 If y_d is even
Move Horizontal Left till ($y_s = y_d$ -1)
Move (x_s-1, y_s/2 +1, x_d, y_d)
Move (x_s-1, y_s, x_d, y_d)
Move (x_s-1, 2y_s-1, x_d, y_d)
Move (x_s, y_s + 1, x_d, y_d)
Else if y_d is odd
Move Horizontal Left till ($y_s = y_d$)
Move Vertical Top till ($x_s = x_d$)

Sub case: 2-B1

Else if (Original source node y_s is odd) && (faulty node is any of 3 consecutive neighbors of original source node)
Step-1 Move (x_s, y_s+1, x_d, y_d)
Go to Sub case: 2-A1

Sub case: 2-C1

Else if (faulty node = x_s+1 w.r.t the node (x_s, y_s), where (x_s, y_s) is any node on the path other than the original source node)

Step-1 Move Vertical Bottom till ($x_s = x_s + 1$)
Step-2 Move (x_s+1, $2y_s - 2$, x_d, y_d)
Step-3 Go to step-2 of sub case: 2-A1

Sub case 2-D1

Else if faulty node ($x_s = x_d$) && (y_s is odd or even w.r.t their respective nodes)
Step-1 Go to Sub case-2-A1

Sub case 2-E1

Else if (y_d is even) && (faulty node is neighbor of y_d i.e. at the line x_d+1) && (faulty node is w.r.t the node which is other than the original source node)

Step-1 Move (x_s+1, y_s, x_d, y_d)
Step-2 Move (x_s+1, $2y_s$-2, x_d, y_d)
Step-3 Move Horizontal to Left till ($y_s = y_d - 1$)
Step-4 Move Vertical Top till ($x_s = x_s - 3$)
Step-5 Move Horizontal right till ($y_s = y_s + 1$)

Sub case 2-F1

Else if (y_d is odd) && (faulty node is neighbor of y_d i.e. at the line x_d) && (faulty node is w.r.t the node which is other than the original source node) && (faulty node is w.r.t the node which is other than the original source node)

Step-1 Move Vertical Bottom till ($x_s = x_s$+3)
Step-2 Move Horizontal Left till ($y_s = y_d$)
Step-3 Move Vertical Top till ($x_s = x_s$- 3)

Else if (X_s = [(d*5) + (d-2)])

Sub case 2-A2

If [(the original source node y_s is even) && (faulty node at (x_s, y_s-1) || faulty node occurs in line (x_s-1) || at (x_s, y_s-2) with respect to original source node)]

Step-1 Move Vertical Top till ($x_s = x_s$-3, y_s)
Step-2 If y_d is even
Move Horizontal Left till ($y_s = y_d - 1$)
Move (x_s+1, y_s/2 +1, x_d, y_d)
Move (x_s+1, y_s, x_d, y_d)
Move (x_s+1, $2y_s$-1, x_d, y_d)
Move Horizontal Left till ($y_s = y_d$)
Else if (y_d is odd)
Move Horizontal Left till ($y_s = y_d$)
Move Vertical Bottom till ($x_s = x_d$)

Sub case 2 –B2

Else if (Original source node y_s is odd) && (faulty node is any of 3 consecutive neighbors of original source node)

Step-1 Move Horizontal Right till ($y_s = y_s+1$)

Step-2 Go to Sub case 2-A2

Sub case 2-C2

Else if (faulty node = x_s-1 w.r.t the node (x_s, y_s), where (x_s, y_s) is any node on the path other than the original source node)

Step-1 Move Vertical Top till ($x_s = x_s$-1)

Step-2 Move (x_s-1, $2y_s - 2$, x_d, y_d)

Step-3 Move step 2 of Sub case 2-A2

Sub case 2-D2

Else if faulty node ($x_s = x_d$) && (y_s is odd or even w.r.t their respective nodes)

Step-1 Go to Sub case-2-A2

Sub case 2-E2

Else if (y_d is even) && (faulty node is neighbor of y_d i.e. at the line x_d-1) && (faulty node is w.r.t the node which is other than the original source node)

Step-1 Move (x_s-1, y_s, x_d, y_d)

Step-2 Move (x_s-1, $2y_s$-2, x_d, y_d)

Step-3 Move Horizontal Left till ($y_s = y_d$ -1)

Step-4 Move Vertical Bottom till ($x_s = x_d$)

Step-5 Move Horizontal Right till ($y_s = y_d$)

Sub case 2-F2

Else if (y_d is odd) && (faulty node is neighbor of y_d i.e. at the line x_d) && (faulty node is w.r.t the node which is other than the original source node)

Step-1 Move Vertical Top till ($x_s = x_s$-3)

Step-2 Move Horizontal Left till ($y_s = y_d$)

Step-3 Move Vertical Bottom till ($x_s = x_s$+3)

Example: Let (X_s, Y_s) = (4, 8), (X_d, Y_d) = (4, 2) and the faulty node at (5,3) with depth 5.

Using the algorithm of Horizontal moves in [1], If all the nodes are fault free, then we have the optimal path to reach the destination is : (4,8) →(4,7) →(5,4) →(4,6) →(4,5) → (5,3) →(4,4) →(4,3) →(5,2) →(4,2). The shortest path length is 9.

Now using the above algorithm we have the following fault free path.
(4,8) →(4,7) →(5,4) →(6,4)→(7,6) →(7,5) →(8,3) →(7,4) →(7,3) →(8,2)→(7,2) →(7,1) →(6,1) →(5,1) →(4,1) →(4,2). The path length is 15.

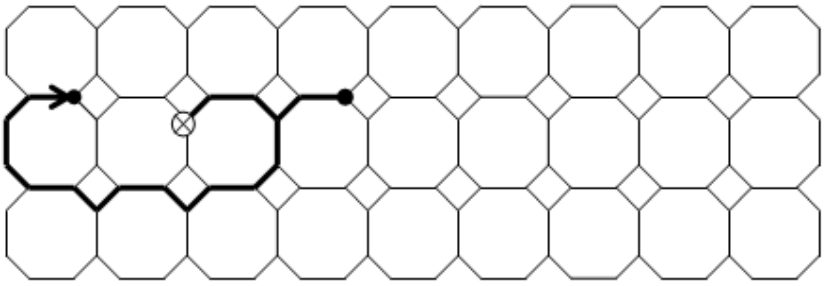

Figure 4: [we have drawn only first 10 lines of the Octagon-Cell network of depth 5, which shows the fault free path]

Case: 3 Fault tolerant routing algorithm for horizontal move for lines m, Where m mod 3 ≠ 1 [Move from left to right, if ($X_s = X_d$ && $Y_s < Y_d$)]

Sub case-3A1

If (faulty node is any two consecutive node of original source node)
If (x_s mod 3 = 2) {
Step-1 Move (x_s + 1, $\mathbf{y}_s$, x_d, y_d)
Step-2 Move Horizontal Right till ($y_s = y_d$)
Step-3 Move Vertical Top till ($x_s = x_d$)}
Else If (x_s = 3n) {
Step-1 Move (x_s - 1, $\mathbf{y}_s$, x_d, y_d)
Step-2 Move Horizontal Right till ($y_s = y_d$)
Step-3 Move Vertical Bottom till ($x_s = x_d$)}

Sub case-3B1

If (faulty node is along the original source node) || (faulty node is the 3rd consecutive w.r.t original source node)
If (x_s mod 3=2) && (original source node x_s =2)
{
Go to Subcase-3A1 (If case)
Else If ($x_s > 2$)
Step-1 Move (x_s - 1, 2y_s-1, x_d, y_d)
Step-2 Move (x_s, y_s+1, x_d, y_d)
Step-3 Move (x_s -1, y_s/2 +1, x_d, y_d)
Step-4 Move (x_s + 1, 2y_s-1, x_d, y_d)
Step-5 Move (x_s, $\mathbf{y}_s$+1, x_d, y_d)
Step-6 Move (x_s +1, y_s/2 +1, x_d, y_d)
Else If ($x_s = x_d$) && ($y_s = y_d$)
Destination reached.
Else Move Horizontal Right till destination reached
}
Else If (x_s = 3n)
{
If (x_s< [(d*5) + (d-2)]-1)

{
Step-1 Move (x_s+1, 2y_s-1, x_d, y_d)
Step-2 Move (x_s. y_s+1, x_d, y_d)
Step-3 Move (x_s+1, y_s/2+1, x_d, y_d)
Step-4 Move (x_s-1, 2y_s-1, x_d, y_d)
Step-5 Move (x_s, y_s+1, x_d, y_d,)
Step-6 Move (x_s-1, y_s/2+1, x_d, y_d)
Else If (x_s = x_d) && (y_s = y_d)
Destination reached
Else Move Horizontal Move to Right till (y_s = y_d) && (x_s = x_d)
}
Else If (x_s = [(d*5) + (d-2)]-1)
{
Step-1 Move (x_s-1, y_s, x_d, y_d)
Step-2 Move Horizontal Right till (y_s = y_d)
Step-3 Move Vertical Bottom till (x_s = x_d)
}
}

Sub case-3C1

If (x_s mod 3 = 2) && (faulty node is at x_s-1, where x_s is original source node) && (faulty node is w.r.t the node which is other than the original source node)

{
If (faulty node y_s is even)
Step-1 Move (x_s+1, y_s/2+1, x_d, y_d,)
Go to Subcase-3A1 (If case)
Else If (faulty node y_s is odd)
Step-1 Move (x_s, y_s+1, x_d, y_d)
Go to Subcase-3C1 (If case)
}

Else If (x_s = 3n) && (faulty node x_s is on the line x_s+1) && (faulty node is w.r.t the node which is other than the original source node)

{
If (faulty node y_s is even)
Step-1 Move (x_s-1, y_s/2 +1, x_d, y_d)
Go to Subcase-3A1 (Else case)
Else If (faulty node y_s is odd)
Step-1 Move (x_s, y_s+1, x_d, y_d)
Go to above case (for y_s even)
}

Sub case-3D1

If (faulty node is neighbor of (x_d, y_d) || neighbor of neighbor of (x_d, y_d)) && (faulty node is w.r.t the node which is other than the original source node)

If (x_s mod 3=2)
Go to Subcase-3C1 (for x_s mod 3 = 2)

Else If (x_s = 3n)
Go to Subcase-3C1 (for x_s = 3n)

Example: Let (X_s, Y_s) = (2, 3), (X_d, Y_d) = (2, 8) and the faulty node at (1, 10) with depth 4.

Using the algorithm of Horizontal moves [1], We have the optimal path to reach the destination is : (2,3) →(1,5) →(1,6) →(2,4) →(1,7) → (1,8) →(2,5) →(1,9) →(1,10) →(2,6) →(1,11) →(1,12) →(2,7) → (1,13) →(1,14) →(2,8) The shortest path length is 15.

Now using the above algorithm we have the following fault free paths.

(2,3) →(1,5) →(1,6) →(2,4) →(1,7) → (1,8) →(2,5) →(3,5) →(4,9) →(4,10) →(3,6) →(4,11) →(4,12) → (3,7) →(4,13) →(4,14) →(3,8) →(2,8). The path length is 17.

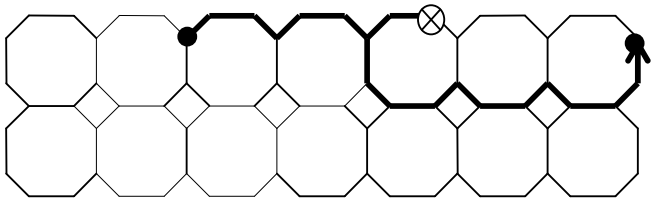

Figure 5: [we have drawn only first 7 lines of the Octagon-Cell network of depth 4, which shows the fault free path]

Case: 4 Fault tolerant routing algorithm for horizontal move for lines m, Where m mod 3 ≠ 1 [Move from right to left, if ($X_s = X_d$ && $Y_s > Y_d$)]

Sub case-4A1

If (faulty node is any of two consecutive nodes of original source node)
If (x_s mod 3 = 2) {
Step-1 Move (x_s + 1, $\mathbf{y}_s$, x_d, y_d)
Step-2 Move Horizontal Left till ($y_s = y_d$)
Step-3 Move Vertical Top till ($x_s = x_d$) }
Else {
If (x_s = 3n)
Step-1 Move (x_s - 1, $\mathbf{y}_s$, x_d, y_d)
Step-2 Move Horizontal Left till ($y_s = y_d$)
Step-3 Move Vertical Bottom till ($x_s = x_d$)}

Subcase-4B1

If (faulty node is along the line on original source node) || (faulty node is 3rd consecutive node w.r.t original source node)

If (x_s mod 3=2) && (original source node x_s =2) {
Go to Subcase-4A1 (If case)
Else

If ($x_s > 2$)
Step-1 Move (x_s - 1, 2y_s-2, x_d, y_d)
Step-2 Move (x_s, y_s-1, x_d, y_d)
Step-3 Move (x_s -1, y_s/2 +1, x_d, y_d)
Step-4 Move (x_s + 1, 2y_s-2, x_d, y_d)
Step-5 Move (x_s, **y**$_s$-1, x_d, y_d)
Step-6 Move (x_s +1, y_s/2 +1, x_d, y_d)
Else If ($x_s = x_d$) && ($y_s = y_d$)
Destination reached.
Else Move Horizontal Left till destination reached}
Else If ($x_s = 3n$)
{ If (x_s< [(d*5) + (d-2)]-1)
{
Step-1 Move (x_s+1, 2y_s-2, x_d, y_d)
Step-2 Move (x_s, y_s-1, x_d, y_d)
Step-3 Move (x_s+1, y_s/2+1, x_d, y_d)
Step-4 Move (x_s-1, 2y_s-2, x_d, y_d)
Step-5 Move (x_s, y_s-1, x_d, y_d)
Step-6 Move (x_s-1, y_s/2+1, x_d, y_d)
Else If ($x_s = x_d$) && ($y_s = y_d$)
Destination reached
Else Move Horizontal Left till ($y_s = y_d$)
}

Else If (x_s = [(d*5) + (d-2)]-1)
{
Step-1 Move (x_s - 1, y_s, x_d, y_d)
Move Horizontal Left till ($y_s = y_d$)
Move Vertical Bottom till ($x_s = x_d$)
}
}

Sub case-4C1

If (x_s mod 3 = 2) && (Faulty node is at x_s-1, x_s is original source node) && (faulty node is w.r.t the node which is other than the original source node)
{
If (faulty node y_s is odd) {
Step-1 Move (x_s+1, y_s/2+1, x_d, y_d)
Go to Sub case - 4A1 (If case)}
Else If (faulty node y_s is even) {
Step-1 Move (x_s, y_s-1, x_d, y_d)
Go to Subcase-4C1 (If case i.e. odd condition)}
}

Else If (x_s =3n) && (Faulty node x_s is on the line x_s+1) && (faulty node is w.r.t the node which is other than the original source node)
{
If (faulty node y_s is odd) {
Step-1 Move (x_s-1, y_s/2 +1, x_d, y_d)

Go to Sub case - 4A1 (Else case)}
Else If (faulty node y_s is even) {
Step-1 Move (x_s, y_s-1, x_d, y_d)
Go to Sub case - 4C1 (for x_s =3n, y_s is odd)
}

Sub case-4D1

If (faulty node is neighbor of (x_d, y_d)) || (neighbor of neighbor of (x_d, y_d))
{
If (x_s mod 3=2)
Go to Subcase-4C1 (for x_s mod 3 = 2)
Else If (X_s = 3n)
Go to Subcase-4C1 (for x_s = 3n)
}

Example: Let (X_s, Y_s) = (5, 9), (X_d, Y_d) = (5, 6) and with link failure at (5, 7) → (4,12) with depth 4. This case is same with when the faulty node is (5, 7). The algorithm will work w.r.t the node (5, 8).

Using the algorithm of Horizontal moves [1], We have the optimal path to reach the destination is : (5,9) →(4,16) →(4,15) →(5,8) →(4,14) → (4,13) →(5,7) →(4,12) →(4,11) →(5,6). The shortest path length is 9.

Now using the above algorithm w.r.t the node (5, 8) we have the following fault free paths.

(5,9) →(4,16) →(4,15) →(5,8) →(4,14) → (4,13) →(3,7) →(4,12) →(4,11) →(5,6). The path length is 9.

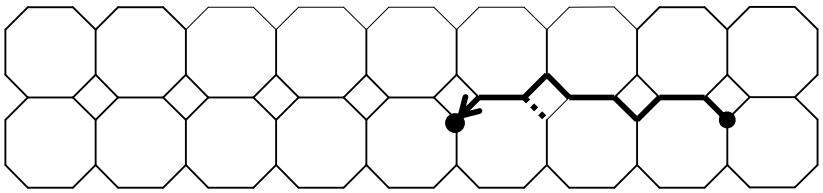

Figure 6: [we have drawn only first 7 lines of the Octagon-Cell network of depth 5, which shows the fault free path]

4. CONCLUSION

Octagon-Cell interconnected network has attractive features like small network diameter, better bisection width and constant node degree etc. This paper introduces a fault tolerant routing algorithm for horizontal moving messages in the Octagon-Cell interconnected network topology. If any system can't solve the faulty problem, it is considered as unreliable and inefficient. We have already described the algorithm in [1], to find optimal path in any direction from source node to destination node in Octagon-Cell network. In our fault tolerant scheme, the optimal path is calculated from source to destination node for horizontal moving messages. If any fault occurs

along the path, the algorithm will find the alternative fault free path. This can be utilized in mobile networks and different wide range networks.

REFERENCES

[1] Sanjukta Mohanty and Prafulla Ku. Behera, (2011) "Optimal Routing Algorithm in a Octagon-Cell Network" *International Journal of Advanced Research in Computer Science*, Vol. 2, No.5 pp.625-637.

[2] Ahmad Sharieh, Mohammad Qatawneh, Wesam Almobaideen, Azzam Sleit, (2008) "Hex-Cell: Modeling, Topological Properties and Routing Algorithm", *European Journal of Scientific Research*, Vol.22, No.2, pp. 457-468.

[3] N. Gopalakrishna Kini, M. sathish Kumar and Mruthyunjaya H.S.,(2009) "An Optimal Data Routing Scheme for Mesh Embedded Hypercube Interconnection network with Multiple Faulty Nodes", *International Journal of Computer Science and Engineering* , 3:1 , pp.26-30.

[4] Ahmed Louri and Hongki Sung, (April 1994) "An Optical Multi-Mesh Hypercube: A Scalable Optical interconnection Network for Massively Parallel Computing", *Journal of Lightwave Technology*, Vol.12, No.4, pp. 704-716.

[5] Tze Chiang Lee and John P.Hayes, (Oct 1992) "A Fault-Tolerant Communication Scheme for Hypercube Computers", *IEEE Transactions on Computers*, Vol.41, No.10, pp. 1242-1256.

[6] Sandeep Sharma, P.K.Bansal,(2002) " A New Fault Tolerant Multistage Interconnection Networks", *IEEE TENCON*' 02, Vol.1, pp.347-350.

[7] Shih-Chang Wang and Sy-Yen Kuo, (Dec. 1994) "Fault Tolerance in Hyperbus and Hypercube Multiprocessors Using Partitioning Scheme", *IEEE International Conference on Parallel and Distributed Systems*, pp. 340-347.

[8] D. Nassimi and S. Sahni, Jan. (1980) " An Optimal Routing Algorithm for Mesh Connected Parallel Computers", *Journal of the ACM*, Vol.27, pp.6-29.

[9] Shen. J.P.,(1982) " Fault Tolerance Analysis of Several Interconnection Networks", *Proceedings of International Conference on Parallel Processing*, pp.102-112.

[10] Jehoshua Bruck, Member, IEEE, Robert Cypher, and Danny Soroker, (May 1992) "Tolerating Faults in Hypercubes Using Subcube Partitioning", *IEEE Trasactions on Computers*, Vol.41, No.5, pp.599-605.

[11] K.V. Arya, R.K. Ghosh. (2005) "Designing a New Class of Fault Tolerant Multistage Interconneconnection Networks", *Journal of Interconneconnection Network*, Vol.6, No.4, pp.361-382.

[12] Kuo-Hsuan Chen and Ge-Ming Chiu, (1998) "Fault-Tolerant Routing Algorithm for Meshes", *Journal of Information Science and Engineering*, Vol.14, pp.765-783.

[13] Jianer Chen, Iyad A. Kanj and Guojun Wang, (2002) " Hypercube Network Fault Tolerance: A Probabilistic Approach", *Proceedings of the IEEE International Conference on Parallel Processing.*

[14] M.J. Serrano and B. Parhami, (Oct. 1993) "Optimal Architectures and Algorithms for Mesh-Connected Parallel Computers with Separable Row/Column Buses", *IEEE Trasactions on Parallel and Distributed Systems*, Vol.4, No.10, pp.1073-1080.

[15] Mohammad Qatawneh, Bdour Hamed, Wesam AlMobaideen, Azzam Sleit, Amal Qudat, Wala'a Qutechat, Roba Al-Soub, (Dec. 2009) " FTRH: Fault Tolerance Routing Algorithm for Hex-Cell Networks", *International Journal of Computer Science and Network Security*, Vol.9, No.12, pp. 268-274.

[16] K. Day, K.S. Harous, A. Al-Ayyoub, (2000) "A Fault Tolerant Routing Scheme for Hypercubes", *Telecommunication Systems*, Vol-13, No.1, pp. 29-44.

[17] Catherine Decayeux, David Seme, (2005) "3D Hexagonal Network: Modeling, Topological Properties, Addressing Scheme and Optimal Routing Algorithm", *IEEE Trasactions on parallel and systems*, Vol-16, No.9, pp.875-884.

[18] Qatawneh Mohammad (2006) "Adaptive Fault Tolerant Routing Algorithm for Tree-Hypercube Multicomputer", *Journal of Computer Science*, Vol.2, No.2, pp. 124-126.

[19] Suresh Chalasani, Rajendra V. Boppana, (1994) "Fault-Tolerant Wormhole Routing in Tori", *International Conference on Supercompuing*, , pp.146-155.

[20] Mahmoud Al-Omari, Mohammed Mahafzah, (1999) "Fault-Tolerant Routing in hypercubes using masked interval Routing Scheme", *Proceedings of the 1999 ACM symposium on Applied Computing*, pp.481-485.
[21] Jipeng Zhou and Francis C.M. Lau, (2004) "Multi-Phase Minimal Fault-Tolerant Wormwhole Routing in Meshes", *Parallel Computing*, Vol.30, No.3, pp. 423-442.
[22] Jong-Hoon Youn, Bella Bose, Seungjin Park, (2006) "Fault-Tolerant Routing Algorithm in Meshes with solid faults", *Journal of Supercomputing*, 37, pp. 161-177.
[23] Flaviu Cristian, Bob Dancey, Jon Dehn, (1996) "Fault-Tolerance in Air Traffic Control Systems", *Trasaction on Computer Systems (TOCS)*, Vol.14, No.3, pp. 256-286.
[24] Dajin Wang, (2001) "A Low Cost Fault-Tolerant Structure for Hypercube", *Journal of Supercomputing*, 20, pp. 203-216.
[25] Dan Gordon, Israel Koren, Gabriel M. Silberman, (1987) "Reconstructing Hexagonal Arrays of Processors in the Presence of Faults", *Journal of VLSI and Computer Systems*, Vol.2, pp. 23-35.
[26] Mostafa Rezazad, Hamid Sarbazi-Azad, (2003) "A Routing Algorithm for Star Interconnection Network in the Presence of Faults", *The CSI Journal on Computer Science and Engineering,* Vol.1, No. 4(b), pp. 11-18

Authors

Sanjukta Mohanty has received her Master in Computer Application from North Orissa University of Odisha, India, M.Sc in Mathematics from Fakir Mohan University of Odisha, India, M.Tech (Comp.Sc & Engg) from North Orissa University of Odisha, India in 2013 and Continuing Ph.D in Computer Science & IT in North Orissa University Sriram Chandra Vihar, Baripada of Odisha, India. Her research interests include on the study of routing algorithms in interconnected networks.

Dr. Prafulla Ku. Behera has received his Ph.D in Computer Science from Utkal University of Odisha, India in 2007. His research interests include on the study of Mobile Ad-hoc Networks. He is the faculty member of Department of Computer Science & Application, Utkal University, Vani Vihar, Bhubaneswar of Odisha, India.