An Alternative Routing Algorithm for Hex-Cell Network

Mohammad Qatawneh, Hamed Bdour, Shrouq Sabah, Rana Samhan, Azzam Sleit, Jafar Alqatawna, Wesam al-Mobaideen

King Abdullah II School for Information Technology, University of Jordan,
P.O. Box 13898, Amman 11942, Jordan
{mohd.qat, azzam.sleit, j.alqatawna, wes.moba}@ju.edu.jo

Abstract

In distributed systems, the underlying communication network and the routing algorithm affect the overall performance of the computing nodes. Although the Hex-Cell network represents an interesting network topology with features such as the capability of embedding several topological structures and extensibility at a minimal cost; yet it needs an efficient routing algorithm in order to fully benefit from these interesting features. This paper proposes a new routing algorithm for Hex-Cell network. This routing algorithm is based on a new addressing mode which utilises the addresses of the two neighbours’ cells of each source and destination nodes. The proposed routing algorithm uses two levels of routing: internal Node Routing which operates between the nodes within the cell and external Cell Routing which route messages from cell to cell. This concept is very useful with structure of Hex-Cell network specifically for grouping (clustering) six processors as one component (Cell) and there are central processor which organises these processors; therefore, Cell Routing can be used for routing from one group to another group, and for internal routing each group may use the proposed Node Routing or any another centre or distribute routing algorithm.

Keywords: Hex-Cell, Routing Algorithm, Interconnection topology.

1. Introduction

The key component of high performance distributed system architecture is the interconnection network topology which enables the communication among different nodes of a large distributed system [6, 7]. Several interconnection network topologies have been proposed [1, 2, 3, 4, 5, 8, 9]. One of the new and efficient network topology is the hex-cell network [1]. Despite the interesting features of the Hex-Cell network its current routing algorithm which has been proposed in our earlier work has its own limitations. In fact it depends on the depth of Hex-Cell network which mean that the topology must be fixed or there is a need for updating the depth value and readdressing the nodes every time a new level is added. In fact this is because the current routing algorithm is build upon an addressing mode (discussed in the next subsection) which uses the network depth as building block for identifying the location of the nodes in the Hex-Cell network. This motivated us to propose a new addressing mode which could allow us write more efficient routing algorithm that does not depend on the depth of the Hex-Cell network.
This paper proposes a new routing algorithm for Hex-Cell network, which can utilise the capability of clustering, by splitting the routing algorithm into two levels. The first one is internal cell routing (between nodes in the cell), which we call node routing and the second is external cell routing (between cells) which we call cell routing. One of the main advantages of the proposed routing algorithm is that it does not need readdressing for nodes as the number of levels increased. Moreover, it requires less computational power than the current routing algorithm.

The rest of the paper is organized as follows. Section 2 explains the Hex-Cell interconnection network and its routing algorithm. Section 3 presents the proposed routing algorithm for Hex-Cell network with several illustrative examples. Section 4 provides an evaluation by comparing the new algorithm with the current one in terms of performance and flexibility. Finally, section 5 concludes this article with future research avenues.

2. Hex–Cell Network Topology and Current Routing Algorithm

A Hex-Cell network [1] with depth d is denoted by HC(d) and can be constructed by using units of hexagon cells, each of six nodes. A Hex-Cell network with depth d has d levels numbered from 1 to d, where, level 1 represents the innermost level corresponding to one hexagon cell. Level 2 corresponds to the six hexagon cells surrounding the hexagon at level 1. Level 3 corresponds to the 12 hexagon cells surrounding the six hexagons at level 2 as shown in Fig.1. The levels of the HC(d) network are labeled from 1 to d. Each level i has Ni nodes, representing processing elements and interconnected in a ring structure.

![Fig.1. (a) HC (one level) (b) HC (two levels) (c) HC (three levels)](image)

This architecture has some of the interesting characteristics of hypercube, binary tree, linear array, star and 2D-torus [6, 9]. The degree and total number of links of the HC is less than those of hypercube. In hypercube [10] the node degree of each node is a logarithmic function of the total number of nodes, which is a drawback of the topology. Consequently, the hypercube topology is not a good candidate for
interconnection network for a very large parallel computer that might contain hundreds of thousands of nodes due to limitations concerning integrated circuit technology and port number. The proposed topology has the ability to efficiently simulate programs written for architectures such as linear arrays, rings and meshes. The Hex-Cell topology can easily embed these structures into it.

The current addressing mode of Hex-Cell network uses the level numbering scheme as follows. Each node in the HC is identified by a pair (X,Y), where X denotes the line number in which the node exists, and Y denotes the location of the node in the line as in Fig.2. A node with the address 1.1 is the first node that exists at line number 1. 1.2 refers to the second node that exists at line number 1, and so on.

![Fig.2. Current addressing mode of HC network](image)

The movement from one node to another can be done by using one of three cases (Move horizontal, move Up or move Down) which selected by comparing the line number of both source and destination nodes. If the line number of the source is equal the line number of the destination then move horizontal. If the line number of the source is less than the line number of the destination move up. Finally if the line number of the source is larger than the line number of the destination then move down.

### 3. Proposed Routing Algorithm for Hex-Cell Network

The limitations of the current routing algorithm stem from the fact that it depends on the depth of Hex-Cell network which mean that the topology must be fixed or there is a need for updating the depth value and readdressing the nodes every time a new level is added. The current addressing mode uses the network depth as building block for identifying the location of the nodes in the Hex-Cell network. In contrary, our proposed routing algorithm is more flexible because its calculations and comparisons don not depend on the depth of the network, which implies that adding or removing network levels does not need re-addressing the network nodes. It is based on a new addressing mode which utilizes the addresses of the two neighboring cells of each source and destination nodes. The proposed routing
algorithm uses two levels of routing: internal Node Routing which operates between the nodes within the cell and external Cell Routing which route messages from cell to cell. Section 3.1 describes the proposed addressing mode followed by the Cell Routing in section 3.2 and Node Routing in section 3.3.

3.1 The New Addressing Mode

The building block for the new routing algorithm is an addressing scheme which uses the neighboring cells of each source and destination nodes. This addressing scheme is based on the following key assumptions:

1. Each cell in the HC network is identified by a pair \((X,Y)\), where \(X\) denotes the horizontal, and \(Y\) denotes the vertical as shown in Fig.3.

2. The address of each node in the HC is represented by three parts \((L, C, R)\), where \(L\) is the left cell \([L= (XL, YL)]\), \(C\) is the centre cell in which the node exists \([C= (XC, YC)]\), and \(R\) is the right cell \([R= (XR, YR)]\) as shown in Fig.4 and Fig.5.

3. We take in consideration that, the network border nodes are special case, because one or two of their triple parts are not defined, therefore, we assume a virtual level \((d+1)\) to represent the address of all border nodes as shown in Fig.6.

4. The proposed routing algorithm consists of two parts, cell routing as shown in Fig.7 which route the message from cell to cell and node routing as shown in Fig.12 which route the message from node to node within the cell.

![Cell addressing scheme for HC network](image)
Fig. 4. Node addressing scheme data structure

Fig. 5. Example of Node addressing scheme
Based on the above addressing scheme the proposed routing algorithm is designed to consist two parts, cell routing which route the message from cell to cell and node routing which route the message from node to node within the cell. To describe these two parts of the algorithm specific terminologies shown in Table 1 will be used.

### Table 1. Description of the properties used in the proposed algorithm

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>Next Cell</td>
</tr>
<tr>
<td>LDN</td>
<td>Local Destination Node</td>
</tr>
<tr>
<td>SDN</td>
<td>Source Destination Node</td>
</tr>
<tr>
<td>SN.L</td>
<td>Left Cell of Source Node</td>
</tr>
<tr>
<td>SN.C</td>
<td>Centre Cell of Source Node</td>
</tr>
<tr>
<td>SN.R</td>
<td>Right Cell of Source Node</td>
</tr>
<tr>
<td>DN.L</td>
<td>Left Cell of Destination Node</td>
</tr>
<tr>
<td>DN.C</td>
<td>Centre Cell of Destination Node</td>
</tr>
<tr>
<td>DN.R</td>
<td>Right Cell of Destination Node</td>
</tr>
<tr>
<td>LDN.L</td>
<td>Left Cell of Local Destination Node</td>
</tr>
<tr>
<td>LDN.C</td>
<td>Centre Cell of Local Destination Node</td>
</tr>
<tr>
<td>LDN.R</td>
<td>Right Cell of Local Destination Node</td>
</tr>
<tr>
<td>SDN.L</td>
<td>Left Cell of Local Source Node</td>
</tr>
</tbody>
</table>

### 3.2 Cell Routing

The cell routing is based on the addresses of source and destination cells in which the source and destination nodes exist. The complete algorithm is shown in the following figure followed by a description of the cell routing part supported by examples.
Input(Sn: Source node , Dn: Destination node)
Output (Path)
{
    If(Sn = Dn)
    {
        Path.Add (Sn );
        Path.Add( Dn);
        Return Path;
    }
}

(Sc: Source cell , Dc: Destination cell) $\leftarrow$ CellsOfMinPath(Sn,Dn);
If(Sc = Dc)
Path.Add( NodeRouting(Sn, Dn , Dn));
Else
{
    //Find the next cell
    Xstep $\leftarrow$ Compr(Sc.x , Dc.x);
    C $\leftarrow$ Compr(Sc.y , Dc.y);
    If( C=0 ) Ystep $\leftarrow$ Xstep;
    Else If (C >0 && Xstep =0)
        Ystep $\leftarrow$ 2;
    Else If(C>0 ) Ystep $\leftarrow$ 1;
    Else If(Xstep =0 ) Ystep$\leftarrow$ -2;
    Else Ystep$\leftarrow$ -1;
    Nc $\leftarrow$ (Sc.x + Xstep , Sc.y + Ystep);
    //Move to next cell by NodeRouting to LDn
    LDn $\leftarrow$ SelectLocalDestNode(Sn , Sc , Nc);
    Path.Add( NodeRouting(Sn , LDn , Dn ));
}
Return Path;

Fun : SelectLocalDestNode
Input (Sn , Sc , Nc)
Output(LDn)
{
    Xs $\leftarrow$Nc.x – Sc.x ; Ys $\leftarrow$Nc.y – Sc.y;
    If(Sc= Sn.L  || Sc = Sn.R )  x $\leftarrow$Sn.C.x ;
    Else  x $\leftarrow$ Sn.L.x;
    If(Xs = 0 && Ys=2) LDn $\leftarrow$ (Sc , (x ,Sc.y+1),Nc);
    Else If (Xs = 0 && Ys=-2) LDn $\leftarrow$(Nc , (x , Sc.y-1), Sc);
    Else If((Xs=1 && Ys=1) ||(Xs=-1 && Ys=1))
If(Sn.R = Sc) LDn ← ((Nc.x, Sc.y-1), Sc, Nc);
Else LDn ← (Sc, Nc, (Sc.x, Nc.y+1));
ElseIf((Xs=1 && Ys=-1)OR(Xs=-1 && Ys=-1))
If(Sn.L = Sc) LDn ← (Nc, Sc, (Nc.x, Sc.y+1));
Else LDn ← ((Sc.x, Nc.y-1), Nc, Sc);
Return LDn;
}

Fun : CellsOfMinPath
Input(Sn, Dn)
Output(Sc, Dc)
{
Select source cell and destination cell which have the minimum distance by: **mathematically**: calculate distance between two nodes
\[
\min \left( \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \right)
\]
**OR Logically** as we mention previously in the routing
}

As mentioned previously in addressing scheme, each node has left, centre, and right. The left and right X-axis are the same but the Y-axis is different in two steps. So as result in the following we will deal with the left and the right cells as one unite and the centre cell as another unite.

1) Specify the left, centre and right cells of the source and destination nodes. For example suppose the source node = [(1, -3), (2, -2), (1, -1)], The Left Cell: (1, -3), Centre Cell: (2, -2) Right Cell: (1, -1). The Destination node: [(-1, -1), (0, 0), (-1, 1)], Left Cell: (-1, -1), Centre Cell: (0, 0), Right Cell: (-1, 1) as shown in Fig.8.
1) Specify the source and destination cells in which the source and destination nodes exist as shown in Fig.9.

a. **Mathematically** calculate the distance using \(((X_1 - X_2)^2 + (Y_1 - Y_2)^2)\) formula between two points \((X_1, Y_1)\) and \((X_2, Y_2)\). Apply the distance formula between two cells to find the distance. Find the distances for each cell of source node with each cell of destination node. Select the source cell and destination cell of the minimum distance.

<table>
<thead>
<tr>
<th></th>
<th>SN.L</th>
<th>SN.C</th>
<th>SN.R</th>
</tr>
</thead>
<tbody>
<tr>
<td>DN.L</td>
<td>(((X_{SN.L} - X_{DN.L})^2 + (Y_{SN.L} - Y_{DN.L})^2))</td>
<td>(((X_{SN.C} - X_{DN.L})^2 + (Y_{SN.C} - Y_{DN.L})^2))</td>
<td>(((X_{SN.R} - X_{DN.L})^2 + (Y_{SN.R} - Y_{DN.L})^2))</td>
</tr>
<tr>
<td>DN.C</td>
<td>(((X_{SN.L} - X_{DN.C})^2 + (Y_{SN.L} - Y_{DN.C})^2))</td>
<td>(((X_{SN.C} - X_{DN.C})^2 + (Y_{SN.C} - Y_{DN.C})^2))</td>
<td>(((X_{SN.R} - X_{DN.C})^2 + (Y_{SN.R} - Y_{DN.C})^2))</td>
</tr>
<tr>
<td>DN.R</td>
<td>(((X_{SN.L} - X_{DN.R})^2 + (Y_{SN.L} - Y_{DN.R})^2))</td>
<td>(((X_{SN.C} - X_{DN.R})^2 + (Y_{SN.C} - Y_{DN.R})^2))</td>
<td>(((X_{SN.R} - X_{DN.R})^2 + (Y_{SN.R} - Y_{DN.R})^2))</td>
</tr>
</tbody>
</table>

Example: Source node = [(1, -3), (2, -2), (1, -1)], Destination node: [(-1, -1), (0, 0), (-1, 1)]

<table>
<thead>
<tr>
<th></th>
<th>SN.L: (1, -3)</th>
<th>SN.C: (2, -2)</th>
<th>SN.R: (1, -1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DN.L</td>
<td>(-1, -1)</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>DN.C</td>
<td>(0, 0)</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>DN.R</td>
<td>(-1, 1)</td>
<td>20</td>
<td>18</td>
</tr>
</tbody>
</table>

Source Cell → SN.R : (1,-1)
Destination Cell → DN.C : (0,0)

b. **Logically** by set of comparisons between source node’s cells and destination node’s cells:

i. IF \(|X_{SN.C} - X_{DN.C}| < |X_{SN.C} - X_{DN.L}|\) THEN DC = DN.C
   ELSE IF \(|Y_{SN.C} - Y_{DN.L}| < |Y_{SN.C} - Y_{DN.R}|\) THEN DC = DN.L ELSE DC = DN.R

ii. IF \(|X_{SN.C} - X_{DN.C}| < |X_{SN.L} - X_{DN.C}|\) THEN SC = SN.C
    ELSE IF \(|Y_{SN.L} - Y_{DN.C}| < |Y_{SN.R} - Y_{DN.C}|\) THEN SC = SN.L ELSE SC = SN.R

Example: Source node = [(1, -3), (2, -2), (1, -1)], Destination node: [(-1, -1), (0, 0), (-1, 1)]

i. IF \(|2 - 0| < |2 - -1|\) THEN DC = (0,0)
ii. IF \(|2 - 0| < |1 - -1|\) → true
    ELSE IF \(|1 - 3| < |1 - 1|\) → false


ELSE SC = (1, -1)
Source Cell → SN.R : (1,-1)
Destination Cell → DN.C : (0,0)

2) When source cell equal destination cell then apply node routing between source node and destination node otherwise continue cell routing steps.

3) Specify the next cell by calculating x and y steps between source cell and destination cell as shown in Fig.10.

   a. Find the Xstep
      i. IF(X_{SC} == X_{DC}) THEN Xstep = 0.
      ii. IF(X_{SC} > X_{DC}) THEN Xstep = -1.
      iii. IF(X_{SC} < X_{DC}) THEN Xstep = 1.

   b. Find the Ystep
      i. IF(Y_{SC} == Y_{DC}) THEN Ystep = Xstep.
      ii. IF(Y_{SC} > Y_{DC}) THEN (IF(Xstep == 0) THEN Ystep = -2 ELSE Ystep = -1).
      iii. IF(Y_{SC} < Y_{DC}) THEN (IF(Xstep == 0) THEN Ystep = 2 ELSE Ystep = 1).

   c. Next Cell = (X_{SC} + Xstep, Y_{SC} + Ystep)

   d. Example:
      Source node = [(1, -3), (2, -2), (1, -1)]
      Destination node: [(-1, -1), (0, 0), (-1, 1)]
      Source Cell = (1, -1), Destination Cell = (0, 0)
      IF(X_{SC} > X_{DC}) THEN Xstep = -1. \rightarrow Xstep = -1
      IF(Y_{SC} < Y_{DC}) THEN (IF(Xstep == 0) THEN Ystep = 2 ELSE Ystep = 1).
4) Specify the local destination node in the source cell by using Xstep, Ystep and Next Cell as shown in Fig.11:

a. IF (Xstep==0 AND Ystep==2) THEN
   
   IF (SC == SN.R) THEN
     LDN=[SC,(X_{SN.C}, Y_{SC}+1), NC]
   ELSE
     LDN=[SC,(X_{SN.L}, Y_{SC}+1), NC]

b. IF (Xstep==0 AND Ystep== -2) THEN
   
   IF (SC == SN.L) THEN
     LDN=[NC,(X_{SN.C}, Y_{SC} -1), SC]
   ELSE
     LDN=[NC,(X_{SN.L}, Y_{SC} -1), SC]

c. IF ((Xstep ==1 OR Xstep== -1) AND Ystep==1) THEN
   
   IF (SC == SN.R) THEN
     LDN=[(X_{NC}, Y_{SC} -1), SC, NC]
   ELSE
     LDN=[SC,NC,(X_{NC}, Y_{SC}+1)]

d. IF ((Xstep =1 OR Xstep= -1) AND Ystep= -1) THEN
   
   IF (SC == SN.L) THEN
     LDN=[NC,SC,(X_{NC}, Y_{SC}+1)]
ELSE

\[ LDN = [(X_{NC}, Y_{NC} - 1), NC, SC] \]

Example: Source node = [(1, -3), (2, -2), (1, -1)]

Destination node: [(-1, -1), (0, 0), (-1, 1)]

Source Cell = (1,-1) , Destination Cell = (0,0)

Xstep= -1 , Ystep= 1 , Next Cell = (0,0)

IF ((Xstep ==1 OR Xstep== -1) AND Ystep==1) THEN

IF (SC == SN.R) THEN

\[ LDN = [(X_{NC}, Y_{SC} - 1), SC, NC \rightarrow [(0,-2),(1,-1),(0,0)] \]

---

5) Apply Node Routing between source node and local destination node.

6) Set the source node to be local destination node, then go to step one in cell routing.

3.3 Node Routing

Based on the source and destination nodes the cell routing algorithm determines the source and destination cells. Furthermore the cell routing algorithm determines the local destination node within the source cell. The local destination node becomes the new source for the cell routing algorithm which first the new local destination within the new cell. This process is repeated recursively, until the source and the destination nodes become in the same cell. It’s important to observe that the node routing algorithm is utilized repeatedly every time; we look for a local destination within the same cell.
The node routing algorithm as shown in Fig.12 works according these cases such that the source node called local source node and the destination node called local destination node in the same cell:

1. When the local destination node is in the right of local source node then go one step to the right.
2. When the local destination node is in the left of local source node then go one step to the left.
3. When the local destination node is in the top of local source node then goes one step to the top.
4. When the local destination node is in the bottom of the local source node then goes one step to the bottom.

```
Input (LSn, LDn, Dn)
Output(Path)
{
    Path .Add( LSn);
    If(LSn ≠ LDn)
    {
            LNN ← ((LSn.C.x , LSn.L.y-1), LSn.L , LSn.C);
        Else
            If(LSn.C.x < LDn.C.x)
                LNN ← (Lsn.L , (LSn.C.x+2 , LSn.C.y),LSn.R);
            Else
                LNN ← (LSn.L , (LSn.C.x-2 , LSn.C.y) , LSn.R);
        
        Path .Add( NodeRouting(LNn , LDn , Dn)) ;
    }
    ElseIf (LDn ≠ Dn)
    {
        Path.Add( CellRouting (LDn , Dn));
    }
    Return Path;
}
```

Example:

Source node = [(1, -3), (2, -2), (1, -1)]
Destination node: [(-1, -1), (0, 0), (-1, 1)]
Local Destination Node: [(0,-2),(1,-1),(0,0)]

The node routing applied between SN and LDN because the DN not equal LDN.

First step → Inside the cell as shown in Fig.13: LSN = [(1, -3), (2, -2), (1, -1)], LDN=[(0,-2),(1,-1),(0,0)]

Second step → according 4th case: LSN = [(1, -3), (0, -2), (1, -1)], LDN=[(0,-2),(1,-1),(0,0)]

Third step → according 1st case: LSN = [(0, -2), (1, -1), (0, 0)], LDN=[(0,-2),(1,-1),(0,0)]

Forth step → (LSN==LDN) AND (LDN != DN) : set SN=LSN call cell routing between SN and DN.

Fig.13. Example for node routing inside cell (1, -1)

Note: we can replace proposed node routing algorithm by any routing algorithm between nodes inside cell like ring routing. also we can use routing table.

4. Comparison between current and proposed algorithms

The current routing algorithm uses one level of routing which move from node to node without using the cell concept. It is traditional routing and easy. The proposed routing uses two levels of routing: internal Node Routing which operates between the nodes within the cell and external Cell Routing which route messages from cell to cell. This concept is very useful with structure of hex-cell topology specifically when we talk about grouping (clustering) six processors as one component (Cell) and there is a central processor which organizes these processors. In such case Cell Routing can be used to routing from one group to another group and for the internal group routing the proposed Node Routing can be used or any another centre or distribute routing algorithm.

The current routing calculations and comparisons depend on the depth of hex-cell network which means that the topology must be fixed. Adding or removing levels need to update the depth value and readdressing nodes since the addressing scheme also depend on the depth. The proposed routing is more
flexible since its calculations and comparisons do not depend on the depth of the network, which implies that adding or removing network levels do not need re-addressing the network nodes.

The number of comparisons in the current routing algorithm is less than the number of compassions in the proposed routing algorithm. However, the current routing algorithm needs sum, sub and mod operations while the proposed routing algorithm needs only sum and sub operations.

The existing addressing scheme [1] depends on the depth of the Hex-Cell network. However, the proposed addressing scheme does not depend on the depth of the network, which helps avoiding reconfiguring the address scheme, when the number of nodes or level change for any reasons.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Existing Routing Algorithm</th>
<th>Proposed Routing Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Logic Operations</td>
<td>8</td>
<td>Cell Routing = 7+6+5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Node Routing = 9</td>
</tr>
<tr>
<td>Min Logic Operations</td>
<td>4</td>
<td>Cell Routing = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Node Routing = 2</td>
</tr>
<tr>
<td>How many times the algorithm called</td>
<td>Number of nodes in path</td>
<td>Cell Routing= number of cells in path</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Node routing= number of nodes in path</td>
</tr>
<tr>
<td>Arithmetic operations</td>
<td>Sum, Sub, Mod</td>
<td>Sum. Sub</td>
</tr>
<tr>
<td>Comparison depends on</td>
<td>1) Addresses of source and destination nodes</td>
<td>Address of source and destination nodes</td>
</tr>
<tr>
<td></td>
<td>2) depth</td>
<td></td>
</tr>
<tr>
<td>Addressing scheme</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Depends on depth</td>
<td>yes</td>
<td>No</td>
</tr>
<tr>
<td>Structure</td>
<td>(X, Y)</td>
<td>((X1, Y1), (X2, Y2), (X3, Y3))</td>
</tr>
</tbody>
</table>

5. Conclusion

In this paper a new routing algorithm for Hex-Cell network has been proposed. It provides the shortest path from source to destination based on a new addressing mode for the network nodes and cells. The proposed routing algorithm uses two levels of routing: internal node routing which route messages between the nodes within a single cell and external cell routing which route messages from cell to cell. In contrast to the current routing algorithm this algorithm does not depend on the depth of the Hex-Cell network which eliminates the need for updating the depth value and readdressing the nodes every time a new level is added. In additional to the flexibility provided by the new addressing mode this algorithm required less arithmetic operations which makes it more efficient than the existing one. In the future, we will focus on implementing a fault tolerance algorithm based on the proposed routing algorithm.
References


