Shortest Path Routing on Multi-Mesh of Trees

Sudhanshu Kumar Jha, Prasanta K. Jana, *Senior Member, IEEE*

***Abstract*— Multi-Mesh of Trees (MMT) is an efficient interconnection network for massively parallel processing system. This network is a hybrid product of multi-mesh and mesh of trees. Many interesting topological properties and various algorithms have been developed on this architecture. In this paper, we propose an algorithm for shortest path routing on an n^4-processor multi-mesh of trees network. The proposed algorithm requires 12 log n + 1 time in the worst case to deliver a message from a source node to a destination node.**

***Index Terms*—Interconnection network, mesh of trees, multi-mesh of trees, shortest path routing, time complexity**

I. Introduction

MULTI-MESH OF TREES (MMT) [1] is an efficient network that takes the benefits of the topological structures of two interconnection networks, the mesh of trees (MOT) [2] and the multi-mesh (MM) network [3]. The diameter of an n^4-processor MMT is shown to be 4 log n + 2 [1]. As a result many diameter based algorithms such as Lagrange's interpolation, polynomial root finding, matrix-vector multiplication, DFT computation and sorting have been shown to map in $O(\log n)$ time [1] in comparison with $O(n)$ time on multi-mesh network [3]. Recently parallel prefix [4] and multi-sort [5] have been reported on the same architecture with 3.75 log N + 6 and $O(n^2)$ time respectively.

Shortest path routing is a kind of routing which attempts to send data packets over an interconnection network in such a way that the path taken from the sending node to the recipient one is minimized. It is one of the most fundamental and most commonly encountered problems in the study of communication networks. A great deal of research [17] has been devoted to this subject. Efe [6], Xin Yu [7] presented an algorithm for shortest path routing on crossed cube network in $O(n^2)$ time. Dobravec [8] reported an optimal dynamic two terminal message routing algorithm for k-circulant ($k \geq 2$) networks for the restricted shortest path in $O(\log n)$ time. Shortest path routing on a binary de Bruijn network with 2^n processors was reported by J. W. Mao et al. [9] in $O(n^2)$ time. Ming-Yang Su et. al.[10] presented a shortest-path routing on a WK-recursive network having network size d^t. Their algorithm was shown to require $O(t)$ time for each intermediate node to determine the next node along the shortest path with $O(d - t)$ pre-processing time. Sau and Zerovnik [11] presented an optimal permutation routing algorithm on full-duplex hexagonal mesh networks. Ke Qiu [12] presented a routing algorithm that finds n disjoint shortest paths from the source node to n target nodes in the n-dimensional hypercube in $O(n^3 \log n)$ time. Pradhan [13] reported a shortest path routing on a binary tree in O (log n) time. Nguyen [14] developed a routing algorithm for hyper-de Bruijn networks. Recently, we have reported shortest path routings on n^2-processor MOT and n^4-processor OTIS-MOT [15] in 8 log n and 8 log n + 1 time respectively.

In this paper, we present an algorithm for shortest path routing on n^4-processor MMT. Our proposed algorithm is shown to require 12 log n + 2 time in the worst situation.

The rest of the paper is organized as follows. Section 2, describes the topological structure of the multi-mesh of trees network. In section 3 we present our proposed algorithm for the shortest path routing followed by the conclusion in section 4.

Sudhanshu Kumar Jha is associated with the Department of Computer Science and Engineering, Indian School of Mines, Dhanbad – 826 004 (e-mail: sudhanshukumarjha@gmail.com).
Prasanta K. Jana is associated with the Department of Computer Science and Engineering, Indian School of Mines, Dhanbad – 826 004 (e-mail: prasantajana@yahoo.com).

II. Topology of Multi-Mesh of Trees

An $n \times n$ MMT network is built around n^2 blocks arranged in a two-dimensional lattice with n rows and n columns. Each block is basically an $n \times n$ MOT with n^2 processors. Therefore the total number of processors in the MMT is n^4. There are two types of links in MMT: intra-block links (i.e., among the processors inside a block) and the inter-block links (i.e., among the processors of two different blocks). As an example, 3 × 3 MMT is shown in Fig. 1 in which all the inter-block links are not shown.

Let $B(\alpha,\beta)$ denote the block placed in the α^{th} row and β^{th} column, then we address the processor placed in the x^{th} row and y^{th} column within the block $B(\alpha, \beta)$ as $P(\alpha, \beta, x, y)$. Then the intra-block links can be defined as follows:

1) **Horizontal intra-block links:** The processors in each row are connected to form a binary tree rooted at $P(\alpha, \beta, x, 1)$, i.e., the processor $P(\alpha, \beta, x, y)$ is directly connected to the processors $P(\alpha, \beta, x, 2y)$ and $P(\alpha, \beta, x, 2y+1)$ if they exist for $1 \leq \alpha, \beta, x, y \leq n$.
2) **Vertical intra-block links:** The processors in each column are connected to from a binary tree rooted at $P(\alpha, \beta, 1, y)$, i.e., the processor $P(\alpha, \beta, x, y)$ is directly connected to the processors $P(\alpha, \beta, 2x, y)$ and $P(\alpha, \beta, 2x+1, y)$ if they exist for $1 \leq \alpha, \beta, x, y \leq n$.

The inter-block links are also of two types defined as follows:

3) **Horizontal inter-block links:** The processor $P(\alpha, \beta, x, 1)$ is directly connected to the processor $P(\alpha, x, \beta, n)$, $1 \leq \alpha, \beta, x \leq n$. As a special case when $\beta = x$, this link connects two processors within the same block.
4) **Vertical inter-block links:** The processor $P(\alpha, \beta, 1, y)$ is directly connected to the processor $P(y, \beta, n, \alpha)$, $1 \leq$

ISBN: 978-988-19251-4-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

$\alpha, \beta, y \leq n$. When $\alpha = y$, this link connects two processors within the same block.

In Fig. 1, the intra-block and the inter-block links are shown by solid and dashed lines respectively. The address of each block is shown by a pair of indices in boldfaces above the block and the processor indices are shown adjacent to each processor. This is shown only for the first block. We assume that all the links are bi-directional, i.e., full duplex, so that the data movements can be accomplished in both the directions. The network is shown to have diameter 4 log n + 2 and bisection width $2n$ $(n - 1)$ [1].

III. PROPOSED ALGORITHM

For the completeness of the paper, we first describe the shortest path routing algorithm on a MOT as we reported in [15]. We next present our proposed routing algorithm on the MMT network.

A. Shortest path routing on MOT (*SPR_MOT*)

For the purpose of shortest path routing on MOT, we address each node of the MOT by two-index binary labels as shown in Fig. 2 for a 7×7 mesh of trees. In this binary labelling, the root of a row-tree or column-tree is labelled with 1; if any internal node is labelled with u then its left child and right child are labelled with $2u$ and $2u$+1 respectively. Let (x, y) and (u, v) denote the source and destination nodes respectively in a MOT and (x_b, y_b) and (u_b, v_b) be their corresponding binary representations. For the shortest path routing, we obtain the edges of the path between these nodes as follows. We first traverse the row-tree following the shortest path routing on a binary tree [13], [16] over column-index and stops when $y_b = v_b$. We then follow the shortest path routing on the column tree until $x_b = u_b$. This requires 4 log n time. Fig. 2 illustrates the shortest path routing in the worst as well as the best cases between the source node S1 (10, 100) and the destination node D1 (111, 110) and between S2 (11, 11) and D2 (110, 11) respectively. Note that in the worst case, the shortest path routing on the row-tree ends at the node W (10, 110) as its column index matches with that of the destination node D1.

We refer the above shortest path routing on MOT between the source node S and the destination node D as *SPR_MOT*(S, D) for its use in the rest of the paper. This is to note that *SPR_MOT* requires 4 log n time as reported in [15].

B. Shortest path routing on MMT

We assume here that $P(\alpha_s, \beta_s, x_s, y_s)$ and $P(\alpha_d, \beta_d, x_d, y_d)$ are the source and destination nodes respectively. We can observe from Fig. 1 that there exists more than one inter-block link between any two blocks in the same row or column. Also there is more than one path between any two nodes within each block in the MMT network. Therefore, we have to find out the shortest path among the number of available paths. We perform the shortest path routing (SPR) on MMT following the two major cases as described below.

Case *A*: When both the source and destination nodes lie in the same block of the MMT, i.e, $\alpha_s = \alpha_d$ and $\beta_s = \beta_d$.
As the block is basically a mesh of trees (MOT), we simply perform, the shortest path routing on mesh of trees as discussed in [15]. However, we take care here the two special cases namely $\beta_s = x_d$ and $\alpha_s = y_d$, i.e., when the inter-block link connects the two processors within the same block. We use a function called *SPR_Block_Level_MMT* (S, D) to incorporate these two special cases as follows in which S and D denote the source and destination nodes respectively.

Algorithm *SPR_Block_Level_MMT* $((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, x_d, y_d))$

```
/* P(α_s, β_s, x_s, y_s) is the source and P(α_s, β_s, x_d, y_d) is the
   destination */
{
  If (β_s = x_d)
  {
     Route the message from (α_s, β_s, x_s, y_s) to (α_s, β_s, x_d, y_d)
     using horizontal inter-block links.
  }
  Else if (α_s = y_d)
  {
     Route the message from (α_s, β_s, x_s, y_s) to (α_s, β_s, x_d, y_d)
     using vertical inter-block links.
  }
  Else
  {
     Call SPR_MOT ((α_s, β_s, x_s, y_s), (α_s, β_s, x_d, y_d))
  }

} // end of the algorithm
```

This is obvious to note that the *SPR_Block_Level_MMT* algorithm requires 4 log n time since *SPR_MOT* requires the same time.

Case *B*: When source and destination nodes lie in two different blocks of the MMT.

This has the following two sub-cases.

i) Both the source and the destination nodes may exist in the same row (or column) block of the MMT, i.e. here $\alpha_s = \alpha_d$ and $\beta_s \neq \beta_d$ (or $\alpha_s \neq \alpha_d$ and $\beta_s = \beta_d$). In this case, the source and the destination blocks are directly connected by two horizontal (or vertical) inter-block links. Therefore, starting from the source node, we first visit the nearest exit node in the source block and reach to the destination block via the inter-block link and finally visit the destination node by applying shortest path routing *SPR_MOT* discussed previously.

ii) They may not exist in the same row (or column) block of the MMT, i.e., $\alpha_s \neq \alpha_d$ and $\beta_s \neq \beta_s$. In this case, there is no direct inter-block link from the source block to the destination block. Therefore, we require to traverse via one intermediate block to reach the destination block.

Note that for the above case, we can exit from the source block via either the root node $P(*, *, *, 1)$ or leaf node $P(*, *, *, n)$ with respect to the destination block where '*' denotes any possible value of the index. We refer such nodes as the exit nodes. From the source node, we first reach to the nearest exit node (in the source block) with respect to the destination block. We use a function *dist* (S, *) to obtain the nearest exit node from the source node. The *dist*(S, Y) function actually calculates the shortest distance

ISBN: 978-988-19251-4-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

between the source node S and a possible exit node Y within a same block.

The rough sketch of the shortest path routing on the MMT considering all the above case is shown in Fig. 3 in which S_i and D_i denotes the source and the destination nodes.

We now formally present the shortest path routing algorithm on MMT as follows covering both the cases A and B.

Algorithm *SPR_MMT* $((\alpha_s, \beta_s, x_s, y_s), (\alpha_d, \beta_d, x_d, y_d)$

/* $P(\alpha_s, \beta_s, x_s, y_s)$ is the source and $P(\alpha_d, \beta_d, x_d, y_d)$ is the destination */

{

Case 1: $(\alpha_s = \alpha_d)$ AND $(\beta_s = \beta_d)$

/* Both source and destination lie in the same block*/

Call *SPR_Block_Level_MMT* $((\alpha_s, \beta_s, x_s, y_s), (\alpha_d, \beta_d, x_d, y_d))$

Case 2: $(\alpha_s = \alpha_d)$ AND $(\beta_s \neq \beta_d)$

/* Both source and destination are in different blocks but their blocks lie in the same row of MMT */

If(*dist*$((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, \beta_d, 1)) \leq$ *dist*$((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, \beta_d, n))$

{

2.1 Call *SPR_MOT* $((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, \beta_d, 1))$

2.2 Route the message from $(\alpha_s, \beta_s, \beta_d, 1)$ to $(\alpha_s, \beta_d, \beta_s, n)$ using horizontal inter-block link.

2.3 Call *SPR_Block_Level_MMT* $((\alpha_s, \beta_d, \beta_s, n), (\alpha_d, \beta_d, x_d, y_d))$

}

Else

{

2.1 Call *SPR_MOT* $((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, \beta_d, n))$

2.2 Route the message from $(\alpha_s, \beta_s, \beta_d, n)$ to $(\alpha_s, \beta_d, \beta_s, 1)$ using horizontal inter-block link.

2.3 Call *SPR_Block_Level_MMT* $((\alpha_s, \beta_d, \beta_s, 1), (\alpha_d, \beta_d, x_d, y_d))$

}

Case 3: $(\alpha_s \neq \alpha_d)$ AND $(\beta_s = \beta_d)$

/* Both source and destination are in a different blocks but their blocks lie in the same column of MMT */

If(*dist*$((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, 1, \alpha_d)) \leq$ *dist*$((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, n, \alpha_d))$

{

3.1 Call *SPR_MOT*$((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, 1, \alpha_d))$

3.2 Route the message from $(\alpha_s, \beta_s, 1, \alpha_d)$ to $(\alpha_d, \beta_s, n, \alpha_s)$ using vertical inter-block link.

3.3 Call *SPR_Block_Level_MMT* $((\alpha_d, \beta_s, n, \alpha_s), (\alpha_d, \beta_d, x_d, y_d))$

}

Else

{

3.1 Call *SPR_MOT*$((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, n, \alpha_d))$

3.2 Route the message from $(\alpha_s, \beta_s, n, \alpha_d)$ to $(\alpha_d, \beta_s, 1, \alpha_s)$ using vertical inter-block link.

3.3 Call *SPR_Block_Level_MMT* $((\alpha_d, \beta_s, 1, \alpha_s), (\alpha_d, \beta_d, x_d, y_d))$

}

Case 4: $(\alpha_s \neq \alpha_d)$ AND $(\beta_s \neq \beta_d)$

/* Source and destination are in different blocks and their blocks are not also in same row or same column of MMT. Therefore, in this case we have to route the message in a particular block from where, either case 2 or case 3 satisfy */

If(*dist*$((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, 1, \alpha_d)) \leq$ *dist*$((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, n, \alpha_d))$

{

4.1 Call *SPR_MOT* $((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, 1, \alpha_d))$

4.2 Route the message from $(\alpha_s, \beta_s, 1, \alpha_d)$ to $(\alpha_d, \beta_s, n, \alpha_s)$ using vertical inter-block link.

4.3 Call *SPR_MOT* $(((\alpha_d, \beta_s, n, \alpha_s), (\alpha_d, \beta_s, \beta_d, n))$

4.4 Route the message from $(\alpha_d, \beta_s, \beta_d, n)$ to $(\alpha_d, \beta_d, \beta_s, 1)$ using horizontal inter-block link.

4.5 Call *SPR_Block_Level_MMT* $((\alpha_d, \beta_d, \beta_s, 1), (\alpha_d, \beta_d, x_d, y_d))$

}

Else

{

4.1 Call *SPR_MOT* $((\alpha_s, \beta_s, x_s, y_s), (\alpha_s, \beta_s, n, \alpha_d))$

4.2 Route the message from $(\alpha_s, \beta_s, n, \alpha_d)$ to $(\alpha_d, \beta_s, 1, \alpha_s)$ using vertical inter-block link.

4.3 Call *SPR_MOT* $((\alpha_d, \beta_s, 1, \alpha_s), (\alpha_d, \beta_s, \beta_d, n))$

4.4 Route the message from $(\alpha_d, \beta_s, \beta_d, n)$ to $(\alpha_d, \beta_d, \beta_s, 1)$ using horizontal inter-block link.

4.5 Call *SPR_Block_Level_MMT* $((\alpha_d, \beta_d, \beta_s, 1), (\alpha_d, \beta_d, x_d, y_d))$

}

} /* *End of algorithm* */

Time Complexity: Case 1 requires 4 log n time. Case 2 and case 3 each requires 8 log n + 1 time as in both the cases, there is a single call of *SPR-MOT*, a single call of *SPR_Block_Level_MMT* and a single inter-block communication including else part. Case 4 requires a maximum of 12 log n + 2 time due to two calls of *SPR_MOT*, one call of *SPR_Block_Level_MMT* and two inter-block communications including else part. Therefore, the shortest path routing on MMT requires 12 log n + 2 time in the worst situation.

The correctness of the above algorithm can be readily seen from the above illustrations and the description of the algorithm.

IV. Conclusion

We have presented a shortest path routing algorithm on the multi-mesh of trees network having n^4-processor. The proposed algorithm is based on shortest path routing on the binary tree. We have shown that the algorithm runs in 12 log n + 2 time in the worst situation.

ISBN: 978-988-19251-4-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

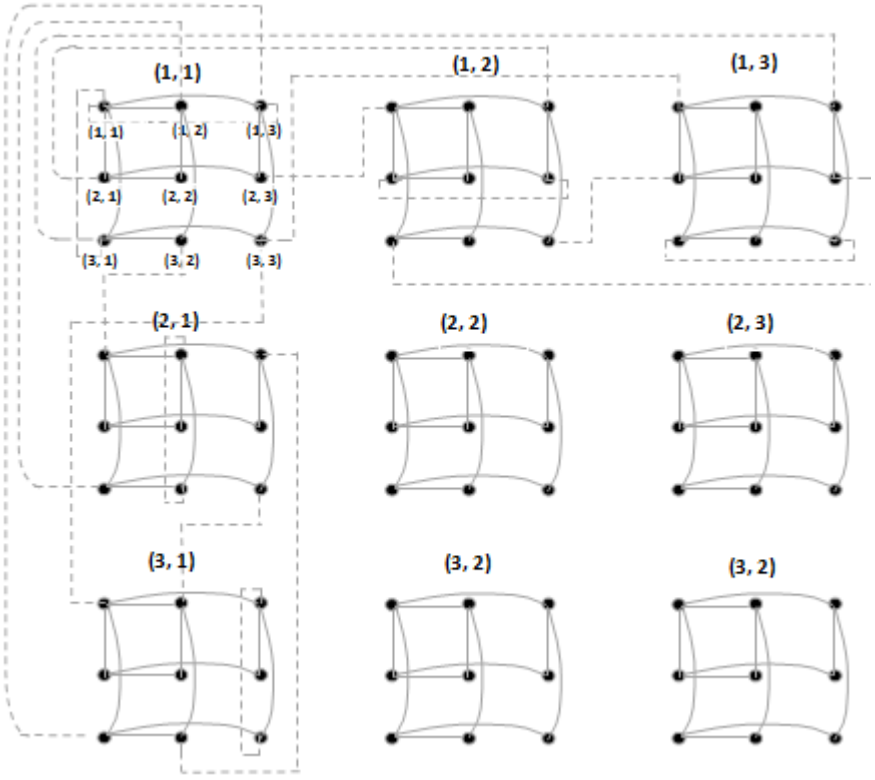

Fig. 1 Graph topology of Multi-Mesh of Trees with 81 processors. All inter-block links are not shown

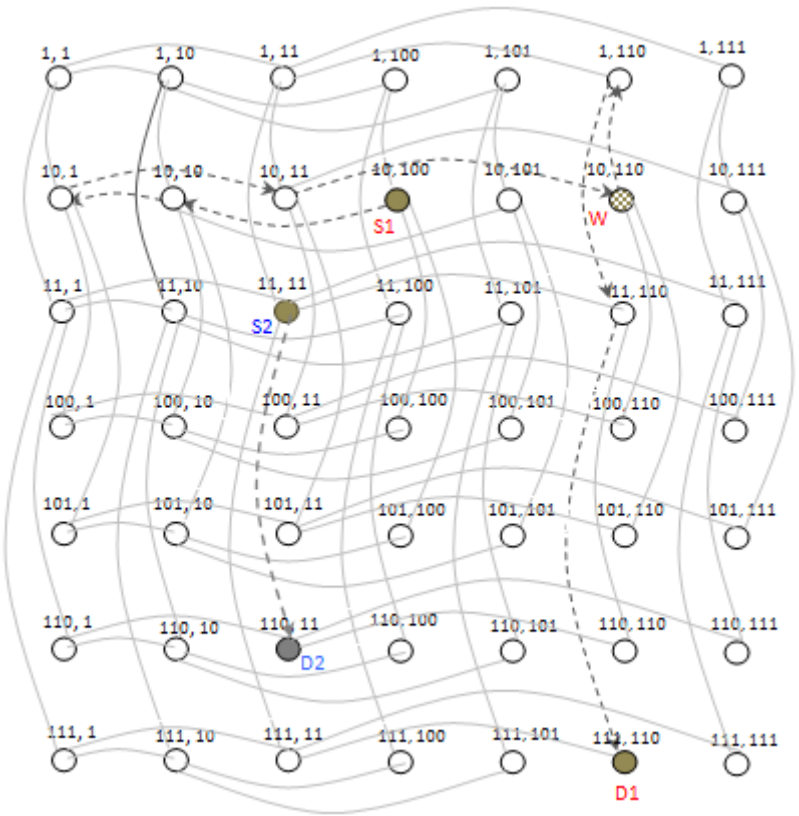

Fig.2 Shortest path in a MOT between nodes (10,100) (source) and (111,110) (destination). The traversal of row- tree ends at the node W(10,110). The figure is taken from [15].

ISBN: 978-988-19251-4-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

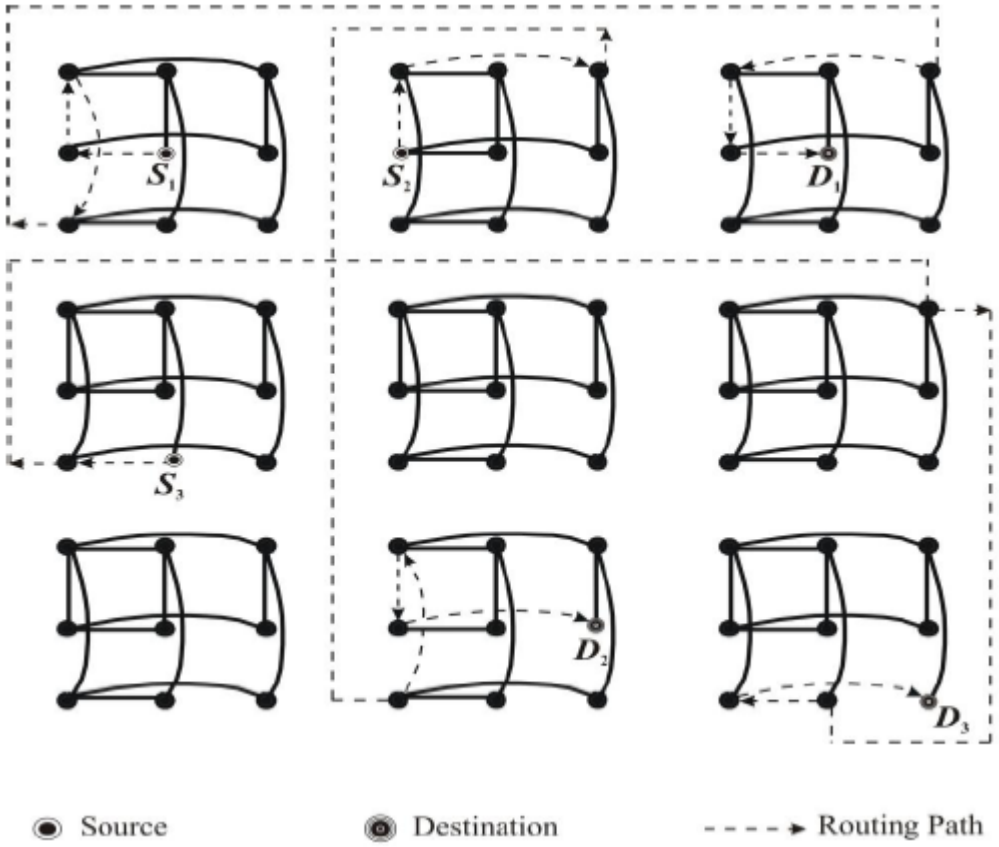

FIG. 3. ILLUSTRATION OF SHORTEST PATH ROUTING ALGORITHM

REFERENCES

[1] P. K. Jana, Multi-mesh of trees with its parallel algorithms, Journal of Systems Architecture 50 (2004) 193–206.
[2] S.G. Akl, The Design, Analysis of Parallel Algorithms, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[3] D. Das, B.P. Sinha, Multi-mesh—an efficient topology for parallel processing, in: Proceedings of the Ninth International Parallel Processing Symposium, Santa Barbara, CA, April 25–28, 1995, pp. 17–21.
[4] . K. Jha, P. K. Jana, "Fast Parallel Prefix Computation on Multi-Mesh of Trees", *accepted for presentation* in *IEEE* Int'l. Conf. on Computer and Communication Technology (ICCC2010), MNNIT, Allahabad, 17-19 September 2010.
[5] N. Rakesh, Multi-sort Algorithm on Multi-Mesh of Trees, MASAUM Journal of Computing Vol.1 No.1 August 2009.
[6] K. Efe. The crossed cube architecture for parallel computing". IEEE Trans. Parallel and distributed Systems, 3(1992) 513-524.
[7] X. Yu, M. Wu, and G. J. Wang. A novel shortest path routing algorithm in the crossed cube, Chinese Journal of Computers, 30, (2007), 615-621.
[8] Tomaz Dobravec, Janez Z Erovnik, Borut Robic, An optimal message routing algorithm for circulant networks, Journal of Systems Architecture, 52 (2006) 298–306.
[9] Jyh-Wen Mao and Chang-Biau Yang, Shortest path routing and fault tolerant routing on de Bruijn networks, Journal of Networks, 35(2000) 207-215.
[10] Ming-Yang Su, Gen-Huey Chen, and Dyi-Rong Duh, A Shortest-Path Routing Algorithm for Incomplete WK-Recursive Networks, IEEE transactions on parallel and distributed systems, vol. 8, no. 4, April 1997, pp 367- 379.
[11] Ignasi Sau and Janez Zerovnik, An optimal permutation routing algorithm for full-duplex hexagonal mesh networks, University of Ljubljana, preprint series, Vol. 44 (2006), 1017, ISSN 1318-4865.
[12] Ke Qiu, An Efficient Disjoint Shortest Paths Routing Algorithm for the Hypercube, in Proc. of 14th IEEE Intl. Conf. on Parallel and Distributed Systems, 2008.
[13] D. K. Pradhan, *Fault-Tolerant Computing: Theory and Techniques*, (Englewood Cliffs, Prentice Hall, 1986).
[14] Ngoc Chi Nguyen, Thanh Vu Dinh, Tuan Dang Anh, Improving shortest path routing in hyper-de Bruijn networks, in Proc. of IFOST 2000, 288 – 292.
[15] Keny T. Lucas, P. K. Jana, "Sorting and routing on OTIS-mesh of trees", *Parallel Processing Letters*, Vol. 20, Issue 2, 2010, pp. 145-154.
[16] K. Efe, Mesh-connected trees: a bridge between grids and meshes of trees, *IEEE Trans. Parall. Distri. Syst.* 7 (1996) 1281–1291.\
[17] N. Deo and C. Pang, "Shortest-Path Algorithms: Taxonomy and Annotation," Networks, 14 (1984) pp: 275–323.

ISBN: 978-988-19251-4-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)