ISSN 2278-3083
Volume 8, No.6, November –December 2019
International Journal of Science and Applied Information Technology
Available Online at http://www.warse.org/ijsait/static/pdf/file/ijsait13862019.pdf
https://doi.org/10.30534/ijsait/2019/138620198

ROUTING CAPABILITIES OF MESH-HYPERCUBE NETWORK

Mahmoud Omari
Department of Computer Science, Amman Arab University, Amman, Jordan

ABSTRACT

The aim of this research is to develop optimal packet routing strategies for the Mesh Hypercube (MH) network of arbitrary size. MH has been introduced as a new interconnection network for multicomputer systems. The basic structure for this network is a combination of both mesh and hypercube networks. In this research, the parallel path property in the MH is studied and the length and the number of parallel shortest paths are computed. An optimal packet routing algorithm is given which uses minimal length path. The algorithm is simple and appears to be well suited to VLSI implementation. The simulation results of the proposed algorithm on various network designs showed that it performs very well at various network conditions. In a MH of size m×n nodes, the algorithm performs outstandingly when n>m. Specifically, the best results of both waiting-time and packet-latency network performance measures were achieved when m=4 and n=16, while the worst values of all measures were obtained when m=8 and n=8.

Key words: Parallel Processing, Multicomputer, Routing, Packet-Routing, Interconnection Networks, Mesh-Hypercube, Hypercube, Mesh

1. INTRODUCTION

The design of efficient routing algorithms has received significant attention due to its importance to high performance in multicomputer systems [1-14]. Many applications that run on multicomputer systems depend highly on communication time. Examples of such applications include sorting, matrix computations, parallel prefix, and Fourier Transform.

Packet routing is the fundamental form of data communication in multicomputer systems [15-21]. An algorithm for packet routing has to determine each packet's path through a network by using various information, such as source node address, destination node address, and the configuration of the network. The routing algorithm can be centralized or distributed [8]. In centralized routing, a single processor determines one shortest path from a given source node to a given destination node. Finding the shortest path by a single, central processor is very slow and leads to a major bottleneck. On the other hand, if every source node computes one shortest path to the destination and sends that path along with the message to guide it through the intermediate nodes, the bottleneck problem is avoided but traffic is increased and routing remains relatively slow. In a distributed routing, however, all intermediate nodes on the shortest path cooperate to find the shortest path using the destination address. In this case, each intermediate node needs only the destination address to determine which neighbor falls on the shortest path to a given destination. An optimal routing strategy in any network would route packets from their source to their destination along shortest paths. The algorithm that routes packets along shortest paths is called minimal algorithm. The shortest paths can be found in a distributed fashion whereby each node in a path determines the address of the next node in the path.

Routing capabilities are usually measured by the number of parallel paths between any two nodes, the diameter, and the average distance of the underlying topology of the network. The routing capabilities of Mesh-Hypercube were studied based on its average distance and diameter in [22].

This paper is organized as follows. The next section gives an overview of the Mesh-Hypercube network. The analysis of Mesh-hypercube paths presented in section 3. Routing strategies and a routing algorithm with its analysis are presented in section 4. Simulation and results analysis of the proposed routing algorithm are given in section 5. Section 6 concludes the paper.

2. MESH-HYBERCUVE NETWORK OVERVIEW

The network topology defines how the nodes are interconnected and generally modeled as a graph. The vertices in the graph represent the nodes (processors) and the edges denote the communication links (channels). The basic topologies used in most parallel computers are meshes and *k*-ary *n*-cube. In this section we give the basic definition of MH [22].

The topology of MH can be described as undirected graph, $G_{MH} = (V, E)$, where V denotes a set of nodes and E denote a set of edges. The MH can be viewed as a product graph because it combines a 1-dimensional mesh graph, hereafter we simply call it mesh; and a hypercube graph in such a way that if $G=G_1G_2$ where G_1 represents the mesh graph and G_2 the hypercube then G is constructed by having *n* copies of the mesh graph and connecting each sibling in a hypercube

structure. The graph can also be viewed as having *m* rows and *n* columns, where nodes in each column form the mesh and the nodes in each row are connected in a hypercube structure of (*log n*) dimensions.

The size of the MH is characterized by a two-tuple (*m, n*), where *m* defines the number of nodes forming a 1-dimenssional mesh and *n* defines the number of nodes in a hypercube. For an MH(*m, n*), where *n* is a power of 2, the total number of nodes is equal to *m*×*n*. A node address in the MH(*m, n*) is denoted by a two-tuple (*l*, X), where $1 \le l \le m$, $X=x_k...x_1x_0$, $0 \le k \le (\log n)-1$, and $x_i \in \{0, 1\}$.

Given the set of nodes (V), the set of edges (E) is constructed as follows: For two nodes (l, $x_k ...x_0$) and (r, $y_k \ldots y_0$) where $1 \le l, r \le m$, $0 \le k \le (\log n)-1$, and [x_i, $y_i \in \{0, 1\}$, for all $0 \le I \le k$.

1. The two nodes on the same mesh if $x_i=y_i$ for all $0 \le i \le k$.
2. There is a hypercube link between two nodes *iff* $l=r$, $x_i \ne y_i$, and $x_j=y_j$ for all $j \ne i$ for $0 \le i, j \le k$.

Figure 1 shows a MH(3, 8) network, the dotted lines represent hypercube links and solid lines represent 1-dimensional meshes.

The choice of the two parameters, *m*, and *n* determines the size of the network, the implementation requirements, and the scaling complexity. The *m* parameter determines the size of the meshes while the parameter *n* defines the size of hypercubes on each row of the network.

over the paths between the two nodes such that each path can carry a portion of the data.

The length of a path between any two nodes is equal to the number of links in the path. The *distance* between two nodes *X* and *Y* in a hypercube is equal to the *Hamming distance* between their binary addresses, denoted by *H*(*X, Y*). That is, if *H*(*X, Y*)=*d*, then *X*'s and *Y*'s binary addresses differ in exactly *d* bit positions. A path between *X* and *Y* is called *optimal path* if its length is equal to the distance between the two nodes. A *shortest path* is a path of minimal length among all possible paths between the two nodes. A routing algorithm that uses a shortest possible path for each packet is called *minimal*. In the next proposition we compute the length of the shortest path between any two nodes in a MH.

Proposition 1: The length of the shortest path from a source node (L*s* , S) to a destination node (L*d* , D) in MH is |L*d*-L*s*|+ H(S , D).

To explain the proposition, we give an example. Suppose (L*s*, S)=(1,000) and (L*d*, D)=(3,111). The length of the path between (L*s*, S) and (L*d*, D) is equal to L*d* - L*s* + H(S, D) = (3-1)+H(000,111) =2+3=5. This example can be seen clearly in Figure 2 that shows all possible parallel shortest paths between (1,000) and (3,111) in a MH. The total number of parallel shortest paths in a mesh-hypercube is given in proposition 2.

Proposition 2: The total number of *parallel* shortest paths between any two nodes (L*s*, S) and (L*d*, D) in MH is H(S , D) , if L*s* = L*d* or H(S , D)+1 , otherwise.

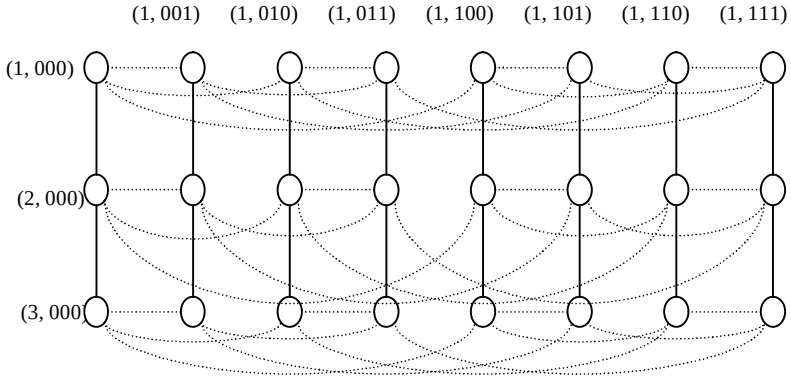

Figure 1:MH(3, 8) consists of 3 rows and 8 columns.

3. MESH-HYPERCUBE PATHS ANALYSIS

An important feature in a multicomputer system is the existence of multiple paths between every two processors [6] [23-24]. The existence of multiple paths between every two processors enhances the fault tolerance of the system and the speedup of the transfer of data between pairs of processors [25-27]. When routing between processors, if there is a faulty link or node in the current path, the routing algorithm can choose a non-faulty path. A routing algorithm that allows packets to adapt to traffic conditions by selecting alternate paths is called *adaptive*. If a large amount of data to be exchanged between two processors, then it can be distributed

It can be seen that using proposition 2, the total number of parallel shortest paths between (1,000) and (3,111) is equal to four (see figure 2).

(1,000)→ (1,100) → (1,110) → (2,110) → (3,110) → (3,111)
(1,000)→ (1,001) → (1,101) → (2,101) → (3,101) → (3,111)
(1,000)→ (1,010) → (1,011) → (1,111) → (3,111) → (3,111)
(1,000)→ (2,000) → (3,000) → (3,001) → (3,011) → (3,111)

Figure 2: Multiple parallel paths between (1,000) and (3,111).

4. ROUTING STRATEGIES

An optimal (minimal) routing algorithm should take the packet from a source node to a destination node using shortest path. In MH, there may be several routing strategies.

Due to the regularity of the mesh-hypercube structure, a distributed routing scheme can be implemented without global information. At the source node, the packet contains the source address, the destination address, message, and additional information, see figure 3. The inter-processor traffic of a node gets redistributed into two categories, the hypercube communication and the mesh communication. If the source and the destination of a packet are within the same hypercube subnetwork of the MH network, the routing procedure is exactly the same as of the regular hypercube network. Similarly, if the source and the destination of the packet are within the same mesh subnetwork of the MH network, the routing procedure is exactly the same as that of a regular mesh interconnected network.

Source address	Destination address	Additional information	Message

Figure 3: Packet format.

If neither of the previous two cases is true, the source and the destination are on different hypercubes and on different meshes subnetworks of the MH network. A routing strategy for this case is first to use the hypercube routing scheme until the packet arrives at the same mesh where the destination resides, and then to use the mesh routing scheme for the packet to arrive at the destination. Or the mesh routing scheme can first be applied to forward the packet to the same hypercube where the destination resides, and then the packet can reach the destination using the hypercube routing scheme. A third routing strategy would route packets by mixing the hypercube and the mesh routing until the packet is forwarded to the same hypercube or to the same mesh where the destination resides, and then forward the packet to the destination using the hypercube or the mesh routing scheme, respectively. A specification of the algorithm for the third strategy is given bellow.

Description of the Algorithm

An algorithm for packet routing has to determine each packet's path through a network by using various information, such as destinations addresses and the configuration of the network [28]. In the following, the case when $Ls \leq Ld$ will be considered The algorithm takes as input: the local node address (L, X), the destination node address (L*d*, D), and a TAG. The argument TAG can take one of three values: S means that the current node is the source, H means that the packet has been received through a hypercube link, or, M, the packet was received through a mesh link. The output of the algorithm is the address of next node on the shortest path to the destination. Once an intermediate node (L, X) in the path receives the packet, it does the following. If (L, X) equal to (L*d* , D), then the packet is consumed and the routing stops. Otherwise, the next node to receive the packet is determined as follows. If the cube addresses of (L, X) and (L*d*, D), namely X and D, are equal, then the next node is (L+1, D). If both nodes (L, X) and (L*d*, D) are on the same hypercube subnetwork, then the next node is (L, x_{k-1}…..d_i…X_0), where d_i is the right most bit over which X and D differ. Otherwise, the two nodes are on different mesh subnetwork and different hypercube subnetwork (i.e., $L \neq L_d$ and $X \neq D$). In this case, if the packet has been received on a mesh link, then the next node is (L, x_{k-1}…..d_i…X_0), where d_i is the right most bit over which X and D differ. If the packet has been received on hypercube link then the next node is (L+1, X). In the case where $L>L_d$, the algorithm works in a similar fashion. The details of both cases are given in the algorithm shown in figure 4.

Procedure Packet-Routing:
Inputs: (L , X): current node address; (L*d* , D): destination node address
TAG: The link at which the packet has been received.
begin
case 1: (L, X) = (L*d*, D):
Send packet to local processor and exit the algorithm.

case 2: X = D: /* on the same mesh*/
if L > L*d* **then** Send packet to (L-1, X);
else Send packet to (L+1, X); // i.e., L < L*d*

case 3: L = L*d*: /* i.e. on the same hypercube*/
Call procedure **Route-in-hypercube**

case 4: TAG = S or TAG = M /* Current node is the source or packet received through a Mesh link */
Call procedure **Route-in-hypercube** /* route using Hypercube link */

case 5: TAG = H /* Packet received through a Hypercube link, route it using a Mesh link */
if L > L*d* **then** Send packet to (L-1, X);// send it to previous level
else Send packet to (L+1, X); // send it to next level

End Procedure Packet-Routing.

Procedure Route-in-hypercube
Inputs:(L , X): current node address
(L*d* , D): destination node address
begin
Assume the current-node's hypercube address is
X= x_{k-1}….x_0;
Let Y = X ⊕ D /* the symbol ⊕ is the bitwise XOR operation
Let y_i be the right most bit of Y such that y_i =1
Send packet to (L , x_{k-1}...~x_i…x_0) ; where ~x_i is the bit complement of x_i
End Route-in-hypercube.
Figure 4.: Mesh-hypercube routing algorithm.

An Illustrative Example

To illustrate how the algorithm works, we consider a MH(3, 8) as shown in Figure 5. Suppose a packet is sent from the source (1, 000) to the destination (3, 111). Following the execution of procedure **Packet-Routing** at the source node (1, 000), case 1, case 2, and case 3 are not satisfied. Case 4 is satisfied since the TAG is equal to S, this leads to a call for procedure **Route-in-hypercube.** This procedure in turn sends the packet to (1, 001). At node (1, 001), case 5 is satisfied since the packet at this node has been received through a hypercube link (i.e., TAG = H) and $L < L_d$. As a result, the packet will be forwarded to (2, 001) using mesh link. At node (2, 001), case 4 is satisfied since TAG is equal to M, the packet will be forwarded to (2, 011) by **Rout-in-hypercube** procedure using a hypercube link. Similarly, at node (2, 001), the packet is sent to node (3, 011) in case 5 using a mesh link. Finally, case 1 is satisfied and the packet is sent by **Rout-in-hypercube** procedure to node (3,111) which is the destination. A path of length 5 is shown with arrows in figure 5.

To

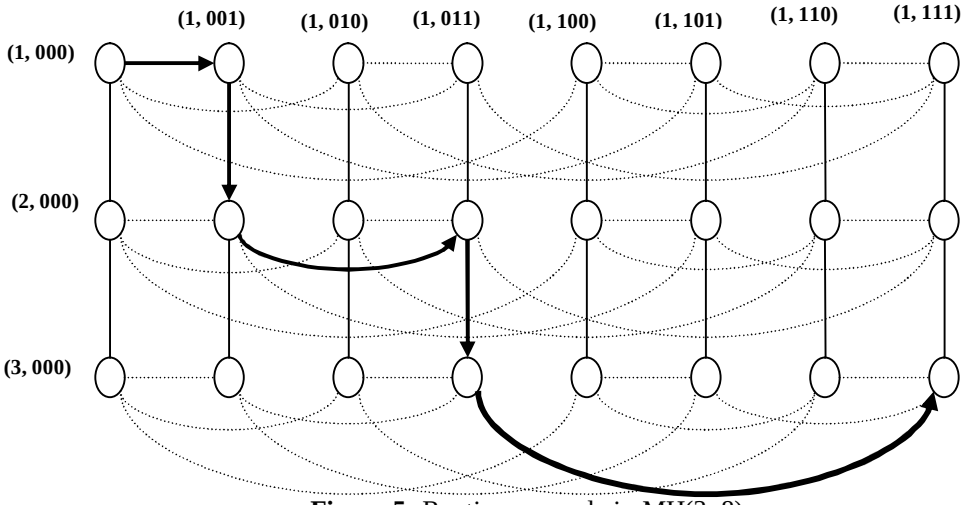

Figure 5: Routing example in MH(3, 8).

Analysis of the Algorithm

As we mentioned earlier the routing algorithm works in a distributed fashion where each intermediate node on the shortest path executes the algorithm when it receives a packet. The only information needed by the intermediate node to decide the next neighbor falling on the shortest path is the address of the destination node, i.e., L*d* and D. Therefore, the algorithm is optimal in space. Every step in the algorithm requires only a few (constant) clock cycles to execute (mostly bitwise operations). Therefore, the time complexity of the algorithm is *O*(1) and hence it's optimal in time.

The efficiency of a routing algorithm is generally measured by its *running time* which is the total number of communication time-units the algorithm requires to route packets to their destinations. It can be seen from the algorithm that the process to propagate the packet to its destination is done by modifying the source node level number or modifying the source node hypercube address bit by bit at the source node and at each intermediate node. Therefore, if the source and the destination are at the same hypercube then the algorithm takes at most O(log *n*) steps, since in the worst case the address of the source node will be transformed bit by bit to agree with the address of the destination node, and the number of bits in the address is *log n* bits where *n* is the number of nodes in the hypercube of MH(*m*, *n*). If the source and destination nodes are at the same mesh, then the algorithm takes at most O(*m*) steps. Otherwise, the source and destination are on different meshes and on different hypercubes, which is the worst case, the algorithm takes O(*m*+*log n*) steps. It should be observed that O(*m*+*log n*) corresponds to the diameter of MH(*m*, *n*) which represents the maximum distance between any two nodes [15], and hence no algorithm can route a packet between any two nodes in MH(*m*, *n*) in less than O(*m*+*log n*).

Simulation and Results Analysis

Simulations of the proposed packet routing algorithm on various MH networks have shown that there is a relative relationship between the injection rate of packets and the network performance measures. These measures are average network load, average waiting time, and average network latency. The simulation is conducted to compare the performance of the algorithm on different network configurations of the same number of nodes.

In the simulation, the following assumptions were set: the network is fault-free, each node has infinite queue size, packets at each node get their service based on First-Come-First-Serve (FCFS), all packets have the same size, packet injection rate follows an exponential distribution, source and destination nodes are selected randomly according to uniform distribution, a node cannot send a message to itself, and finally, a packet needs one-time unit to traverse a

link. The simulation is performed on MH(*m*, *n*) networks of size 64 nodes with different values of *n* and *m*. Specifically, MH(1, 64), MH(4, 16), and MH(8, 8).

Figure 6 shows the simulation results for the average network load. In general, figure 6 demonstrates a proportional association between the number of injected packets and the average network load. It can be seen that the worst network load was obtained for MH(8, 8), while the best results were achieved for both MH(1, 64) and MH(4, 16), respectively. It was noticed that the network load in MH network decreases with lower values of *m* and higher values of *n*. The reason is due to the fact that, both the average distance and the diameter of the hypercube subnetwork are less than those for a MH of the same size. Therefore, a network with small average distance between nodes and small diameter exhibit faster packet delivery which reduces the overall network load. On the other hand, increasing the value of *m* results in an increase in the diameter and the average distance of a MH network, which leads to a higher network load. Table 1 shows the simulation results of average network load for various MH networks taking into consideration different packet injection rates.

Table 1: Network load for various MH networks.

Packets	MH(4, 16)	MH(8, 8)	MH(1, 64)
1000	67.68	86.56	65.63
2000	136.62	161.95	127.234
3000	201.78	241.83	194.77
4000	269.18	323.02	252.23
5000	341.18	403.86	318.1
6000	402.79	485.28	377.22

T

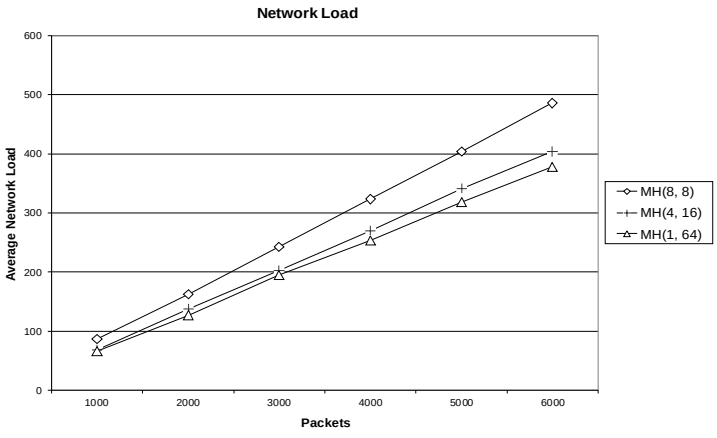

Figure 6:. Network Load for various MH networks.

Table 2 and figure 7 show the average waiting time for the packets in the three MH networks. The average waiting time in MH(4, 16) network is better than in both MH(1, 64) and MH(8, 8). The reason is that the packets in MH(4, 16) benefit from the large number of parallel shortest paths offered by the hypercube subnet works available in MH.

Table 2: Waiting Time for various MH networks.

Packets	MH(4, 16)	MH(8, 8)	MH(1, 64)
1000	12.34	33.08	18.55
2000	25.55	51.83	30.63
3000	34.88	77.77	44.69
4000	40.72	101.11	53.29
5000	47.84	125.75	63.18
6000	57.18	150.22	74.06

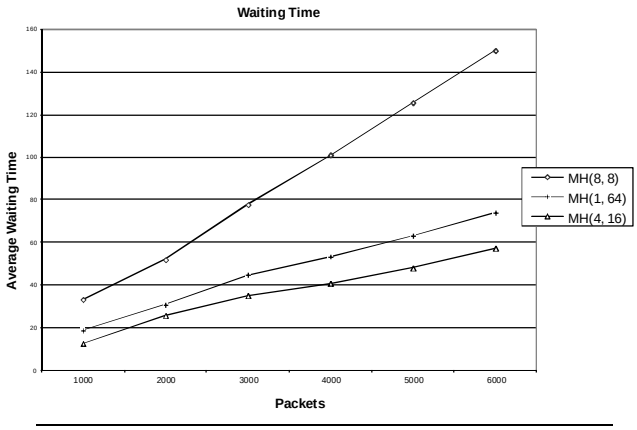

Figure 7: Waiting time for various MH networks.

Table 3 and figure 8 show a comparison between the packet latency on the various MH networks. The packet latency in MH(4, 16) tend to be less than in both MH(1, 64) and MH(8, 8). These results are attributable to the fact that the packets in MH(4, 16) network benefit from the available mesh links to escape a congested hypercube links. Therefore, the waiting time at the intermediate nodes of MH(4, 16) is less than that in the case of the MH(1, 64) and much better than that in MH(8, 8).

Table 3: Packet latency for various MH networks.

Packets	MH(4, 16)	MH(8, 8)	MH(1, 64)
1000	15.60	37.29	21.59
2000	28.80	55.96	33.68
3000	38.14	81.90	47.73
4000	44.00	105.24	56.32
5000	51.13	129.91	66.21
6000	60.47	154.37	77.08

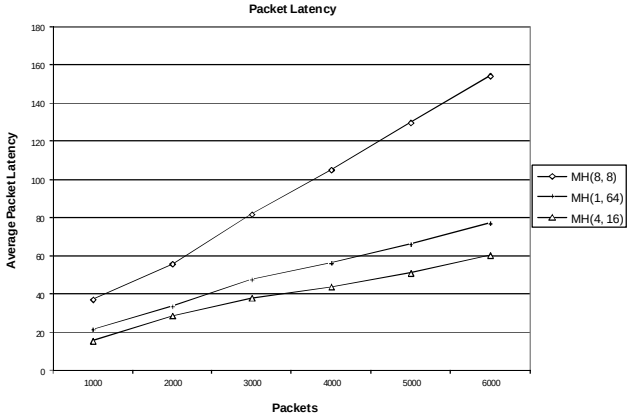

Figure 8: Packet latency for various MH networks.

4 0 CONCLUSIONS AND SUGGESTIONS

In this research, the routing capabilities of mesh-hypercube network have been studied. First, the length and the number of parallel shortest paths in MH were computed. Then a distributed routing algorithm was developed, analyzed, and simulated. The analysis shows that the communication time between every two nodes in a MH(m, n) is O(m + log n) since the algorithm uses only minimal length paths. The proposed algorithm is simple and requires a fixed number of execution cycles and appears to be well-suited to VLSI implementation. Simulation results revealed that the proposed routing algorithm performs better in mesh-hypercube networks with low values of m and large values of n. When trying different network designs by changing the values of m and n of a MH(m, n) and keeping the network size fixed, it was noticed that the worst values for the network performance measures were obtained when m=8 and n=8. The best results for both waiting time and packet latency were achieved when m=4 and n=16. This is due to the fact that the average distance and the diameter of the hypercube subnetworks are superior to those of mesh subnetworks. Therefore, the algorithm benefits from these features of the hypercube along with its high connectivity. In addition, the large number of parallel shortest paths in the hypercube subnetwork of MH led to small waiting time and latency.

REFERENCES

1. Adhikari, Nibedita and Chitta Ranjan Tripathy. 2014. Star-crossed cube: An alternative to star graph, Turkish Journal of Electrical Engineering and Computer Sciences, January, vol. 22, no. 3, pp. 719-734.
2. Alam, Mahfooz and Ankur K. 2015. A Comparative Study of Interconnection Network, International Journal of Computer Applications, October, vol. 127, no. 4, pp. 37-43.
3. Bokhari, S. 1990. Communication Overhead on the Intel ipsc-860 Hypercube, Icase Interim Report 10, ICASE, May.
4. Ho, C.-T and Johnsson, L. 1986. Distributed Routing Algorithms for Broadcasting and Personalized Communication in Hypercubes, Proceedings of the 1986 International Conference on Parallel Processing, pp. 640-648.
5. Nasir, Faizan and Jamshed Siddiqui. 2018. Comparative Analysis of Cube and Star Based Networks, INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING, November, vol. 6, no. 11, pp. 51-59.
6. Padmanbhan, K. and Lawrie, D. 1983. A class of Redundant Path Multistage Interconnection Networks, IEEE Transactions on Computers, December, vol. C-32, no. 12, pp. 1099-1108.
7. Saad, Y. and Schultz, M. H. 1987. Data Communication in Hyperbus, Journal of Parallel and Distributed Computing, vol. 9, pp. 115-153.
8. Omari, Mahmoud. 2014. Adaptive Algorithms for Wormhole-Routed Single-Port Mesh-Hypercube Network, IJCSI International Journal of Computer Science Issues, Vol. 11, No 1, January, pp. 66-73.
9. Al-Mahadeen, Bassam and Mahmoud Omari. 2004. A Broadcast Algorithm for All-Port Wormhole-Routed Mesh-Hypercube Network, Information Technology Journal, Vol.3, No. 3, pp.283-289.
10. Al-Mahadeen, Bassam and Mahmoud Omari. 2004. Adaptive Wormhole Routing in Mesh-Hypercube Network, Journal of Applied Sciences, Vol.4, No. 4, pp.568 – 574. https://doi.org/10.3923/jas.2004.568.574
11. Omari, Mahmoud and Mohammed Mahafzah. 2001. Fault-Tolerant Routing in Hypercube Networks, Al-Manarah Research Journal, published by AL al-bayt University, Vol. 7, No. 1, May, pp. 49-60.
12. Luccio, F., M. Mahafzah, M. Omari, and L. Pagli. 2000. Masked Interval Routing: A New Routing Scheme, The Computer Journal, Vol. 43, No.2, May, pp. 121-129.
13. Luccio, F., M. Mahafzah, M. Omari, and L. Pagli. 1998. "Routing with the Use of Masks," Proceedings of The 5th Colloquium on Structural Information and Communication Complexity (SIROCCO'98), pp. 188-200. June 22-24, 1998, Amalfi, Italy. Carleton Scientific.
14. Omari, Mahmoud. 2008. Packet Routing in Mesh-Hypercube Network. 2008. Abhath Al-Yarmouk Journal: Basic Science& Engineering Series, published by Yarmouk University, Vol. 17, No. 1(B), pp. 177-189.
15. Khan, Zaki A. 2016. Optimal Dynamic Scheduling Algorithm for Cube Based Multiprocessor Interconnection Networks, International Journal of Control Theory and Applications, December, vol. 9, no. 40, pp. 485-490.
16. Seo, Daeho, Ali, Akif, Lim, Won-Taek, Rafique, Nauman, and Mithuna Thottethodi. 2005. Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks, Proceedings of the 32nd Annual

International Symposium on Computer Architecture, June, 04-08, pp. 432-443.
17. Stout, Q. F. and Wagar, B. 1990. Intensive Hypercube Communication, Journal of Parallel and Distributed Computing, vol. 10, pp. 167-181.
18. Wu, Jie. 2002. A Deterministic Fault-tolerant and Deadlock-free Routing Protocol in 2-D Meshes Based on Odd-even Turn Model, Proceedings of the 16th international conference on Supercomputing, New York, New York, USA, pp. 67-76.
19. Ye, Terry Tao, Benini, Luca, and Giovanni De Micheli. 2004. Packetization and Routing Analysis of On-chip Multiprocessor Networks, Journal of Systems Architecture: the EUROMICRO Journal, February, vol. 50 no. 2-3, pp. 81-104.
20. Yeung, K. H. and Yu, T. S. 1997. Selective Broadcast Data Distribution Systems, IEEE Transactions on Computers, January 1997, vol. 46, no. 1, pp. 100-104. https://doi.org/10.1109/12.559808
21. Abualigah, L. M. Q. (2019). Feature selection and enhanced krill herd algorithm for text document clustering. Berlin: Springer.
22. Omari, M. 2003. Mesh-Hypercube: A Network Topology for Parallel Systems, Mu'tah Lil-Buhuth wad-Dirasat (Natural and Applied Sciences Series), Mu'tah University, vol. 18, no. 1, pp. 37-60.
23. Banner, R., Orda, A. 2007. Multipath Routing Algorithms for Congestion Minimization, IEEE/ACM Transactions on Networking, vol. 15, no. 2, April, pp. 413 – 424.
24. Nation, W. G. and Siegel, H. J. 1990. Disjoint Path Properties of the Data Manipulator Network Family, Journal of Parallel and Distributed Computing, vol. 9, pp. 419-423.
25. Gaughan. P., Dao, B., Yalamanchili, S., and Schimmel, D. 1996. Distributed, Deadlock-Free Routing in Faulty, Pipelined, Directed Interconnection Networks, IEEE Transactions on Parallel and Distributed Systems, vol. 45, no. 6, pp. 651-665.
26. Lan, Y. 1995. An Adaptive Fault-Tolerant Routing Algorithm for Hypercube Multicomputers, IEEE Transactions on Parallel and Distributed Systems, vol. 6, no. 11, pp. 1147-1152.
27. Raghavendra, C. S., Avizienis, A., and Ercegovac. M. D. 1984. Fault Tolerance in Binary Tree Architectures, IEEE Transactions on Computers, June, vol. C-33, no. 6, pp. 568-572.
28. Grammatikakis, M. D., Hsu, D.F., and Andsibeyn, J. F. 1998. Packet Routing in Fixed-Connection Networks: A Survey, Journal of Parallel and Distributed Computing vol. 54, no. 2, November, pp. 77–132. https://doi.org/10.1006/jpdc.1998.1483