The 2008 World Congress in Computer Science, Computer Engineering, and Applied Computing. pp. 69 - 75, July 14 - 17, 2008

An Efficient Routing Algorithm for Mesh-Hypercube (M-H) Networks

Syed S. Rizvi[1], Khaled M. Elleithy[1], and Aasia Riasat[2]
[1]Computer Science and Engineering Department, University of Bridgeport, Bridgeport, CT USA
[2]Department of Computer Science, Institute of Business Management, Karachi, Pakistan

Abstract - This paper presents an efficient routing algorithm for the Mesh-Hypercube (M-H) network. The M-H network is one of the new interconnection networking techniques use to build high performance parallel computers. The combination of M-H networks offers high connectivity among multiple nodes, fault-tolerance, and load scalability. However, the performance of M-H networks may degrade significantly in the presence of frequent link or node failures. When a link or node failure occurs, neither the hardware schemes nor point to point and multistage routing algorithms can be used without adding extra links. This paper presents an efficient single bit store and forward (SBSF) routing algorithm for M-H network that based on the round robin scheduling algorithm. Simulation and numerical results suggest that the proposed routing algorithm improves the overall performance of M-H network by both reducing the transmission delay and increasing the total data throughput even in the presence of faulty nodes.

Keywords: High performance systems, Mesh-Hypercube networks, routing algorithm, fault-tolerance, and load scalability

1 Introduction

A key component of high performance parallel computer architecture is the interconnection network topology which enables the communication among different nodes of a large parallel computer. Several different interconnection network topologies have been proposed over the years [1, 2, 3]. One of the most popular, new, and an efficient network topology is the mesh-hypercube (M-H) networks. This network has been used in different systems such as n-Cube, Intel IPSC [13, 14, 15].

With the continuous growth in network size, it is hard to find a large size network such as an M-H network without the presence of faulty nodes and or links. It has been shown that the performance of M-H networks degrades as the number of faulty links increases in a large parallel network [11, 12]. When the numbers of faulty links or nodes increase in the parallel system, the non-faulty reachable nodes of the network decrease proportionally. This, therefore, results performance degradation due to the lack of efficient routing algorithm that can provide full reachability to each non-faulty node of the parallel network.

This paper presents a new and an efficient single bit store and forward (SBSF) routing algorithm for M-H network that based on the round robin scheduling algorithm. The proposed SBSF routing algorithm can be implemented on a large parallel network which may consist of multiple nodes where each node can send and receive random point-to-point messages. The primary objective of this paper is to simulate the data throughput of an M-H network as well as its performance in the presence of faulty nodes and or links and hot spots. In addition, the simulation results of this paper allow one to examine the impact of different architectures on the overall performance of the M-H networks. These different architectures include dimension of the M-H networks, input parameter values such as the variation in the total message generation rate per clock cycle, and the introduction of faulty links. We show that the performance of M-H networks remains consistent even in the presence of multiple faulty nodes. Our numerical analysis shows that the proposed algorithm maintains a consistent performance even in the presence of faulty nodes by making non-faulty nodes fully reachable.

The combination of M-H network offers several attractive features. First, it has high connectivity between the various processors [5]. In a P processor system, a message must traverse no more than $\log_2 (P)$ links before reaching its final destination. Second, it is a multiple instructions multiple data (MIMD) machine, allowing different nodes of a network to implement different component models [2]. Third, the architecture has good scaling properties for a large value of P [3].

As the numbers of nodes grow in an M-H network, the number of links required at each node increases logarithmically where as the required communication bandwidth increases linearly. Nodes in an M-H network can also be viewed as a directed graph where all interconnections between the nodes occur in a synchronous order. In addition, for the sake of experimental verification, we assume that each link in an M-H network may carry no more than one unit

message in one step where each node during a step can send at most one message to each of its neighboring node.

The remainder of this paper is organized as follows. Section 2 presents some of the well known routing algorithms for M-H networks. Section 3 presents the proposed SBSF routing algorithm along with a comprehensive discussion on its implementation. In Section 4, we examine and simulate the data throughput and the hot spot properties to determine the overall performance of the proposed algorithm. Finally, section 5 concludes the paper.

2 Routing algorithms for M-H networks

Substantial efforts have been devoted to the study of message routing for M-H networks [1, 4, 5]. Routing algorithms for M-H network are broadly classified into two categories: *oblivious* routing and *adaptive* routing [9, 10]. In an *oblivious* routing algorithm, the path of one message is unaffected by the presence of other messages in the network [6]. On the other hand, in an *adaptive* routing algorithm, messages may be directed away from the congested parts of the network [5, 7]. Optimal routing for given paths on arbitrary networks has been studied extensively in the context of store-and-forward algorithms. Routing algorithms, in which packets do not strictly follow specific paths, are *matching routing* and *hot-potato routing* [7, 8].

In randomized routing, a message is sent from a source to a destination in two stages [9, 10]. In the first stage, the message is sent from the source to a random intermediate node. In the second stage, the intermediate node forwards the received message to the intended destination. In each stage of randomized routing, the message is routed using the *bit-fixing* algorithm, which is also known as *dimension-order* and the *e-cube* routing algorithms. The randomized routing algorithms are fast and can solve any one-to-one packet routing problems on a *P*-node hypercube in the order of $O(\log_2 P)$ packet steps [9]. One drawback to such algorithm, however, is that they all require in order of $O(\log_2 P)$ bit steps when implemented on real parallel machines such as the connection machines. Since packets always contain at least in order of *log (P)* bits of addressing information, a typical packet step really consists of *O (log P)* bit steps [10].

In bit-fixing routing algorithm, dimensions are ordered in an arbitrary way and a message is always typically directed along the lowest dimension of the network in which its current position and its final destination differ [11]. A simple implementation of this algorithm is shown in Fig. 1. The routing via bit-fixing algorithm can take exponential time. Consider a permutation on the n-cube where n = 4 and to route a packet from 0000 to 1111, the bit-fixing routing algorithm uses the following path: $0000 \rightarrow 1000 \rightarrow 1100 \rightarrow 1110 \rightarrow 1111$. It has been shown [9] that this simple permutation causes the bit-fixing routing algorithm to take in order of $\Omega(2^{n/2})$ steps to route a message. From the performance point of view, the resultant computational complexity is not desirable.

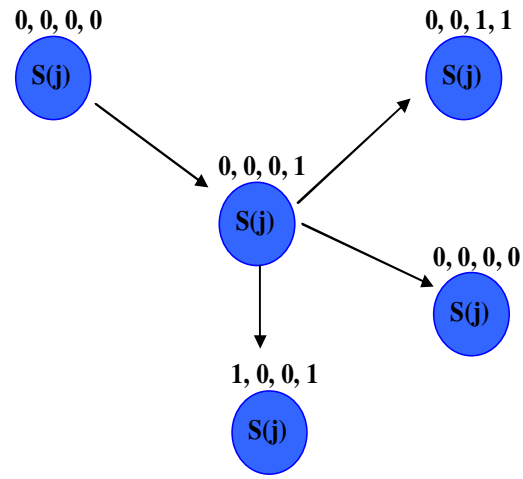


Fig.1. Bit-fixing routing algorithm

3 Proposed SBSF routing algorithm for the M-H networks

We propose an efficient SBSF message routing algorithm for the M-H networks which based on a simple round robin scheduling algorithm. The proposed SBSF algorithm shows some good load-scalability characteristics among the M-H nodes. One of the reasons for achieving high scalability is due to the use of round robin scheduling algorithm. Instead of separately using a scheduling algorithm on each output queue, we implement the round robin algorithm as a part of our proposed routing algorithm. By doing this, we greatly reduce the average waiting time of each output queue associated with each outgoing link. Our choice for using the round robin scheduling algorithm with the proposed SBSF algorithm is due to the fact that it is both simple and easy to implement as well as starvation-free. The advantages of the proposed SBSF routing algorithm such as the reduced average waiting time per output queue and consequently the improved data throughput can be seen in our simulation results.

3.1 Proposed M-H model and assumptions

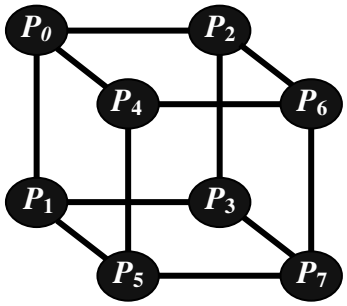
Before going to present the proposed SBSF routing algorithm for an M-H network, it is worth mentioning some of our key assumptions:

1. We assume that the total number of nodes '*P*' in the M-H network is equal to 2^k where *k* represents the dimension of a hypercube.
2. The total nodes *P* are identified with a unique node number in the range of 0 to $2^k - 1$.
3. Every node in an M-H network has *k* bi-directional links with the other nodes.

4. Two nodes are said to be connected if and only if their binary addresses representation differs in exactly one bit order.
5. The binary representation of a node in an M-H network with k dimension has k bits (the exception is the singular case of hypercube with $k = 0$).

A shortest path between any two nodes of an M-H network is represented as a sequence of nodes $P_0, P_1, \ldots\ldots, P_i$ where subscript i indicates the length of the path. In an M-H network, multiple shortest paths may exist between any two nodes. For instance, if there are P total nodes exist in an M-H network which is numbered from 0 to P-1 in such a way that addresses of connected nodes differ exactly by one bit in their binary representation (see Fig. 2), then the M-H network exhibits some desirable properties.

It can be seen in Fig. 2 that the routing in the cube is simplified to find any direct neighbor that reduces the number of bit differences between the address of the recipient and the messenger. As an illustration, let P_0 and P_7 be the nodes in a k-dimensional M-H network. The total number of bit positions at which these binary equivalents of P_0 and P_7 differ can be considered as the hamming distance H. In other words, the length of the shortest path between P_0 and P_7 is equal to the hamming distance H. For instance, in Fig. 2, in order to route a message from a node P_0 $(000)_2$ to a node P_7 $(111)_2$, nodes P_4 $(100)_2$ and P_6 $(110)_2$ provides an optimal route since at each given node along the path, the bit differences with P_7 $(111)_2$ are reduced by one.



$P_0 = 0 = 000$
$P_1 = 1 = 001$
$P_2 = 2 = 010$
$P_3 = 3 = 011$
$P_4 = 4 = 100$
$P_5 = 5 = 101$
$P_6 = 6 = 110$
$P_7 = 7 = 111$

For example:
$P_0 = 0 = 000$ can go to P_1, P_2, and P_4. This can also be represented as:
$P_0 = 0 = 000$ → $P_1 = 1 = 001$
$P_2 = 2 = 010$ →
$P_4 = 4 = 100$ →
H. Distance *One Bit*

Fig.2. Numbering scheme for three dimensional M-H networks

3.2 Proposed SBSF message passing algorithm

Fig. 3 provides the formal description of the proposed SBSF algorithm. The proposed SBSF routing algorithm starts by searching nodes in an M-H network. The searching is performed in a sequential manner starting from the minimum value of the node-address to a maximum value. For instance, in a three dimensional hypercube within an M-H network, the proposed routing algorithm starts searching the node addresses from *000* and proceeds in a sequential manner until it reaches the maximum value (i.e., *111*). In other words, this sequential searching of nodes will continue until the node array becomes empty. Once it reaches to the end of the array, the algorithm starts over again and this cycle of searching goes on.

We assume that each node in an M-H network can either generate or receive a message but not both during the same clock cycle. Fig. 4 shows the internal architecture of each node in an M-H network that includes one processor P, one communication interface C, and four incoming and outgoing links. Each link in the node architecture has an output queue for outgoing messages. Also, it should be noted that there is no input queue for the incoming messages as shown in Fig. 4. This assumption implies that each incoming message for a destination node simply goes into sink. We assume that a node can receive or send a message during a cycle. This implies that while C receives a message from a different node through one of its incoming links, it can not send a message using an output queue from a link different than the one currently receiving a message as shown in Fig. 4. This assumption further leads us to the following two facts: An incoming message for a destination node can either directly go into an output queue or be absorbed by the processor. A message needs to wait at least one cycle before it can be sent once it is received. In other words, you cannot receive a message and send it out during the same cycle as if the node did not exist.

For the sake of simulation, we assume that an M-H network can have at most 256 nodes at one time. In other words, for ease of understanding, our simulation supports at most 8 dimensional hypercube within an M-H network that can have at most 256 nodes. In addition, step 2.3 of Fig. 3 shows the use of round robin scheduling algorithm for selecting appropriate communication link to receive messages. For instance, when a certain node is in the busy state (i.e., the current state of the node is either sending or receiving or idle), it can not receive messages from other nodes. Once the proposed routing algorithm gets a free-node, it implements the round robin scheduling algorithm to check the status (Busy or Free) of each outgoing link attached to the output queue. These communication links are further attached to other nodes as shown in Fig. 5.

Once the proposed SBSF algorithm gets a non-busy link, it performs the following tasks. First, it checks the output queue of the attached node (destination node) to determine is there any message left in the queue. If the output queue does not contain a single message, the round robin scheduling

algorithm will move to the next immediate communication link. Secondly, if the output queue contains at least one or more messages, the sinking node starts retrieving the message(s) from the output queue of the attached destination node. Finally, the round robin scheduling algorithm changes the status of both the sinking node and the communication link and set them as busy.

The important point that we should note here is the advancement of the round robin scheduling algorithm from the current link to the next communication link. If we do not advance to the next link, the proposed SBSF routing algorithm does not achieve the load-scalability that results higher values of average waiting time in each output queue. The higher values of average waiting time in each output queue would likely to reduce the overall data throughput of an M-H network.

4 Experimental verifications and the simulation results

We categorize our performance analysis of proposed SBSF routing algorithm for an M-H network in two parts. In

Formal Specification of Proposed SBSF Routing Algorithm for M-H Networks

1- Starts searching the nodes
Search Node_ Address [Minimum] to Node_ Address [Maximum]
2- Check the status of each visiting node
2.1. [Set Node status]
Set Node Status = = Sending or
Set Node Status = = Receiving
[Repeat Step 2.2 or Step 2.3] While Node ≠ NULL
2.2. [Check the Condition] If Node Status = Sending Then
Set Node Status = = Creating a Message or
Set Node Status = = Receiving a Message or
Set Node Status = = Sending a Message or
Set Node Status = = Idle
[End of Step 2.2]
[Go back to step-2] Advanced to Next Node
2.3. [Check the Condition] If Status = Receiving Then
2.3.1. Implement Round Robin Scheduling Algorithm
[Start Sequential Search for Node Status]
2.3.1.1. If Node = Busy Then
2.3.1.1.1. Advanced to Next Node
2.3.1.1.2. Go back to Step 2.3.1
2.3.1.2 If Node ≠ Busy Then
[Start Sequential Search for each outgoing link status]
2.3.1.2.1. If Link = Busy Then
Advanced to Next Lin
2.3.1.2.2. If Link ≠ Busy Then
[Get message from the free-node]
[Change the status of each node]
Set Sink Node ==BUSY
Set Destination Node = BUSY
2.3.1.2.3. Advanced the Link
2.3.1.2.4. Go back to Step 2.3.1
[End of Step 2.3(Round Robin Scheduling Algorithm for Message Reception)]
[Go back to Step 2] Advance to Next Node
2.4. [End of Step-2*]*
[Go back to Step-1]
3- [End of Step-1*]*
[EXIT]

Fig. 3. Formal description of the proposed SBSF routing algorithm

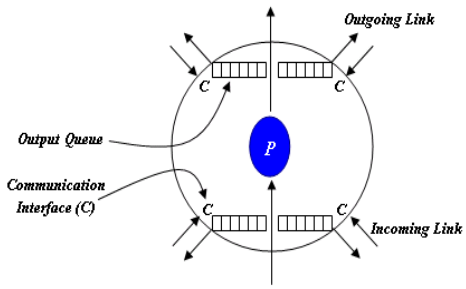


Fig.4. Internal architecture of a node in a hypercube network

the first part, we present our performance analysis when all communication links are in the operational mode where as in the second part faulty links or nodes will be considered.

4.1 Performance analysis in the absence of faulty links or nodes

For this section, a three dimensional M-H network is considered that can have at most 8 nodes. Furthermore, we run all simulations for 10000 cycles. In addition, the hot spot property is also set to zero (i.e., the probability of creating hot spot is set to zero to ensure an ideal condition for an M-H network).

Fig. 6 shows the *average waiting time* in the output queue and the maximum output *queue size* with respect to a range of message transmission probabilities. Average waiting time in the output queue is the amount of time by which each packet needs to wait until it reaches to the final destination node as shown in Fig. 6. On the other hand, the maximum link output queue size is the size found on any given node during the simulation. As we increase the probability of message transmission, nodes in M-H network create more messages and send them to one or more output queues where

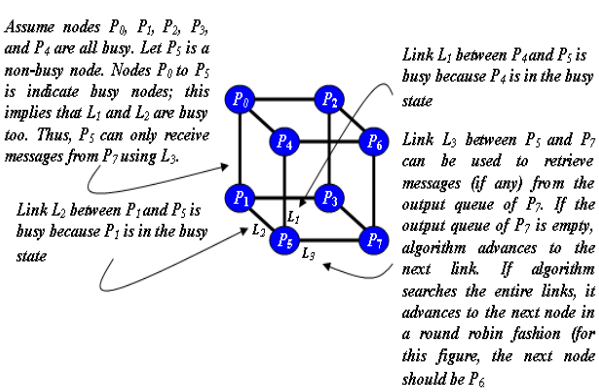


Fig.5. Implementation of the proposed SBSF message passing routing algorithm

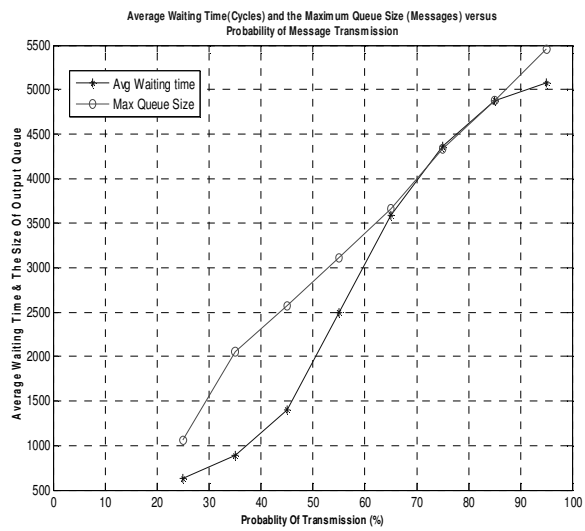


Fig.6. Average waiting time (Cycles) and the maximum queue size (Messages) versus probability of message transmission

they reside until they get a chance to transmit to other nodes.

It can be seen in Fig. 6 that the increase in probability of message transmission results a large number of message creation which might make the output queue congested. As the output queue becomes congested due to a linear increase in the probability of message transmission, the average waiting time per node in the output queue increases significantly. In addition, the simulation result of Fig. 6 proves the correctness of our proposed SBSF routing algorithm.

Fig. 7 shows the data throughput with respect to the probability of message transmission. The data throughput is the product of utilization per node and the total messages received successfully. In harmony to our expectations, the reduction in the messages received per node causes a reduction of data throughput for the M-H network. It should be noted that the data throughput in Fig.7 represents the number of messages received per node in total cycles (i.e., we set the design parameter C_{Total} 10000 cycles for all simulations that we run). For instance, according to our numerical analysis, when we have 25% of probability, the total number of messages received is around 13043 that generate approximately 4891 messages per node. This implies that each node processes approximately half messages (0.489 messages) of information per clock cycle. It can be seen in Fig. 7 that the throughput reduces due to the reduction in the total messages received successfully.

4.2 Performance analysis in the presence of faulty links or nodes

For this section, we introduce faulty links and nodes. As one can easily observe that the introduction of faulty links and nodes in M-H network reduces message transmission and

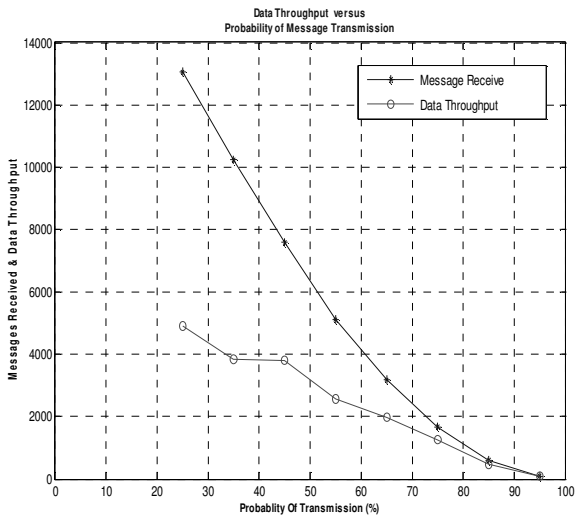


Fig.7. Data throughput versus probability of message transmission

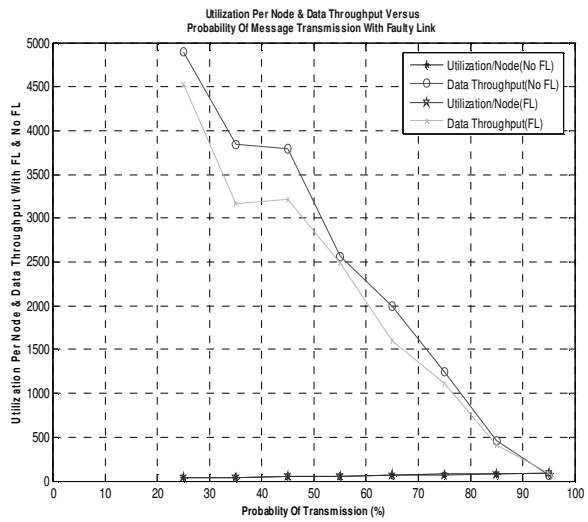


Fig.9. Utilization per node and data throughput versus probability of message transmission with faulty link (FL)

reception as well as a slight decrease in overall data throughput.

By carefully looking at the simulation results of Fig. 8, one can make a conclusion that broken links in M-H network are more affected to message reception as compared to message transmission. With harmony to our expectations, the simulation results of Fig. 8 shows a nice reduction in both message transmission and reception.

Fig. 9 shows the simulation results for utilization per node and the data throughput in the presence of faulty links. It should be noted in Fig. 9 that an average utilization per node in the presence of faulty link is the same as in the presence of no faulty link except at one place. This is due to the fact that the utilization per node heavily relies on the total number of messages generated and received. As we have mentioned in Fig. 8 that the presence of faulty links and nodes does not have any sever effects on the message creation as compared to the message reception. This implies that the message creation in the presence of faulty link dominates the total number of messages created and received per node and thus cancels out the effects of reduced message reception on the utilization.

This is one of the reasons that why utilization per node remains the same for both faulty and non-faulty link. On the other hand, the data throughput in the presence of faulty link has decreased slightly as shown in Fig. 9. It should be noted that the data throughput is a product of utilization per node and the total number of messages received successfully. Since the utilization is same for both faulty and non-faulty links, the reduction in the total number of received messages due to a faulty link causes a slight decrease in the data throughput. However, the overall data throughput performance is almost overlapping for most of the values of transmission probability as shown in Fig. 9.

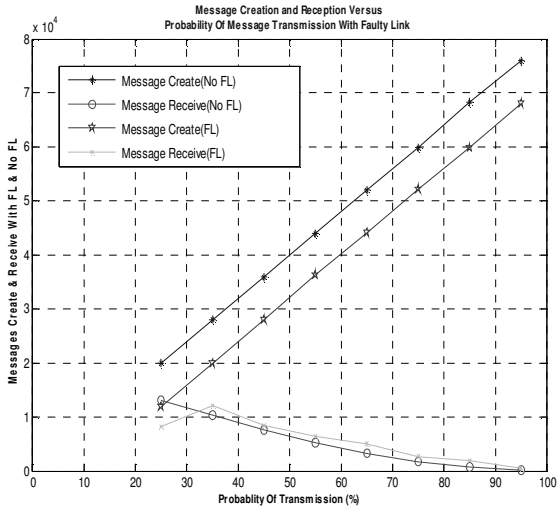


Fig.8. Message creation and reception versus probability of message transmission with faulty link (FL)

5 Conclusion

In this paper, we presented a new SBSF routing algorithm for an efficient transmission and reception of messages between the nodes within an M-H network. Our simulation results have shown that the proposed algorithm maintains a consistent performance even in the presence of faulty nodes or links. In addition, our experimental verifications suggest that the proposed routing algorithm improves the overall performance of the M-H networks by providing scalable performance for large networks. In addition, both numerical and simulation results presented in this paper show the effectiveness of the proposed SBSF

algorithm by minimizing the average waiting time per output queue as well as increasing the total data throughput for both faulty and non-faulty links.

6 Reference

[1] S. Cheng and J. Chuang, "Varietal hypercube-a new interconnection network topology for large scale multicomputer," International Conference on Parallel and Distributed Systems, Vol. 19, Issue 22, pp. 703 – 708, Dec 1994.

[2] Z. Sergio N., "Analysis of Cluster Interco connection Network Topologies," M. S. Thesis, The University of Texas At El Paso, July 2004.

[3] A. Louriand and H. Sung, "An optical multi-mesh hypercube: a scalable optical interconnection network for massively parallel computing," Journal of Lightwave Technology, Vol. 12, Issue. 4, pp. 704 – 716, Apr 1994.

[4] T. Liu, W. Huang, F. Lombardi et al. "A submesh allocation scheme for mesh-connected multiprocessor systems," In Proc. Int. Conf. Parallel Processing II, 1995, pp.159-163.

[5] B. Al-Mahadeen and M. Omari, "Adaptive wormhole routing in mesh-hypercube network," Journal of Ap- plied Sciences, vol. 4, no. 4, pp. 568–574, 2004

[6] C. Kaklamanism, D. Krizanc, and T. Tsantilas, "Tight bounds for oblivious routing in the hypercube," Proceedings of the second annual ACM symposium on Parallel algorithms and architectures, pp. 31 – 36, 1990.

[7] I. Ben-Aroya, D. Chinn, A. Schuster, "A lower bound for nearly minimal adaptive and hot potato algorithms," *Algorithmica*, vol. 21, pp. 347-376, 1998.

[8] C. Busch, M. Herlihy, R. Wattenhofer, "Hard-potato routing," *in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pp. 278-285, 2000.

[9] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, January 31, 2005.

[10] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 2000.

[11] M. Harrington, "New method for synchronizing distributed systems in the presence of faults," *MSEE thesis, Dep. Elec. Eng., Univ. of Washington*, March. 1991.

[12] M. Harrington and A. Somani, "Synchronizing hypercube networks in the presence of faults," *IEEE Trans. on Computers Archive*, vol. 43, Issue 10, pp. 1175 – 1183, 1994.

[13] C. McCreary, M. McArdle, J. McCreary, "Broadcast communication delay metric for the iPSC/2 and iPSC/860 hyper-cubes," Proceedings of the 30th annual Southeast regional conference, Session 3A, pp. 53 – 60, 1992.

[14] A. Littlefield, "Characterizing and tuning communication performance on the Touchstone DELTA and iPSC/860," In Proceedings of the 1992 Intel User's Group Meeting, Dallas, TX, October 4-7 1992, Intel Corp

[15] L. Bomans and D. Roose, "Communications Benchmarks for the iPSC/2", Hypercube and Distributed Computers, Elsevier Science Publishers B.V., North-Holland, 1989.