An Efficient Adaptive Fault-Tolerant Routing in Hypercubes in the Presence of Dynamic Faults

Hamid Reza Zarandi
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
zarandi@ce.sharif.edu

Hamid Sarbazi-Azad
School of Computer Science
Institute for Studies in Theoretical Physics & Maths (IPM), Tehran, Iran
azad@sharif.edu/azad@ipm.ir

Abstract

This paper presents a new fault-tolerant routing algorithm in hypercube multicomputers based on the local-information of each node. In this algorithm, each node is only aware of its neighbors' status and makes routing decision using this information and the destination address of a message. This bounded local-information, whose values are numerable, causes the algorithm to be more adaptive with respect to the similar works, e.g. probability vector. Due to the low overhead of communications needed for constructing the local-information, dynamic faults are highly tolerable as well as static faults. Simulation results reveal that, in the case of static faults, the latency of messages is reduced up to 20 percent and in the presence of dynamic faults, the throughput of message delivery with wormhole switching is considerably improved.

Keywords: Hypercube, Fault-tolerance, Routing algorithm, Static/Dynamic fault.

1 Introduction

The success of large-scale multicomputers is highly dependent on the efficiency of their underlying interconnection networks. The performance of a given multicomputer network greatly depends on its routing algorithms which specify the path between two nodes involved in an end-to-end of communication. The hypercube has been one of the most popular networks for multicomputers due to its desirable and powerful topological properties, including regular structure, low diameter, and ability to exploit communication locality. The iPSC [1], N-CUBE [2], and SGI2000 [3] are examples of practical systems that are based on the hypercube.

Routing in fault-free hypercube has been studied in the literature [4], [5], [6], [7]. Some of the most important issues in the design of a routing algorithm are high throughput, low latency message delivery, avoidance of deadlock, livelocks and starvations and ability to work under various traffic patterns [8].

As the number of processors in a network increases, the probability of processor failure also increases. The failures can be either static or dynamic. Static failures are present in the network when the system is powered on and dynamic failures occur at random during operation. Dynamic faults can interrupt a message in progress and are more difficult to handle since pipelined switching techniques, e.g. wormhole switching, are particularly susceptible to faults due to the existence of dependencies across multiple routers. Consequently, system reliability becomes a key issue in the design of large scale multicomputer. Fault tolerance is ability of a network to function in the presence of component failure. For networks with faults, a routing algorithm should exhibit a few additional features: graceful performance degradation, and ability to handle faults with only a small increase in routing complexity and local knowledge of faults –each nonfaulty processor knows only the status of its neighbors [8].

If each node is equipped with the information on all faulty components, then it can always determine a fault-free path for every message to its destination. However, this is usually too costly (in space and time), especially when the network is large. Hence, it is important to develop routing schemes which require each node to keep only the failure information essential for making correct routing decision.

For the above reasons, we develop a routing scheme which requires each node to know only the condition of its own links and neighbor nodes. This paper describes a fault-tolerant routing for hypercubes with dynamic and static faults which routes messages as well as the probability vector algorithm [9] with reduced complexity. The main feature of the routing is its ability to capture network performance when location and number of failure nodes change dynamically. Simulation results reveal that the latency of a typical message is reduced especially when number of faults increases. Compared to the similar fault-tolerant routing, proposed algorithm exhibits a considerable throughput. The algorithm makes decision adaptively based on local failure information only and is simple to implement and needs a very small message overhead. A performance comparison against the probability-vector algorithm through extensive simulation experiments is conducted. The results reveal that the new routing algorithm outperforms the probability vector algorithm in terms of routing distance, calculation time, and throughput.

2 Related work

Lee and Hayes [10] have used the concept of unsafe nodes to design a fault-tolerant strategy. Message routing is achieved by avoiding unsafe nodes, which could possibly lead to communication difficulties and excessive delays. Chiu and Wu [11] used the concept of unsafe nodes with some extensions and showed that a feasible path of length not more than the Hamming distance plus four can be guaranteed provided that the number of faulty nodes does not exceed n-1. The concept of unsafe nodes has also been used in [12] for broadcasting in faulty hypercubes.

Su and Shin [13] present an adaptive fault-tolerant routing for meshes and hypercubes using two virtual channels per physical channel. In their algorithm, the network is decomposed into two virtual networks to support reliable and fully-adaptive routing. Their method can tolerate faulty cubes which has no more than n/2 faulty nodes.

Gordon and Stout [14] have proposed a fault-tolerant routing based on sidetracking, where a message is de-routed to a randomly chosen fault-free neighboring node when there is no fault-free neighbor along any of existing optimal paths leading to the destination. With this approach a routing failure may occur. Furthermore, excessive delay may arise even in the presence of few faulty components [11].

Chen and Shin [15] have proposed a routing strategy based on depth-first search in which backtracking is required if all the required forward links cannot be used due to faulty components. The traversed path is recorded and attached to the message. A simplified version of this approach that tolerates fewer faults was presented in [16], where routing is progressive without backtracking, and a message is routed to its destination on an optimal path with high probability. Lan [17] has presented a fault-tolerant routing algorithm based on local information which guarantees an optimal and near-optimal routing. However, the algorithm is based on a restricted model of fault distribution as it can tolerate only n-1 faulty nodes (and/or links) in an n-cube.

Wu [18] has presented the concept of safety levels, based on the limited global information, as an enhancement of the unsafe node concept. The safety level is an approximate measure of the number as well as the distribution of faulty nodes. Optimal routing is guaranteed if the safety level of the node is no less than the Hamming distance between the source and destination. Chiu and Chen [19] have proposed the concept of routing capability, which further enhances the safety level concept. Using the routing capability concept, n binary bits are attached to ever node. A node is called k-capable if the k-th bit is 0, in which case optimal routing to nodes at Hamming distance up to k is guaranteed.

The Safety vectors algorithm proposed in [20] is similar concept to the routing capability, with some extensions related to dynamic adaptivity analysis and application to the generalized hypercube. The safety vector approach requires each processor to maintain a bit vector (called safety vector) computed through a number of fault information exchanged between adjacent processors. The algorithm guarantees optimal routing to all destination that are at a Hamming distance k from a given node A, if and only if, the k-th bit of safety vector at node A is set. The major drawback of this algorithm is related to its conservation approach. Indeed, when the k-th bit of the safety vector of node A is not set, node A is not considered for forwarding messages to any destination at distance k from A. Even when all destinations at distance k are reachable from node A via fault-free optimal paths except for one such destination, node A is excluded as a forwarding node for all these destinations. This causes high percentages of suboptimal routing and routing failure.

A limited-global-information-based algorithm was proposed in [21] for the hypercube, which is based on a calculated probability vector of every node. The probability vector approach requires each processor to maintain a vector of length n of floating-point numbers computed through n-1 information exchange of fault information between adjacent processors. The algorithm routes messages by the expected length of routing path via each of its neighbors. However, the specified routing algorithm uses spare dimensions with some restrictions which are not necessary. Moreover, the calculation of the probability vector is approximated leading to considerable errors in some cases.

The rest of this paper is organized as follows. Section 3 gives the notation and preliminaries used in the next sections. Section 4 presents the proposed fault-tolerant algorithm and then derives some of its properties. Section 5 presents a comparative evaluation between the new algorithm and the probability vectors algorithm through simulation experiments. Section 6 concludes the paper.

3 Notation and Preliminaries

The n-dimensional hypercube (or binary n-cube) is an undirected graph with 2^n vertices (nodes or processors) labeled by 2^n binary strings of length n. Two nodes are joined by an edge if, and only if, their labels differ in exactly one bit position. The label of node A is written as $a_n\, a_{n-1} \dots a_1$, where $a_i \in \{0, 1\}$ is the i-th bit (or bit at i-th dimension). The neighbor of a node along the i-th dimension is denoted by $A^{(i)}$. The *hamming distance* between nodes A and B, denoted as H(A, B), is the number of bits at which their labels differ. In other words, $H(A, B) = |A \oplus B|$ where $\oplus$ denotes the exclusive-or binary operation. |X| gives the number of 1's in the binary pattern of X. A path between two nodes A and B is minimal path if its length is equal to H(A, B).

If the distance between node A and node B is d then $A \oplus B$ has value a bit value 1 at d positions corresponding to d distinct dimensions. These d dimensions are called *preferred* dimension, while the remaining n-d dimensions are called *spare* dimensions. If a shortest path is not optimal, it contains one or more spare dimensions in addition to the preferred dimensions. A faulty dimension refers to either a faulty neighboring node or a faulty link at that dimension.

4 The proposed fault-tolerant routing algorithm

The routing process makes decisions based on the probability of the all preferred and spared neighbors. Routing through a spare neighbor increases the routing distance by two over the minimal distance. A minimal path can be obtained by routing through all preferred dimensions in some order. We make the following assumption in this paper: 1) a faulty n-cube contains faulty nodes and/or links, 2) there are some algorithms for fault detection and diagnosis that are responsible for activation of the probability recalculation and 3) each of two nodes at the end sides of a faulty link considers the node at the other end as faulty. We furthermore, assume that each node knows the probability of all its neighbors and can distinguish an adjacent link from an adjacent faulty node when probability needs to be calculated.

The proposed fault-tolerant algorithm is an alternative approach to the probability vector method [21]. In [21] a probability vector was used for routing messages in the networks. However, the specified routing algorithm uses spare dimensions with some restrictions which are not necessary. Moreover, the calculation of the probability vector is approximated leading to considerable errors in some cases. The basic steps of this algorithm are:

1) each node determines number of its faulty neighbors and then calculates probability of a neighborhood to be faulty,
2) each node exchanges its probability to each dimension one time and
3) performs efficient fault-tolerant routing.

4.1 Calculation of the probability

In the proposed approach, local-fault information is captured in a real number P_C which is between 0 and 1, that is associated with each node C in an *n*-cube. Specifically, P_C represents the routing capability of node C to route a message received from one of incident edges. This probability is calculated by the following algorithm.

```
Algorithm LOCAL_STATUS (C: node)
/* Determine the probability of faulty neighbors for node C */
begin
        $P_C = 0$;
        for $i = 1$ to $n$ do
           if $C^{(i)}$ is faulty then
                 $P_C = P_C + 1/n$;
        next i
end
```

Each node uses the probability of each of its neighbors and calculates the *effective* probability according to the following equation. This is due to the fact that current node has additional information about itself and thus can accurate its knowledge from the faulty neighbors.

$P^*_{C^{(i)}}$ = Prob. {a neighbor of $C^{(i)}$ be faulty | node C is not faulty}

$$P^*_{C^{(i)}} = \begin{cases} P_{C^{(i)}} \,.\, n/(n-1), & \text{if } C^{(i)} \text{ is not faulty} \\ 1, & \text{if } C^{(i)} \text{ is faulty} \end{cases} \qquad (1)$$

4.2 The fault-tolerant routing

In this algorithm, routing decisions are made via the status of the local-information stored in each neighbor of a node. In other word, each node that contains a message for routing, sends it to one of its spare dimensions which has most non-faulty neighbors. Hence, routing in the next hop of the message during the routing path has been guaranteed because it has at least one non-faulty neighbor. Therefore, routing in this scheme is greedy due to the fact that each node routes a message based on the information stored in its neighbors and is not aware of the faulty components in the network. Information of each node is about of all nodes within the two-hop distances. This scheme attempts to be far away, at least 2 hops, from the faults in the network and sends messages to a node which has an escape channel that prevents to failure.

```
Algorithm Neighbor_Based_Routing (M: message; C, D: node)
begin
    N = C ⊕ D;
    H = |C ⊕ D|;
    if H=0 then
    /* The current node is destination */
        Deliver_Message (M);
    if M.Routing_Distance < H + 2 * no_of_faults then
    /* routing distance of message is less than maximum threshold*/

    if ∃ i (N(i) = 1 ∧ $P^*_{C^{(i)}}$ <1 ∧ (∀ k | N(k) = 1. $P^*_{C^{(i)}} \le P^*_{C^{(k)}}$ ))
    /* i is a preferred neighbor with least probability and at least has two non-faulty neighbors */
    then OPTIMAL_ROUTING
        Send (M, C(i), D);
        M.Routing_Distance = M.Routing_Distance+1;

    elseif ∃ j (N(j) = 0 ∧ $P^*_{C^{(j)}}$ <1 ∧ (∀ k | N(k) = 0. $P^*_{C^{(j)}} \le P^*_{C^{(k)}}$ ))
    /* j is a spare neighbor with least probability and at least has two non-faulty neighbors */
    then SUBOPTIMAL_ROUTING
        Send (M, C(j), D);
        M.Routing_Distance = M.Routing_Distance+1;
    else
        Failure_Message(M);
    else
        Loop_Message(M);
end
```

Based on this routing algorithm, suboptimal routing increases the routing distance by 2 over the Hamming distance of source and destination nodes. Therefore, in the worse case, if the routing is feasible the length of resulted path is at most $k+2f$ hops between two nodes with a Hamming distance of k where f is number of faulty nodes in the n-cube. This is because of the condition used for detecting loops in the above algorithm. To find maximum adaptivity, we need a simple but effective way of gathering information about faulty nodes in the neighborhoods. The probability used in this paper has more adaptivity than the method proposed in [21].

THEOREM 1. *In the routing of a message m from source node to destination node that is d hope away, let U_d^i be the node that message m has been reached to it after traversing i hops. If for every i the condition $P^*_{U_d^i} <(d-i)/(n-1)$ is true, then the algorithm routes the message using an optimal path.*

Proof: *If $P^*_{U_d^i} <(d-i)/(n-1)$ then using Eq. 1, it can be implied that $P_{U_d^i} <(d-i)/n$. This means that node U_d^i has at most (d-i)-1 faulty neighbors. Since the node U_d^i is i hops away from the source node (which is d hops away from the destination node), if i hops were traversed in preferred dimensions, then in node U_d^i, the message should be sent via one of the remaining d-i dimensions and, therefore, at least one preferred neighbor exists such that, according to the neighbor-based algorithm node, U_d^i can send the message to it.*

5 Performance comparison

We have analyzed experimentally the performance of the proposed neighbor-based fault tolerant approach using simulation experiments. A simulation study has been conducted for both the proposed neighbor-based routing and the probability vector algorithm [21]. The network cycle time is defined as the transmission time of a single flit across a physical channel. Messages are generated at each node with rate of one message to all nodes per cycles. Message length is fixed at M flits. The mean latency is defined as the mean amount of time from the generation of a message until the last data flit is absorbed at the destination node. For the sake of specific illustration, we provide results for the following cases:

- Network size $N = 2^{10}$ nodes
- Message length $M = 64$ flits
- Number of faulty nodes was started from 0% and increase up to 75% of network size

We assume that each number traverses a link in one channel cycle time. A total of 100,000 source-destination pairs were selected randomly during each run. The following variables are used to define some performance measures.

- Unreachibility: a routing for non-faulty source and destination nodes that the algorithm could not route a message between them.
- Looping: routing of a message which crosses a number of links beyond the maximum threshold before being discarded.
- Deviation time: Time difference between message latency using the proposed algorithm and the message latency using optimal routing.

The percentage of unreachability measures the percentage of message that the algorithm fails to deliver to destination due to faulty component. The percentage of looping indicates the ratio of messages that fail to reach destinations due to network looping. Figures 1, 2, 3 and 4 show the results of comparison between the probability vector and neighbor-based algorithms. Figure 1 shows the percent of unreachability with respect to the number of faulty component. It reveals that the failure rate of delivery in neighbor-based algorithm is equal to the probability vector approach. Figure 2 shows that neighbor-based routing is very similar to the probability vector when number of faults is less than 400. When number of faults further increases, there are some differences in the looping messages. However in Figure 3, it has been showed that the one proposed algorithm outperforms the probability vector scheme in the average time for delivering messages when the network has large number of faulty nodes. It reduces the average time up to 20 percent in the presence of faulty components, compared to the probability vector routing.

The calculation time of probability vector and the probability used in this scheme for various faulty nodes has been compared and shown in Figure 4. In this Figure, the parameter Pf denotes the percent of faulty nodes in the network. As shown in this figure, the time of calculating probability vector increases exponentially with respect to network dimensions (i.e., the order of calculation is $O(2^n)$) while the time needed for calculating the probability used in our scheme has taken only few cycles (i.e., the order of calculation is $O(1)$).

In the case of dynamic faults, since the number and location of faults move dynamically during the run time, it is necessary the information required for fault tolerant routings was updated. Figure 5 shows the average throughput of the network in term of the time between probability recalculations (i.e., time to update information) when the percent of faulty nodes is 85%. *Throughput* is the maximum amount of message delivered per time unit in nodes (cycle) [22]. As shown in the Figure 5, the proposed algorithm outperforms the probability vector algorithm, particularly when the time between probability recalculations needs to be not large due to tolerate more dynamic faults. It is due to the facts that 1) probability vector algorithm calculates probability vectors in a time higher than the neighbor-based and therefore causes to communicate less messages, and 2) neighbor-based algorithm routs message more optimal than probability vector as shown in the Figure 3 that causes messages to be received early in destination.

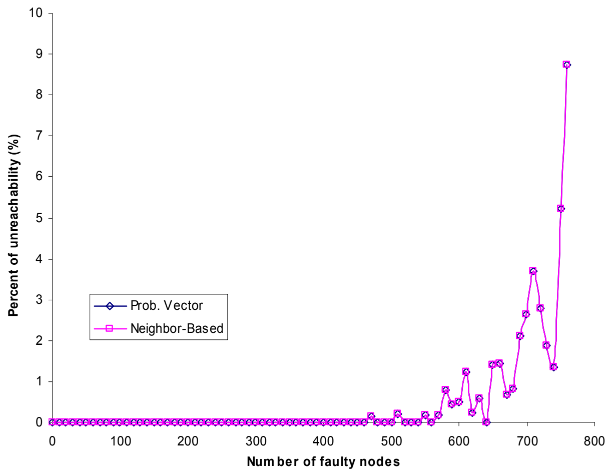

Figure 1. Percent of unreachability of Prob. Vector and Neighbor-based algorithms

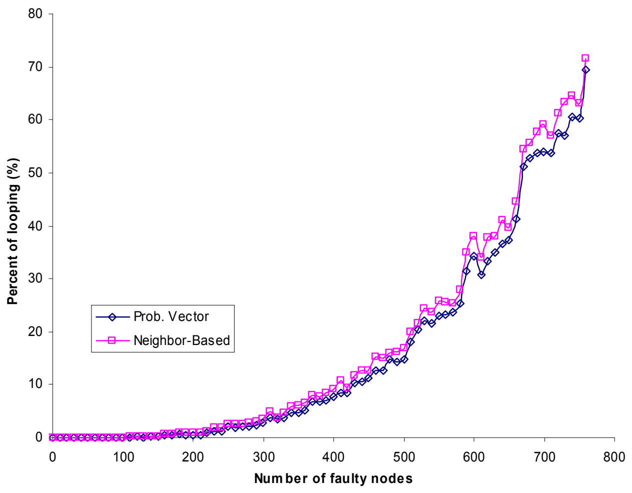

Figure 2. Percent of looping of Prob. Vector and Neighbor-based algorithms

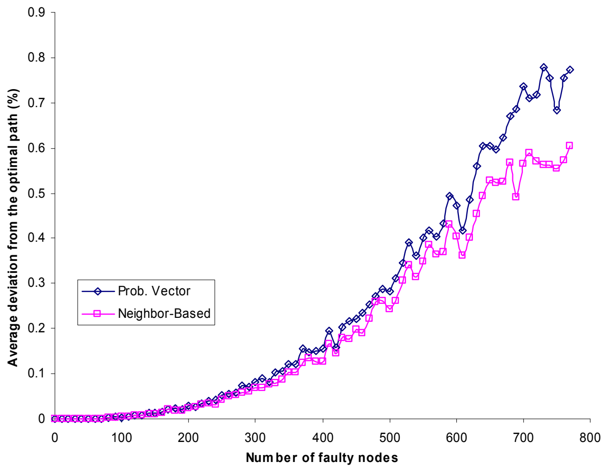

Figure 3. Average deviation from the minimal routing path of Prob. Vector and Neighbor-based algorithms in the case of wormhole routing

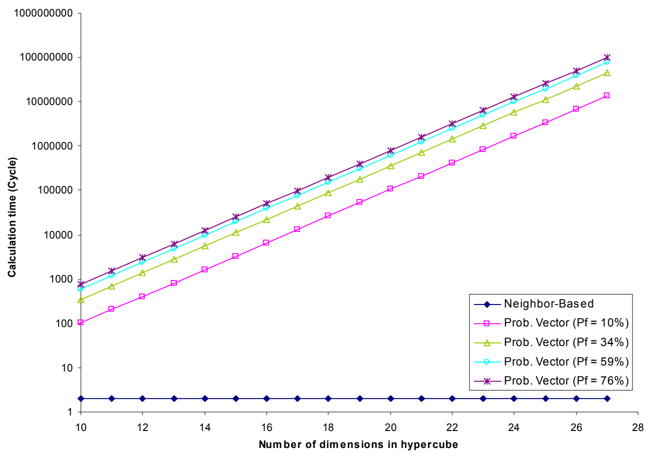

Figure 4. Calculation time of the probability required in Prob. Vector and Neighbor-based algorithms

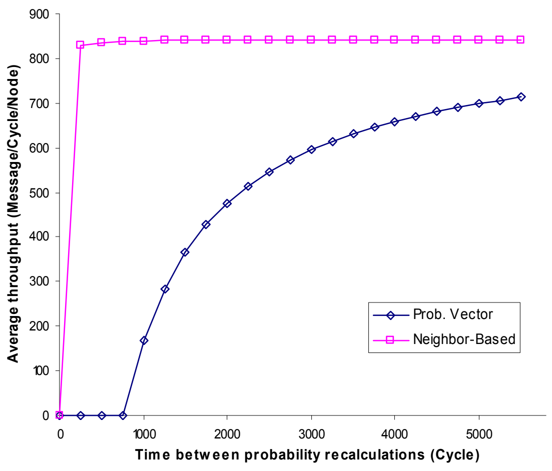

Figure 5. Average throughput of Prob. Vector and Neighbor-based algorithms vs. time between probability recalculations

Although probability vector eventually approaches to the throughput of the neighbor-based, but it is suitable for tolerating static faults rather dynamic faults. Hence in order to tolerate dynamic faults, simulation results reveals that the neighbor-based algorithms is more efficient than the probability vector and throughput of message delivery with wormhole switching is considerably improved.

6 Conclusion

This paper proposed a new fault tolerant routing which routes messages based on the local-information of faulty nodes. Due to the low communication overhead needed for updating the local fault-information of nodes, this scheme tolerates dynamic faults very efficiently. According to the similar works such as probability vector [21], this scheme has message latency lower than the previous published works. Simulation results revealed that, in the case of static faults, the latency of messages was reduced up to 20 percent and in the presence of dynamic faults, the throughput of message delivery with wormhole switching was considerably improved.

References

[1] S.F Nugent, "The iPSC/2 Direct –Connect Communication Technology," *Proc. Conf. Hypercube Concurrent Computers and Applications,* vol. 1, pp. 51-60, 1988.

[2] N-Cube Company, *NCUBE-2 Processor Manual*. 1990.

[3] J. Laudon and D. Lenoski, "The SGI Origin:A ccNUMA Highly Scalable Server," *Proc. ACM/IEEE 24th Int'l Symp. Computer Architecture,* pp. 241-251, 1997.

[4] P.T. Gaughan and S. Yalmanchini, "A Fault-Tolerant Routing Sterategy in Hypercube Multicomputers." *IEEE Trans. Computers,* vol. 45, no. 2, pp. 143-156, Feb. 1996.

[5] L.M. Ni and P.K. McKinley, "A Survey of Routing Techniques in Wormhole Networks," *Computer,* vol. 26, no. 2, pp. 62-76, Feb. 1993.

[6] Y. Saad and M.H. Schultz, "Data Communication in Hypercubes," *Technical Report* YALEU/DCS/RR-428, Dept. of Computer Science, Yale Univ., June 1985.

[7] H. Sullivan, T. Bashkow, and D. Kalppholz, "A Large Scale, Homogenouse, Fully Distributed Parallel Machine," *Proc. Fourth Ann. Symp. Computer Architecutre,* pp. 350-361, May 1977.

[8] R. V. Boppana, and S. Chalsani, "Fault-Tolerant Wormhole Routing Algorithms for Meshes Networks," *IEEE Trans. Computers,* vol. 44, no. 7, pp. 848-868, 1995.

[9] J. Al-Sadi, K. Day, and M. Ould-Khaoua, "Fault-Tolerant Routing in Hypercubes Using Probability Vectos," *Parallel Computing,* vol. 27., pp. 1381-1399, 2001.

[10] T.C. Lee, and J.P. Hayes, "A Fault-Tolerant Communication Scheme for Hypercube Computers," *IEEE Trans. Computers,* vol. 41, pp 1242-1256, 1992.

[11] G. M. Chiu, and S. P. Wu, "A Fault-Tolerant Routing Strategy in Hypercube Multicomputres," *IEEE Trans. Computers,* vol 45, 143-156, 1996.

[12] J. Wu, and E.B. Fernandez, "Broadcasting in faulty hypercubes," *in Proc. 11th Symp. Reliable Distrib. Sys.,* PP. 122-129, 1992.

[13] C. Su, and K. G. Shin, "Adaptive Fault-Tolerant Deadlock-Free Routing in Meshes and Hypercubes", *IEEE transaction on computers*, Vol. 45, No. 6, June 1996, pp 666-683.

[14] J.M. Gordon, and Q.F. Stout, "Hypercube Message Routing in the Presence of Faults," *In Proc. 3rd Comf. Hypercube Concurrent Computer and Applications, pp. 83-8, 1988.*

[15] M.S. Chen, and K. G. Shin, "Depth-First Search Approach for Fault-Tolerant Routing in Hypercube Multicomputers," *IEEE Trans. Par. Distrb. Sys.,* vol 1, pp 152-159, 1990.

[16] M. S. Chen, and K. G. Shin, "Adaptive Fault-Tolerant Routing in Hypercube Multicomputers," *IEEE Trans. Computers,* vol. 39, pp. 1406-1416, 1990.

[17] Y. Lan, "A Fault-Tolerant Routing Algorithm in Hypercubes," *Proc. 1994 Int. Conf. Par. Process,* vol II, pp. 163-166, 1997.

[18] J. Wu, "Adaptive Fault-Tolerant Routing in Cube-Based Multicomputers Using Safety Vectors," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 4, April 1998.

[19] G.M. Chiu, and K. S. Chen, "Use of Routing Capability for Fault-Tolerant Routing in Hypercubes Multicomputers," *IEEE Trans. Computers*, vol. 46, pp. 953-958.

[20] J. Wu, "Adaptive Fault-Tolerant Routing in Cube-based Multicomputers Using Savety Vectors," *IEEE Trans. Par. Distrib. Sys.,* vol. 9, pp. 321-334, 1998.

[21] J. Al-Sadi, K. Day, and M. Ould-Khaoua, "Probability-based Fault-Tolerant Routing in Hypercubes," *The Computer Journal,* vol. 44., no. 5, 2001.

[22] J. Duato, "*Interconnection Networks: An Engineering Approach*," IEEE Computer Society, 1997.