International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:2, 2015

Fault-Tolerant Optimal Broadcast Algorithm for the Hypercube Topology

Lokendra Singh Umrao, Ravi Shankar Singh

Abstract—This paper presents an optimal broadcast algorithm for the hypercube networks. The main focus of the paper is the effectiveness of the algorithm in the presence of many node faults. For the optimal solution, our algorithm builds with spanning tree connecting the all nodes of the networks, through which messages are propagated from source node to remaining nodes. At any given time, maximum $n-1$ nodes may fail due to crashing. We show that the hypercube networks are strongly fault-tolerant. Simulation results analyze to accomplish algorithm characteristics under many node faults. We have compared our simulation results between our proposed method and the Fu's method. Fu's approach cannot tolerate $n-1$ faulty nodes in the worst case, but our approach can tolerate $n-1$ faulty nodes.

Keywords—Fault tolerance, hypercube, broadcasting, link/node faults, routing.

I. Introduction

EFFICIENT broadcast of information is a critical issue and affect the performance of a multicomputers. Broadcast is an important service for transmitting information from one node, called the source node, to all other nodes in a system and used to implement many parallel and distributed algorithms [1]. However, if a component (nodes or links) fails before the transmission completes, hours of work may be wasted. So, an optimal fault-tolerant broadcast algorithm is needed for faulty components.

The n-dimensional hypercube topology [2] is a most popular architecture for parallel computing because of high symmetry, strong hierarchical structure, maximal fault-tolerance, and low diameter. Existence of alternate paths between any pair of nodes provides the maximal fault-tolerance and small diameter (which is the maximum shortest path between any two nodes) for efficient communication. For a hypercube network the diameter is identical to the degree of a node $n = log_2 N$. The binomial tree is used to implement optimal broadcasting in hypercubes. Optimal broadcasting means the broadcast information reaches its destination through a shortest path. The binomial-tree-based broadcasting guarantees that the broadcast information is propagated to each destination once and only once through a shortest path [3].

In this paper a fault-tolerant optimal broadcast algorithm is proposed. This algorithm uses spanning tree rooted at any node. Both nodes and links may be faulty. However, we will consider only node faults, link faults can be tolerated by assuming that there is an alternative path from one node to another node. It will be assumed that faulty nodes can not perform any computation. The terms node and process are used interchangeably. Processes can fail only by crashing and a crash is permanent. A process is correct if it sends the information from one node to another correctly during the execution, otherwise, it is faulty.

An n-dimensional hypercube consists of 2^n nodes, which are labeled $0, 1, \ldots, 2^{n-1}$; two nodes are adjacent if their labels differ in exactly one bit position. This property highly facilitates the routing of messages through the network. In addition, the regular and symmetric nature of the network provides fault tolerance. The hypercube has been widely used as the interconnection network in a variety of parallel systems such as nCUBE-4, Intel iPSC/860 [4], Connection Machine CM-5 [5], and SGI Origin 2000 [6].

The paper is organized as follows: Section II reviews the concepts of hypercubes, binomial-tree-based broadcasting, and fault-tolerance. In Section III we describe some previous work and their related results. In Section IV we present the routing scheme in the presents of faults. Section V presents the optimal fault-tolerant broadcasting algorithm. Simulation results are discussed in Section VI. Finally, Section VII concludes this paper.

II. Preliminaries

For an n-dimensional hypercube, H_n, each node has n links. Since there are 2^n nodes in the system, so there are a total of $n.2^{n-1}$ links. Every node have the address of form $(a_{n-1}, a_{n-2}, \ldots, a_0)$, where $a_i \in \{0,\ 1\}$ for $i = 0, 1, 2, \ldots, n-1$. The i^{th} bit is denoted to i^{th} dimension or dimension i. The proportional address of two nodes, a and b, is the bitwise exclusive-or operation of the addresses of these two nodes, i.e., $a \oplus b = c = (c_{n-1}, c_{n-2}, \ldots, c_0)$, where $c_i = a_i \oplus b_i$ for $i = 0, 1, 2, \ldots, n-1$. If a link connecting two nodes a and b in dimension i, and link denoted by $(a_{n-1}, a_{n-2}, \ldots, a_{i+1}, -, a_{i-1}, \ldots, a_0)$, where the address of node a is $(a_{n-1}, a_{n-2}, \ldots, a_0)$. For example, the nodes 01000 and 01010 are connected by link 010-0 in dimension 1. The relative address of two links l and m is also bitwise exclusive-or of their addresses, i.e., $l_i \oplus m_i = 1$.

For the construction of spanning tree of the hypercube, the binomial spanning tree is well-known spanning tree. The binomial tree [7] B_n in a n-dimensional hypercube is an ordered tree defined recursively. A tree consisting of a single node is called a B_0 binomial tree. The binomial tree B_n consists of two binomial trees B_{n-1} that are linked together: the root of one is the leftmost child of the root of the other.

L. S. Umrao, and R. S. Singh are with the Department of Computer Science & Engineering, Indian Institute of Technology (BHU), Varanasi-221 005, India. E-mail: {lokendra.rs.cse12,ravi.cse}@iitbhu.ac.in.

Normally an incomplete binomial tree [8] is used to implement an optimal broadcast process. A spanning incomplete binomial tree B_n in an offended hypercube is a connected subtree with the same root node that connects all the non-faulty nodes in the hypercube. If a H_n is divided along k dimensions $(d_1, d_2, \ldots, d_k)$, then the size of subcubes will be H_{n-k}.

Errors in any system are caused by faults and those errors ultimately result in failure. Fault is nothing but a defect within the system where error is detected by a deviation from the expected behaviour of the system. When the system can no longer perform as desired output, failure is said to occurred. To provide a service even with errors in a system, an ability of fault-tolerance is required [9].

There are two distinct feature of fault-tolerance in a network: tolerance in links and nodes faults. Link faults may be tolerated because there are redundant paths in the hypercube topology. Hamming distance is used for shortest path from source to destination in the network and given by the number of bits position changed.

III. Related work

Blough et al. [10] developed a routing algorithm using a directed depth-first search technique. The algorithm routes messages provided there exists at least one non-faulty path between sender and receiver. The algorithm takes the shortest path when there are no faulty elements in the cube. Paper [11] considers the problem of fault-tolerant routing in multiprocessor systems when incomplete, or partial, diagnostic information is available.

Several fault-tolerant algorithms for hypercube topology for routing and broadcasting have been proposed in [12], [13], [14], [15], [16], [17], [18]. Actually we are interested in broadcasting information from any node to other nodes non-redundantly. Paper [13] describes routing scheme based on local safety information is proposed and the extra cost to obtain local safety information is comparable to the one based on global safety information. The proposed algorithm guarantees to find a minimum feasible path if the spanning subcube is contained in a maximal safe subcube and the source is locally safe in the maximal safe subcube.

Chen et al. [19] proposed hypercube networks with a very large number of faulty nodes. The algorithms are distributed and local-information-based in the sense that each node in the network knows only its neighbors' status and no global information of the network is required by the algorithms. For a locally subcube-connected hypercube network that may contain up to 37.5% faulty nodes, this algorithm runs in linear time and, for any two given non-faulty nodes and finds a routing path of length bounded by four times the Hamming distance between the two nodes.

Huang et al. [20] present three kinds of broadcasting tree for the even dimensional uni-directional hypercube (UHC) and some applications (ASCEND/DESCEND algorithms and bitonic sorting). They propose an all-port fault-tolerant broadcasting tree (a family of arc-disjoint spanning trees) whose height is no more than $\frac{3}{2}n + 1$.

Paper [21] presents a method to cope with reliable broadcasting in faulty hypercubes using local safety information. Local safety information is well used in the fault-tolerant broadcast algorithm by considering only safety of the broadcast subcube.

Paper [22] developed new techniques that enable to analyze a more realistic fault tolerance model and derive lower bounds for the probability of hypercube network fault tolerance in terms of node failure probability. Results are both theoretically significant and practically important. From the theoretical point of view, this method offers very general and powerful techniques for formally proving lower bounds on the probability of network connectivity, while from the practical point of view, the results provide formally proven and precisely given upper bounds on node failure probabilities for manufacturers to achieve a desired probability for network connectivity.

Chen and Yu-Wei [23] proposed one-to-all broadcasting algorithms that can tolerate $\lfloor 3n/2 \rfloor - 1$ faulty nodes. The sequence of dimensions used for broadcasting in each algorithm is the same regardless of which node is the source. The fault-tolerance improvement of this paper is about 50%.

Paper [24] proved that a fault-free cycle of length at least $2^n - f - (n - 5)$ can be embedded in an n-dimensional restricted hypercube-like network with f faulty nodes, where $n \geq 5$ and $f \leq 2n - 7$. They show that cycle-structured parallel algorithms can be efficiently executed in restricted hypercube-like networks with faulty nodes.

IV. Routing in the hypercube

In a faulty hypercube, it is necessary for message delivery to find a path of non-faulty nodes from the source to the destination. For this purpose, each node can store some information about neighbors to select one of them for message sending by using it with the address of the destination. In addition, a detour must be detected even if any shortest path can not be found by using the information. An algorithm to perform these operations is called a routing algorithm. In this situation, a good algorithm finds as many shortest paths as possible while it holds as simple information as possible

If there are no faulty links in a dimension, then it is called fault-free dimension. Suppose H_n is divided into H_{n-1} and H'_{n-1} along d dimension and there exist faulty links l in H_{n-1} and l' in H'_{n-1} that differ only in the d^{th} bit. This type of dimension called unsafe. If there are no faulty link exists, then the dimension d is called safe. For example, suppose there are two faulty links, 00-0 and 00-1, then dimension 0 is unsafe. Remaining dimensions are safe. So, we can say that if a dimension is unsafe, then there exist two faulty links l and l' such that $l \oplus l' = 1$. If a dimension is fault-free and safe, then it is called fault-free safe dimension. Dimension 0 in 0-00 and 00-1 is fault-free safe.

Let the set of faulty links are $F = \{f_1, f_2, \ldots, f_m\}$ and all these faulty links are in the same dimension. In this situation, if there exists two faulty links f_i and f_j such that they differ only in the d^{th} bit and are in same dimension, then dimension d is unsafe. Let, $F = \{00011-, 00010-, 00111-, 00110-\}$, then a list of unsafe dimension is $\{00001-, 00100-\}$. So, dimensions 1 and 3 are unsafe, other dimensions are safe.

This approach is based on the use of spanning trees to configure a given hypercube in order to bypass faulty nodes. The algorithm starts by constructing the spanning tree (ST) of the hypercube. This is followed by a reconfiguration phase which requires two steps. During the first step, a faulty node is removed from the ST. During the second reconfiguration step, a new ST is constructed by reconnecting the children of the removed node.

For example, consider the hypercube shown in Fig. 1. The ST of the hypercube shown in Fig. 1 is shown in Fig. 2.

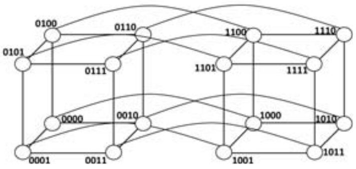

Fig. 1. A 4-dimensional hypercube

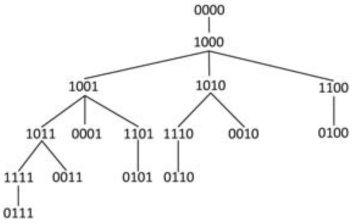

Fig. 2. A Spanning tree (ST) of the hypercube in Fig. 1

Since node (1010) is faulty, then it should be removed from ST and its children be reconnected as shown in Fig. 3. The reconfiguration process defines a new parents and new children in order to circumvent faulty node(s). For example, in Fig. 3 the new children of node (1100) become node (1110) and (0100) while the parent of node (0010) becomes node (0011).

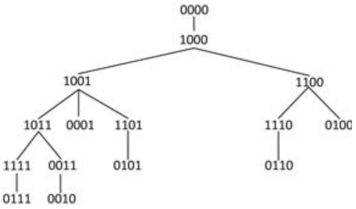

Fig. 3. The modified ST under faulty node

V. Broadcasting in the Hypercube

A spanning tree for an undirected graph $G = (V, E)$ is a subgraph that is a tree containing every vertex of G. Spanning tree provides an efficient method for nodes to communicate in a distributed system by cutting down the links that the information must reach all the processes. The broadcasting of H_n is done at source node with the help of dimension based partitioning of the subcubes. The binomial-tree are built using the coordinate set (CS) of the hypercube network. The coordinate set are based on the sequence of ternary symbols $(t_{n-1}, t_{n-2}, \ldots, t_0), t_i \in \{0, 1, *\}$ for $0 \leq i \leq n-1$, where $*$ is a don't care condition. In Fig. 1, the CS at 0000 is 0123. Therefore, the dimension order is dimension 0, dimension 1, dimension 2, and dimension 3. First, the 4-cube $****$ at 0000 is partitioned into $***0$ and $***1$ along dimension 0. $***0$ is then partitioned into $**00$ and $**10$ along dimension 1. Then $**00$ is partitioned into $*000$ and $*100$ along dimension 2. Finally $*000$ is partitioned into 0000 (root) and 1000. That is, $**** = ***1, **10, *100, 0000, 1000$. B_3, B_2, B_1, and B_0 are incomplete spanning binomial trees of subcubes $***1$, $**10$, $*100$, and 1000, respectively. Fig. 4 shows the broadcast process rooted at node 0000 based on the binomial tree of Fig. 1.

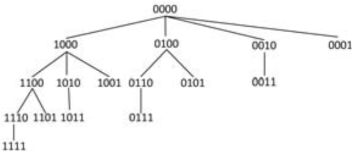

Fig. 4. A spanning binomial tree rooted at 0000

Let n=6, the source node is 110101 and 101111 is the destination node. Let $\{a_1 = 101101, a_2 = 110110, a_3 = 111111\}$ are the faulty nodes, other nodes are non-faulty. a_1 and a_2 are different at 0^{th} bit and 1^{st} bit of a_1 and a_3 are also different. Hence the two bits (0^{th} and 1^{st}) of a_1, a_2, and a_3 are all distinct. If we divide H_6 along the 4^{th} and 5^{th} bits, the subcube will be $\{00****, 01****, 10****, 11****\}$. Any CS will be of the form $**a_3a_2a_1a_0$ where the bits a_3, a_2, a_1, and a_0 are fixed. Because the faulty nodes have distinct bits in the lowest significant bits of their addresses, in this case each CS can have at most one faulty node. If H_n is divided along one more dimension that is not part of the distinguishing bits of the faulty node address bits, each CS still contains at most one faulty node. Any process in hypercube topology can be the root.

We present a new way of construction of optimal spanning tree and show its strong relation with the binomial tree. We have n spanning trees that are disjoint in an n-cube in which every edge consists of two directed links, hence the trees are actually Edge-disjoint. Here we will assume that the n-cube consists of undirected edges.

Before the spanning tree algorithm proposed, we will defined some functions, considering a process i as the root. The first level of the tree is determined by the function $neighborhood_i$. From the first level, each process i can define the subsequent level in the tree of the root j using the function $neighborhood_i(j)$. Using the function $cluster_i(j)$, a process i can get the upper level process in the tree of the root j recursively.

Function $cluster_i(j) = s$ gives the index of the cluster of process i that contains process $j, 1 \leq s \leq d$, where $s = 1, \ldots, log_2 n$ and n is the total number of nodes in a system. For example, considering the 4-cube of Fig. 1, $cluster_8(0) = 4, cluster_4(0) = 3, cluster_2(0) = 2, cluster_1(0) = 1$.

Function $neighbor_i(s) = j$ identifies the first fault-free process j in cluster s of process i. For example, considering Fig. 1 that shows all clusters in a 4-cube, $neighbor_4(1) = 5, neighbor_4(2) = 6, neighbor_4(3) = 0, neighbor_4(4) = 12$.

Function $neighborhood_i = \{j | j \in neighbor_i(s), 1 \leq s \leq d\}$ gives the set of all $neighbor_i(s)$ for $s = 1...log_2 n$. For example, considering all nodes are fault-free in a 4-cube, $neighborhood_0 = \{1, 2, 4, 8\}$. If process 8 alone is faulty, then $neighborhood_0 = \{1, 2, 4, 9\}$.

Function $neighborhood_i(j) = \{k | k = neighbor_i(s), 1 \leq s < cluster_i(j)\}$ is used by process i to identify to which processes a message received from process j should be sent. For example, considering the spanning tree with root at process 0 in Fig. 4, $neighborhood_1(0) = \phi, neighborhood_2(0) = \{3\}, neighborhood_4(0) = \{5, 6\}$ and $neighborhood_8(0) = \{9, 10, 12\}$.

The use of independent spanning trees has scientific applications in fault-tolerant protocols for distributed computing networks. Broadcasting in a hypercube network is sending a message from source node to all the remaining nodes in the network. In this paper, we proposes a fault-tolerant broadcasting algorithm with the help of independent spanning trees. We can achieve fault-tolerance by sending n copies of the message rooted at the source node along n independent spanning trees. If the source node is faultless, our broadcast algorithm can tolerate up to $n - 1$ faulty nodes. Algorithm 1 shows a pseudo-code for the fault-tolerant broadcast at process p_i.

Algorithm 1: Broadcasting procedure

```
Input: an $n$-dimensional hypercube $H_n$ with faulty
       processes and at least one non-faulty process $p_i$ in
       $H_n$
Output: the broadcast paths from $p_i$ in $H_n$ of non-faulty
        processes
1. It is easy to construct a spanning tree taking $p_i$ as the
root node and the same time broadcast its information
through the tree.
2. At the same time, it sends the information to all its
neighbor processes.
if some of its neighbor node has already received the
information then
    discard the information and don't pass it on;
else if there are faulty nodes in its neighbor nodes then
    find the adjacent non-faulty nodes, pass information
    to them and return to step 2.
3. Repeat the algorithm until all the non-faulty nodes has
received this information.
```

VI. Simulation results

We have simulated our work on CPN (Colored Petri nets) tool to validate the performance of the improved algorithm in hypercube topology. Fig. 5 shows the simulator's configuration. Firstly, service requests are sent to the client from the users, then the service requests are transferred to the scheduler. The scheduler receives the service request and decides its sequence according to the schedule rules and dispatches these results to processor and further it decides which one processor to execute the service request. The main purpose of our algorithm is to realize fault-tolerance. Our simulations are based on exponential distribution of node failures, i.e., every node has an continuous and independent failure probability.

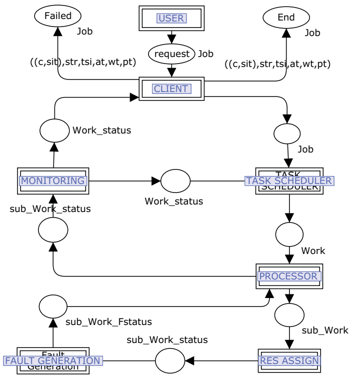

Fig. 5. CPN model for computing

Fig. 6 represents the monitoring of service where the complete service having no fault is sent to client and then to user. If there is any fault in the execution of service, then service is sent to task scheduler from where it is either rescheduled or is aborted.

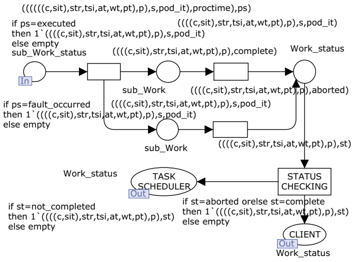
((((((c,sit),str,tsi,at,wt,pt),p),s,pod_it),proctime),ps)
if ps=executed
then 1`((((c,sit),str,tsi,at,wt,pt),p),s,pod_it)
else empty
sub_Work_status
((((c,sit),str,tsi,at,wt,pt),p),s,pod_it)
((((c,sit),str,tsi,at,wt,pt),p),complete)
Work_status
In
sub_Work
((((c,sit),str,tsi,at,wt,pt),p),aborted)
if ps=fault_occurred
then 1`((((c,sit),str,tsi,at,wt,pt),p),s,pod_it)
else empty
((((c,sit),str,tsi,at,wt,pt),p),s,pod_it)
sub_Work
((((c,sit),str,tsi,at,wt,pt),p),st)
Work_status
TASK SCHEDULER
Out
STATUS CHECKING
if st=not_completed
then 1`((((c,sit),str,tsi,at,wt,pt),p),st)
else empty
if st=aborted orelse st=complete
then 1`((((c,sit),str,tsi,at,wt,pt),p),st)
else empty
CLIENT
Out
Work_status

Fig. 6. CPN model for monitoring

We have compared our simulation results between the improved method and the Fu's method [25]. Fu's approach cannot tolerate $n - 1$ faulty nodes in the worst case, but our approach can tolerate $n - 1$ faulty nodes. Hence, our result is optimal. The simulation results are given in Table I. The simulation results show that the proposed method provides the better results in comparison of Fu's method.

VII. Conclusion

This paper introduced an optimal broadcast algorithm for the hypercube networks using spanning trees. For the optimal solution, our algorithm builds with spanning tree connecting all the nodes, through which messages are propagated from source node to the remaining nodes. Simulation results were analyzed to accomplish algorithm characteristics under many node faults. We have compared our simulation results between our proposed method and the Fu's method. Fu's approach

TABLE I
SIMULATION RESULTS ON THE 8-CUBE

No. of faults	Fu's method (%)	Our method (%)
0	100.0	100.0
1	99.8	100.0
2	99.6	99.8
3	97.2	98.4
4	92.4	95.2
5	87.4	91.6
6	76.8	83.4
7	n/a	72.8

cannot tolerate $n-1$ faulty nodes in the worst case, but our approach can tolerate $n-1$ faulty nodes. As future work, other broadcast algorithms will be developed.

REFERENCES

[1] Lin, Jeng-Wei, "Broadcast scheduling for a p2p spanning tree", IEEE International Conference on Communications. ICC'08, pp. 5614–5618, 2008.

[2] Saad, Youcef and Schultz, Martin H, "Topological properties of hypercubes", IEEE Transactions on Computers, vol. 37, no. 7, pp. 867–872, 1988.

[3] Figueira, Silvia M and Mendes, Christine, "Dynamically adaptive binomial trees for broadcasting in heterogeneous networks of workstations", High Performance Computing for Computational Science-VECPAR 2004, pp. 480–495, 2005.

[4] Dunigan, Thomas H., "Performance of the Intel iPSC/860 and Ncube 6400 hypercubes", Parallel Computing, vol. 17, no. 10, pp. 1285–1302, 1991.

[5] Palmer, John and Steele Jr, Guy L, "Connection Machine model CM-5 system overview", Fourth Symposium on the Frontiers of Massively Parallel Computation., pp. 474–483, 1992.

[6] Whitney, Steve and McCalpin, John and Bitar, Nawaf and Richardson, John L and Stevens, Luis, "The SGI Origin software environment and application performance", IEEE Proceedings, Compcon'97., pp. 165–170, 1997.

[7] Lee, Tze Chiang and Hayes, John P., "A fault-tolerant communication scheme for hypercube computers"", IEEE Transactions on Computers, vol. 41, no. 10, pp. 1242–1256, 1992,

[8] Wu, Jie and Fernandez, Eduardo B, "Broadcasting in faulty hypercubes", Microprocessing and microprogramming, vol. 39, no. 1, pp. 43–53, 1993.

[9] Lee, Peter Alan and Anderson, Thomas, "Fault tolerance", 1990.

[10] Blough, Douglas M and Bagherzadeh, Nader, "Near-optimal message routing and broadcasting in faulty hypercubes", *International Journal of Parallel Programming*, vol. 19, no. 5, pp. 405–423, 1990.

[11] Blough, Douglas M and Wang, HY, "Cooperative diagnosis and routing in fault-tolerant multiprocessor systems", *Journal of Parallel and Distributed Computing*, vol. 27, no. 2, pp. 205–211, 1995.

[12] Krull, Jace W and Wu, Jie and Molina, Andres M, "Evaluation of a fault tolerant distributed broadcast algorithm in hypercube multicomputers", *Proceedings of the 1992 ACM annual conference on Communications*, pp. 459–466, 1992.

[13] Xiang, Dong, "Fault-tolerant routing in hypercube multicomputers using local safety information", *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 9, pp. 942–951, 2001.

[14] Xiang, Dong and Chen, Ai, "Partial path set-up for fault-tolerant routing in hypercubes", *International Proceedings on Parallel and Distributed Processing Symposium*, pp. 8–18, 2003.

[15] Xiang, Dong and Chen, Ai and Wu, Jie, "Local-safety-information-based fault-tolerant broadcasting in hypercubes", *J. Inf. Sci. Eng.*, vol. 19, no. 3, pp. 467–478, 2003.

[16] Liu, Fangai and Song, Ying, "Broadcast in the locally k-subcube-connected hypercube networks with faulty tolerance", *Networking and Mobile Computing*, pp. 305–313, 2005.

[17] Xiang, Dong, "Fault-tolerant routing in hypercubes using partial path set-up", *Future Generation Computer Systems*, vol. 22, no. 7, pp. 812–819, 2006.

[18] Jiang, Zhen and Wu, Jie and Wang, Dajin, "A new fault-information model for adaptive & minimal routing in 3-D meshes", *IEEE Transactions on Reliability*, vol. 57, no. 1, pp. 149–162, 2008.

[19] Chen, Jianer and Wang, Guojun and Chen, Songqiao, "Locally subcube-connected hypercube networks: Theoretical analysis and experimental results", *Computers, IEEE Transactions on*, vol. 51, no. 5, pp. 530–540, 2002.

[20] Huang, Huang-Ming and Yang, Chang-Biau and Tseng, Kuo-Tsung and others, "Broadcasting on uni-directional hypercubes and its applications", *J. Inf. Sci. Eng.*, vol. 19, no. 2, pp. 183–203, 2003.

[21] Xiang, Dong and Chen, Ai and Wu, Jie, "Reliable broadcasting in wormhole-routed hypercube-connected networks using local safety information", *IEEE Transactions on Reliability*, vol. 52, no. 2, pp. 245–256, 2003.

[22] Chen, Jianer and Kanj, Iyad A and Wang, Guojun, "Hypercube network fault tolerance: A probabilistic approach", *Journal of Interconnection Networks*, vol. 6, no. 01, pp. 17–34, 2005.

[23] Chen, Yu-Wei, "Improved one-to-all broadcasting algorithms on faulty SIMD hypercubes", *Journal of Parallel and Distributed Computing*, vol. 65, no. 12, pp. 1596–1600, 2005.

[24] Dong, Qiang and Yang, Xiao-Fan, "Fault-Tolerant Cycle Embedding in Restricted Hypercube-like Networks with More Faulty Nodes", *Journal of Information Science and Engineering*, vol. 28, pp. 419–426, 2012.

[25] Fu, Jung-Sheng, "Longest fault-free paths in hypercubes with vertex faults", Information Sciences, vol. 176, no. 7, pp. 759–771, 2006.