One important performance measure of CED designs is how quickly the error can be detected. The input checksum of a set of input data can be computed while they propagate through the network. The time for calculating the output checksum, however, should be separately included between the initiations of two consecutive FFT computations. Consequently, the output checksum needs to be quickly computed. As shown in Fig. 1, the output checksum of our design requires only two levels of adders, while other designs [2], [3], [4], [5], [6] require adders and multipliers. In [6], the output checksum is calculated by one level of multiplication plus two levels of addition. Therefore, the design requires much larger time than our design for calculating the output checksum. If a typical floating point hardware is used where the circuits for addition and multiplication have the same complexity, multiplication takes w times longer than addition for w-bit data.

V. Conclusion

In this brief contribution, we have presented a new efficient concurrent error detection design for FFT networks. In the proposed design, simple linear weight factors were employed to obtain the input and output checksum for detecting an error in the network. As a result, any single error can be effectively detected with relatively small hardware overhead. No time overhead was required since the original input as well as the network were not modified for the CED operation. It has been shown that our design allows high fault coverage with low false alarm rate for the practical size FFT networks. The proposed design also allows high throughput and short error detection latency. The hardware overhead ratio is as small as $\frac{1.2}{\log_2 N}$ or $\frac{2}{\log_2 N}$ when the input data are real or complex, respectively. The proposed design can also be easily expanded for multidimensional FFT networks. We currently investigate the schemes tolerating multiple faults. [11]

ACKNOWLEGMENTS

This work was supported in part by National Science Foundation Grant MIP-9196165 and Texas State Government Grant ATP 003656-075.

References

[1] C.D. Thompson, "Fourier transforms in VLSI," *IEEE Trans. Computers*, vol. 32, pp. 1,047-1,057, Nov. 1983.
[2] J.Y. Jou and J.A. Abraham, "Fault tolerant FFT networks," *IEEE Trans. Computers*, vol. 37, pp. 548-562, May 1988.
[3] A.L. Narasimha Reddy and P. Banerjee, "Algorithm-based fault detection for signal processing applications," *IEEE Trans. Computers*, vol. 39, pp. 1,304-1,308, Oct. 1990.
[4] D.L. Tao, C.R.P. Hartmann, and Y.S. Chen, "A novel concurrent error detection scheme for FFT networks," *Proc. IEEE FTCS 90*, Univ. of North Carolina at Chapel Hill, pp. 114-121, 1990.
[5] D.L. Tao and C.R.P. Hartmann, "A novel concurrent error detection scheme for FFT networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, pp. 198-221, Feb. 1993.
[6] S. Wang and N.K. Jha, "Algorithm-based fault-tolerance for FFT networks," *Proc. IEEE Int'l Symp. Circuits and Systems*, pp. 141-144, 1992.
[7] Y. Choi and M. Malek, "A fault-tolerant FFT processor," *IEEE Trans. Computers*, vol. 37, pp. 617-621, May 1988.
[8] R.M. Mersereau and T.C. Speake, "A unified treatment of Cooley-Tukey algorithms for the evaluation of the multidimensional DFT," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. 29, pp. 1,011-1,018, Oct. 1981.
[9] P.D. Welch, "A fixed-point fast Fourier transform error analysis," *IEEE Trans. Audio Electroacoustics*, vol. 17, pp. 151-157, June 1969.
[10] A. Roy-Chowdhury and P. Banerjee, "Tolerance determination for algorithm-based checks using simplified error analysis techniques," *Proc. IEEE FTCS 93*, Toulouse, France, pp. 290-298, 1993.
[11] C.G. Oh and H.Y. Youn, "On concurrent error location and correction of FFT networks," *IEEE Trans. VLSI Systems*, vol. 2, pp. 257-260, June 1994.

A Limited-Global-Information-Based Multicasting Scheme for Faulty Hypercubes

Jie Wu, *Senior Member, IEEE*, and Kejun Yao

***Abstract*—We study the multicast problem in hypercubes with faulty nodes. We assume that each node has limited global information of fault distribution, which is captured by the safety level associated with the node. Basically, the safety level is an approximate measure of the distribution of faulty nodes in the neighborhood. We first propose a safety-level-based multicasting scheme which guarantees time optimality. We then present a variant of this scheme that reduces the number of traffic steps. Finally, an *address-sum-based multicasting* scheme is studied which utilizes both the distribution of destination nodes and the safety level of neighboring nodes. Simulation results show that the traffic generated in both schemes is very close to optimal. Time optimality is guaranteed when the source is safe and at most two additional *time steps* are required when the source is unsafe in an *n*-dimensional cube (denoted as *n*-cube) with up to *n* − 1 node faults.**

***Index Terms*—Fault tolerance, hypercubes, multicasting, parallel processing.**

I. Introduction

Message passing techniques are commonly used in hypercubes [8] for interprocessor communication. Usually, a message must go through several intermediate nodes to reach its destination. Thus, the routing scheme used [7] becomes very important for the performance of interprocessor communication. There are three types of communications: *one-to-one*, *one-to-many*, and *one-to-all* [2]. In many parallel applications, communication between a subtask in one node and several subtasks, located in different nodes, often arises. This motivates the introduction of a *multicast* (one-to-many communication) facility that sends messages from one node to a subcube or several subcubes (a node is a trivial case of a subcube).

The performance and reliability of communication schemes are crucial to the success of massive distributed-memory multicomputers. *Time steps* and *traffic steps* [2] are the main criteria used to measure the performance of communication at the system level. The maximum number of links the message traverses to reach one of the destinations is defined as time steps, and the total number of distinct links the message traverses to reach all destinations is measured in traffic steps. All the nodes and links used in a multicast form a *multicast tree*.

It has been shown [5] that the problem of finding a time and traffic optimal solution for multicasting in hypercubes is NP-hard. A heuristic multicast algorithm proposed by Lan, Esfahanian, and Ni [2] achieves time optimality and traffic suboptimality in a nonfaulty hypercube. Several fault-tolerant multicasting schemes have been proposed which can be classified by the amount of network information of fault distribution used at each node. In local-information-based

Manuscript received Sept. 24, 1993; revised June 20, 1994.
The authors are with the Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431; e-mail: jie@cse.fau.edu.
IEEECS Log Number C95096.

multicasting each node knows only the status of its adjacent links and nodes. Simplicity is the main advantage of this scheme, although a large number of additional time steps may be needed in the worst case. A time-optimal multicasting has been proposed [1] which can be categorized as local-information-based multicasting; however, it is based on a restricted fault model—each node in the cube has at most one faulty neighbor. Global-information-based multicasting [9] assumes that each node knows the distribution of faults in the network. This scheme guarantees time optimality. However, it requires a complex process that collects global information. Limited-global-information-based multicasting is a compromise between these two schemes. An algorithm of this type obtains an optimal or suboptimal solution and requires a relatively cheap process that collects and maintains limited global information. Liang, Bhattacharya, and Tsai [3] recently proposed a multicasting scheme in which each node knows the status of all the links within two hops (a particular form of limited global information). This scheme can tolerate up to $n-1$ faulty links and the number of additional time steps required in the worst case is $2n$ in an n-cube. Although this scheme can be easily extended to tolerate node failures, it has two flaws: first, it cannot guarantee time optimality; second, it has to maintain at each node a relatively large table which contains the status of components within two hops from the node. In general, limited global information should be defined in such a way that it is easy to obtain and maintain by compressing information in a concise format.

In this paper, we proposed a multicasting scheme based on limited global information. The limited global information is captured by the *safety level* [10], which is an integer associated with each node. Basically, the safety level associated with a node is an approximate measure of the location and number of faulty nodes in the neighborhood, rather than just the number of faulty nodes. An efficient $(n-1)$-round iterative algorithm has been proposed in [10] that calculates the safety level of each node. The proposed multicasting scheme uses safety level information and/or distribution of destination nodes. It can tolerate $n-1$ faulty nodes in an n-cube while still guaranteeing time optimality and traffic suboptimality. Several variations are also studied and presented.

II. Preliminaries

An n-cube, Q_n, is a graph having 2^n nodes. Every node u has a binary address $u(n)u(n-1)\cdots u(1)$ and $u(i)$ is called the ith dimension of the address. For convenience, we use u to represent a node as well as its associated address. We denote node $u^{(d)}$ the neighbor of u along dimension d, and therefore address $u^{(d)}$ can be obtained by changing the dth bit of u. A *Hamming distance path* between two nodes is a path that has a length equal to the Hamming distance between these two nodes. Let $G(V, E)$ be a graph that corresponds to a hypercube structure [1]. $D = \{u_1, u_2, ..., u_m\} \subseteq V(G)$ is a *destination set*. The multicasting problem can be described as finding a subtree $T(V, E)$ of $G(V, E)$, such that 1) $D \subseteq V(T)$, 2) $d_T(s, u_i) = d_G(s, u_i)$, where s is the source node and d_T and d_G are distances in graphs T and G, respectively, and 3) $|E(T)|$ is as small as possible. Our objective here is to find a multicasting scheme that guarantees time optimality and traffic suboptimality.

In our approach the concept of limited global information is captured by assigning a safety level to each node in a hypercube with node faults. More specifically, let k be the safety status of node u, where k ranges from 0 to n. k is referred to as the *level of safety*, and u is called *k–safe*. An n-safe node is called a *safe node* and a k-safe ($k \neq n$) node is called an *unsafe node*. A faulty node is 0-safe, which corresponds to the lowest level of safety. Let $(s_0, s_1, ..., s_{n-2}, s_{n-1})$ be the nonascending safety level sequence of node u's neighbors, then

1) The safety level of u is n if $(s_0, s_1, ..., s_{n-2}, s_{n-1}) \geq (n-1, n-2, ..., 1, 0)$[1]
2) The safety level of u is k if $(s_{n-k}, s_{n-(k-1)}, ..., s_{n-2}, s_{n-1}) \geq (k-1, k-2, ..., 1, 0) \wedge (s_{n-(k+1)} = k-1)$, where $1 \leq k \leq n-1$.

The safety level can be intuitively explained in the following two ways: 1) If a node is k-safe, there exists a Hamming distance path from this node to any other nodes within k distance. 2) If a node is k-safe, then there exists an $l(\leq k)$-level *incomplete spanning binomial tree* of any l-subcube, with this node being the root. An incomplete spanning binomial tree is a spanning binomial tree in which all the faulty nodes are leaves of the tree.

We have shown [10] that for a given faulty hypercube, there is only one number (safety level) for each node that satisfies the safety level definition. The safety level of each node can be easily calculated through a simple iterative algorithm. Initially, all the faulty nodes are 0-safe and all the nonfaulty nodes are n-safe. Then, each node exchanges its level with all its neighbors' and updates its level based on the safety level definition. The effect of 0-safe status of faulty nodes will first propagate to their neighbors, then neighbors' neighbors and so on. In the worst case at most $n-1$ rounds are required to stabilize the safety levels of all the nodes in the given n-cube and this is independent of the number of faulty nodes [10].

Fig. 1 shows a Q_4 with four faulty nodes (represented as black nodes): 1100, 0110, 0011, and 0001. Initially, all nonfaulty nodes are 4-safe, i.e. safe. After the first round of safety level exchange among neighboring nodes, the nodes (0010, 0111, 0100, and 1110) that have two or more faulty neighbors change their status from 4-safe to 1-safe. The status of all the other nodes remains unchanged. After the second round, the status of nodes 0000 and 0101 changes to 2-safe, since each of them has two 1-safe neighbors and one 2-safe neighbor. The safety level of every node remains stable after two rounds and the value associated with each node of Fig. 1 represents the safety level of the node.

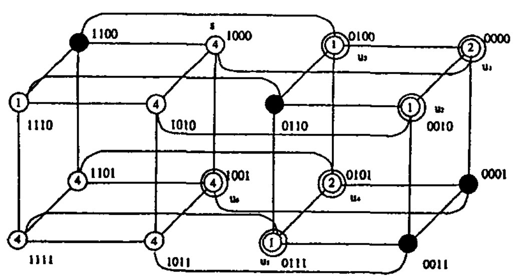


Fig. 1. An Q_4 with four faulty nodes.

III. Basic Strategy

The key issue in a multicast is how each intermediate node u (including the source node s) forwards a set of destination nodes $\{u_1, u_2, ..., u_m\}$ to its appropriate neighboring nodes. We represent destination nodes in terms of their relative addresses with respect to node u, $R = \{r_i\}$, where $r_i = u \oplus u_i$, $1 \leq i \leq m$. The *address summation*,

$$as = \sum_{r_i \in R} r_i ,$$

represents the distribution of destination nodes along different dimensions.

1. $seq_1 \geq seq_2$ if and only if each element in seq_1 is greater or equal to the corresponding element in seq_2.

$$|r_i| = \sum_{1 \le j \le n} r_i(j)$$

represents the distance between nodes u and u_i. For example, if a set of destination nodes $\{u_1, u_2, u_3\} = \{0101, 1001, 0000\}$ and node $u = 1010$, then $R = \{r_1, r_2, r_3\} = \{1111, 0111, 1010\}$ with $|r_1| = 4$, $|r_2| = 3$, and $|r_3| = 2$ and $as = 2232$. To avoid recalculation of relative addresses, we only calculate relative addresses at the source. Whenever a destination node with relative address r is passed to the next node along dimension d, the dth bit of r is set to zero, i.e. the new relative address associated with this destination node will be $r^{(d)}$. To ensure time optimality, we use the following simple strategy [2].

STRATEGY. *When the* dth *bit of* r_i, *the relative address of destination node* u_i *with respect to an intermediate node* u, *equals one,* $r_i^{(d)}$ *should be sent to* u*'s neighbor,* $u^{(d)}$, *along dimension* (d).

When r_i of destination node u_i has more than one 1 value at different bits (dimensions), then relative address r_i could be forwarded along either of these dimensions. In this case a *conflict* arises. To resolve the conflict, a priority order is determined among n dimensions. The formation of this priority order determines the result of a multicast. The priority order should be defined in such a way that it should minimize traffic by maximum sharing of common paths to destination nodes and the multicast message should not reach a *dead end* in a multicast. A dead end happens at an intermediate node when all the Hamming distance paths of a particular destination node are blocked by faulty neighbors. In this case, a detour path or a backtrack process must be used to reach that destination. To avoid a dead end situation we should limit the number of destinations to be forwarded to neighbors that have faulty nodes nearby. Actually, this is the reason that we use the safety level concept in the dimension priority decision.

Three approaches, safety-level-based multicasting (SLBM), modified safety-level-based multicasting (MSLBM), and address-sum-based multicasting (ASBM), are proposed in this paper. In the SLBM approach, the priority of a dimension is determined a priori based on the safety level of the neighbor along this dimension. The higher the safety level of the neighbor along a dimension, the higher the priority order of this dimension. Two approaches can be used when there are two or more dimensions along which the corresponding neighbors have the same highest safety level. In SLBM, a priority order among these dimensions is randomly selected. In the modified SLBM (MSLBM), the priority of a dimension is based on the corresponding bit value in the address summation of the destinations. Basically, if the neighbor along dimension d can carry maximum possible destination nodes, i.e., $as(d)$ is the maximum value of all the bits in the address summation, then d has the highest priority. When there is more than one bit that has the same maximum value in the address summation, the selection is random. The next highest priority dimension is determined using the same approach but is based on the updated destination set, i.e., the one after removing those nodes to be forwarded to dimensions in higher priorities.

In the ASBM approach, the dimension priority depends primarily on bit values in the address summation. A dimension has the highest priority if the corresponding neighbor can carry the maximum possible nodes. To ensure time optimality, only those destination nodes that are no more than k distance away from the selected neighbor will be included, where k is the safety level of this neighbor. In this case, the safety levels of all the neighbors and the relative distances of the destination nodes are used in the decision. A modified ASBM approach, following a similar approach as used in MSLBM, can be adopted when there is more than one neighbor that can carry the same maximum number of destination nodes. In this case, the priority among these neighbors is based on their safety levels. In ASBM, the priority among these nodes is randomly selected. The following two lemmas apply to all three multicasting approaches.

LEMMA 1. *If the source node is safe and an intermediate node* u *is* k*-safe, then any destination node received at node* u *is no more than* k *Hamming distance away from* u.

LEMMA 2 *The multicast message reaches each destination through a Hamming distance path from the source to the destination. None of the three multicasting approaches will cause a dead end situation.*

THEOREM 1. *A multicast generated by SLBM, MSLBM, or ASBM is guaranteed to be time optimal if the source node is safe in any faulty* n*-cube. When the source node is unsafe and there are no more than* $n - 1$ *faulty nodes, the length of each path from the source to a destination is either the same or exactly two more than that of the corresponding Hamming distance path.*

THEOREM 2. *A multicast generated by SLBM, MSLBM, or ASBM is guaranteed to be time optimal if the relative distance between the source and any destination is no more than the safety level of the source.*

Theorems 1 and 2 can be easily proved based on Lemmas 1 and 2 and the fact that for each unsafe node in an n-cube with at most $n - 1$ faulty nodes there is at least one safe neighbor [10]. In Theorem 1, destination nodes are random and the source node is restricted. In Theorem 2, the source node is random. The proposed approaches can still be applied if the destination nodes satisfy specific constraints. Theorem 2 applies to any faulty hypercubes, including disconnected hypercubes.

IV. EXAMPLES

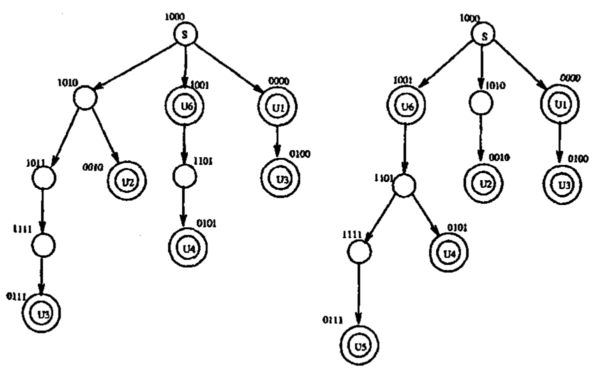
Let's reexamine the Q_4 shown in Fig. 1. Table 1 shows the descending status sequence (st) and the corresponding neighbor dimension sequence (ds) associated with each node in this Q_4. We use "–" to present a don't-care symbol for st and ds associated with faulty nodes. In Fig. 1, each destination is represented by a double-circled node, and each black node is a faulty node. Suppose the source node is the safe node 1000, and the multicasting set is $u = \{u_1, u_2, u_3, u_4, u_5, u_6\} = \{0000, 0010, 0100, 0101, 0111, 1001\}$. The set of relative addresses between the source and destinations nodes is $R = \{r_1, r_2, r_3, r_4, r_5, r_6\} = \{1000, 1010, 1100, 1101, 1111, 0001\}$. Therefore, the address summation is $as = 5323$.

TABLE I
THE st AND ds FOR EACH NODE IN THE Q_4 OF FIG. 1

node	*st*	*ds*	*node*	*st*	*ds*
0000	(4,1,1,0)	(4,3,2,1)	0001	-	-
0010	(4,2,0,0)	(4,2,3,1)	0011	-	-
0100	(2,2,0,0)	(3,1,4,2)	0101	(4,1,1,0)	(4,2,1,3)
0110	-	-	0111	(4,2,0,0)	(4, 2,3,1)
1000	(4,4,2,0)	(2,1,4,3)	1001	(4,4,4,0)	(3,2,1,4)
1010	(4,4,1,1)	(2,1,4,3)	1011	(4,4,4,0)	(3,2,1,4)
1100	-	-	1101	(4,4,2,0)	(3,2,4,1)
1110	(4,4,0,0)	(3,2,4,1)	1111	(4,4,1,1)	(3,2,4,1)

The SLBM approach uses only neighbors' safety levels in terms of the neighbor dimension sequence (ds) to determine the priority among the neighboring nodes. In this case, dimension 2 has the highest priority, followed by dimension 1, and then dimension 4. Dimension 3 has the lowest priority. Since the second bit value is one in r_2 and r_5, $r_2^{(2)}$, and $r_5^{(2)}$ together with the multicast message are passed to node 1010, the neighbor of 1000 along dimension 2. In the subsequent discussion, we assume that the multicast message is always attached with the relative address of each destination node forwarded from one node to another. Among the remaining nodes in R, r_4, and r_6 have value one in the first bit. Both $r_4^{(1)}$ and $r_6^{(1)}$ are passed to node 1001. Since the fourth bit value is one in the remaining r_1 and r_3, addresses $r_1^{(4)}$ and $r_3^{(4)}$ are passed to 1000's neighbor along dimension 4. No destination nodes are passed to the neighbor along dimension 3. The procedure is applied recursively to 1000's neighbors that receive destination nodes, and a multicast tree is formed as shown in Fig. 2a. The depth of this tree is the number of time steps used and the number of edges in the tree is the number of traffic steps used. In this case, the time steps are four and the traffic steps are 10.

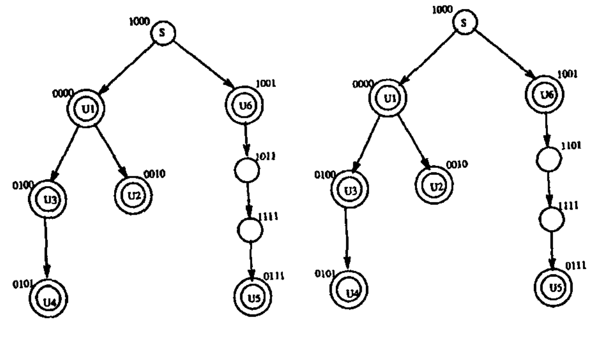
The MSLBM approach also uses the neighbor dimension sequence (ds) in determining the priority. However, when two or more two neighbors have the same highest safety level, the address summation (as) on the remaining destination nodes is used to break the tie. In Fig. 1, two neighbors of the source node 1000 along dimensions 1 and 2 have the same safety level. Based on as = 5323, the neighbor along dimension 2 (node 1010) can carry 2 (the second bit value of as) destination nodes, and the neighbor along dimension 1 (node 1001) can carry three destination nodes. Therefore, node 1001 has a higher priority over node 1010. As a result, $r_4^{(1)}$, $r_5^{(1)}$, and $r_6^{(1)}$ are passed to 1001, $r_2^{(2)}$ is passed to node 1010. Both $r_1^{(4)}$ and $r_3^{(4)}$ are passed to 1000's neighbor along dimension 4. Fig. 2b shows the resultant multicast tree with four time steps and nine traffic steps.



(a) (b)

Fig. 2. The multicast tree generated using (a) SLBM and (b) MSLBM.

In the ASBM approach, the dimension priority is dependent on the address summation of destination nodes. That is, the dimension with the maximum bit value in the address summation has the highest priority. The destination addresses with value one at this dimension will be sent to the corresponding neighbor. However, in order to avoid forwarding too many destinations to an unsafe or faulty neighbor, we forward destination addresses to a k-safe neighbor only when the corresponding destination nodes are less than or equal to k Hamming distance away from this k-safe neighbor. When there are two or more neighbors that can carry the same maximum number of destination nodes, the selection is random, although we can easily extend the ASBM algorithm in such a way that the selection among these neighbors is based on their safety levels. For the example of Fig. 1, as = 5323 at node 1000, therefore dimension 4 has the highest priority. Potentially, node 0000 can carry five destination nodes. However, only those nodes which are no more than two steps (two is the safety level of node 0000) away from 0000 will be passed to it. In this case, only the relative addresses of nodes u_1, u_2, u_3, and u_4 are passed to node 0000. The next highest dimension is determined based on the updated address summation, as = –112, on the remaining destination nodes u_5 and u_6. (Since dimension 4 has been selected, we use the don't-care symbol "–" at the fourth bit.) Therefore, dimension 1 is selected. Since the safety level of node 1001, the neighbor along dimension 1, is four, all the remaining destination nodes can be passed to it. At node 1001 the as = –11–, and neighbors 1011 and 1101 of 1001 have the same safety level and the same bit value in as. Therefore, u_5 can be forwarded through either node 1011 or node 1101 to the destination. The two resultant multicast trees are shown in Fig. 3a and b.



(a) (b)

Fig. 3. Two multicasting trees generated using ASBM.

V. Performance

A simulation was conducted on 4-cubes and 5-cubes with various numbers of faulty nodes and destination nodes. To simplify the simulation, we assume that all the destination nodes are fault free. Both faulty nodes and destination nodes are randomly selected from the 2^k nodes of the k-cube. Fig. 4 shows the average traffic steps generated from the three algorithms in 5-cubes with four faulty nodes. The optimal result and the lower bound are also shown in this figure, where optimal results are obtained by enumerating all the possible multicast trees that contain the destination set. Fig. 5 shows the percentage of saved traffic using MSLBN and ASBM compared with the one obtained by using SLBM in 5-cubes, where f represents the number of faulty nodes.

We make the following observations from the simulation results shown in Figs. 4 and 5:

- The average traffic steps achieved by using MSLBN and ASBM are close to the optimal case. In general, ASBM outperforms MSLBN. The SLBM approach does not produce results which are close to the optimal case as do the other two approaches.

  This result is expected since ASBM and MSLBM use more information than SLBM.

- Compared to the SLBM approach, the percentages of saved traffic generated by using the MSLBM and ASBM approaches reach their maximum when the number of destination nodes is about half of the total number of nodes. When the number of destination nodes is more than half of the total number of nodes, the percentage of saved traffic decreases.

  This occurs because there are not many nonoptimal paths to each destination when there are few destinations or when nearly all the nodes are destinations. Naturally, results generated by the SLBM are close to optimal, and there is not much room left for improvement. The number of nonoptimal paths to each destination reaches its maximum when the number of destinations reaches half of the total number of nodes. The effect of incorporating more information in path selection is most effective in this case.

In summary, the performances of ASBM and MSLBM are close to the optimal result. On the average, ASBM outperforms MSLBM which in turn outperforms SLBM. Because it is relatively simple to implement, MSLBM seems to be a cost-effective approach.

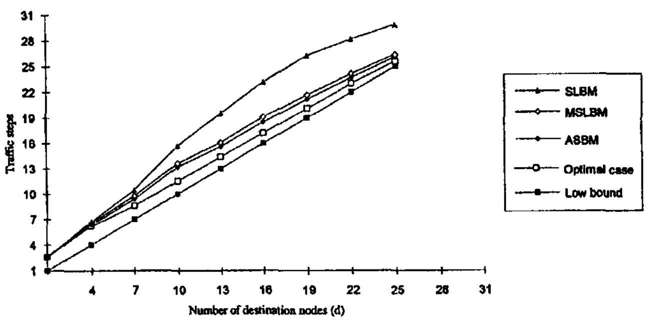


Fig. 4. Traffic comparison of the three algorithms.

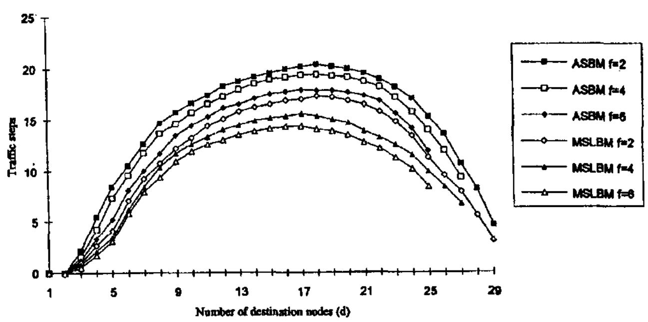


Fig. 5. Percentage of saved traffic with respect to SLBM.

VI. Conclusions

We have studied a fault-tolerant multicasting scheme based on limited global information. The concept of safety level associated with each node has been used to represent limited global information of fault distribution. Three approaches have been proposed which are applicable to hypercubes with up to $n-1$ faulty nodes. Our results show that, when the source node is safe, all three algorithms can find the shortest path for each destination node. When the source node is unsafe, multicasting can be initiated from one of its safe neighbors. Then in the worst case, two extra time steps are required for each destination node. Simulation results show that the performance of both the modified safety-level-based multicasting and the address-sum-based multicasting algorithms are very close to the traffic optimal solution with the former being a more cost-effective approach. Extending this approach to cover both faulty nodes and links and applying this approach to other network topologies such as meshes and tori are interesting problems for future research.

Appendix

A. Proof of Lemma 1

Let us classify all the nodes in the cube based on their distances from the source node. Clearly there are n groups of nodes in an n-cube. The source node s is a node that is at zero distance from the source node. Since all the destination nodes in the n-cube are no more than n Hamming distance away from s, the theorem automatically holds when s is n-safe. Assume the theorem holds for all the nodes which are distances j away from the source node and we arbitrarily select a node u which is k-safe from these nodes. That is, all the destination nodes received at u have at most k 1-bits (bits that take value 1) in their addresses. Suppose $(st_0, st_1, ..., st_{n-1})$ is the us status sequence of neighboring nodes in nonascending order and $(d_0, d_1, ..., d_{n-1})$ is the corresponding neighbor dimension sequence. Based on the safety definition in Section II, $(st_0, st_1, ..., st_{n-(k+1)}, st_{n-k}, st_{n-(k-1)}, ..., st_{n-2}, st_{n-1}) \geq (k-1, k-1, ..., k-1, k-1, k-2, ..., 1, 0)$. Clearly, the theorem holds for all the neighbors along dimensions $d_0, d_1, ..., d_{n-(k+1)}$, since their safety levels are at least $k-1$. (Note that the number of 1-bits in destination addresses will be reduced by one when these destination nodes are forwarded to appropriate neighbors.) Safety level of the neighbors along dimension $d_{n-(k-i)}$, $1 \leq i \leq k-1$, is at least $st_{n-(k-i)} = k-i-1$. Based on SLBM and MSLBM, a destination node r is selected to be forwarded to the neighbor along $d_{n-(k-i)}$ only when all the bit values of r at dimensions $d_0, d_1, ..., d_{n-k}, d_{n-(k-1)}, ..., d_{n-(k-(i-1))}$ are zero. That is, there are at least $n-(k-(i-1))+1 = n-(k-i)$ zero bits in r, i.e., at most $n-(n-(k-i)) = k-i$ 1-bits in r. The 1-bit at $d_{n-(k-i)}$ will be set to zero when r is forwarded to the neighbor along that dimension. Therefore, there are at most $k-i-1$ 1-bits in r which are less than or equal to the safety level of the neighbor along that dimension. For the ASBM approach, the theorem clearly holds. □

B. Proof of Lemma 2

All three approaches are based on the same strategy discussed in Section III: When the multicast message reachs a destination it must be through a Hamming distance path. We only need to prove the absence of dead ends. For the SLBM and MSLBM approaches, we consider the following two cases: Suppose node u is k-safe, $k > 1$. Based on the safety level definition, there is at most one faulty neighbor (assume it is along dimension d). For each destination r (excluding u and the neighbor along dimension d) there is at least one bit other than d that has value 1. Since dimension d has the lowest priority, r will not be forwarded to it. If the neighbor along dimension d is nonfaulty, the multicast message is forwarded to it; otherwise, no message is passed to this neighbor. The only destination node (if one exists) that is not selected is the node u itself, and a copy of the multicast message is kept at node u. Suppose node u is 1-safe. Based on the definition of the safety level concept, there are between 2 and $n-1$ faulty neighbors. Based on Lemma 1, only those destinations which are no more than 1 Hamming distance away from u are forwarded to u, i.e., each destination is either u's neighbor or u itself. Therefore, a dead end situation will never occur. For the ASBM approach, we only need to prove that each destination will be selected to be passed to one of the neighbors. Suppose the safety level of the current node is k and we randomly select a destination which is l $(\leq k)$ distance away. That is, there are l one bits in its relative address and there are l neighbors that are candidates to relay the multicast message. Based on the safety level definition, there is at least one neighbor that has a safety level no less than $l-1$. Therefore, this destination will be selected, i.e., no dead end situation will occur. □

References

[1] Y. Lan, "Fault-tolerant multidestination routing in hypercube multicomputers," *Proc. 12th Int'l Conf. Distributed Computing Systems*, pp. 632-639, June 1992.

[2] Y. Lan, A.H. Esfahanian, and L.M. Ni, "Multicast in hypercube multiprocessors," *J. Parallel and Distributed Computing*, vol. 8, pp. 30-41, 1990.

[3] A.C. Liang, S. Bhattacharya, and W.T. Tsai, "Fault-tolerant multicasting on hypercubes," to appear in *J. Parallel and Distributed Computing*.

[4] X. Lin and L.M. Ni, "Multicast communication in multicomputer networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 10, pp. 1,105-1,117, Oct. 1993.
[5] X. Lin and L.M. Ni, "Multicast communication in multicomputer networks," *Proc. 1990 Int'l Conf. Parallel Processing*, vol. III, pp. 114-118, 1990.
[6] 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.
[7] G.D. Pifarre, S.A. Felperin, L. Gravano, and J.L.C. Sanz, "Routing techniques for massively parallel systems," *Proc. IEEE*, vol. 74, no. 4, pp. 488-503, Apr. 1991.
[8] Y. Saad and M.H. Schultz, "Topological properties of hypercubes," *IEEE Trans. Computers*, vol. 37, no. 7, pp. 867-872, July 1988.
[9] J.P. Sheu and M.Y. Su, "A multicast algorithm for hypercube multiprocessors," *Proc. 1992 Int'l Conf. Parallel Processing*, vol. III, pp. 18-22, 1992.
[10] J. Wu, "Broadcasting in injured hypercubes using incomplete spanning binomial trees," Technical Report TR-CSE-92-29, Dept. of Computer Science and Eng., Florida Atlantic Univ., Nov. 1992.

Correction

Corrections to "Low Overhead Multiprocessor Allocation Strategies Exploiting System Space Capacity for Fault Detection and Location"

Srinivasan Tridandapani, Arun K. Somani, and Upender R. Sandadi

In the July 1995 issue of this transaction in the above-mentioned article, errors were made in the printing of two equations. The corrected equations follow:

$$FDC(n,\rho)=\left(\left(\frac{1}{n}\right)\sum_{i=3}^{n} i\times\left(\left(\gamma_{r,i}/\mu_{r,i}\right)+\left(\gamma_{v,i}/\mu_{v,i}\right)\right)\right)\times 100. \quad (16)$$

$$FLC(n,\rho)=\left(\left(\frac{1}{n}\right)\sum_{i=3}^{n} i\times\left(\left(\gamma_{r,i}/\mu_{r,i}\right)+\left(\gamma_{v,i}/\mu_{v,i}\right)\right)\right)\times 100. \quad (17)$$