JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 19, 467-478 (2003)

Local-Safety-Information-Based Fault-Tolerant Broadcasting in Hypercubes*

DONG XIANG, AI CHEN AND JIE WU+
School of Software
Tsinghua University
Beijing, 100084 P.R.C.
+*Department of Computer Science and Engineering*
Florida Atlantic University
Boca Raton, FL 33431, U.S.A.

This paper presents a method for fault-tolerant broadcasting in faulty hypercubes using a new metric called local safety. A new concept of the broadcast subcube is introduced, based on which various techniques are proposed to improve the performance of a broadcast algorithm. An unsafe hypercube can be split into a set of maximal safe subcubes. We show that if these maximal safe subcubes meet certain requirements given in the paper, broadcasting can still be carried out successfully, and in some cases optimal broadcast is still possible. The sufficient condition for optimal broadcasting is also presented. Limited backtracks are utilized in the process of broadcasting by setting up a partial broadcast tree. Extensive simulation results are presented.

Keywords: broadcasting, fault tolerance, hypercubes, local safety information, optimal solutions

1. INTRODUCTION

The hypercube architecture can handle a reasonable amount of message traffic and also provide some degree of fault-tolerance [5]. Recently, Duato [4] and Ould-Khaoua [7] presented experimental studies and showed that hypercubes are quite suitable for distributed shared memory systems and multicomputers.

Efficient broadcasting of data is one of the keys to the performance of a multicomputer. Broadcasting is the process of transmitting data from a node called the source to all other nodes once and only once. One-to-all fault-tolerant broadcasting in a faulty hypercube passes a message from a source to all fault-free nodes [3, 6, 10, 11, 13]. Some limited-global-fault-information-based methods have been introduced to deal with fault-tolerant communication in hypercubes. Lee and Hayes [6] proposed a fault-tolerant broadcast algorithm based on the safe node concept. Priority order is determined by the safety levels of neighbors. Wu and Fernandez [13], and Chiu and Wu [2] refined the safe node concept. As in [6], a message can be broadcast reliably only if the binary *n*-cube is safe although reliable message passing is still possible in an unsafe hypercube in many cases. A mechanism called the safety level was proposed to assist an

Received May 15, 2002; accepted July 25, 2002.
Communicated by Biing-Feng Wang, Stephan Olariu and Gen-Huey Chen.
* A preliminary version of the paper was presented at the 2002 International Conference on Parallel and Distributed Systems, Chungli, Taiwan.

efficient fault-tolerant broadcast in [14]. Priority order for forwarding the broadcast data is determined by the safety level numbers. A directed safety level [3] further improves the performance of the algorithm presented in [14].

We have noticed that some resilience properties of hypercube topology still have not been utilized by the above methods. Local safety [15] is proposed in this paper to explore the resilience properties of the hypercube. An unsafe hypercube can be split into a unique set of maximal safe subcubes. Message-passing inside a maximal safe subcube can be completed reliably.

2. PRELIMINARIES

A binary n-cube (or simply n-cube) has 2^n nodes (or processors). Each node can be represented by a sequence of binary bits ($a_n a_{n-1} \ldots a_1$), where $a_i \in \{0, 1\}$. A subcube SC of a hypercube can be represented by a sequence of n bits $c_n c_{n-1} \ldots c_2 c_1$, where $c_i \in \{0, 1, *\}$ and "*" indicates a don't care (can be assigned both 0 and 1). The *Hamming distance* between nodes x and y is denoted as $H(x, y)$. The *spanning subcube* $SC(x, y)$ is the smallest subcube that contains both x and y. $s^{(i)}$ is the neighbor of s along dimension i in the hypercube. Two nodes are connected by a bidirectional link if and only if the binary representations of the two nodes differ in exactly one bit. We only consider message-passing between fault-free nodes. A path is feasible if there is no faulty node in it. A path is called a minimum path if the length of the path is equal to the Hamming distance from the source to the destination. We consider only node faults, which can be extended to the mixed fault model of both node and link faults. An incomplete spanning binomial tree is utilized to implement broadcasting. Definition 2 presents the refined safe node concept of [2, 13].

Definition 1 A binomial tree in an n-cube is a connected subgraph of an n-level spanning binomial tree with the same root node that connects all the nodes in the n-cube. An incomplete spanning binomial tree in a faulty binary n-cube is a connected subgraph of an n-level spanning binomial tree with the same root node that connects all the fault-free nodes in the n-cube.

Definition 2 A fault-free node in an n-cube is *unsafe* if it has at least two faulty neighbors, or three or more unsafe or faulty neighbors. An unsafe node is *ordinarily unsafe* if it has at least one safe neighbor; otherwise, it is *strongly unsafe*. A faulty hypercube is called *unsafe* if it contains no safe node; otherwise, it is a safe cube.

The incomplete binomial tree for broadcasting a message in the faulty 4-cube is presented in Fig. 1. Each fault-free node has a broadcast label indicating the range the message that should be distributed from the node. A message is broadcast as follows: when a node receives a broadcast label (which is initialized to all 1's), it resets a 1-bit in the broadcast label (say at dimension i) and sends the updated broadcast label to the neighbor. This process is repeated until all the 1-bits in the original broadcast label are reset.

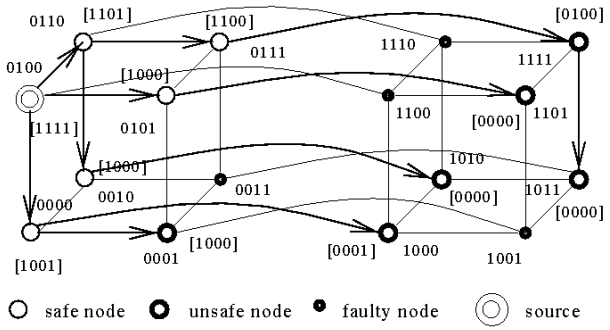

Fig. 1. Binomial-tree-based broadcasting via limited-global safety.

Definition 3 Broadcast subcube of a node is a subcube to which the node should broadcast the message.

The broadcast subcube at *u* can be derived by replacing certain bits of *u*'s address with don't cares. These bits correspond to 1-bits in the broadcast label. Let node 10100 in a 5-cube receive a broadcast label [11010]. The broadcast subcube of node 10100 is **1*0. Let node 01011 receive a broadcast label [11010]. The broadcast subcube of 01011 is **0*1. The broadcast subcube of the source of the broadcast message is the *n*-cube.

Wu and Fernandez [13] have shown that any *n*-cube is safe if the number of faulty nodes is no more than *n*. It is quite possible for an *n*-cube to be unsafe when it contains *n* or more than *n* faulty nodes [2, 13]. Fault-tolerant communication inside an unsafe *n*-cube is impossible according to the safe node concept [2, 6, 13].

Definition 4 A node in an *n*-cube is locally unsafe inside a subcube if it has at least two faulty neighbors, or at least three locally unsafe or faulty neighbors inside the subcube; otherwise, it is locally safe in the subcube. The subcube is unsafe if it contains no locally safe node; otherwise, it is a safe subcube. Locally unsafe nodes inside a subcube SC are classified as follows: a locally unsafe node is locally ordinarily unsafe if it has at least one locally safe neighbor in SC; otherwise, it is a locally strongly unsafe node.

A subcube can still be safe even though all the nodes outside of it are faulty. A definition of a maximal safe subcube is presented as follows. Each node keeps its own local safety information and that of its fault-free neighbors.

Definition 5 An *m*-dimensional subcube is a maximal safe subcube if it is safe, and any *k*-dimensional ($k \geq m + 1$) subcube that contains it is safe.

3. FAULT-TOLERANT BROADCASTING IN A SAFE SUBCUBE

Wu and Fernandez [13] proposed a fault-tolerant broadcast algorithm in a safe hypercube, which presents only optimal broadcasting based on the refined safe node con-

cept (Definition 2). It is believed that broadcasting is feasible in most cases if the broadcast subcube of the source is contained in a safe subcube. A new algorithm supporting non-optimal broadcasting is presented here. Some effective steps are also introduced in the following to improve the effectiveness of fault-tolerant broadcasting:

- try to avoid sending the broadcast label and message to fault-free neighbors which have at least two faulty neighbors in the broadcast subcube;
- consider a source that has at least two faulty neighbors in the broadcast subcube, and send the broadcast label to the last fault-free neighbor along dimension *i* without resetting the *i*-th bit.

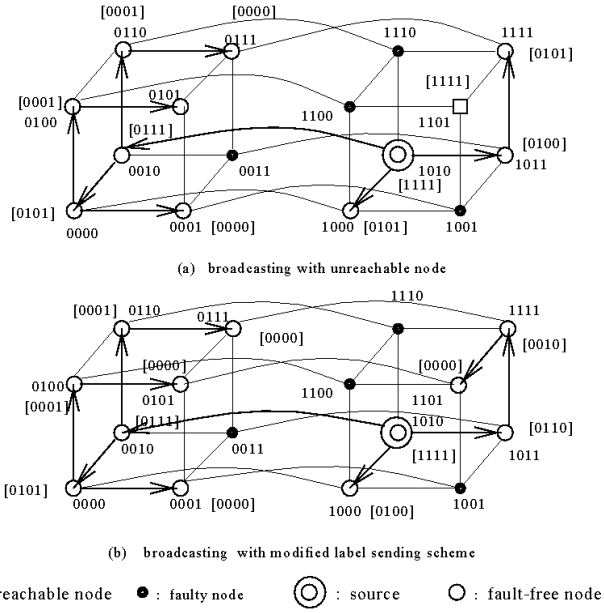

Fig. 2. Avoidomg forwarding a message and label to nodes with at least two faulty neighbors in the broadcast subcube.

In order to implement the first scheme, it is reasonable for each node *s* to keep an $n \times n$ matrix F to record its faulty neighbors and its fault-free neighbor's faulty neighbors. This scheme is quite useful. Consider the broadcast problem with the source 1010 in the faulty hypercube as given in Fig. 2. It is clear that the message should be sent to 0010 with broadcast label [0111] first because 0010 is safe in the 4-cube. The message can be passed optimally in its broadcast subcube 0***. The message should be sent to 1000 with label [0101] according to the safety information of 1010's neighbors because 1000 is ordinarily unsafe and 1011 is strongly unsafe [8]. This prevents the message from reaching 1101 as shown in Fig. 2(a). The message should be sent to 1011 with label [0110] according to the first scheme because 1000 has two faulty neighbors in its broadcast subcube, which causes the message to reach all the nodes along minimum paths as shown in Fig. 2(b). The reason that the above scheme can reach node 1101 as shown

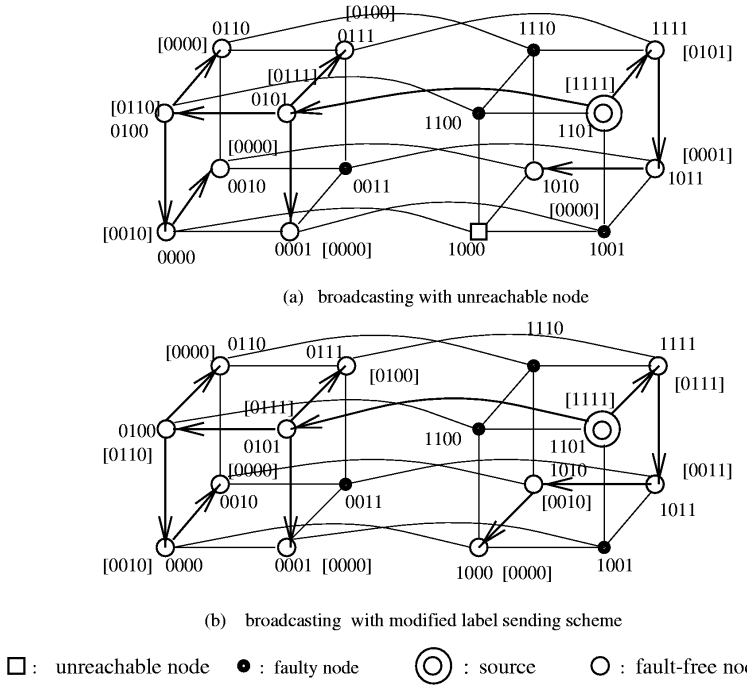

Fig. 3. Fault-tolerant broadcasting using the modified label sending scheme.

in Fig. 2, while the procedure in [8] cannot, is that an ordinarily unsafe node in the hypercube may be locally strongly unsafe in a subcube, while a strongly unsafe node in the hypercube may be locally safe in the subcube.

Our method supports non-optimal broadcasting if optimal broadcasting is impossible. The second scheme can make many unreachable nodes reachable. Consider that a message that is broadcast from 1101 as shown in Fig. 3. The source sends the message with a label [0111] to 0101 because 0101 is safe. The message received by 0101 can be optimally sent to all the fault-free nodes inside its broadcast subcube as shown in Fig. 3(a). Node 1101 then sends the message to 1111 with a label [0101]. It is clear that the message cannot reach 1000 by using the label sending scheme [5, 8]. As shown in Fig. 3(b), 1101 sends label [0111] to node 0101, and sends label [0111] to node 1111 without resetting the 2-nd bit. Node 1111 does not send the message back to 1101. So far, node 1000 is reachable from the source although the message is not passed along minimum paths. The above techniques can be used to broadcast a message along non-minimum paths when optimal broadcasting is unavailable, and when the broadcast subcube is contained in a maximal safe subcube. The second scheme introduced above, which is important, makes the message deroute inside some small subcubes. As shown in Fig. 3(b), the message is broadcast in 4 steps although it is not broadcast optimally. The number of deroutes is limited to no more than 2. Details about broadcasting a message inside a safe subcube can be found in [11].

4. FAULT-TOLERANT BROADCASTING WITH LIMITED BACKTRACKS

Assume that each fault-free node keeps the local safety of its fault-free neighbors and of itself by using the scheme introduced in section 3. We will show optimal broadcasting is still possible in many cases even though the hypercube is unsafe.

Theorem 1 An optimal broadcast from node s exists if s is locally safe in its broadcast subcube.

Theorem 1 implies that a message can be broadcast optimally even though the n-cube is unsafe. We construct several broadcast subcubes starting from the source. Consider that the source has at most one faulty neighbor in the n-cube. Let Q_{n-1}, Q_{n-2}, Q_{n-3}, …, Q_{n-m} be broadcast subcubes of the fault-free neighbors $s^{(i_1)}$, $s^{(i_2)}$, …, $s^{(i_m)}$ ($n - 1 \leq m \leq n$) of the source, where $i_1, i_2, \ldots, i_m \in \{1, 2, \ldots, n\}$ and the subscripts indicate the sizes of the corresponding broadcast subcubes. The following theorem presents the sufficient condition for the existence of an optimal broadcast inside an unsafe n-cube.

Theorem 2 There exists an optimal broadcast if Q_{n-1}, Q_{n-2}, …, Q_{n-m} are safe and $s^{(i_1)}$, $s^{(i_2)}$, …, $s^{(i_m)}$ are locally safe in the corresponding broadcast subcubes, respectively, even though the n-cube is unsafe.

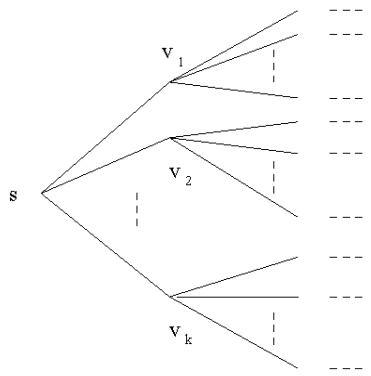

Fig. 4. Backtracking when setting up a partial broadcast tree.

We consider an approach that deals with the way of partitioning the hypercube to meet the following condition: each fault-free neighbor receives a broadcast label stating that its broadcast subcube is contained in a maximal safe subcube in which the fault-free neighbor has less than two faulty nodes inside its broadcast subcube. An empirical measure is utilized to handle the subcube partitioning when some fault-free neighbors of the source cannot meet the above conditions.

$$safe(v) = size(k) \cdot safety(v, k),$$

where $size(k)$ is the size of the maximal safe subcube k that contains the node v, in which the local safety information of the node is $safety(v, k)$ (empirical constants 5 for locally

safe, 3 for locally ordinarily unsafe, 2 for locally strongly unsafe, and 0 for faulty). Our algorithm allows backtracking when setting up a partial broadcast tree. First, we try to find a maximal safe subcube that contains the biggest broadcast subcube, starting from the fault-free neighbor. The algorithm can set up a partial broadcast tree as shown in Fig. 4, whose depth is at most 3 (3 is an empirical constant). If a maximal safe subcube cannot be found to contain the broadcast subcube, we backtrack to the fault-free neighbor with the second most heuristic value. We continue the above process until a maximal safe subcube is found to contain the broadcast subcube of the largest broadcast subcube. A similar technique is adopted to find a maximal safe subcube that contains the broadcast subcube of the second largest broadcast subcube. We continue the above process until a maximal safe subcube is found that contains the broadcast subcube of the source. It should be noted that when the header of the broadcast message is forwarded to the next neighbor, the broadcast label of the node and the neighbor get a broadcast label with the corresponding bit reset, similar to the process of broadcasting.

Algorithm broadcast2()

1. If node s is the broadcast source, for $i = 1$ to n, $label[i] \leftarrow 1$; do 2, 3, 4.
2. If the broadcast subcube of s is contained in a maximal safe subcube *msc*, then call *broadcast*1(s) based on the local safety information of *msc*; otherwise $f_{br} \leftarrow 0$; if s has at least two faulty neighbors in its broadcast subcube, then 5, otherwise 3, 4.
3. While $f_{br} = 0$, do
   (a) $f_{br} \leftarrow 1$, for $i = 1$ to n
      (i) if $label[i] = 1$ and the broadcast subcube of $s^{(i)}$ is contained in a maximal safe subcube, in which $s^{(i)}$ is locally safe, then (ii),
      (ii) $label[i] \leftarrow 0$; send the message and *label* to $s^{(i)}$; call broadcast1($s^{(i)}$); $f_{br} \leftarrow 0$.
   (b) for $i = 1$ to n
      (i) if $label[i] = 1$ and the broadcast subcube of $s^{(i)}$ is contained in a maximal safe subcube, and if $s^{(i)}$ has at most one faulty neighbor inside the broadcast subcube, then (ii),
      (ii) $label[i] \leftarrow 0$, send the message and the *label* to $s^{(i)}$; call broadcast1($s^{(i)}$); $f_{br} \leftarrow 0$.
   (c) for $i = 1$ to n
      (i) if $label[i] = 1$ and the broadcast subcube of $s^{(i)}$ is contained in a maximal safe subcube, then (ii),
      (ii) $label[i] \leftarrow 0$, send the message and *label* to $s^{(i)}$; call broadcast1($s^{(i)}$); $f_{br} \leftarrow 0$.
4. If there still exists a fault-free neighbor of s, which has not received the message and broadcast label, call the above backtracking process and send the message along the set up partial broadcast tree.
5. Do the same steps as in 3 and 4; each time, only check whether node $s^{(i)}$ is the last unprocessed fault-free neighbor of s; if it is not, send the message and *label* by resetting $label[i]$; if $s^{(i)}$ is the last unprocessed fault-free neighbor of s, send the message and *label* to $s^{(i)}$ without resetting $label[i]$.

A flag f_{br} , initially set to zero, is adopted to guide whether the broadcast should be continued or not. Let us show how to generate the broadcast subcube of $s^{(i)}$ in step 3. For example, if node 10101 (v) has a broadcast label [11011], the broadcast subcube of 00101 ($v^{(5)}$) should be 0*1**, while the broadcast subcube of 10100 ($v^{(1)}$) should be **1*0. The following theorem presents a sufficient condition for optimal broadcasting inside an unsafe n-cube.

Theorem 3 The algorithm *broadcast2*() can optimally broadcast the message of the source s in n steps if s has at most one faulty neighbor, and if a sequence of fault-free neighbors $s^{(i_1)}$, $s^{(i_2)}$, …, $s^{(i_m)}$ ($n - 1 \le m \le n$) is locally safe in a sequence of maximal safe subcubes which contain the broadcast subcubes of the nodes $s^{(i_1)}$, $s^{(i_2)}$, …, $s^{(i_m)}$, respectively, where $i_1, i_2, \ldots, i_m \in \{1, 2, \ldots, n\}$.

5. SIMULATION RESULTS

Flit-level simulators were implemented in a centralized environment [1], which provided a comparison of the latency, throughput, broadcast ratio, and minimum broadcast ratio between [3, 14] and the proposed method under various conditions, including different faulty nodes, message lengths, and load rates. Figs. 5-8 present flit-level performance evaluations of the local safety, safety level and directed safety level. The message length was 16 flits, the load rate (flit/node/cycle data inserted) was set to 1.0, and the buffer size was 64 flits. Results for each pattern were obtained by running 30,000 cycles on the system, where the start-up cycles (the first 10,000 cycles) were not included. Throughput was obtained by using the following expression:

$$\text{throughput} = \frac{\text{no. of messages} \times \text{message length}}{\text{cycles} \times \text{no. of fault - free nodes}}.$$

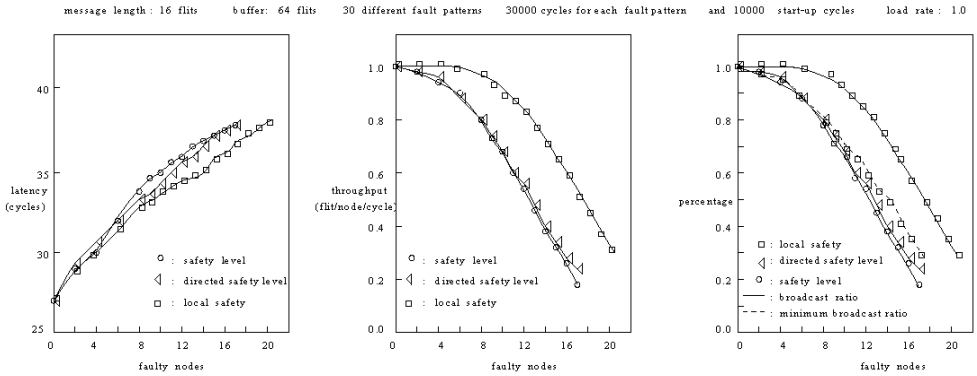

Fig. 5. Performance comparison in 6-cubes.

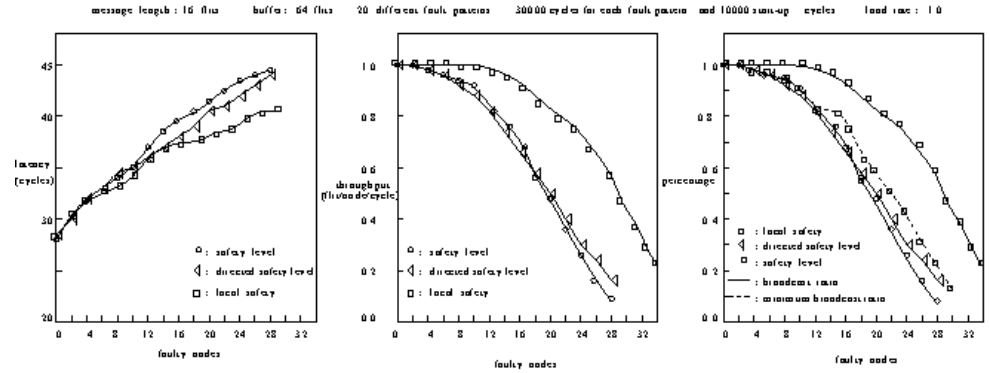

Fig. 6. Performance comparison in 7-cubes.

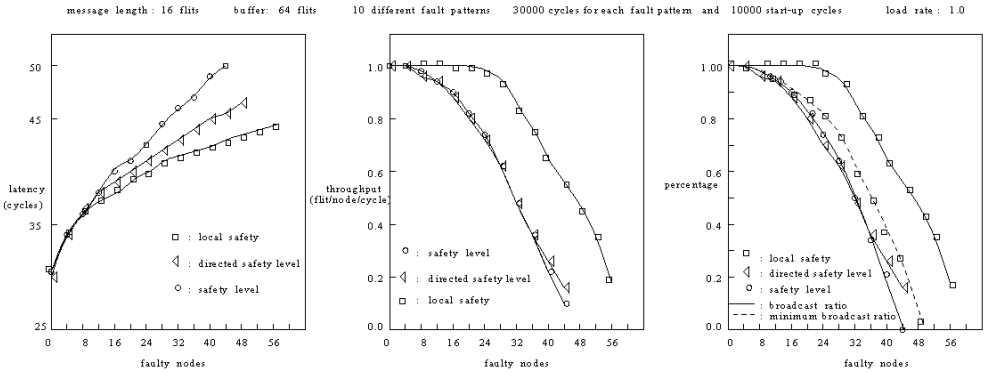

Fig. 7. Performance comparison in 8-cubes.

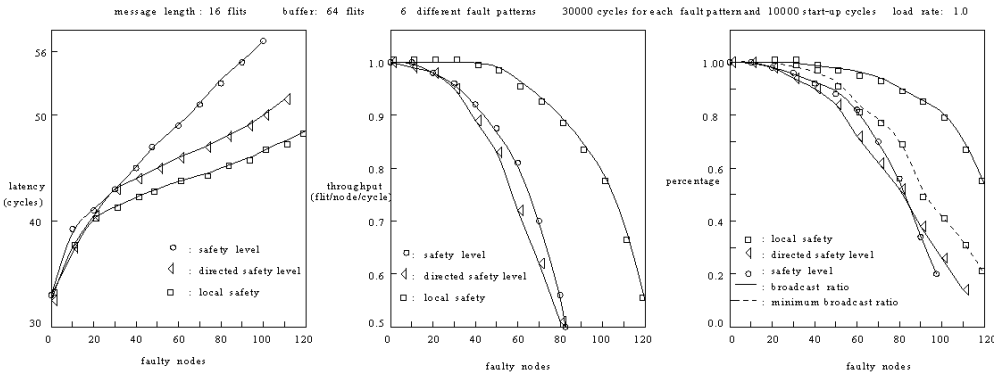

Fig. 8. Performance comparison in 10-cubes.

Local safety was consistently better than the safety level under latency when broadcasting a message. The latency of a message based on the safety level was always greater than that of local safety. The difference in latency between local safety and the two metrics became greater as the number of faults increased in a system. Local safety was consistently better than the safety level under throughput when broadcasting a message. Local safety achieved much better throughput than directed safety did in almost all cases. Local safety also achieved a better broadcast ratio than both the safety level and directed safety level did in all cases as shown in the figures.

Fig. 9 presents performance of local safety when the message length was 64 flits and the buffer size was 256 flits, and when the system had various load rates. It is observed that the latency of a broadcast message increased drastically when the load rate reached 1.4 for the faulty 10-cube with 80 faulty nodes. As for the fault-free 10-cube, latency increases greatly when the load rate was about 1.6. Fig. 10 presents performance of local safety for messages of different sizes in a faulty 10-cube with 80 faulty nodes. It is shown that the throughput and broadcast ratio of the system were not sensitive to the size of the message.

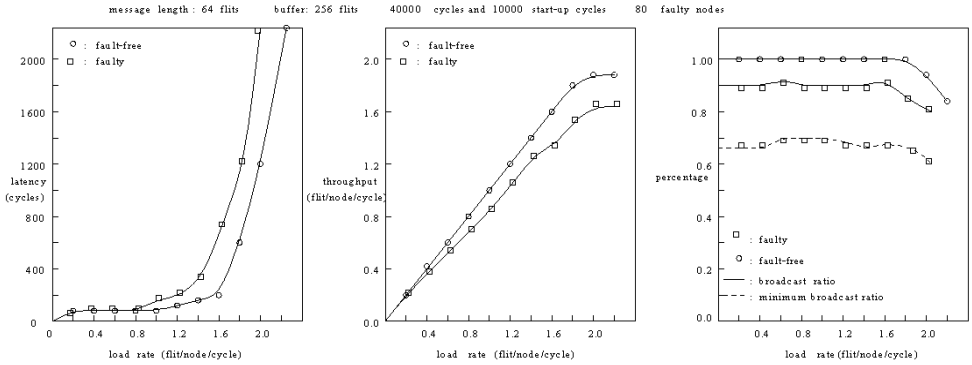

Fig. 9. Performance evaluation of 10-cubes with different load rates.

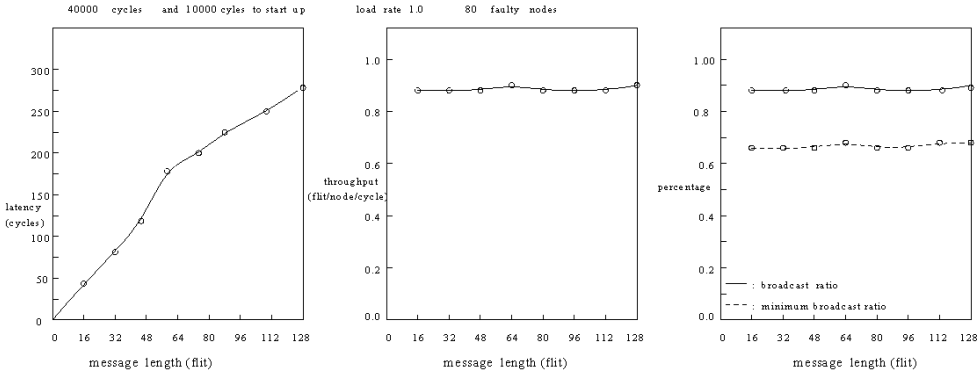

Fig. 10. Performance evaluation of 10-cubes with different message lengths.

6. CONCLUSIONS

Local safety information was adopted in this study to handle fault-tolerant broadcasting in wormhole-routed hypercubes. Some further resilience of the hypercube topology is utilized by the broadcast algorithm. Local safety information is well utilized in the fault-tolerant broadcast algorithm by only considering the safety of the broadcast subcube. The sufficient condition for optimal broadcast of a message in an unsafe hypercube has also been presented. The proposed method allows limited backtracking for setting up a partial broadcast tree. Extensive simulation results have been presented, and the proposed method compared with previous ones. It has been shown that the proposed method greatly outperforms the previous methods [3, 14] in almost all cases.

REFERENCES

1. R. V. Boppana and S. Chalasani, "A framework for designing deadlock-free wormhole routing algorithms," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, 1996, pp. 169-183.
2. G. M. Chiu and P. S. Wu, "A fault-tolerant routing strategy in hypercube systems," *IEEE Transactions on Computers*, Vol. 45, 1996, pp. 143-155.
3. G. M. Chiu, "Fault-tolerant broadcasting algorithm for hypercubes," *Information Processing Letters*, Vol. 66, 1998, pp. 93-99.
4. J. Duato and M. P. Malumbres, "Optimal topology for distributed shared-memory multiprocessors: hypercubes again?" in *Proceedings of 2nd International Euro-Par Conference*, 1996, pp. 205-212.
5. J. Laudon and D. Lenoski, "The SGI origin: A ccNUMA highly scalable server," in *Proceedings of International Symposium on Computer Architecture*, 1997, pp. 241-251.
6. T. C. Lee and J. P. Hayes, "A fault-tolerant communication scheme for hypercube computers," *IEEE Transactions on Computers*, Vol. 41, 1992, pp. 1242-1256.
7. G. Min and M. Ould-Khaoua, "A performance model of pipelined circuit switching in hypercubes," *Microprocessors and Microsystems*, Vol. 25, 2001, pp. 213-220.
8. M. Ould-Khaoua, "On optimal network for multicomputers: Torus or hypercube?" in *Proceedings of 4-th International Euro-Par Conference*, 1998, pp. 989-992.
9. S. Park and B. Bose, "All-to-all broadcasting in faulty hypercubes," *IEEE Transactions on Computers*, Vol. 46, 1997, pp. 749-755.
10. C. S. Raghavendra, P. J. Yang, and S. B. Tien, "Free dimensions-An effective approach to achieving fault tolerance in hypercubes," *IEEE Transactions on Computers*, Vol. 44, 1995, pp. 1152-1157.
11. P. Ramanathan and K. G. Shin, "Reliable broadcast in hypercube multicomputers," *IEEE Transactions on Computers*, Vol. 37, 1988, pp. 1654-1657.
12. C. L. Seitz, "The cosmic cube," *Communications of the ACM*, Vol. 28, 1985, pp. 22-33.
13. J. Wu and E. B. Fernandez, "Reliable broadcasting in faulty hypercube computers," *Microprocessing and Microprogramming*, Vol. 46, 1993, pp. 241-247.
14. J. Wu, "Safety levels-An efficient mechanism for achieving reliable broadcasting in hypercubes," *IEEE Transactions on Computers*, Vol. 44, 1995, pp. 702-706.

15. D. Xiang, "Fault-tolerant routing in hypercube multicomputers using local safety information," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, 2001, pp. 942-951.

Dong Xiang received the BS and MS degrees in Computer Science from Chongqing University in 1987 and 1990, respectively. He received the Ph.D. degree in Computer Engineering from the Institute of Computing Technology, the Chinese Academy of Sciences, in 1993. He visited Concordia University, Canada, and the University of Illinois, Urbana Champaign, in 1994-1995 and 1995-1996, respectively, both times as a postdoctor. He has been with the Institute of Microelectronics, Tsinghua University, as an Associate Professor since October, 1996. He is now with School of Software, Tsinghua University. His research interests include the design and testing of digital systems, fault-tolerant computing, and distributed computing. He is the author of "Digital Systems Testing and Design for Test ability" (Science Press, in Chinese). Dong Xiang is a member of IEEE.

Ai Chen received his BS degree from the Institute of Microelectronics, Tsinghua University, in 2001. He is now working towards the MS degree at the Institute of Microelectronics, Tsinghua University. His research interests include routing protocols in interconnection networks and fault-tolerant computing.

Jie Wu received the BS and MS degrees from Shanghai University of Science and Technology (now Shanghai University) in 1982 and 1985, respectively; and the Ph.D. degree from Florida Atlantic University in 1989. He is currently a Professor in the Department of Computer Science and Engineering, Florida Atlantic University. He has published over 150 papers in various journals and conference proceedings. His research interests are in the areas of mobile computing, routing protocols, fault-tolerant computing, and interconnection networks. Dr. Wu serves on many conference committees and editorial boards. He was a co-guest-editor of IEEE Trans. on Parallel and Distributed Systems and the Journal of Parallel and Distributed Computing. He is the author of the text "Distributed System Design," published by CRC press. Dr. Wu was the recipient of the 1996-1997 and 2001-2002 Researcher of the Year Award at Florida Atlantic University. He served as an IEEE Computer Society Distinguished Visitor. Dr. Wu is a member of ACM and a senior member of IEEE.