Reliable Broadcasting in Wormhole-Routed Hypercube-Connected Networks Using Local Safety Information

Dong Xiang, *Member, IEEE*, Ai Chen, and Jie Wu, *Senior Member, IEEE*

***Abstract*—This paper presents a method to cope with reliable broadcasting in faulty hypercubes using local safety information. A new definition, broadcast subcube, is introduced, with which various techniques are proposed to improve performance of the broadcast algorithm. Local safety information is well used in the fault-tolerant broadcast algorithm by considering only safety of the broadcast subcube. An unsafe hypercube can be split into a set of maximal safe subcubes. If these maximal safe subcubes meet certain requirements (listed in this paper), then broadcasting can still be done successfully and, in some cases, optimal broadcast is still possible. The sufficient condition for optimal broadcast of a message is presented in an unsafe hypercube. Extensive simulation results show that the proposed method outperforms previous methods, in all cases.**

***Index Terms*—Broadcast subcube, fault-tolerant broadcasting, hypercube, local safety, maximal safe subcube.**

ACRONYMS[1]

BSC	broadcast SC
MSC	maximal safe SC
SC	subcube.

NOTATION

$H(x, y)$	the Hamming distance between nodes x and y
$p_h(i, m)$	Pr{i, of the m faults, fall into an h-dimensional SC}
SC(x, y)	spanning SC: the smallest SC that contains both x and y
$s^{(i)}$	the neighbor of s along dimension i in the hypercube
$*$	a don't-care (can be assigned both 0 and 1).

I. INTRODUCTION

Definition

Broadcasting: The process of transmitting data from a node (called the source) to all other nodes, once and only once.

The hypercube architecture can handle a reasonable amount of message traffic, and also provide some degree of fault-tolerance. Several commercial or research hypercube systems have been constructed in the past 2 decades [9], [14]. For example, the recently built SGI Origin 2000 multiprocessor machine of SGI [9] uses hypercube interconnection structure. References [5] and [10] present experimental studies and show that hypercubes are quite suitable for distributed shared memory systems and multi-computers. When some nodes or links fail, communication between fault-free nodes should still continue. Fault-tolerant communication [2]–[4], [8], [11]–[13], [15]–[19] has been studied extensively.

Efficient broadcasting of data is one of the keys to the performance of a multi-computer. Data broadcasting in fault-free networks is studied intensively in [6], [7]. One-to-all fault-tolerant broadcasting passes a message from a source to all fault-free nodes in a faulty hypercube [8], [12], [13], [15]–[17]. Reference [13] introduces a reliable broadcast scheme, in which each node can receive more than 1 copy of the broadcast data. This method is particularly suitable for critical applications. A free dimension is defined as a dimension across which both end-nodes are fault-free [12]. Free dimensions can be used to partition an n-cube into SC such that each SC contains at most 1 faulty node. Such partitioning helps in designing efficient fault-tolerant communication algorithms. Reference [11] presents an all-to-all broadcast algorithm for hypercube with up to $n/2$ link failures in a binary n-cube, and presents a new concept, free dimension, corresponding to link failures.

Several limited-global-fault-information-based methods are introduced to deal with fault-tolerant communication in hypercubes [2], [3], [8], [15]–[19]. Reference [8] proposes a fault-tolerant broadcast algorithm based on the safe-node concept. Priority-order is determined based on status of neighbors of the node under process to send the broadcast label and the message in order to avoid communication difficulties. References [2], [15] refine the safe-node concept. Like [8], 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. Reference [16] proposes a mechanism, safety level, to assist an efficient fault-tolerant broadcast. Priority-order to forward the broadcast data is determined by the safety-level numbers. Directed safety level [3] improves performance of the algorithm in [16].

Much further resilience of hypercube topology has not been used by the above methods. Reference [18], [19] present local safety to handle fault-tolerant multi-casting and routing in hypercubes. Local safety is proposed to cope with fault-tol-

Manuscript received June 25, 2001; revised December 4, 2001; March 25, 2002. Responsible Editor: W.-T. K. Chien.

D. Xiang is with the School of Software, Tsinghua University, Beijing 100084, PR China (e-mail: DXiang@mail.tsinghua.edu.cn).

A. Chen is with the Institute of Microelectronics, Tsinghua University, Beijing 100084, PR China.

J. Wu is with the Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: Jie@cse.fau.edu).

Digital Object Identifier 10.1109/TR.2003.810071

[1]The singular and plural of an acronym are always spelled the same.

erant broadcasting for hypercube multi-computers, which is completely different from the techniques in [18], [19]. The definition of local safety is the same as in [18], [19]. An unsafe hypercube can be split into a unique set of maximal safe SC. Message-passing inside a maximal safe SC can be completed reliably. Several techniques are proposed to improve performance of the broadcast algorithm. Optimal broadcasting is still possible in many cases even though the hypercube is unsafe. The sufficient condition for optimal broadcast of a message is presented in an unsafe hypercube.

Section II provides definitions. Section III proposes a scheme to calculate local safety information to assist fault-tolerant communication, and presents properties of local safety information. Section IV presents techniques for improving performance of the fault-tolerant broadcasting algorithm. Section V presents a fault-tolerant broadcast algorithm according to local safety information. Section VI extends the method to the mixed fault model. Section VII presents extensive simulation results.

II. 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 $a_i \in \{0, 1\}$. An SC of a hypercube can be represented by a sequence of n bits $c_n, c_{n-1}, \ldots, c_2, c_1$, where $c_i \in \{0, 1, *\}$. Two nodes are connected by a bidirectional link if and only if the binary representations of the two nodes differ in exactly 1 bit. This paper considers only message-passing between fault-free nodes.

- A path is feasible if there is no faulty node or link in the path.
- A path is minimum if the length of the path equals the Hamming distance from the source to the destination.

This paper first considers only node faults; then it is extended to cases where the system contains both node and link faults.

Definition 1 [2], [15]: A fault-free node in an n-cube is unsafe if it has at least 2 faulty neighbors, or 3 unsafe or faulty neighbors. An unsafe-node is ordinarily unsafe if it has at least 1 safe neighbor; otherwise, it is strongly unsafe. A faulty hypercube is unsafe if it contains no safe-node; otherwise, it is a safe cube.

A broadcasting based on incomplete spanning binomial tree [15]–[17] is the following: 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 its fault-free neighbor along dimension i. This process is repeated until all 1-bits in the original broadcast label are reset.

Definition 2: The BSC of a node is an SC to which this node should broadcast the message.

The BSC at u can be derived by replacing certain bits of u's address by don't-cares. These bits correspond to 1-bits in the broadcast label. For example, let node 10100 in a 5-cube receive a broadcast label [11 010]. The BSC of node 10 100 is then **1*0.

III. Local Safety

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 at least n faulty nodes [2], [15]. Reliable broadcasting inside an unsafe n-cube is impossible according to the safe-node concept [2], [8], [15]. The safety level [16] and directed safety level [3] consider safety in the k-distance neighborhood.

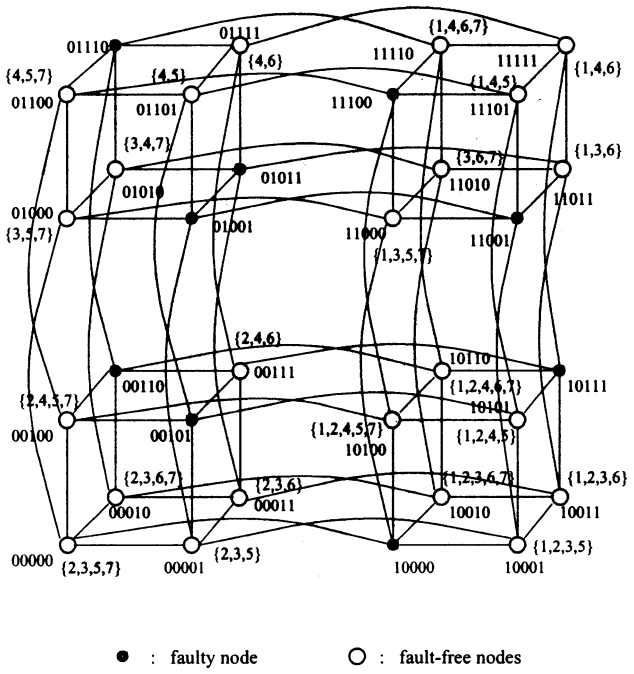

Fig. 1. Local safety for fault-tolerant broadcasting.

The definition of local safety is completely different from the refined safe-node concept presented in Definition 1. Local safety considers safety in a specific SC where the message of a node should be distributed, but the refined safe-node concept [2], [15] considers safety in the whole hypercube. Local safety has been used to guide fault-tolerant multi-casting [18] and routing [19]. However, techniques adopted in this paper are completely different from those in [18], [19]. The safety level in [16] and directed safety level in [3] consider safety in the k-distance neighborhood. A scheme to calculate local safety information is presented. And several properties of local safety are introduced.

A. Definition of Local Safety

Definition 3: A node in an n-cube is locally unsafe inside an SC if it has at least 2 faulty neighbors, or at least 3 locally unsafe or faulty neighbors inside the SC; otherwise, it is locally safe in the SC. The SC is unsafe if it contains no locally safe-node; otherwise, it is a safe SC. Locally unsafe-nodes inside an SC are classified as: A locally unsafe-node is locally ordinarily unsafe if it has at least 1 locally safe neighbor in the SC; otherwise, it is a locally strongly unsafe-node.

An SC can still be safe even though all nodes outside of it are faulty. Fig. 1 presents an unsafe 5-cube with 9 faulty nodes. Node 00 000 is locally safe in SC ****0, ***0*, and **0**. One node can have different local safety parameters in different SC.

Definition 4: An m-dimensional SC is a maximal safe SC if it is safe, and any k-dimensional ($k \geq m+1$) SC that contains it is unsafe.

The faulty 5-cube in Fig. 1 contains 7 4-dimensional maximal safe SC:

1****, *0***, **0**, **1**, ***0*, ***1*, ****0.

Fig. 1 shows that each fault-free node keeps the maximal safe SC that contain it. Labels of the MSC correspond to their sequence. Each node keeps local safety information of itself and its fault-free neighbors. A scheme to calculate local safety information is introduced first, and then properties of local safety are presented.

B. Calculation of Local Safety Information

This scheme is used to obtain local safety information for all nodes concurrently if the n-cube is unsafe. For each node v_n, v_{n-1}, $\ldots v_2$, v_1 (binary representation), check local safety of the node in $v_n * \cdots * *$, $*v_{n-1} * \cdots * *$, ..., $* * \cdots * v_2 *$, $* * * \cdots * v_1$ concurrently. If the node has at least 2 faulty neighbors or at least 3 locally unsafe or faulty neighbors inside an SC, then the node is locally unsafe in the SC. The node stores local safety information of its neighbors and itself if the SC has been found to be safe. When an $(n-1)$-dimensional SC that contains the node is found unsafe, local safety of all $2(n-2)$ SC that contains the node should be checked, and so on. This system does not consider local safety of SC that are contained in a maximal safe SC. This process continues until local safety of all maximal safe SC with sizes greater than the given limit has been obtained. More details on calculation of local safety information are in [17]–[19]. The faulty 5-cube in Fig. 1 contains 7 4-dimensional maximal safe SC: 1****(1), *0***(2), **0**(3), **1**(4), ***0*(5), ***1*(6), ****0(7); the numbers in the parentheses represent the corresponding labels of all maximal safe SC. Fig. 1 shows that each fault-free node keeps the maximal safe SC that contain it. Each node keeps local safety information of itself and its fault-free neighbors. The following lists present local safety information of 00 000 and its fault-free neighbors.

$00\,000^{(1)}$: $5(2)$, $5(3)$, $3(5)$

$00\,000^{(2)}$: $5(2)$, $5(3)$, $5(6)$, $5(7)$

$00\,000^{(3)}$: $3(2)$, $2(4)$, $5(5)$, $5(7)$

$00\,000^{(4)}$: $5(3)$, $5(5)$, $5(7)$

$00\,000^{(5)}$: 0

00 000: $5(2)$, $5(3)$, $5(5)$, $5(7)$.

Each item, $a(b)$, represents local safety information of the node in the maximal safe SC b is a. Values of a can be

$5 \equiv$ locally safe,

$3 \equiv$ locally ordinarily unsafe,

$1 \equiv$ locally strongly unsafe,

$0 \equiv$ faulty.

Each node has a label, as shown in Fig. 1, which indicates the labels of maximal safe SC that contain the node.

C. Properties of Local Safety

Let m faulty nodes be contained in an n-dimensional hypercube; the nodes are distributed randomly, *ie*, each fault-free node has the same probability to be the next faulty node. Theorem 1 presents a lower bound of the probability for an SC to be safe.

Theorem 1: Let $H = 2^h$, $N = 2^n$, and faulty nodes be distributed randomly inside a faulty n-cube. The probability for an h-dimensional SC to be safe is not less than P if the n-cube contains m faults,

$$P = \sum_{i=0}^{h-1} \frac{\binom{H}{i} \cdot \binom{N-H}{m-i}}{\binom{N}{m}}.$$

Property 1: If a node is locally unsafe in a k-dimensional SC, SC_1, it is still locally unsafe in an m-dimensional ($m > k$) SC, SC_2, if SC_2 contains SC_1.

Property 2: Let node s be locally safe in an m-dimensional SC, SC_2; then it is locally safe in a k-dimensional SC, SC_1, ($k < m$) if SC_2 contains SC_1.

Property 3: Let x be locally strongly unsafe in an SC, then there exists at least 1 locally ordinarily unsafe neighbor of x in the SC if the SC is safe.

Chiu [2] proved a strongly unsafe-node has at least 1 ordinarily unsafe neighbor in a safe hypercube. Property 3 can be extended easily.

Property 4: There always exists a minimum feasible path between 2 fault-free nodes x and y if the SC (x, y) is safe even though the hypercube is unsafe.

Property 5: A minimum feasible path between the source s and destination d, is available using local safety of the MSC that contains the SC (s, d) if one of s and d is locally safe in MSC even though the hypercube is unsafe.

There might still exist a feasible path of length no more than $H(s, d) + 4$ even if SC (s, d) is unsafe, where SC (s, d) is contained in an MSC.

Property 6: A feasible path of length no more than $H(s, d)+4$ between the source s and the destination d using local safety information of an MSC that contains the SC (s, d) if both s and d are locally unsafe in the MSC even if the hypercube and SC (s, d) are unsafe.

There might still exist a minimum feasible path between the source and destination although the conditions in Property 5 are not met; for simplicity, this is not stated in this paper. Properties 5 and 6 can be extended from [2], detailed proofs of which are in [17]–[19].

IV. Techniques for Improving Performance of Fault-Tolerant Broadcasting

Broadcasting in a fault-free hypercube can be completed easily and systematically. The regular structure for broadcasting in a faulty hypercube can be destroyed. The number of steps required to broadcast a message depends on the relative fault-positions and the source node. Most broadcast algorithms pass messages by attaching a broadcast label [3], [7], [8], [15]–[17], which indicates the area that the message from the node should be distributed in. The source s has a label with all bits assigned the value 1. When the message is passed to the next node along dimension i, the bit i of the broadcast label

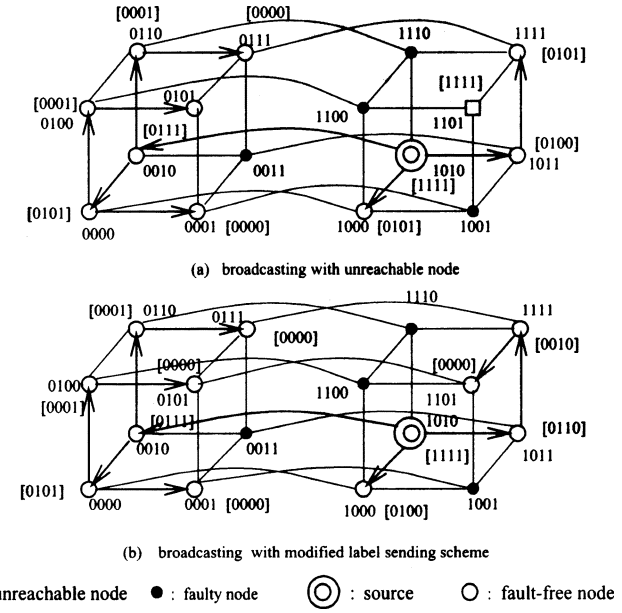

Fig. 2. Avoid forwarding message and label to nodes with at least 2 faulty neighbors in the BSC.

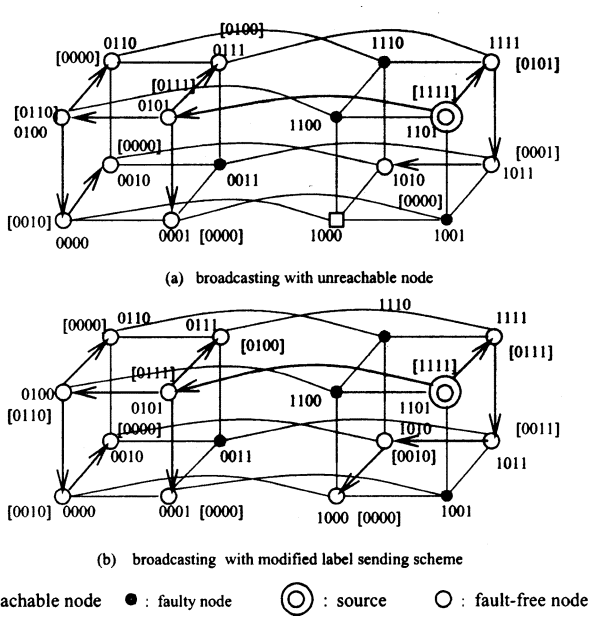

Fig. 3. Fault-tolerant broadcasting using the modified label-sending scheme.

is reset, which is also sent to $s^{(i)}$. The most important thing is how to determine the priority order to forward the message from each node in a broadcast algorithm.

Lemma 1: If a node has at least 2 faulty neighbors in the BSC, the broadcasting message starting from that node cannot always be sent along the minimum feasible paths.

Proof: It is clear that the distance between d and the source s is 2, where d is connected with both faulty nodes. There exists no minimum feasible path between s and d because both minimum feasible paths are blocked by faulty nodes. ◁

Let each node keep local safety-information of itself and its neighbors. Consider a message being broadcast with the source 1101 inside the faulty hypercube as shown in Fig. 2(a). Because node 1000 receives a broadcast label [0101], according to which its BSC should be 1*0*. The node 1000 has 2 faulty neighbors, 1100 and 1001, in the BSC; therefore, the message from 1000 is unable to reach 1101. The message is unable to be broadcast successfully according to algorithms in [8], [15]. Actually, 1101 is unreachable from 1000 using the label-sending scheme. The following procedure modifies the label-sending broadcast scheme in order to make the message reach all fault-free nodes when the BSC of a node is contained in an MSC. The procedure adopts the following 2 efficient schemes:

Scheme 1. Try to avoid sending the broadcast label and message to fault-free neighbors which have at least 2 faulty neighbors in the BSC.

Scheme 2. If the source has at least 2 faulty neighbors in the BSC, then send the broadcast label to the last fault-free neighbor along dimension i without resetting the bit i.

The node receiving the unmodified label does not send the message back to its predecessor. A fault-free node has knowledge of status of its neighbors. In order to implement scheme 1, it is reasonable for each node s to keep an $n \times n$ matrix F to record its faulty neighbors. $F(i, j) = 1$ if the neighbor of $s^{(i)}$ along dimension j is faulty; otherwise, $F(i, j) = 0$. The following matrix records the faulty neighbors of node 00 000 in the 5-cube as given in Fig. 1.

$$F_{00\,000} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The scheme to avoid sending the message to nodes which have at least 2 faulty neighbors in the BSC is quite useful. Consider the broadcast problem with the source 1010 in the faulty hypercube 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 BSC 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 [2], [15]. This makes the message not reach 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 2 faulty neighbors in its BSC, which makes the message reach all nodes along minimum feasible paths as shown in Fig. 2(b). The reason why this scheme can reach node 1101 as shown in Fig. 2(b), but the procedure in [15] cannot, is that an ordinarily unsafe-node in the hypercube can be locally strongly unsafe in an SC, while a strongly unsafe node in the hypercube can be locally safe in the SC.

Scheme 2 is important because it makes the message deroute inside some small SC. The total broadcast steps can still be no more than the size of the hypercube according to this scheme, although the message is not broadcast optimally. Let a message be broadcast from 1101 based on techniques in [8], [15] as shown in Fig. 3(a). 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 fault-free nodes inside its BSC as shown in Fig. 3(a). The node 1101 then sends the message to 1111 with a

label [0101]. It is clear that the message cannot reach 1000 based on techniques in [8], [15]. As shown in Fig. 3(b), 1101 sends the label [0111] to the node 0101, and sends the label [0111] to node 1111 (the last processed fault-free neighbor of 1101) without resetting bit #2. The node 1111 does not send the message back to 1101. Up to now, node 1000 is reachable from the source although the message is not broadcast optimally. The message at the source 1101 can be broadcast in 4 steps although it is derouted inside the 3-dimensional SC 1***. These techniques can be used to improve performance of a broadcast algorithm when the BSC is contained in a maximal safe SC.

Procedure: Broadcast1(s)/* Fault-tolerant broadcasting in a safe SC MSC */

1. Let s be the broadcast source; set the broadcast label of s as $[11\cdots11]$. If s has no more than 1 faulty neighbors in the BSC, then 2∼5, otherwise 6.
2. For $i = 1$ to n

a) if label$[i] = 1$, and node $s^{(i)}$ is locally safe, then (b);
b) label$[i] \leftarrow 0$, send the message and label via dimension i.

3. For $i = 1$ to n

a) if label$[i] = 1$ and node $s^{(i)}$ is locally ordinarily unsafe, then (b),
b) if the node $s^{(i)}$ has at most 1 faulty neighbor inside the BSC, then (c);
c) label$[i] \leftarrow 0$, send the message and label via dimension i.

4. For $i = 1$ to n

a) if label$[i] = 1$ and node $s^{(i)}$ is locally strongly unsafe, then (b),
b) if node $s^{(i)}$ has at most 1 faulty neighbor inside the BSC, then (c);
c) label$[i] \leftarrow 0$, send the message and label to node $s^{(i)}$ via dimension i.

5. For $i = 1$ to n

a) if label$[i] = 1$, then (b);
b) label$[i] \leftarrow 0$, send the message and label to node $s^{(i)}$ via dimension i.

6. Do the same steps as $2 \sim 5$; each time check only whether the node $s^{(i)}$ is the latest unprocessed fault-free neighbor of s. If it is not, send the message and label by resetting label$[i]$; if node $s^{(i)}$ is the last unprocessed fault-free neighbor of s, send the message and label to node $s^{(i)}$ without resetting label$[i]$.

The node $s^{(i)}$ in Procedure broadcast1() is the neighbor of s along dimension i. This procedure supports n-port broadcasting in an n-cube, *ie*, a message can be broadcast to its neighbors concurrently. When s has at least 2 faulty neighbors, then send the message and the broadcast label without resetting label$[i]$ to the latest unprocessed fault-free neighbor $s^{(i)}$.

Theorem 2: The procedure broadcast1() can always optimally pass a message if node s is locally safe in an MSC that contains the BSC of the node s.

Theorem 3: The procedure broadcast1() always broadcasts a message to all fault-free nodes in its BSC at most once no matter whether the message can be successfully broadcast or not.

Proof: Each fault-free neighbor $s^{(i)}$ of s passes the message inside its own BSC if it receives a broadcast label with the bit i reset. Let the fault-free neighbor $s^{(j)}$ receive the unmodified label, it will not send the message back to s because the BSC of $s^{(j)}$ does not contain any feasible path going back to s. The situations for all $s^{(i)}$ and $s^{(j)}$ are similar. Thus, any visited node by the procedure broadcast1 (s) will never be revisited. ◁

There still exist some cases where broadcast1() cannot find a successful broadcast, although there exists an MSC that contains the BSC. For example, the procedure broadcast1() cannot successfully broadcast a message if the BSC of one of the source's neighbors gets a disconnected BSC. However, the procedure broadcast1() can broadcast a message successfully inside its BSC in most cases if BSC is contained in a maximal safe SC.

V. Fault-Tolerant Broadcasting via Local Safety Information

Assume each fault-free node keeps local safety of its fault-free neighbors and itself by using the scheme introduced in Section III. Optimal broadcasting is still possible in many cases even though the hypercube is unsafe. This section presents:

- the sufficient condition for the existence of an optimal broadcasting,
- a fault-tolerant broadcast algorithm,
- the sufficient condition for optimal broadcasting by the Algorithm.

A. Sufficient Condition for Existence of Optimal Broadcasting

Theorem 4: An optimal broadcast exists from a node s if s is locally safe in its BSC.

Theorem 4 implies that a message can be broadcast optimally even though the n-cube is unsafe. Construct several BSC starting from the source. Consider the source has at most 1 faulty neighbor in the n-cube. Let Q_{n-1}, Q_{n-2}, …, Q_{n-m} be BSC of the fault-free neighbors, $s^{(i_1)}$, $s^{(i_2)}$, …, $s^{(i_m)}$ $(n-1 \leq m \leq n)$ of the source, where $i_1, i_2, \ldots, i_m \in \{1, 2, \ldots, n\}$; the subscripts indicate the sizes of the corresponding BSC. Theorem 5 presents the sufficient condition for existence of an optimal broadcasting inside an unsafe n-cube.

Theorem 5: There exists an optimal broadcasting 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 BSC, respectively, even though the n-cube is unsafe.

Proof: There exists an optimal broadcasting from $s^{(i_1)}$ in the BSC Q_{n-1} because $s^{(i_1)}$ is locally safe in Q_{n-1} according to theorem 4. There exists an optimal broadcasting from $s^{(i_2)}$ in the BSC Q_{n-2} because $s^{(i_2)}$ is locally safe in Q_{n-2}, and similar cases for other BSC. Therefore, there exists an optimal broadcasting from the source s even though the n-cube is unsafe. ◁

B. Algorithm

Algorithm `broadcast2()` broadcasts a message using local safety information. When the n-cube is unsafe, `broadcast2()` tries to find a fault-free neighbor $s^{(i)}$ of s along dimension i, whose BSC is contained in a maximal safe SC. The message can be broadcast reliably if the BSC is contained in a maximal safe SC in many cases.

Let size (k) be the size of the maximal safe SC k that contains the node v, in which the local safety information of the node is safety (v, k);

- 5 for locally safe,
- 3 for locally ordinarily unsafe,
- 2 for locally strongly unsafe,
- 0 for faulty.

The safety measure is calculated using:

$$\text{safe}\,(v) = \text{size}\,(k) \cdot \text{safety}\,(v, k).$$

Algorithm broadcast2() /* Broadcasting via Local Safety Information */
1. If node s is the broadcast source, for $i = 1$ to n, label$[i] \leftarrow 1$; do 2, 3, 4;
2. If the BSC of s is contained in a maximal safe SC MSC, then call broadcast1 (s) based on the local safety information of MSC; otherwise $f_{\text{br}} \leftarrow 0$; if s has at least two faulty neighbors in its BSC, then 5, otherwise 3, 4.
3. While $f_{\text{br}} = 0$, do
   a. $f_{\text{br}} \leftarrow 1$, for $i = 1$ to n
      i. if label$[i] = 1$ and the BSC of $s^{(i)}$ is contained in a maximal safe SC, in which $s^{(i)}$ is locally safe, then ii.
      ii. label$[i] \leftarrow 0$, send the message and label to $s^{(i)}$; call broadcast1($s^{(i)}$); $f_{\text{br}} \leftarrow 0$.
   b. $f_{\text{br}} \leftarrow 1$, for $i = 1$ to n
      i. if label$[i] = 1$ and the BSC of $s^{(i)}$ is contained in a maximal safe SC, and $s^{(i)}$ has at most 1 faulty neighbor inside the BSC, then (ii),
      ii. label$[i] \leftarrow 0$, send the message and the label to $s^{(i)}$; call broadcast1($s^{(i)}$); $f_{\text{br}} \leftarrow 0$.
   c. $f_{\text{br}} \leftarrow 1$, for $i = 1$ to n
      i. if label$[i] = 1$ and the BSC of $s^{(i)}$ is contained in a maximal safe SC, then (ii),
      ii. label$[i] \leftarrow 0$, send the message and label to $s^{(i)}$; call broadcast1($s^{(i)}$); $f_{\text{br}} \leftarrow 0$.
4. If there still exists at least 1 fault-free neighbor of s, which has not received the message and broadcast label, for each $label[i] = 1$ $(1 \le i \le n)$
   a. if $s^{(i)}$ is fault-free and has the most safety measure as given in 1) [$s^{(i)}$ has greater priority if it has at most 1 faulty neighbor inside its BSC], then (b);
   b. label$[i] \leftarrow 0$; send the message, label to $s^{(i)}$ via dimension i; $f_{\text{br}} \leftarrow 0$, go to step 3.
5. Do the same steps as 3,4, each time; but check whether the 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$[i]$.
End Algorithm

A flag f_{br} is adopted to guide whether the broadcast should be continued or not; it is set at 0 initially. The way to generate the BSC of $s^{(i)}$ in step 3 is: For example, the node $10\,101(v)$ has a broadcast label [11 011], the BSC of 00 101 ($v^{(5)}$) should be 0*1**, while the BSC of 10 100 ($v^{(1)}$) is **1*0.

Theorem 6 presents a sufficient condition for optimal broadcasting inside an unsafe n-cube.

Theorem 6: The algorithm broadcast2() can optimally broadcast the message of the source s in n steps if s has at most 1 faulty neighbor and a sequence of fault-free neighbors $s^{(i_1)}$, $s^{(i_2)}$, ..., $s^{(i_m)}$ $(n - 1 \le m \le n)$ are locally safe in a sequence of maximal safe SC which contain the BSC of the nodes $s^{(i_1)}$, $s^{(i_2)}$, ..., $s^{(i_m)}$ respectively, where $i_1, i_2, \ldots, i_m \in \{1, 2, \ldots, n\}$.

Proof: Assume the BSC of $s^{(i_1)}$, $s^{(i_2)}$, ..., $s^{(i_m)}$ are contained in maximal SC $\text{MSC}_1, \text{MSC}_2, \ldots, \text{MSC}_m$, in which $s^{(i_1)}$, $s^{(i_2)}$, ..., $s^{(i_m)}$ are locally safe. The message of $s^{(i_1)}$, $s^{(i_2)}$, ..., $s^{(i_m)}$ can be broadcast optimally inside their BSC according to local safety information of the maximal safe SC MSC_1, MSC_2, ..., MSC_m respectively, as illustrated in theorem 2. ◁

The algorithm broadcast2() can still find a successful broadcast in many cases even though the conditions in theorem 6 are not met.

C. A Case Study

This example illustrates how the algorithm broadcast2() works. Fig. 4 presents the same faulty 5-cube with 9 faulty nodes as in Fig. 1, which is unsafe. There are 7 4-dimensional MSC in the faulty 5-cube: 1****, *0***, **0**, **1**, ***0*, ***1*, ****0. Consider broadcasting from the source 00 010. The algorithm broadcast2() sends the message and the label [11 101] to 00 000 from the source, because 00000 is locally safe in the MSC ***0* (step 3a). The message can be broadcast optimally inside the SC ***0* according to theorem 2 using broadcast1(). The source then sends the message and the label [11 100] to 00 011 from the source. The BSC of node 00 011 is ***11, which is contained in the MSC ***1*, and 00 011 is locally safe in ***1* (step 3a). The message can thus be broadcast optimally in ***11 according to theorem 2. The message and the label [10 100] are sent to 01010, and the BSC of 01 010 is *1*10, which is contained in the MSC ****0. Node 01 010 is locally safe inside ****0, in which the message can be broadcast optimally according to theorem 2 (step 3a). After these processes, the message and label [00 100] are sent to 10 010. There is only 1 fault-free node inside 10*10. Therefore, the message from 00010 can be broadcast optimally in the unsafe 5-cube in Fig. 4. The message is broadcast optimally along minimum feasible paths, which is quite compatible with theorem 6.

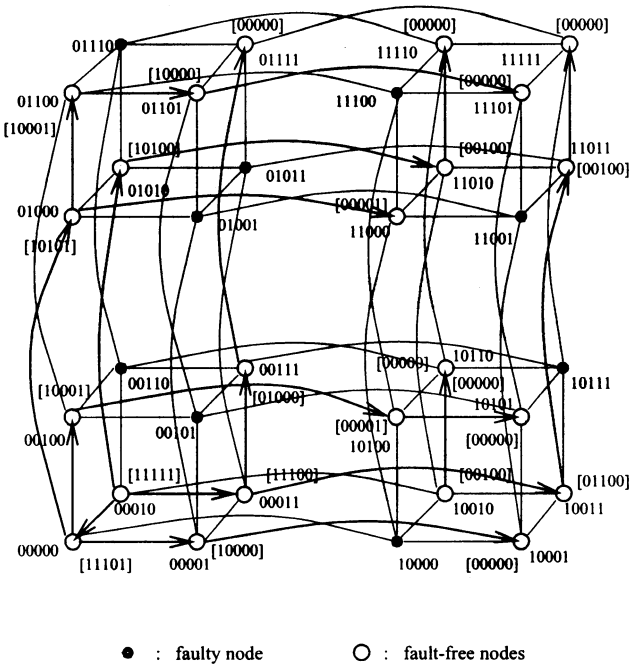

Fig. 4. Optimal fault-tolerant broadcasting with local safety.

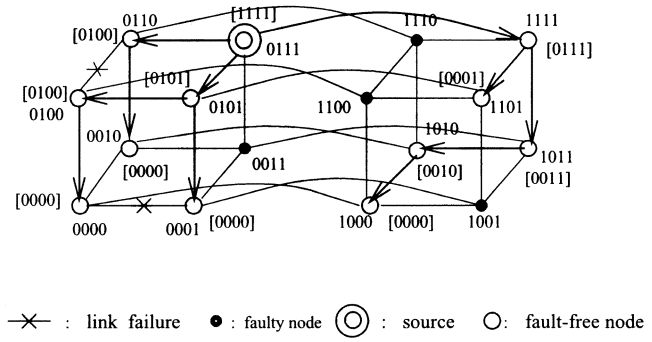

Fig. 5. Fault-tolerant broadcasting in hypercubes with node and link faults.

Actually, a message can be nonredundantly broadcast to all fault-free nodes in no more than 7 steps with the source being any one of the fault-free nodes in the faulty 5-cube, as in Fig. 4. The message can be broadcast optimally if the source is any one of 00000, 00010, 00011, 10 011.

VI. Extension to Reliable Broadcasting for Hypercubes With Both Node and Link Faults

This section shows that it is quite easy to extend the local-safety-based broadcast procedure to the case when the hypercube system contains node and link faults. The faulty 4-cube in Fig. 5 contains 4 faulty nodes, 0011, 1100, 1110, 1001, and 2 link failures 000- and 01-0 (000- indicates the link that connects nodes 0000 and 0001). While the safety of a hypercube system is identified, the following schemes like those in [2] are adopted:

1. The end nodes of a link failure are considered as faulty nodes.

2. The end nodes of a link failure are considered as unsafe after safety information of the hypercube system has been determined.

These schemes are different from the ones in [2] because the method in this paper considers safety inside SC. Any link and node faults outside an SC do not have any influence on safety of the SC. The scheme to identify states of nodes in a faulty hypercube is similar to that in Section III. Nodes 0110, 0100, 0000, 0001 are thought of as faulty, in order to determine safety of the 4-cube in Fig. 5. The 4-cube is fully unsafe. Now local safety of the 4-cube can be determined with the schemes introduced in Section III. The faulty nodes or link failures are not considered when they are not contained in the SC under consideration. Consider the local safety of the SC ***0, nodes 0000, 0010, 1000, 1010 are locally safe, while nodes 0100, 0110 are locally ordinarily unsafe. The following MSC exists in the fully unsafe 4-cube in Fig. 5: 1***, *1**, **1*, ***0, ***1, 0*0*. Theorems 2∼6 and Properties 1∼7 still hold when the mixed fault model is considered. A message should never be routed to a node whose shortest paths leading to the destination are blocked by faulty links.

The message can be routed to a locally safe node in the MSC if the source s is locally safe in the MSC, and $H(s, d) \geq 3$ (d is the destination). The message should be routed to a fault-free neighbor whose link leading to d is not blocked by a link failure in a minimum path from the source s to d if $H(s, d) = 2$. To implement this scheme, each node keeps fault information (including faulty node and link failure) of its fault-free neighbors just like the scheme in Section IV. The schemes in Section IV should be modified as follows:

1. Try to avoid sending the message and the broadcast label to fault-free neighbors which have at least 2 faulty neighbors or are connected with a faulty link in the BSC.

2. If the source has 1 connected faulty link or at least 2 faulty neighbors inside the BSC, send the broadcast label to the last fault-free neighbor along dimension i inside the BSC without resetting the bit i.

The node receives the unmodified label: do not send the message back to its predecessor. The algorithm broadcast2() still works with slight modification by combining these modified schemes. Also, the safety measure in 1) is adopted to forward the broadcast message when the condition in theorem 6 is not satisfied.

Consider the broadcast problem with the source 0111 in Fig. 5. Node 0111 sends the message and the broadcast label [0111] to 1111, because 1111 is locally safe in the MSC 1***. The message of 1111 can be optimally passed inside 1***. The source 0111 can send a label [0101] to 0101 or a label [0110] to 0110. The broadcast label [0110] cannot be sent to 0110, which might cause node 0110 to be connected with a faulty link in the BSC according to the modified scheme illustrated in Section IV. Node 0111 should send the label [0101] to node 0101. The message can be optimally passed inside 0*0* because 0101 is locally safe in the MSC 0*0*. Node 0101 can send the message to 0001 with a label [0001], or to 0100 with a label [0100]. Let 0101 send the message with the label [0001] to 0001. The message cannot be passed to 0000 in this case. Node 0101 should not send the message with the label [0001] to

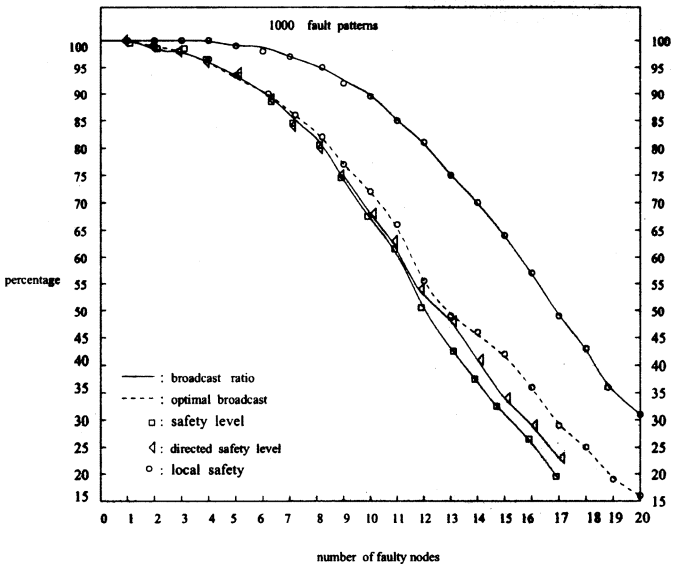

Fig. 6. Performance evaluation in the 6-cubes.

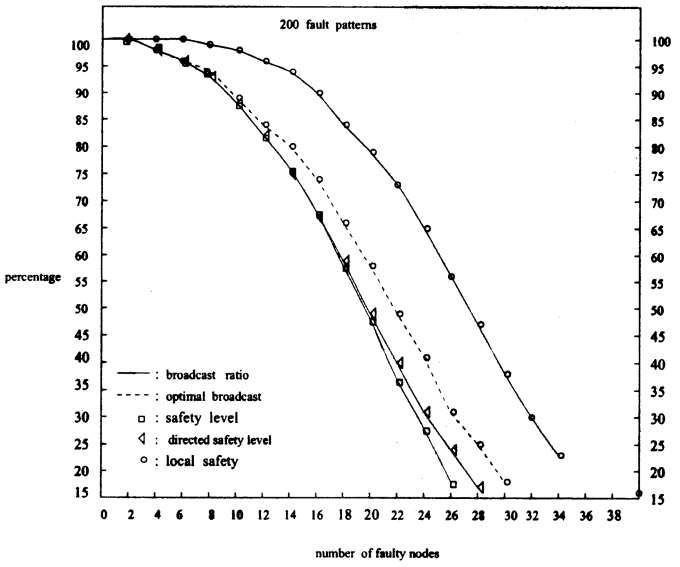

Fig. 7. Performance evaluation in the 7-cubes.

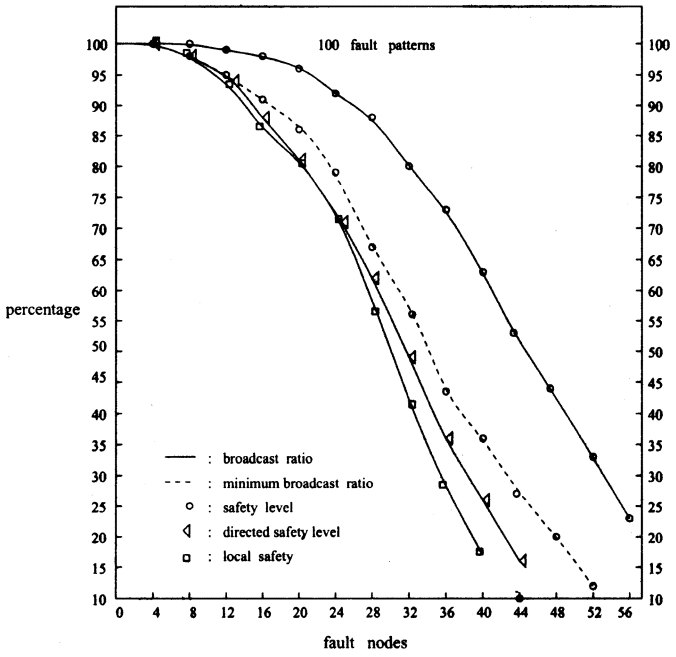

Fig. 8. Performance evaluation in the 8-cubes.

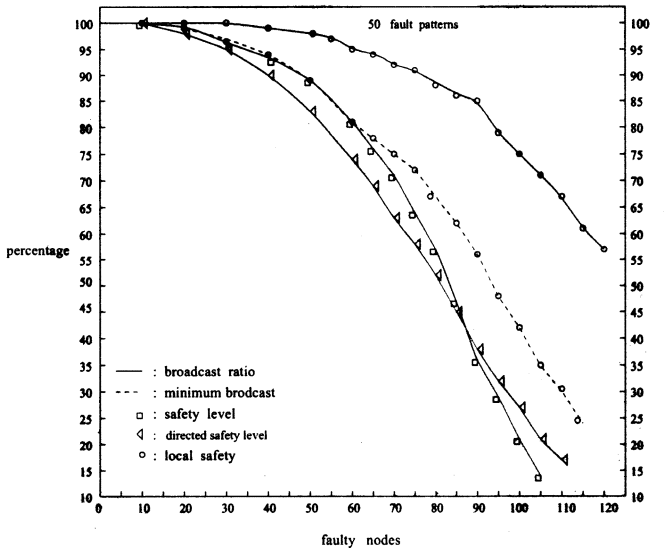

Fig. 9. Performance evaluation in the 10-cubes.

0001 because 0001 is connected with a faulty link 000- inside the BSC according to the scheme introduced in this section. The node should send the message with the label [0100] to node 0100. Fig. 5 presents the time-optimal broadcast result.

VII. Simulation Results

Two types of experiments have been done. Type #1 selects each of the fault-free nodes as the broadcast source, which obtains broadcast ratio (fraction of nodes can broadcast a message to all other fault-free nodes) and minimum broadcast ratio (fraction of sources broadcast a message along the minimum paths). Type #2 are flit-level simulations, which presents latency, throughput, broadcast ratio, and minimum broadcast ratio under various conditions. A flit-level simulator is an event-driven one, which emulates wormhole-routed systems with respect to message-passing, deadlock avoidance, *etc*. These conditions include the number of faulty nodes, the message length, and load rate. All simulation results for the flit-level simulators are obtained for systems in a centralized environment [1].

Figs. 6–9 present type #1 of simulation results on 6-cube, 7-cube, 8-cube, 10-cube by comparing local safety with safety level [16] and directed safety level [3], respectively. Safety level and directed safety level present only minimum broadcast ratio. However, minimum broadcast ratio based on local safety is also better than broadcast ratio according to safety level and directed safety level in almost all cases. Local safety obtains up to 60% broadcast ratio more than safety level. It is observed that the difference between two metrics is even more obvious as the size of the system increases.

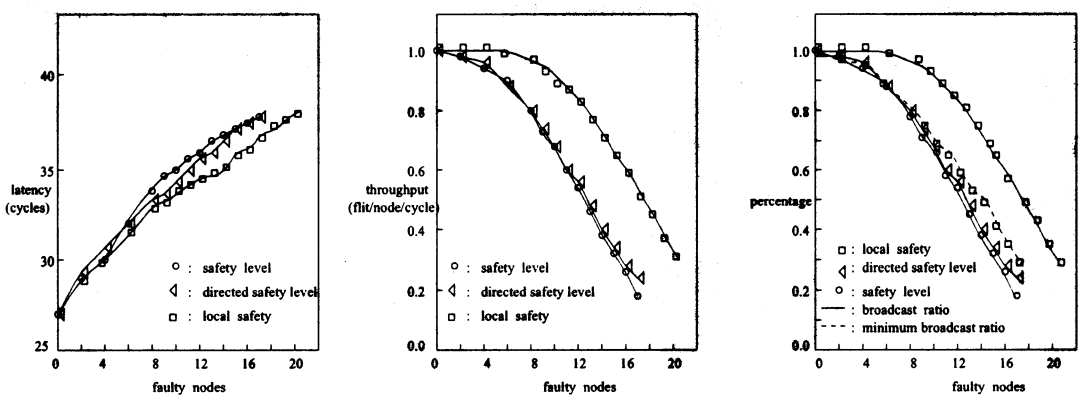

Fig. 10. Performance comparison in the 6-cubes.

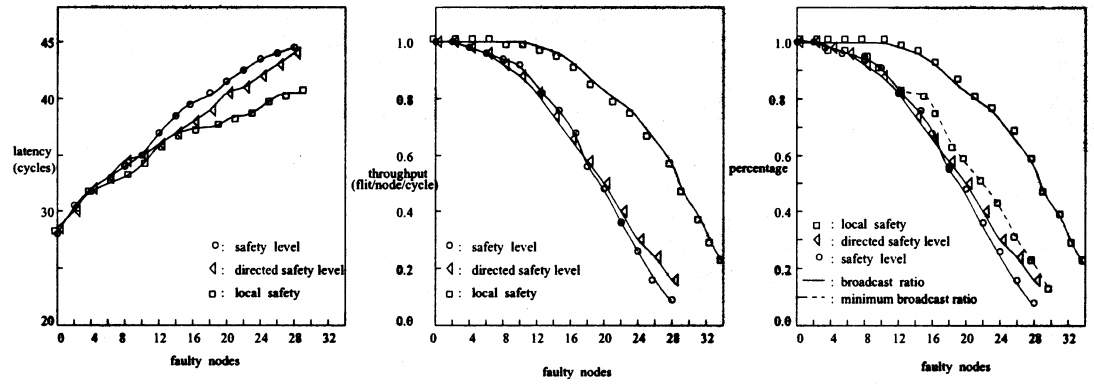

Fig. 11. Performance comparison in the 7-cubes.

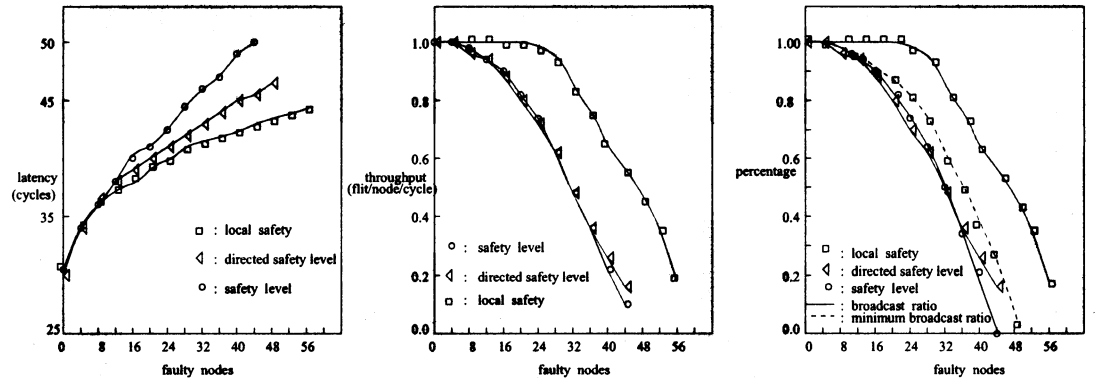

Fig. 12. Performance comparison in the 8-cubes.

Figs. 10 through 13 present flit-level performance evaluation of local safety, safety level, and directed safety level. Message length is 16 flits, load rate (flit/node/cycle data inserted) is set at 1.0, and buffer size is 64 flits for each node. The results are average ones of various fault patterns. Results of each pattern are obtained by running the system 30 000 cycles, where the start-up cycles (the first 10 000 cycles) are not included. Figs. 10–13 present latency, throughput, broadcast ratio and minimum broadcast-ratio comparison between local safety and the 2 metrics. Throughput is obtained by using:

$$\text{throughput} = \frac{A \cdot B}{C \cdot D}$$

$A \equiv$ number of delivered messages
$B \equiv$ message length
$C \equiv$ cycles
$D \equiv$ number of fault-free nodes.

Local safety consistently obtains better results than safety level and directed safety level on latency to broadcast a message. Latency of a message based on safety level and directed safety level is more than that of local safety in all cases. The difference of latency between local safety and the two metrics becomes greater as the number of faults increases in a system.

- Safety level needs at most 2 cycles more than local safety to broadcast a message in the 6-cube.

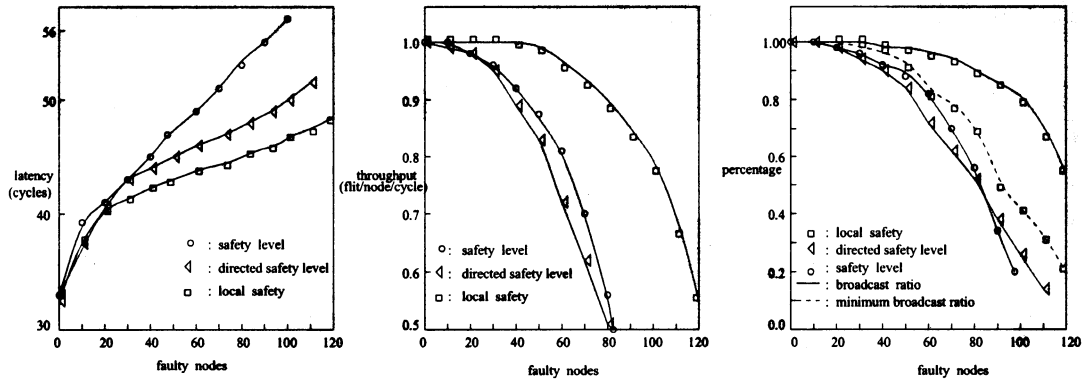

Fig. 13. Performance comparison in the 10-cubes.

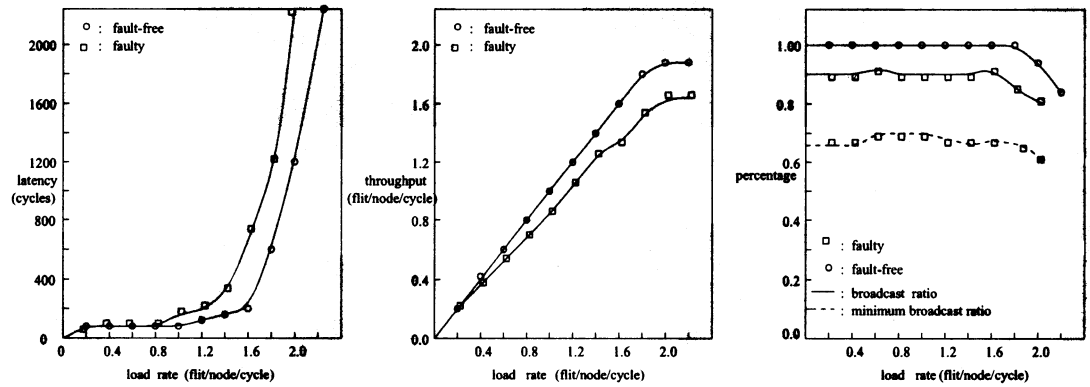

Fig. 14. Performance evaluation of 10-cube with various load rates.

- Safety level needs up to 4 cycles more than local safety to broadcast a message in the 7-cube.
- Up to 7 more cycles is required for safety level to broadcast a message than local safety in the 8-cube.
- up to 10 more cycles is required for safety level than local safety to broadcast a message in the 10-cube.

Generally, latency of directed safety level is between local safety and safety level. Latency differences between local safety and directed safety level reach up to

- 2 cycles in 6-cubes,
- 3 cycles in 7-cubes,
- 3 cycles in 8-cubes,
- 4 cycles in 10-cubes.

Local safety consistently obtains better results than safety level on throughput to broadcast a message. Let the load rate be 1.0. Throughputs of safety level and local safety in the 6-cube are 0.183 and 0.494, respectively when the system contains 20 faulty nodes. Throughputs of safety level and local safety in the 7-cube are 0.1 and 0.472, respectively when the system contains 28 faulty nodes. For the 8-cube, throughput difference between two metrics is up to 0.43 when the system contains 44 faults. Throughputs for local safety and safety level are 0.769 and 0.220 when the 10-cube contains 100 faults. Directed safety level obtains a little better throughput than safety level in the 6-cube, 7-cube, and 8-cube in almost all cases, but a bit worse throughput than safety level in the 10-cube. Local safety gets better throughput than directed safety level in all cases.

Local safety also obtains better broadcast ratio and minimum broadcast ratio in all cases than safety level. Broadcast ratio differences for both metrics reach

- 31% for the 6-cubes,
- 36% for the 7-cubes,
- 43% for the 8-cubes,
- 54.9% for the 10-cubes.

Minimum broadcast ratio differences between two metrics are up to

10.0% for the 6-cubes,
14.4% for the 7-cubes,
17.1% for the 8-cubes,
22.5% for the 10-cubes.

The broadcast ratio and minimum broadcast ratio differences between local safety and directed safety level reach

26% and 5% for the 6-cubes,
32% and 9% for the 7-cubes,
34% and 9% for the 8-cubes,
57% and 21% for the 10-cubes.

The difference between performance of local safety and the other 2 metrics is even clearer when the size of the system increases.

Fig. 14 presents performance of local safety when the message length is 64 flits and buffer size is 256 flits when the system has various load rates. The latency of a broadcast message increases drastically when load rate reaches 1.4 for the faulty

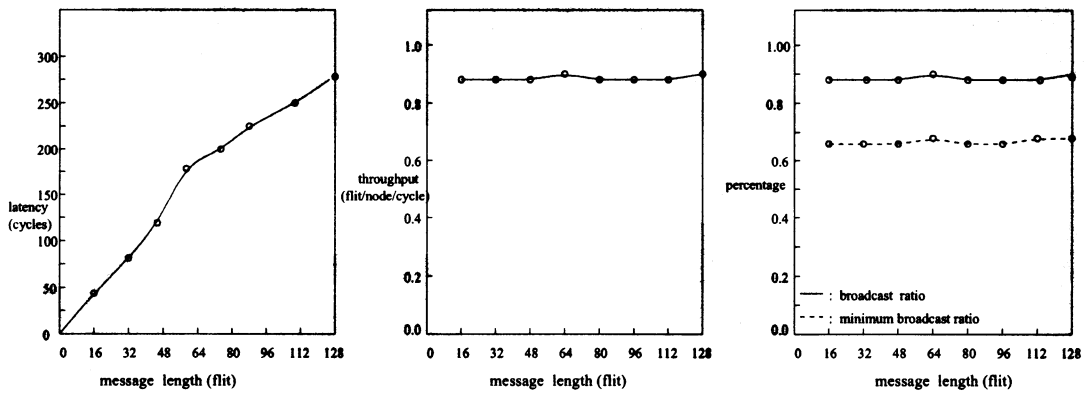

Fig. 15. Performance evaluation of 10-cube with various message lengths.

10-cube with 80 faulty nodes. For the fault-free 10-cube, latency increases greatly when the load rate is about 1.6.

Fig. 15 presents performance of local safety for messages of various sizes in a faulty 10-cube with 80 faulty nodes. The throughput and broadcast ratio of the system are not sensitive to the size of the messages.

APPENDIX
PROOFS OF SOME THEOREMS

A. Proof of Theorem 1

The probability for the h-dimensional SC to contain i of the m faults is ($i \leq h-1$),

$$P_h(i, m) = \frac{\binom{H}{i} \cdot \binom{N-H}{m-i}}{\binom{N}{m}}.$$

The h-dimensional SC contains no more than $h-1$ faults which are separate events. Therefore, the probability P for the SC to contain no more than $h-1$ faults is,

$$P = p_h(0, m) + p_h(1, m) + \cdots + p_h(h-1, m);$$

$$P = \sum_{i=0}^{h-1} \frac{\binom{H}{i} \cdot \binom{N-H}{m-i}}{\binom{N}{m}}.$$

The SC might still be safe even though it contains greater than $h-1$ faulty nodes [2]. That is, this equation presents a lower bound of the probability for an h-dimensional SC to be safe. ◁

B. Proof of Theorem 2

The theorem is proved by induction of the size of the node's BSC. Let the size of the BSC of s be 2. Node s has at most 1 faulty neighbor according to Definition 2; therefore, broadcast1() can optimally pass the message by using the local safety information of the MSC in this case.

Assume the theorem holds when the size of the BSC of node s is k ($k > 2$). The theorem also holds when the size of the BSC of s is $k+1$. The node s has at least 1 safe neighbor $s^{(i)}$ inside its BSC along a dimension i (where i is the least important dimension to meet the above conditions). The procedure broadcast1() passes the message and the broadcast label by resetting label$[i]$. The size of the BSC of $s^{(i)}$ is k, therefore, the message from $s^{(i)}$ can be passed along minimum feasible paths according to the assumption. The size of the BSC of s is reduced to k, where s has the same broadcast label as the node $s^{(i)}$. The message of node s can also be passed optimally by broadcast1() according to the assumption. ◁

C. Proof of Theorem 4

The theorem is proved by induction of the size of the BSC. Let the size of the BSC be 2. At most, 1 neighbor of s inside the BSC is faulty according to Definition 3; the theorem clearly holds in this case.

Let the theorem hold when the size of the BSC is k ($k \geq 3$), then the theorem also holds when the size of the BSC is $k+1$. The message and the label of the source s can be sent to a locally safe neighbor $s^{(i)}$ of s [$s^{(i)}$ is always available according to Definition 3] inside BSC by resetting label$[i]$. Therefore, s and $s^{(i)}$ have new BSC: BSC_1 and BSC_2 of size k, inside which, s and $s^{(i)}$ are locally safe according to Property 2, respectively. The message of s and $s^{(i)}$ can thus be broadcast optimally inside BSC_1 and BSC_2, respectively, according to the assumption. ◁

REFERENCES

[1] R. V. Boppana and S. Chalasani, "A framework for designing deadlock-free wormhole routing algorithms," *IEEE Trans. Parallel and Distributed Syst.*, vol. 7, no. 2, pp. 169–183, 1996.

[2] G. M. Chiu and P. S. Wu, "A fault-tolerant routing strategy in hypercube systems," *IEEE Trans. Comput.*, vol. 45, no. 2, pp. 143–155, 1996.

[3] G. M. Chiu, "Fault-tolerant broadcasting algorithm for hypercubes," *Inform. Proc. Lett.*, vol. 66, no. 2, pp. 93–99, Apr. 1998.

[4] J. Duato, "A theory of fault-tolerant routing in wormhole networks," *IEEE Trans. Parallel and Distributed Syst.*, vol. 8, no. 8, pp. 790–802, 1997.

[5] J. Duato and M. P. Malumbres, "Optimal topology for distributed shared-memory multiprocessors: Hypercubes again?," in *Proc. 2nd Int. Euro-Par Conf.*, 1996, pp. 205–212.

[6] S. L. Johnsson and C. T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Comput.*, vol. 38, no. 9, pp. 1249–1268, 1989.

[7] H. P. Katseff, "Incomplete hypercubes," *IEEE Trans. Comput.*, vol. 37, no. 5, pp. 604–608, 1988.

[8] T. C. Lee and J. P. Hayes, "A fault-tolerant communication scheme for hypercube computers," *IEEE Trans. Comput.*, vol. 41, no. 10, pp. 1242–1256, 1992.
[9] J. Laudon and D. Lenoski, "The SGI origin: A ccNUMA highly scalable server," in *Proc. Int. Symp. Computer Architecture*, 1997, pp. 241–251.
[10] M. Ould-Khaoua, "On optimal network for multicomputers: Torus or hypercube?," in *Proc. 4th Int. Euro-Par Conf.*, 1998, pp. 989–992.
[11] S. Park and B. Bose, "All-to-all broadcasting in faulty hypercubes," *IEEE Trans. Comput.*, vol. 46, no. 7, pp. 749–755, 1997.
[12] C. S. Raghavendra, P. J. Yang, and S. B. Tien, "Free dimensions—An effective approach to achieving fault tolerance in hypercubes," *IEEE Trans. Comput.*, vol. 44, no. 9, pp. 1152–1157, 1995.
[13] P. Ramanathan and K. G. Shin, "Reliable broadcast in hypercube multicomputers," *IEEE Trans. Comput.*, vol. 37, no. 12, pp. 1654–1657, 1988.
[14] C. L. Seitz, "The cosmic cube," *Commun. ACM*, vol. 28, no. 1, pp. 22–33, 1985.
[15] J. Wu and E. B. Fernandez, "Reliable broadcasting in faulty hypercube computers," *Microprocessing and Microprogramming*, vol. 46, pp. 241–247, 1993.
[16] J. Wu, "Safety levels—An efficient mechanism for achieving reliable broadcasting in hypercubes," *IEEE Trans. Comput.*, vol. 44, no. 5, pp. 702–706, 1995.
[17] D. Xiang and J. Wu, "Fault-tolerant broadcasting in hypercube multicomputers using local safety information," Florida Atlantic Univ., Technical Report TR-CSE-99-36, 1999.
[18] D. Xiang and J. Wu, "Reliable multicasting in hypercube multicomputers using local safety information," in *Proc. 13th Int. Conf. Parallel and Distributed Computing Systems*, 2000, pp. 529–534.
[19] D. Xiang, "Fault-tolerant routing in hypercube multicomputers using local safety information," *IEEE Trans. Parallel and Distributed Syst.*, vol. 12, no. 9, pp. 942–951, 2001.

Dong Xiang received the B.S. in 1987 and the M.S. in 1990 in Computer Science from Chongqing University. He received the Ph.D. in 1993 in Computer Engineering from the Institute of Computing Technology, the Chinese Academy of Sciences, Beijing. He visited Concordia University, Montreal, Canada as a post-doctor from 1994 to 1995, and the University of Illinois, Urbana Champaign from 1995 to 1996. He has been with Institute of Microelectronics, Tsinghua University since 1996 October as an Associate Professor, and is now with the School of Software at Tsinghua University. His research interests include design and test of digital systems (design for testability, testability analysis, and BIST) and fault-tolerant computing, distributed computing, and computer networking. He authored *Digital Systems Testing and Design for Testability* (in Chinese, Science Press, 1997).

Ai Chen received the B.S. in 2001 in Electronic Engineering from Tsinghua University. He is working toward the M.S. degree at the Institute of Microelectronics, Tsinghua University. His research interests include fault-tolerant computing and distributed computing.

Jie Wu received the B.S. in 1982 and the M.S. in 1985 from Shanghai University of Science and Technology (now Shanghai University), and the Ph.D. in 1989 from Florida Atlantic University. He is a Professor in the Department of Computer Science and Engineering at Florida Atlantic University. He has published more than 150 papers in various journals and conference-proceedings. His research interests are in mobile computing, routing protocols, fault-tolerant computing, and interconnection networks. He serves on many conference committees and editorial boards. He was a co-guest-editor of IEEE TRANSACTIONS PARALLEL AND DISTRIBUTED SYSTEMS and Journal of Parallel and Distributed Computing. He authored the text *Distributed System Design*, CRC press. Dr. Wu received the 1996–1997 and 2001–2002 Researcher-of-the-Year Award at Florida Atlantic University. He served as an IEEE Computer Society Distinguished Visitor, and is a Member of ACM and a Senior Member of IEEE.