Optimal Broadcasting in Injured Hypercubes Using Directed Safety Levels *

Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

Abstract

Reliable communication in injured hypercubes with faulty links/nodes using directed safety levels is studied in this paper. In this approach, each node u in an n-dimensional hypercube (n-cube) is associated with a sequence of directed safety levels. A directed safety level associated with node u is an approximated measure of broadcast capability of performing optimal broadcasting from u in an $(n-1)$-subcube containing u. By optimal broadcasting, we mean that the broadcast message reaches each destination through a shortest path (i.e., the length of each path is equal to the Hamming distance between the source and destination). Directed safety levels are based on a special coding scheme generalized from Wu's safety level model and are calculated through $n-1$ rounds of information exchanges and updates among neighboring nodes. In this model, fault information of nodes within distance-d is precisely represented to better capture link faults, whereas fault information of nodes outside distance-d is approximated as in the regular safety level model. Optimal broadcasting at node u is guaranteed if node u is globally safe, which is defined in terms of the directed safety levels associated with u. The directed safety level model with $d = 1$ or $d = 2$ has the same asymptotic complexity as the regular safety level model. Simulation results show a significant improvement in terms of optimal broadcast capability in injured hypercubes.

*This work was supported in part by NSF grants CCR 9900646 and ANI 0073736.

1 Introduction

Several experimental and commercial multicomputers use the direct-connected network based on the hypercube topology because of its desirable topological properties, which include regular structure, embedability, and relatively small diameters [4, 12, 14, 19]. An *n-dimensional hypercube* (also called n-cube) consists of exactly 2^n processors, which can be addressed distinctively by n-bit binary numbers. Two nodes are directly connected by a link if and only if their binary addresses differ in exactly one bit position. Several research prototypes and systems have been built in the past two decades, such as NCUBE-2 [2], Intel iPSC [11], and the Connection Machine [5]. The more recently built SGI Origin 2000 [3] uses a variation of the hypercube topology.

Efficient broadcasting [6] of data can positively affect the performance of a multiprocessor system. Basically, broadcasting is the process of transmitting data from one node, called the source node, to all the other nodes. Broadcasting provides basic functions to implement distributed agreement [7], clock synchronization [10], and broadcast-and-aggregate type algorithms [9]. Because of its optimality and ease of implementation, the *binomial tree* [13] is frequently used to implement optimal broadcasting in hypercubes. By optimal broadcasting, we mean that the broadcast message reaches each destination through a shortest path (i.e., the length of each path is equal to the Hamming distance between the source and destination). The *binomial-tree-based broadcasting* ensures that the broadcast message is forwarded to each destination once and only once through a shortest path. The unique matching of hypercube and binomial tree structures facilitates a simple implementation of broadcasting in hypercubes.

An *injured hypercube* is a connected hypercube with faulty nodes and/or links. Optimal broadcasting in an injured hypercube is defined as successful broadcasting of the message to all the nonfaulty nodes through shortest paths. An *incomplete binomial tree* is normally used to implement an optimal broadcast process. An incomplete binomial tree in an injured n-cube is a connected subtree of an n-level binomial tree with the same root node that connects all the nonfaulty nodes through shortest paths. Note that the existence of an optimal broadcasting from a particular node in an injured hypercube depends on several factors: the number and type of faults and the distribution of faults. An *r-node* [15] is a node that, when used as the source node, can lead to the generation of at least one incomplete binomial tree. Determining the r-node status in a given injured hypercube is conjectured to be an NP-complete problem. Therefore, efforts have been directed to find a subset of r-nodes based on approximation. The challenge is to find the largest possible subset using a simple and efficient method, especially in a distributed and asynchronous hypercube multicomputer.

In an injured hypercube, we assume that faulty nodes/links occur (recover) dynamically. Most existing works focus on local distribution of information about fault occurrence/recovery rather than global distribution. This is done through a localized *marking process* [8], where each nonfaulty node is marked either safe or unsafe based on the safety status of its neighbors. When a node is marked safe, the optimal broadcast from that node is guaranteed. In general, the concept of *safe node* requires a much stronger condition than that of r-node. Therefore, the safe node set [8] is a relatively small subset of the r-node set. The safe node set can be decided in $O(n^2)$ rounds of information exchanges among neighboring nodes. Note that the marking process itself does not need to be synchronous. However, the scheme in [8] can only be used for injured n-cubes with fewer than $\lceil \frac{n}{2} \rceil$ node failures. That is, there are cases where no safe nodes exist in an injured n-cube with more than $\lceil \frac{n}{2} \rceil$ faulty nodes. Wu and Fernandez [16] gave a refined definition of safe node by relaxing certain conditions, and hence, increasing the size of the safe node set. The process that identifies node status needs fewer rounds than that of Lee and Hayes' safe node model in general. However, it still requires $O(n^2)$ rounds in the worst case.

Wu [15] proposed the concept of *safety level*, where each node in an n-cube is associated with a safety level l (an integer). A 0-safe node corresponds to a faulty node with the lowest level of safety, while an n-safe node (also referred to as a *safe node*) corresponds to the highest level of safety. The safety level model has the following property: If the safety level of node u is l $(0 \leq l \leq n)$, there exists a spanning incomplete binomial tree rooted at u in any m-subcube (with $m \leq l$) containing u. Clearly, when $l = n$, an incomplete binomial tree rooted at u exists in the n-cube, and thus, node u can be used to carry out an optimal broadcast. In other words, a safe node in the safety level model is an r-node. The safety level of each node can be calculated in $n - 1$ rounds in the worst case. The asymptotic complexity of the safety level model at each node is $O(n^2)$, since the cost of each round is $O(n)$. The safety level model can capture over 99% of r-nodes for $n(< 10)$-cubes with $k(< n)$ faulty nodes. However, the safety level model is not effective in handling link faults where two end nodes of each faulty link are treated faulty. In addition, the optimal broadcast capability reduces quickly as the number of faults increases, especially when the number is more than the dimension of the hypercube.

To provide a physical interpretation of the safety level, we envision a traffic control system within a city, where the street layout can be considered as a network topology. Each traffic congestion occurrence/recovery on a street and at an intersection can be viewed as a fault occurrence/recovery at a link and a node, respectively. Instead of flooding new traffic update, such information is spread locally through a marking process where the status of each intersection depends on the status of its neighbors, and is spread only to some "critical" intersections in the neighborhood. An

intersection is critical if a wrong turn made at the intersection will prevent a driver from reaching the destination through a shortest path. Specifically, each intersection has a safety level (k), indicating any destination within k hops from the intersection can be reached through a shortest path. In addition, the safety level can be used as a "guide" to find a shortest path. The marking process should not be overly "conservative" (a zero safety level assignment is always a safe choice, but it does not allow any traffic flow). Instead the safety status derived from the marking process should closely reflect the optimal routing/broadcasting capability. On the other hand, unlike the Bellman-Ford approach of deriving shortest paths through many iterative updates, the marking process, as an approximation, should be localized without long sequential information propagation. It has been shown in [15] that on the average, the safety level of each node can be calculated within 2 or 3 rounds in 9-cubes and 10-cubes under various fault distributions.

In this paper, we extend the the safety level model to make the safe node set closer to the r-node set. In addition, the model covers link faults effectively. All this is done *without increasing the complexity of the original safety level model*, i.e., $O(n^2)$. In the extended safety level model, each node u is associated with a sequence of directed safety levels. Each directed safety level corresponds to the broadcast capability of performing optimal broadcasting in a particular subcube from node u. Specifically, directed safety levels associated with node u are for $(n-1)$-dimensional subcubes only (there are n such subcubes that contain node u). A global safety level is then defined based on directed safety levels. In addition, the computation procedure for safety levels keeps exact fault information within distance-d (to better capture link faults) and approximates fault information outside distance-d (like in the regular safety level model). We show that the complexity will not increase when $d = 2$. One of the main reasons for using $d = 2$ is to handle link faults more effectively as will be discussed later. Simulation results show a significant improvement in terms of optimal broadcast capability in an injured hypercube, that is, the safe node set is very close to the r-node set under different combinations of link/node faults. Recently, another directed safety node model has been proposed for both unicasting [17] and broadcasting [18]. However, this model is built on Lee and Hayes' original safety node concept and, hence, it is weaker that the model proposed in this paper.

Note that another possible approach is based on routing tables by applying the Bellman-Ford algorithm, where each node maintains a routing table containing shortest path information to each destination. However, routing tables converge more slowly than safety levels [15], especially in a dynamic system. Formation of safety levels is free from the looping problem exhibited in routing tables. Also, a shortest path to each destination in a routing table associated with u does not guarantee the existence of an incomplete binomial tree rooted at u. In the absence of an

incomplete binomial tree, the broadcast process is more complicated unless it is implemented as a multiple-unicast (i.e., broadcast message is sent to each destination individually as one unicast).

The paper is organized as follows: Section 2 reviews concepts of hypercubes, binomial-tree-based broadcasting, and safety level model. Section 3 presents the proposed directed safety level model. Simulation results are provided in Section 4. The paper concludes in Section 5.

2 Preliminaries

2.1 Hypercubes

In an n-cube Q_n, a node u is represented (addressed) as $u_0u_1...u_{n-1}$, where u_k, $k = 0, 1, ..., n-1$, is called the k-dimensional bit. Two nodes are connected if and only if their addresses differ in one bit. u^k represents the neighbor of node u along dimension k. Each $m-$dimensional subcube Q_m (or $m-$subcube) has a unique address $q_0q_1 \dots q_{n-1}$ with $q_k \in \{0,1,*\}$, where exactly m bits take the value $*$, which is a don't care symbol that can be either 0 or 1.

2.2 Binomial-tree-based broadcasting

An *n-level binomial tree*, B_n, is recursively defined as follows: (1) Any tree consisting of a single node is a B_0 tree. (2) Suppose that T and T' are disjoint B_{n-1} trees for $n \geq 1$. Then the tree obtained by adding an edge to make the root of T become the left most offspring of the root T' is a B_n tree.

Definition 1: *A spanning incomplete binomial tree B_n in an injured n-cube is a connected subtree of an n-level binomial tree with the same root node that connects all the nonfaulty nodes in the n-cube.*

Figure 1 (b) shows an injured 3-cube with a spanning incomplete binomial tree rooted at node 111, where faulty nodes are represented by black nodes. One property of a spanning incomplete binomial tree is that starting from the root and following the tree each nonfaulty node can be reached through a Hamming-distance path, i.e., a shortest path.

Binomial-tree-based broadcasting follows a binomial tree to carry out the broadcast process. An n-level binomial tree B_n can also be recursively defined as a tree with one root node and n disjoint subtrees B_{n-1}, B_{n-2}, ..., B_1, and B_0. The root node connects the roots of these n subtrees.

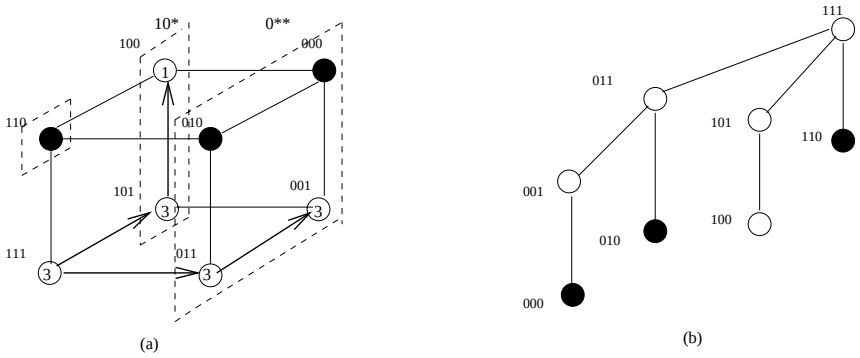

Figure 1: (a) An injured 3-cube and (b) a spanning incomplete binomial tree rooted at 111.

This recursive definition of B_n provides us an easy way to determine an r-node. Basically, node u is an r-node in B_n if and only if its neighbors can generate n disjoint subtrees B_{n-1}, B_{n-2}, ..., B_1, and B_0, where each B_k is an incomplete spanning binomial tree of a subcube $Q_k^{'}$ in a partition $\{Q_{n-1}^{'}, Q_{n-2}^{'}, ..., Q_1^{'}, Q_0^{'}, u\}$ of Q_n. This property of binomial tree will serve as a basis for the safety level model discussed later. The partitioning of a cube (subcube) depends on a *coordinate sequence* (cs) associated with each node. Basically, cs: $i_1 i_2 ... i_n$ is a permutation of bit positions and it defines the dimension order a given cube (subcube) is partitioned at a given node. In Figure 1 (a), the cs at 111 is 021. Therefore, the dimension order is dimension 0, dimension 2, and dimension 1. The 3-cube $***$ at 111 is partitioned into $0**$ and $1**$ along dimension 0 first. $1**$ (that does not contain 111) is then partitioned into $10*$ and $11*$ along dimension 2. Finally, $11*$ is partitioned into 110 and 111 (root). That is, $***=\{0**, 10*, 110, 111\}$. B_2, B_1, and B_0 are incomplete spanning binomial trees of subcubes $0**$, $10*$, and 110, respectively. Figure 1 (a) shows the broadcast process rooted at node 111 based on the binomial tree of Figure 1 (b). A simple header encoding scheme can be adopted where $0**$ and $10*$ are destination subcubes forwarded to neighbors 011 and 101, respectively. Neighbor 110 is faulty and there is no need of forwarding the message to it.

In general, the broadcasting of Q_n at source node u with a given cs consists of the following steps: (1) Partition the Q_n into, $Q_{n-1}^{'}, Q_{n-2}^{'}, ..., Q_0^{'}$, at source node u with respect to cs: $i_1 i_2 ... i_n$. (2) Send subcube $Q_{n-k}^{'}$ to u^{i_k}, $k = 1, 2, ..., n$. (3) Repeat steps (1) and (2) to each $Q_{n-k}^{'}$ at node u^{i_k} with the same cs or a different one.

To send address of subcube $Q_{n-k}^{'}$ from u to u_{i_k}, it is not necessary to send the string of $(0, 1, *)$ which corresponds to the subcube. We only need a representation that can indicate the locations of those $*$ in $Q_{n-k}^{'}$. In the following algorithm, an n-bit control word LABEL is used to represent

a subcube. In LABEL each bit is either 1 or 0 with each 1 corresponding to a $*$ in the subcube.

Safety-level-based broadcast
{ At node u of an n-cube with cs: $i_1 i_2 ... i_n$ }
begin
  if $u = source\_node$
    then for $k = 1$ **to** n $LABEL[k] := 1$
  for $k = 1$ **to** n
    if $LABEL[i_k] \neq 0$
      then begin
        $LABEL[i_k] := 0$;
        send the broadcast data and $LABEL$ to u^{i_k}
        end
end.

Broadcasting to all the neighbors of a given node is performed by a sequence of splitting and sending subcubes. A subcube is split by assigning the value 0 to a particular 1-bit in LABEL and that new label together with the broadcast data is immediately sent to the appropriate neighbor. The splitting process continues on the new LABEL which corresponds to a new subcube one dimension less than the previous subcube. In fault-free hypercubes, cs can be selected arbitrarily. However, cs at each node has to be carefully determined to guarantee an incomplete spanning binomial tree in an injured hypercube. One way to select cs is based on the safety level model.

2.3 Safety levels

The safety level model is a labeling scheme that assigns a label to each node in a given n-cube. Labels are used to capture fault information in the neighborhood. Each label must satisfy the following property:

Embedding property: *If the label of node u is l (the node is called l-safe), then u can be the root of a spanning incomplete binomial tree in any m-subcube ($m \leq l$) containing u of a given injured hypercube.*

Specifically, when the safety level of a node is an integer l, it is a measure of binomial-tree

embedability of the node. Clearly, a node is an r-node if its safety level is n in an n-cube. The safety level of node u is l if u can embed binomial tree B_l in any l-subcube Q_l (that contains u). This can be done by partitioning Q_l into $\{Q'_{l-1}, Q'_{l-2}, ..., Q'_1, Q'_0, u\}$ and each Q'_k can generate a B_k rooted at the neighbor of u, that is, the safety level of that neighbor is at least k.

Definition 2 [15]: *The safety level of a faulty node is* 0. *For a nonfaulty node u, let* $(ns_{i_0}, ns_{i_1}, ns_{i_2}, ..., ns_{i_{n-1}})$, $ns_{i_k} \in \{0, 1, 2, ..., n\}$, *be a nondescending safety level sequence of node u's neighbors in an n-cube. The safety level of node u is defined by the following* decision process*: if* $(ns_{i_0}, ns_{i_1}, ns_{i_2}, ..., ns_{i_{n-1}}) \geq (0, 1, 2, ..., n-1)$, *then* $s(u) = n$ *else if* $(ns_{i_0}, ns_{i_1}, ns_{i_2}, ..., ns_{i_{n-l}}) \geq (0, 1, 2, ..., l) \wedge (ns_{i_{l+1}} = l)$ *then* $s(u) = l + 1$.

In the above definition, $seq_1 \geq seq_2$ if and only if each element in seq_1 is greater than or equal to the corresponding element in seq_2. Note that $(0, 1, 2, ..., n-1)$ in Definition 2 is also called a *reference sequence* for the decision process. The safety level of a nonfaulty node is initialized to n and the safety level of a faulty node is initialized to 0. Two end nodes of a faulty link are considered faulty.

The safety level of each node in Figure 1 (a) is shown inside with cycle (representing a node). The following shows details of deriving the safety level for node 101 of Figure 1. Two rounds of updates are shown and each update is based on neighbor safety levels (ns_0, ns_1, ns_2) obtained at the current round. Initially, the safety level for node 101 (and every nonfaulty node) is 3. In round 1, $(ns_0, ns_1, ns_2) = (3, 3, 3)$. Since $(3, 3, 3) \geq (0, 1, 2)$, $s(101) = 3$. After round 1, the safety level of the neighbor 100 (along dimension 2) drops to 1. Hence, $(ns_0, ns_1, ns_2) = (3, 3, 1)$ after round 2. Since $(1, 3, 3) \geq (0, 1, 2)$, $s(101)$=3 still holds. As shown in Figure 1 (a), all nonfaulty nodes (except 100) are safe (with a safety level of 3). It has been shown [15] that the safety level is well-defined and satisfies the embedding property, i.e., an n-safe node (called *safe*) is an r-node. Other nodes are called *unsafe*.

With the safety level model, the broadcast algorithm can be modified as follows: If the source node is safe then apply the safety-level-based broadcast algorithm with *cs of node u (source or an intermediate node) being a nonascending safety level sequence of u's neighbors.* If the source node is unsafe, the broadcast can start from a safe neighbor (such a neighbor always exists as long as the number of node faults is less than n [15]).

However, the safety level model cannot be used if there is no safe node in the cube. Figures 2 (a) and (b) show two examples of injured 4-cubes without any safe nodes. Again, the safety level of each node is placed inside the circle representing the node. In particular, the safety level is not effective in handling link faults. As shown in Figure 2 (b), the highest safety level is 3 while many

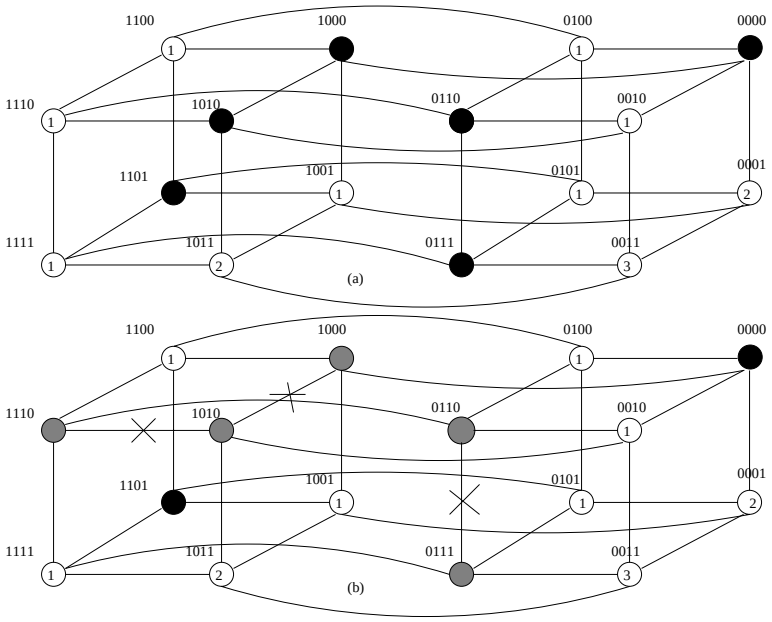

Figure 2: Injured 4-cubes without safe nodes: (a) node faults, (b) a mix of node and link faults.

nodes in this example are r-nodes.

The following results show the complexity of the safety level model. Results in [15] show that the safety level can be calculated in a few rounds (2 or 3 when the number of faults is less than the dimension) in average and $n-1$ in the worst case (Lemma 1a). The complexity of decision process is $O(n^2)$ at each node as shown in Theorem 1.

Lemma 1a: *At the kth round of the decision process, nodes and only the nodes that are k-safe change their labels (from a high level) to k.*

Proof: We prove this lemma using mathematical induction on k. When $k = 1$, there are at least two faulty (0-level) neighboring nodes for any 1-safe node based on the safety level definition. Also, based on the definition that $s = m + 1$ if $(ns_{i_0}, ns_{i_1}, ..., ns_{i_m}) \geq (0, 1, ..., m)$ and $ns_{i_{m+1}} = m$, for a node to change to $(m+1)$-safe with $m \neq 0$ there must be at least one neighbor that is m-safe. However, all nodes are initialized to 0 or n. Therefore, nodes and only nodes that are 1-safe change their status at round one. Assume that the lemma holds for all k-safe nodes, $k \leq l$, exactly k rounds are required for these nodes to stabilize their status. Based on the safety level definition, an $(l+1)$-safe node can identify its status once all its neighbors with safety levels lower than $l+1$ have the stable status. By induction, exactly after the lth round all those neighbors are stabilized.

To prove that only nodes that are $(l+1)$-safe stablize, we only need to observe the fact that for an $(m+1)$-safe node with $m > l$ to stablize at round $(l+1)$ at least one neighbor's status should be m. Based on induction assumption, it is impossible to have a stable m-safe node within l rounds (with $l < m$). □

As a simple corollary of Lemma 1a, at most $n-1$ rounds of decision process are needed; therefore, the calculation procedure, that consists of one round of initialization and $n-1$ rounds of decision process, correctly determines safety levels for all nodes.

Lemma 1b: *At each round of the decision process, each node can sort safety levels of neighbors in* $O(n)$.

Proof: Since safety levels can take up to n distinct values, counting sort that has a complexity of $O(n)$ can be applied. Basically, safety levels of neighbors can be placed in n ordered boxes (one for each distinct value). The leftmost box holds safety levels of 0 and the rightmost box holds safety levels of n. Re-scanning safety levels in these boxes from left to right will generate sorted safety levels of neighbors. □

Actually, at the k-th round, only those safety levels of neighbors that are no less than k need to be sorted. However, overall complexity in the worst case is still $O(n)$.

Theorem 1: *The complexity of the calculation procedure for safety level is* $O(n^2)$ *at each node.*

Proof: Safety level initialization takes $O(1)$ for each node. The decision process at each node consists of two steps: sort the safety levels of its neighbors and update its safety level. Based on Lemma 1b, sorting takes $O(n)$. Update is based on comparing two n-element sequences and it takes n steps. The decision process is repeated $n-1$ times to derive the final safety level (based on Lemma 1b). Therefore, the overall complexity is $O(n^2)$. □

3 Proposed Model

3.1 Basic ideas

We aim to devise a good labeling scheme that generates a larger safe node set than does the safety level model and that is close to the r-node set. Moreover, if a node u is not an r-node, we try to find a maximum k such that node u is the r-node of any m-subcube (with $m \leq k$) containing u. Such a node is called an r-k-node. Clearly, the safety level (l) of an r-k-node is no more than k.

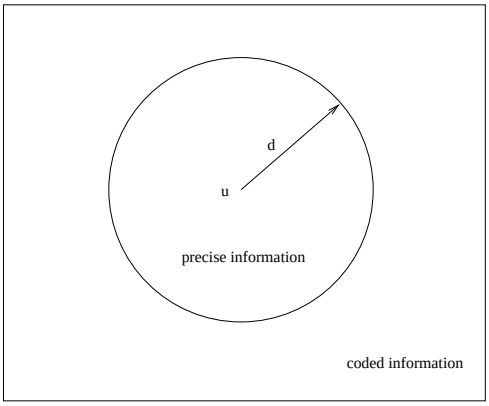

Figure 3: A new coding method.

The closer l to k, the better the approximation of node status of its neighbors. The challenge here is to design a simple labeling scheme, and at the same time, to meet the embedding property. Our approach consists of the following two steps (see Figure 3):

1. Each node knows exact fault information within distance-d. A node can be labeled precisely if it is an r-k-node such that $k \leq d$.

2. Fault information outside distance-d is approximated. The label is approximated using the safety level model if a node is an r-k-node such that $k > d$.

A new labeling scheme is used in the proposed model. In addition to the safety level (SL) model that assigns one integer to each node, the *directed safety level* (DSL) model is introduced that assigns a vector of integers to each node. The directed safety level is an extension of the safety level by determining n safety levels, one for each $(n-1)$-subcube containing u (a total of n such subcubes exist in an n-cube). Two initializations are used that correspond to distance-1 information and distance-2 information, respectively. Therefore, there are four schemes: SL(1) (regular safety level), SL(2), DSL(1), and DSL(2). For models with $d > 2$, the corresponding complexity (both in time and space) is too high. Also, although theoretically each node u can maintain directed safety levels for all $(n-c)$-subcubes containing u, with $c > 1$, the corresponding complexity is again too high. Therefore, all these general cases will not be considered in this paper. Note that a concept similar to directed safety levels has been studied in [1]. Our approach here is more general with distance-d fault information at each node.

3.2 SL(2)

Safety levels based on $d = 2$ initialization can be calculated in a manner similar to that for $d = 1$ initialization. The main purpose of using $d = 2$ initialization is to handle link faults more effectively. In this model, each node knows the exact broadcast capability of neighbors that are within 2 hops.

For node u, its safety level s is initialized as follows: (1) s is set to 0 if node u is faulty or there is an adjacent faulty link; otherwise, (2) s is set to 1 if node u cannot reach a distance-2 node v, i.e., two shortest paths from u to v are blocked by faulty nodes and/or links. (3) s is set to n for all other cases. Node u is an *assigned node* if its safety level is initialized to either 0 or 1; otherwise, it is called an *unassigned node*.

The safety level of an assigned node is that assigned in the initialization. The safety level of an unassigned node is based on safety levels of its neighbors. Suppose ns_k is the safety level of u's neighbor along dimension k. Node u gets n safety levels, $ns_{i_0}, ns_{i_1}, ns_{i_2}, ns_{i_3}$,..., ns_{n-1}, from n neighbors. The following decision process with reference sequence $(0, 0, 2, 3, ..., n-1)$ is applied:

- Arrange safety levels of its neighbors in a nondescending sequence $(ns_{i_0}, ns_{i_1}, ns_{i_2}, ns_{i_3}, ..., ns_{i_{n-1}})$.
- If $(ns_{i_0}, ns_{i_1}, ns_{i_2}, ns_{i_3}, ..., ns_{i_{n-1}}) \geq (0, 0, 2, 3, ..., n-1)$, then $s = n$.
- If $(ns_{i_0}, ns_{i_1}, ns_{i_2}, ns_{i_3}, ..., ns_{i_l}) \geq (0, 0, 2, 3, ..., l)$ and $ns_{i_{l+1}} = l$, then $s = l + 1$.

In reference sequence $(0, 0, 2, 3, ..., n-1)$, the first two elements are *masked* since safety levels of 0 and 1 have been initialized. Because the highest safety level (n) and two lowest safety levels (0 and 1) are derived after the initialization phase, a total of $n - 2$ rounds of iteration are needed. Again, a node is *safe* if it has a safety level of n; otherwise, it is called *unsafe*. The complete calculation for $SL(2)$ is as follows:

In order to show the superiority of the proposed model over the regular safety level model in terms of better handling of link faults, let us consider node 0100 of Figure 2 (b). Under the regular safety level model, the safety level of 0100 is 1, because it has two adjacent faulty neighbors 0000 and 0110 (end node of a faulty link). However, by inspection 0100 can reach any nonfaulty node within 2 hops. Under the proposed model, node 0100 will not be initialized to 1 and its final safety level is 2. The safety levels of nodes in Figure 2 (b) (from 0 to 15) under SL(2) are: 0, 4, 1, 4, 2, 4, 0, 0, 0, 4, 0, 4, 1, 0, 0, 2. Note that SL(2) is the same as SL(1) when the hypercube contains node faults only (as in Figure 2 (a)).

Calculation procedure for SL(2)

1. Initialize the safety level for each node. This can be done based on distance-2 information through two rounds of information exchanges among neighbors.
2. Use the decision process with reference sequence $(0, 0, 2, 3, ..., n-1)$ to update each safety level for an unassigned node based on safety levels of n neighbors.
3. Repeat step 2. $n-2$ times.
4. The safety level of an assigned node is its initial value and the safety level of an unassigned node is the one decided by the $(n-2)$-round decision process.

Theorem 2: *The complexity of the calculation procedure for* $SL(2)$ *is* $O(n^2)$.

Proof: In step 1, each node collects distance-2 information in two rounds of information exchanges among neighbors. There are $O(n^2)$ distance-2 neighbors and the reachability to each neighbor can be checked in constant time, and hence, step 1 takes $O(n^2)$. The other three steps are the same as those in the regular safety level model with $d = 1$ initialization. Therefore, the overall complexity of the calculation procedure is still $O(n^2)$. □

3.3 DSL(1)

In the directed safety level model, each node u keeps a safety level for each $(n-1)$-subcube containing u. Therefore, the directed safety level model offers a better approximation than the regular safety level model. Each node u in an n-cube is associated with n *directed safety levels*, $(s_0, s_1, ..., s_{n-1})$, where s_k, $k = 0, 1, ..., n-1$, is called node u's directed safety level *excluding dimension* k. Each directed safety level s_k indicates the broadcast capability of the current node in the $(n-1)$-subcube generated from the partition of the n-cube along dimension k. For example, suppose $u = 101$ in a 3-cube, s_0, s_1, and s_2 represent broadcast capability in subcubes $1**$, $*0*$, and $**1$, respectively. Once the directed safety levels are obtained through rounds of information exchanges among neighbors, an *adjustment process* is applied to generate *adjusted directed safety levels* (ADSL), denoted as $(as_0, as_1, ..., as_{n-1})$. ADSL corresponds to a nondescending directed safety level sequence of neighbors based on a particular hypercube partition. Finally, a *global safety level* (GSL), s, is derived from the decision process with ADSL being the reference sequence.

Figure 4: (a) Directed safety level ns_{kk} excluding dimension k. (b) Selecting a partition dimension p in the adjustment process.

Directed safety levels offer a finer model of approximation (i.e., broadcast capability within each $(n-1)$-subcube). Once a node knows the broadcast capability of its neighbors in different subcubes, global safety level can be derived which is the broadcast capability in the whole n-cube.

Directed safety levels of a nonfaulty node are initialized to $((n-1),(n-1),...,(n-1))$ and directed safety levels of a faulty node are initialized to $(0,0,...,0)$. Two end nodes of a faulty link are considered faulty. The calculation of directed safety levels of a nonfaulty node is based on directed safety levels of its neighbors. Suppose $ns(k) = (ns_{k0}, ns_{k1}, ..., ns_{k(n-1)})$ are directed safety levels of node u^k (u's neighbor along dimension k), where ns_{kj} is u^k's directed safety level excluding dimension j. The directed safety level excluding dimension k for node u^k, ns_{kk}, is called the *relevant directed safety level* RDSL of neighbor u^k with respect to node u. Figure 4 (a) illustrates the meaning of ns_{kk}. Basically, if the n-cube is partitioned into Q_{n-1} (containing u) and Q'_{n-1} along dimension k, ns_{kk} is the directed safety level of u^k (neighbor of u along dimension k) in subcube Q'_{n-1}. Node u gets n RDSL's from n neighbors, ns_{00}, ns_{11}, ..., $ns_{(n-1)(n-1)}$. Then s_k is defined by the following decision process with reference sequence $(0,1,...,n-2)$:

- Arrange directed safety levels of neighbors in $\{ns_{00}, ns_{11}, ..., ns_{k-1k-1}, ns_{k+1k+1}, ..., ns_{n-1n-1}\}$ in a nondescending sequence $\{ns_{i_0i_0}, ns_{i_1i_1}, ..., ns_{i_{n-2}i_{n-2}}\}$.
- If $(ns_{i_0i_0}, ns_{i_1i_1}, ..., ns_{i_{n-2}i_{n-2}}) \geq (0,1,...,n-2)$ then $s_k = n-1$.
- If $(ns_{i_0i_0}, ns_{i_1i_1}, ..., ns_{i_li_l}) \geq (0,1,...,l)$ and $ns_{i_{l+1}i_{l+1}} = l$, then $s_k = l+1$.

Note that ns_{kk} is not included in the decision process for s_k since dimension k is excluded in the corresponding $(n-1)$-subcube. Because the highest directed safety level $(n-1)$ and the lowest directed safety level (0) are derived after initialization, a total of $n-2$ rounds of the above updates are needed. s_k's have at most 3 distinct values and they can be calculated in $O(n^2)$ as will be discussed later in Lemma 3a.

We use node 0001 of Figure 2 (b) to show the process of deriving directed safety levels. Directed safety levels of neighbors are $ns(0), ns(1), ns(2), ns(3)$ where each RDSL is tagged with $*$. Directed

safety levels for node 0001, (s_0, s_1, s_2, s_3), are calculated through a 2-round process. Initially, the directed safety levels for node 0001 are (3,3,3,3). After round 1, $ns(0) = (3^*, 3, 3, 3)$, $ns(1) = (3, 3^*, 3, 3)$, $ns(2) = (3, 3, 3^*, 3)$, and $ns(3) = (0, 0, 0, 0^*)$. Therefore, $(ns_{33}, ns_{22}, ns_{11}, ns_{00}) = (0, 3, 3, 3)$, and then, $(s_0, s_1, s_2, s_3) = (3, 3, 3, 3)$. After round 2, $ns(0) = (1^*, 3, 1, 3)$, $ns(1) = (3, 1^*, 3, 1)$, $ns(2) = (3, 3, 3^*, 3)$, and $ns(3) = (0, 0, 0, 0^*)$. Therefore, $(ns_{33}, ns_{11}, ns_{00}, ns_{22}) = (0, 1, 1, 3)$, and then, $(s_0, s_1, s_2, s_3) = (3, 3, 2, 3)$.

The global safety level of a node (i.e., safety level in the n-cube) is defined based on a nondescending sequence of directed safety levels of neighbors, also called adjusted directed safety levels (ADSL), based on a particular hypercube partition. ADSL of each node is decided by an adjustment process that determines a hypercube partition that can generate a *highest* ADSL. An ADSL is said to be higher than the other if each of its elements is no smaller than the corresponding one of the other. In addition, at least one of its elements is larger than that of the other. Note that the only purpose of ADSL is to derive the global safety level of the node as will be discussed later. To determine ADSL, node u collects stabilized $ns(k) = (ns_{k0}, ns_{k1}, ..., ns_{k(n-1)})$ from neighbors u^k, where $k = 0, 1, ..., n-1$. $ns = (ns(0), ns(1), ..., ns(n-1))^T$ is a vector and it can also be considered as an $n \times n$ matrix $ns{=}(ns_{ij})$, where $i, j = 0, 1, ..., n-1$. ADSL, $(as_{i_0}, as_{i_1}, ..., as_{i_{n-1}})$ are calculated one by one in the adjustment process.

Adjustment process

1. $k := n-1, selected\_dimension := \{\}$. /* initialization */
2. $ns_{pq} := \max\{ns_{ij} | i \notin selected\_dimension, j \in selected\_dimension \cup \{i\}\}$.
   /* select the next as from an unselected row p */
3. $i_k := p, as_{i_k} := \min\{ns_{pq}, as_{i_{k+1}}\}$.
   /* ensuring as as a nondescending sequence and assuming $as_{i_n} = n$ */
4. $k := k-1$, $selected\_dimension := selected\_dimension \cup \{p\}$.
   /* update for the next round of selection */
5. If $k \geq 0$, goto step 2.

Figure 4 (b) explains the logic behind the adjustment process: Elements in ADSL are selected one by one in $n-1$ rounds. In each round, the largest element ns_{pq} (highest safety level) from an unselected row p in the $n \times n$ matrix ns is selected. The column number q must be either p or one of previously selected rows. More specifically, initially a neighbor (of u) with the largest ns_{qq}

is selected (i.e., the cube is partitioned along dimension q). Suppose we are at an intermediate step of the partition process and Q_k was partitioned into Q_{k-1} and Q'_{k-1} along q at a previous step. To partition Q_{k-1} at u, we can select dimension p such that u^p has the largest directed safety level excluding dimension q, ns_{pq}, or excluding dimension p, ns_{pp}. Note that dimension q here should be interpreted as any dimension that has been selected. Clearly, ns_{pq} is the safety level in an (n-1)-subcube that contains Q_{k-1}.

Again, we use node 0001 of Figure 2 (b) to show the process of deriving adjusted safety levels. Matrix ns shows the final directed safety levels of neighbors. The following shows a 4-round of iterations to select as_{i_3}, as_{i_2}, as_{i_1}, and as_{i_0}, respectively. In each round, directed safety levels under consideration are underlined. The selected element as_{i_k} at round k is tagged with $*$. In the first round, only diagonal elements are considered.

$$\text{Round 1} : ns = \begin{pmatrix} \underline{1} & 3 & 1 & 3 \\ 3 & \underline{1} & 3 & 1 \\ 3 & 3 & \underline{3}^* & 3 \\ 0 & 0 & 0 & \underline{0} \end{pmatrix} \quad \text{Round 2} : ns = \begin{pmatrix} \underline{1} & 3 & \underline{1} & 3 \\ 3 & \underline{1} & \underline{3}^* & 1 \\ 3 & 3 & 3 & 3 \\ 0 & 0 & \underline{0} & \underline{0} \end{pmatrix}$$

$$\text{Round 3} : ns = \begin{pmatrix} \underline{1} & \underline{3}^* & \underline{1} & 3 \\ 3 & 1 & 3 & 1 \\ 3 & 3 & 3 & 3 \\ 0 & \underline{0} & \underline{0} & \underline{0} \end{pmatrix} \quad \text{Round 4} : ns = \begin{pmatrix} 1 & 3 & 1 & 3 \\ 3 & 1 & 3 & 1 \\ 3 & 3 & 3 & 3 \\ \underline{0}^* & \underline{0} & \underline{0} & \underline{0} \end{pmatrix}$$

Finally, the *global safety level* is defined based on ADSL. If $(as_{i_0}, as_{i_1}, ..., as_{i_{n-1}}) \geq (0, 1, ..., n-1)$, where $(as_{i_0}, as_{i_1}, ..., as_{i_{n-1}})$ is the adjusted directed safety level of u, the global safety level of u, s, is n; otherwise, if $(as_{i_0}, as_{i_1}, ..., as_{i_l}) \geq (0, 1, ..., l)$ and $as_{i_{l+1}} = l$, then $s = l + 1$.

In the example of Figure 2 (b), since the adjusted safety levels for node 0001 are $(as_{i_0}, as_{i_1}, as_{i_2}, as_{i_3}) = (0, 3, 3, 3)$, the final global safety level for node 0001 is 4. The calculation procedure for $DSL(1)$ is given as follows:

Lemma 3a: *Directed safety levels can be calculated in* $O(n^2)$.

Proof: Without loss of generality, assume that $(ns_{00}, ns_{11}, ..., ns_{(n-1)(n-1)})$ is the nondescending relevant directed safety levels of neighbors for node u. Note that s_k of u is derived by applying the

---

Calculation procedure for $DSL(1)$

1. Initialize directed safety levels for each node.
2. Use the decision process with reference sequence $(0, 1, ..., n-2)$ to update each directed safety level s_k based on relevant directed safety levels RDSL's of n neighbors (excluding the one from the neighbor along dimension k).
3. Repeat step 2. $n-2$ times.
4. Collect n directed safety levels (each is a vector DSL) from n neighbors. Apply the adjustment process to obtain adjusted directed safety levels (a vector ADSL) for each node.
5. Use the decision process with reference sequence $(0, 1, ..., n-1)$ to determine the global safety level GSL of each node based on ADSL of each node.

---

decision process to $(ns_{00}, ns_{11}, ..., ns_{(k-1)(k-1)}, ns_{(k+1)(k+1)}, ..., ns_{(n-1)(n-1)})$ with reference sequence $(0, 1, ..., n-2)$. If the safety level of u (based on Definition 2) is $l+1$ by applying the decision process to sequence $(ns_{00}, ns_{11}, ..., ns_{(n-1)(n-1)})$ with reference sequence $(0, 1, ..., n-1)$, we consider the following three cases:

1. The sequence is of the form: $(ns_{00}(\geq 0), ns_{11}(\geq 1), ..., ns_{ll}(= l), ns_{(l+1)(l+1)}(= l), ns_{(l+2)(l+2)}$ $(= l)$, $ns_{(l+3)(l+3)}(\geq l), ...)$, any element removed from $\{ns_{00}, ns_{11}, ..., ns_{(n-1)(n-1)}\}$ in the above sequence will generate a directed safety level of $l+1$.
2. The sequence is $(ns_{00}(\geq 0), ns_{11}(\geq 1), ..., ns_{ll}(= l), ns_{(l+1)(l+1)}(= l), ns_{(l+2)(l+2)}(\geq l+1)$, $ns_{(l+3)(l+3)}$ $(\geq l+2), ..., ns_{(l+k)(l+k)}$ $(\geq l+k-1), ns_{(l+k+1)(l+k+1)}(= l+k-1), ...)$, any element removed from $\{ns_{00}, ns_{11}, ..., ns_{ll}\}$ in the above sequence will generate a directed safety level of $l+k$ and any element removed from $\{ns_{(l+1)(l+1)}, ns_{(l+2)(l+2)}, ..., ns_{(n-1)(n-1)}\}$ in the above sequence will generate a directed safety level of $l+1$.
3. The sequence is $(ns_{00}(\geq 0), ns_{11}(\geq 1), ..., ns_{ll}(= l), ns_{(l+1)(l+1)}(= l), ns_{(l+2)(l+2)}(\geq l+1)$, $ns_{(l+3)(l+3)}$ $(\geq l+2), ..., ns_{(n-1)(n-1)}$ $(\geq n-2))$, any element removed from $\{ns_{00}, ns_{11}, ..., ns_{ll}\}$ in the above sequence will generate a safety level of $n-1$ and any element removed from $\{ns_{(l+1)(l+1)}, ns_{(l+2)(l+2)}, ..., ns_{(n-1)(n-1)}\}$ in the above sequence will generate a safety level of $l+1$.

It is clear from the analysis of the above three cases that s_k's have three distinct possible values,

and therefore, directed safety levels can be updated in $O(n)$ at each round. Therefore, after $n-2$ rounds, directed safety levels are calculated with a total cost of $O(n^2)$. □

Lemma 3b: *The adjustment process takes* $O(n^2)$.

Proof: Based on Lemma 3a, there are at most three distinct values in the directed safety levels $(s_0, s_1, s_2, ..., s_{n-1})$ for each node. A simple preprocessing with a cost of $O(n^2)$ can determine distinct values of directed safety levels of all n neighbors. With up to three distinct values from each row of the matrix ns, the cost of step 2 in the adjustment process is $O(n)$. Since step 2 is repeated $O(n)$ times, the overall cost is $O(n^2)$. □

Theorem 3: *The complexity of the calculation procedure for DSL(1) is* $O(n^2)$.

The proof of Theorem 3 can be directly derived from Lemmas 3a and 3b and Theorem 2.

3.4 DSL(2)

Directed safety levels based on $d = 2$ initialization can be calculated in a manner similar to that for $d = 1$ initialization. However, this model is more efficient in handling link faults.

For node u, the directed safety level along dimension k, s_k, is initialized as follows: (1) s_k is set to 0 if node u is faulty or there is an adjacent faulty link in the subcube $Q_{n-1}(u, k)$, which is an $(n-1)$-subcube containing u and is derived by partitioning Q_n along dimension k. (2) s_k is set to 1 if node u cannot reach a distance-2 node v in the subcube $Q_{n-1}(u, k)$, i.e., two shortest paths from u to v are blocked by faulty nodes and/or links. (3) s_k is set to $n-1$ for all other cases.

A node is called an assigned node if one of its directed safety levels is initialized to 0 or 1. Otherwise, it is called an unassigned node. A safety status in a directed safety level of a node (assigned or unassigned) is called an assigned element if it is initialized to 0 or 1; otherwise, it is called an unassigned element.

The procedure for directed safety levels is the same as the one for DSL(1) except that the reference sequence is $(0, 0, 2, 3, ..., n-2)$. Because the highest directed safety level is $n-1$ and the two lowest directed safety levels are 0 and 1 after initialization, a total of $n-3$ rounds of the decision process are needed. The procedure for adjusted directed safety levels is the same as the one for DSL(1). Finally, the global safety level is given in the following decision process with reference sequence $(0, 0, 2, 3, ..., n-1)$: If a node is an assigned node, its global safety level is the minimum of its assigned elements. If a node is an unassigned node and $(as_{i_0}, as_{i_1}, as_{i_2}, as_{i_3},$

Table 1: (a) DSL(1) in Figure 2 (a) and DSL(1) in Figure 2 (b). (b) DSL(2) in Figure 2 (a). (c) DSL(2) in Figure 2 (b).

node	0000	0001	0010	0011	0100	0101	0110	0111
DSL	0,0,0,0	3,3,2,3	1,1,1,1	3,3,3,3	1,2,2,1	3,1,3,1	0,0,0,0	0,0,0,0
ADSL	-	0,3,3,3	0,0,0,0	0,1,3,3	0,0,2,2	0,0,2,2	-	-
GSL	0	4	1	4	1	1	0	0
node	1000	1001	1010	1011	1100	1101	1110	1111
DSL	0,0,0,0	1,3,1,3	0,0,0,0	2,3,3,3	1,2,1,2	0,0,0,0	2,2,1,1	3,1,3,1
ADSL	-	0,0,3,3	-	0,3,3,3	0,0,2,2	-	0,0,2,2	0,0,2,3
GSL	0	1	0	4	1	0	1	1

(a)

node	0000	0001	0010	0011	0100	0101	0110	0111
DSL	0,0,0,0	3,3,2,3	1,1,1,1	3,3,3,3	1,2,2,1	3,1,3,1	0,0,0,0	0,0,0,0
ADSL	-	0,3,3,3	0,0,0,0	0,1,3,3	0,0,1,1	0,0,2,2	-	-
GSL	0	4	1	4	1	1	0	0
node	1000	1001	1010	1011	1100	1101	1110	1111
DSL	0,0,0,0	1,3,1,3	0,0,0,0	2,3,3,3	1,1,1,1	0,0,0,0	0,0,0,0	1,1,1,1
ADSL	-	0,0,3,3	-	0,1,3,3	0,0,0,2	-	-	0,0,0,3
GSL	0	1	0	4	1	0	0	1

(b)

node	0000	0001	0010	0011	0100	0101	0110	0111
DSL	0,0,0,0	3,3,3,3	3,1,3,1	3,3,3,3	3,3,3,2	3,3,3,3	0,0,0,3	0,0,0,3
ADSL	-	0,3,3,3	-	3,3,3,3	0,3,3,3	0,3,3,3	-	-
GSL	0	4	1	4	4	4	0	0
node	1000	1001	1010	1011	1100	1101	1110	1111
DSL	0,0,3,0	3,3,3,3	0,0,0,0	3,3,3,3	1,3,3,1	0,0,0,0	0,3,0,0	3,2,3,3
ADSL	-	0,3,3,3	-	0,3,3,3	-	-	-	0,3,3,3
GSL	0	4	0	4	1	0	0	4

(c)

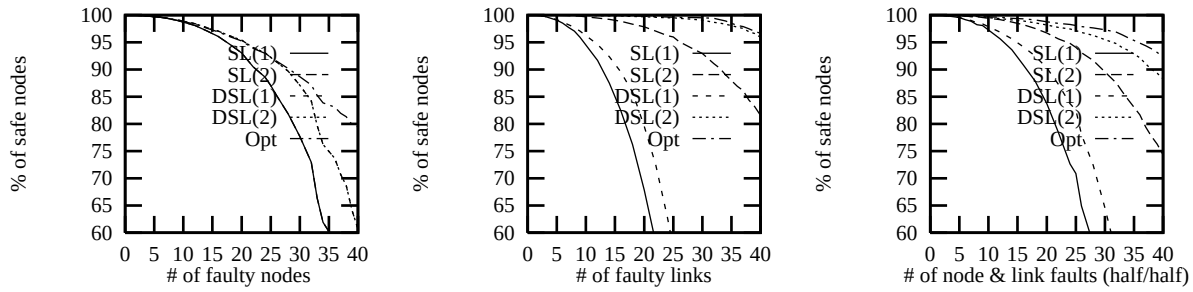

Figure 5: Simulation results.

$..., as_{i_{n-1}}) \geq (0, 0, 2, 3, ..., n-1)$, where $as_{i_0}, as_{i_1}, ..., as_{i_{n-1}}$ is the adjusted directed safety level of u, the global safety level is n; otherwise, if $(as_{i_0}, as_{i_1}, as_{i_2}, as_{i_3}, ..., as_{i_l}) \geq (0, 0, 2, 3, ..., l)$ and $as_{i_{l+1}} = l$, then $s = l + 1$.

The final safety status can be determined by exchanging the final directed safety levels among neighbors. A node is called safe if its global safety level is n; otherwise, it is unsafe. The calculation procedure is almost the same as the one for the directed safety level with $d = 1$ initialization, except for the initialization phase. Based on Theorem 2, the initialization process takes $O(n^2)$ for each node, therefore, both models have the same complexity.

3.5 Examples

For the regular safety level model, we calculate the safety level for each node under different initializations. Similarly for the directed safety level model, we calculate directed safety levels and the global safety level for each node under different initializations. Table 1 shows safety levels of nodes in Figures 2 (a) and (b) under DSL(1) and DSL(2). In the example of Figure 2 (a) with faulty nodes only, DSL(1) is the same as DSL(2). In the example of Figure 2 (b) with a mix of node and link faults, the order in terms of better accuracy in approximation is the following: SL(1), DSL(1), SL(2), and DSL(2). In Table 1, directed safety levels are dimensional-ordered, i.e., (s_0, s_1, s_2, s_3), whereas safety levels are arranged in a nondescending order, i.e., $(as_{i_0}, as_{i_1}, as_{i_2}, as_{i_3})$.

4 Simulation

A simulation study has been conducted based on the following variables (parameters): (1) n: the dimension of hypercubes. In our simulation, 9-cubes are considered. (2) f: the number of faults. f from 1 to n are considered. f's ($> n$) are selectively included. (3) Three mixes of node and link faults: all node faults, all link faults, and half link and half node faults. Five methods are considered: SL(1), SL(2), DSL(1), DSL(2), and global information: Opt (i.e., the corresponding safe node set is the r-node set). Percentage of safe nodes is collected for each case.

Figure 5 shows the percentages of safety nodes under three different distributions of faults. When there are node faults only, results are the same for SL(1) and SL(2) (similarly for DSL(1) and DSL(2)). Otherwise, the order in terms of increasing percentage of safe nodes is SL(1), DSL(1), SL(2), and DSL(2). Results for DSL(2) are very close to the optimal case (Opt), especially when there are link faults only. In all cases, DSL(2) outperforms SL(1) significantly, especially when the number of faults is large (relative to the cube size 9) and when there is a mix of node and link faults. DSL(2) curves stay very close to those for the optimal case, especially when all faults are link faults.

5 Conclusions

We have proposed a binomial-tree-based broadcasting in an injured hypercube using the directed safety level model. This model is an enhancement of the safety level model which is an approximation model for optimal broadcast capability of injured hypercubes. Simulation results have shown a significant improvement in terms of optimal broadcast capability in an injured hypercube using the directed safety level model, compared with the one using the safety level model, although both models exhibit the same level of asymptotic complexity.

The proposed model can be further extended. In general, $s_{k_1 k_2 \ldots k_c}$ can be used to represent the broadcast capability of a node in an $(n-c)$-subcube excluding dimensions k_1, k_2, ..., k_c. Therefore, each node has a set of safety status of different levels. $\{s_{k_1 k_2 \ldots k_c}\}$ is a set of level-c safety status with a cardinality of $\binom{n}{c}$ associated with each node. Simulation results in this paper show that there is not much room for improvement, for example, the gap between DSL(2) and Opt is relatively small for a relatively small number of faults. Therefore, the above extension is of theoretical interest only.

References

[1] G. M. Chiu and K. S. Chen. Efficient fault-tolerant multicast scheme for hypercube multicomputers. *IEEE Trans. on Parallel and Distributed Systems.* Vol.9, No. 10, 1998, 952-962.

[2] NCUBE Company. *NCUBE 6400 Processor Manual.* 1990.

[3] Silicon Graphics. Origin 200 and Origin 2000. Technical Report, Dec. 1996.

[4] Q. P. Gu and H. Tamaki. Routing a permutation in the hypercube by two sets of edge-disjoint paths. *Journal of Parallel and Distributed Computing.* Vol.44, 1997, 147-152.

[5] W.D. Hills. *The Connection Machine.* MIT Press, Cambridge, MA, 1985.

[6] S. L. Johnsson and C.-T. Ho. Optimal broadcasting and personalized communication in hypercubes. *IEEE Transactions on Computers.* 38, (9), Sept. 1989, 1249-1268.

[7] S. L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problems. *ACM Transactions on Programming Languages and Systems.* 4, July 1982, 382-401.

[8] T. C. Lee and J. P. Hayes. Routing and broadcasting in faulty hypercube computers. *Proc. 3rd Conf. on Hypercube Concurrent Computers and Applications.* Vol. I, Jan. 1988, 346-354.

[9] P.A. Nelson and L. Snyder. Programming paradigms for nonshared memory parallel computers. in The Characteristics of Parallel Algorithms, edited by L. H. Jamieson et al. 1987.

[10] P. Ramanathan, K. G. Shin, and R. W. Butler. Fault-tolerant clock synchronization in distributed systems. *Computer.* 23, (10), Oct. 1988, 1654-1657.

[11] J. Rattler. Concurrent processing: A new direction in scientific computing. *Proc. of AFIPS Conference.* Vol. 54, 1985, 157-166.

[12] X. Shen, Q. Hu, and W. Liang. Embedding k-ary complete trees into hypercubes. *Journal of Parallel and Distributed Computing.* Vol.24, 1995, 100-106.

[13] H. Sullivan, T. Bashkow, and D. Klappholz. A large scale, homogeneous, fully distributed parallel machine. *Proc. of the 4th Annual Symposium on Computer Architecture.* March 1977, 105-124.

[14] N. F. Tzeng and H. L. Chen. Structural and tree embedding aspects of incomplete hypercubes. *IEEE Trans. on Computers.* Vol.43, No. 12, 1994, 1434-1439.

[15] J. Wu. Safety level – an efficient mechanism for achieving reliable broadcasting in hypercubes. *IEEE Transactions on Computers.* 44, (5), May 1995, 702-705.

[16] J. Wu and E. B. Fernandez. Reliable broadcasting in faulty hypercube computers. *Microprocessing and Microgramming.* 39, 1993, 43-53.

[17] D. Xiang. Fault-tolerant routing hypercube multicomputers using local safety information. *IEEE Transactions on Parallel and Distributed Systems.* Vol.12, No. 9, 2001, 942-951.

[18] D. Xiang, A. Chen, and J. Wu. Reliable broadcasting for hypercube networks using local safety information. Accepted to appear in *IEEE Transactions on Reliability.*

[19] S. Q. Zheng and S. Latifi. Optimal simulation of linear multiprocessor architectures on multiply-twisted cube using generalized gray codes. *IEEE Transactions on Parallel and Distributed Systems.* Vol.7, No. 6, 1996, 612-619.