Tolerating Faults in Hypercubes Using Subcube Partitioning

Jehoshua Bruck, *Member, IEEE*, Robert Cypher, and Danny Soroker

Abstract— **We examine the issue of running algorithms on a hypercube which has both node and edge faults, and we assume a worst case distribution of the faults. We prove that for any constant c, an n-dimensional hypercube (n-cube) with n^c faulty components contains a fault-free subgraph that can implement a large class of hypercube algorithms with only a constant factor slowdown. In addition, our approach yields practical implementations for small numbers of faults. For example, we show that any regular algorithm can be implemented on an n-cube that has at most $n-1$ faults with slowdowns of at most 2 for computation and at most 4 for communication.**

To the best of our knowledge this is the first result showing that an n-cube can tolerate more than $O(n)$ arbitrarily placed faults with a constant factor slowdown.

Index Terms—**Fault-tolerance, hypercubes, parallel computing, reconfiguation, subcubes.**

I. Introduction

THE n-dimensional hypercube (n-cube) is one of the most popular interconnection topologies for parallel computers. Hypercube-based parallel machines are built and sold commercially and it is expected that they will continue to play an important role in the future. One of the most important issues related to such parallel machines is how they can compute in the presence of faults. In this paper we study how algorithms that are designed for fault-free hypercubes can be implemented on hypercubes that contain faults. The efficiency of the implementation will be measured by its *slowdown*, which is defined to be the algorithm's time requirements on the faulty hypercube divided by its time requirements on the fault-free hypercube. In the following discussion we will view a parallel computer as being a graph in which the nodes correspond to processors and the edges correspond to communication links.

The issue of computing with faulty hypercubes and related neworks has been addressed in several recent papers [1]–[8], [10], [11], [13]. Particularly notable is the result by Hastad, Leighton, and Newman [8]. They considered a faulty hypercube in which every node is faulty with constant probability $p < 1$ and the faults are independently distributed. They proved that, with high probability, the faulty hypercube can simulate a fault-free hypercube with only a constant factor slowdown. Thus, the hypercube is extremely tolerant of randomly distributed faults.

In this paper we study worst case distributions of faults. Several other researchers have examined this issue [3], [6], [10], [11]. One approach that has been studied is to locate a large fault-free subcube and to use that subcube to emulate the entire hypercube. However, it has been shown that in order to guarantee a constant factor slowdown, the n-cube must have only $O(\log n)$ faults [3], [11].

Our approach is to partition the hypercube into small subcubes, each of which has a small number of faults. More precisely, we guarantee that the majority of the nodes in each subcube form a fault-free connected component. We then show that the existence of such fault-free connected components can be used to obtain efficient implementations for a wide range of hypercube algorithms. We focus on two classes of hypercube algorithms, namely *regular* algorithms and *single-port* algorithms. In regular algorithms all processors communicate along the same dimension in each communication step, while in single-port algorithms each processor sends or receives at most one message during each communication step. The classes of regular and single-port algorithms include a large number of hypercube algorithms, including all of the algorithms in the classes Ascend and Descend as defined by Preparata and Vuillemin [12].

A different but related approach to hypercube fault-tolerance was studied by Chan and Lee [6]. They showed that the Benes routing algorithm can be implemented on an n-cube that has fewer than n faults with a factor of 9 slowdown. In contrast, we prove that any regular algorithm can be implemented on an n-cube that has fewer than n faults with slowdown factors of 2 for computation and 4 for communication. We also prove that for any constant c, an n-cube with n^c faults can implement any single-port algorithm with only a constant factor slowdown. To the best of our knowledge this is the first proof that an n-cube can tolerate more than $O(n)$ arbitrarily placed faults and still be guaranteed to implement a large class of algorithms with only a constant factor slowdown. Following the original appearance of this result [5], Aiello and Leighton obtained the same result for any hypercube algorithm, whether or not it is a single-port algorithm [1].

It will be assumed throughout this paper that all faults are static and are known. Both nodes and edges may be faulty. However, we will only consider node faults, as an edge fault can be tolerated by assuming that one of the nodes incident upon it is faulty. It will be assumed that faulty nodes can neither perform calculations nor route data.

Manuscript received June 24, 1991; revised December 4, 1991.

J. Bruck and R. Cypher are with IBM Almaden Research Center, San Jose, CA 95120-6099.

D. Soroker is with Shell Development Company, Houston, TX 77001-0481. This work was done while he was with IBM Almaden Research Center, San Jose, CA.

IEEE Log Number 9108215.

The rest of this paper is organized as follows. Definitions and notation are presented in Section II. Section III shows how faulty hypercubes can be partitioned into subcubes with large fault-free connected components, and Section IV uses these partitions to obtain efficient implementations of regular and single-port algorithms on faulty hypercubes. Section V presents conclusions and lists some open problems.

II. Definitions And Notation

We denote the set $\{0, 1, \cdots N-1\}$ by $[N]$. We will need to define several operators for multisets, which are collections of objects in which repetitions are allowed. Let T be a multiset and let X be any element in T. The *multiplicity of X in T*, denoted $mult(X,T)$, is the number of times X appears in T. The *multiplicity of T*, denoted $mult(T)$, is the maximum, over all $X \in T$, of $mult(X,T)$. Also, $set(T)$ is the set of all $X \in T$ (that is, $set(T)$ is the set obtained by removing duplicates from T). Given a multiset T and a set S such that for all $X \in S$, both $X \in T$ and $mult(X,T) = 1$, the *difference of T and S*, denoted $T \setminus S$, is the multiset obtained by removing the elements of S from T.

The n-cube is a graph that contains 2^n nodes, each of which is labeled with a unique n-dimensional vector of the form $X = (x_{n-1}, x_{n-2}, \cdots x_0)$, where for all i, $0 \leq i < n$, $x_i \in \{0,1\}$ (for notational simplicity, some vectors will be written without commas). Any two nodes in an n-cube are adjacent iff their vectors differ in exactly one dimension. The *Hamming weight* of a node X in an n-cube is $\sum_{i=0}^{n-1} x_i$. A node is *even* if it has an even Hamming weight, and it is *odd* otherwise. An m-dimensional subcube of an n-cube is denoted $S = (s_{n-1}, s_{n-2}, \cdots s_0)$, where exactly m of the s_i's are $*$'s and the remaining s_i's are either 0's or 1's. A node X is contained in S iff for all i, $0 \leq i < n$, either $x_i = s_i$ or $s_i = *$.

An *m-partition* of an n-cube is a partition of the n-cube into m-cubes. The partition is defined by identifying the m dimensions that are "internal" to the m cube. More formally, the set $P \subseteq [n]$ is an m-partition of an n-cube if $|P| = m$. An m-partition P can be viewed as a set of m-cubes where an m-cube S is in P iff for all i, $0 \leq i < n$, $s_i = *$ iff $i \in P$. Given a multiset of n-dimensional binary vectors T and an m-partition of an n-cube P, the *projection onto P of T*, denoted $proj(P,T)$, is the multiset of $(n-m)$-dimensional binary vectors obtained by removing from each vector in T the m dimensions specified by P. Let $\alpha(P,T)$ denote $mult(proj(P,T))$. Note that if T is a set of faults, $\alpha(P,T)$ is the maximum number of faults contained in any m-cube in P.

Recall that it will be assumed throughout that all faults are node faults. A node which is not faulty will be called *healthy*. Let F be a set of faulty nodes in an n-cube. We will say that an m-cube S *tolerates* F iff S contains a connected component of $2^{m-1}+1$ or more nodes, all of which are healthy. We will also say that an m-partition P *tolerates* F iff for every m-cube S in P, S tolerates F. Tolerant partitions are valuable because they guarantee that for any pair of adjacent m-cubes there is at least one edge connecting the large healthy connected components in the m-cubes.

Example: Let $F = \{(0011), (0010), (1011)\}$, let $P_1 = \{3\}$, let $P_2 = \{2,3\}$, and let $P_3 = \{1,2\}$. Given these definitions, $proj(P_1, F) = \{(011), (010), (011)\}$, $proj(P_2, F) = \{(11), (10), (11)\}$, $proj(P_3, F) = \{(01), (00), (11)\}$, $proj(P_1, proj(P_3, F)) = \{(1), (0), (1)\}$, $\alpha(P_1, F) = 2$, $\alpha(P_2, F) = 2$, and $\alpha(P_3, F) = 1$. Partition P_3 tolerates F, but neither P_1 nor P_2 tolerates F.

III. Fault-Tolerant Partitions

Our general approach to hypercube fault-tolerance consists of identifying a partition of the hypercube which tolerates the faults and then using this partition to implement regular or single-port algorithms efficiently. In this section we will show how to find such a partition given an arbitrary set of faulty nodes. We will consider three cases based on the number of faults that are tolerated.

Throughout this section, F will denote the set of faulty nodes in an n-cube. Our goal will be to find an m-partition P of the n-cube which tolerates F. Smaller values of m will lead to more efficient implementations of regular and single-port algorithms, so we will always attempt to minimize m.

A. $n-1$ Faults

Note that even if F consists of only a single fault, there are no m-partitions that tolerate F for which $m < 2$. As a result, we will only consider m-partitions for values of m that are 2 or larger. The following theorem shows that 2-partitions can be used whenever the n-cube contains fewer than n faults. The theorem depends on the following two lemmas, the first of which was proven by Chan and Lee [6].

Lemma 3.1: For all $n \geq 1$, given any set F of n or fewer faulty nodes in an n-cube, there exists a 1-partition P of the n-cube such that $\alpha(P,F) \leq 1$.

Lemma 3.2: For all $n \geq 2$, given any set F of $n-1$ or fewer faulty nodes in an n-cube, there exists a 2-partition P of the n-cube such that $\alpha(P,F) \leq 1$.

Proof: From Lemma 3.1 there exists at least one dimension i such that $\alpha(\{i\}, F) \leq 1$. Let p_1 be the largest such dimension i and let $F' = proj(\{p_1\}, F)$. Note that F' is a set of at most $n-1$ $(n-1)$-dimensional binary vectors, so from Lemma 3.1 there exists a dimension p_2 such that $\alpha(\{p_2\}, F') \leq 1$. Let $P = \{p_1, p_2\}$. Then $\alpha(P,F) = mult(proj(P,F)) = mult(proj(\{p_2\}, F')) \leq 1$. □

Example: Assume that a 5-cube has the following set of faults.

$$F = \{(00000), (01110), (01000), (01001)\}.$$

If $P = \{2,4\}$ then $proj(P,F) = \{(000), (110), (100), (101)\}$ and $\alpha(P,F) = mult(proj(P,F)) = 1$.

Theorem 3.3: For all $n \geq 2$, given any set F of $n-1$ or fewer faulty nodes in an n-cube, there exists a 2-partition P of the n-cube which tolerates F.

Proof: From Lemma 3.2 there exists a 2-partition P for which $\alpha(P,F) \leq 1$. Thus each 2-cube in P contains at most 1 faulty node. Therefore, each 2-cube in P contains a connected component of 3 healthy nodes and P tolerates F.

B. $2n - 5$ Faults

Theorem 3.3 showed that 2-partitions are capable of tolerating any set of $n-1$ or fewer faults. The following theorem shows that for n faults we may be forced to use m-partitions with $m \geq 4$.

Theorem 3.4: For all $n \geq 3$, there exists a set F of n faulty nodes in an n-cube such that no 2-partition tolerates F and no 3-partition tolerates F.

Proof: Let F be the set of n nodes with Hamming weight 1 and let Z be the node with Hamming weight 0. It is easily verified that any 2-cube containing Z does not contain a connected component of 3 or more healthy nodes, and any 3-cube containing Z does not contain a connected component of 5 or more healthy nodes. □

The following theorem demonstrates that 4-partitions are, in fact, able to tolerate significantly more than n faults. The theorem depends on the following lemma which was proven by Kleitman [9].

Lemma 3.5: Let S be an m-cube and let F_S be a set of faulty nodes in S. If S does not tolerate F_S then $|F_S| \geq \binom{m}{\lfloor \frac{m}{2} \rfloor}$.

Theorem 3.6: For all $n \geq 4$, given any set F of $2n-5$ or fewer faulty nodes in an n-cube, there exists a 4-partition P of the n-cube which tolerates F.

Proof: We will show that there exists a 4-partition P such that $\alpha(P,F) \leq 5$. It will then follow immediately from Lemma 3.5 that P tolerates F.

Let F_e and F_o denote the even nodes in F and the odd nodes in F, respectively. Assume without loss of generality that $|F_o| \leq n-3$. From Lemma 3.2 there must exist a 2-partition P_1 such that $\alpha(P_1, F_o) \leq 1$. Note that any 2-cube in P_1 contains only 2 even nodes, so $\alpha(P_1, F_e) \leq 2$ and $\alpha(P_1, F) \leq 3$.

Let the multiset $T = proj(P_1, F)$ and note that $mult(T) = \alpha(P_1, F) \leq 3$. Let the set $U = \{X \in T \mid mult(X,T) = 1\}$, and let the sets U_e and U_o denote the even nodes in U and the odd nodes in U, respectively. Assume without loss of generality that $|U_o| \leq |U_e|$. Let the multiset $V = T \setminus U$, let the multiset $W = T \setminus U_e$, and let the set $Y = set(W)$ (see Fig. 1). Because $Y = U_o \cup set(V)$, $|Y| \leq |U_o| + |set(V)|$. But $|U_o| \leq |U|/2$ and $|set(V)| \leq (2n - 5 - |U|)/2$ so $|Y| \leq n-3$. Therefore, from Lemma 3.2 there must exist a 2-partition P_2 such that $\alpha(P_2, Y) \leq 1$. Note that any 2-cube in P_2 contains only 2 even nodes, so $\alpha(P_2, U_e) \leq 2$, and note that $mult(W) \leq 3$, so $\alpha(P_2, W) \leq 3$. Therefore, $\alpha(P_2, T) \leq \alpha(P_2, U_e) + \alpha(P_2, W) \leq 5$.

The desired 4-partition P is obtained by merging the 2-partitions P_1 and P_2. More formally, $P = P_1 \cup P_2$ (assuming for the sake of notational simplicity that both dimensions in P_1 are greater than both dimensions in P_2). Then

$$\begin{aligned}\alpha(P,F) &= mult(proj(P,F)) \\ &= mult(proj(P_2, proj(P_1,F))) \\ &= mult(proj(P_2,T)) \\ &= \alpha(P_2,T).\end{aligned}$$

Thus, $\alpha(P,F) \leq 5$, which completes the proof. □

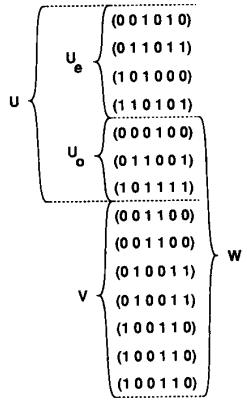


Fig. 1. Divisions of the multiset T.

C. Asymptotic Results

In this subsection we will prove that for any n-cube with a set of faults F where $|F|$ is polynomial in n, there exists an m-partition of the n-cube which tolerates F, where m is a constant. The proof is in two main steps. First we prove that we need only to consider the set $\mathcal{Z}$ of m-cubes that contain node 0. Then we prove by an averaging argument that if no m-cube in $\mathcal{Z}$ tolerates a set of faults F, then $|F|$ must be large. We first need several new definitions.

Let $\mathcal{P}(n,m)$ be the set of all m-partitions of an n-cube. Let n, m, and x be integers where $n \geq m \geq 0$ and $1 \leq x \leq 2^m$. For any set of nodes F in an n-cube, let

$$\beta(n,m,F) = \min_{P \in \mathcal{P}(n,m)} \alpha(P,F).$$

Thus, $\beta(n,m,F)$ is the maximum number of elements of F that are guaranteed to occur in at least one of the m-cubes, regardless of which m-partition is chosen. Let $\gamma(n,m,x)$ be the smallest y such that there exists a set of n-cube nodes F where $|F| = y$ and $\beta(n,m,F) \geq x$. In other words, $\gamma(n,m,x)$ is the smallest number of faults such that every m-partition of an n-cube contains an m-cube with at least x faults.

Also, let

$$ball(n,r) = \sum_{i=0}^{r} \binom{n}{i},$$

let $rad(n,x) = \max\{r \mid ball(n,r) \leq x\}$, and let $rem(n,x) = x - ball(n, rad(n,x))$. Thus, $ball(n,r)$ is the number of items in a ball of radius r in an n-cube, $rad(n,x)$ is the radius of the largest complete ball contained in a (possibly incomplete) ball of x items in an n-cube, and $rem(n,x)$ is the number of items in the outermost layer of an incomplete ball of x items in an n-cube (or 0 if the ball is complete). Finally, let

$$density(n,m,r) = \frac{\binom{n}{m}}{\binom{n-r}{m-r}} = \frac{n!(m-r)!}{(n-r)!m!}.$$

We will now show that a large number of faults are required to make every m-partition of an n-cube contain an m-cube

that has many faults. The following technical lemma shows that we need only to consider a set of m-cubes that contain a common node.

Lemma 3.7: Let $\mathcal{Z}$ be the set of all m-cubes in an n-cube that contain the node with Hamming weight 0. Given any set of n-cube nodes F there exists a set of n-cube nodes G such that $|G| = |F|$ and for all $Z \in \mathcal{Z}$, $|G \cap Z| \geq \beta(n, m, F)$.

Proof: For each partition $P \in \mathcal{P}(n, m)$, let S_P be an m-cube in P that contains the largest number of elements in F. Note that $|S_P \cap F| \geq \beta(n, m, F)$. Let $\mathcal{S} = \{S_P \mid P \in \mathcal{P}(n, m)\}$. We will show that we can transform F and $\mathcal{S}$, one dimension at a time, to obtain G and $\mathcal{Z}$. Let j be any dimension, $0 \leq j < n$. Given the dimension j, we can create F' and $\mathcal{S}'$ from F and $\mathcal{S}$ as follows.

1) $f' = (f'_{n-1}, \cdots f'_{j+1}, 0, f'_{j-1}, \cdots f'_0)$ is in F' iff $(f'_{n-1}, \cdots f'_{j+1}, 0, f'_{j-1}, \cdots f'_0)$ or $(f'_{n-1}, \cdots f'_{j+1}, 1, f'_{j-1}, \cdots f'_0)$ is in F.
2) $f' = (f'_{n-1}, \cdots f'_{j+1}, 1, f'_{j-1}, \cdots f'_0)$ is in F' iff $(f'_{n-1}, \cdots f'_{j+1}, 0, f'_{j-1}, \cdots f'_0)$ and $(f'_{n-1}, \cdots f'_{j+1}, 1, f'_{j-1}, \cdots f'_0)$ are in F.
3) $S' = (S'_{n-1}, \cdots S'_{j+1}, 0, S'_{j-1}, \cdots S'_0)$ is in $\mathcal{S}'$ iff $(S'_{n-1}, \cdots S'_{j+1}, 0, S'_{j-1}, \cdots S'_0)$ or $(S'_{n-1}, \cdots S'_{j+1}, 1, S'_{j-1}, \cdots S'_0)$ is in $\mathcal{S}$.
4) $S' = (S'_{n-1}, \cdots S'_{j+1}, *, S'_{j-1}, \cdots S'_0)$ is in $\mathcal{S}'$ iff S' is in $\mathcal{S}$.

Note that Rules 1 and 2 above compare each pair of nodes that are neighbors across dimension j. If exactly one of them is faulty, the fault is moved to the node whose label has a 0 in dimension j. Rule 3 puts an m-cube whose label has a 0 in dimension j into $\mathcal{S}'$ if and only if either it or its neighbor across dimension j is in $\mathcal{S}$. Rule 4 puts an m-cube from $\mathcal{S}$ into $\mathcal{S}'$ if and only if j is one of the internal dimensions of the m-cube. Note that Rules 3 and 4 transform each m-cube in $\mathcal{S}$ into an m-cube in $\mathcal{S}'$ with the same set of internal dimensions.

It is easy to verify that $|F'| = |F|$ and for all $S' \in \mathcal{S}'$, $|F' \cap S'| \geq \beta(n, m, F)$. As a result, the procedure which produced F' and $\mathcal{S}'$ from F and $\mathcal{S}$ can be applied iteratively for all dimensions j, $0 \leq j < n$, to obtain G and $\mathcal{Z}$. □

Theorem 3.8: For any n, m, and x where $n \geq m \geq 0$ and $1 \leq x \leq 2^m$, $\gamma(n, m, x) \geq ball(n, r) + rem(m, x) \cdot density(n, m, r + 1)$ where $r = rad(m, x)$.

Proof: By contradiction. Assume the claim is false, in which case there exist n, m, x, and $r = rad(m, x)$ such that $\gamma(n, m, x) < y$ where $y = ball(n, r) + rem(m, x) \cdot density(n, m, r + 1)$. Therefore, there exists a set of n-cube nodes F where $|F| < y$ and $\beta(n, m, F) \geq x$. Let $\mathcal{Z}$ be the set of all m-cubes in an n-cube that contain the node with Hamming weight 0. From Lemma 3.7 it follows that there exists a set of n-cube nodes G where $|G| < y$ and for all $Z \in \mathcal{Z}$, $|Z \cap G| \geq x$.

For any n-cube node g, let $hits(g) = |\{Z \in \mathcal{Z} \mid g \in Z\}|$. Note that if node g has a Hamming weight w where $w > m$ then $hits(g) = 0$, while if $w \leq m$ then $hits(g) = \binom{n-w}{m-w}$. Therefore,

$$x \cdot \binom{n}{m} \leq \sum_{Z \in \mathcal{Z}} |Z \cap G| = \sum_{g \in G} hits(g).$$

Clearly, in order to maximize the total number of hits for G while minimizing its size, G should consist of a set of items with minimal Hamming weights. But a ball of radius r centered on the node with Hamming weight 0 has $ball(n, r)$ items and only $ball(m, r) \cdot \binom{n}{m}$ hits. Therefore, G obtains at least

$$(x - ball(m, r)) \cdot \binom{n}{m} = rem(m, x) \cdot \binom{n}{m}$$

hits from nodes with Hamming weight $r + 1$ or more. As a result,

$$\begin{aligned} |G| &\geq ball(n, r) + \frac{rem(m, x) \cdot \binom{n}{m}}{\binom{n-r-1}{m-r-1}} \\ &= ball(n, r) + rem(m, x) \cdot density(n, m, r + 1) \\ &= y \end{aligned}$$

which is a contradiction. □

Let

$$\phi(n, m) = ball(n, r) + rem\left(m, \binom{m}{\lfloor m/2 \rfloor}\right) \cdot density(n, m, r+1)$$

where

$$r = rad\left(m, \binom{m}{\lfloor m/2 \rfloor}\right).$$

Combining Theorem 3.8 and Lemma 3.5 yields the following result.

Theorem 3.9: For all $n \geq m \geq 0$, given any set F of fewer than $\phi(n, m)$ faulty nodes in an n-cube, there exists an m-partition of the n-cube which tolerates F.

For example, $\phi(n, 2) = 1 + n/2$. Thus, Theorem 3.9 shows that any set of $n/2$ or fewer faults can be tolerated by some 2-partition, which is weaker than the bound of $n - 1$ or fewer faults given by Theorem 3.3. Also, note that $\phi(n, 4) = 1 + (n^2 + 11n)/12$. Thus Theorem 3.9 shows that any set of $(n^2 + 11n)/12$ or fewer faults can be tolerated by some 4-partition, which is stronger than the bound of $2n - 5$ or fewer faults given by Theorem 3.6 for all values of n. However, the proof of Theorem 3.9 is nonconstructive, so all $\Theta(n^4)$ 4-partitions may have to be tested in order to find one which tolerates F, while the proof of Theorem 3.6 implies a more efficient algorithm for determining such a 4-partition.

IV. Implementing Regular and Single-Port Algorithms

In this section we will apply the partitioning results from the previous section to obtain efficient implementations of regular and single-port algorithms in the presence of faults. We will concentrate on the most efficient case, in which there are fewer than n faults, and on asymptotic bounds.

A. $n-1$ Faults

Using a technique which involves partitioning into subcubes, Chan and Lee have shown that the Benes routing algorithm can be implemented on an n-cube with less than n faulty nodes with a factor of 9 slowdown [6]. They accomplish this by partitioning the n-cube into subcubes based on the dimension that is being used for communication. Theorem 4.1 improves the slowdown and generalizes their result, as it can be used to implement arbitrary regular algorithms. We will assume that in a single step of the regular algorithm, pairs of processors which differ in a given dimension can exchange a packet. The regular algorithm will be implemented on a faulty n-cube in which each processor can both send and receive a single packet in one time step.

Theorem 4.1: Any regular hypercube algorithm can be implemented on an n-cube that has at most $n-1$ faulty nodes with a factor of 2 slowdown for computation and a factor of 4 slowdown for communication.

Proof: Let F be the set of faulty nodes. From Lemma 3.2 there must exist a 2-partition P such that $\alpha(P,F) \leq 1$. Each 2-cube in P is a set of 4 nodes connected as a square, where at most one of the nodes is faulty. We will use partition P to simulate a healthy hypercube by the faulty hypercube. The nodes of the simulated hypercube will be called *virtual* nodes and the nodes of the faulty hypercube will be called *actual* nodes. Virtual nodes which correspond to healthy actual nodes will be called *vh-nodes*, and virtual nodes which correspond to faulty actual nodes will be called *vf-nodes*. Each vh-node is simulated by the corresponding actual node, while each vf-node is simulated by the node which is diagonally opposite it in its 2-cube in P. Because each actual node is responsible for at most 2 virtual nodes, all computations can be performed with a factor of 2 slowdown.

Each communication operation is implemented in at most four steps as described below. We call a message that goes between a pair of vh-nodes a vh-message, and a message that goes to or from a vf-node a vf-message. Recall that all virtual nodes communicate along the same dimension, which will be called the *selected* dimension.

First, consider the case in which the selected dimension is local to the 2-cubes in P. In this case, fault-free 2-cubes perform the communication in one time step. Those 2-cubes that have a fault first send the vh-messages in one time step. Then the vf-messages are sent in one time step, as the actual nodes that are communicating are adjacent.

Now consider the case in which the selected dimension is external to the 2-cubes in P. In this case, pairs of adjacent 2-cubes exchange messages. We focus on one such pair, denoted $\{A,B\}$. We number the nodes in each 2-cube 1 through 4 in clockwise order, starting at the upper left corner (see Fig. 2). There are three cases to consider.

If neither A nor B has a fault, all messages are vh-messages and they are implemented in one time step.

If A and B contain a total of exactly one fault, then by symmetry we can assume that the faulty node is A_1. In this case, the vh-messages are first sent in one time step. Then the vf-messages are sent in three time steps along the paths

$$A_3 \rightarrow B_3 \rightarrow B_2 \rightarrow B_1$$
$$B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow A_3.$$

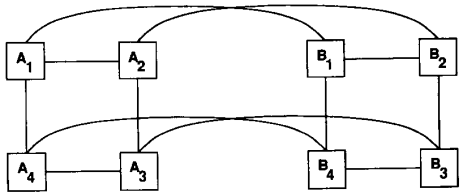


Fig. 2. Node labels in communicating 2-cubes A and B.

If A and B each contain a fault, then by symmetry we can assume that either A_1 and B_1 are faulty, or that A_1 and B_2 are faulty, or that A_1 and B_3 are faulty. If A_1 and B_1 are faulty, the vh-messages are sent in the first time step and then the vf-messages take the paths $A_3 \rightarrow B_3$ and $B_3 \rightarrow A_3$. If A_1 and B_2 are faulty, the vh-messages are sent in the first time step and then the vf-messages take the paths

$$A_3 \rightarrow B_3 \rightarrow B_4 \rightarrow B_1$$
$$B_1 \rightarrow B_4 \rightarrow B_3 \rightarrow A_3$$
$$B_4 \rightarrow A_4 \rightarrow A_3 \rightarrow A_2$$
$$A_2 \rightarrow A_3 \rightarrow A_4 \rightarrow B_4.$$

Finally, if A_1 and B_3 are faulty, the vh-messages take the paths

$$A_2 \rightarrow A_2 \rightarrow A_2 \rightarrow A_2 \rightarrow B_2$$
$$B_2 \rightarrow B_2 \rightarrow B_2 \rightarrow B_2 \rightarrow A_2$$
$$A_4 \rightarrow B_4 \rightarrow B_4 \rightarrow B_4 \rightarrow B_4$$
$$B_4 \rightarrow A_4 \rightarrow A_4 \rightarrow A_4 \rightarrow A_4$$

and the vf-messages take the paths

$$A_3 \rightarrow A_2 \rightarrow B_2 \rightarrow B_1 \rightarrow B_1$$
$$B_1 \rightarrow B_2 \rightarrow A_2 \rightarrow A_3 \rightarrow A_3$$
$$A_3 \rightarrow A_3 \rightarrow A_4 \rightarrow B_4 \rightarrow B_1$$
$$B_1 \rightarrow B_1 \rightarrow B_4 \rightarrow A_4 \rightarrow A_3.$$

It is straightforward to verify that no node either receives or sends more than one message at any given time. □

B. Asymptotic Results

We will now show that the results of the previous section can be used to implement any single-port algorithm on an n-cube that has $n^{O(1)}$ faulty nodes with only a constant factor slowdown. We will divide the n-cube into m-cubes, where $m = O(1)$, and we will use a single healthy node in each m-cube to simulate the actions of the remaining nodes in the same m-cube. The use of a single healthy node per m-cube will simplify the presentation, but it should be noted that in practice many healthy nodes could be used to improve the performance. In order to facilitate communication between neighboring m-cubes, we will require that the single healthy node be in a connected component of healthy nodes that consists of a majority of the nodes in the m-cube.

Theorem 4.2: Any single-port algorithm can be implemented on an n-cube that has $n^{O(1)}$ faulty nodes with only a constant factor slowdown.

Proof: Let F be the set of faulty nodes. First, calculate the smallest value of m such that $\phi(n,m) > |F|$. It is straightforward to show that $m = O(1)$.

From Theorem 3.9 there must exist an m-partition P which tolerates F. Note that the desired m-partition P can be found in $n^{O(1)}$ time sequentially, as there are $n^{O(1)}$ possible m-partitions and each can be checked in $n^{O(1)}$ time.

In each m-cube in P, designate the healthy nodes in the largest connected component of healthy nodes as the *active* nodes for that m-cube. Also, in each m-cube select one of the active nodes to be the *primary* node for that m-cube. In order to implement a single-port algorithm, use the primary node in each m-cube to perform the calculations for all of the nodes in the m-cube. Because $m = O(1)$, the primary node simulates only a constant number of nodes and the slowdown in computation time is constant.

The communication operations of the single-port algorithm are implemented by using only the active nodes. Specifically, when the single-port algorithm communicates between two nodes in the same m-cube, the primary node for that m-cube is responsible for both nodes and no actual communication is required. When the single-port algorithm communicates between nodes in different m-cubes, the primary nodes of those different m-cubes must communicate. Assume that a node in m-cube A has to send a message to a node in m-cube B. First, note that A and B must be adjacent m-cubes and that there must exist a link between one of the active nodes in A and one of the active nodes in B (because a majority of the nodes in each m-cube are active). Thus, the primary node in A sends the message through the active nodes in A, across the link between active nodes in A and B, and through the active nodes in B to the primary node in B. Because each node in a single-port algorithm only sends and receives a single message in unit time, and because $m = O(1)$, all of the communications that are required to implement a single operation of the single-port algorithm require only constant time and constant memory per node. □

V. Conclusions and Open Problems

We have presented a new technique to tolerate faults in an n-cube in a worst case scenario. We obtained the best known results in terms of the number of faults assuming a constant factor slowdown in communications and computation. In particular, our technique can be used to handle any number of faults that is polynomial in the dimension n. We also presented a practical implementation of regular algorithms on n-cubes with at most $n-1$ faults.

There are several open problems related to the results given here. One open problem is to find an efficient algorithm for constructing a tolerant m-partition when there are over $2n-5$ faults. Currently, finding such a partition requires $\Omega(\binom{n}{m})$ sequential time. Another important open problem is to find practical implementations of single-port algorithms on n-cubes with n or more faults.

References

[1] B. Aiello and T. Leighton, "Coding theory, hypercube embeddings, and fault tolerance," in *Proc. 3rd Annu. ACM Symp. Parallel Algorithms Architectures*, 1991, pp. 125–136.

[2] F. Annexstein, "Fault tolerance of hypercube-derivative networks," in *Proc. 1st Annu. ACM Symp. Parallel Algorithms Architectures*, 1989, pp. 179–188.

[3] B. Becker and H.U. Simon, "How robust is the n-cube?," *Inform. Computat.*, vol. 77, pp. 162–178, 1988.

[4] J. Bruck, "Optimal broadcasting in faulty hypercubes via edge-disjoint embeddings," IBM Res. Rep., RJ7147, 1989.

[5] J. Bruck, R. Cypher, and D. Soroker, "Running algorithms efficiently on faulty hypercubes," in *Proc. 2nd Annu. ACM Symp. Parallel Algorithms Architectures*, 1990, pp. 37–44.

[6] M.Y. Chan and S.J. Lee, "Fault-tolerant permutation routing in hypercubes," Univ. Texas at Dallas Tech. Rep., UTDCS-5-90.

[7] D. Dolev, J.Y. Halpern, B. Simons, and R. Strong, "A new look at fault-tolerant network routing," *Inform. Computat.*, vol. 72, no. 3, pp. 180–196, Mar. 1987.

[8] J. Hastad, T. Leighton, and M. Newman, "Fast computation using faulty hypercubes," in *Proc. 21st Annu. ACM Symp. Theory Comput.*, 1989, pp. 251–263.

[9] D. Kleitman, "On the problem by Yuzvinsky on separating the n-cube," *Discrete Math.*, vol. 60, pp. 207–213, 1986.

[10] T. Leighton and B. Maggs, "Expanders might be practical: Fast algorithms for routing around faults on multibutterflies," in *Proc. 30th Annu. IEEE Symp. Foundations Comput. Sci.*, 1989, pp. 384–389.

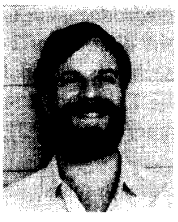
[11] M. Livingston, Q. Stout, N. Graham, and F. Harary, "Subcube fault-tolerance in hypercubes," Tech. Rep. CRL-TR-12-87, Univ. Michigan Comput. Res. Lab., Sept. 1987.

[12] F.P. Preparata and J. Vuillemin, "The cube-connected cylces: A versatile network for parallel computation," *Commun. ACM*, vol. 24, no. 5, pp. 300–309, May 1981.

[13] M.O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. ACM*, vol. 36, no. 2, pp. 335–348, Apr. 1989.

Jehoshua Bruck (S'86–M'89) was born in Haifa, Israel, on April 19, 1956. He received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion–Israel Institute of Technology in 1982 and 1985, respectively, and the Ph.D. degree in electrical engineering from Stanford University in 1989.

From 1982 to 1985 he was with the IBM Haifa Scientific Center, Israel. In March 1989, he joined IBM Research Division at the Almaden Research Center, San Jose, CA, where he currently manages the Foundations of Massively Parallel Computing group. His research interests include parallel computing, fault-tolerant computing, error-correcting codes, and neural networks.



Robert Cypher was born in Schenectady, NY, in 1959. He received the B.S. degree in mathematical sciences from Stanford University in 1982 and the M.S. and Ph.D. degrees in computer science from the University of Washington in 1987 and 1989, respectively.

He is currently a Research Staff Member of the IBM Almaden Research Center and a Consulting Assistant Professor in the Stanford University Computer Science Department. He is interested in both the theoretical and practical aspects of parallel processing. He has done research in parallel algorithms for image processing, computational geometry, sorting, and data routing. He is also interested in VLSI, signal processing, fault-tolerance, and the design of interconnection networks.

Danny Soroker was born in Jerusalem, Israel, in 1959. He received the B.Sc. degree in computer engineering and the M.Sc. degree in electrical engineering from the Technion–Israel Institute of Technology, Haifa, in 1981 and 1983, respectively, and the Ph.D. degree in computer science from the University of California, Berkeley, in 1987.

From 1988 to 1990 he was a Visiting Scientist at IBM Almaden Research Center, San Jose, CA. In 1990 he joined the Computer Science Department of Shell Development Company, where he is currently an Associate Research Computer Scientist at the Bellaire Research Center, Houtston, TX. His current research interests encompass many aspects of parallel computing, including algorithms, networks, programming, and fault tolerance.

Dr. Soroker is a member of the Association for Computing Machinery and the IEEE Computer Society.