Fault Tolerance of Hypercube-based Multicomputers: A Survey

Khalid M. Al-Tawil * Roland N. Ibbett

Department of Computer Science
The University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ, UK.

Abstract

The problem of tolerating faulty processors or links in hypercubes has been studied by many researchers, either by using spares or by reconfiguration. Massively parallel computers, using thousands of processors, will be the future trend for producing tremendous computational power. However, in the current technology, if one processor fails, the entire system may fail. A major drawback of hypercubes is that a single processor failure may destroy the whole network. The existence of a large number of components in such systems makes them subject to failures. As the probability of any one or more processors failing in such a complex system is large, building some fault-tolerance feature into them becomes extremely important. Fault tolerance in highly parallel computers is important for achieving high-performance reliable computing. This manuscript is mainly a survey of fault tolerance and related issues of hypercube-based multicomputers.

Key Words - *Fault Tolerance, Multicomputers, Hypercubes, Reconfiguration, Broadcasting*

*Currently on research visit from Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia, e-mail: khalid@ccse.kfupm.edu.sa

1 Introduction

Hypercubes are attracting much attention from both parallel processing and communication [17] areas. Research in this area has revealed a number of challenges that need to be addressed for building high performance reliable multicomputer systems. A major drawback of hypercubes is that a single processor failure may destroy the whole network. The existence of a large number of components in such systems makes them subject to failures. As the probability of any one or more processors failing in such a complex system is large, building some fault-tolerance features into them becomes extremely important. Hypercube networks are attractive structures for parallel processing because of their symmetry and regularity. They are highly regular, symmetric, and recursive structures, having nice properties [66, 78, 82]. A comparative study of topological properties of hypercubes and star graphs is presented in [39]. A comparative study of fault-tolerant computation on the star graph and hypercubes can be found in [13]. A major reason that the hypercubic networks are so commonly used in parallel machines is that they can efficiently simulate any bounded-degree communication network.

Fault tolerance in hypercube networks has received substantial attention from both parallel processing and fault tolerance research communities. As a result, several approaches for solving the problem of tolerating faults in hypercubes have been proposed. The problem of embedding binary trees in hypercubes has been presented in several papers, such as [21, 76, 93, 101]. Reconfiguration of binary trees in hypercubes also has been studied by many researchers. The problem of finding $(n-1)$-binary tree in faulty hypercubes is studied in [30]. A distributed scheme for reconfiguring binary trees in faulty hypercubes is presented in [102]. Fault tolerance of binary tree embedding in hypercubes in a way that minimizes congestion, dilation, and slowdown is described in [42]. Reconfiguration of spanning trees in faulty hypercubes is presented in [6]. Mapping general trees and graphs into the hypercube is presented in [95]. Numerous researchers have looked at the problem of reconfiguration in hypercubes using hardware approaches. Other researchers have proposed software approaches to reconfigure hypercubes. One approach is to identify the maximum size of a fault-free subcube in the hypercube [18, 43] and to run the parallel application on that smaller subcube. Another software approach would be to distribute the work allocated to the faulty processor to some other processor or set of processors.

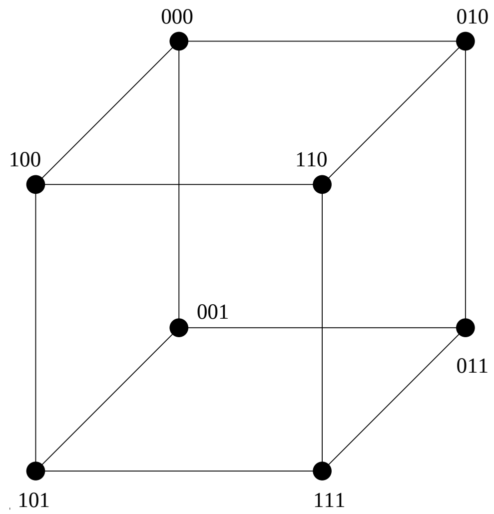

Figure 1: A 3-dimensional hypercube with eight nodes.

1.1 Definitions

Definition 1.1 Hypercube *of dimension n has* $N = 2^n$ *nodes. Node i has address* $(i_{n-1}i_{n-2}\cdots i_0)$ *with address bits numbered from 0 through* $n-1$. *The mth bit corresponds to dimension m in the Boolean space. Two nodes i and j are connected by an edge in the hypercube if their addresses differ by only one bit. The edge* (i, j) *where* $i \oplus j = 2^m$ *(m ranges from 0 to* $n-1$*) is a connection through dimension m.*

Figure 1 shows a hypercube of dimension 3 with eight nodes. An N-node hypercube can be constructed from two $\frac{N}{2}$-node hypercubes by connecting the ith node in one $\frac{N}{2}$-node hypercube to the ith node in the other one for $0 \leq i < \frac{N}{2}$. In addition to the simple recursive structure, the hypercube possesses many nice properties. For example, it is node and edge symmetric, and it has low diameter ($logN$). These nice properties make hypercubes suitable for embedding and reconfiguration, as we will see later. The edges of the hypercube can be partitioned according to the dimensions that they traverse. An edge is called a dimension k edge if it links two nodes that differ in the kth bit position. For example, the edge $(001, 101)$, in Figure 1, is a dimension 2 edge because the nodes 001 and 101 differ in dimension 2.

Definition 1.2 Partial failure *is a failure where the processor can continue to handle communication (i.e. loss of the main processor, but continuing operation of the communication coprocessor).*

Definition 1.3 Dilation *is normally defined as the number of links of the logical replacement of a physical link caused by failure. A dilation of edge e is the length of the path between its hypercube nodes.*

Definition 1.4 Extra-dilation *is defined as the maximum number of extra links (over the shortest path) necessary to reconstruct a path between two nodes when reconfiguring the hypercube.*

Definition 1.5 Congestion *of a link (node) is the number of logical paths that uses each fault-free physical link (node) in a reconfigured hypercube. The congestion of an edge e (node n) is the number of paths that uses e (n).*

Definition 1.6 Extra-congestion *of a link (node) is the maximum number of extra paths that uses a hypercube edge (node) as a result of the reconfiguration process.*

Definition 1.7 Fault coverage *is the percentage of tolerated faults, which is the ratio between the number of tolerated fault combinations and the total number of possible fault combinations.*

Definition 1.8 Network Reliability (NR)*: The reliability measure assuming that the system works as long as all nodes are working and connected.*

Definition 1.9 Task-Based Reliability (TBR)*: The reliability measure assuming that the system works as long as some minimum number of connected nodes are available in the system for task execution.*

Definition 1.10 Terminal Reliability (TR)*: The reliability measure assuming that the system works as long as two specified nodes are working and connected.*

Definition 1.11 Subcube Reliability (SR)*: The reliability measure assuming that the system works as long as some functional minimum degree subcube exists.*

Definition 1.12 Expansion *is the ratio of number of nodes of the host graph to the number of nodes of the mapped graph.*

Definition 1.13 Load *is the maximum over all processors of the number of nodes that are mapped to the same processor.*

Definition 1.14 Virtual channel *is a logical entity associated with a physical link used to distinguish multiple streams using the same physical link.*

Definition 1.15 Static routing *is routing messages along a static (i.e. fixed) path defined before the computation starts. The path a message takes is determined by the source-destination pair without considering the network state (e.g., busy or faulty).*

This kind of routing is also called oblivious or deterministic routing [102]. This kind of routing does not use the bandwidth properly, and messages may be blocked even if alternative paths exist.

Definition 1.16 Dynamic routing *is routing messages making use of the alternative paths between communicating processors. For a given source and destination, the path taken depends on the network state taking care of the faulty and congested channels.*

This routing makes efficient use of the bandwidth providing more fault tolerance. This kind of routing is also called adaptive routing [102].

2 Fault Tolerance in Hypercubes

Numerous researchers have looked at the problem of reconfiguration in hypercubes using hardware approaches. Other researchers have proposed software approaches to reconfigure hypercubes. One approach is to identify the maximum size fault-free subcube in the hypercube and to run the parallel application on that smaller subcube. Another software approach would be to distribute the work allocated to the faulty processor to some other processor or set of processors. In this section, we discuss both the hardware and software approaches for solving the problem of fault tolerance in hypercubes.

2.1 Hardware Approach

Fault tolerance in hypercubes can be achieved by adding extra hardware (e.g., nodes or links) to the standard hypercube structure. Such hypercubes are called non-degradable hypercubes, because they achieve the same hypercube performance through the use of extra hardware [49].

Rennels suggested two schemes that use spare nodes to tolerate node failures in a hypercube. In both schemes, the nodes are built with extra ports to communicate with spares. In the first scheme, an $n-$cube consists of subcubes, each containing 2^m nodes, where $s + m = n$. One spare node is provided for each m-cube, and the nodes are connected through their extra port to the spare. Whenever a primary node fails, the spare is connected to the m neighboring nodes in the subcube and to the s neighbors in other neighboring subcubes. Two crossbar switches per spare are used for reconfiguration. The Connection Crossbar (CCB) has $2^m + s$ inputs and n outputs. The outputs are connected to the spare. Out of the $2^m + s$ inputs, 2^m are connected to the primary processors. Each of the remaining s inputs comes from one output of the Relay Crossbar (RCB) of each of the other s subcubes to which the module is connected. The RCB has 2^m inputs and s outputs. The inputs come from all the nodes in the subcube and each output goes to one of the s subcubes to which the subcube is connected. On failure in a subcube, its CCB connects the spare processor to the m neighbours of the failed processor. The RCB in each neighbouring subcube reconfigures so that the neighbour of a failed processor is now connected to the replacement of the failed processor. The various components of this scheme are shown in Figure 2. The reliability of this scheme is calculated as follows. Each subcube of order m can tolerate at most one fault. There are 2^s such subcubes. Therefore the system reliability of this scheme is:

$$RB_{m,s} = [r^{2^m} + 2^m r^{2^m}(1 - r)c]^{2^s} \tag{1}$$

where r is the reliability of each node, and c is the fault coverage. For high reliability systems, another scheme is proposed in which each module is made internally redundant. The redundancy is in the form of spare memory modules, processors etc. One such module is shown in Figure 3. Four processors and a spare are connected by a high speed bus and packaged as a multiprocessor. Four ports are provided to communicate with other multiprocessors. To realize a 64-node cube, for example, 16 multiprocessors are interconnected as a 4-cube. The major disadvantage of this scheme is the large hardware cost involved in the design of the crossbar switches.

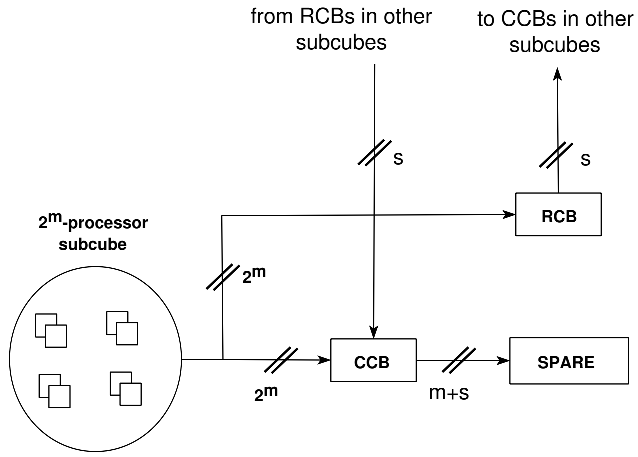

Figure 2: A subcube in Rennels' scheme.

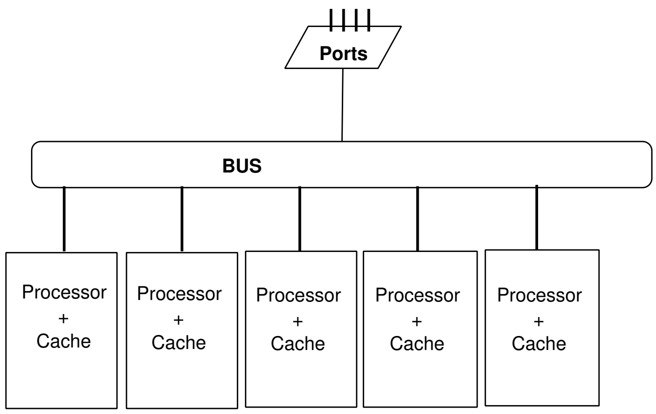

Figure 3: Redundant multiprocessor for use in large array.

A similar scheme for reconfiguring faulty processors is proposed by Banerjee et al. [16]. The scheme involved the detection and location of faulty processors concurrently with the actual execution of parallel applications on the hypercube. Algorithmic-based fault tolerance for matrix multiplication and FFT is discussed. The algorithms are modified to include system level checks. The simulation was done using the Intel 16-node Hypercube. The problem of placing spare processors and links in an augmented hypercube topology is presented. Reconfiguring faulty processors was done by using two spare processors for every eight normal processors of the original hypercube. Each normal hypercube node has a degree of $n+1$, and the degree of each spare node is $n+2$. Redundant nodes and links are added to the hypercube, increasing the dimension of the cube by one. However, the regularity of the structure is maintained. Two other reconfiguration strategies for hypercube multicomputer architectures under failures are presented in [14]. The first one used spare processors attached to certain processors in the hypercube. The second scheme placed spare processors between specific links in the hypercube. Both of the reconfiguration schemes described in the paper used the notion of replacing physical links by logical links (i.e. dilation), and hence suffer some performance degradation. In the first approach, there are two kinds of nodes, the P nodes and S nodes. The P and the S nodes are embedded in the hypercube in such a way that each P node is adjacent to exact exactly one S node in the cube, as shown in Figure 4a. This is called *perfect embedding.* The architectures of P nodes and S nodes are shown in Figure 4b and Figure 4c respectively. The P node consists of a computation processor (CPU) connected through an internal bus to a local memory and message routing logic consisting of the DMA unit and a $(d+1)\times(d+1)$ crossbar switch for a 2^d processor hypercube. The S-node consists of two copies of CPU and memory connected to two internal busses. The two processing units share the DMA and message routing logic. One of the processors is active under normal conditions; the other is a standby spare. When any processing element fails, either within the S node or in a nearby P node, the spare processor/memory/bus from the corresponding S node replaces it. A centralized algorithm that runs on the host computer to reconfigure the hypercube under multiple faults is presented. This algorithm used weighted bipartite matching to allocate spares to faulty nodes. In the second case, the spares are inserted at links between nodes. The nodes are first collapsed in any dimension, then the algorithm in the first approach is applied. Finally, the collapsed cube is expanded and for each spare node, a spare is inserted. The reliability of the first scheme is good for relatively small number of faults. As the number of faults increases, the reliability goes down, as each primary node is covered by only one spare. A major drawback of this scheme is that nodes do not have

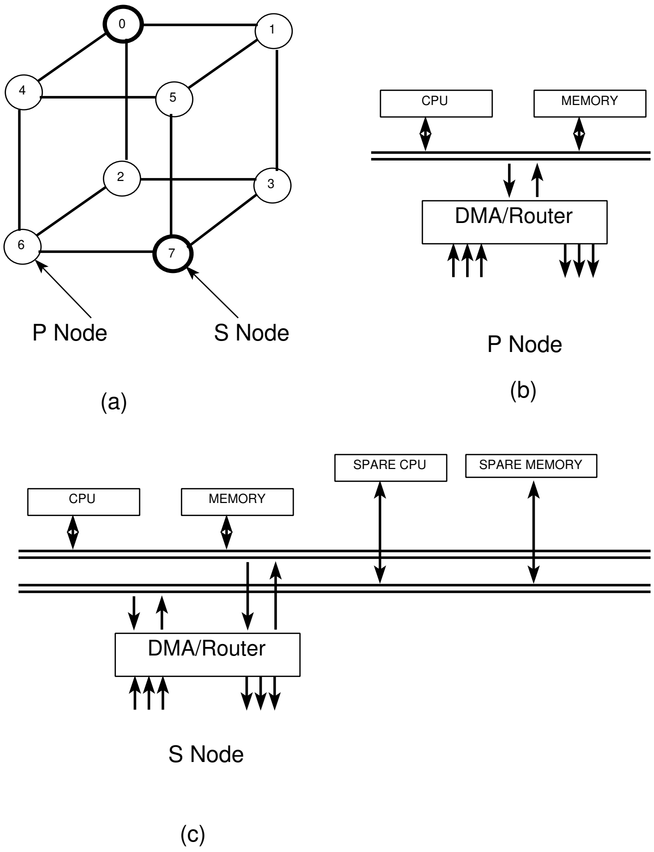

Figure 4: The architecture of P and S nodes.

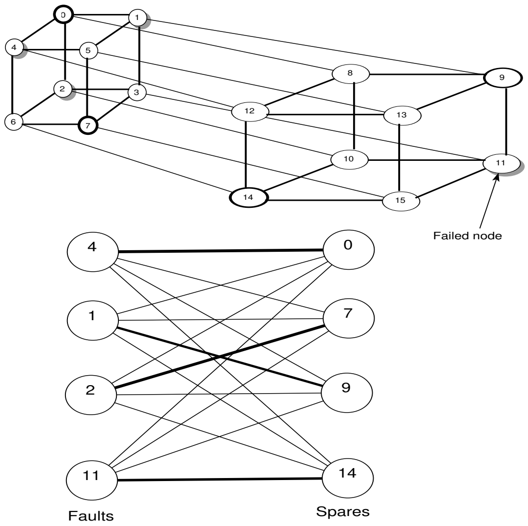

Figure 5: Reconfiguration in Banerjee's scheme.

the same structure. Spare nodes consists of two CPUs; however, normal nodes have only one CPU each. The simulation results for both schemes showed that the maximum dilation is five and the maximum congestion is six for a hypercube of dimension 4. Design and evaluation of these two strategies for reconfiguring hypercubes under faults is also discussed in [15, 70, 71].

Several proposals for fault tolerance in hypercubes rely on the concept of fault tolerant basic blocks. Chau and Liestman [27] proposed a fault-tolerant-hypercube architecture based on modular construction. Each fault-tolerant module (FTM) consists of 2^m active nodes and k spare nodes. Decoupling networks are used to realize the connections within and among the FTMs. Figure 6 shows k levels of decoupling networks being used to connect 2^m primary nodes (m=2 in the figure) and k spare nodes from one FTM to another. To realize intra-modular connections, m groups of k level decoupling networks are used. Figure 7 shows how a 2-dimensional hypercube can be constructed using two decoupling networks. When $m = 2$ and

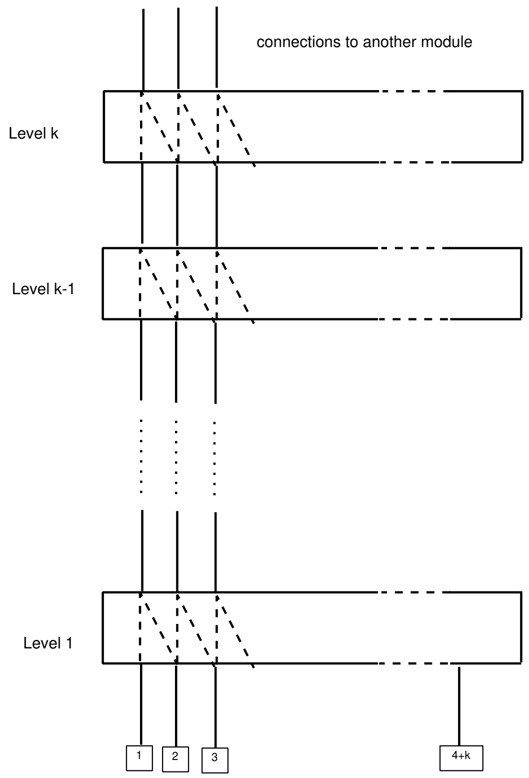

Figure 6: Decoupling networks used to interconnect fault-tolerant modules.

with k spares, the intra-modular connections can be realized using soft switches as shown in Figure 8. When a fault occurs, the soft switches bypass the failed node. At the same time, the

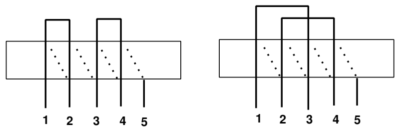

Figure 7: Using 2 decoupling networks to form a 2-dimensional fault-tolerant hypercube.

decoupling networks are reconfigured, as shown in Figure 9 so that the hypercube structure is maintained. This scheme is generalized for global sparing, where $m = n$ and the entire network is a fault-tolerant module. In the local sparing scheme, where the n-cube consists of FTMs

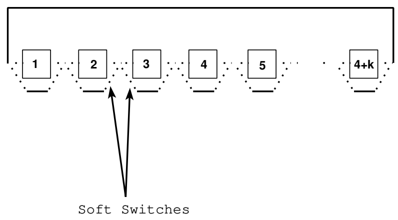

Figure 8: A fault-tolerant module with 4 primary nodes and k spare nodes.

having 2^m primary nodes and k spare nodes, the reliability of a module is given by:

$$RM_{m,k} = RM_{m,k-1} + \binom{2^m + k - 1}{k} r^{2^m}(1-r)^k c^k \tag{2}$$

where r is the reliability of a node. The reliability of an n-dimensional hypercube with 2^m active processors and k spares in each module is given by:

$$RS_{n,m,k} = (RM_{m,k})^{2^{n-m}} \tag{3}$$

The global sparing scheme uses fewer spares than the Rennels basic scheme to achieve the same level of reliability. Chau showed that when $n \leq 8$, global sparing is preferable to modular sparing. However, the assumption that any nonfaulty processor in a module can recognize the faults is not realistic. This scheme takes longer to reconfigure in the presence of faults because, on average, the states of half the nodes will have to be shifted to neighboring nodes. Furthermore, this scheme does not account for link failures. Only the system reliability under node failures has been analyzed.

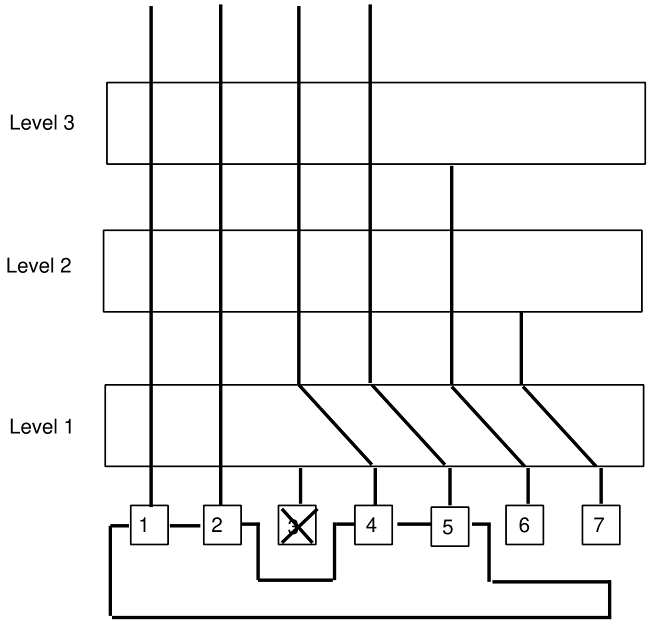

Figure 9: Reconfiguring an FTM with 4 primary nodes and 3 spare nodes when a fault has occurred.

Sultan and Melhem [86] introduced two methodologies for reconfiguration in fault-tolerant modules. The first scheme uses hardware switches to reconfigure in the presence of faults, and provides full spare utilization. As an example, consider a Fault Tolerant Basic Block (FTBB) with M primary nodes $P_1, P_2, ...P_M$ and K spare nodes $S_1, S_2, ..., S_K$. Multiplexers and demultiplexers are used to implement the switching logic for reconfiguration. A 1-to-(K+1) demultiplexer is used for each P_j to divert, when needed, the links of P_j to any S_i. Also, an M-to-1 multiplexer is used for each S_i to connect it to the M neighbors of the failed node. If a primary node P_j is non-faulty, then the demultiplexer is set to select the 0th output. In case P_j is faulty, the replacement of P_j by S_i requires that the demultiplexer associated with P_j be set to its ith output and the multiplexer associated with S_i be set to select its jth input. The scheme can also support spare failures and does not have the overhead of switching processor states, which is present in Chau's scheme. This leads to significant reduction in reconfiguration time and overhead. Also, Chau's scheme requires $2^{n-m}\, nm$ more switches as compared to this scheme to achieve the same level of reliability while using the same number of spares per module. Assuming the reliability of a node to be r, the reliability of this scheme is calculated as follows:

$$R_{FTBB} = \sum_{i=0}^{k} \binom{m+k}{i} r^{m+k-i}(1-r)^i \quad (4)$$

$$R_{sys} = R_{FTBB}^{2^{n-m}} \quad (5)$$

In the second technique, reconfiguration is done using a two-phase routing algorithm. In the first phase, the message is routed to the destination FTBB, i.e. the FTBB which contains the destination node. In the second phase, the message is routed to the destination node within that FTBB. It is assumed that the failure of a node is the failure of the processor, the router and the links of the node. It is also assumed that the spare which replaces a failed node inherits its address. The routing algorithm is distributed and requires only the neighbours of the faulty nodes to know about the faults. These two techniques can be used to tolerate node failures only. The failure of any active node can be replaced by only two spare neighbouring nodes in the same module, compared to more spare nodes using Rennels's scheme. This achieves better system reliability on the expense of the cost of the many spare nodes added to the system.

Another scheme using fault-tolerant modules (FTMs) is proposed by Yang et al [98]. In this scheme, spare nodes in a fault-tolerant module can be used as local spares to replace the faulty nodes in the fault-tolerant module or as remote spares to replace the faulty nodes in other fault-tolerant modules via spare sharing links. Each module contains 2^m active nodes and p local spare nodes. An Internal Switching Connector (ISC) connects 2^m out of a total of $2^m + p$ nodes in each FTM to form an m-cube topology. The FTMs are interconnected using $(n-m)$ External Switching Connectors (ESCs) per module. To provide global sparing, the 2^{n-m} FTMs are also connected as a ring to allow the idle local spares in one module to be used by another. An FTM with k $(p < k \leq 2p)$ faulty nodes can use it's own local spare nodes as well as the remote spares supplied from other FTM's to replace the faulty nodes. Thus, the fault-tolerance capacity of each FTM is improved, and the overall system reliability is enhanced. Node as well as link failures are tolerated in this scheme. Hai and Abd-El-Barr [1, 3] have suggested another fault-tolerant hypercube architecture. In this architecture, the basic block can be a $3-$cube, in which a spare node has been embedded. This spare node is connected to all the eight primary nodes through spare links. Yang and Wu proposed a fault-tolerant Boolean nCube architecture [99]. To keep the system topology unchanged, spares are used, including nodes, links, and switches for reconfiguring a faulty hypercube. A basic fault-tolerant module is first constructed, then the proposed network is designed by connecting the basic modules through several extra switches and links. The new design required 2^p spares to tolerate 2^p faulty nodes. To reduce the cost, multistage interconnection networks are used instead of crossbar switches for reconfiguration. The system reliability is also compared with other hardware approaches. Dutt and Hayes [41] presented an approach for designing fault-tolerant hypercubes using graphs. The approach they presented is based on graph automorphisms and is applicable to any graph structure and any degree of fault tolerance. The scheme presented is amenable to low-cost switch implementations. Lee and Hayes [63, 64] simplified the reconfiguration scheme by modifying the hypercube with the addition of spares and links to improve its fault tolerance, while maintaining a specified level of performance. They used the scalability property of hypercube architecture to tolerate faults gracefully by confining a program to a fault-free subcube. Only processor failure is considered. The design increased the node degree as the hypercube size increased, but provided a simpler reconfiguration process.

There are many other proposals for providing fault tolerance in hypercube based architecture, without the use of spares. A design for fault-tolerant hypercube multiprocessors is presented by Latifi [61]. The degree of connectivity is increased by augmenting the hypercube topology with some extra links. The new network is constructed from the original hypercube by adding spare links between each node and its farthest node. Subcube reliability is evaluated using a node-failure model based on the Markov chain. The extended network contains more subcubes than the original network. Therefore, task allocation and processor utilization becomes more efficient with the proposed hypercube network. Enhanced hypercubes are introduced by Tzeng and Wei [90]. The extra connections are utilized to improve the inter-node distance, diameter and traffic density. The diagnosibility of the enhanced hypercubes is studied by Wang in [92]. Folded hypercubes and their performance and properties are presented in [43].

The major disadvantage of all of these approaches is the extra hardware added to the hypercube, and currently none is available in the marketplace. This extra hardware is introduces by adding redundant nodes and/or redundant links.

2.2 Software Approach

The inherent redundancy in hypercubes makes them more robust than other architectures. [78]; in fact, at least n faults are needed to disconnect an n-dimensional hypercube. Thus, the symmetry and robustness of hypercubes can be exploited to tolerate faults, without adding extra nodes and links. The price of this robustness is that by assigning the task of the faulty node to some other node, there must be some degradation in computation performance and/or communication performance.

Peercy and Banerjee have proposed a distributed table-filling method for routing in faulty hypercubes [70]. These tables are used to find the shortest path from source to destination. The tables are modified dynamically and each node searches its n neighbors to find the shortest path. Table-routing methods for one-to-all broadcasting are also presented, and the modifications necessary to make the algorithms deadlock-free are explained. The table-filling algorithm presented for routing executes in $O(n^3 logn)$ and the one for broadcasting runs in $O(n^2 logn)$ worst case time. Even though no extra hardware is added to the hypercube structure, the execution time and the storage needed for storing the tables do not make these algorithms practical. The design of a software reconfiguration strategy for hypercube multicomputer architectures under multiple faults is presented in [71]. The proposed approach is based on the idea of using multiple virtual processors on a single physical processor and redistributing the work of the faulty processors to other active processors, thus supporting graceful degradation. Centralized algorithms running on the host computer are assumed for reconfiguring a hypercube in the presence of multiple failures. After the algorithm is completed and the paths are determined, the fault status information is relayed to the host and passed to the fault-free nodes.

An approach for achieving fault tolerance using the inherent redundancy of hypercubes without requiring additional spare nodes or links is presented in [74, 75]. The concept of free dimension is introduced and used; a dimension is said to be free if no pair of nodes across that dimension link is faulty. Free dimensions are used to partition the hypercube into subcubes such that each subcube contains, at most, one faulty node. Two algorithms to find free dimensions are presented. The first one is executed in a distributed manner to handle any $f < n$ faults in $O(n)$ steps. The second algorithm, executed in some central processor, takes $O(f^2+n)$ steps, but can be applied to any number of faults. Using the free dimension concept, a scheme is proposed for tolerating any $(f < n)$ faults. The hypercube is partitioned into two subcubes along some free dimension. Two copies of the task graph are embedded; one copy in each subcube. For the two nodes that are assigned to the same task, at least one node is fault-free by the concept of free dimension. Even though this approach does not use additional spare nodes or links, it loses

50% of the hypercube performance by having two copies of the task graph in two subcubes. Moreover, global information about faulty nodes is required, and the reconfiguration algorithms presented took at least $O(n)$ steps. Other software approaches to achieve fault-tolerance when executing application tasks in hypercubes are presented in [59, 89].

An algorithm for tolerating faulty nodes in hypercubes is presented in [4, 9]. The algorithm is based on using general spanning trees for reconfiguring the hypercube to avoid faulty nodes. For reconfiguration and recovery, a number of assumptions are made [6, 8, 11, 12, 101, 102] (these are typical in relation in fault tolerance of hypercubes):

- No failure can occur during the reconfiguration process.
- The detection of faults can be done by any appropriate software or hardware techniques. When a fault exists in a processor (i.e., it cannot perform the correct computation), all the neighbors of the faulty processor will recognize the situation.
- When a node fails, it will be disconnected without affecting the operations of other nodes.
- The root is fault-free or supported by a spare.
- Total node failure (i.e. the processor can handle neither communication, nor computation).
- A copy of the task assigned is kept in the parent node, so that if the processor fails, its parent can take over its task.

The algorithm consists of two phases: the first phase involves the construction of the spanning tree, the second is for reconfiguring the hypercube should a faulty node be detected. The reconfiguration process introduced consists of two basic steps. First, the faulty node is disconnected from the spanning tree. Then, a new spanning tree is constructed by reconnecting the children of the faulty node to the spanning tree. The same algorithm can be generalized to work for the reconfiguration of multicomputer networks in general in the presence of faults. Two more algorithms are presented in [4, 5, 7]; one uses completely unbalanced spanning trees (CUST) and the other uses balanced spanning trees (BST). Both algorithms require, at most, one used link and one unused link for every reconstructed path in the reconfigured hypercube. To illustrate this, assume that node 6 is detected to be faulty (i.e. f=00110) as shown in Figure

10. Using the normal hypercube definitions, the children and the parent of the faulty node f are found to be: $Children(f)$ = {01110,10110} and $parent(f)$ = 00010. The reconfiguration process defines a new parent for the children of the faulty node. For the child c = 01110, the method used define a new parent to be: $newparent$(01110) = 01010, and for the child c = 10110, it is: $newparent$(10110) = 10010. So now node 00010 (i.e. $parent$(00110)) can take over the task of the faulty node 00110, since it has a copy of it, and it can communicate with the children of the faulty node using the reconstructed paths. The reconstructed path to node 01110 is 00010 $\rightarrow$ 01010 $\rightarrow$ 01110 and to node 10110 is 00010 $\rightarrow$ 10010 $\rightarrow$ 10110. The

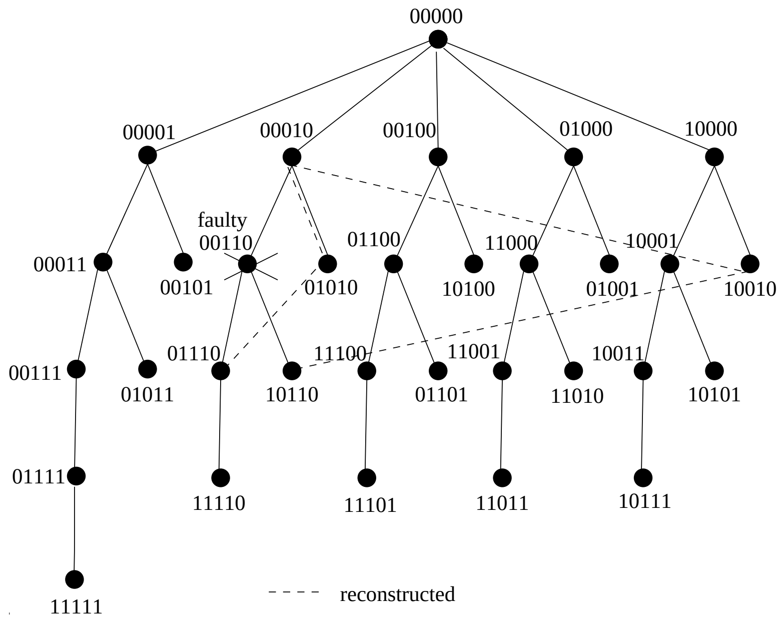

Figure 10: Reconfigured BST under a faulty node

algorithms are optimal, in terms of the reconfiguration time and may increase the congestion of a link by at most one, with no extra-dilation. Single-fault coverage of 100% and almost 100% fault coverage of double and triple faults are achieved by the proposed algorithms for hypercubes having a dimension of $n \geq 10$. Simulation results for the algorithms under more than three faults are also presented. Fault coverage and congestion results for up to 60 faults having different cube sizes are discussed. The major disadvantage of this approach is that the root node should be duplicated by a spare and exchanging copies between nodes may take a long time.

The major advantage of the software approach is that no hardware modification is needed for the hypercube, therefore enabling its use on commercially available hypercubes. On the other hand, the disadvantage is a performance degradation. The drawback of the approach that finds a maximal fault-free subcube and uses it to run the parallel application is the great reduction in systems performance due to a few faults. The cost effectiveness of the two different strategies for fault-tolerant hypercubes is analyzed and compared in [49]. The fault model used for comparing a degradable hypercube to a non-degradable hypercube assumes partial node failure only. In a non-degradable hypercube, spare processors are added to the system. In this case, the faulty processor is replaced by one of the spares using a switching mechanism. In the second strategy, no spares are added to the system, and if a processor is detected to be faulty, the execution continues with one less processor. The minor cost of the degradable hypercube is found to outweigh its performance degradation. The numerical results obtained suggested that the degradable hypercube is more cost effective and, hence, it better exploits the available hardware, relative to the non-degradable hypercube. This suggests that the hypercube architecture is sufficiently robust, and adding redundant hardware would not be the most effective way of tolerating faults in hypercubes.

3 Evaluation of Reliability

Evaluations of performance and reliability are important factors when studying the effectiveness of multicomputers. Hypercubes provide cost-effective ways for improving a computer system's performance; therefore, improvement of reliability becomes critical. Improving performance and increasing reliability are two important issues for all multicomputer networks. Reliability of a distributed processing system can be described by the reliability of links and processing elements, and the redundancy of programs and data files. In designing computer networks with a large number of components, one is interested in maximizing *inter alis* the reliability. Evaluation of the reliability of a hypercube is essential for its usability in critical applications. Several models for evaluating the reliability of hypercubes have been presented in the literature. Here we discuss first the reliability analysis of networks and distributed systems in general. Then we describe some of the research reported about reliability of hypercubes in particular.

3.1 Reliability of Multicomputers

It is very important to evaluate the reliability of any multicomputer system in order to achieve a reliable high performance system. In this section, some of the work done in the evaluation of reliability in general networks is included for interested readers. An algorithm for evaluating global reliability of a network is presented in [51]. Global reliability is the probability of existence of a minimal set of links required for a network to remain connected. The global reliability is found using disjoint spanning trees of the network graph. Reliability analysis in distributed systems is discussed in [73]. Two reliability measures are introduced: distributed program reliability and distributed system reliability. Algorithms using a graph approach for evaluating reliability in distributed systems are presented. Small, not massive, redundancy in resources is found to be desirable for improving reliability. These two measures are also used to express the reliability of distributed systems in [28]. Some other related problems, including the reliability of more than one copy of programs running, the reliability of a program running from different sites, and the reliability of more than one program running on a system, are discussed. A graph approach is used to discuss least-reliable networks in [19]. Edge failures are assumed to be independent and nodes are assumed not to fail.

3.2 Reliability of Hypercubes

The first work in evaluating reliability in hypercubes is done by Abraham and Padmanabhan in [2]. They proposed a subcube reliability model for finding a working $(n-1)$-cube in an n-cube. The ability of finding a d-cube in a hypercube with faulty components is investigated. Expressions for the reliability and mean time to failure of a d-cube system are found for several conditions. These conditions include node failures only, link failures only, and both node and link failures. The hypercube structure is destroyed by any component failure; however, the hypercube is nevertheless found to be capable of supporting tasks requiring smaller subcubes. Constructing parallel paths between subcubes is considered in [29]. Alternative paths can be used to enhance reliable broadcasting in hypercubes. Reliability of redundant-path interconnection networks is studied in [91].

Two graph theoretical results for the reliability analysis of the hypercube network are presented in [100]. Here it is assumed that all nodes are perfectly reliable, and that all links may fail independently with the same probability. The number of spanning trees of the hypercube is found and used to compute the reliability function. The two extreme cases for the link failure rate, close to zero and close to one, are considered. The number of links needed to disconnect the network is also calculated. A new model for reliability evaluation of hypercube multicomputers is presented in [57]. The model is based on the decomposition principle, where the hypercube is decomposed into smaller hypercubes until the reliability of the smallest cube is found. Here it is assumed that the failure rate of the links is negligible and that the nodes have an exponential distribution failure rate. Moreover, a host processor is assumed to perform detection and maintenance actions.

A spanning tree approach for evaluating hypercube reliability is presented in [54]. The paper evaluates the reliability of hypercubes as a function of link reliability, based on the the recursive decomposition of the network. An approach for improving the reliability of communication in hypercubes by adding redundant links to the structure is presented in [62]. This approach is not practical because of the cost of the extra hardware introduced to the system. Subcube reliability is evaluated using a node-failure model, based on the Markov chain, in [61]. However, the hypercube topology is modified by augmenting extra links. In [37, 38, 55], a task-based dependability model is used to evaluate subcube dependability and hypercube availability. This model uses the probability of having j connected nodes out of x working nodes in a hypercube.

The problem of determining the exact reliability of hypercubes is know to be computationally hard. Bulka and Dugan [20] present a lower bound on the NR of an n-cube recursively as computed from a lower bound on the reliability of two component (n-1)-cubes. Let NR(n,p) denote the NR of an n-cube (C_n) with link reliability p. They use a C_2 as the basic cube for which the exact reliability is given as follows:

$$NR(2,p) = 4p^3(1-p) + p^4 \tag{6}$$

The reliability of a C_n is calculated from the reliability of a C_{n-1}. The C_n is decomposed into two congruent C_{n-1}s such that each node in one C_{n-1} is connected by an *exterior* link to its congruent node in the other C_{n-1}. There are 2^{n-1} exterior links interconnecting the nodes in the C_{n-1}s. The links within each C_{n-1} are termed *interior* links. The BD approach computes the reliability of a C_n by computing the probability of three disjoint cases of working and failed $(n-1)$-cubes and exterior links.

Case 1: Both $(n-1)$-cubes are operating and i exterior links operate, $1 \leq i \leq 2^{n-1} - 2$.

$$T1 = NR(n-1,p)^2 \sum_{i=1}^{2^{n-1}-2} \binom{2^{n-1}}{i} p^i \; q^{2^{n-1}-i}$$

Case 2: Atleast one (n-1)-cube is operating, and one exterior link has failed. The node at the endpoint of the failed link in the non-operating C_{n-1} is linked to atleast one of its neighbours by an interior link.

$$T2 = [2.NR(n-1,p)(1-q^{n-1}) - NR(n-1,p^2)] \times 2^{n-1}p^{2^{n-1}-1}q$$

Case 3: All 2^{n-1} exterior links operate . Let $p' = p^2 + 2pq$.

$$T3 = NR(n-1,p')p^{2^{n-1}}$$

The lower bound on the NR of the n-cube is then:

$$NR(n,p) = T1 + T2 + T3 \tag{7}$$

This analysis can be combined with complementary analyses to get more accurate models of the system without the need for very complicated models. The same approach can also be used to analyze the reliability of any other network, assuming that only links are subject to failure.

Chang and Bhuyan [26] study the problem of subcube allocation in faulty hypercubes. A divide-and-conquer technique is used to form the set of disjoint subcubes in the faulty hypercube. The subcube partitioning method proposed can tolerate up to $\frac{n}{2}$ faults while maintaining a fault-free $(n-1)-$cube in a faulty n-cube. It is also shown that a fault-free $(n-m-1)$-cube can be found in case of $2^m.(\frac{n-m}{2}+1) + 2^{m-1} - 1$ or fewer faults. A probability fault model is developed in [25] to derive an exact expression for the $(n-1)-$cube reliability in an n-cube. Subcube reliability is defined as the probability that a subcube of a specified size is available in the system. Given that the reliability of a single node is p in an n-cube, the reliability of an $(n-1)-$cube is found to be:

$$R_{n,n-1} = \sum_{i=1}^{n}(-1)^{i-1}S^n_{n-i}p^{N-2^{n-i}} + (-1)^n p^N \tag{8}$$

where S^n_{n-i} is the number of $(n-i)-$cubes in an $n-$cube, which is equal to :

$$2^i.(\frac{n.(n-1).(n-2).....(i+1)}{(n-i)!}). \tag{9}$$

Even though, some approximate analysis to find the $m-$cube reliability for $1 \leq m \leq n-2$ is given, it remains an open problem to find the exact reliability expressions for subcubes of size smaller than $n-1$ in an n-cube. Most of the results reported about the evaluation of hypercube reliability assume restricted models, without considering all reliability parameters, such as failure rate, recovery rate, repair rate, and coverage factor. In [4, 10], an approach based on fast reconfiguration algorithms using spanning trees, for improving the reliability of hypercubes, is presented. It is shown that, with a fast reconfiguration algorithm, the system recovers more quickly, improving hypercube reliability. A dependability evaluation study, based on a Markov chain model for a cluster of four hypercube nodes, is presented. This paper does not assume link failure only as done in [54, 100]. Instead of needing a separate host processor to perform the reconfiguration, as proposed in [57], a reconfiguration algorithm is used to improve hypercube reliability. The inherent redundancy of the hypercube structure is used to improve reliability compared to introducing extra redundant links to the structure of the standard hypercube, which is done in [61, 62]. The reliability approach presented in [4] can also be used for evaluating the dependability of multiprocessor systems in general.

4 Fault Tolerant Routing

The performance of multicomputers is highly dependent on the communication method and the message routing scheme used. Processors (or nodes) in a hypercube communicate with each other by passing messages. The large number of processors that are provided by current hypercubes increase the probability of failure in the system. Therefore, fault-tolerant routing is important for achieving highly dependable computing.

Two basic routing strategies exist for routing messages in multicomputers, static routing and dynamic routing. In static routing, messages are routed along a static (i.e. fixed) path defined before the computation starts. The path taken by a message is determined by the source-destination pair without considering the network state. This kind of routing is also called oblivious or deterministic routing [102]. It does not use the bandwidth effectively, and messages may be blocked even if alternative paths exist. This is similar to centralized routing in [67]. In dynamic routing, the routing of messages makes use of the alternative paths between communicating processors. For a given source and destination, the path taken depends on the network state taking care of the faulty and congested channels. This routing makes efficient use of the bandwidth, providing more fault tolerance. This kind of routing is also called adaptive routing [102] and is similar to distributed routing [67].

In multicast routing [84, 85], messages are sent from the source node to destination nodes in a tree-like pattern. To reduce the length of the multicast path, path-like routing is presented in [67], where the destination nodes are divided into several subsets and the source message is sent using separate multicast paths. Unlike tree routing, path-like routing does not need to replicate messages at each intermediate node. In store-and-forward routing, a message is sent as a whole packet, and deadlock can be prevented by using properly structured buffers [68]. In wormhole routing, messages consist of a sequence of flits (e.g. words). If the header flit is blocked in one channel during the routing process, the node sending the header does not buffer the whole message and the blocked message remains in the network [67, 68]. As a result, the progress of messages is stopped in the tree, causing more congestion. Because the tree-like routing model is not suitable for multicast routing in wormhole networks, a multicast star wormhole routing model is presented in [67]. Two deadlock-free methods, which reduce the communication traffic using that model, are presented. As the flits of the message are pipelined in the channels in wormhole routing, the network latency is reduced and almost independent of the length of the path [68]. In store-and-forward routing, the number of buffers grows with network size. However, the number of queues grows with the number of channels in the network [68]. Pipelined-circuit-switching is similar to wormhole routing, except that the data flits do not follow the header immediately, but wait until an acknowledgment is received by the source node [47]. Virtual-cut-through routing is like wormhole routing except that the blocked messages are buffered, removing them from the network. Forwarding of flits in virtual-cut-through routing depends on the intermediate node and the routing information [47]. The nCUBE 2 system [69] adopts wormhole routing (tree-like) and supports multiple node addresses in the address field of the message.

Dally and Seitz [36] proposed a deadlock-free, message routing algorithm that uses the concept of virtual channels: groups of channels that share a physical channel with a separate queue for each virtual channel. Deadlock is avoided by removing the cycles of the channel dependency graph by splitting physical channels into groups of virtual channels. The concept of virtual channels is extended to multiple virtual communication systems in [68] to provide fault tolerance and adaptability. Chen and Shin [31, 32] propose a distributed adaptive fault-tolerant routing scheme for faulty hypercubes in which each node needs to know the conditions of its own links. The proposed scheme cannot tolerate more than n faults using depth-first search and network delay tables. A deadlock-free routing scheme in faulty hypercubes with wormhole routing capability is proposed by Kim and Shin in [58]. The scheme is based on the re-establishment of a routing path to the destination and dose not guarantee the shortest path. It uses either wormhole routing or staged routing, depending on the availability of one or more healthy $(n-2)$-cubes within the faulty hypercube. In case the system cannot find a healthy $(n-2)$-cube, then if the message encounters a faulty node, it is sent to the node in the alternative path found with a depth-first search. Routing decisions are made in a decentralized manner. Chiu and Wu [33] adopt the concept of an unsafe node to find those nodes that might cause routing problems. Each node is required to contain its local state information, and a routing algorithm is proposed making use of the state information. The algorithm guarantees routing the message when either the source or the destination of the message is a safe node. The issue of freedom from deadlock is addressed, but the algorithm cannot handle more than $n-1$ faults.

Kim and Das analyze hypercube communication delay with a deadlock-free wormhole routing scheme [56]. The model they present can handle any type of message destination distribution. The traffic density and the message delay are analyzed. They assume that any number of messages can be received by any node. The results presented show some agreement with simulation results. The routing problem in faulty supercubes is discussed in [81]. An algorithm, which does the routing when the number of faulty nodes is less than the node connectivity, is presented. The properties of incomplete hypercubes [53] in the presence of node failures are also studied. It is assumed that in the faulty condition, each fault-free node knows the identity of all the faulty nodes. Other fault tolerant routing algorithms for hypercubes can also be found in [48].

5 Fault Tolerant Embedding

The importance of the problem of tree embedding is motivated by the fact that many parallel algorithms, including parallel prefix, broadcasting, and divide-and-conquer algorithms have a natural tree structure. The problem of embedding and reconfiguration of binary trees in hypercubes has also been studied by many researchers [30, 42, 46, 76, 93, 94, 101]. Because both the hypercube and the complete binary tree can be defined recursively, embedding complete binary trees in faulty hypercubes in a recursive manner is described. The ability of the hypercube to implement tree-structured algorithms in the presence of faults is examined in [94]. The paper proves an upper bound on the number of faults that can be avoided when a natural class of embedding techniques is used.

The ability to embed large binary trees in smaller hypercubes by increasing the congestion of the nodes is discussed in [42]. The ability to tolerate node failures with minimum slowdown in executing binary tree algorithms is also addressed. The fault-tolerance schemes presented are based on finding a replacement processor at a different level of the tree. Two types of embedding are discussed: recursive embedding, where the congestion is increased by extending the basis in the construction with appropriate edges, and inorder embedding, where larger trees are embedded in smaller cubes by using the concept of folding. Embedding and reconfiguration of binary trees in faulty hypercubes is also discussed in [101]. It is assumed that the location of faulty nodes is known and no more than n faults can be tolerated. The concept is based on the concept of free-dimension, explained earlier. Using this approach, two distributed schemes for embedding and reconfiguring binary trees in faulty hypercubes are provided. The first scheme is recursive embedding, in which the maximum size binary tree is found. This scheme can obtain an n-level dilation 2 binary tree with $2^n - f - 1$ nodes. The second scheme cannot tolerate more than $n-1$ faults with expansion 2 dilation 2 initial embedding and constant task migration time. The reconfiguration scheme is executed in a distributed manner. If a node becomes faulty, one of its neighbours is designated to initialize the reconfiguration process. The problem of finding $(n-1)$-binary trees in faulty hypercubes is studied in [30]. A way to find a fault-avoiding binary tree, in which each node has exactly two or zero sons, is presented. The general problem of determining if a k-tree exists when a number of nodes are removed from the hypercube is found to be NP-complete; however, it is shown in [30] that an $(n-1)$-tree is guaranteed to exist if no more than $n - 1 - logn$ nodes are removed from the hypercube. The paper assumes that all faults are globally known by the fault-free nodes. A distributed scheme for reconfiguring binary trees in faulty hypercubes is presented in [102]. The concept of free dimension, explained earlier, is generalized to the concept of degree of occupancy, which is the number of pairs of faulty nodes that occupy a dimension. A dilation 3 expansion 2 scheme, which can reconfigure round any $3n/2$ faults with $O(n)$ reconfiguration time, is presented. The scheme assumes (1) the partial fault model with global knowledge of faulty nodes and (2) the existence of a fault detection algorithm.

The hypercube is not only suitable for tree embedding, but also for embedding many other networks, such as a linear array, grids, and general meshes. The problem of assigning points from a two-dimensional grid to hypercube nodes with at most one grid point per node, is addressed by Chan and Chin in [23]. A result about minimal mesh embedding in binary hypercubes is presented in [79]. Embedding of grids into optimal hypercubes is also addressed by Chan in [22]. Gupta and Hambrusch consider the problem of embedding multiple networks into a k-dimensional hypercube such that the dilation and congestion are minimized [50].

In general, the problem of embedding trees or graphs into hypercubes is found to be NP-complete. Two heuristics for solving this problem are presented in [95]. In general, the problem of tree and graph embedding is not easy. The problem of minimizing dilation, when the host graph is a line (bandwidth minimization), is NP-complete even when the source graph is a binary tree [46]. It is proved in [83] that there exist polynomial time algorithms that determine whether a graph can be embedded into a complete binary tree with fixed dilation or fixed congestion. The results show that the problem of minimizing dilation is harder than the problem of minimizing congestion.

6 Reliable Broadcasting

Reliable broadcasting and effective utilization of communication resources are very important for good performance in multicomputers. Broadcasting is also useful in many linear algebra algorithms, such as matrix multiplication and Gaussian elimination. The problem of broadcasting information in a hypercube in which links fail independently with fixed probability is considered in [34]. Messages may be directly transmitted to adjacent nodes only, and every node may communicate with, at most, one neighbor in a given unit of time. Nodes are assumed to be fault-free and communication between adjacent nodes is assumed to be bidirectional. The broadcasting algorithm is adaptive and does not require a central monitor to supervise the broadcast scheme. The algorithm requires $O(logn)$ expected time and makes $O(n)$ expected number of transmissions. A broadcasting algorithm that sends multiple copies of the message to all nodes in the hypercube using a disjoint path is presented in [72]. The proposed algorithm does not require information about the identity of the faulty processors. The idea of the algorithm is that the node that wants to broadcast the message sends it to its neighbours first, and then each neighboring node broadcasts the message using a doubling algorithm. The proposed approach can tolerate up to $n-1$ faults in $n+1$ steps, assuming that all outgoing links of a node can be used at the same time. Broadcasting algorithms, based on different types of spanning trees [52] in incomplete hypercubes, are presented in [88]. Packet switching with both one-port and all-port communications are considered. With one-port communication, the best algorithm is found to have an optimal bandwidth utilization within a factor of less than or equal to two. For all-port communication, an optimal algorithm is found for bandwidth utilization. Various algorithms for reliable broadcasting and gossiping in faulty hypercube multicomputers are described and analyzed in [45]. Fault-tolerant broadcasting algorithms, using the same approach used in [72] by sending multiple copies of a message through disjoint paths, are constructed. The same approach is also used to develop a fault-tolerant gossiping algorithm for hypercubes. It is assumed that the communications are based on message-passing procedures in a store-and-forward mode. Links are assumed to be bidirectional, and every node is able to communicate through all its ports simultaneously. Communication under the *whispering mode*, where each processor can only use one port at a given time, is also considered. No information on the identity of the faulty units (nodes or links) is required. The proposed algorithms are asymptotically optimal when the lengths of the messages went up, and optimal when the messages are short.

Wu and Yao have studied the problem of multicasting in injured hypercubes with faulty nodes [96, 97]. Their method is based on limited global information captured by the safety level. The safety level associated with a node is a measure which approximates the number of faulty nodes in the neighbourhood. An algorithm for calculating the safety level and for updating it for each node in the hypercube is used. The proposed algorithms are applicable to injured hypercubes with up to $n-1$ faulty nodes. An address-sum based multicasting method, using the distribution of destination nodes and safety levels of neighbouring nodes, is discussed. Simulation analysis for the traffic generated for the algorithms is conducted. Message routing in an injured hypercube is also studied in [24]. The all-to-all reliable broadcasting problem is addressed in [65]. Here a general method for performing such broadcasting in an interleaved manner is proposed. The broadcasts from nodes are interleaved in such a way that no two packets contend for the same link at any time. The method is found to be useful for systems that use virtual cut-through or wormhole routing. The algorithm assumes that all links adjacent to a node can be used at the same time. Several other virtual cut-through solutions and a store-and-forward solution are compared to the proposed method. The algorithm is found to minimize the execution time and can be used for a large class of interconnection networks.

7 Future Directions and Open Problems

Massively parallel computers using thousands of processors are becoming more feasible as a result of advances in technology. This survey has addressed some important issues related to fault tolerance and reliability of hypercubes. It also reveals a number of future research directions for fault tolerance and reliability of multicomputers in general. The techniques described here can also be applied to other connected networks, such as multibutterflies, meshes, fat-trees, and de-Bruijn networks. Thus, it is expected that ideas presented in this paper may contribute to the development of new reconfiguration algorithms for other multicomputer systems.

The algorithms presented in [6] for reconfiguration of spanning trees in faulty hypercubes may also be investigated for decreasing the traffic generated by using multiple paths and increasing the time needed for delivering messages. They can also be used to introduce effective fault-tolerant broadcasting algorithms. These algorithms will use the redundant paths in the hypercube, instead of generating multiple copies of the same message and sending it through alternative paths to enhance fault tolerance. This might lead to algorithms that minimize the execution time of one-to-all and all-to-all reliable broadcasting.

Traditionally, reliability analysis for complex systems has been a challenging problem because of the unknown size and nature of the state space. There have been several attempts of approximating the reliability of hypercubes or improving its lower bounds; however, there are no known polynomial time algorithms for exact computation of terminal reliability or network reliability for the hypercubes. Even though some approximate analysis to find the $m-$cube reliability for $1 \leq m \leq n-2$ has been given, it remains an open problem to find the exact reliability expressions for subcubes of size smaller than $n-1$ in an n-cube. Evaluation of terminal and network reliability in other multicomputer networks is an open area of research that needs more investigation.

In general, the problem of embedding trees or graphs into hypercubes is found to be NP-complete; however, some heuristics for solving the problem can be considered, like those presented in [95]. A lot of research has been done on fault tolerance of binary tree embedding; however, more research will be needed for adding fault tolerance to the other tree embeddings in hypercubes.

8 Conclusions

Massively parallel computers, using thousands of processors, will be the future trend for producing tremendous computational power. As the probability of any one or more processors failing in such a complex system is large, building some fault-tolerance feature into them becomes extremely important. Fault tolerance in highly parallel computers is important for achieving high-performance reliable computing. The problem of tolerating faulty processors or links in hypercubes has been studied by many researchers, either by using spares or by reconfiguration. This paper is mainly a chronological survey of fault tolerance and related issues of hypercube-based multicomputers. Several important issues related to fault tolerance of such multicomputers have been addressed, unfolding a number of future research directions for fault tolerance and reliability of multicomputers in general.

Acknowledgments We would like to thank King Fahd University of Petroleum an Minerals and the British Council for support to carry out this work. The first author would like to acknowledge the hospitality and facilities offered at the Department of Computer Science, University of Edinburgh, United Kingdom, where part of this work was done.

References

[1] M. Abd-El-Barr, M. Abdul Hai, and M. Benten, "Subcube Reliability of a Modular Fault-Tolerant Hypercube Architecture," *Proc. ISCA International Conference on Parallel and Distributed Computing Systems*, Orlando, Florida, USA, pp. 268-274, Sep. 1995.

[2] S. Abraham and K. Padmanabhan, "Reliability of the Hypercube," *1988 Int'l Conf. on Par. Proc.,* vol I, pp. 90-94, Aug. 1988.

[3] A. M. Abdullah, Reliability of Modular Fault-Tolerant Hypercube Networks, M.S. Thesis, Dept. of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran,31261. Saudi Arabia, Jan. 1995.

[4] K.M. Al-Tawil, Reconfiguration Algorithms for Fault Tolerance in Hypercubes, Ph.D. dissertation, Department of Computer Science, Texas A&M University, College Station, TX, Dec. 1994.

[5] K.M. Al-Tawil and D.R. Avresky, "An Effective Approach to Achieving Fault Tolerance in Hypercubes: Algorithms and Simulation," *1994 IEEE Workshop on Fault-Tolerant Par. and Dist. Sys.,* June 1994.

[6] K.M. Al-Tawil and D.R. Avresky, "Reconfiguration of Spanning Trees in Faulty Hypercubes," *Proc. 23rd Int'l Conf. on Par. Proc.,* vol. II, pp. 319-323, Aug. 1994.

[7] K.M. Al-Tawil, D.R. Avresky, and D.K. Pradhan, "Fault Tolerance of Hypercubes Using Spanning Trees," Technical Report 93-055, Department of Computer Science, Texas A&M University, College Station, TX, Dec. 1993.

[8] D.R. Avresky and K.M. Al-Tawil, "Reconfiguration of Faulty Hypercubes," Technical Report 94-008, Department of Computer Science, Texas A&M University, College Station, TX, Jan. 1994.

[9] D.R. Avresky and K.M. Al-Tawil, "Reconfiguration of Faulty Hypercubes," *First European Dependable Computing Conference,* Berlin, Germany, Oct. 4-6, 1994.

[10] D.R. Avresky and K.M. Al-Tawil, "Improvement of Reliability in Hypercubes Using a Fast Reconfiguration Algorithm," *Proc. 7th ISCA Int'l Conf. Parallel and Distributed Computing,* Oct. 6-8, 1994.

[11] D.R. Avresky and J. Arlat, "Functional Programming for Fault-Tolerance in Parallel Computing Systems," Research Report no. 93085, LAAS-CNRS, 7 Avenue du Colonel Roche, 31077 Toulouse Cedex, France, Mar. 1993.

[12] D.R. Avresky and J. Arlat, "Functional Programming for Fault-Tolerance in Parallel Computing Systems," *Proc. ISCA Int'l Conf. Parallel and Distributed Computing,* pp. 38-44, Oct. 14-16, 1993.

[13] N. Bagherzadah and M. Dowd, "Computaion in Faulty Stars," *IEEE Trans. on Reliability* vol. 44, no. 1, pp. 114-119, Mar. 1995.

[14] P. Banerjee, "Strategies for Reconfiguring Hypercubes Under Faults," *Proc. 20th Int'l Symp. on Fault Tolerant Computing,* pp. 210-217, June 1990.

[15] P. Banerjee and M. Peercy, "Design and Evaluation of Hardware Strategies for Reconfiguring Hypercubes and Meshes Under Faults," *IEEE Trans. on Comp.,* vol. C-43, no. 7, pp. 841-848, July 1994.

[16] P. Banerjee, J. Rahmeh, C. Stunkel, V. Nair, K. Roy, and J. Abraham, "An Evaluation of System-Level Fault Tolerance on the Intel Hypercube Multiprocessor," *Proc. 18th Int'l Symp. on Fault Tolerant Computing,* pp. 362-367, June 1988.

[17] D. Bertsekas, C. Ozveren, G. Stamoulis, P. Tseng, and J. Tsitsiklis, "Optimal Communication Algorithms for Hypercubes," *J. of Par. and Dist. Computing,* vol. 11, pp. 263-275, 1991.

[18] Bruck *et al.*, "Tolerating Faults Using Subcube Partitioning," *IEEE Trans. on Comp.,* vol. C-41, no. 5, pp.599-605, May 1992.

[19] F. Boesch, A. Satyanarayana, and C. Suffel, "Least Reliable Networks and the Reliability Domination," *IEEE Trans. on Communications,* vol. C-38, no. 11, pp. 2004-2009, Nov. 1990.

[20] D. Bulka and J. Dugan, "A Lower Bound on the Reliability of an n-dimensional Hypercube," *Proc. 9th Symp. Reliable Distributed Systems,* pp. 44-53, 1990.

[21] B. Chan, F. Chin, and C. Poon, "Optimal Simulation of Full Binary Trees on Faulty Hypercubes," *IEEE Trans. on Par. and Dist. Sys.,* vol. 6, no. 3, pp. 269-283, March 1995.

[22] M.Y. Chan, "Embedding of Grids into Optimal Hypercubes," *SIAM J. Comput.,* vol. 20, no. 5, pp. 834-864, Oct. 1991.

[23] M. Chan and Y. Chin, "On Embedding Rectangular Grids in Hypercubes," *IEEE Trans. on Comp.,* vol. C-37, no. 10, pp. 1285-1288, Oct. 1988.

[24] M. Chan and K. Chin, "Message Routing in an Injured Hypercube," *Proc. 4th Hypercube Conference,* pp. 312-317, 1988.

[25] Y. Chang and L. Bhuyan, "A Combinatorial Analysis of Subcube Reliability in Hypercubes," *IEEE Trans. on Comp.,* vol. C-44, no. 7, pp.952-956, July 1995.

[26] Y. Chang and L. Bhuyan, "Subcube Fault Tolerance in Hypercube Multiprocessors," *IEEE Trans. on Comp.,* vol. C-44, no. 9, pp.1108-1120, Sep. 1995.

[27] S. Chau and A. Liestman, "A Proposal for a Fault Tolerant Binary Hypercube Architecture," *Proc. 19th Int'l Symp. on Fault Tolerant Computing,* pp. 323-330, June 1989.

[28] D.J. Chen and T.H. Huang, "Reliability Analysis of Distributed Systems Based on a Fast Reliability Algorithm," *IEEE Trans. on Par. and Dist. Sys.,* vol. 3, no. 2, pp. 139-154, Mar. 1992.

[29] G.I. Chen and T.H. Lai, "Constructing Parallel Paths Between Two Subcubes," *IEEE Trans. on Comp.,* vol. C-41, no. 1, pp. 118-123, Jan. 1992.

[30] M.Y. Chen and S. Lee, "Fault-Tolerant Embedding of Complete Binary Trees in Hypercubes," *IEEE Trans. on Par. and Dist. Sys.,* vol. 4, no. 3, Mar. 1993.

[31] M.S. Chen and K.G. Shin, "Adaptive Fault-Tolerant Routing in Hypercube Multicomputers," *IEEE Trans. on Comp.,* vol. C-39, no. 12, pp. 1406-1416, Dec. 1990.

[32] M.S. Chen and K.G. Shin, "Depth First Search Approach for Fault-Tolerant Routing in Hypercube Multicomputers," *IEEE Trans. on Par. and Dist. Sys.,* vol. 2, no. 4, pp. 152-159, April 1990.

[33] G.M. Chiu and S.P. Wu, "Fault-Tolerant Routing Strategy in Hypercube Systems," *Proc. 24th Int'l Symp. on Fault Tolerant Computing,* June 1994.

[34] B. Chlebus, K. Diks, and A. Pelc, "Optimal Broadcasting in Faulty Hypercubes," *Proc. 21st Int'l Symp. on Fault Tolerant Computing,* pp. 266-273, Montreal, Canada, June 1991.

[35] W.J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels," *IEEE Trans. on Par. and Dist. Sys.,* vol. 4, no. 4, pp. 466-474, Apr. 1993.

[36] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. on Comp.,* vol. C-36, no. 5, pp. 547-552, May 1987.

[37] C. Das and J. Kim, "An Analytical Model for Computing Hypercube Availability," *Proc. 19th Int'l Symp. on Fault Tolerant Computing,* pp. 530-537, June 1989.

[38] C. Das and J. Kim, "A Unified Task-Based Dependability Model for Hypercube Computers," *IEEE Trans. on Par. and Dist. Sys.,* vol. 3, no. 3, pp. 312-324, May 1992.

[39] K. Day and A. Tripathi, "A Comparative Study of Topological Properties of Hypercubes and Star Graphs," *IEEE Trans. on Par. and Dist. Sys.,* vol. 5, no. 1, pp. 31-37, Jan. 1994.

[40] F. Daud, Design and Analysis of a Hierarchical Fault-Tolerant Multicomputer Network, M.S. Thesis, Dept. of Computer Science, King Fahd University of Petroleum and Minerals, Dhahran 31261. Saudi Arabia, May 1996.

[41] S. Dutt and J. Hayes, "An Automorphic Approach to the design of Fault-Tolerant Multiprocessors," *Proc. 19th Int'l Symp. on Fault Tolerant Computing,* pp. 496-503, June 1989.

[42] Kemal Efe and Kumar Ramaiyer, "Congestion and Fault Tolerance of Binary Tree Embeddings on Hypercube," *Fifth Int'l Par. Proc. Symp.,* 1991.

[43] A. El-Amawy and S. Latifi, "Properties and Performance of Folded Hypercubes," *IEEE Trans. on Par. and Dist. Sys.,* vol. 2, no. 1, pp. 31-42, Jan. 1991.

[44] A. El-Amawy and R. Raja, "Split Sequence Generation Algorithms for Efficient Identification of Operational Subcubes in Faulty Hypercubes," *Parallel Computing,* vol. 19, no. 7, pp. 789-805, July 1993.

[45] P. Fraigniaud, "Asymptotically Optimal Broadcasting and Gossiping in Faulty Hypercube Multicomputers," *IEEE Trans. on Comp.,* vol. C-41, no. 11, pp. 1410-1419, Nov. 1992.

[46] M. Garey, R. Graham, D. Johnson, and D. Knuth, "Complexity Results for Bandwidth minimization," *SIAM J. Appl. Math.,* vol. 34, pp. 477-495, 1978.

[47] P. Gaughan and S. Yalamanchili, "Adaptive Protocols for Hypercube Interconnection Networks," *IEEE Computer,* May 1993.

[48] J. Gordon and Q. Stout, "Hypercube Message Routing in the Presence of Faults," *Proc. 3rd Conf. Hypercube Concurrent Computers and Applications,* pp. 318-327, Jan. 1988.

[49] V. Grassi, "Cost Effectiveness of Different Fault Tolerance Strategies for Hypercube Systems," *Proc. 21st Int'l Symp. on Fault Tolerant Computing,* pp. 196-203, Montreal, Canada, June 1991.

[50] A. Gupta and S. Hambrusch, "Multiple Network Embeddings into Hypercubes," *J. of Parallel and Distributed Computing,* vol. 19, pp. 73-82, 1993.

[51] S. Jain and K. Gobal, "An Efficient Algorithm for Computing Global Reliability of a Network," *IEEE Trans. on Reliability,* vol. 37, no. 5, pp. 488-492, Dec. 1988.

[52] S. Johnsson and C. Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes," *IEEE Trans. on Comp.,* vol. C-38, no. 9, pp. 197-202, Sept. 1989.

[53] H.P. Katseff, "Incomplete Hypercubes," *IEEE Trans. on Comp.,* vol. C-37, no. 5, pp. 604-608, May 1988.

[54] D. Kaur, H. Singh, and R. Kaushal, "Reliability Evaluation of Hypercubes and Hypernets Using Spanning Tree Approach," *Proc. 32nd Midwest Symp. on Circuits and Systems,* pp. 927-9330, Aug. 1989.

[55] J. Kim and C. Das, "On Subcube Dependability in a Hypercube," *Proc. ACM SIGMETRICS,* pp. 111-119, May 1991.

[56] J. Kim and C. Das, "Hypercube Communication Delay with Wormhole Routing," *IEEE Trans. on Comp.,* vol. C-43, no. 7, pp. 806-814, July 1994.

[57] J. Kim, C. Das, W. Lin, and T. Feng, "Reliability Evaluation of Hypercube Multicomputers," *IEEE Trans. on Reliability,* vol. 38, no. 1, pp. 121-129, April 1989.

[58] J. Kim and K. Shin, "Deadlock-Free Fault-Tolerant Routing in Injured Hypercubes," *IEEE Trans. on Comp.,* vol. 42, no. 9, pp. 1078-1088, Sept. 1993.

[59] K. Kim and A. Kavianpour, "A Distributed Recovery Block Approach to Fault-Tolerant Execution of Application Tasks in Hypercubes," *IEEE Trans. on Par. and Dist. Sys.,* vol. 4, no. 1, pp. 104-111, Jan. 1993.

[60] S. Kwan and W. Ruzzo, "Adaptive Parallel Algorithms for Finding Minimum Spanning Trees," *Proc. 1984 Int'l Conf. on Par. Proc.,* pp. 439-443, Aug. 1984.

[61] S. Latifi, "Fault-Tolerant Hypercube Multiprocessors," *IEEE Trans. on Reliability,* vol. 39, no. 3, pp. 361-368, Aug. 1990.

[62] S. Latifi and A. El-Amawy, "Enhancing Communication Reliability in Hypercube Networks," *Proc. 21st Southeastern Symp. on System Theory,* pp. 234-237, Mar. 1989.

[63] T. Lee and J. Hayes, "Routing and Broadcasting in Faulty Hypercube Computers," *Proc. Third Conf. on Hypercube Computers and Applications,* 1988.

[64] T. Lee and J. Hayes, "Design of Gracefully Degradable Hypercube-Connected Systems," *J. of Parallel and Distributed Computing,* vol. 14, pp. 390-401, 1992.

[65] S. Lee and K. Shin, "Interleaved All-to-All Reliable Broadcast on Meshes and Hypercubes," *IEEE Trans. on Par. and Dist. Sys.,* vol. 5, no. 5, pp. 449-458, May 1994.

[66] F. Thomson Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes*, Morgan Kaufmann Publishers, San Mateo, CA, pp. 389-437, 1992.

[67] X. Lin and L. Ni, "Deadlock-Free Multicast Wormhole Routing in Multicomputer Networks," *18th Annual Int'l Sym. on Computer Architecture,* Toronto, pp. 116-125, May 1991.

[68] D. Linder and J. Harden, "An Adaptive and Fault-Tolerant Wormhole Routing Strategy for k-ary n-cubes," *IEEE Trans. on Comp.,* vol. C-40, no. 1, pp. 2-12, Jan. 1991.

[69] nCUBE 2 Systems: Technical Overview, nCUBE Corporation, Foster City, CA, 1992.

[70] M. Peercy and P. Banerjee, "Distributed Algorithms for Shortest-Path, Deadlock-Free Routing and Broadcasting in Arbitrarily Faulty Hypercubes,"*Proc. 20th Int'l Symp. on Fault Tolerant Computing,* pp. 218-225, June 1990.

[71] M. Peercy and P. Banerjee, "Design and Analysis of Software Reconfiguration Strategies for Hypercube Multicomputers under Multiple Faults," *Proc. 22th Int'l Symp. on Fault Tolerant Computing,* pp. 448-455, June 1992.

[72] P. Ramanathan and K. Shin, "Reliable Broadcast in Hypercube Multicomputers," *IEEE Trans. on Comp.,* vol. C-37, no. 12, pp. 1654-1657, Dec. 1988.

[73] C. Raghavendra, V. Prasanna, and S. Hariri, "Reliability Analysis in Distributed Systems," *IEEE Trans. on Comp.,* vol. C-37, no. 3, pp. 352-358, Mar. 1988.

[74] C. Raghavendra, P. Yang, and S. Tien, "Free Dimensions - An Effective Approach to Achieving Fault Tolerance in Hypercubes," *Proc. 22nd Int'l Symp. on Fault Tolerant Computing,* pp. 170-177, July 1992.

[75] C. Raghavendra, P. Yang, and S. Tien, "Free Dimensions - An Effective Approach to Achieving Fault Tolerance in Hypercubes," *IEEE Trans. on Comp.,* vol. 44, no. 9, pp. 1152-1156, Sept. 1995.

[76] S. Ravindran and A. Gibbons, "Dense Edge-Disjoint Embedding of Complete Binary Trees in the Hypercube," *Information Processing Letters,* vol. 45, pp. 321-325, Apr. 1993.

[77] D. Rennels, "On Implementing Fault Tolerance in Binary Hypercubes," *Proc. 16th Int'l Symp. on Fault Tolerant Computing,* Vienna, Austria, pp. 344-349, June 1986.

[78] Y. Saad and M. Schultz, "Topological Properties of Hypercubes," *IEEE Trans. on Comp.,* vol. C-37, no. 7, pp. 867-872, July 1988.

[79] D. Scott and J. Brandenburg, "Minimal Mesh Embeddings in Binary Hypercubes," *IEEE Trans. on Comp.,* vol. C-37, no. 10, pp. 1284-1285, Oct. 1988.

[80] S. Soh, S. Rai, and J. Trahan, "Improved Lower Bounds on the Reliability of Hypercube Architectures,"*IEEE Trans. on Par. Dist. Sys.,* vol. 5, no. 4, pp. 364-378, April 1994.

[81] A. Sen, A. Sengupta, and S. Bandyopadhyay, "On the Routing Problem in Faulty Supercubes," *Information Processing Letters,* vol. 42, pp. 39-46, Apr. 1992.

[82] A. Sen, A. Sengupta, and S. Bandyopadhyay, "On Some Topological Properties of Hypercube, Incomplete Hypercube and Supercube," *Seventh Int'l Par. Proc. Symp.,* 1993.

[83] S. Simonson and I. Sudborough, "On the Complexity of Tree Embedding Problems," *Information Processing Letters,* vol. 44, pp. 323-328, Dec. 1992.

[84] G. Stamoulis and J. Tsitsiklis, "An Efficient Algorithm for Multiple Simultaneous Broadcasts in the Hypercube," *Information Processing Letters,* vol. 46, pp. 219-224, July 1993.

[85] G. Stamoulis and J. Tsitsiklis, "Efficient Routing Schemes for Multiple Broadcasts in Hypercubes," *IEEE Trans. on Par. and Dist. Sys.,* vol. 4, no. 7, pp. 725-739, July 1993.

[86] Sultan and Melhem, "An Efficient Modular Spare Allocation Scheme and its Application to Fault-Tolerant Binary Hypercubes," *IEEE Trans. on Par. and Dist. Sys.,* vol. 2, no. 1, pp. 117-126, Jan. 1991.

[87] S. Tien and C. Raghavendra, "Algorithms and Bounds for Shortest Paths and Diameters in Faulty Hypercubes," *IEEE Trans. on Par. and Dist. Sys.,* vol. 4, no. 6, pp. 713-718, June 1993.

[88] J. Tien, C. Ho, and W. Yang, "Broadcasting on Incomplete Hypercubes," *3rd Great Lakes Symp. on VLSI Design Automation of High Performance VLSI Systems,* Mar. 1993.

[89] S. Tien, C. Raghavendra, and M. Sridhar, "Reconfiguring Embedded Task Graphs in Faulty Hypercubes by Automorphisms," *Proc. 23rd Annual Hawaii Int'l Conf. on Sys. Sciences,* pp. 91-100, Jan. 1990.

[90] N.F. Tzeng and S. Wei, "Enhanced Hypercubes," *IEEE Trans. on Comp.,* vol. C-40, no. 10, pp. 1362-1371, Oct. 1991.

[91] A. Varma, and C. S. Raghavendra, Reliability Analysis of Redundant-Path Interconnection Networks, *IEEE Transactions of Reliability*, vol. 38, no.1 , pp. 130-137, April 1989.

[92] D. Wang, "Diagnosibility of Enhanced Hypercubes," *IEEE Trans. on Comp.,* vol. C-43, no. 9, pp. 1054-1061, Sept. 1994.

[93] A. Wagner, "Embedding All Binary Trees in the Hypercube," *J. of Par. and Dist. Computing,* vol. 18, pp. 33-43, 1993.

[94] A. Wang, R. Cypher, and E. Mayr, "Embedding Complete Binary Trees in Faulty Hypercubes," *the 3rd IEEE Symp. on Parallel and Distributed Processing,* Dallas, pp. 112-119, 1991.

[95] W.W. White, "Mapping General Trees and Graphs into the Hypercube," *Proc. of the ISCA Int'l. Conf., Parallel and Distributed Computing,* pp. 157-161, Oct. 14-16, 1993.

[96] J. Wu and K. Yao, "Multicasting in Injured Hypercubes Using Limited Global Information," *the 5th IEEE Symp. on Parallel and Distributed Processing,* pp. 548-555, 1993.

[97] J. Wu and K. Yao, "A Limited-Global-Information-Based Multicasting Scheme for Faulty Hypercubes," *IEEE Trans. on Comp.,* vol. 44, no. 9, pp. 1162-1167, Sep. 1995.

[98] C. Yang *et al.* "A Reconfigurable Modular Fault-Tolerant Hypercube architecture," *IEEE Trans. on Par. and Dist. Sys.,* vol.5 , no. 10, pp. 1018-1032, Oct. 1994.

[99] C. Yang and S. Wu, "Fault-Tolerance on Boolean n-Cube Architectures," *First European Dependable Computing Conference,* Berlin, Germany, Oct. 4-6, 1994.

[100] C. Yang, J. Wang, J. Lee, and F. Boesch, "Graph Theoretic Reliability Analysis for the Boolean n cube Networks," *IEEE Trans. on Circuits and Systems,* vol. 35, no. 9, pp. 1175-1179, Sept. 1988.

[101] P. Yang and C. Raghavendra, "Embedding and Reconfiguration of Binary Trees in Faulty Hypercubes," *Sixth Int'l Par. Proc. Symp.,* pp. 2-9, 1992.

[102] P. Yang and C. Raghavendra, "Reconfiguration of Binary Trees in Faulty Hypercubes," *Seventh Int'l Par. Proc. Symp.,* pp. 401-405, 1993.