Information Sciences 276 (2014) 295–318

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Reliability analysis and fault tolerance for hypercube multi-computer networks

Mostafa Abd-El-Barr [a,*], Fayez Gebali [b]

[a] Dept. of Information Science, Kuwait University, Kuwait
[b] Dept. of Elect. & Comp. Eng., University of Victoria, Victoria, BC V8W 3P6, Canada

ARTICLE INFO

*Article history:*
Received 4 March 2011
Received in revised form 20 August 2013
Accepted 19 October 2013
Available online 14 November 2013

*Keywords:*
Hypercube multi-computer networks
Fault tolerance
Fault-tolerant routing
Reliability computation
Reliability analysis
Multicasting and broadcasting in faulty hypercube

ABSTRACT

A multi-computer system (MCS) offers the high speed and throughput needed in solving computing-intensive problems. Two main components constitute a MCS. These are the processing elements (PEs) and the interconnection network (IN). A faulty IN can lead to data losses and/or throughput degradation. Hence, it is necessary to consider the fault tolerance and reliability aspects in assessing the efficacy of INs. This paper provides coverage of the fault tolerance and reliability aspects of hypercube multi-computer networks (HCNs). In particular, we cover three broad aspects: task-based reliability, fault-tolerant routing, and communication in faulty HCNs. Our coverage includes introducing the particular HC architecture, analyze its reliability and assess its fault tolerance. The analysis provided in the paper is deemed helpful to HCN designers in making informed decisions about the appropriate approaches that can be used to assess the reliability and fault tolerance of existing HCNs.

© 2013 Elsevier Inc. All rights reserved.

## 1. Introduction

Every high-performance computing system now is based on multicore processor technology. In fact, multicore processors can be found in the smallest embedded systems all the way up to server farms [77,33,85,26,60]. The performance of such multicore systems rely heavily on the interconnection network connecting these cores [34,33,21]. This has sparked new interest in the study of interconnection networks for such systems in general. Of note is the hypercube network due to its inherent structural advantages and fault-tolerance properties. A simple search of IEEE Xplore database indicates that the rate of published papers related to hypercube multiprocessor systems averages at about 20 papers annually since 1980. In terms of topology, interconnection networks can be classified as static versus dynamic. Static interconnection networks (INs) include linear, ring (loop), trees, mesh, and hypercube (HC) networks. Dynamic INs include bus-based and switch-based networks. Bus-based networks include single-bus and multiple-bus while switch-based networks include single-stage (SS), multi-stage (MS), and the crossbar. Hypercube (HC) multi-computer interconnection networks represent distinct topology in connecting distributed memory multi-computer systems. A number of distributed memory parallel computers utilizing HCs have been introduction in the literature. Examples of these include iPSC/2, iPSC/860, nCUB/2, Caltech Hypercube, and SGI Origin [4,3,2,1,49,86]. Performance measures such as speed and throughput have traditionally been used as the central measures for the efficiency of multi-computer systems. However, designers of such systems assert that assessing the fault tolerance and reliability, is as important as other performance measures. It is possible to achieve fault

* Corresponding author. Tel.: +965 2498 3301.
*E-mail address:* mostafa.abdelbarr@gmail.com (M. Abd-El-Barr).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.10.031

tolerance in multi-computers through various forms of redundancy. In particular, hardware and software redundancy techniques are the two commonly used techniques.

**Definition 1.** An $n$-dimensional hypercube $Q_n$, is defined recursively as

$$Q_n = \begin{cases} Q_0 & n = 0 \\ K_2 \times Q_{n-1} & n > 0 \end{cases} \quad (1)$$

where $Q_0$ is a trivial graph having one node, $K_2$ is a complete graph with two nodes, and $\times$ is the product operation of two graphs. □

According to the above definition, a 2-cube is constructed using two 1-cube, a 3-cube is constructed using 2 2-cube, etc. The address of a node in an n-cube is specified by $S_nS_{n-1}\cdots S_1$ , where $S_i \in \{0,1\}$. Fig. 1 shows 4-cube architecture [6].

**Definition 2.** The Hamming distance between two addresses $S_nS_{n-1}\cdots S_1$ and $D_nD_{n-1}\cdots D_1$ in HC $Q_n$ is defined as:

$$H(S,D) = \sum_{i=1}^{n} S_i \oplus D_i \qquad \square \quad (2)$$

A path between a source node $S$ and a destination node $D$ in a hypercube is specified by a coordinate sequence. The coordinate sequence contains all the dimensions which need to be traversed to reach $D$ from $S$. An important property of a hypercube is the availability of a number of paths between any pair of nodes [81,83]. In $Q_n$, there are $k$ disjoint paths of length $k$ and $n-k$ disjoint paths of length $k+2$ between any pair of nodes $S$ and $D$ with $H(S,D) = k$. Consider, for example, the 4-cube shown in Fig. 1. Assume that the source node $S$ address is 0000 and the destination node $D$ address is 1110. The Hamming distance is 3 and therefore there are three paths of length 3 (shown using solid lines) and one path of length 5 (shown using dashed lines).

It has been shown that in $Q_n$ if the number of faulty components are $f$ links and $g$ nodes is less than $n$, then there is at least one path of length less than or equal to $k+2$ between any two non-faulty nodes $S$ and $D$ where $k = H(D,S)$. The availability of abundant disjoint paths is a characteristic that makes a hypercube interconnection network a desirable fault- tolerant network.

We start this discussion with some definitions then review some hypercube architectures.

**Definition 3.** Network degree ($d$): is defined as the maximum number of ports per node for all nodes in the network. □

**Definition 4.** Network Diameter ($D$): is defined as the maximum shortest path consisting of distinct hops, between any two nodes in the network. □

**Definition 5.** Cost is defined as the product of the degree ($d$) and the Diameter ($D$) of a network, i.e. □

**Definition 6.** A failure rate of a system, $\lambda$, is defined as the expected number of failures of the system per unit time. □

**Definition 7.** The Reliability of a system, $R(t)$, is defined as the conditional probability that the system operates correctly throughout the time interval $[t_0, t]$ given that it was operating correctly at $t_0$. □

A relationship between reliability and time that has been widely accepted for digital systems is the exponential function, $R(t) = e^{-\lambda t}$. Reliability modeling aims at using abstract representation of systems as means for assessing their reliability. Two basic techniques have been used. These are the empirical and the analytical techniques. In this paper, we use the analytical model. This model is based on the use of the probability of failure of individual components in computing the probability of failure of a given system. The overall failure probability of a system will depend on the individual failure probability of its individual components and the way these components are interconnected. Two techniques exist for assessing system

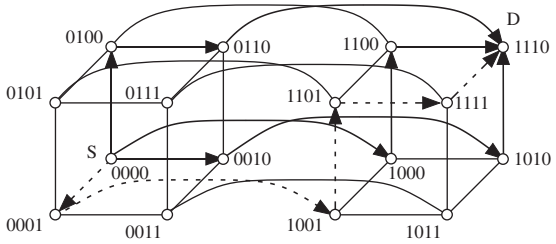


**Fig. 1.** A 4-cube architecture showing all possible paths between nodes 0000 and 1110.

reliability according to the analytical technique. These are the Combinatorial and the Markovian models. Unless otherwise indicated, the combinatorial model is used in the analysis provided in this paper. According to this technique, the number of ways in which a system can continue to operate, given the probability of failure of its individual components, is enumerated [25].

In the context of HCNs, there exist a number of measures for hypercube reliability. According to Terminal Reliability (TR), the system is considered working as long as two specified non-faulty nodes are connected. According to Network Reliability (NR), the system is considered working as long as all non-faulty nodes are connected. According to Subcube Reliability (SR), the system is considered working as long as some minimum-degree subcube is functional. Reliability measures TR and NR are useful in measuring the robustness of system topology. Reliability measure SR is particularly useful in HCNs due to its recursive nature and the ability to execute a given algorithm on various smaller size sub-cubes. In the next section, we begin our coverage by introducing the basic issues involved in reliability computation in non-redundant HCNs.

## 2. Reliability model using graph theoretic approach

The topological structure of the hypercube network is typically represents by the undirected graph $G(V,E)$ where $V$ is the set of vertices corresponding to the switches or routers of the network and $E$ is the set of edges connecting the vertices. The graph is of limited use for high-degree networks where the graph becomes very had to use. A simpler approach is to use the adjacency matrix concept [13]. The adjacency matrix $A$ corresponding to a given undirected graph $G(V,E)$ is a symmetric square matrix such that element $a_{i,j}=1$ if there is an edge connecting vertices $i$ and $j$ [79]. The number of rows or columns of $A$ is $N$ which is the number of vertices of the network. $N$ is related to the dimension or degree $n$ of the hypercube network by the relation

$$N = 2^n \tag{3}$$

We note that the diagonal elements $a_{i,i}=0$ because self-loops are not allowed. We use a single label to number our vertices corresponding to the row or column in $A$ and the indices range between 0 and $N-1$. We will find it convenient to represent the index of a vertex using binary notation using $n$ bits to represent the numbers between 0 and $N-1$.

### 2.1. Generating the adjacency matrix

Vertex $v_j$ is connected to vertex $v_i$ when the binary representations of the two nodes differs only in one bit. In other words vertices $v_i$ and $v_j$ have a Hamming distance of 1. For a hypercube network of degree $n$, the adjacency matrix is $N \times N$ and we use the last property of the Hamming distance between connected nodes to generate the hypercube adjacency matrix:

$$A_{i,j} = \begin{cases} 1 & \text{when } \mathcal{H}(i,j)=1 \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

where $\mathcal{H}(i,j)$ is the Hamming distance between the two binary representations of indices $i$ and $j$. As an example, $A$ for the case $n=3$ and $N=8$ is given by

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{5}$$

The graph associated with such network is simple to visualize for this case and is shown in Fig. 2 where the node indices are indicated in the figure corresponding to the row and column indices of $A$.

### 2.2. New network algebra for determining path length

Raising $A$ to the power $k$, i.e. $A^k$ gives us the number of $k$-hop paths between any two nodes. However, simple matrix multiplication does not give the correct result in a symmetric adjacency matrix since it include self-loops. For example, $A^2$ gives all the 2-hop paths between vertices. For our case, simple squaring of the adjacency matrix produces:

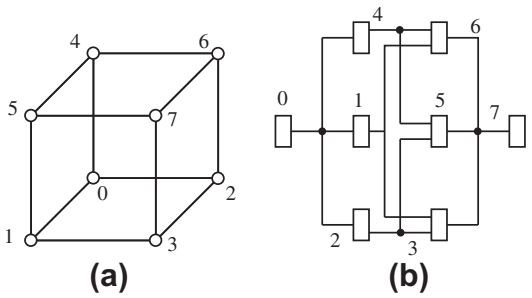


**Fig. 2.** Example of a hypercube network when $n = 3$ and $N = 8$. (a) Example 3-cube. (b) Reliability Block Diagram (RBD).

$$A^2 = \begin{bmatrix} 3 & 0 & 0 & 2 & 0 & 2 & 2 & 0 \\ 0 & 3 & 2 & 0 & 2 & 0 & 0 & 2 \\ 0 & 2 & 3 & 0 & 2 & 0 & 0 & 2 \\ 2 & 0 & 0 & 3 & 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 & 3 & 0 & 0 & 2 \\ 2 & 0 & 0 & 2 & 0 & 3 & 2 & 0 \\ 2 & 0 & 0 & 2 & 0 & 2 & 3 & 0 \\ 0 & 2 & 2 & 0 & 2 & 0 & 0 & 3 \end{bmatrix} \tag{6}$$

All the diagonal entries are false since they indicated presence of three 2-hop to each vertex— node 0 would have the loop $0 \rightarrow 1 \rightarrow 0$ which traverses the same path twice. This problem has been recognized earlier but no solution was proposed [79]. The off-diagonal entries are correct, however. We can see that vertex 0 has two 2-hop paths to vertex 5 for example.

The multiplication algorithm we explain here is developed to cancel the self-loop effect that results when the adjacency matrix $A$ is raised to an arbitrary power $k > 1$ [29]. Algorithm 1 shows how we can generate a new matrix $A^k$ such that all entries resulting from redundant paths and self-loops are eliminated.

**Algorithm 1.** Pseudo code for new matrix multiplication algorithm to eliminate self-loops.

**Require:** Input: $A, K$ // to find $A^K$
$B = \mathbf{I}; F = \mathbf{I}$ // $F$ is a flag matrix to prevent self loops and $\mathbf{I}$ is unit matrix
**for** $k = 1 : K$ **do**
  $B = B \times A$;
  **for** $i = 1 : n$ **do**
    **for** $j = 1 : n$ **do**
      **if** $(B(i,j) > 0)\&\&(F(i,j) == 1)$ **then**
        $B(i,j) = 0$;
      **end if**
      **if** $(B(i,j) > 0)\&\&(F(i,j) == 0)$ **then**
        $F(i,j) = 1$;
      **end if**
    **end for**
  **end for**
**end for**
**return** $B$;

### 2.3. Reliability measurement using minimum edge cut-set

An edge cut-set is a set of edges whose removal from the graph will disconnect the vertices and partition the network into two or more subnetworks [78]. For a hypercube network the edge-connectivity of the $n$-dimensional hypercube is equal to $n$ [78,57].

Assuming the probability of link failure is $p_l$. A vertex will be disconnected from the network with probability $p_v$ given by

$$p_v = p_l^n \tag{7}$$

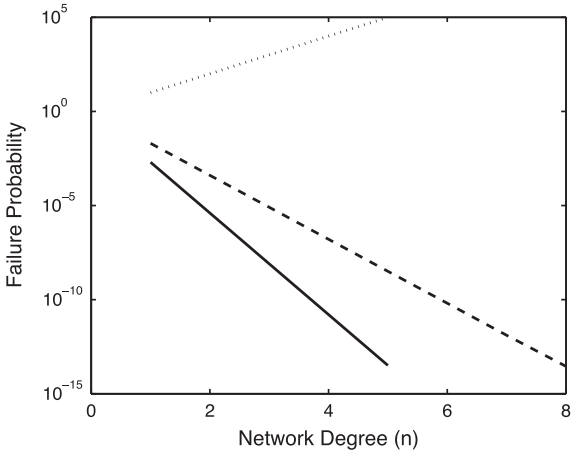


**Fig. 3.** Failure probability of hypercube network versus network degree $n$. Solid line is when $p_l = 0.001$ and dashed line is when $p_l = 0.01$. The dotted line is for the reliability ratio.

We immediately see from the above equation that increasing the network degree reduces the probability of a vertex being completely disconnected for the same link failure probability. Therefore, we expect to see improved reliability for increasing network degree as will be verified by numerical simulations.

The network fails when one or more nodes of the network get disconnected. The network failure probability is given by

$$\begin{aligned} p_n = & \sum_{i=1}^{N} (N//i) p_v^i (1-p_v)^{N-i} \\ = & \ 1-(1-p_v)^N \qquad (8) \\ Approx & \ Np_v \quad \text{when } Np_v \ll 1 \qquad (9) \end{aligned}$$

We define the reliability of a network with $N$ nodes $R(N)$ as the probability that all network vertices will not be disconnected

$$R(N) = 1 - p_n \qquad (10)$$

Assuming two identical networks but two different link failure probability, the reliability ratio can be approximated as

$$\begin{aligned} \text{Reliability ratio} = & \ \frac{R_1(N)}{R_2(N)} \\ Approx & \ \left(\frac{1-p_{v1}}{1-p_{v2}}\right)^N \qquad (11) \end{aligned}$$

Fig. 3 shows the failure probability of hypercube network versus network degree $n$. Solid line is when $p_l = 0.001$ and dashed line is when $p_l = 0.01$. The dotted line is for the reliability ratio. We see that for a fixed $p_l$, the network failure probability $p_n$ is exponentially decreased. This results in exponentially increased reliability. Of course, reducing $p_l$ results in reduced $p_n$ and increased reliability. The increase is exponentially proportional to the network degree $n$.

## 3. Reliability computation based on network partitioning

In this section, we present the approach for reliability computation in non-redundant hypercube based on partitioning the network into subnetworks and the external links connecting the subnetworks. We start our coverage by considering first the network reliability of the hypercube network. This computation makes use of the recursive nature of the hypercube [87,53]. This recursive nature allows us to decompose a given $n$-dimensional hypercube $Q_n$ into two $(n-1)$-dimensional cubes $Q_{n-1}$ using $2^{n-1}$ exterior links. Similarly, each $Q_{n-1}$ can be decomposed into two $Q_{n-2}$ using $2^{n-2}$ exterior links and so on. The exact Failure probability of the 2-cube can be taken as the base case for this recursion. The exact failure probability of a 2-cube assuming $p_l$ is the link failure probability and fault-free nodes ($p_n = 0$) is given by

$$p_n(2) = 1 - (1-p_l)^4 - 4p_l(1-p_l)^3 \qquad (12)$$

The reliability of the $Q_2$ network is the complement of $p_n(2)$:

$$R(2) = (1-p_l)^4 + 4p_l(1-p_l)^3 \qquad (13)$$

Consider, for example, a given $n$-dimensional hypercube, $A$. Such $Q_n$ can be decomposed into two $Q_{n-1}$ , call them $B$ and $C$, connected using $(n-1)$-dimension links. In computing the network reliability $R(Q_n)$ of $Q_n$, we consider also that the network

reliability is the function $NR(Q_n)$ which depends on $p_l$ and $p_n$. We have the following three disjoint cases of working, failed congruent $(n-1)$-cubes and exterior links (assuming $p_n = 0$).

### 3.1. Case #1 (C1)

Both $(n-1)$-dimension cubes $B$ and $C$ are operational and that $i$ exterior links, $0 < i \leqslant N/2$, are operational. The probability of case C1 occurring is computed as follows:

$$P(C1) = R(Q_{n-1})^2 \sum_{i=1}^{N/2} (N/2//i)(1-p_l)^i p_l^{N/2-i} \tag{14}$$

This expression reduces to

$$P(C1) = R(Q_{n-1})^2 \left[1 - p_l^{N/2} - (1-p_l)^{N/2} - \frac{Np_l}{2}(1-p_l)^{N/2-1}\right] \tag{15}$$

### 3.2. Case #2 (C2)

At least one $(n-1)$-dimension cube, $B$ or $C$, is operational and the other is not operational and exactly one exterior link failed and the node at the end point of the failed link in the none operating $Q_{n-1}$ is linked to at least one of its neighbors by an interior link. The probability of case C2 occurring is computed as follows:

$$P(C2) = \frac{Np_l R(Q_{n-1})}{2}\left[2(1-p_l^{n-1}) - R(Q_{n-1})\right](1-p_l)^{N/2-1} \tag{16}$$

### 3.3. Case #3 (C3)

At least one $(n-1)$-dimension cube, $B$ or $C$, is operational and $N/2$ exterior links are operational. The probability of case C3 is computed as follows:

$$P(C3) = R(Q_{n-1})[2 - R(Q_{n-1})](1-p_l)^{N/2} \tag{17}$$

Considering cases C1, C2, and C3 the overall network reliability is computed as:

$$R(N) = P(C1) + P(C2) + P(C3) \tag{18}$$

The set of graphs presented in Fig. 4 shows the network reliability for a set of different size HCNs as a function of the link failure probability $p_l$.

Having considered the hypercube network reliability, we now move to consider the terminal reliability $TR$ of HC. In Fig. 5, we present an example for the reliability block diagram (RBD) of a 3-dimensional cube showing the different paths between node 0 and node 7 taking into consideration node failure while assuming perfect links.

Using the combinatorial reliability model and assuming that $R$ is the node reliability, we obtain the following expression for the $TR(0{-}7)$ between node 0 and node 7 in a 3-dimensional cube

$$TR(0{-}7) = -2R^8 + 6R^7 - 3R^6 - 6R^5 + 6R^4 \tag{19}$$

A number of reliability models have been used to compute the terminal reliability of hypercube. Two of these are the shortest paths generation model and the parallel paths generation model. In an n-cube, there are n! shortest paths between any

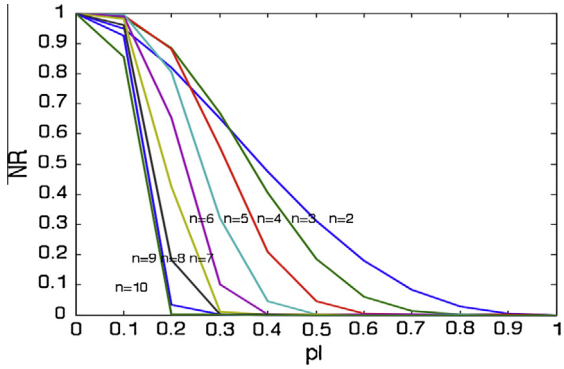


**Fig. 4.** Network reliability graphs for different size HCNs.

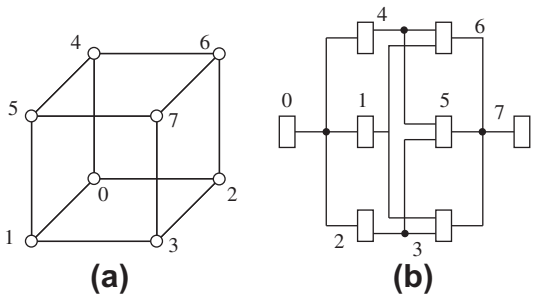


**Fig. 5.** Terminal reliability ($R_t$) computation illustrative example. (a) Example 3-cube. (b) Reliability Block Diagram (RBD).

pair of nodes. Consider, for example the 3-cube shown in Fig. 3. The six (3!) shortest paths between nodes 0 and 7 are: 0137}, {0167}, {0237}, {0257}, {0457}, and {0467}. In computing the terminal reliability, the shortest path model considers all shortest paths together with the recursive nature of the hypercube. On the other hand, the parallel path considers only the disjoint (parallel) paths in computing the terminal reliability of a hypercube. Assuming that the reliability of each node is $R$ and that the Hamming distance of the given cube is $H$, then the two-terminal reliability of an n-cube can be computed as

$$TR(n) = \left[1 - \left(1 - R^{H-1}\right)^{H}\left(1 - R^{H+1}\right)^{n-H}\right]R^2 \tag{20}$$

The terminal reliability for the 3-cube in Fig. 5 is given by:

$$TR(3) = \left[1 - \left(1 - R^{2}\right)^{3}\right]R^2 \tag{21}$$

Having considered the $R_n$ and $R_t$ of a hypercube, we now turn our attention to what is known as task-based (sub-Cube) reliability.

## 4. Sub-cube reliability computation in hypercube

For the purpose of discussion in this section, we assume that the node failure obey the exponential failure law with node failure rate denoted by $\lambda_n$. Unless otherwise specified, links are considered to be perfect. According to the subcube reliability measure, reliability is computed in terms of the number of disjoint sub-cubes that can be embedded in a d-cube in the presence of node and/or link failures [5,8,59]. In particular, the ability of a $d$-cube to embed a $(d-1)$ sub-cube (the largest fault-free sub-cube in a $d$-cube) in the presence of failures is considered. Consider embedding (finding) a fault-free $(d-1)$-sub-cube in a $d$-cube in the presence of node failure. A single node failure will always leave a fault-free $(d-1)$-sub-cube but two node failures could destroy all $(d-1)$-sub-cubes. For example, if node 0 (having address 00..00) and node $(N-1)$ (having address 1111) fail, there is no way of embedding a $(d-1)$-sub-cube. A fault-free $(d-1)$-sub-cube exists if and only if all failures occur such that they can be enclosed in $i$-sub-cube with $i < d$ [5]. Fig. 6 illustrates this concept in the case of a 3-cube.

**Definition 8.** Define $s_i$ as the system state whereby all node failures that have occurred could be enclosed in a maximal $i$ sub-cube, i.e.. a lower order sub-cube that encloses all failed nodes does not exist. □

Consider, for example, the state diagram of a d-cube shown in Fig. 7. In this figure state $s_*$ represents the fault free initial state or perfect state while state $s_d$ represents the case in which all failures have occurred in a maximal $d$ cube. In terms of functional cubes, state $s_i$ is characterized by embedding $(d-i)$ disjoint fault free sub-cubes of order $(d-1), (d-2)$, $i$,

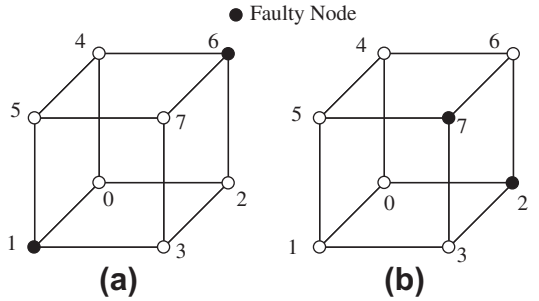


**Fig. 6.** Embedding of fault-free 2-cube in a faulty 3-cube. (a) No fault-free 2-cube. (b) Fault-free 2-cube exists.

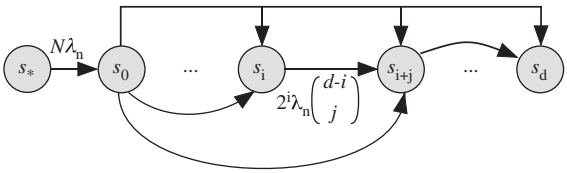


**Fig. 7.** State diagram for a d-cube.

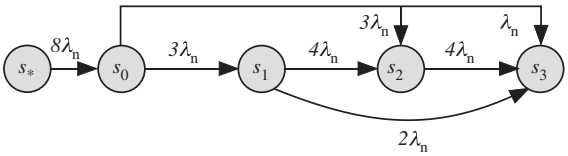


**Fig. 8.** System state diagram for a 3-cube under the node failure model.

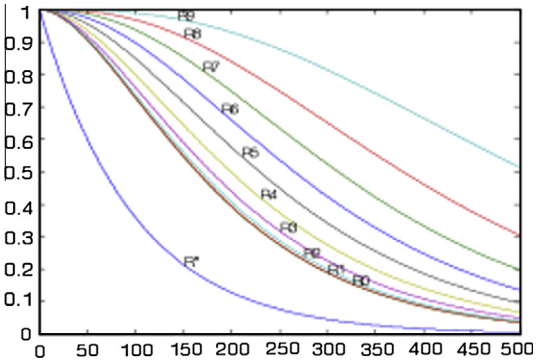


**Fig. 9.** Reliability curves for a number of HCs assuming $\lambda_n = 10^{-5}\ \mathrm{h}^{-1}$.

respectively. Consider the transition into state $s_0$ . A single node failure can cause such transition. There are $N$ ways to cause such transition therefore the transition rate is $N\lambda_n$ (recall that $N2^n$). Now, consider the state transition from state $s_i$ to state $s_{i+j}$. State $s_i$ signifies that all failures have occurred within $i$ sub-cubes. This leaves $(d-i)$ fault free dimensions. State $s_{i+j}$ signifies that all failures have occurred in a $(i+j)$ sub-cube. The transition from $s_i$ to state $s_{i+j}$ means that additional $j$ dimensions become faulty out of the fault free $(d-i)$ dimensions. A total of $j$ dimensions out of the $(d-i)$ can be chosen in $(d-i//j)$ ways. Considering one of these ways we see that the original faulty i dimensions could take up values either 0s or 1s (total of $2^i$ ways). Each of the different ways represents the address of the nodes which can fail to cause the transition. Therefore, the transition rate is labeled as $(d-i//j)2^i\lambda_n$.

Fig. 8 shows the system state diagram for a 3-cube under the node failure model. Let the probability of the system existing in state $s_i$ at time $t$ being $P_i(t)$. The state equations for a $d$-cube system using discrete modeling system are computed as follows:

$$\frac{\partial P_*}{\partial t} = -\lambda_n N P_* \tag{22}$$

$$\frac{\partial P_0}{\partial t} = -\lambda_n (N-1) P_0 + \lambda_n N P_* \tag{23}$$

$$\frac{\partial P_i}{\partial t} = -\lambda_n (N-2^i) P_i + \sum_{j=0}^{i-1} \lambda_n 2^j (d-j//i-j) P_j 0 < i \leqslant d \tag{24}$$

The initial conditions are $P_* = 1$ and $P_i(0) = 0$ for all $i > 0$. It can be shown by induction on $i$ that the solution to this system of equations is as follows [5]:

$$P_i(t) = (d//i)\left[(-1)^{i+1}2^{d-i}e^{-\lambda_n N t} + \sum_{m=0}^{i}(-1)^{i-m}(i//m)2^{d-m}e^{-(N-2^m)\lambda_n t}\right] \tag{25}$$

Define the reliability expressions as follows

$$R_*(t) = P_*(t) \tag{26}$$

$$R_0(t) = P_0(t) + P_*(t) \tag{27}$$

$$R_i(t) = P_i(t) + R_{i-1}(t) \qquad i > 0 \tag{28}$$

Thus $P_i(t)$ is the probability that all failures are enclosed in an $i$ sub-cube, while $R_i(t)$ is the probability that all failures have occurred in a sub-cube of order less than or equal to $i$. Fig. 9 shows a set of sub-cube reliability curves for HCNs of different sizes ($d = 2$ to $d = 10$) and assuming node failure rate $\lambda_n = 10^{-5}\ h^{-1}$. The figure shows that as the size of the subcube, $i$, increases $R_i(t)$ increases. This is because more node failure can be tolerated.

## 5. Fault tolerance in redundant HCNs

Fault tolerance techniques in HCs can be classified as hardware versus software [11,40,70]. Hardware techniques attempt to reconfigure a hypercube using redundant components (nodes and/or links) in order to replace faulty ones.

Software fault tolerance attempt to make use of the inherent redundancy, symmetry, and robustness of the hypercube in reconfiguring the architecture. In the following subsections, we present a number of fault-tolerant hypercube architectures [6,23,36,11].

### 5.1. Hardware-based fault tolerance in HCNs

#### 5.1.1. Use of fault tolerant module (FTM)

According to this approach, a fault-tolerant hypercube architecture is created using fault-tolerant modules (FTMs) together with decoupling networks and soft switches [7]. A FTM consists of m active nodes and k spare (redundant) nodes and a group of k level decoupling networks. When a node fails, the FTM is reconfigured using soft switches so as to bypass the failed node and replace it with a spare node.

#### 5.1.2. Use of fault-tolerant basic block (FTBB)

In this approach, a given hypercube is reconfigured in the presence of faults using a set of hardware switches to provide full spare nodes utilization, i.e.. any spare node can replace any primary node within a given FTBB. The scheme can also support spare failures [66].

#### 5.1.3. Use of spare node embedding

This scheme uses spare nodes attached to specific nodes in the hypercube using certain embedding techniques. A hypercube in this scheme has two types of nodes, called the $P$ and the $S$ nodes. The $P$ and $S$ nodes have different internal architectures. The $P$ node consists of a computation processor (CPU) connected through an internal bus to a local memory and message routing logic consisting of a 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 a CPU and a local memory connected to two internal busses. The DMA and message routing logic is shared between the two processing units, one of which is active under normal conditions; the other is a standby spare. Under failure of any processing element either within the $S$ node or in a nearby $P$ node, the spare processor/memory/bus from the corresponding $S$ node is brought on line. A perfect embedding for allocation of $S$ nodes in the cube requires that each $P$ node be adjacent to exactly one $S$ node in the cube. An algorithm for reconfiguration in the presence of primary node failure is used such that an $S$ node will replace the failed $P$ node [15].

#### 5.1.4. Use of two-level hypercube clustering

For connecting very large number of processors, the link cost of a hypercube becomes prohibitively expensive and therefore hierarchical interconnection network (HINs) schemes should be used. A homogeneous two-level Binary Hypercube/Binary Hypercube (BH/BH) HIN hardware technique is shown in Fig. 10(a) [27].

Each level 1 network is called a *cluster*. A cluster of size 8 and two levels are found to be optimal for connecting very large number of processors in a BH/BH network based on a cost/performance tradeoff. Major disadvantages of HINs include the potential for high traffic rates on inter-cluster links, and thus the potential degradation in performance, and the potential for diminished fault tolerance due to the special role played by the interface nodes. It should be noted that standard techniques for fault tolerance can be applied to the interface nodes to improve the reliability of the BH/BH networks. One such technique is the Replication Technique, see Fig. 10(b). According to this technique, the level-two network is duplicated for each interface node. This network is referred to as BH/BH-RS (RS for Replicated Second level network). It is recommended that the two interface nodes in a cluster be kept as far apart as possible.

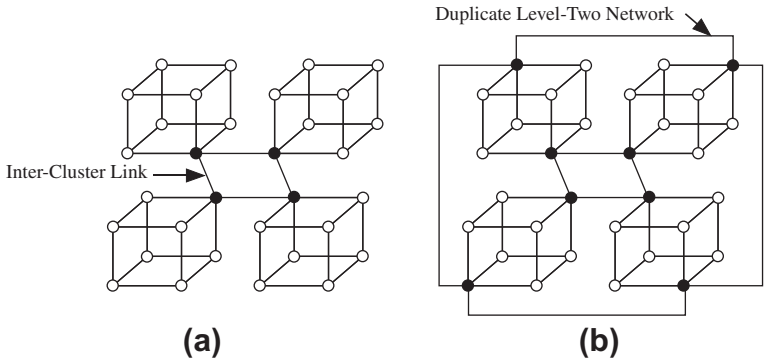


**Fig. 10.** Two examples for using the two-level hypercube clustering. (a) A BH/BH-RS network with a cluster size of 8. (b) A BH/BH HIN with a cluster size of 8.

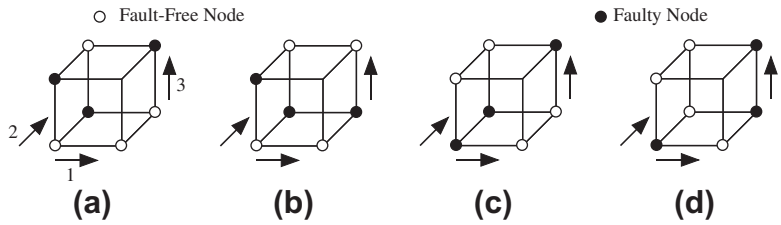


**Fig. 11.** Illustration of the free-dimension concept. (a) All three dimensions are free. (b) Only dimensions 2 and 3 are free. (c) Only dimensions 1 and 3 are free. (d) Only dimensions 1 and 2 are free.

### 5.2. Software-based fault tolerance in HCNs

#### 5.2.1. Use of hypercube free dimension

According to this technique, a given HC is partitioned into a number of subcubes such that each subcube contains at most one faulty node. This partition is made using the concept of free dimension. A dimension in a hypercube $Q_n$ is said to be free if there is no pair of faulty nodes across that dimension. Consider, for example, the $Q_3$ shown in Fig. 11. In part (a) of the figure, all 3 dimensions are free dimensions. In part (b) dimensions 2 and 3 are free but not dimension 1. In part (c) dimensions 1 and 3 are free but not dimension 2. In part (d) dimensions 1 and 2 are free but not dimension 3. From the above illustrations, it can be shown that given $n$ or fewer faulty nodes in $Q_n$ there exists at least one free dimension. With more than $n$ faulty nodes it may still be possible to have some free dimension (s) depending on the fault patterns. Hence, given any $f \leqslant n$ faulty nodes, a $Q_n$ can be partitioned into $(n-f+1)$-dimensional subcubes such that each subcube spanned by the same set of dimensions contains at most one faulty node. Consider, for example the $Q_3$ shown in Fig. 12. The number of faulty nodes in part (a) of the figure is $f = 3$. As can be seen there is no way that such $Q_3$ be partitioned into smaller size subcubes with at most one faulty node. On the other hand, the number of faulty nodes in part (b) of the figure is $f = 2$. It can be verified that $Q_3$ can be partitioned into two subcubes each has at most one faulty node. The free dimension technique can be used to achieve fault tolerance in HCs. This is based on the findings in that given $Q_n$ with $n > 3$ and $f \leqslant n$ it is possible to find out at least two free dimensions. According to such findings, a given $Q_n$ is partitioned into two subcubes along some free dimension. Two copies of the task graph are then embedded; one in each subcube. In addition to requiring global knowledge about faulty nods, this technique leads to a 50% lost in the hypercube performance.

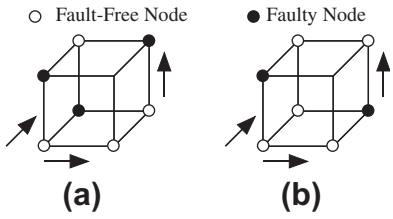


**Fig. 12.** Hypercube partitioning using free dimension.

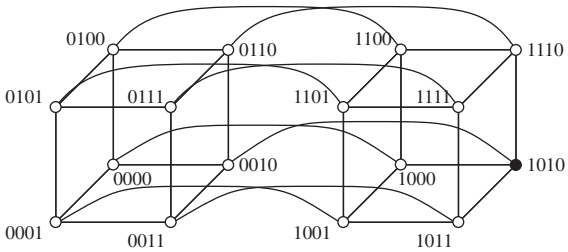


**Fig. 13.** A 4-cube and its spanning tree (ST).

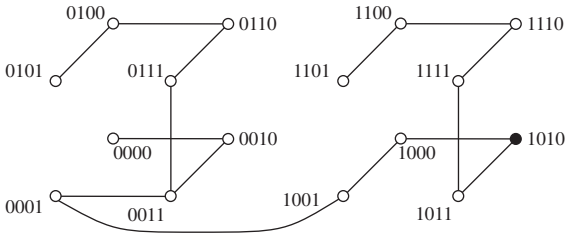


**Fig. 14.** A spanning tree (ST) of the 4-cube in Fig. 13.

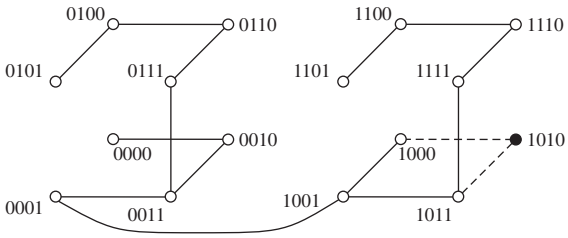


**Fig. 15.** Reconfiguration of ST under faulty node (1010).

*5.2.2. Use of spanning tree*

This approach is based on the use of spanning trees to configure a given hypercube in order to bypass faulty nodes. The algorithm starts by constructing the spanning tree (ST) of the hypercube. This is followed by a reconfiguration phase which requires two steps. During the first step, a faulty node is removed from the ST. During the second reconfiguration step, a new ST is constructed by reconnecting the children of the removed node.

Consider, for example the 4-cube shown in Fig. 13. The ST of the 4-cube shown in Fig. 13 is shown in Fig. 14.

Since node (1010) is faulty, then it should be removed from ST and its children be reconnected as shown in Fig. 15. The reconfiguration process defines a new parents and new children in order to circumvent faulty node(s). For example, in Fig. 15 the new children of node (1001) become node (1000) and node (1011) while the parent of node (1011) becomes node (1001).

*5.3. Reliability performance measures*

In order to provide the reader with reliability performance measure of hardware-based fault tolerance hypercube architecture, we present three well-known HCNs together with their corresponding reliability equations.

*5.3.1. Rennels' scheme [58]*

In this HC architecture an $n$-cube with $2^s$ sub cubes, each containing $2^m$ nodes, where $n = m + s$. One spare node is provided in 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 sub-cube and to the $s$ neighbors in other neighboring sub-cubes. Two crossbar switches are used for reconfiguration. According to this scheme, each sub-cube of order $m$ can tolerate at most one fault. There are $2^s$ such sub-cubes. The system reliability is given by:

$$R_R = \left[r^{2^m} + 2^m \times r^{2^m}(1-r)\right]^{2^s} \tag{29}$$

where $r$ is the reliability of each node. Table 1 gives the system reliability values.

**Table 1**
Rennels' system reliability model.

| $r$ | 1 | 0.9 | 0.8 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $R_R$ | 1 | 0.844 | 0.544 | 0.279 | 0.114 | 0.035 | 0.008 | 0.001 | 0.0004 | 0.0002 | 0.0 |

**Table 2**
Chau's system reliability model for $m = 3$ and $n = 3$.

| $r$ | 1 | 0.9 | 0.8 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $R_C$ | 1 | 0.775 | 0.436 | 0.196 | 0.071 | 0.02 | 0.004 | 0.0004 | 0.0001 | 0.000 | 0.00 |

*5.3.2. Chau's scheme [20]*

In this scheme 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 FTMs. Assuming that the $n$-cube consists of FTMs having $2^m$ primary nodes and $k$ spare nodes, the system reliability can be computed recursively as:

$$R_C = \left[R_{m,n,k-1} + \left(2^n + k - 1//k\right) \times r^{2^n}(1-r)^k\right]^{2^{n-m}} \tag{30}$$

where $k$ is the number of levels of decoupling networks ($k$ spares in $m$-cube fault module) and $R_{n,n,0} = 2^{2^n}$. For $m = 3$ and $n = 3$, we can write, after some simplification:

$$R_{3,3,0} = r^8 \tag{31}$$

$$R_{3,3,1} = r^8 + 8r^8(1-r) \tag{32}$$

$$R_{3,3,2} = r^8 + 8r^8(1-r) + 36r^8(1-r)^2 \tag{33}$$

Table 2 gives the system reliability values for $m = 3$ and $n = 3$.

*5.3.3. Sultan and Melhems scheme [66]*

In this scheme a fault tolerant building block (FTBB) consists of $M$ primary nodes and $k$ spare nodes. Multiplexers and de-multiplexers are used to configure the system such that a faulty primary node can be replaced by a spare node. The scheme is also set to support faulty spare nodes. The system reliability of this scheme is:

$$R_{S-M} = R_{FTBB}^{2^{n-m}} \tag{34}$$

where

$$R_{FTBB} = \sum_{i=0}^{k} (m + k//i) r^{m+k-i}(1-r)^i \tag{35}$$

Assuming that $k = 1$ $m = 2$ and $n = 3$, we get:

$$\begin{aligned} R_{S-M} &= \left[\sum_{i=0}^{k} (3//i) r^{3-i}(1-r)^i\right]^2 \\ &= \left[r^3 + 3r^2(1-r)\right]^2 \end{aligned} \tag{36}$$

Table 3 gives the system reliability values for $k = 1, m = 2$ and $n = 3$.
Table 4 compares the three models for the case $k = 1, m = 2$ and $n = 3$.

**Table 3**
Sultan and Melhem's reliability model for $k = 1, m = 2$ and $n = 3$.

| $r$ | 1 | 0.9 | 0.8 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $R_{S-M}$ | 1 | 0.945 | 0.803 | 0.615 | 0.420 | 0.25 | 0.124 | 0.047 | 0.011 | 0.001 | 0 |

**Table 4**
Comparing Rennels', Chau's and Sultan and Melhem's reliability models for the case $k = 1, m = 2$, and $n = 3$.

| $r$ | 1 | 0.9 | 0.8 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $R_R$ | 1 | 0.844 | 0.544 | 0.279 | 0.114 | 0.035 | 0.008 | 0.001 | 0.0004 | 0.0002 | 0.0 |
| $R_C$ | 1 | 0.775 | 0.436 | 0.196 | 0.071 | 0.02 | 0.004 | 0.0004 | 0.0001 | 0.000 | 0.00 |
| $R_{S-M}$ | 1 | 0.945 | 0.803 | 0.615 | 0.420 | 0.25 | 0.124 | 0.047 | 0.011 | 0.001 | 0 |

We note that the reliability models of Rennels and Chua consistently give comparable values. Sultan and Mulehm's reliability model gives consistently higher reliability estimates and diverge widely for small values of $r$.

## 6. Performance measures of hierarchical hypercube architectures

In this section, we introduce a number of hierarchical hypercube architectures that extend the properties of the basic hypercube. In each case, we introduce the architecture, and two of its characteristics, i.e. the degree and the diameter (for definitions, please refer to Section 1) (see Table 5).

### 6.1. Hierarchical hypercube (HHC) [82]

The structure of an $n$-HHC consists of three levels of hierarchy. At the lowest level of hierarchy, there is a pool of $2^n$ nodes. These nodes are grouped into $2^m$ clusters of nodes each, and the nodes in each cluster are interconnected to form an $m$-cube called the Son-cube or the S-cube. The set of the S-cubes constitutes the second level of hierarchy. A father-cube or the F-cube connects the $2^{n-m} = 2^{2^m}$ S-cubes in a hypercube fashion. It is assumed that $n = 2^m + m$ for a nonnegative integer $m$. The Fig. 16 shows a 5-HHC. The degree of an HHC is $(m+1)$ while its diameter is $2^{m+1}$.

### 6.2. Block Shift Network (BSN) [54]

There are $N = 2^2$ nodes in a BSN. In each step of constructing a BSN network, only $(a)$ bits can be changed within the section of the right most bits $(b)$. This gives the resulting network the label BSN $(a, b)$. Changing the two parameters defines the network connection type. An example BSN $(2,1)$ is shown in Fig. 17. The degree of a BSN is computed as $(2^a - 1) \times (a/b) + 2$ and its diameter is $(1 + b/a)\lceil n/b \rceil$.

### 6.3. The folded HC basic clusters (HFCube) [67]

A hierarchical interconnection network using folded hypercubes as basic clusters is denoted as HFCube. An HFCube $(n, n)$ has $2^n$ clusters, where each cluster is a folded hypercube FHC $(n)$. Each node in the HFCube $(n, n)$ has $(n+2)$ links connected to it. Accordingly, the degree and diameter of a HFCube $(n, n)$ is $(n+2)$ and $(n+1)$, respectively. An HFCube $(3,3)$ is illustrated in Fig. 18.

**Table 5**
Comparison of the properties of hierarchical hypercube architecture.

| HC architecture | # Nodes | Degree $d$ | Diameter $D$ | Cost $C = d \times D$ |
|---|---|---|---|---|
| HHC | $2^{n-2^m+m}$ | $m+1$ | $2^{m+1}$ | $(m+1) \times 2^{m+1}$ |
| BSN (a, b) | $2^n$ | $(2^a - 1)(b/a) + 2$ | $(1 + b/a)\lceil n/b \rceil$ | $2(2^a + 1)$ for $a = b$ |
| HFCube | $2^{2n}$ | $n+2$ | $n+1$ | $(n+1)(n+2)$ |
| FoLH | $7^n \times 2^n$ | $3n$ | $3n$ | $(3n)^2$ |
| THIN | $3^n$ | 3 | $2^{\log_3 N} - 1$ | $3\left(2^{\log_3 N} - 1\right)$ |
| (HCN) | $2^{2n}$ | $n+1$ | $n + \lfloor (n+1)/3 \rfloor + 1$ | $(n+1)(n + \lfloor (n+1)/3 \rfloor + 1)$ |

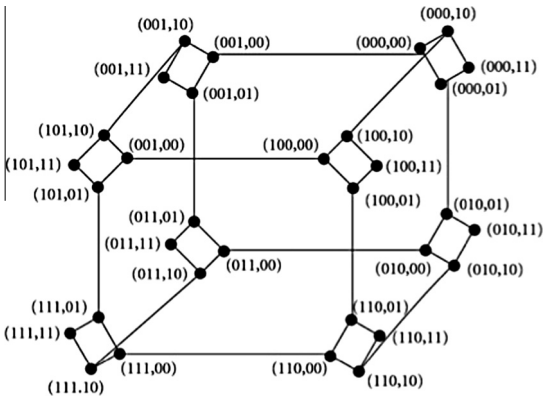


**Fig. 16.** The HCC hypercube network.

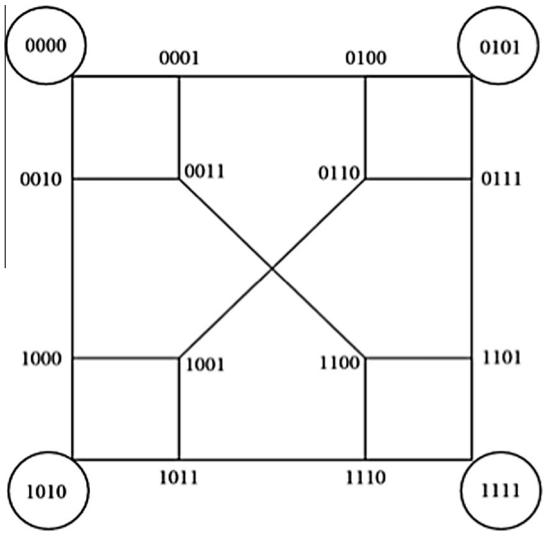


**Fig. 17.** The BSN hypercube network.

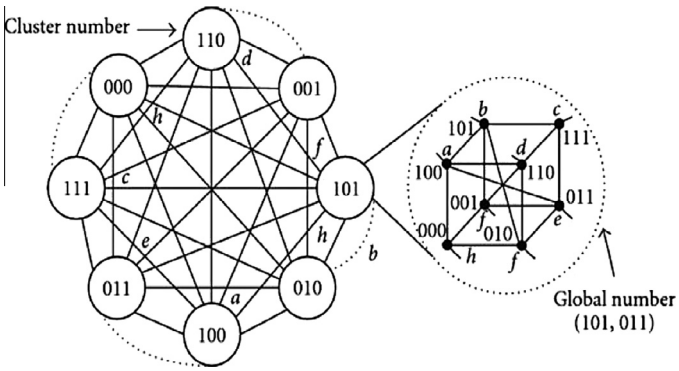


**Fig. 18.** The HFC hypercube network.

### 6.4. The folded Heawood (FoLH) [61]

A Heawood graph has fourteen nodes with twenty-one links connecting them. Each node in a Heawood network has three neighboring nodes. For any pair of nodes, there are three paths for routing a message between them and the minimum length of cycles containing any pair of nodes is 6. The Flooded Heawood HIN is obtained by recursively expanding the basic Heawood network as illustrated in Fig. 19.

### 6.5. The Triple-based HIN (THIN) [41]

The Triple-based HIN (THIN) is a hierarchal, symmetric, and scalable hierarchical HC based architecture. The degree and diameter of THIN is 3 and $2^{\log_3 N} - 1$, respectively. Fig. 20 shows a level 3 THIN.

### 6.6. Hierarchical Cubic Networks (HCN) [56]

The Hierarchical Cubic Network HCN is a hierarchical network consisting of $2^n 4$ clusters, each of which is an $n$-dimensional hypercube. Each node in the HCN $(n, n)$ has $(n+1)$ links connected to it. The HCN uses almost half as many links as a comparable hypercube and yet has a smaller diameter than a comparable hypercube while emulating desirable properties of a hypercube. Fig. 21 shows HCN (2,2). The degree of an HCN $(n, n)$ is $(n+1)$ while its diameter is $n + \lfloor (n+1)/3 \rfloor + 1$.

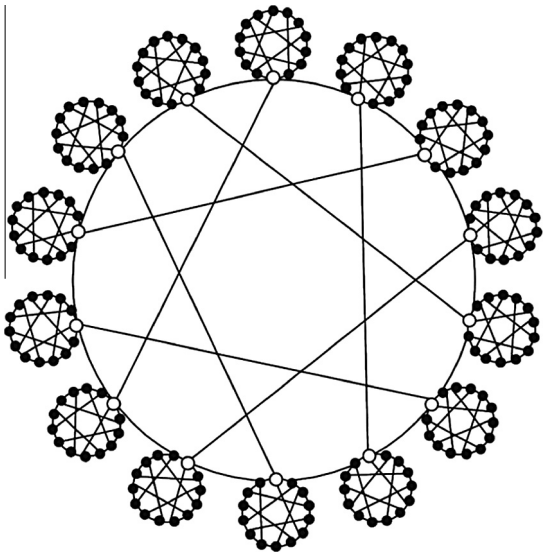

**Fig. 19.** The FoLH hypercube network.

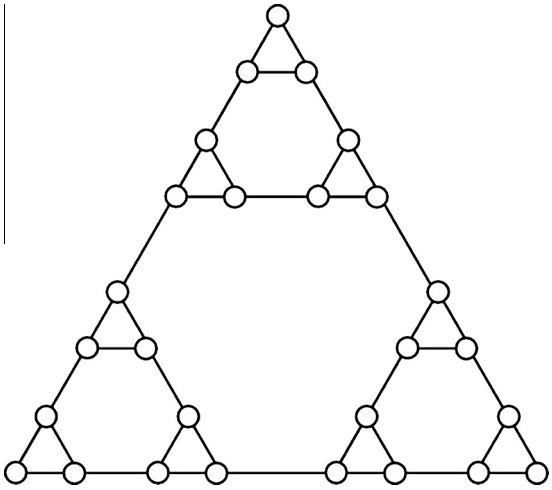

**Fig. 20.** The THIN hypercube network.

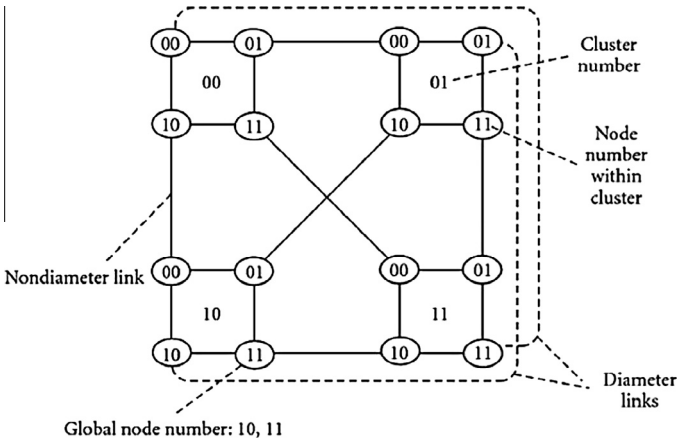


**Fig. 21.** The HCN hypercube network.

## 7. Fault-tolerant routing techniques in HCNs

When a hypercube network becomes faulty, the simple e-cube or coordinate sequence algorithms may fail to route messages properly. Therefore, a number of adaptive algorithms have been introduced to achieve correct message routing in the presence of faulty (nodes or links) for both meshes and hypercube networks [14,46,51,18,12,30,35,76,9,10,8,32,68,16,69,48,43,40,63,55,11,19,17,62,52,75,71,50]. Gunes et al. dealt with fault-tolerant unicast routing [39]. Other authors dealt with fault-tolerant multicast routing [37,31,72,64,74,28,65]. Xiang and Chen dealt with fault-tolerant broadcast routing in hypercubes [84].

A possible classification for the hypercube routing algorithms is shown in Fig. 22 [6].

As can be seen, there are three different levels of classification. In the first level, protocols are divided based on whether they are progressive or backtracking. Progressive protocols move forward and have only limited ability to backtrack. Backtracking protocols, on the other hand, searches the network systematically, backtracking as needed. They store history information in the header to ensure that no path is searched more than once. In the second level, protocols are classified according to whether they are profitable or misrouting. Profitable protocols consider only those links for routing the message which are profitable. A profitable link is a link over which a message moves closer to its destination. A misrouting protocol considers both profitable and non-profitable links as candidates for routing decisions at each node. In the third level, protocols can be completely or partially adaptive. A completely adaptive protocol can use all paths in its class. It is not deterred by routing restrictions from establishing a path as the case in partial adaptive protocols. Several adaptive algorithms for routing in faulty hypercube exist in the literatures.

### 7.1. Depth-first search approach [24]

This is an example of a partially adaptive, misrouting, and progressive routing algorithm. It attempts to find the optimal path having a length equal to the hamming distance between the source and destination nodes. The algorithm requires each node to know only the condition of its own links. It guarantees routing only when the number of faulty elements (nodes and links) is less than $n$ (the dimension of the cube). A routed message carries what is called the *Coordinate Sequence* (CS) whose function is to store the dimensions that the message has to follow from a source node to a destination node. The protocol tries to follow the e-cube pattern as much as possible therefore the smallest dimension is tried first. If the smallest dimension specified by the coordinate sequence is blocked, the node tries to route the message in the next higher dimension as specified by the coordinate sequence. If there is no such dimension in the coordinate sequence, then misrouting is performed. In handling misrouting the algorithm uses what is known as a *tag vector* (TV) whose function is to keep track of the spare dimensions that have been used to bypass faulty components. The tag vector is initialized to all 0s at the source. The algorithm keeps track of the remaining length of the path to the destination. If along routing no faulty components are encountered, the remaining length is decremented by 1 and the dimension in which routing is being performed is removed from the coordinate sequence. For each misrouting the data takes two additional hops to reach the destination. Consider the 4-cube shown in Fig. 23. In this figure, faulty links are shown in dashed lines. The message starts at source $S = 0110$ heading

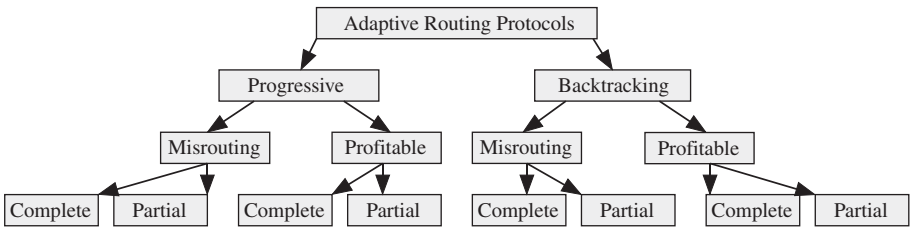


**Fig. 22.** A possible classification for adaptive routing algorithms in hypercube networks.

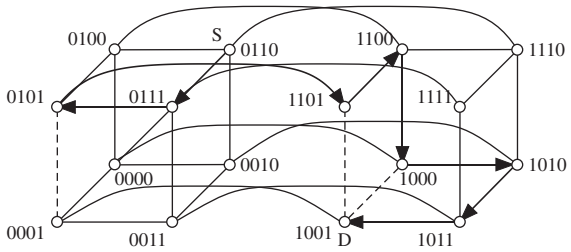


**Fig. 23.** Example routing in 4-cube using depth first algorithm (Algorithm A1).

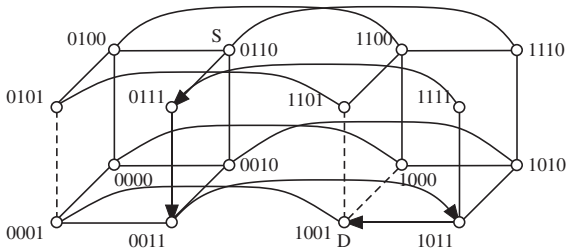


**Fig. 24.** Shortest Path routing using Algorithm A2.

for destination $D = 1001$. The CS of the message at the $S$ beginning is $CS = [1, 2, 3, 4]$. The figure shows the sequence of cube dimensions that have to be traversed from the source node to the destination node. It should be noted that misrouting is performed twice while the message is moving: at node 1101 and at node 1000. As can be seen from the figure, a path length of 8 is traversed. It has been shown that the Depth-First algorithm (called A1 in the literature) is able to successfully route message between any pair of non-faulty nodes as long as the number of faulty components is less than $n$ in a $Q_n$.

The lack of availability of global information about faulty components in the whole hypercube to each node in algorithm A1 leads to an increase in the path length taken while routing messages between any pair of non-faulty nodes. It has been subsequently shown that if every node in a hypercube is provided with information not only about the condition of its own links (as in algorithm A1), but also that of those links which are one hop away from it then the probability of routing messages between any pair of non-faulty nodes via the shortest path (the Hamming distance between the two nodes) increases [59]. A modified algorithm, called A2, which guarantees routing via the shortest path in a hypercube $Q_n$ having total number of faulty components less than $n$ has been introduced in [24]. Algorithm A2 requires that each node, $v$, keeps information about two types of faulty links. The first type, called F1, is the set of those links which block all the optimal paths from $v$ to another node, $u$. The set F1 will be propagated from node $v$ to its neighboring nodes so as to prevent them from choosing node $v$ as a next hop toward node $u$. The second faulty links type, called F0, is the set of faulty links whose status has not yet been propagated from node $v$ to its neighboring nodes. When a node receives the information about the failure of a given link $r$ it will update its F1 and F0 accordingly and check if any further propagation of information on other link failures in F0 is required. In order to show the improvement made over *A1* through *A2* Fig. 24 shows that using the *A2* algorithm, it is possible to route messages between source node $S = 0110$ and destination node $D = 1001$ using the shortest path if algorithm *A2* is used.

A heuristic-based improvement to algorithm *A1* has been proposed in [6]. This algorithm, called *A3*, is introduced below.

### 7.2. Heuristic-based routing algorithm [6]

This heuristic algorithm uses a total of five passes, see Fig. 25. In the first pass, if the message is not routed in the initial pass, another routing pass of the algorithm begins. The failure of the first pass is detected when the originator does not receive an acknowledgment from the destination node that it has received the message, within a pre-decided time interval. If this time interval elapses, the originator sends the same message again, after timing out.

When the above scenario is followed, the message will be considered in pass 1. One of the requirements of the heuristic algorithm is that the dimension in which a message is to be sent will be asked to return a dimension open vector of $n$ bits (for $n$-cube). The bits of this vector will indicate the status of all the dimensions of the next node. After getting this information, each node will make sure that when it routes a message to an intermediate node, all its other remaining dimensions are open. By other dimension, it is meant dimensions other than the one which connects the present node to the next prospective node. In addition, to search for new paths, the process starts with misrouting, rather than with routing. The coordinate sequence is searched and the minimum dimension where a 0 is found is tried for misrouting. A set of four examples that illustrate message routing using the A3 heuristic algorithm in faulty hypercube, under the link fault model, is shown in Fig. 26. Table 6 compares the routing made according to algorithm A1 and that according to A3. It should be noted that the examples in Table 6 are particularly selected to show cases in which algorithm A1 was unable to route messages from the indicated sources and destinations while the heuristic algorithm A3 was successful in routing the same message and under the same faulty links conditions.

### 7.3. Routing techniques based node safety [22]

In this technique, the notion of degree of node safety has been introduced and used to select feasible paths for routing messages in faulty HCs with the main objective being to avoid routing messages through unsafe nodes in order to avoid routing difficulties. A node is said to be unsafe if it has either two or more faulty nearest neighbor nodes or if it has three or more faulty or unsafe nearest neighbors. Fig. 27 shows an example of a 4-cube in which unsafe nodes are identified.

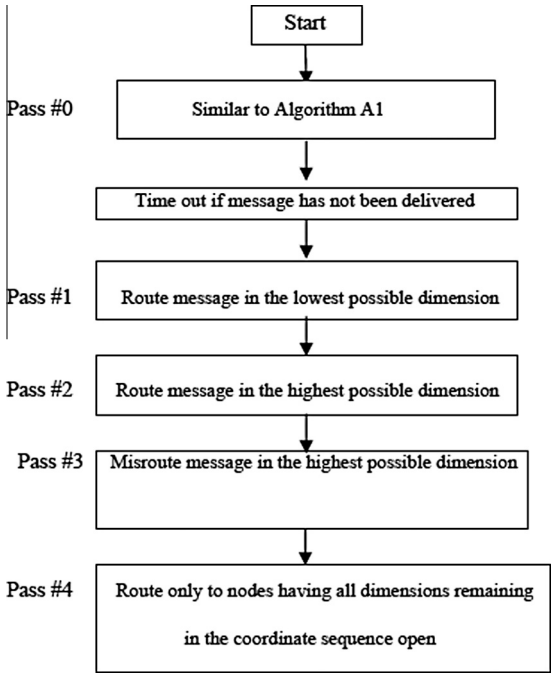


**Fig. 25.** The A3 heuristic-based routing Algorithm.

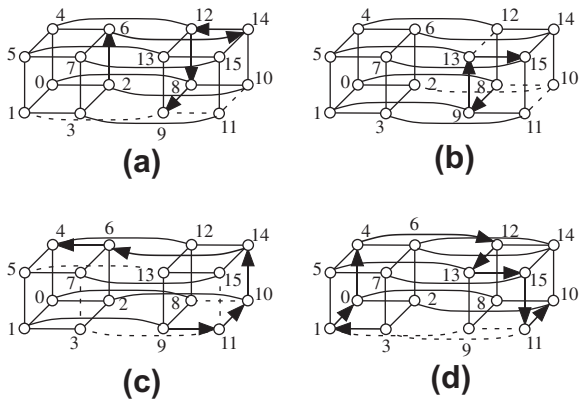


**Fig. 26.** Routing Examples using the heuristic routing algorithm [6]. (a) Routing Example 1. (b) Routing Example 2. (c) Routing Example 3. (d) Routing Example 4.

**Table 6**
Summary of the results using the proposed heuristic routing algorithm.

| Source | Destination | Faulty links | A1 routing | Heuristic routing |
|---|---|---|---|---|
| 0010 | 1001 | See Fig. 26(a) | 2–3–11–3–2–0–1 (Routing fails) | 2–6–14–12–8–9 (Successful) |
| 0000 | 1111 | See Fig. 26(b) | 0–1–3–2–6 (Routing fails) | 0–2–8–9-13–15 (Successful) |
| 1001 | 0100 | See Fig. 26(c) | 9–8–12–13–15–14–12–8 (Routing fails) | 9–11–10–14–6–4 (Successful) |
| 0011 | 1010 | See Fig. 26(d) | 3–1–0–4–6 (Routing fails) | 3–1–0–4–12–13–15–11–10 (Successful) |

It is important to reduce the number of unsafe nodes in a given HC in order to increase the number of possible feasible routes. It should be noted that in Fig. 27 there exists two disconnected sets of safe nodes: set 1: 0001,0011,0101,0111 and set 2: 1000,1010,1100,1110 and that each set forms a subcube of dimension 2. A faulty HC is called *fully unsafe* if every node in the hypercube is either faulty or unsafe. Routing difficulties are expected to be higher in a fully unsafe hypercube. If an $n$-cube having $n$ faulty nodes is fully unsafe, then any two of the n faulty nodes cannot be nearest neighbors. On the other hand, if the number of faulty nodes in an $n$-cube is less than $n$, then the $n$-cube cannot become fully unsafe. Given an $n$-cube, if a node A has $k$, $2 \leqslant k \leqslant n$, faulty neighbors, then every non-faulty node in the associated $k$-subcube, which includes node A and

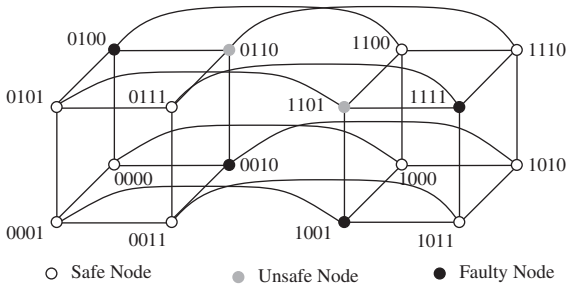


**Fig. 27.** Node status illustration.

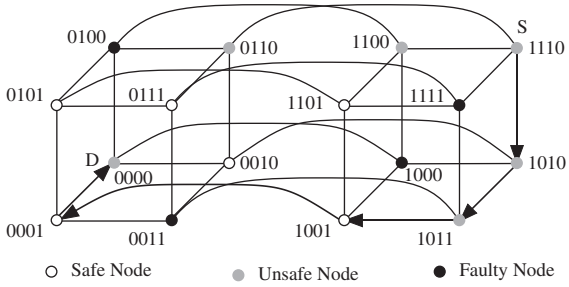


**Fig. 28.** Illustration of strongly and ordinary unsafe nodes.

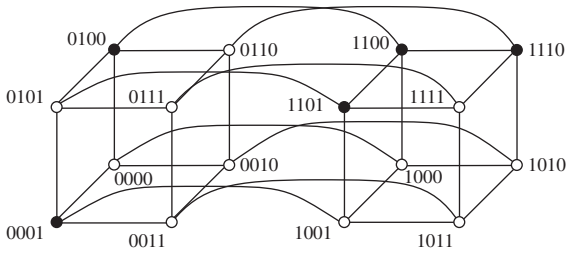


**Fig. 29.** Illustration of node reachability.

the k faulty neighbors, is unsafe. Unsafe nodes are further classified into ordinary and strong unsafe. An unsafe node is called *ordinary unsafe* unless all its nearest neighbors are either unsafe or faulty.

Consider, for example, the faulty 4-cube shown in Fig. 28. There are five unsafe nodes, i.e.. {0000,0110,1010,1100,1110}. Among these only node 1110 is characterized as strongly unsafe node while the remaining nodes are all ordinary unsafe nodes. The figure shows the routing according to the algorithm introduced in from the strongly unsafe node $S = 1110$ and the ordinary unsafe node $D = 0000$. Notice that routing is done over the feasible minimum (not shortest) length route. The algorithm reported in can also handle faulty links. It does that by considering any fault-free end node of a faulty link as faulty node while identifying unsafe nodes. However, this node same node is treated as an unsafe while identifying unsafe nodes. The definition of strongly unsafe node remains the same.

### *7.4. Reachability based routing techniques [42]*

This technique is an improvement over the node safety technique discussed above. The technique defines a given node as a *fully reachable* node with respect to (Hamming) distance $h$, if every non-faulty node which is apart from that node by Hamming distance $h$ is reachable via a path of length $h$ from the said node. Based on this definition, a non-faulty node in an $n$-cube is defined recursively as a safe node with respect to distance $h$ if it is adjacent to at least $n - h - 1$ safe nodes with respect to distance $h - 1$. Notice that every non-faulty node is a safe node with respect to (Hamming) distance 1 (the base case for recursion). Consider the 4-cube shown in Fig. 29. In this figure the following set of nodes are faulty 0001,0100,1100,1101,1110. In this figure node 1111 is fully reachable with respect to distance 2 since nodes 0011, 0101, 0110, 1001, and 1010 are reachable from node 1111 via path of length 2 (the Hamming distance from 1111). It should however be noted that node 1111 is not safe with respect to distance 2. Define $S_h$ as the set of safe nodes with respect to distance $h$ and $R_h$ as the set of fully reachable nodes with respect to distance $h$. Table 7 lists the set $S_h$ and the set $R_h$ for the 4-cube in Fig. 29. Notice that $S_h \subset R_h$. The set $S_h$ can be easily determined by exchanging information just between neighboring nodes.

**Table 7**
Sets $S_h$ and $R_h$ for the 4-cube in Fig. 21.

| Set of fully reachable nodes $R_h$ | Set of safe nodes with respect to distance $h$, $S_h$ |
|---|---|
| $R_1 = \{0,2,3,5,6,7,8,9,10,11,15\}$ | $S_1 = \{0,2,3,5,6,7,8,9,10,11,15\}$ |
| $R_2 = \{2,3,6,7,8,10,11,15\}$ | $S_2 = \{2,3,7,8,10,11\}$ |
| $R_3 = \{0,2,3,6,7,9,10,11,15\}$ | $S_3 = \{0,2,3,6,9,10,11,15\}$ |

A routing algorithm based on node reachability has been introduced in [42]. The performance of the proposed algorithm has been compared with that of the algorithm based on node safety. It has been shown that the algorithm in [42] can detect communication paths which do not include any faulty nodes.

## 8. Multicasting/broadcasting in faulty hypercube

In this section we discuss the issues related to multicasting/broadcasting in faulty hypercube [47,45,38,80]. In multicasting messages in a healthy hypercube from a given node, the source node would decide which of its neighboring node(s) should receive the message to be sent. Upon receiving the multicast message, each node checks its own address with the destination address contained in the message. If matched, then the node will route the message to its local processor and remove its own address from the list. If the list becomes empty, then the receiving node is a leaf node and no more forwarding is needed. Otherwise, the receiving node is a forward node and it has to determine its descending neighbors in the multicast tree [44]. At each forward node, the relative binary addresses of all destination nodes are computed and used to vote for the preferred (selected) dimension(s). The maximum number of ones in the corresponding dimension position determines the selected dimension(s). Two cases of multicasting in faulty hypercube are presented in [44]. In the first one, it was assumed that each fault-free node has at most one faulty neighboring node. In the second case each fault-free node is assumed to have two or more faulty nodes. Multicasting algorithms which guarantee multicast routing of messages in each of these two cases have been introduced in [44]. The introduced algorithms make use of $n$-bit status vector, $S$, where each bit represents the status of the corresponding neighboring node such that a 1 in a given bit position indicates that the corresponding neighboring node is fault-free while a 0 indicates that the corresponding neighboring node is faulty. Consider the case of multicasting message from source node #6 (00110) in a 5- cube. The destinations are: node #0 (00000), node #1 (00001), node #7 (00111), node # 18 (10010), node # 20 (10100), and node #29 (11101). Assume also that neighboring node #4 (00100) at dimension #1 is faulty. The status vector $S = (11101)$. The resulting relative addresses with respect to the source node (00110) are computed by performing XOR operation between the source node address and the destination nodes addresses assuming that dimension # 1 is faulty (has 0, see Table 8). From the table it is clear that bit positions 0, 2, and 4 have column $sum = 3$. Any one of these can be selected. Assume for simplicity that bit position 0 is selected. Since rows 2, 3, and 6 are having a 1 in bit position 0 are picked up to form a destination sub-list. The corresponding neighbor is $00110 \oplus 00001$ and the sub-list is $\{00111, 11101, 00001\}$. Now, row 2, 3, and 6 are reset to all zeros and the new reference array is constructed based on which the next sub-list is composed as $\{10010, 00000\}$. The process is repeated in order to arrive at the multicast tree shown in Fig. 30.

The problem of broadcasting a message on an $n$-cube in exactly $n$ time units in the presence of up to $(n-2)$ arbitrary node or edge faults assumes that the set of faults $F$ is known to all nodes in the hypercube. The solution intended is required to use the minimal number of messages possible, i.e. $2^n - 1$. The algorithm is based on constructing path collection (PC) for the given faulty hypercube. The PC of hypercube is defined as the collection of paths that connect a given node (called the origin node) to all other nodes in the hypercube. The collection of paths are then partitioned into a number of sub-collections, $P(x)$, one for each destination node, $x$. A *prefix-fan* path collection is defined as a path collection satisfying the following properties:

1. For every node, $x$, the sub-collection $P(x)$ consists of at least $(n-1)$ paths from the origin node to node $x$.
2. For every node, $x$ the paths in the sub-collection are mutually node-disjoint.
3. The paths in each sub-collection are ordered in an increasing path length.
4. The maximal length of a path in the collection is $n$.
5. For every path in the sub-collection the prefix of this path leading from the origin node to node $y$ belongs to $P(x)$.

**Table 8**
The Reference array at source node #6 (00110).

| Fault-free case | Dimension #1 is faulty | Row sum |
|---|---|---|
| $00110 \oplus 00000 = 00110$ | 00100 | 1 |
| $00110 \oplus 00001 = 00111$ | 00101 | 2 |
| $00110 \oplus 00111 = 00001$ | 00001 | 1 |
| $00110 \oplus 10010 = 10100$ | 10100 | 2 |
| $00110 \oplus 10100 = 10010$ | 10000 | 1 |
| $00110 \oplus 11101 = 11011$ | 11001 | 3 |
| Column sum | 31303 | |

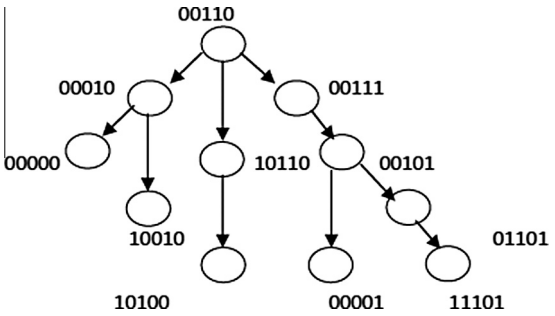


**Fig. 30.** Multicast tree of the hypercube with reference address Table 8.

A fault configuration $F$ is defined as the collection of up to $(n-2)$ faulty nodes and edges. A path is said to be injured by $F$ if it contains a faulty node or edge from $F$. The first path in the sub-collection that is not injured by $F$ is called the choice path of node $x$ with respect to $F$. The predecessor of $x$ on that path is called the choice informer of $x$. In performing broadcasting from origin node $O$ under a given fault configuration $F$, a given node $x$ will receive the message exactly once, from its choice informer node. According to the broadcast algorithm in [87] a given node y upon receiving a message for the first time it performs the following: it computes separately the choice path for each neighboring node, $x$, and determines the choice informer of $x$. In this choice informer ends being the node $y$ itself, then node $y$ forwards the message to $x$. As long as $|F| \leqslant (n-2)$, the broadcast algorithm will deliver the broadcast message to each non-faulty destination node, $x$, within time equal to the length of the choice path of $x$ with respect to $F$.

### 8.1. Distributed binary fault-tolerant multicast algorithm

In this section we propose a distributed binary fault-tolerant multicast algorithm based on multicast set (MS) construction. Multicast algorithms in the presence of faults has been studied by several authors [44,73,31]. The algorithm proposed here is executed by each forwarding node. The advantage of this algorithm is that is minimizes the multicast traffic and the multicast delay, expressed as the number of hops. We define the source node $u_s$ as any node that is required to multicast a message to a set of nodes. Assume a set of destination nodes $D = \{ d_0 \quad d_1 \quad \cdots \quad d_{k-1} \}$ is the set of address of $k$ multicast nodes. The source node groups the multicast nodes into two multicast sets based on the first address bit. Set $M_0$ contains all the multicast nodes whose address is $0xx\cdots x$. Set $M_1$ contains all the multicast nodes whose address is $1xx\cdots x$. The source node decides on the forwarding node for each set based on the minimum Hamming distance relative to the source node. This operation is repeated in the forwarding nodes but based on $n-1$ bits only and so on. Algorithm 2 shows the pseudo code for the multicast routing algorithm.

**Algorithm 2.** Pseudo code for new matrix multiplication algorithm to eliminate self-loops.

**Require** Input: $n, u_s$ and $D$
1: **while** $n \neq 0$ **do**
2: Use bit at position $i$ from the all the address bits of $D$
3: Construct $M_0$ and $M_1$
4: Construct $H_0$ and $H_1$, Hamming distance sets for nodes in $M_0$ and $M_1$, respectively.
5: Choose nearest forwarding nodes $f_0$ and $f_1$
6: Send $M_0$ to $f_0$ and end $M_1$ to $f_1$
7: **for** $d_i \in M_0$ **do**
8: **if** $d_i = f_0$ **then**
9: Remove bit at position $i$ from $M_0$
10: **end if**
11: **end for**
12: **for** $d_i \in M_1$ **do**
13: **if** $d_i = f_1$ **then**
14: Remove bit at position $i$ from $M_1$
15: **end if**
16: **end for**
17: Remove $n_i$ from all the address bits of $D$ // Strip off most significant bit of the node addresses
18: **end while**

Step 2: We use any bit from the address bits of the destination nodes in $D$ to start the algorithm. Typically, one would choose a value of $i = 0$.

Step 3: Based on the value of bit $n_i$ in each node in $D$, we divide the nodes into two multicast sets $M_0$ and $M_1$.

Step 4: Based on $H_0$ and $H_1$, we use the address of the nearest node in $M_0$ and $M_1$ to decide on the nearest forwarding nodes $f_0$ and $f_1$. If a potential forwarding node is faulty, then source node $u_s$ will check the address of next nearest node and choose the nearest forwarding node.

## 9. Concluding remarks

The availability of a wide range of techniques for achieving fault tolerance and reliability in multi-computer networks makes it difficult for designers of such networks to adopt the appropriate network configuration, given particular system performance constraints. This paper is an attempt to help network designers to overcome such difficulty. The paper provides designers of hypercube multi-computer interconnection networks with the central criteria needed to assess the reliability and fault tolerance of their designs. The paper shows designers the different methods that can be used to enhance the reliability and fault tolerance aspects of existing HCNs. The paper also presented a number of reliability computations and analysis of HCNs. A number of hardware and software based techniques for HC fault tolerance has been presented and analyzed. Simple examples were provided to illustrate the basic concepts and the algorithms used for achieving fault tolerance and the reliability computation in each case.

## References

[1] Sequent Computer Systems, Inc., Balance Technology Summary, Beaverton OR 97006-6063, 1984.
[2] Intel Scientific Computers, IPSC System Overview, Beaverton, Oregon, 1986.
[3] Intel ipsc/2, Intel Scientific Computers, 1988.
[4] Ncube Corporation, Ncube 2 Processor Manual, December 1990.
[5] M.H. Abd-El-Barr, H. Benten, M.A. Hai, Subcube reliability of a modular fault-tolerant hypercube architecture, Kuwait Journal of Science and Engineering 2 (1) (1996) 7–25.
[6] M.H. Abd-El-Barr, M. Nadeem, K. Al-Tawil, A heuristic-based routing algorithm for hypercube multicomputer networks, Cluster Computing Journal 4 (2001) 253–262.
[7] S. Abraham, K. Padmanabhan, Reliability of the hypercube, in: International Conference on Parallel Processing, October 1988, pp. 7–19.
[8] J. Al-Sadi, O.-K.M. Day K, Unsafety vectors: a new fault-tolerant routing for k-ary n-cubes, Microprocessors and Microsystems 25 (5) (2001) 239–246.
[9] J. Al-Sadi, K. Day, M. Ould-Khaoua, Probability-based fault-tolerant routing in hypercubes, The Computer Journal 44 (5) (2001) 368–373.
[10] J. Al-Sadi, K. Day, M. Ould-Khaoua, Unsafety vectors: a new fault-tolerant routing for the binary n-cube, Journal of Systems Architecture 47 (9) (2002) 783–793.
[11] N. Allahverdi, A fault-tolerant routing algorithm based on cube algebra for hypercube systems, Journal of Systems Architecture 46 (2) (2000) 201–205.
[12] N. Allahverdi, S. Kahramanli, K. Erciyes, A fault-tolerant routing algorithm based on cube algebra for hypercube systems, Journal of Systems Architecture 46 (2) (2000) 201–205.
[13] G. Angransson, R. Greenlaw, Graph Theory: Modeling, Applications, and Algorithms, Prentice-Hall, 2006.
[14] S. Balakrishnan, F. Ozguner, B. Izadi, Fault tolerance in hypercubes <http://www.engr.newpaltz.edu/bai/Research/nato.pdf>.
[15] P. Banerjee, M. Peercy, Design and evaluation of hardware strategies of reconfiguring hypercube and meshes under faults, IEEE Transactions on Computers 43 (7) (1994) 841–848.
[16] J. Brian, R. Vijayagopal, S. Latifi, Testing the reliability of a hypercube and folded hypercube in the presence of random and dynamic faults, in: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'03), vol. 3, Las Vegas, Nevada, 2003, pp. 1267–1271.
[17] J. Brian, R. Vijayagopal, S. Latifi, Testing the reliability of a hypercube and folded hypercube in the presence of random dynamic faults, in: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA03), 2003, pp. 1267–1271.
[18] H.-W. Chang, C.-W. Yu, A novel optimal routing and a fault- tolerant routing for the incrementally extensible hypercube network, WSEAS Transactions on Systems 4 (1) (2005) 27–31.
[19] H.-W. Chang, C.-W. Yu, A novel optimal routing and a fault-tolerant routing for incrementally extensible hypercube network, WSEAS Transactions on Systems 4 (1) (2005) 27–31.
[20] S. Chau, A. Liestman, A proposal for a fault-tolerant binary hypercube architecture, in: Proceedings International Symposium Fault-Tolerant Computing, June 1989, pp. 323–330.
[21] J. Chen, P. Gillard, C. Li, Performance evaluation of three network-on-chip (NoC) architectures (invited), in: 1st IEEE International Conference on Communications in China (ICCC), 2012, pp. 91–96.
[22] J. Chen, I. Kanj, G. Wang, Hypercube network fault tolerance: a probabilistic approach, in: Proceedings of the IEEE International Conference on Parallel Processing (ICPP02), 2002, pp. 65–72.
[23] J. Chen, L. Kanj, G. Wang, Hypercube network fault tolerance: a probabilistic approach, Journal of Interconnection Networks 6 (1) (2005) 17–34.
[24] M.-S. Chen, K. Shin, Depth-first approach for fault-tolerant routing in hypercube multi-computers, IEEE Transaction on Parallel and Distributed Systems 1 (2) (1990) 152–159.
[25] V. Chirivlla, R. Alcover, J. Duato, Accurate reliability and availability models for direct interconnection networks, in: Proceedings International Conference on Parallel Processing, September 2001, pp. 517–524.
[26] M. Coppola, B. Falsafi, J. Goodacre, G. Kornaros, From embedded multi-core SoCs to scale-out processors, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013, pp. 947–951.
[27] S. Dandamudi, D. Eager, Hierarchical interconnection networks for multi-computer systems, IEEE Transactions on Computers C-38 (3) (1990) 786–797.
[28] M. Dasgupta, S. Choudhury, N. Chaki, A secure hypercube based team multicast routing protocols (S-HTMRP), in: Proceedings of the 2009 IEEE International Advance Computing Conference (IACC 2009), March 2009, pp. 1265–1269.
[29] H. El-Miligi, M.W. El-Kharashi, F. Gebali, Reliability-aware design methodology for networks-on-chip applications, in: IEEE International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS09), Cairo, Egypt, April 2009, pp. 107–112.
[30] L.-J. Fan, C.-B. Yang, S.-H. Shiau, Routing algorithms on the bus-based hypercube network, IEEE Transactions on Parallel and Distributed Systems 16 (4) (2005) 335–348.
[31] C. Francalanci, P. Giacomazzi, A high performance deadlock-free multicast routing algorithm for k-ary n-cubes, IEEE Transactions on Computers 59 (2) (2010) 174–187.

[32] F. Gao, Z.-C. Li, Y.-H. Min, J. Wu, A fault-tolerant routing strategy based on extended safety vectors in hypercube multicomputers, Chinese Journal of Computers 23 (3) (2000) 248–254.
[33] F. Gebali, Analysis of Computer and Communication Networks, Springer, New York, 2008.
[34] F. Gebali, H. Elmiligi, M. El-Kharashi, Networks on Chips: Theory and Practice, CRC Press, Boca Raton, Florida, 2008.
[35] M. Gomez, N. Nordbotten, J. Flich, P. Lopez, A. Robles, J. Duato, T. Skeie, O. Lysne, A routing methodology for achieving fault tolerance in direct networks, IEEE Transactions on Computers 55 (4) (2006).
[36] P. Gregor, Recursive fault tolerance of fibonacci cube in hypercubes, Discrete Mathematics 306 (13) (2006) 1327–1341.
[37] S. Gunes, N. Yilmaz, N. Allahverdi, A multicast routing algorithm based on parallel branching method for faulty hypercubes, Journal of Electrical and Electronics 2 (2) (2002) 477–481.
[38] S. Gunes, N. Yilmaz, N. Allahverdi, A multicast routing algorithm based on parallel branching method for faulty hypercubes, Journal of Electrical & Electronics 2 (2) (2002) 477–481.
[39] S. Gunes, N. Yilmaz, E. Yaldiz, Fault-tolerant unicast routing algorithm based on parallel branching method for faulty hypercube, in: Proceedings of the 8th IEEE International Conference on Electronics, Circuits, and Systems, vol. 1, 2001, pp. 103–106.
[40] B. Izadi, F. Ozguner, Real-time fault-tolerant hypercube multicomputer, IEE Proceedings of the Computer Digital Techniques 149 (5) (2002) 197–202.
[41] G. Jan, Y. Hwang, M. Lin, D. Liang, Novel hierarchical interconnection networks for high performance multi-computer systems, Journal of Information Science and Engineering 20 (6) (2004) 1213–1229.
[42] K. Kaneko, H. Ito, Fault-tolerant routing algorithms for hypercube interconnection networks, IEICE Transactions on Information and Systems E84-D (1) (2001) 121–128.
[43] H.M.T. Kawasaki, T. Kawamura, A fault-tolerant routing algorithm for hypercube multi-computers, in: Proceedings IEEE International on Conference on Systems, Man and Cybernetics, 2002, pp. 629–633.
[44] Y. Lan, A. Esfahanian, L. Ni, Multicast in hypercube multiprocessors, Journal of Parallel and Distributed Computing 50 (8) (1990) 30–41.
[45] S. Latifi, M. Lee, Wormhole broadcast in hypercubes, Journal of Supercomputing 15 (2000) 183–192.
[46] Y. Li, S. Peng, W. Chu, An efficient algorithm for fault-tolerant routing based on adaptive binomial-tree techniques in hypercubes, in: Proceedings of the 5th International Conference on PDCAT, December 2004, pp. 196–201.
[47] F. Liu, Y. Song, Broadcast in the locally k-subcube-connected hypercube network with faulty tolerance, in: International Conference on Communications, Networking, and Mobile Computing (ICCNMC 2005), 2005, pp. 305–313.
[48] P. Loh, W. Hsu, Fault tolerance of complete Josephus cubes, Journal of Systems Architecture 49 (1–2) (2003) 1–21.
[49] P. Loh, W. Hsu, Y. Pan, The exchanged hypercube, IEEE Transactions on Parallel and Distributed Systems 16 (9) (2005) 866–874.
[50] Q. Mohammad, Adaptive fault-tolerant routing algorithm for tree-hypercube multicomputer, Journal of Computing Science 2 (2) (2006) 124–126.
[51] Q. Muhammad, Adaptive fault-tolerant routing algorithm for tree-hypercube multicomputer, Journal of Computer Science 2 (2) (2006) 124–126.
[52] E. Oh, Strong Fault Tolerance in Multicomputer Networks, Doctoral dissertation, Texas A&M University, August 2004.
[53] J. Ou, On maximal 3-restricted edge connectivity and reliability analysis of hypercube networks, Journal of Applied Mathematics and Computation 217 (6) (2010) 2602–2607.
[54] Y. Pan, Fault tolerance in the block shift network, IEEE Transactions on Reliability 50 (1) (2001) 88–90.
[55] M. Qatawneh, Adaptive fault-tolerant algorithm for tree-hypercube multicomputer, Journal of Computer Science 2 (2) (2006) 124–126.
[56] B. Qiao, F. Shi, W. Ji, THIN: a new hierarchical interconnection network-on-chip for SOC, in: 7th International Conference on Algorithms and Architectures for Parallel Processing, 2007, pp. 446–457.
[57] M. Ramras, Minimum cut-sets in hypercubes, Discrete Mathematics 289 (2004) 193–198.
[58] D. Rennels, On implementing fault tolerance in binary hypercubes, in: Proceedings IEEE Fault Tolerant Computing, vol. 4, 1988, pp. 463–468.
[59] T. Sasama, On fault tolerance of hypercubes using subcubes, International Journal of Reliability, Quality, and safety Engineering 9 (2) (2002) 151–161.
[60] Q. Shang, W. Zhang, X. Chen, X. Guo, Storage performance evaluation of media server based on multi-core network processors, in: IEEE Wireless Communications and Networking Conference Workshops (WCNCW), 2013, pp. 76–79.
[61] Y. Shi, Z. Hou, J. Song, Hierarchical interconnection networks with folded hypercube as basic cluster, in: 4th International Conference on High Performance Computing, vol. 1, 2000, pp. 134–137.
[62] J. Shih, Fault-tolerant routing in hypercube networks without virtual channels, IEE Proceedings of Computers and Digital Techniques 151 (5) (2004) 377–384.
[63] J.-D. Shih, Fault-tolerant wormhole routing for hypercube networks, Information Processing Letters 86 (2) (2003) 93–100.
[64] L. Song, F. BaoHua, D. Yong, Y. XiaoDong, Clustering multicast on hypercube networks, in: High Performance Computing Conference (HPCC), LNCS, vol. 4208, 2006, pp. 61–70.
[65] L. Song, X. Yang, A multicast path algorithm on hypercube interconnection networks, in: Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications, 2008, pp. 641–646.
[66] S. Sultan, R. Melhem, An efficient modular spare allocation scheme and its application to fault-tolerant binary hypercube, IEEE Transactions on Parallel and Distributed Systems 2 (1) (1991) 117–126.
[67] T. Takabatake, K. Kaneko, H. Ito, HCC: generalized hierarchical completely connected networks, IECE Transactions on Information & Systems 2 (E-83-d) (2000) 1216–1224.
[68] S. Tian, A fault-tolerant routing strategy based on extended optimal path matrices in hypercube multi-computers, Chinese Journal of Computers 25 (1) (2002) 87–92.
[69] S. Tian, Y. Lu, D. Zhang, Optimal fault-tolerant routing scheme for generalized hypercube, in: Proceedings 11th Pacific International Symposium on Dependable Computing (PRDC'05), 2005, pp. 91–100.
[70] C. Tripathy, Star-cube: A new fault-tolerant interconnection topology for massively parallel systems, Journal of Electronics and Telecom Engineering 84 (2004) 83–92.
[71] C. Tripathy, N. Adhikari, On a new multicomputer interconnection topology for massively parallel systems, International Journal of Distributed and Parallel Systems (IJDPS) 2 (4) (2011) 162–180.
[72] H. Wang, Z. Wu, A level-wise clustering algorithm for multicast on hypercube network, in: Proceedings of the 2010 Conference on Dependable Computing (CDC-2010), November 2010, pp. 263–266.
[73] H. Wang, Z. Wu, A level-wise clustering algorithm for multicast on hypercube network, in: World Automation Congress (WAC), 2012, pp. 263–266.
[74] H. Wang, Z. Wu, X. Yang, H. Liu, A novel ACO-based multicast path algorithm in hypercube networks, Journal of Intelligent Automation and Soft Computing 17 (5) (2011) 541–549.
[75] H. Wang, Z. Wu, X. Yang, H. Liu, Dong, A fault-tolerant routing model based on locally k-subcube connected hypercube networks, Journal of Computational Information Systems 8 (9) (2012) 3659–3669.
[76] L. Wang, Y.-P. Lin, Z.-P. Chen, X. Wen, Fault-tolerant routing based on safety path vectors in hypercube systems, Journal of Software 15 (5) (2004) 783–790.
[77] M. Wehner, L. Oliker, J. Shalf, A real cloud computer, IEEE Spectrum 46 (10) (2009) 24–29.
[78] D. West, Introduction to Graph Theory, second ed., Prentice-Hall, 2001.
[79] E. Witzke, S. Frese, Some limitations of adjacency matrices in computer network analysis, SIGCOMM Computer Communication, Review 18 (5) (1988) 43–47.
[80] J. Wu, E. Fernandez, Reliable broadcasting in faulty hypercube computers, in: Proceedings 11th Symposium on Reliable Distributed Systems, October 1992, pp. 122–129.

[81] R.-Y. Wu, J. Chang, G.-H. Chen, Node-disjoint paths in hierarchical hypercube networks, in: 20th International Parallel and Distributed Processing Symposium, April 2006, pp. 1–5.
[82] R.-Y. Wu, G.-H. Chen, Y. Kuo, G. Chang, Node-disjoint paths in hierarchical cubic networks, Information Sciences 177 (2007) 4200–4207.
[83] R.-Y. Wu, G.-H. Chen, Y.-L. Kuo, G. Chang, Node-disjoint paths in hierarchical hypercube networks, Journal of Information Sciences 177 (2007) 4200–4207.
[84] D. Xiang, A. Chen, Local-safety-information-based broadcasting in hypercube multicomputers with node and link failure, Journal of Interconnection Networks 2 (3) (2001) 365–378.
[85] Y. Xiang, S. Pasricha, Thermal-aware semi-dynamic power management for multicore systems with energy harvesting, in: 14th International Symposium on Quality Electronic Design (ISQED), 2013, pp. 619–626.
[86] L. Youyao, H. Jungang, D. Huimin, A hypercube-based scalable interconnection network for massively parallel computing, Journal of Computers 3 (10) (2008) 58–65.
[87] Q. Zhu, J.-M. Xu, X. Hou, M. Xu, On reliability of the folded hypercubes, Information Sciences 177 (8) (2007) 1782–1788.