International Conference on Parallel and Distributed Systems (ICPADS'2001), Jun. 26-29, Kyongju, Korea, 2001, pp.315-322

Fault-tolerant Routing and Disjoint Paths in Dual-cube: a New Interconnection Network

Yamin Li and Shietung Peng
Faculty of Computer and Information Sciences
Hosei University
Tokyo 184-8584 Japan
E-mail: {speng,yamin}@k.hosei.ac.jp

Abstract

In this paper, we introduce a new interconnection network, the dual-cube, its topological properties, and the routing/broadcasting algorithms in the dual-cube. The advanced subjects such as fault-tolerant routing and constructing multiple disjoint paths in dual-cubes are also included in this paper. The binary hypercube, or r-cube, can connect 2^r nodes. In contrast, a dual-cube with r links for each node, F_r, can connect 2^{2r-1} nodes while keeps most of topological properties of hypercubes. Fault-tolerant routing and constructing multiple disjoint paths in dual-cubes can be solved elegantly using a new structure, called extended cube. We show that for any two nonfaulty nodes s and t in F_r which contains up to $r-1$ faulty nodes, we can find a fault-free path s to t, of length at most $3d(s,t)$ in $O(r)$ optimal time, where $d(s,t)$ is the distance between s and t. We also show that, in a fault-free F_r, r disjoint paths s to t, of length at most $d(s,t)+6$ can be constructed in $O(r^2)$ optimal time.

keywords: Interconnection networks, hypercube, broadcasting, routing, fault-tolerance, disjoint paths

1 Introduction

The binary hypercube has been widely used as the interconnection network in a wide variety of parallel systems such as Intel iPSC, the nCUBE [5], the Connection Machine CM-2 [9], and SGI Origin 2000 [8]. A hypercube network of dimension *n* contains up to 2^n nodes and has *n* edges per node. If unique *n*-bit binary addresses are assigned to the nodes of hypercube, then an edge connects two nodes if and only if their binary addresses differ in a single bit. Because of its elegant topological properties and the ability to emulate a wide variety of other frequently used networks, the hypercube has been one of the most popular interconnection networks for parallel computer/communication systems.

However, the conventional hypercube has a major shortage, that is, the number of links per node in a system increases logarithmically as the total number of nodes in the system increases. Since the number of links is limited to eight per node with current IC technology, the total number of nodes in a hypercube parallel computer is restricted to several hundreds. Therefore, it is interesting to develop an interconnection network which keeps most of topological properties of hypercubes, and increase the total number of nodes in the system with a fixed amount of links per node.

Several variations of the hypercube have been proposed in the literature. Some variations focused on the reduction of diameter of the hypercube, such as folded hypercube [1] and crossed cube [2]; some focused on the reduction of the number of edges of the hypercube, such as cube-connected cycles [6] and reduced hypercube [10]; and some focused on the both, like hierarchical cubic network [3]. The variations of the hypercube that reduce the diameter, e.g. crossed cube and hierarchical cube, do not satisfy the following key property of the hypercube which is at the core of many algorithmic designs for efficient routing and communication: each node can be represented by a unique binary number such that two nodes are connected by an edge if and only if the two binary numbers differ in one bit only.

It is practically important to refine the hypercube networks such that the size of the network can be increased while the number of the links per node is limited by the technology. In this paper, a new interconnection network, the *dual-cube*, is introduced. The dual-cube shares the desired properties of the hypercube, and increases tremendously the total number of nodes in the system with limited links per node. The key property of the hypercube mentioned above is also true in the dual-cube. The size of the dual-cube can be as large as thirty-two thousands with up to eight links per node. If the number of links per node is *n*, the conventional hypercube can connect up to 2^n nodes; the hierarchical cubic network can connect 2^{2n-2} nodes; while

the dual-cube can connect 2^{2n-1} nodes.

A dual-cube uses binary hypercubes as basic components. Each such hypercube component is referred to as a *cluster*. Assume that the number of nodes in a cluster is 2^m. In a dual-cube, there are two *classes* with each class consisting of 2^m clusters. The total number of nodes is $2^m \times 2^m \times 2$, or 2^{2m+1}. Therefore, the node address has $2m+1$ bits. The leftmost bit is used to indicate the type of the class (class 0 and class 1). For the class 0, the rightmost m bits are used as the node ID within the cluster and the middle m bits are used as the cluster ID. For the class 1, the rightmost m bits are used as the cluster ID and the middle m bits are used as the node ID within the cluster. Each node in a cluster of class 0 has one and only one extra connection to a node in a cluster of class 1. These two node addresses differ only in the leftmost bit position.

In an r-connected dual-cube F_r, $r-1$ edges are used within cluster to construct an $(r-1)$-cube and a single edge is used to connect a node in a cluster of another class. There is no edge between the clusters of the same class. If two nodes are in one cluster, or in two clusters of distinct classes, the distance between the two nodes is equal to its Hamming distance, the number of bits where the two nodes have distinct values. Otherwise, it is equal to the Hamming distance plus two: one for entering a cluster of another class and one for leaving.

The rest of this paper is organized as follows. Section 2 introduces the topological properties of the dual-cube. Section 3 gives the routing and broadcasting algorithms in the dual-cube. Section 4 describes the structure of extended-cube, and applies it to the fault-tolerant routing in the dual-cube. Section 5 describes an algorithm for constructing r node-disjoint paths in F_r. Section 6 concludes the paper and presents some future research directions.

2 Topological Properties of the Dual-Cube

An r-connected dual-cube F_r is an undirected graph on the node set $\{0,1\}^{2r-1}$ such that there is an edge between two nodes $u = (u_{2r-1} \dots u_1)$ and $v = (v_{2r-1} \dots v_1)$ in F_r if and only if the following conditions are satisfied:

(1) u and v differ exactly in one bit position i.

(2) if $1 \le i \le r-1$ then $u_{2r-1} = v_{2r-1} = 0$.

(3) if $r \le i \le 2r-2$ then $u_{2r-1} = v_{2r-1} = 1$.

Intuitively, the set of the nodes u of form $(0u_{2r-2} \dots u_r * \dots *)$, where $*$ means "don't care", constitutes an $(r-1)$-dimensional hypercube. We call these hypercubes *clusters* of class 0. Similarly, the set of the nodes u of form $(1 * \dots * u_{r-1} \dots u_1)$ constitutes an $(r-1)$-dimensional hypercube, and we call them clusters of class 1. The edge connects two nodes in two clusters of distinct class is called *cross-edge*. In the other word, $\langle u, v \rangle$ is a cross-edge if and only if u and v differ at the leftmost bit position only.

We divide the binary representation of a node into three parts: Part I is the rightmost $r-1$ bits, part II is the next $r-1$ bits, and part III is the leftmost bit. For the nodes in a cluster of class 0 (class 1), part I (part II) is called *node ID* and part II (part I) is called *cluster ID*. Part III is a *class indicator*. The cluster contains node u is denoted as C_u. For any two nodes u and v in F_r, $C_u = C_v$ if and only if u and v are in the same cluster.

Fig. 1 depicts an F_3 network. The class indicator is shown at the top position in the node address. For the nodes of class 0 (class 1), the node ID (cluster ID) is shown at the bottom, and the cluster ID (node ID) is shown at the middle. Fig. 2 shows an F_4 network in which only those edges connecting to cluster 0 of class 1 are shown.

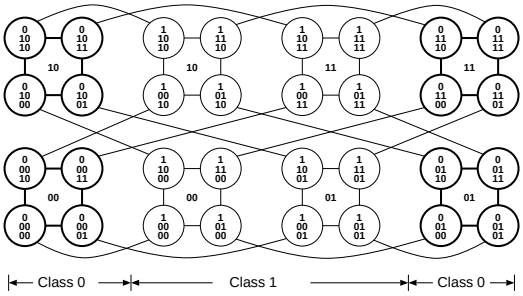

Figure 1. The dual-cubes F_3

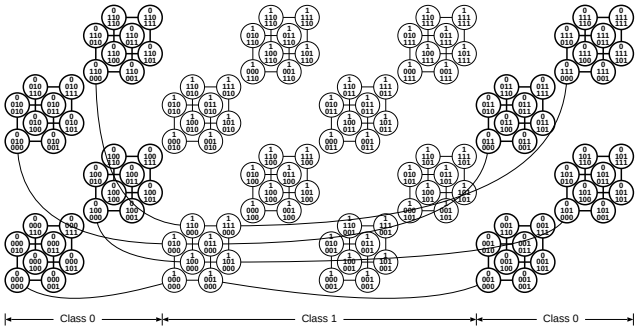

Figure 2. The dual-cubes F_4

Through this paper, we use the following terminologies. Let G be an undirected graph. A path from node s to node t in G is denoted by $s \to t$. The length of the path is the number of edges in the path. For any two nodes s and t in G, we denote $d(s,t)$ as the length of a shortest path connecting s and t. The diameter of G is defined as $d(G) = \max\{d(s,t) | s,t \in G\}$. The connectivity of G is defined to be the minimum number of nodes whose removal disconnects G or reduces it to a single node. G is k-connected if its connectivity is k.

A topology is evaluated in terms of a number of param-

Table 1. Hypercube v.s. Dual-cube

Network	Degree	Diam.	Cost	Avg. distance	Bisec. width	# of edges
Hypercube	n	n	n^2	$n/2$	$2^n/2$	$2^n n$
Dualcube	$(n+1)/2$	$n+1$	$(n+1)^2/2$	$n/2+1-1/2^{(n-1)/2}$	$2^n/4$	$2^n(n+1)/2$

eters such as degree, diameter, bisection width, cost (defined as the product of the degree and diameter), average distance for any two nodes, regularity, symmetry etc. We should consider the network cost as the main parameter for measuring and comparing of the different topologies. Other important measures for networks include the existence of simple routing and communication algorithms. The dual-cube networks have a binary presentation of nodes in which two nodes are connected by an edge if and only if they differ in one bit position, just as in hypercubes. This feature is the key for designing efficient routing and communication algorithms on dual-cubes. Another important feature of the dual-cubes is that, within the given bound on the number of links per node, say r, the network can have up to 2^{2r-1} nodes, more than the hypercubes or the hierarchical cubes can have with the same bound on the node degree.

Table 1 summarizes the degree, diameter, cost, average node distance, and bisection width of the hypercube and the dual-cube networks, assuming that the two networks have the same number of nodes which is 2^n, where n is an odd integer. The dual-cube shows a significant gain in the cost of the network. Since the number of nodes in F_r is 2^{2r-1}, we have $r = (n+1)/2$. The diameter and the average node distance of F_r will be shown in the next section. The bisection width of F_r is the number of edges between F_r^1 and F_r^2 where F_r^1 = half of the clusters of class 0 + half of the clusters of class 1, and $F_r^2 = F_r - F_r^1$. It is easy to see that the removal of $2^n/4$ edges will disconnect F_r^1 and F_r^2, and this number is the minimum for bisecting F_r.

Next, we consider the problem of recursive construction of F_r from F_{r-1}. Since the number of nodes in F_r is four times the number of nodes in F_{r-1}, we need four F_{r-1} in order to construct an F_r. First, we introduce the following notation: $S_a^{b_2 b_1}$ (a, b_1, and b_2 are single bits) is defined as the set of clusters of class a in the ith F_{r-1}, where $i = b_2 b_1$ $(0 \le i \le 3)$.

The node address in F_r is assigned as follows. Suppose the node address in F_{r-1} has the format of $(a x_{r-2} \ldots x_1 y_{r-2} \ldots y_1)$, where $x_{r-2} \ldots x_1$ is a cluster (node) ID and $y_{r-2} \ldots y_1$ is a node (cluster) ID for $a = 0$ ($a = 1$), then the node address in F_r has the format of $(a b_2 x_{r-2} \ldots x_1 b_1 y_{r-2} \ldots y_1)$.

The F_r is constructed as follows. 2^{r-1} clusters of class 0 in F_r are formed by pairwisely connecting nodes in S_0^{00} with nodes in S_0^{01}, and nodes in S_0^{10} with nodes in S_0^{11}; similarly, 2^{r-1} clusters of class 1 in F_r are formed by pairwisely connecting nodes in S_1^{00} with nodes in S_1^{10}, and nodes in S_1^{01} with nodes in S_1^{11}. By following this rule, it is easy to see that if two nodes are connected, their addresses differ in only one bit position: the addresses of two nodes of class 0 (1) differ in b_1 (b_2), and the addresses of two nodes of distinct classes differ in the class indicator bit.

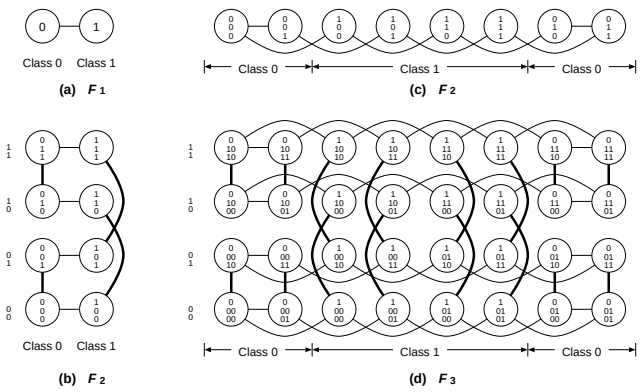

Figure 3. The construction of F_2 and F_3 from F_1 and F_2 respectively

Fig. 3 shows the recursive construction of F_2 and F_3 from F_1 and F_2, respectively. The recursive connections are marked with bold lines and curves. F_1 in Fig. 3(a) is a K_2 that has only two nodes, one for each class. Fig. 3(c) is the same as Fig. 3(b), and Fig. 3(d) is the same as Fig. 1. The $b_2 b_1$ is written in the left side in Fig. 3(b) and Fig. 3(d). It is easy to check that the cross-edges are correctly placed in this construction.

3 Routing and Broadcasting

The problem of finding a path from a source s to destination t and forwarding a message along the path is known as the routing problem. The broadcasting task is to send a message from a source to all other nodes. Routing and broadcasting are the basic communication problems for interconnection networks. In this section, we will describe routing and broadcasting algorithms for the dual-cube networks.

In order to describe the routing algorithm, we need some notation for the neighbor nodes of a node $s \in F_r$. The r neighbor nodes of s, $s^{(i)}$, $1 \le i \le r$, are denoted as follows. Assume s is of class 0, then $s^{(i)} =$

$(0a_{2r-2}\dots a_{i+1}\overline{a}_i a_{i-1}\dots a_1)$, where $1 \le i \le r-1$, and $s^{(r)} = (1a_{2r-2}\dots a_1)$. Assume s is of class 1, then $s^{(i)} = (1a_{2r-2}\dots a_{j+1}\overline{a}_j a_{j-1}\dots a_1)$, where $r \le j \le 2r-2$ and $i = j-(r-1)$, and $s^{(r)} = (0a_{2r-2}\dots a_1)$.

The routing algorithm is as follows. If $C_s = C_t$ then it is the routing in hypercubes. Assume that $C_s \neq C_t$. If C_s and C_t are of different classes, say C_s of class 0 and C_t of class 1, then we first routing s to s' in C_s such that the node ID of s' is the same as the cluster ID of t. Then, routing s' to $t' \in C_t$ through a cross-edge. Finally, t' can be routed to t in C_t. Next, assume that C_s and C_t are of the same class, say C_s and C_t are of class 0. We first routing s to s' in C_s such that the node IDs of s' and t are the same. Then, we route s' to s'' through a cross-edge (the cluster ID of s'' is equal to the node ID of t). Next, we route s'' to t' in $C_{s''}$ (of class 1) such that the node ID of t' is the same as the cluster ID of t. Finally, route t' to t in one step through a cross-edge.

From the above routing algorithm, assuming that s and t are different in $k \le 2r-1$ bits, the length of the routing path is k if $C_s = C_t$ or C_s and C_t are of different classes, otherwise, it is $k+2$. Notice that if C_s and C_t are of the same class (say, class 0)), then the path connecting s and t should pass through a cluster of class 1. This implies that the shortest path connecting s and t is of length at least $k+2$. Therefore, Theorem 1 is true.

Theorem 1 *Assume that nodes s and t in F_r differ in k bit-positions. The distance between s and t, $d(s,t) = k+2$ if s and t are in the different clusters of the same class; otherwise $d(s,t) = k$. The diameter of F_r, $d(F_r) = 2r$.*

In an n-dimensional hypercube, the average distance between any two nodes is $(\sum_{i=0}^{n} C(n,i) \times i)/2^n = n/2$. To calculate the average node distance of F_{2r-1}, assume that node s is in the cluster of class 0 in F_{2r-1}. Then the average distance between s and nodes u in the clusters of class 1, avg$(d(s,u))$, can be calculated as follows. First, we compute the average distance between s and the 2^{r-1} clusters of class 1. It is $(1/2^{r-1})\sum_{i=0}^{r-1} C(r-1,i) \times (i+1) = 1+(r-1)/2$. Therefore, we get avg$(d(s,u)) = 1+(r-1)/2+(r-1)/2 = r$. Similarly, the average distance between s and node v in the clusters of class 0, avg$(d(s,v))$ can be derived as follows: avg$(d(s,v)) = (2+(r-1)/2-2/2^{r-1})+(r-1)/2 = r+1-1/2^{r-2}$, where the first term, representing the average distance between s and the 2^{r-1} clusters of class 0, comes from the fact that the shortest paths from s to all clusters of class 0 except the cluster containing s, require two cross edges. From the above two formula, we conclude that the average node distance of F_{2r-1} is $r+1/2-1/2^{r-1}$.

The broadcasting process should satisfy some desirable properties: (1) a node should not send (receive) the message simultaneously to (from) more than one of its neighbors; (2) a node receives the message exactly once for the whole duration of the broadcasting process. We show an optimal broadcasting algorithm which completes broadcasting in optimal time (i.e., the diameter of the dual-cube) under the two restrictions listed above.

The algorithm for broadcasting from a source s works as follows. Assume that C_s is of class 0 (the case that C_s is of class 1 can be done similarly). The source s first send the message to its neighbor $s' = s^{(r)}$ through a cross-edge. Then, s and s' broadcasts simultaneously the message to all nodes in C_s and $C_{s'}$ using binomial trees of C_s and $C_{s'}$ with roots s and s', respectively. Next, every node $u \in C_s \setminus \{s\}$ and every node $u' \in C_{s'} \setminus \{s'\}$ sends the message to its neighbor $v = u^{(r)}$ and $v' = (u')^{(r)}$ through a cross-edge, respectively. Finally, every v and v' broadcasts the message to all nodes in C_v and $C_{v'}$. From the above algorithm, the broadcasting is completed in $1+(r-1)+1+(r-1) = 2r$ steps. Therefore, the following theorem is true.

Theorem 2 *Broadcasting in F_r can be done in $d(F_r)$ optimal time under the restricted one-port communication model.*

4 Extended Cube and Fault-Tolerant Routing

In this section, we first describe the structure of extended cube in the dual-cube. This structure is useful for designing algorithms in F_r, based on the hypercube algorithms. Then, we use this structure for the fault-tolerant routing in the dual-cube. The fault-tolerant routing in hypercubes has been studied extensively. It had been shown that in an n-dimensional hypercube with up to $n-1$ faulty nodes, given any two nonfaulty nodes s and t, a fault-free path of length at most $d(s,t)+2$ can be found in $O(n)$ time [4].

An r-dimensional extended cube in F_r has a similar structure as an r-dimensional hypercube H_r. The only difference is that, in the rth dimension, a pair of nodes in the two $(r-1)$-subcubes (or clusters), one in each cluster, are connected by a path, instead of an edges in H_r. The 2^{r-1} paths for connecting the 2^{r-1} pairs of nodes are disjoint in F_r. The construction of such an extended cube in F_r is described below. It is divided into two cases: Case 1 is for the clusters of the same class and Case 2 for the clusters or different classes. Let the two clusters to be joined to form an extended cube be C_s and C_t.

Case 1: C_s and C_t are of the same class. The extended r-cube is a subgraph of F_r which contains clusters C_s and C_t, and a set of clusters of the other class defined below. Since C_s and C_t are $(r-1)$-cubes, consider each pair of nodes $u \in C_s$ and $v \in C_t$, where u and v have the same node IDs. Nodes u and v can be connected via a unique cluster of the other class, denoted as $C_{(u,v)}$, where the cluster ID of $C_{(u,v)}$ is same as the node IDs of u and v. The extended r-cube

for s and t is a subgraph of F_r reduced by the nodes in $C_s \cup C_t \cup \{C_{(u,v)} | u \in C_s,\ v \in C_t$, and u, v have the same node ID $\}$. $u^{(r)}$ and $v^{(r)}$ are the nodes in $C_{(u,v)}$ that are connected to u and v by a cross edge, respectively. We have $d(u^{(r)}, v^{(r)}) = d(u,v) = d(C_s, C_t)$, u and v can be connected by a path of length $d(u,v)+2$, and the paths for all such node pairs in the two $(r-1)$-cubes are disjoint.

Fig. 4 shows the extended cube for the two clusters of class 0 in F_3, where the cluster ID of C_s is 00 and the cluster ID of C_t is 11. There are four pairs of nodes, their node IDs are 00, 01, 10, and 11. Each pair is connected via a cluster of class 1 with its cluster ID equal to the corresponding node IDs of the nodes of class 0.

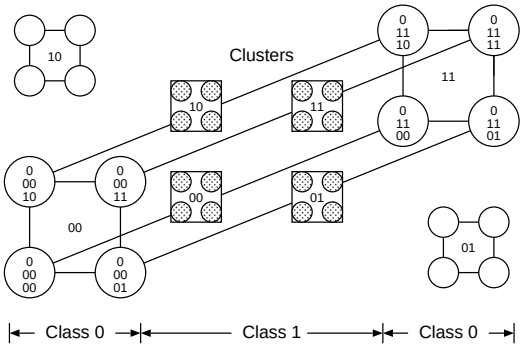

Figure 4. Extended cube example of Case 1

Case 2: C_s and C_t are of the distinct classes. The extended cube in this case is the subgraph of F_r reduced by the set of nodes in clusters C_s and C_t, and the subsets of the other $2(2^r-1)$ clusters used as intermediate nodes for connecting node pairs in C_s and C_t. There exists a node pair which is connected directly by a cross-edge. All other pairs can be connected via two clusters, one is of class 0 and the other is of class 1, and three cross-edges.

In order to form an r-dimensional cube out of the two $(r-1)$-cubes C_s and C_t, the node pairs (u,v) for $u \in C_s$ and $v \in C_t$ is assigned as follows. Suppose C_s is of class 0 and C_t is of class 1 (the case that C_s is of class 1 and C_t is of class 0 can be done similarly). Let the cluster IDs of C_s and C_t be $(s_{r-1} \ldots s_1)$ and $(t_{r-1} \ldots t_1)$ respectively. Every node $u \in C_s$ and $v \in C_t$ can be expressed as $(0 s_{r-1} \ldots s_1 a_{r-1} \ldots a_1)$ and $(1 b_{r-1} \ldots b_1 t_{r-1} \ldots t_1)$, respectively. u and v form a pair (connected by a hypercube edge) if and only if $a_i \oplus b_i = s_i \oplus t_i$, for all $i,\ 1 \le i \le r-1$. Notice that if $a_i = t_i$ and $b_i = s_i$, for all $i,\ 1 \le i \le r-1$, then $d(u,v) = 1$, and the hypercube edge is simply a cross-edge in F_r.

Fig. 5 shows the extended cube for the two clusters of distinct classes in F_3, where C_s is of class 0 with the cluster ID 00, and $C_t = 11$ is of class 1 with cluster ID 11. Nodes 00011 and 10011 are connected directly by a cross-edge. Other pairs are connected via two clusters and three cross-edges.

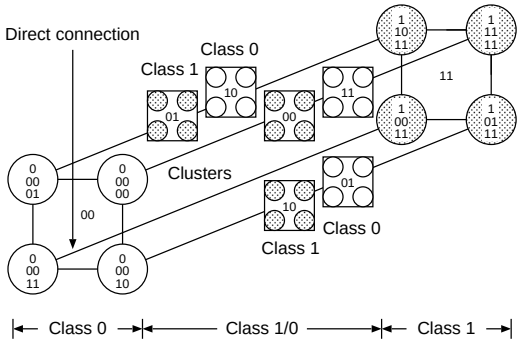

Figure 5. Extended cube example of Case 2

With the extended cube, the following fault-tolerant routing problem can be solved efficiently in F_r: given two nonfaulty nodes s and t, and up to $r-1$ faulty nodes in F_r, find a fault-free path connecting s and t. This problem is solved using the corresponding hypercube algorithm presented in [4]. The hypercube algorithm can be described briefly as follows. First, we partition H_n into two $(n-1)$-dimensional subcubes along a dimension k such that s and t are in the distinct subcubes. If the subcube contain t has less faulty nodes then we route s to another subcube by a fault-free path $s \to s^{(k)}$ or $s \to s^{(j,k)}$ for some $j,\ 1 \le j \ne k \le n$ (of length at most 2). Then the problem is reduced to the subproblem in the subcube containing t. Repeat this process (at most $\log n$ times) until the subcube contains no faulty node. The path $s \to s^{(j,k)}$ is a bad candidate if the jth bit of s is the same as that of t (a bad candidate will increase the length of the routing path). The algorithm guarantees that the bad candidate is used in at most one partition. The following lemma from [4] will be used in the analysis below.

Lemma 3 *Given two nonfaulty nodes s and t, and up to $n-1$ faulty nodes in H_n, we can find a fault-free path of length at most $d(s,t)+2$ in $O(n)$ time.*

Obviously, If $C_s \ne C_t$ then the fault-tolerant routing in F_r can be solved using extended cube and the above hypercube algorithm. The case $C_s = C_t$ is trivial and stated as follows. If C_s contains at most $r-2$ faulty nodes then we apply the hypercube algorithm. Otherwise, all faulty nodes are in C_s, and we can find a fault-free path $s^{(r)} \to t^{(r)}$ of length $d(s,t)+2$ which does not pass through C_s. The length of the fault-free path $s \to t$ in this case is $d(s,t)+4$.

Next, we analyze the length of the fault-free path and the running time of the algorithm in F_r. For Case 1, assume that k_1 (k_2) is the number of bits in the node ID (cluster ID) where s and t takes different values. We have $d(s,t) = k_1 + k_2 + 2$ in this case. The length of the fault-free path in the corresponding H_r is at most $k_1 + 1 + 2 = k_1 + 3$ from

the hypercube algorithm. Since the path connecting a pair of nodes in C_s and C_t is of length k_2+2, the fault-free path in the extended cube is at most $(k_1+3)-1+(k_2+2) = d(s,t)+2$. We say an rth dimensional edge (u,v) in the corresponding H_r faulty if and only if there is at least one faulty node in $C_{(u,v)}$ in the extended cube. We can identify all the faulty nodes and faulty edges (at most $r-1$ of them) in the corresponding H_r in $O(r)$ time. The algorithm takes $O(r)$ time to find a fault-free path in H_r. Finally, it takes $O(r)$ time to construct the subpath $u \rightarrow v$ in the extended cube. Therefore, the running time of the algorithm in the extended cube for Case 1 is $O(r)$.

For Case 2, since the path in the extended cube which corresponds to the hypercube edge (u,v) is of length $2d(s,t)$ in the worst case, the length of the fault-free path in the extended cube that corresponds to the fault-free path in the hypercube is $3d(s,t)$ in the worst case. For example, consider $s = 0000000$ and $t = 1000111$ in F_4. Assume that nodes $s^{(i)},\ i = 1,2,3$ are faulty. In the extended cube, s and node 1111111 will form a pair. The path used in the extended cube for this pair is of length 9. Then, since the subpath from node 1111111 to t is of length 3, the fault-free path $s \rightarrow t$ is of length 12, which is equal to $3d(s,t)$. The running time of the algorithm in Case 2 is still $O(r)$ since identifying all faulty edges at the rth dimension of the corresponding H_r takes $O(r)$ time as in Case 1.

We summarize the result of the above argument into the following theorem.

Theorem 4 *Given any two nonfaulty nodes s and t, and at most $r-1$ faulty nodes in F_r, a fault-free path $s \rightarrow t$ of length at most $3d(s,t)$ can be found in $O(r)$ optimal time.*

5 Multiple Disjoint Paths

In this section, we show how to construct r disjoint paths between any two nodes s and t in F_r. The disjoint paths here means node-disjoint. That is, any two communication paths $s \rightarrow t$ cannot share a node $v \neq s$ or t.

For n-dimensional hypercube H_n, the n disjoint paths can be constructed in $O(n^2)$ time. The n disjoint paths are constructed as follows. Without loss of generality, we assume that $s = s_n \ldots s_1$, where $s_i = 0$ for all $i,\ 1 \leq i \leq n$, and $t = t_n \ldots t_{k+1} t_k \ldots t_1$, where $t_i = 0$ for $i > k$, and $t_i = 1$ for $i \leq k$. Let $s^{(i)}$ be the node which differs with s in only the ith bit position. $s^{(i,j)}$ be the node which differs with s in the ith and the jth bit positions, and so on. The k disjoint paths of length $d(s,t)$ are $s \rightarrow s^{(i)} \rightarrow s^{(i,(i+0) \bmod k+1)} \rightarrow s^{(i,(i+0) \bmod k+1,(i+1) \bmod k+1)} \ldots \rightarrow t$, where $1 \leq i \leq k$. The rest $n-k$ disjoint paths are $s \rightarrow s^{(i)} \rightarrow t^{(i)} \rightarrow t$, where $k+1 \leq i \leq n$, $s^{(i)} \rightarrow t^{(i)}$ is a shortest path, and $d(s^{(i)},t^{(i)}) = d(s,t)$. The construction of the n disjoint paths in H_n will be used as a subroutine in the construction of multiple disjoint paths in F_r. The following lemma from [7] will be used for constructing disjoint paths in F_r.

Lemma 5 *For any two nodes s and t in H_n, we can find n disjoint paths of length at most $\min\{d(H_n)+1, d(s,t)+2\}$ in $O(n^2)$ time.*

For the dual-cube F_r, if $C_s = C_t$, the r disjoint paths can be constructed easily as follows. The first $r-1$ disjoint paths of length at most $d(s,t)+2$ can be constructed by Lemma 3. The last one is constructed through $s^{(r)}$ and $t^{(r)}$. Because we cannot use C_s (C_t) again for this path, we should establish a path between $s^{(r)}$ and $t^{(r)}$ through clusters other than C_s (C_t). It can be done by the follow steps. First, let $s^{(r)}$ and $t^{(r)}$ go one step to their neighbors $s_i = s^{(r,i)}$ and $t_i = t^{(r,i)}$ respectively, where $1 \leq i \leq r-1$. Then, let them reach $s_i^{(r)}$ and $t_i^{(r)}$ by using their cross edges. $s_i^{(r)}$ and $t_i^{(r)}$ fall in a same cluster and $d(s_i^{(r)},\ t_i^{(r)}) = d(s,t)$. The reason why $s_i^{(r)}$ and $t_i^{(r)}$ are of the same cluster is explained below. Because $C_s = C_t$, $s^{(r)}$ and $t^{(r)}$ have the same node IDs. Similarly, the node IDs of s_i and t_i are also the same. Therefore, $s_i^{(r)}$ and $t_i^{(r)}$ are located in the same cluster whose cluster ID is the same as the node IDs of s_i and t_i. Finally, a path between $s_i^{(r)}$ and $t_i^{(r)}$ in that cluster can be constructed easily. Therefore, for the case of $C_s = C_t$, we can find r disjoint paths of length at most $d(s,t)+6$.

Next, we assume that $C_s \neq C_t$. We first give the algorithms for finding r disjoint paths using extended cube. Then we propose a new approach for the case that C_s and C_t are of different classes to improve the upper bound of the length of the disjoint paths.

The construction of r disjoint paths between s and t for the case that C_s and C_t are of the same class is as follows. Assume $s = (0 s_{2r-2} \ldots s_0)$ and $t = (0 t_{2r-2} \ldots t_0)$. The case that s and t belong to the clusters of class 1 can be done similarly. The algorithm is described as follows. First, we form an H_r with its two $(r-1)$-subcubes H_{r-1}^0 and H_{r-1}^1 isomorphic to C_s and C_t, respectively, where H_{r-1}^0 (H_{r-1}^1) is an $(r-1)$-subcube of H_r containing all elements in H_r whose leftmost bit is 0 (1). The nodes correspond to s and t in H_r are nodes $s' = (0 s_{r-1} \ldots s_0)$ and $t' = (1 t_{r-1} \ldots t_0)$. Next, we use Lemma 3 to construct r disjoint paths of length at most $\min\{r+1, d(s',t')+2\}$ connecting s' and t' in H_r. Finally, for each path $s' \rightarrow t'$, we construct a corresponding path $s \rightarrow t$ in the extended cube by replacing the hypercube edge in the rth dimension by a path of length $d(s_{2r-2} \ldots s_r, t_{2r-2} \ldots t_r)+2$ as described in the previous paragraph. In this case, $d(s,t) \leq r-1$. This implies that the length of the paths is at most $r+5 \leq 2r+3 = d(F_r)+3$ for $r \geq 2$.

Example Let $s = 00000$ and $t = 01111$ in F_3. Then, $s' = 000$ and $t' = 111$. The three disjoint paths in H_3

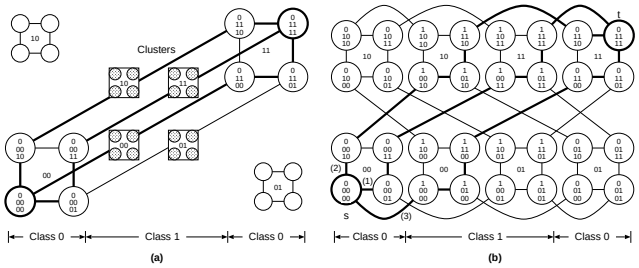

Figure 6. Extended cube(a) & disjoint paths(b)

are P_1 : 000 → 001 → 011 → 111, P_2 : 000 → 010 → 110 → 111, and P_3 : 000 → 100 → 101 → 111. The path corresponding to P_1 in F_3 is P_1' : 00000 → 00001 → 00011 → 10011 → 10111 → 11111 → 01111, where the hypercube edge 011 → 111 is replaced by the subpath, 00011 → 10011 → 10111 → 11111 → 01111, of length 4. P_2' and P_3' can be obtained similarly. Fig. 6(b) depicts the three disjoint paths between node 00000 and node 01111 constructed by the algorithm.

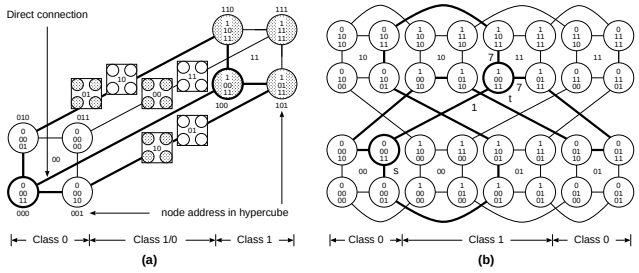

Figure 7. Extended cube(a) & disjoint paths(b)

The construction of r disjoint paths between s and t for the case that C_s and C_t are of distinct classes using extended cube is illustrated by the example. Fig. 7(a) shows the pair assignment for the case $s = 00011$ and $t = 10011$ in F_3. Fig. 7(b) depicts an example that establishes three disjoint paths between node $s = 00011$ and node $t = 10011$ in F_3 using the extended cube. In this example, $C_s = 00$ and $C_t = 11$, and the unique cross-edge between C_s and C_t is $\langle s,t\rangle$. The three disjoint paths in the corresponding H_3 are shown in Fig. 7(a). Then, two hypercube edges (wide lines in Fig. 7(a)) are replaced by two paths of length 5 (see Fig. 7(b)) to obtain the three disjoint paths in F_3. All clusters except two clusters (one of class 0 with cluster ID 11 and the other of class 1 with cluster ID 00) are used in the connections. The lengths of the three disjoint paths are 1, 7, and 7. As stated before, the length of the longest path connecting some node pairs can be $3d(s,t)$ in the worst case.

Next, we propose a method that establishes disjoint paths with shorter longest path. If s and t are of different class, there must be a cross-edge connecting C_s and C_t. Let $s = (0s_{r-1}\dots s_1a_{r-1}\dots a_1)$, $t = (1b_{r-1}\dots b_1t_{r-1}\dots t_1)$, and the cross-edge connects nodes $w_0 = (0s_{r-1}\dots s_1t_{r-1}\dots t_1)$, and $w_1 = (1s_{r-1}\dots s_1t_{r-1}\dots t_1)$.

Assume $s \neq w_0$ and $t \neq w_1$. We first route nodes s and t to nodes w_1 and w_0 in clusters C_s and C_t, respectively, by shortest paths. Let the shortest paths be $s \to s^{(j)} \to w_0$ and $t \to t^{(k)} \to w_1$, respectively. Then we get the first path through cross-edge $\langle w_1, w_0\rangle$, i.e., $s \to s^{(j)} \to w_0 \to w_1 \to t^{(k)} \to t$. If $s = w_0$ ($t = w_1$) then the path is $s = w_0 \to w_1 \to t^{(k)} \to t$ ($s \to s^{(j)} \to w_0 \to w_1 = t$), and $j = r$ ($k = r$). For the other $r-1$ neighbors of s and t, ($s^{(i)}, i = 1,\dots j-1, j+1,\dots r$ and $t^{(i)}, i = 1,\dots k-1, k+1,\dots r$), we can assign any two nodes $s^{(i)}$ and $t^{(i')}$ as a pair. We route these pairs through $s' = s^{(i,r)}$ and $t' = t^{(i',r)}$, and then connect s' and t' by a shortest path in $C_{s'} \cup C_{t'}$. It can be seen easily that the r paths constructed above are disjointed. The routing algorithm is formally described below. Instead of w_0 and w_1, we use w_0^{st} and w_1^{st} notation in the algorithm.

Algorithm 1 (DISJOINT_PATHS(s, t, F_r))

1. **Input:** a dual-cube F_r and nodes s and t in F_r
2. **Output:** r disjoint paths connecting s and t
3. **begin**
   /* Let the cross-edge between C_s and C_t be $\langle w_0^{st}, w_1^{st}\rangle$, $w_0^{st} \in C_s$ and $w_1^{st} \in C_t$. */
4. **if** $s \neq w_0^{st}$
5. **then** find a shortest path in C_s, P_0^s: $s \to s^{(j)} \to w_0^{st}$
6. **else** $P_0^s = \emptyset$ and $j = r$;
7. **if** $t \neq w_1^{st}$
8. **then** find a shortest path in C_t, P_1^t: $t \to s^{(k)} \to w_1^{st}$
9. **else** $P_1^t = \emptyset$ and $k = r$;
10. construct path $P_0^s \cup \{w_0^{st} \to w_1^{st}\} \cup P_1^t$;
11. **for** $i = 1$ **to** $r-1$ **do**
12. $p = ((j+i-1) \bmod r) + 1$;
13. $q = ((k+i-1) \bmod r) + 1$;
14. **if** $p \neq r$
15. **then** find P_0^p : $s \to s^{(p)} \to s^{(p,r)} = s'$
16. **else** find P_0^p : $s \to s^{(p)} = s'$;
17. **if** $q \neq r$
18. **then** find P_1^q : $t \to t^{(q)} \to t^{(q,r)} = t'$
19. **else** find P_1^q : $t \to t^{(q)} = t'$;
    /* Let the cross-edge between $C_{s'}$ and $C_{t'}$ be $\langle w_0^{pq}, w_1^{pq}\rangle$, $w_1^{pq} \in C_{s'}$ and $w_0^{pq} \in C_{t'}$. */
20. construct path $P_0^p \cup \{s' \to w_1^{pq} \to w_0^{pq} \to t'\} \cup P_1^q$;
    /* subpaths $s' \to w_1^{pq}$ and $t' \to w_0^{pq}$ are shortest paths in $C_{s'}$ and $C_{t'}$, respectively. */
21. **endfor**
22. **end.**

Example Consider $s = 0000001$ and $t = 1110111$ in F_4. We have $w_0 = 0000111$ and $w_1 = 1000111$. The algorithm finds shortest paths $s \to 0000101 \to w_0$ and $t \to 1100111 \to w_1$ in C_s and C_t, respectively. From this, we have $j = 3$ and

Table 2. Two examples of disjoint paths

$s=0000001 \rightarrow 1110111=t$				$s=0000111 \rightarrow 1000111=t$			
$s^{(3)} \rightarrow t^{(2)}$	$s^{(4)} \rightarrow t^{(3)}$	$s^{(1)} \rightarrow t^{(4)}$	$s^{(2)} \rightarrow t^{(1)}$	$s^{(4)} \rightarrow t^{(4)}$	$s^{(1)} \rightarrow t^{(1)}$	$s^{(2)} \rightarrow t^{(2)}$	$s^{(3)} \rightarrow t^{(3)}$
0000001 s	0000001 s	0000001 s	0000001 s	0000111 s	0000111 s	0000111 s	0000111 s
0000101 $s^{(3)}$	1000001 $s^{(4)}$	0000000 $s^{(1)}$	0000011 $s^{(2)}$	1000111 t	0000110 $s^{(1)}$	0000101 $s^{(2)}$	0000011 $s^{(3)}$
0000111 w_0	1010001 w_1	1000000	1000011		1000110	1000101	1000011
1000111 w_1	0010001 w_0	1100000	1100011		1001110 w_1	1010101 w_1	1100011 w_1
1100111 $t^{(2)}$	0010101	1110000 w_1	1110011		0001110 w_0	0010101 w_0	0100011 w_0
1110111 t	0010111	0110000 w_0	1111011 w_1		0001111	0010111	0100111
	1010111 $t^{(3)}$	0110100	0111011 w_0		1001111 $t^{(1)}$	1010111 $t^{(2)}$	1100111 $t^{(3)}$
	1110111 t	0110110	0111111		1000111 t	1000111 t	1000111 t
		0110111 $t^{(4)}$	1111111 $t^{(1)}$				
		1110111 t	1110111 t				

$k=2$. The four disjoint paths constructed by the algorithm are given in Tab. 2 (left part). The longest path is of length 9 which is $d(s,t)+4$.

Example Consider $s=0000111$ and $t=1000111$ in F_4. Since $s=w_0$ and $t=w_1$, we have $j=k=p$. The four disjoint paths constructed by the algorithm are given in Tab. 2 (right part). The longest path is of length 7 which is $d(s,t)+6$.

It is easy to check from the algorithm DISJOINT_PATHS that the following statements are true. (1) The r paths are disjoint; (2) The subpath $s^{(p,r)} \rightarrow t^{(q,r)}$ is a shortest path; (3) The time for constructing the r paths is $O(r^2)$. From (2), the length of the longest path at most $d(s,t)+6$. Since s and t are in the clusters of the distinct classes, $d(s,t) \leq d(F_r)-1$. If $d(s,t)=d(F_r)-1$ then $d(s^{(p)},t^{(q)}) \leq d(s,t)$. This implies that the length of paths is at most $d(s,t)+4=d(F_r)+3$. The worst case occurs when $d(s,t)=d(F_r)-2$. In this case, the length of paths is at most $d(s,t)+6=d(F_r)+4$.

From the discussion above, we conclude that we can find r disjoint paths of length at most $\min\{d(F_r)+4, d(s,t)+6\}$ in all three cases. Since the time for construct each of r paths is $O(r)$, the total time complexity for finding r disjoint paths is $O(r^2)$. This result is stated in Theorem 6.

Theorem 6 *For any two nodes s and t in F_r, we can find r disjoint paths of length at most $d(s,t)+6$ in $O(r^2)$ time.*

6 Conclusion

In this paper, we first proposed a new interconnection network (dual-cube), investigated its topological properties, and gave efficient routing and broadcasting algorithms. Then, we proposed an extended cube structure in dual-cubes which then is used to handle fault-tolerant routing and to construct multiple disjoint paths in dual-cubes. Finally, we showed another algorithm to construct disjoint paths of shorter length in dual-cubes. Since dual-cubes can accommodate much more nodes into the system than hypercubes while the number of links per node is fixed, it is a network of great potential. The following issues concerning dual-cubes are worth of further research. (1) Design algorithms for efficient collective communication (multicast, total exchange, etc.) in dual-cubes. (2) Design algorithms for effective embedding of the frequently used topologies (meshes, hypercubes) into dual-cubes.

References

[1] A. E. Amawy and S. Latifi. Properties and performance of folded hypercubes. *IEEE Transactions on Parallel and Distributed Systems*, 2(1):31–42, 1991.

[2] K. Efe. The crossed cube architecture for parallel computation. *IEEE Transactions on Parallel and Distributed Systems*, 3(5):513–524, Sep. 1992.

[3] K. Ghose and K. R. Desai. Hierarchical cubic networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(4):427–435, April 1995.

[4] Q.-P. Gu and S. Peng. Optimal algorithms for node-to-node fault tolerant routing in hypercubes. *The Computer Journal*, 39(7):626–629, 1996.

[5] J. P. Hayes and T. N. Mudge. Hypercube supercomputers. *Proc. IEEE*, 17(12):1829–1841, Dec. 1989.

[6] F. P. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Commun. ACM*, 24:300–309, May 1981.

[7] Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE Transactions on Computers*, 37(7):867–872, July 1988.

[8] SGI. *Origin2000 Rackmount Owner's Guide, 007-3456-003*. http://techpubs.sgi.com/, 1997.

[9] L. W. Tucker and G. G. Robertson. Architecture and applications of the connection machine. *IEEE Computer*, 21:26–38, August 1988.

[10] S. G. Ziavras. Rh: a versatile family of reduced hypercube interconnection networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(11):1210–1220, November 1994.