ACSAC02, Australian Computer Science Communications, Vol.24, No.4, 2001, pp.29–36

Metacube – A New Interconnection Network for Large Scale Parallel Systems

Yamin Li[1] **Shietung Peng**[1] **Wanming Chu**[2]

[1]Department of Computer Science
Hosei University
Tokyo 184-8584 Japan
Email: {yamin,speng}@k.hosei.ac.jp

[2]Department of Computer Hardware
University of Aizu
Aizu-Wakamatsu 965-8580 Japan
Email: w-chu@u-aizu.ac.jp

Abstract

The hypercube has been widely used as the interconnection network for parallel computers. However, in hypercubes, the number of communication links for each node is a logarithmic function of the total number of nodes. Therefore, the hypercube is not a good candidate for an interconnection network for a very large parallel computer that might contain hundreds of thousands of nodes due to IC technology and port number limitations. This paper introduces a new interconnection network for very large parallel computers called *metacube* (MC). An MC network has a 2-level cube structure. An MC(k, m) network can connect 2^{m2^k+k} nodes with $m+k$ links per node, where k is the dimension of the high-level cubes (classes) and m is the dimension of the low-level cubes (clusters). An MC network is a symmetric network with short diameter, easy and efficient routing and broadcasting similar to that of the hypercube. However, an MC network can connect millions of nodes with up to 6 links per node. An MC(2,3) with 5 links per node has 16,384 nodes and an MC(3,3) with 6 links per node has 134,217,728 nodes. We describe the MC network's structure, topological properties and routing and broadcasting algorithms.

Keywords: Interconnection networks, hypercube, routing, broadcasting

1 Introduction

The hypercube has been widely used as the interconnection network in a wide variety of parallel systems such as Intel iPSC (Vanvoorst, Seidel & Barscz 1994), the nCUBE (Hayes & Mudge 1989), the Connection Machine CM-2 (Tucker & Robertson 1988) and SGI Origin 2000 (SGI 1998). An n-dimensional hypercube (n-cube) contains 2^n nodes and has n edges per node. If unique n-bit binary addresses are assigned to the nodes of an n-cube, 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 systems.

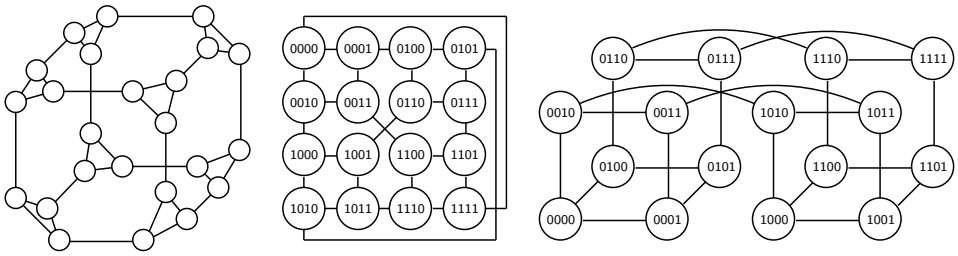
However, the number of edges per node increases logarithmically as the total number of nodes in the hypercube increases. Currently, the practical number of links is limited to about eight per node (SGI 1998). If one node has one processor, the total number of processors in a parallel system with an n-cube connection is restricted to several hundreds. We have found an interconnection network which will link millions of nodes with only a small number of links per node while retaining the hypercube's topological properties.

Several variations of the hypercube have been proposed in the literature. Some variations focused on reduction of the hypercube diameter, for example the folded hypercube (Amawy & Latifi 1991) and crossed cube (Efe 1992); some focused on reduction of the number of edges of the hypercube, for example cube-connected cycles (Preparata & Vuillemin 1981) and reduced hypercube (Ziavras 1994); and some focused on both, as in the hierarchical cubic network (Ghose & Desai 1995). One major property of the hypercube is: there exists an edge between two nodes only if their binary addresses differ in a single bit. This property is at the core of many algorithmic designs for efficient routing and communication in hypercubes. In this paper, we refer to it as the key property. Generally, variations of the hypercube that reduce the diameter, e.g. crossed cube and hierarchical cube, will not satisfy this key property.

Our goal is to accommodate as many nodes as possible with a fixed number of links per node, and at the same time, keep the main structures and interesting properties of the hypercube such as the key property, low diameter and efficient routing etc. We first review cube-connected cycles, hierarchical cubic network, reduced hypercube and Origin2000 that contributed something to achieving this goal.

A cube-connected cycles (Preparata & Vuillemin 1981) CCC(n) network is constructed from an n-dimensional hypercube by replacing each node in a hypercube with a ring containing n nodes. Each node in a ring then connects to a distinct node in one of the n dimensions. Figure 1(a) shows a CCC(3). The advantage of the cube-connected cycles is that the node's degree is 3, independent of the value of n.

The hierarchical cubic network (Ghose & Desai 1995) HCN(n, n) has 2^n clusters, where each cluster is an n-cube. Each node in the HCN(n, n) has $n+1$ links. Of these, n links are used inside the cluster. The additional link is used to connect nodes among clusters. Figure 1(b) shows an HCN(2,2). The advantages of the hierarchical cubic network are that the number of links required is reduced to approximately half as many links per node and the diameter is reduced to about three-fourths that of a corresponding hypercube.



(a) CCC(3) (b) HCN(2,2) (c) RH(2,1)

Figure 1: Three hypercube variations (# edges = 3)

A reduced hypercube (Ziavras 1994) RH(k, m) is obtained from an n-dimensional hypercube by using the following rules to reduce node edges in a hypercube. Let $k + 2^m = n$. There are 2^{2^m} clusters and each cluster is a k-dimensional hypercube. For the higher $n - k = 2^m$ dimensions, a node has only one direct connection to another node. This dimension of the connection is decided by the leftmost m bits in the k-bit field, i.e. the $(2^i + k)$th dimension, where i is the value of the m-bit binary number. Figure 1(c) shows an RH(2,1). Each node in an RH(k, m) has $k + 1$ edges.

An Origin2000 (SGI 1998) is constructed with a hypercube or the folded cube when the number of processors is not too large (less than or equal to 64 processors for instance). Origin2000 reduces the number of links required when the system size increases by introducing CrayRouters to connect hypercubes (called fat-hypercubes). A CrayRouter is the high level router that does not connect processors directly. The processors are attached to regular routers within the hypercubes. Each regular router has six links (five local links and a CrayLink). Two local links connect two nodes, where each node contains two processors. Therefore, a router has four processors. Three local links build the hypercube and a CrayLink connects to a CrayRouter. When the number of processors increases, the number of CrayRouters and the dimension of each CrayRouter will also increase. The largest Origin2000 system can connect up to $2^5 \times 2^3 \times 4$, or 1024, processors. It will use eight 5-dimensional CrayRouters and 256 regular routers.

In this paper, we propose a new interconnection network, called *metacube*, or MC network. The MC network shares many desired properties of the hypercube (e.g., the key property of the hypercube, small diameter etc.), and can be used as an interconnection network for a parallel computer system of almost unlimited size with just a small number of links per node. For example, an MC(2,3) with 5 links per node has 16384 nodes, and an MC(3,3) with 6 links per node has $2^{27} = 134,217,728$ nodes. The number of nodes connected by the MC is much larger than that of the HCN or the RH with the same number of links per node. The CCC uses only 3 links per node. However, because of its ring structure, the diameter or the length of the routing path in CCC is about twice of that of the hypercube. Compared with the CCC, the MC has shorter diameter, length of the routing path, and the broadcasting time.

The rest of this paper is organized as follows. Section 2 introduces the MC network. Section 3 describes its topological properties. Section 4 gives the routing and broadcasting algorithms. Sections 5 concludes the paper and presents some future research directions.

2 Metacube Interconnection Networks

This section formally introduces the MC network. The MC network is motivated by the dual-cube network (Li & Peng 2000) (Li & Peng 2001) that mitigates the port limitation problem in the hypercube network so that the number of nodes in the network is much larger than that of the hypercube with a fixed number of links per node. The MC network includes the dual-cube as a special case. The MC network has a 2-level cube structure: a high-level cube represented by the leftmost k bits of the binary address of the node which contains $m2^k + k$ bits (these k bits serve as a class indicator); and low-level cubes, called clusters that forms the basic component in the network, represented by the m bits of the remain $m2^k$ bits, which occupy the different portion of the $m2^k$ bits for the different classes.

More specifically, there are two parameters in an MC network, k and m. An MC(k, m) contains 2^k *classes*. Each class contains $2^{m(2^k-1)}$ *clusters*, and each cluster contains 2^m *nodes*. Therefore, an MC(k, m) uses $m2^k + k$ binary bits to identify a node and the total number of nodes is 2^n where $n = m2^k + k$. The value of k affects strongly the growth rate of the size of the network. An MC$(1, m)$ containing 2^{2m+1}nodes is called a *dual-cube*. Similarly, an MC$(2, m)$, an MC$(3, m)$, and an MC$(4, m)$ containing 2^{4m+2} nodes, 2^{8m+3} nodes, and 2^{16m+4} nodes, are called *quad-cube*, *oct-cube*, and *hex-cube*, respectively. Since an MC$(3, 3)$ contains 2^{27} nodes, the oct-cube is sufficient to construct practically parallel computers of very large size. The hex-cube is of theoretical interest only. Note that an MC$(0, m)$ is a hypercube.

The $(m2^k + k)$-bit node address in MC(k, m) is divided into three parts: a k-bit classID, an $m(2^k-1)$-bit clusterID, and an m-bit nodeID. The leftmost k-bit binary number defines a class of clusters (classID). There are 2^k classes. Of each class, there are $2^{m(2^k-1)}$ clusters, an $m(2^k - 1)$-bit binary number identifies a cluster of the class (clusterID). An m-bit binary number, located in a special portion of the $m2^k$-bit (will be explained in the next paragraph) identifies a node within the cluster (nodeID).

In the following discussion, we use $(c, m_{h-1}, \ldots, m_1, m_0)$ to denote the node address where $h = 2^k$. c is the k-bit classID, m_c is the m-bit nodeID, and $(m_{h-1}, \ldots, m_{c+1}, m_{c-1}, \ldots, m_0)$ is the $m(2^k - 1)$-bit clusterID. Figure 2 shows the format of the address.

An MC(k, m) is constructed in the following manner. Within a cluster, the m-bit nodeID forms an m-cube with m links or *cube-edges*. For a cluster of class i $(i = 0, 1, \ldots, h - 1)$, the nodeID within the cluster is m_i, while $(m_{h-1}, \ldots, m_{i+1}, m_{i-1}, \ldots, m_0)$

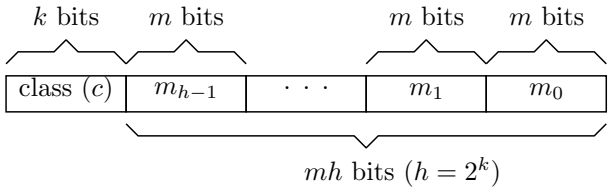


Figure 2: Format of a node address for an MC(k, m)

is the clusterID. The links that connect nodes in different clusters are defined as follows. For any two nodes whose addresses differ only in a bit position in the class field, there is a link, a *cross-edge*, connecting these two nodes. This is, the k-bit defines a high-level k-cube which connects those nodes whose addresses except class field are the same. A cluster then can be considered as a low-level m-cube.

The addresses of two nodes connected by a cross-edge differ only in one bit within the k-bit class field and there is no direct connection among the clusters of the same class. Therefore, a node in an MC(k, m) has $m + k$ links: m links construct an m-cube cluster and k links construct a k-cube. For example, the neighbors in the cluster of the node with address (01,111,101,110,000) in an MC$(2, 3)$ have addresses (01,111,101,111,000), (01,111,101,100,000), and (01,111,101,010,000). The underlined bits are those that differ from the corresponding bits in the address of the referenced node. The two neighbors in the high-level cube are (00,111,101,110,000) and (11,111,101,110,000).

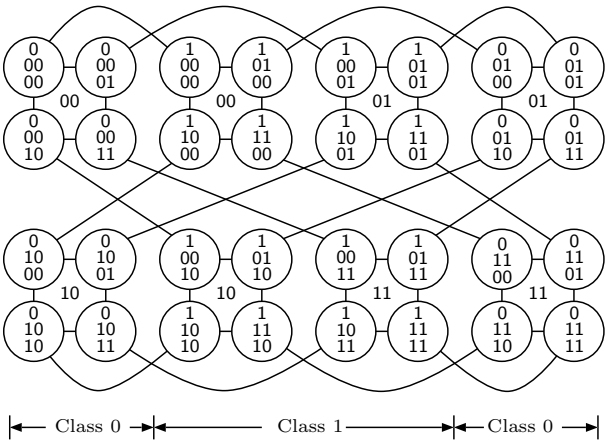


Figure 3: A metacube MC(1,2)

Figure 3 shows the structure of an MC$(1, 2)$, where the cluster is a 2-cube and there are two classes. Each node has a cross-edge attached to a node of the other class. The binary number shown in the center of a cluster is the clusterID. Figure 4 shows the structure of an MC$(2, 2)$, where the clusters in the same square are of the same class. In Figure 4, there are $2^{2(2^2-1)} = 64$ clusters in each square, and each cluster is a 2-cube. The figure shows only 4 high-level cubes which contain 4 nodes in the cluster 0 of the class 0, respectively. Figure 5 shows all the clusters of the MC$(2, 2)$, where the node addresses, except classID, are denoted in decimal numbers. Note that no high-level cube is shown in the figure.

The ratio of the total number of links in a hypercube to the total number of links in an MC network is equal to $n/(m+k)$, where $n = m2^k + k$. For example, for $k = 2$ and $m = 3$ ($n = 14$), each of the two networks contains 16384 nodes: a hypercube contains $2^{14} \times 14/2 = 114688$ links, whereas an MC network contains $2^{14} \times (3+2)/2 = 40960$ links. The reduction in the total number of links for this example is 73728 links or about 64%.

Table 1: Total number of nodes

Links/node	3	4	5	6	7	8
Hypercube	8	16	32	64	128	265
MC$(1, m)$	32	128	512	2,048	8,192	32,768
MC$(2, m)$	64	1,024	16,384	2^{18}	2^{22}	2^{26}
MC$(3, m)$	–	2,048	2^{19}	2^{27}	2^{35}	2^{43}
MC$(4, m)$	–	–	2^{20}	2^{36}	2^{52}	2^{68}

Table 1 lists the number of nodes a metacube can connect. We can see that an MC$(2, 3)$ which requires 5 links per node is sufficient for building an extremely large parallel computer system. To build a system of the same size with a hypercube, each node requires 14 links. The diameter of an MC$(2, 3)$ is 16, only 2 more than that of the hypercube. Other measures used to evaluate an interconnection network, including the topological properties and the efficiency of routing and broadcasting, are discussed in the next two sections.

3 Topological Properties of a Metacube

A topology is evaluated in terms of a number of parameters 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. Table 2 summarizes the degree, diameter, cost, average node distance and bisection width of a hypercube and a metacube network, assuming that both networks have the same number of nodes - 2^n, where $n = m2^k + k$.

The degree, or the number of links per node, of an n-cube is n. For an MC(k, m), because $n = m2^k + k$, we have $m = (n - k)/2^k$. The degree of an MC(k, m) is $m + k = (n - k)/2^k + k$.

The diameter is defined as the maximum of the shortest distances between all pairs of nodes. The diameter of an n-cube is n. The diameter of an MC(k, m) is $m2^k + 2^k$. The first term ($m2^k$) is the cost of traveling within each low-level m-cube (cluster) and the second term (2^k) is the length of a weak Hamiltonian path on a high-level k-cube. The diameter of an MC(k, m) and the weak Hamiltonian path are discussed in the next section.

The average distance in a regular network is defined as the ratio of the sum of the distances betwenn a node and all other nodes to the total number of nodes. For the sake of simplicity, a zero distance is also included. The average distance for an n-cube is $\sum_{i=0}^{n} i\binom{n}{i}/2^n = n/2$, because there are $\binom{n}{i}$ nodes at distance i from a given node. (Li & Peng 2000) derive the average node distance of an MC$(1, m)$, the dual-cube. An upper bound on the average node distance of an MC(k, m) with $k \geq 2$ is shown in the table. This upper bound can also be found in the next section.

The bisection width of an MC(k, m) is defined as the minimum number of edges connecting MC$^0(k, m)$ and MC$^1(k, m)$, where MC$^0(k, m)$ = sum of half of the clusters of class i, for $i = 0, 1, \ldots, h - 1$, and MC$^1(k, m)$ = MC(k, m) − MC$^0(k, m)$. It is easy to see that the removal of $(2^n/2^k)/2$ edges will disconnect MC$^0(k, m)$ and MC$^1(k, m)$, and this number is

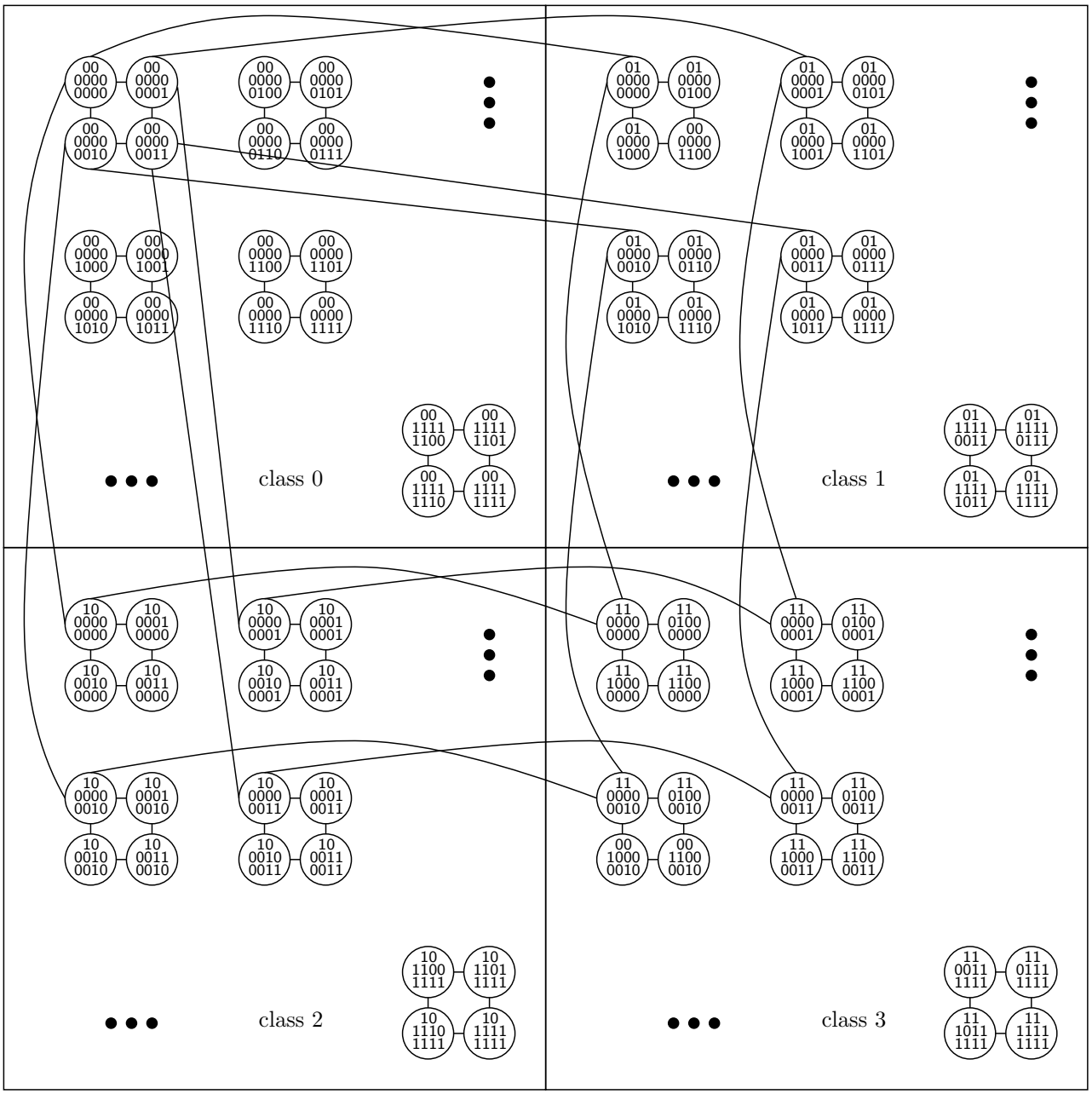


Figure 4: A metacube MC(2,2)

Table 2: Hypercube *vs* metacube

Network	Degree	Diameter	Cost	Average distance	Bisection	# of links
Hypercube	n	n	n^2	$n/2$	$2^n/2$	$2^n n/2$
MC$(1,m)$	$(n-1)/2+1$	$n+1$	$(n+1)^2/2$	$n/2+1-1/2^{(n-1)/2}$	$2^n/4$	$2^n(n+1)/4$
MC$(2,m)$	$(n-2)/4+2$	$n+2$	$n^2/4+2n+3$	$n/2+3$	$2^n/8$	$2^n(n+6)/8$
MC(k,m)	$m+k$	$(m+1)2^k$	$(m+1)2^k(m+k)$	$m2^k/2+2^k$	$2^{m2^k}/2$	$2^{m2^k+k}(m+k)/2$

the minimum for bisecting an MC(k,m). An example of the bisection width of an MC$(2,2)$ is shown in Fig 5, where the MC$^0(2,2)$ consists of the four upper parts of the classes 0, 1, 2, and 3. The nodes with thick cycles in the figure contribute to the bisection.

4 Routing and Broadcasting Algorithms

The problem of finding a path from a source node s to a destination node t, and forwarding messages along the path is known as the routing problem. A broadcast sends a message from the source node to all other nodes in the network. Routing and broadcasting are the basic communication problems for interconnection networks. In this section, we will describe routing and broadcasting algorithms for metacubes.

We adopt the following notations. In the metacube MC(k,m), each node has $m+k$ links. Among them, the m links that form an m-cube are called cube-edges and the other k links are called cross-edges. A node

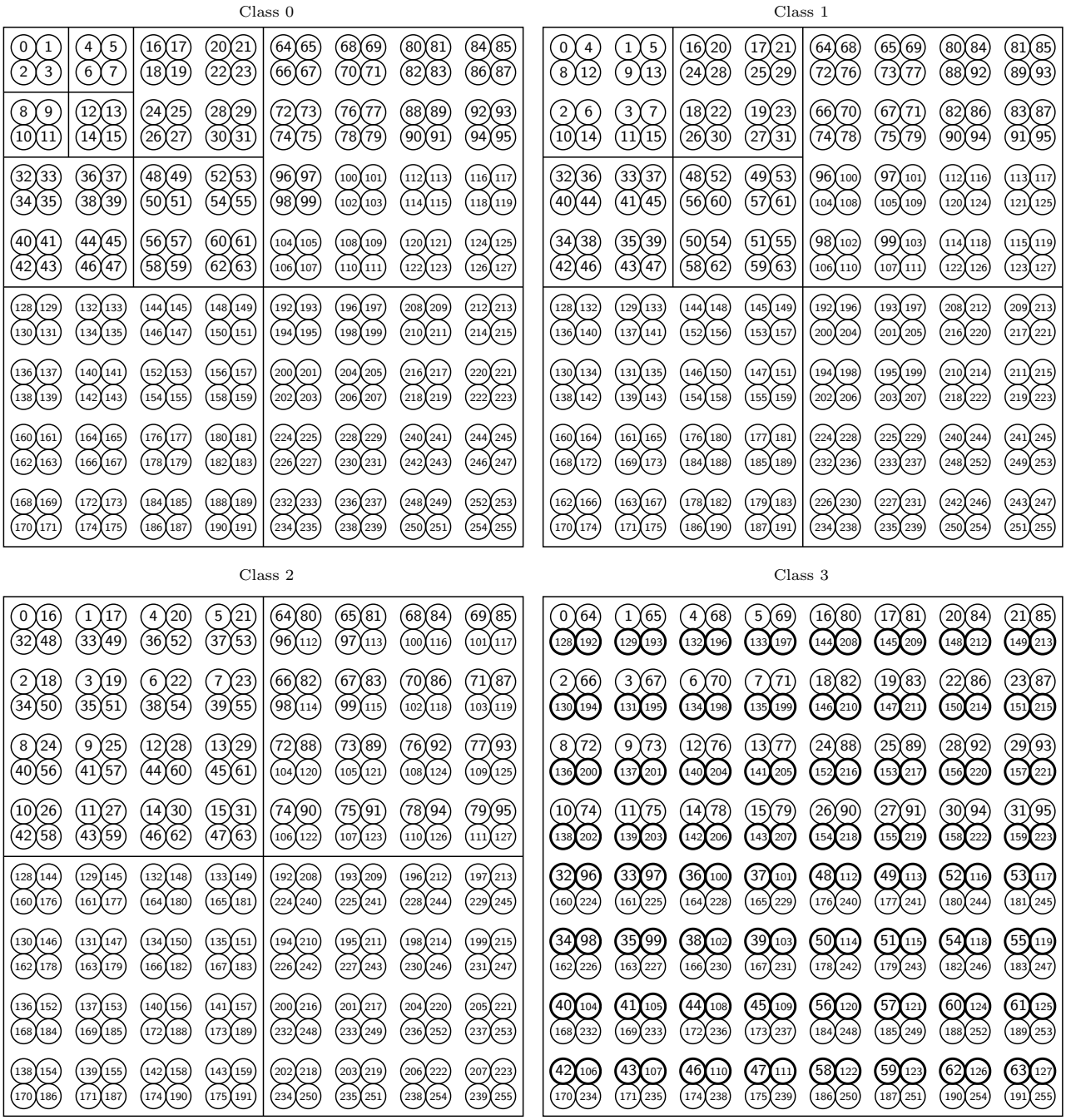


Figure 5: A metacube MC(2,2) and its bisection

address is written $(c, m_{h-1}, \ldots, m_1, m_0)$ as described in the second section. Let $s^{(i)}$, $0 \le i \le m-1$, be the ith dimension neighbor of node s within the low-level cube (cluster) such that the addresses of s and $s^{(i)}$ differ in the ith bit position in the m_j field for $j = c$. Let $s^{(i)}$, $m \le i \le m+k-1$, be the ith dimension neighbor of node s in the high-level cube such that the addresses of s and $s^{(i)}$ differ in the $(i-m)$th bit position in the field c. Let $s^{(i,j)} = (s^{(i)})^{(j)}$ for $0 \le i, j \le m+k-1$.

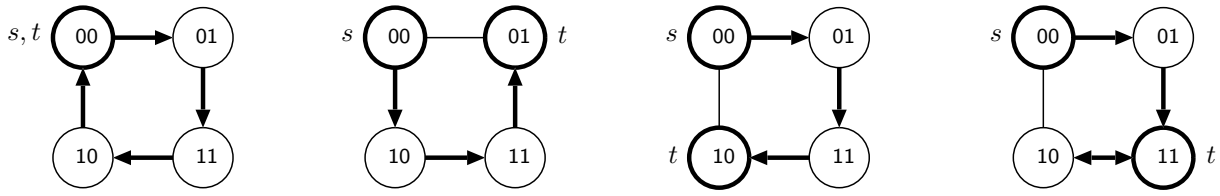
4.1 Point-to-point Routing in a Metacube

Let $P' = \langle s = v_0, v_1, \ldots, v_{h-1} = t \rangle$ be a path in a graph $G = (V, E)$ from node s to node t, where V is the set of all vertices (nodes) and E is the set of all edges in G. We say P' is a *Hamiltonian path* if

1. P' contains every node in V and
2. nodes v_i $(0 \le i \le h-1)$ are all distinct.

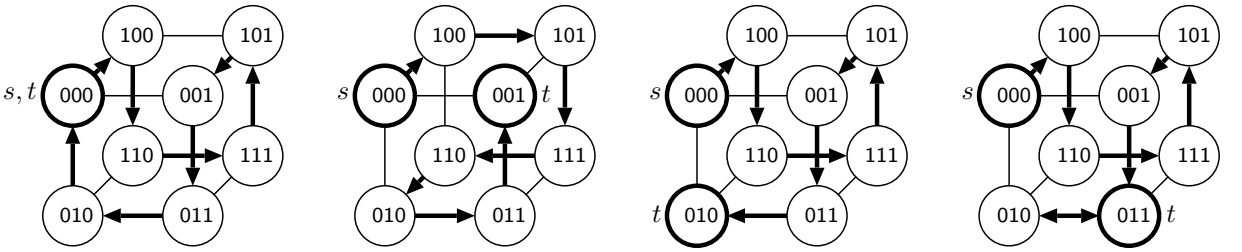
Let $P = P' \cup v_h$ and change t to $t = v_h$. Note that v_h is not a new node, $v_h = v_i$, for $i = 0, 1, \ldots,$ or $h-2$. If $t = s$, i.e. $v_h = v_0$, then P becomes a *Hamiltonian cycle* (a Hamiltonian cycle is defined as a path through a graph which starts and ends at the same vertex and includes every other vertex exactly once); otherwise, we call P a *extended Hamiltonian path*.

The length of a Hamiltonian path in an k-cube is $2^k - 1$; the length of a Hamiltonian cycle or an extended Hamiltonian path is 2^k. In the following discussion, we use the term *weak Hamiltonian path*. A weak Hamiltonian path may mean a Hamiltonian path, a Hamiltonian cycle, or an extended Hamiltonian path. We need the following lemma to solve the routing problem:

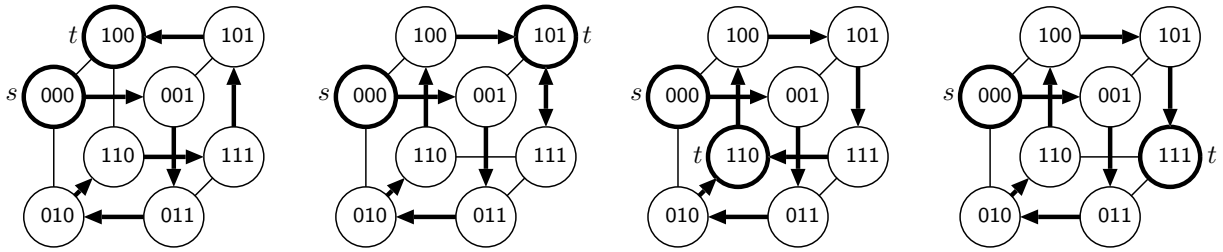


(a) Hamiltonian cycle (b) Hamiltonian path (c) Hamiltonian path (d) Extended Hamiltonian path

Figure 6: The weak Hamiltonian path in 2-cube



(a) Hamiltonian cycle (b) Hamiltonian path (c) Hamiltonian path (d) Extended Hamiltonian path



(e) Hamiltonian path (f) Extended Hamiltonian path (g) Extended Hamiltonian path (h) Hamiltonian path

Figure 7: The weak Hamiltonian path in a 3-cube

Lemma 1 *Given any two nodes s and t in an n-cube, there exists a weak Hamiltonian path from s to t.*

Proof: We first observe that, given an edge $e = (u, v)$ (where u and v are two nodes connected with e) in an n-cube, there always exists a Hamiltonian cycle passing through e. This can be demonstrated easily by renumbering the nodes of the cube so that $u = 0 \ldots 00$ and $v = 0 \ldots 01$, and then the Hamiltonian cycle can be constructed by the Gray code of the new numbers (a Gray code for binary numbers is a listing of all n-bit numbers so that successive numbers, including the first and last, differ in exactly one bit position).

We use induction on n to prove the lemma. For $n = 2$, the lemma is true as shown as in Figure 6 for $s = 00$. Figure 6 (a) is a Hamiltonian cycle ($t = 00$), Figure 6 (b) and (c) are two Hamiltonian paths ($t = 01$ and $t = 10$, respectively), and Figure 6 (d) is an extended Hamiltonian path ($t = 11$) and $P = \langle 00, 01, 11, 10, 11 \rangle$. Note that we did not renumber the node addresses in Figure 6.

Consider $n \geq 3$. An n-cube consists of two $(n-1)$-cubes, with the nodes numbered by preceding the original node numbers of the two subcubes with 0 and 1, respectively, and connecting each node with the number $0x$ to the node with the number $1x$. The two $(n-1)$-cubes forming the n-cube are known as its 0-subcube and 1-subcube, respectively. The proof of the lemma is divided into two cases.

Case 1: The nodes s and t belong to the same $(n-1)$-cube, say the 0-subcube, of the n-cube along the $(n-1)$th dimension. That is, the most-significant-bit of each node address in the 0-subcube is 0. By our induction hypothesis, there is a weak Hamiltonian path P from s to t in the 0-subcube. Suppose that the weak Hamiltonian path contains edge $e = (s, s^{(j)})$, where $s = v_0$ and $s^{(j)} = v_1$ in P. Then we find a Hamiltonian cycle in the 1-subcube that contains edge $e' = (s^{(n-1)}, (s^{(j,n-1)}))$. The weak Hamiltonian path in the n-cube can be obtained by replacing e with the path $(s, s^{(n-1)}) \cup \{s^{(n-1)} \rightarrow (s^{(j,n-1)}\} \cup (s^{(j,n-1)}, s^{(j)})$, where the subpath $\{s^{(n-1)} \rightarrow s^{(j,n-1)}\}$ is the Hamiltonian path in the 1-subcube formed by removing the edge e' from the Hamiltonian cycle.

Case 2: The nodes s and t belong to the distinct $(n-1)$-cubes, say $s \in$ 0-subcube and $t \in$ 1-subcube, of the n-cube along the $(n-1)$th dimension. First, we find a Hamiltonian cycle in the 0-subcube. Suppose that it contains edge $e = (s, s^{(j)})$, where $s = v_0$ and $s^{(j)} = v_{h-1}$. By the induction hypothesis, there is a weak Hamiltonian path P from $s^{(j,n-1)}$ to t in the 1-subcube. Then the weak Hamiltonian path in the n-cube is $\{s \rightarrow s^{(j)}\} \cup (s^{(j)}, s^{(j,n-1)}) \cup P$, where the subpath $\{s \rightarrow s^{(j)}\}$ is the Hamiltonian path in the 0-subcube formed by removing the edge e from the Hamiltonian cycle. ❑

Figure 7 shows the weak Hamiltonian path for a 3-cube. Figure 7(a-d) are in Case 1 and Figure 7(e-h) are in Case 2. Since we can modify only a small portion (m bits) of the $m2^k$-bits ID by cube-edges, we need to move to clusters of distinct classes (along

cross-edges) to modify the other portions of the ID. We also need to arrange the traveling order so that the last portion modified is the nodeID of t or its neighbor, so that we do not need to travel a long distance back to t - doing nothing. Since there are 2^k classes, the efficient way to do this is by following a weak Hamiltonian path from c^s (class number of node s) to c^t (class number of node t) in the k-cube. The routing algorithm is set out in detail below.

Let the node addresses of s and t be $(c^s, m^s_{h-1}, \ldots, m^s_1, m^s_0)$ and $(c^t, m^t_{h-1}, \ldots, m^t_1, m^t_0)$, respectively. A path from s to t can be found as follows:

Step 1: Find a weak Hamiltonian path from c^s to c^t in the k-cube. Let the weak Hamiltonian path be $\langle c^s = c_0, c_1, \ldots, c_r = c^t \rangle$, where $r = h-1$ if the weak Hamiltonian path is a Hamiltonian path and $r = h$ otherwise.

Step 2: For each class with classID $= c_i$, $0 \leq i \leq h-1$, find a shortest path P_i from the node with nodeID $= m^s_{c_i}$ to the node with nodeID $= m^t_{c_i}$ in the cluster with clusterID $= (m^u_{h-1}, \ldots, m^u_{c_{i+1}}, m^u_{c_{i-1}} \ldots, m^u_0)$, where in m^u_j, $j \neq c_i$, $u = t$ if $j \in \{c_0, \ldots, c_{i-1}\}$ and $u = s$ otherwise.

Step 3: The routing path $s \rightarrow t$ is obtained by concatenating all P_i, $0 \leq i \leq h-1$, and interleaving P_{i-1} and P_i for $1 \leq i \leq (h-1)$ by a cross-edge which changes the classID of the nodes from c_{i-1} to c_i. If $c_{h-1} \neq c^t$, i.e. $r = h$, add the cross edge corresponding to the last edge in the weak Hamiltonian path to the routing path.

A routing example in an MC(2, 3) from s = (00,000,000,000,000) to t = (00,001,110,101,011) is shown below, where the weak Hamiltonian path is a Hamiltonian cycle of $00 \rightarrow 01 \rightarrow 11 \rightarrow 10 \rightarrow 00$.

(00,000,000,000,000) → (00,000,000,000,001) →
(00,000,000,000,011) → (01,000,000,000,011) →
(01,000,000,001,011) → (01,000,000,101,011) →
(11,000,000,101,011) → (11,001,000,101,011) →
(10,001,000,101,011) → (10,001,010,101,011) →
(10,001,110,101,011) → (00,001,110,101,011)

Let $d_i(s,t)$, $0 \leq i \leq h-1$, be the Hamming distance between s and t in m_i, i.e. the number of bits with distinct values in m_i for s and t. From the algorithm, the longest length of the routing path is $2^k + d_{sum}(s,t)$, where $d_{sum}(s,t) = \sum_{i=0}^{h-1} d_i(s,t)$. This formula gives an upper bound to the distance $d(s,t)$ between s and t in an MC(k,m). Clearly, we have $d_{sum}(k,m) \leq d(s,t) \leq d_{sum}(s,t) + 2^k$. The longest path in an MC(k,m) is from $s = 0 \ldots 0$ to t, where $c^t = 0 \ldots 0$ and $m^t_i = 1 \ldots 1$ for all i, $0 \leq i \leq h-1$. The length of this path is $2^k(m+1)$. It is easy to see that this path is the shortest path for connecting s and t. Therefore, it is the diameter of an MC(k,m).

Next, since the routing algorithm always travels through a cluster in every class, the length of the routing path is independent of the values of the classID of s or t. Since the average distance in each cluster is $m/2$ and there are 2^k clusters, the average distance of any two nodes in an MC(k,m) is $(m/2)2^k + 2^k = (n-k)/2 + 2^k$, where $n = m2^k + k$. Notice that it is possible to have a routing algorithm in an MC(k,m) which bypasses the class c if $m^s_c = m^t_c$. In such a case, the length of the routing path for some s and t might be shorter than that produced by the algorithm above. We put these results into the following theorem.

Theorem 2 *In an MC(k,m), let $d(s,t)$ and $d_{avg}(s,t)$ be the distance and the average distance between any two nodes s and t, respectively. Let $d_i(s,t)$, $0 \leq i \leq h-1$, be the Hamming distance between s and t in m_i. Then $d(s,t) \leq \sum_{i=0}^{h-1} d_i(s,t) + 2^k$ and $d_{avg} \leq (n-k)/2 + 2^k$. The diameter of an MC$(k,m)$ is $2^k(m+1)$.*

From Theorem 2, the length of the routing path for some s and t might be 2^k longer than that of the routing path for s and t in the n-cube in the worst case. This is mainly due to the fact that there are 2^k classes in an MC(k,m) and the direct connection between two nodes whose IDs differ in 1-bit only is limited to a small portion of $n = m2^k + k$ bits. However, the size of an MC network increases extremely rapidly with k, e.g. the number of nodes in an MC$(k,3)$ are 2^{14}, 2^{27}, and 2^{52}, for $k = 2$, 3, and 4, respectively. Therefore, no matter how large the network required, it is practically sufficient to consider $k \leq 3$. In these cases, the length of the routing path for s and t in the MC will be at most that of the shortest routing path for s and t in the n-cube plus eight, even though the network might have hundreds of millions nodes.

4.2 Broadcasting in a Metacube

In this subsection, we show how to perform one-to-all broadcasting in a metacube. We assume that the communication links are bidirectional; that is, two directly-connected processors can send messages to each other simultaneously. We also assume the processor-bounded model (one-port model) in which each processor cannot use more than one link for sending messages nor receive more than one message at a time. The port model of a network system refers to the number of internal channels at each node. In order to reduce the complexity of communication hardware, many systems support one-port communication architectures, in which each node can send and receive a certain amount of data through a link in one unit time. We want to minimize the transmission time for one-to-all broadcast in metacube. It is known that one-to-all broadcast cannot be done in less than $\log p$ time on any architecture in the one-port model, where p is the number of nodes in the network.

For efficient broadcasting, we arrange the clusters in a Hamiltonian cycle using the Gray code of the classIDs of the clusters. That is, each node u has a link *next* to a node *next*(u) in another cluster and after applying *next* 2^k times, it will return to u again. Let the source node be s. The algorithm for broadcasting a message from s to all other nodes in an MC(k,m) is:

1. Broadcast the message from s to the nodes whose clusterID and nodeID are the same as that of s;

For $i = 0$ to $2^k - 1$ **do**
  For each cluster in which at least one node has the message **pardo**
    2. Broadcast the message to all other nodes in the cluster;
    3. If $(i < 2^k - 1)$, each node u in the cluster sends the message to the *next*(u);
  Endfor
Endfor

Table 3 shows an example of broadcasting with s = 0000000000 for an MC$(2,2)$. Only the first iteration of the "for" loop is listed in the table.

Now, we show that the algorithm is correct and derive the transmission time of the broadcast algorithm.

Table 3: First three broadcasting steps in an MC(2,2)

1. broadcast in k-cube	3. go through the *next* link
0000000000 → 0100000000	0000000000 → 0100000000
0000000000 → 1000000000	0000000001 → 0100000001
0100000000 → 1100000000	0000000010 → 0100000010
2. broadcast in m-cube	0000000011 → 0100000011
0000000000 → 0000000001	0100000000 → 1100000000
0100000000 → 0100000100	0100000100 → 1100000100
1000000000 → 1000010000	0100001000 → 1100001000
1100000000 → 1101000000	0100001100 → 1100001100
0000000000 → 0000000010	1000000000 → 0000000000
0000000001 → 0000000011	1000010000 → 0000010000
0100000000 → 0100001000	1000100000 → 0000100000
0100000100 → 0100001100	1000110000 → 0000110000
1000000000 → 1000100000	1100000000 → 1000000000
1000010000 → 1000110000	1101000000 → 1001000000
1100000000 → 1110000000	1110000000 → 1010000000
1101000000 → 1111000000	1111000000 → 1011000000

In the first step, the source node broadcasts the message in the high-level cube, k-cube, using a binomial tree through the cross-edges. At the end of the first step, each class will have exactly one node containing the message. In the second step, every node which holds the message broadcasts the message within in the low-level cube, m-cube cluster, using a binomial tree through the cube-edges. In the third step, every node which holds the message sends the message to a node in a different cluster through the *next* link. The second and third steps will repeat 2^k times ("for" loop). The number of clusters that hold the message after each iteration will be 2^k times more than before the iteration. That is, at the end of the ith iteration $(0 \leq i \leq 2^k - 1)$, $2^{m(i-1)}$ clusters in each class will be fully covered (i.e. every node in the cluster has received the message). Therefore, at the end of the "for" loop, $2^{m(2^k-1)}$ clusters in each class will be fully covered. That is, all nodes in an MC(k, m) will have received the message.

The transmission time for the broadcasting may be determined as follows. The first step requires k time using a binomial tree in the k-cube. Each iteration of the "for" loop except the last one requires $m + 1$ time since broadcasting inside the cluster requires m time and going through *next* requires 1 time. So, the "for" loop takes $(m + 1)2^k - 1$ time. Therefore, the transmission time for broadcast in an MC(k, m) is $(m+1)2^k + k - 1$. We summarize this result into the following theorem.

Theorem 3 *Assume that each node can use only one link at a time and that sending and receiving a message through a link takes unit time. One-to-all broadcast in an MC(k, m) requires $(m+1)2^k + k - 1$ time.*

5 Conclusion and Future Work

In this paper, we proposed a new interconnection network, the metacube, and showed that point-to-point routing and the broadcasting can be done efficiently on it. The proposed metacube has tremendous potential to be used as an interconnection network for very large scale parallel computers since the metacube can connect hundreds of millions nodes with up to 6 links per node and it keeps some desired properties of the hypercube that are useful for efficient communication among the nodes.

Recently, much of the community has moved on to lower-dimensional topologies such as meshes and tori. However, the SGI Origin2000, a fairly recent multiprocessor, does use a hypercube topology, so the metacube could be of use to industry. A lot of issues concerning the metacube require further research. Some of them are:

1. Evaluate the architecture complexity vs. performance of benchmarks vs. real cost.
2. Investigate the embedding of other frequently used topologies into a metacube.
3. Develop techniques for mapping application algorithms onto a metacube.
4. Develop fault-tolerant routing algorithms for a metacube with faulty nodes.

Acknowledgments

We wish to thank the anonymous reviewers for their comments and suggestions. Special thanks go to John Morris for checking and proofreading this paper.

References

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

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

Ghose, K. & Desai, K. R. (1995), 'Hierarchical cubic networks', *IEEE Transactions on Parallel and Distributed Systems* **6**(4), 427–435.

Hayes, J. P. & Mudge, T. N. (1989), 'Hypercube supercomputers', *Proc. IEEE* **17**(12), 1829–1841.

Li, Y. & Peng, S. (2000), Dual-cubes: a new interconnection network for high-performance computer clusters, *in* 'Proceedings of the 2000 International Computer Symposium, Workshop on Computer Architecture', pp. 51–57.

Li, Y. & Peng, S. (2001), Fault-tolerant routing and disjoint paths in dual-cube: a new interconnection network, *in* 'Proceedings of the 2001 International Conference on Parallel and Distributed Systems', IEEE Computer Society Press, pp. 315–322.

Preparata, F. P. & Vuillemin, J. (1981), 'The cube-connected cycles: a versatile network for parallel computation', *Commun. ACM* **24**, 300–309.

SGI (1998), *Performance tuning optimization for Origin2000 and Onyx2*, http://techpubs.sgi.com/library/manuals/3000/007-3511-001/html/O2000Tuning.0.html.

Tucker, L. W. & Robertson, G. G. (1988), 'Architecture and applications of the connection machine', *IEEE Computer* **21**, 26–38.

Vanvoorst, B., Seidel, S. & Barscz, E. (1994), Workload of an ipsc/860, *in* 'Proceedings of the Scalable High-Performance Computing Conference', pp. 221–228.

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