Multinode Broadcasting in Metacube

Yamin Li, Shietung Peng
Department of Computer Science
Hosei University
Tokyo 184-8584 Japan
{yamin,speng}@k.hosei.ac.jp

Wanming Chu
Department of Computer Hardware
University of Aizu
Aizu-Wakamatsu 965-8580 Japan
w-chu@u-aizu.ac.jp

Abstract

A new interconnection network for very large parallel systems called metacube *(MC) has been introduced recently. An MC(k,m) network has 2^{m2^k+k} nodes with m+k links per node, where k is the dimension of the high-level cubes and m is the dimension of the low-level cubes. For example, an MC(3,3) with 6 links per node can connect more than one hundred of millions of nodes, extremely larger than that of hypercube. Meanwhile, the MC network is a symmetric network and retains the main structures and desirable properties of the hypercube. In this paper, we give efficient algorithms for multinode broadcasting in MC networks. The time complexities of the routing and broadcasting algorithms are analyzed and compared with that of hypercube algorithms. Our results show that the routing and multinode broadcasting can be done efficiently in MC networks.*

Keywords: interconnection networks, hypercube, routing, broadcasting, algorithm

1. Introduction

The hypercube has been widely used as the interconnection network in a wide variety of parallel systems such as Intel iPSC [1], the nCUBE [2], the Connection Machine CM-2 [3], and SGI Origin 2000 [4]. 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 [4]. 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. Therefore, it is interesting to develop an interconnection network which will link a large number of nodes with 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 [5] and crossed cube [6]; some focused on reduction of the number of edges of the hypercube, for example cube-connected cycles (CCC) [7] and reduced hypercube (RH) [8]; and some focused on both, as in the hierarchical cubic network (HCN) [9]. One major property of the hypercube is: there is 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 cubic network, will not satisfy this key property.

Recently, Y. Li et al. introduced a new interconnection network, called *metacube*, or MC network [10]. The MC network shares many desirable properties of the hypercube (e.g., the key property of the hypercube, low 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 amount 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.

In this paper, we give efficient algorithms for routing, one-to-all broadcasting and all-to-all broadcasting in metacubes. The remainder of this paper is organized as follows. Section 2 introduces the metacube interconnection network architecture. Section 3 gives the routing and broadcasting algorithms and their time complexities. Sections 4 concludes the paper and presents some future research directions.

2. Metacube Architecture

This section introduces the MC network and some related notation. The MC network is motivated by the dual-cube network proposed by Li and Peng [11] [12] 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 amount of link per node. The MC network includes the dual-cube as a special case. An MC network has a 2-level cube structure: high-level cubes 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 form the basic components in the network, represented by the m bits of the remain $m2^k$ bits, which occupy the different portions in the $m2^k$ bits for different classes.

More specifically, there are two parameters in an MC network, k and m. An MC(k,m) contains $h = 2^k$ *classes*. Each class contains $2^{m(h-1)}$ *clusters*, and each cluster contains 2^m *nodes*. Therefore, an MC(k,m) uses $mh+k$ binary bits to identify a node and the total number of nodes is 2^n where $n = mh+k$. 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.

A node in an MC(k,m) can be uniquely identified by a $(mh+k)$-bit binary number. The leftmost k-bit binary number defines a class of the node (*classID*). There are h classes. In each class, there are 2^{mh} nodes and each node is represented by a mh-bit binary number. 2^m nodes of the same class form a cluster. Therefore, there are $2^{m(h-1)}$ clusters in each class. An m-bit binary number, located in a special portion of the mh-bit (will be explained in the next paragraph) identifies a node within the cluster (*nodeID*). Therefore, the $(mh+k)$-bit node address in an MC(k,m) is divided into three parts: a k-bit *classID*, an $m(h-1)$-bit *clusterID* and an m-bit *nodeID*.

In the following discussion, we use $u = (c_u, M_u[h-1], \ldots, M_u[1], M_u[0])$ to denote the *ID* of node u, where c_u is a k-bit binary number and $M_u[i],\ 0 \le i \le h-1$ are m-bit binary numbers. Let $classID(u) = c_u$, $nodeID(u) = M_u[c_u]2^{c_u}$ and $clusterID(u) = \sum_{0 \le i \ne c_u \le h-1} M_u[i]2^i$. The mh-bit number $nodeID(u) + clusterID(u)$ is a unique identifier of node u in class c_u. For example, $u = 01,00,11,10,00$ in an MC$(2,2)$ is denoted as node 56 of class 1 and node set $(48,52,56,60)$ in class 1 forms a cluster with $clusterID = 48$.

The links of an MC(k,m) is constructed in the following manner. The m-bit field $M[c]$ in the address of a node of class c forms a low-level m-cube with m links, namely *cube-edges*. These *low-level* m-cubes are called *clusters*. A cluster containing node u is denoted as C_u. The links that connect nodes among clusters are called *cross-edges* and are defined as following. For any two nodes whose addresses differ only in a bit position in the class field, there is a cross-edge connecting these two nodes. That is, the k-bit field c forms a *high-level* k-cube which connects those nodes whose addresses except class field are the same.

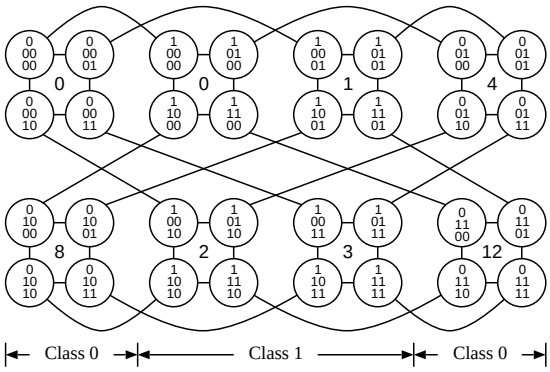

Figure 1. A metacube MC(1,2)

The addresses of two nodes connected by a cross-edge differ only on one bit position within the k-bit class field and there is no direct connection among the clusters of the same class. There-

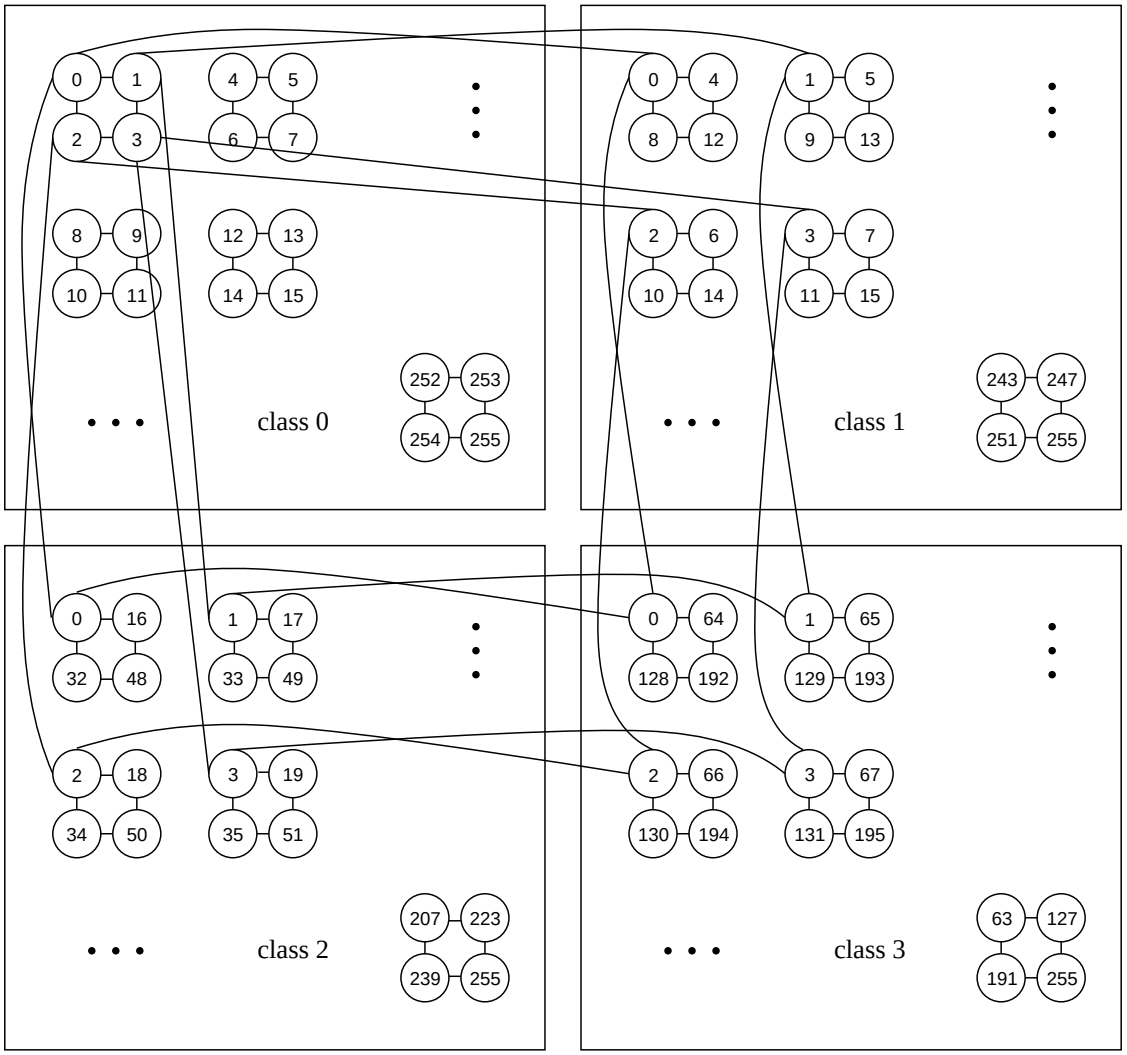

Figure 2. A metacube MC(2,2)

fore, 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 (01,111,101,11$\underline{1}$,000), (01,111,101,1$\underline{0}$0,000) and (01,111,101,$\underline{0}$10,000). The underlined bits are those that differ from the corresponding bits in the address of the referenced node. The two neighbors in the high-level cube are (0$\underline{0}$,111,101,110,000) and ($\underline{1}$1,111,101,110,000).

Figure 1 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 different class. The binary number shown in the center of a cluster is *clusterID*. Figure 2 shows the structure of an MC(2,2), where the clusters in the same square are of the same class. The decimal numbers are $nodeID + clusterID$. In Figure 2, 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, each of which contains a distinct node in the cluster 0 of the class 0.

The ratio of the total number of links in the hypercube to the total number of links in the 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; the hypercube contains $16384 \times 14/2 = 114688$ links and the MC network contains $16384 \times (3+2)/2 = 40960$ links. The reduction in the total number of links for this example is 73728 links or about 64%. Meanwhile, the diameter of the MC network in this example is 16, only 2 more larger than that of the hypercube.

3. Routing and broadcasting algorithms

The problem of finding a path from a source node s to a destination node t and forwarding mes-

sages along the path is known as the routing problem. The broadcast is to send a message from the source node to all other nodes in the network. The multinode broadcast is that a set of nodes broadcast simultaneously to all other nodes in the network. Routing and broadcasting are the basic communication problems for interconnection networks. In this section, we describe routing and broadcasting algorithms for the metacube.

We assume the package routing model (the store-and-forward routing model) in which each package is maintained as an entity that passed from node to node as it moves through the network and a single package can cross each edge during each step of routing. We allows packages that headed for the same destination to be combined. With the package routing, the communication time for message of length l to be sent to a node of distance d is $t_s + d\,l\,t_w$, where t_s is startup latency, the time required for the system to handle the message at the source and destination nodes, t_w is the per-word transfer time.

We adopt the following notation. In the metacube MC(k,m), each node has $m+k$ neighbors. Let $s^{(i)}$, $0 \le i \le k-1$, be the ith dimensional neighbor of node s within the k-cube, that is, the addresses of s and $s^{(i)}$ differ in the ith bit position (the rightmost bit is the 0th bit) in the class field c. Let $s^{(i+k)}$, $0 \le i \le m-1$, be the ith dimensional neighbor of node s in the m-cube, that is, the addresses of s and $s^{(i+k)}$ differ in the ith bit position in the field $M[c]$. Let $s^{(i,j)} = (s^{(i)})^{(j)}$ for $0 \le i, j \le m+k-1$. We use $(u \to v)$ to denote a path from node u to node v. If the length of a path $(u \to v)$ is 1 (through a single edge), the path is denoted as $(u : v)$, and the edge is denoted as (u,v).

3.1. Point-to-point routing in metacube

To present the routing algorithm in the metacube, we first introduce the concept of a *weak-Hamiltonian path*. In a graph $G = (V,E)$ where V is the set of all vertices (nodes) and E is the set of all edges in G, let $P' = (v_0 \to v_{h-1}) = (v_0 : v_1 : \ldots : v_{h-1})$ be a path from node v_0 to node v_{h-1}, where $v_i \in V$ for $0 \le i \le h-1$ and edge $(v_{j-1}, v_j) \in E$ for $1 \le j \le h-1$. 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 $P = (v_0 \to v_h) = (P' : v_h)$, where $v_h = v_i$, for $i = 0, 1, \ldots,$ or $h-2$. If $v_h = v_0$, then P becomes a *Hamiltonian cycle*[1]; otherwise, we call P a *extended-Hamiltonian path*. The length of a Hamiltonian path in a k-cube is $2^k - 1$; the length of a Hamiltonian cycle or an extended Hamiltonian path is 2^k. Let a *weak-Hamiltonian path* be a Hamiltonian path, a Hamiltonian cycle, or an extended-Hamiltonian path. We need the following lemma to solve the routing problem:

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 $e = (u,v)$ in the n-cube, there always exists a Hamiltonian cycle passing through e. This can be demonstrated easily by renumbering the nodes of the cube such that $u = 0\ldots00$ and $v = 0\ldots01$, and then the Hamiltonian cycle can be constructed by the Gray code[2] of the new numbers.

We use induction on n to prove the lemma. For $n = 2$, the lemma is true: the four weak-Hamiltonian paths starting from node 00 are 00 : 01 : 11 : 10 : 00 (Hamiltonian cycle), 00 : 10 : 11 : 01 (Hamiltonian path), 00 : 01 : 11 : 10 (Hamiltonian path) and 00 : 01 : 11 : 10 : 11 (extended-Hamiltonian path). Consider $n \ge 3$. 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. That is, the most-significant-bit of each node address is 0. By our induction hypothesis, there is a weak Hamiltonian path P from s to t in the 0-subcube. Suppose that P contains edge $e = (s, s^{(j)})$, i.e. $P = (s : s^{(j)} \to t)$. Then we find a Hamiltonian cycle in the 1-subcube that contains edge $e' = (s^{(n-1)}, s^{(j,n-1)})$. A weak Hamiltonian path in the n-cube is $(s : s^{(n-1)} \to s^{(j,n-1)} : s^{(j)} \to t)$, where the subpath $(s^{(n-1)} \to s^{(j,n-1)})$ 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. First, we find a Hamiltonian cycle in the 0-subcube. Suppose that it contains edge $e = (s, s^{(j)})$. By the induction hypothesis, there is a weak Hamiltonian path $P = (s^{(j,n-1)} \to t)$ in the 1-subcube. Then the weak Hamiltonian path

[1] 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.

[2] 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.

in the n-cube is $(s \to s^{(j)} : s^{(j,n-1)} \to t)$, where the subpath $(s \to s^{(j)})$ is the Hamiltonian path in the 0-subcube formed by removing the edge e from the Hamiltonian cycle. □

Since we can modify only a small portion (m bits) 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 h 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.

For each node u in the k-cube, let $next(u)$ be the node next to u in the weak-Hamiltonian path from c_s to c_t. $next(u) = \emptyset$ if u is the last node of the weak-Hamiltonian path. 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. The routing algorithm is given below. The **Loop** will terminate when the **break** is executed. Notice that the details of routing in the hypercube is omitted in the algorithm.

Algorithm 1 (One2OneRouting(m,k,s,t))

1. **begin** /* build a $P = (s \to t)$ in MC(k,m) */
2. $u = c_s$;
3. $v = s$;
4. $P = v$;
5. **Loop** always
6. $w = (u, M_v[h-1], \ldots, M_v[u+1], M_t[u], M_v[u-1], \ldots, M_v[0])$;
7. **if** $(w \neq v)$ $P = (P \to w)$;
8. **if** $(w == t)$ **break**;
9. $v = w$;
10. $w = (next(u), M_v[h-1], \ldots, M_v[u+1], M_v[u], M_v[u-1], \ldots, M_v[0])$;
11. $P = (P : w)$;
12. $u = next(u)$;
13. **end.**

Example 1. In an MC(2,3), assume $s = 00,000,000,000,000$, $t = 00,001,110,101,011$. The weak-Hamiltonian path for $(s \to t)$ in the high-level 2-cube is a Hamiltonian cycle of $00 : 01 : 11 : 10 : 00$. The path $(s \to t)$ is

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 $H_i(s,t)$, $0 \leq i \leq h-1$, be the Hamming distance between s and t in $M[i]$, i.e. the number of bits with distinct values in $M_s[i]$ and $M_t[i]$.

From the algorithm, the longest length of the routing path is $2^k + H_h(s,t)$, where $H_h(s,t) = \sum_{i=0}^{h-1} H_i(s,t)$. This formula gives an upper bound to $d(s,t)$, the distance between s and t in an MC(k,m).

Clearly, we have $H_h(s,t) \leq d(s,t) \leq H_h(s,t) + 2^k$. Let $H(s,t)$ be the Hamming distance between s and t, then $H(s,t) = H_h(s,t) + H_k(s,t)$, where $H_k(s,t)$ is the Hamming distance between s and t in c field. We have $H_h(s,t) \leq d(s,t) \leq H(s,t) + 2^k - H_k(s,t)$.

The longest path in an MC(k,m) is from $s = 0\cdots0$ to t, where $c_t = 0\cdots0$ and $M_t[i] = 1\cdots1$ 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).

Since the average distance in each cluster is $m/2$, the average distance between any two nodes in an MC(k,m) is at most $(m/2)2^k + 2^k = (n-k)/2 + 2^k$, where $n = m2^k + k$ (in the case of Hamiltonian path, it is $(n-k)/2 + 2^k - 1$).

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 1. *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 $H(s,t)$ be the Hamming distance between s and t. Then $d(s,t) \leq H(s,t) + 2^k - H_k(s,t)$ and $d_{avg} \leq (n-k)/2 + 2^k$, where $H_k(s,t)$ is the Hamming distance between s and t in the class field. The diameter of an MC(k,m) is $2^k(m+1)$.*

From Theorem 1, the length of the routing path for some s and t might be 2^k longer than that of the routing path for the 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 *ID*s differ in 1-bit only is limited to a small portion of $n = m2^k + k$ bits.

However, the size of the 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 such 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 of nodes.

3.2. One-to-all broadcast in metacube

In this subsection, we show how to perform one-to-all broadcast [13] in 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 certain amount of data through a link in 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 broadcast, we first construct a Hamiltonian cycle in the k-cube using the Gray code of the $classID$s. For each node u in the k-cube, let $next(u)$ is the node next to u on the Hamiltonian cycle. Let the source node be s. The pseudocode of the algorithm for broadcasting a message from s to all other nodes in an MC(k,m) is set out in detail below. Notice that all the nodes execute the pseudocode simultaneously and each node has its own $my\_ID = (my\_classID, my\_clusterID + my\_nodeID)$.

Algorithm 2 (One2AllB(m,k,my_ID,s,msg))

```
1. begin                                  /* Node s broadcasts */
2.   u = c_s;
3.   maskh = 2^{m2^k} - 1;
4.   maskm = 2^m - 1;
5.   for i = 0 to 2^k - 1 do
6.     maskh = maskh ⊕ (maskm << (m × u));
7.     if ((my_ID ∧ maskh) == (s ∧ maskh)) &
           (my_classID == u)
8.       Cube_Bcast(m, my_nodeID, M_s[u], msg);
9.     if (i ≠ 2^k - 1)
10.      if (my_classID == u)
11.        send msg to (next(my_classID),
               my_clusterID + my_nodeID);
12.      u = next(u);
13.  Cube_Bcast(k, my_classID, u, msg);
14. end.
```

```
15. Procedure CubeBcast(d, my_id, s, msg)
16. begin      /* one-to-all broadcast in a d-cube */
17.   mask = 2^d - 1;
18.   For i = 0 to d - 1 do
19.     mask = mask ⊕ 2^i;
20.     if (((my_id ⊕ s) ∧ mask) == 0)
21.       if (((my_id ⊕ s) ∧ 2^i) == 0)
22.         send msg to ((my_id ⊕ s) ⊕ 2^i) ⊕ s;
23. end.
```

Example 2. In an MC$(2,2)$, assume $s = 00,00,00,00,00$. The first "for" loop of Algorithm 2 has four iterations. The nodes of each class that received msg after each iteration are indicated below.

i	$ClassID$	Node ($clusterID + nodeID$)
0	1	0–3
1	3	0–15
2	2	0–15, 64–79, 128–143, 192–207
3	2	All nodes

Next, we show that the algorithm is correct and derive the transmission time of the broadcast algorithm. In the first stage, at each iteration, the node which holds the message broadcasts the message within the cluster, an m-cube, through the binomial tree of the cluster (line 8), and then, each node in the cluster sends the message to a distinct cluster through the cross-edge (line 11). Therefore, the number of clusters that hold the message increased by a factor of 2^m. There are $h = 2^k$ iterations in the first stage. At the end of the first stage, every node in the clusters of class u, where u is the last node of the Hamiltonian path, will receive the message. In the second stage, every node in the cluster of class u broadcasts the message to the nodes in the clusters of all other classes through the binomial tree of the k-cube (line 13). Therefore, at the end of the second stage, all nodes in an MC(k,m) received the message.

The transmission time for the broadcasting may be determined as follows. The broadcast through Hamiltonian path can be done in $h = 2^k$ iterations. Each iteration of the "for" loop, except for the last iteration, takes $(m+1)(t_s + lt_w)$ time since broadcasting inside the cluster requires m steps and going through $next$ requires one step. The last iteration requires m steps only. So, the first stage can be done in $[(h-1)(m+1)+m](t_s + lt_w) = (mh+h-1)(t_s + lt_w)$ time. The second stage requires $k(t_s + lt_w)$ time. Therefore, the transmission time for the broadcast in an MC(k,m) is $(mh+k+h-1)(t_s + lt_w) = (h-1+\log p)(t_s + lt_w)$, where $p = 2^{mh+k}$, the total number of nodes. We summarize the result into the following theorem.

Theorem 2. *Assume that each node can use only one link at a time and the package routing model is adopted. The one-to-all broadcast in an MC(k,m) can be done in* $(2^k - 1 + \log p)(t_s + l\,t_w)$ *time.*

3.3. All-to-all broadcast in metacube

All-to-all broadcast [13] is a generalization of one-to-all broadcast in which all nodes simultaneously initiate a broadcast. A node sends the same l-word message to every other node, but different nodes may broadcast different messages. The communication pattern of all-to-all broadcast can be used to perform some other operations, such as reduction and prefix sums.

The lower bound for the communication time of all-to-all broadcast for parallel computers on which a node can communicate on only one of its ports at a time is $(p-1)lt_w$, where p is the number of nodes. This is because each node receives at least $(p-1)l$ words of data, regardless of the architecture or routing scheme. An efficient way to perform all-to-all broadcast is to perform all p one-to-all broadcasts simultaneously so that all messages traversing the same path at the same time are concatenated into a single message whose size is the sum of the sizes of individual messages.

The algorithm for all-to-all broadcast in metacubes can be described in two stages. In the first stage, the messages are broadcasted inside each cluster and then are sent through cross-edges. In every broadcast step in the cluster, pairs of nodes exchange their data and double the size of the message to be transmitted in the next step by concatenating the received message with their current data. And then, each node in a cluster of class i, $0 \le i \le h-1$, sends the identical message to a node in a cluster of class $next(i)$, where $next(i)$ is defined as the node next to i on the Hamiltonian cycle (similarly, $prev(i)$ is the node before i on the Hamiltonian cycle). This process is repeated $h = 2^k$ times. After this stage, every node of class i received a concatenated message containing messages from all the nodes of class j, for some j. Finally, in the last stage, every node exchanges and combines the concatenated message, through the edges in the k-cube. The pseudocode of the algorithm is listed below. The algorithm is executed at all nodes concurrently. *my_ID* is the *ID* of the node. The initial message to be broadcasted is *my_msg* at each node. At the end of the procedure, each node stores the collection of all p messages in *result*.

Algorithm 3 (All2AllB($m,k,my\_ID,my\_msg,result$))

```
1. begin
2.   result = my_msg;
3.   for j = 0 to 2^k − 1 do
4.     for i = 0 to m − 1 do
5.       partner = my_ID ⊕ 2^(i+m×my_classID);
6.       send result to partner;
7.       receive msg from partner;
8.       result = result ∪ msg;
9.     if (j ≠ 2^k − 1)
10.      send result to (next(my_classID),
                    my_clusterID+my_nodeID);
11.      receive msg from (prev(my_classID),
                    my_clusterID+my_nodeID);
12.      result = msg;
13.  for j = 0 to k − 1 do
14.    partner = my_ID ⊕ 2^(j+m2^k);
15.    send result to partner;
16.    receive msg from partner;
17.    result = result ∪ msg;
18. end.
```

Example 3. Assume the metacube is MC(2,2). The first "for" loop of Algorithm 3 has four iterations. In the following, we show the content of *result* in some nodes after each iteration, that is, the nodes whose *msg* are included in *result*.

- After the 1st iteration, nodes 0, 4, 8 and 12 of class 1 (in one cluster) will contain messages from nodes (0–3), (4–7), (8–11) and (12–15) of class 0, respectively.
- After the 2nd iteration, nodes 0, 64, 128 and 192 of class 3 (in one cluster) will contain messages from nodes (0–15), (64–79), (128 – 143) and (192–207) of class 0, respectively.
- After the 3rd iteration, nodes 0, 16, 32 and 48 of class 2 (in one cluster) will contain messages from (0–15, 64–79, 128–143, 192–207), (16–31, 80–95, 144–159, 208–223), (32–47, 96–111, 160–175, 224–239) and (48–63, 112–127, 176–191, 240–255) of class 0, respectively.
- After the 4th iteration, every node of classes 0, 1, 3 and 2 will contain messages from all the nodes of classes 1, 3, 2 and 0, respectively. After the broadcasting in the k-cube, every node will contain messages from all the nodes in an MC(k,m).

3.4. Transmission Time Analysis

We analyze the transmission time of the algorithm as follows. The time it takes to complete the first stage contains two parts: one for broadcasting inside clusters and one for transferring data between clusters through cross-edges.

$$T_1 = \sum_{j=1}^{h}\sum_{i=1}^{m}(t_s + 2^{(j-1)m+(i-1)}\,l\,t_w) + \sum_{j=1}^{h-1}(t_s + 2^{jm}\,l\,t_w)$$

Table 1. Communication Times: Hypercube vs Metacube

Comm. Pattern	Hypercube	Metacube
$T_{one-to-one}$	$t_s + H(s,t)\,l\,t_w$	$t_s + [H(s,t) + h - H_k(s,t)]\,l\,t_w$
$T_{one-to-all}$	$(\log p)(t_s + l\,t_w)$	$(h-1+\log p)(t_s + l\,t_w)$
$T_{all-to-all}$	$(\log p)\,t_s + (p-1)\,l\,t_w$	$(h-1+\log p)\,t_s + [p-1+(2^{mh}-1)/(2^m-1)-1]\,l\,t_w$

The time it takes to complete the second stage:

$$T_2 = \sum_{i=1}^{k} (t_s + 2^{mh+i-1}\,l\,t_w)$$

The total time to complete the all-to-all broadcast:

$$T_{all-to-all} = T_1 + T_2 =$$

$$(h-1+\log p)\,t_s + [p-1+(2^{mh}-1)/(2^m-1)-1]\,l\,t_w$$

Theorem 3. *Assume that each node can use only one link at a time and the package switching model is used. The all-to-all broadcast in an MC(k,m) can be done in* $(h-1+\log p)\,t_s + [p-1+(2^{mh}-1)/(2^m-1)-1]\,l\,t_w$ *time, where* $h = 2^k$.

The communication times of our routing algorithms for one-to-one, one-to-all, and all-to-all broadcast in metacube are summarized and compared to that of the hypercube in Table 1.

4. Conclusion and future work

In this paper, we showed that routing and broadcasting can be done efficiently in the metacube. The results showed that metacube has tremendous potential to be used as an interconnection network for very large scale parallel computers since it can connect hundreds of millions of nodes with up to 6 links per node and the routing and multinode broadcasting can be done almost as efficient as in hypercube. To make the metacube could be of practical use to academia and industry, some issues concerning the metacube listed below are worth further research.

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.

References

[1] B. Vanvoorst, S. Seidel, and E. Barscz. Workload of an ipsc/860. In *Proceedings of the Scalable High-Performance Computing Conference*, pages 221–228, 1994.

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

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

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

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

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

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

[8] 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.

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

[10] Y. Li, S. Peng, and W. Chu. Metacube – a new interconnection network for large scale parallel systems. *Australian Computer Science Communications*, 24(3):29–36, Jan. 2002.

[11] Y. Li and S. Peng. Dual-cubes: a new interconnection network for high-performance computer clusters. In *Proceedings of the 2000 International Computer Symposium, Workshop on Computer Architecture*, pages 51–57, December 2000.

[12] Y. Li and S. Peng. 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*, pages 315–322. IEEE Computer Society Press, June 2001.

[13] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to parallel computing: design and analysis of algorithms*. Benjamin/Cummings Press, 1994.