A Near-Optimal Broadcasting Algorithm in All-Port Wormhole-Routed Hypercubes

Chien-Min Wang and Chiu-Yu Ku

Institute of Information Science
Academia Sinica
Nankang, Taipei, Taiwan

Abstract-For massively parallel computing mechanism, broadcasting is widely used in a variety of applications. When the computation is distributed among the powerful processors, communication overhead always limits the speedup. To reduce the influence of communication latency, a variety of routing methods have been discussed. In this paper, we propose an efficient routing method of broadcasting on all-port wormhole-routed hybercubes. By exploiting the distance-insensitivity of wormhole switching, this efficient broadcasting algorithm reduce communication latency to $\lceil n/\lfloor \log_2(n+1) \rfloor \rceil$ steps in an n-dimensional hypercube, which is much better than previous results and very close to the theoretical optimum, $\lceil n/\log_2(n+1) \rceil$. Further, by adopting the Hamming code, a well-known algebraic approach in coding theory, this proposed algorithm is contention-free in each broadcasting step.

Keywords: all-port communication, broadcasting, Hamming code, hypercube, wormhole-routing.

1. Introduction

Massively Parallel Computing (MPC) has been a widely-discussed topic on computation-intensive application. Characterized by distributing memory among an ensemble of processors, MPC nodes communicate with each other by sending messages through the network. Because the communication time might significantly affect the execution time of an application, minimizing network latency is critical to the performance of MPC. Among all kinds of the communication traffic, one of the most fundamental communication routines is *broadcasting*, in which the same message is delivered from a source node to all nodes in the network. Not only useful in message-passing application, but this kind of communication routine is also necessary in several other operations, such as replication and barrier synchronization[9], which are supported in data parallel languages.

ICS '95 Barcelona, Spain © 1995 ACM 0-89791-726-6/95/0007..$3.50

Among all MPC architectures, hypercube remains popular for its symmetry, regularity, and embedbility of many other topologies[6]. In an n-dimensional hypercube there are 2^n processors with addresses ranged from 0 to 2^n-1. Any two processors, u and v, are directly connected (i.e. neighbors) if and only if these two binary numbers, u and v, differ in exactly one bit. By using Gray code coding, topologies like ring, mesh, etc., could be embedded into the hypercube space. As now several parallel machines are built in this topology[7] and are scalable with hardware upgrade. With better routing algorithms and hardware improvement, the potential of hypercube in the future is still encouraging.

Previous works on deterministic routing methods, including the *Spanning Binomial Tree* (SBT) by Johnson and Ho[3] and the *Double-Tree* (DT) algorithm by Mckinley *et al*[4], require $O(n)$ steps to complete broadcasting in an n-dimensional hypercube. The standard spanning binomial tree method requires n message-passing steps to reach the farthest node from the source in an n-dimensional hypercube by using the nearest neighbor communication. Afterward the predominant switching technique, *wormhole routing*[1], is proposed. Here each message is divided into numbers of flits. The header flit governs the route while the following flits of the message are pipelined through the network by way of routers at each node. The communication latency of wormhole routing is $(L_f/B)D+L/B$ where D is the number of nodes on the path, L_f is the length of each flit, and L/B is the time required for message of length L to pass through the channels of bandwidth B[5]. Hence, the effect of path length on communication latency can be ignored if the length of the message is relatively large. By exploiting the property of distance insensitivity and relaxing the assumption that all messages must be sent between neighboring nodes, the *Double-Tree* algorithm reduces the broadcasting steps to $\lceil n/2 \rceil$[4].

The characteristic of distance insensitivity of wormhole routing was introduced in another survey by Tsai *et al*[8]. It proposed a dominating-set model for broadcasting in all-port wormhole-routed 2D mesh networks. The *all-port* model possesses several pairs of *internal* and *external* channels as shown in Fig.1, thus it can send and receive messages to and from all its neighbors at the same time. Their algorithm arranges broadcasting nodes by levels in

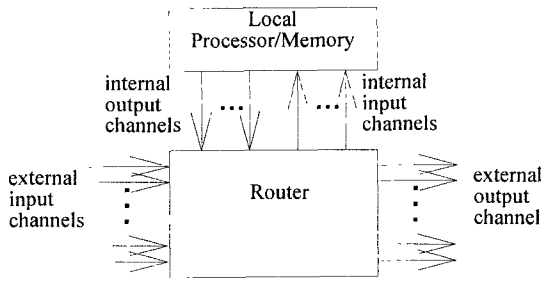

Fig.1. All-port node architecture

the 2D mesh and thereby reduced broadcasting steps to $\lceil \log_5 n \rceil$, which is minimal for the 2D mesh topology.

In this paper, we will focus the routing method on the model of n-dimensional all-port wormhole-routed hypercubes. Our method is intrigued from the idea of constructing an optimal broadcasting tree by regarding the hypercube as an $(n+1)$-ary tree in which each branch or leave node actually represents a subcube. As broadcasting goes down the broadcasting tree, it makes more and more branches receive the messages. In order to prevent message competing for the same resources (channels), Hamming code is adopted in specifying destinations. Hamming code, used as an error detecting and correcting scheme, secures any *codewords* by keeping them in a distance larger than or equal to three. This property adopted in specifying destinations' addresses will ensure the used paths/channels of each step are arc-disjoint. With a proper arrangement of destinations by levels in the optimal broadcasting tree, the broadcasting operation with $\lceil n/\lfloor \log_2(n+1) \rfloor \rceil$ steps is free from contention.

This paper is organized as follows. Section 2 presents the notations, definitions and theories that state the framework of our routing method. Next in section 3, we will describe the basic concept of the minimal-step broadcasting tree and then propose an algorithm to prevent channel contention. Section 4 gives the conclusions of this paper.

2. Preliminaries

An n-dimensional hypercube is consisted of 2^n nodes constructed as follows: all 2^n nodes are addressed distinctly by n-bit binary numbers, $b_{n-1}b_{n-2}...b_1b_0$, from 0 to 2^n-1 and links are directly connected in between if and only if their addresses differ exactly one bit. The *Hamming distance* of nodes u and v, denoted as $\Psi(u,v)$, is equal to the distance (number of links) between them. Nodes u and v are said to be *neighbors* if $\Psi(u,v)=1$.

For any two neighboring nodes, u and v, addressed by binary digits $b_{n-1}...b_i...b_1b_0$ and $b_{n-1}...\overline{b_i}..b_1b_0$, respectively, traverse from u to v through the direct link is denoted as: $b_{n-1}...b_i...b_1b_0 \xrightarrow{b_{n-1}..b_{i+1}\overline{b_i}b_{i-1}..b_1b_0} b_{n-1}...\overline{b_i}..b_1b_0$, and the direct link is perceived as of the ith-dimensional channel in the hypercube space. Here $\overline{b_i}$ is the complement of b_i.

Definition 1: For any given k-bit binary number P, $\{ p \mid$ all k-bit binary number, p, satisfy $\Psi(P,p) = 1\}$ is called the *neighboring set* of P and denoted as $\underline{P}$.

Since we partition addressing bits into blocks to ease the description of our routing algorithm, we will use italic capitals (e.g. A, P, and R) to stand for each partition of addressing bits. For any nodes in an n-dimensional hypercube, addressed as APR, and any of its *neighboring nodes* addressed as $A\underline{P}R$, traverse from APR to $A\underline{P}R$ through the direct link within dimensions of partition P is denoted as: $APR \xrightarrow{A-R} A\underline{P}R$, where A,P and R (i.e. partitions of the n-bit address) stand for $(b_{n-1}b_{n-2}...b_{n-a})$, $(b_{n-a-1}...b_{n-a-p})$, and $(b_{n-a-p-1}...b_1b_0)$, respectively. In general, for message routed from APR to $AP'R$ through channels within dimensions of partition P, it can be denoted as $APR \xrightarrow{\{A-R\}} AP'R$, here {A-R} denotes the path from APR to $AP'R$, which is composed of channels within dimensions of partition P.

Among all routing algorithms, it is an important issue to deliver messages without contention. We adopt Hamming code in specifying routes and destinations to prevent channel contention in any broadcasting steps. The following will introduce related concept and definitions about Hamming code[2].

Hamming code is an encoding scheme for single-error correction. The underlying theory of the Hamming code is to use m-bit parity checks, called *syndrome*, to give the exactly position of the error. Generally the first parity check covers positions 1, 3, 5, 7, 9,... of the codewords (which has '1' in the last position of their binary representation); the second one covers 2, 3, 6, 7, 10,... (which has '1' in the second lowest position); the next covers 4, 5, 6, 7, 12,...; and so on. All the parity checks which cover their corresponding positions can represent at most 2^m states. They need to represent the state of all positions being correct, plus the location of a single error in the digits. Thus we have the following lemma.

Lemma 1: (*Hamming inequality*) Given a syndrome with p parity checks, it can encode s binary source digits, if s and p satisfy inequality, $2^p \geq s+p+1$.

Fig.2(a) demonstrates the standard Hamming encoding for 4-bit binary digits 1101 using 3-bit syndrome. We can see $2^3 \geq 4+3+1$, thus 3-bit syndrome could be worked out by encoding all digits of number 1011. We arrange parity checks to positions 1, 2 and 4, and have the source digits positioned at 3, 5, 6 and 7. Let this encoding digits be 110_1_ _; those spaces are where the parity checks will go. The first parity check, which go in position 1, is computed over positions 1, 3, 5, and 7. Looking at the digits, we can see that position 1 gets a 0. After computing all the positions of the parity checks, we get the final encoded message is 1100110.

In this paper, the Hamming code function is adopted with proper re-arrangement in the bit positions. The syndrome is placed by the right-hand side to construct the

equivalent code. Previous encoded digits 1100110 has its parity checks, 0, 1, and 0, placed at positions 4, 2, and 1, respectively. Now we re-arrange the encoded digits by appending the parity checks 010 to the right-hand side of the original digits 1101. Thus the Hamming code here becomes 1101010.

Definition 2: The Hamming code of A using p-bit syndrome, $\eta(A,p)$, is the concatenation of A and its p-bit syndrome P, i.e. AP.

Note that p-bit syndrome can encode only s digits according to Lemma 1. If bit number of A, a, is greater than s, the leftmost $(a\text{-}s)$ binary digits of A will not be used in calculating the syndrome P. Otherwise, $(s\text{-}a)$ binary zeros are temperately concatenated by the left-hand side of A first before calculating syndrome P. Also note that, by definition, $\eta(A,1)=A$ 0.

Fig.2(b) is an example to encode 101 with 2-bit syndrome. Since the maximal number of s in inequality, $2^2 \geq s+2+1$, is 1, only a source digit is used in calculating the 2-bit syndrome. The syndrome for digit 1 is 11. We have the Hamming code of 101 with 2-bit syndrome, i.e. $\eta(101,2)$, equal to 10111.

Definition 3: The *recursive Hamming code* of A with multi-level syndromes each of which contain p_1, p_2,...,p_j bits, denoted as $\eta(A, p_1, p_2, p_3, ..., p_j)$, is defined as

$$\eta(A, p_1, p_2, ..., p_j) = \eta(\eta(A, p_1, p_2, ... p_{j-1}), p_j);$$

From the equality, we have

$$\begin{aligned}\eta(A, p_1, p_2, ..., p_j) &= \eta(\eta(...\eta(AP_1, p_2), ...), p_j) \\ &= ..... \\ &= \eta(AP_1P_2...P_{j-1}, p_j) \\ &= AP_1P_2P_3 . P_j;\end{aligned}$$

Here P_i is the syndrome of $AP_1P_2..P_{i-1}$, $1 \leq i \leq j$. Also $P_1P_2P_3..P_j$ is called *recursive Hamming syndrome.*

Next are two examples of recursive Hamming code. In Fig.2(c), we encode 101 with two-level syndromes which has three bits each. Therefore we can calculate $\eta(101,3,3)$

encode {1101|xxx}

3-bit syndrome, xxx, can encode 4-bit digits

7 6 5 4 3 2 1	Position
1 1 0 x 1 x x	Encoding message
1 1 0 0 1 1 0	Encoded message
_ _ _ 0 _ 1 0	Syndrome

=> syndrome 010
=> Hamming code 1101|010

(a) encode 1101 with 3-bit syndrome

encode {101|xx}

2-bit syndrome, xx, can only encode 1-bit digit The leftmost 2 bits, 10, are ignored

3 2 1	Position
1 x x	Encoding message
1 1 1	Encoded message
_ 1 1	Syndrome

=> syndrome 11
=> Hamming code 101|11

(b) encode 101 with 2-bit syndrome

recursively encode {101|xxx|yyy}

3-bit syndrome, xxx, can encode 4-bit digits. A bit 0 is attached to the left-hand side of 101

7 6 5 4 3 2 1	Position
0 1 0 x 1 x x	Encoding message
0 1 0 1 1 0 1	Encoded message
_ _ _ 1 _ 0 1	1st-level syndrome

=> syndrome 101
=> Hamming code 101|101|yyy

3-bit syndrome, yyy, can encode 4-bit digits The leftmost 2 bits, 10, are ignored

7 6 5 4 3 2 1	Position
1 1 0 y 1 y y	Encoding message
1 1 0 0 1 1 0	Encoded message
_ _ _ 0 _ 1 0	2nd-level syndrome

=> syndrome 010
=> Hamming code 101|101|010

(c) recursively encode 101 with two-level syndromes which has three bits each

recursively encode {000|101|xxx|yy}

3-bit syndrome, xxx, can encode 4-bit digits The leftmost 2bits, 00, are truncated

7 6 5 4 3 2 1	Position
0 1 0 x 1 x x	Encoding message
0 1 0 1 1 0 1	Encoded message
_ _ _ 1 _ 0 1	1st-level syndrome

=> syndrome 101
=> Hamming code 000|101|101|yy

2-bit syndrome, yy, can only encode 1-bit digits 000|101|10 are ignored.

3 2 1	Position
1 y y	Encoding message
1 1 1	Encoded message
_ 1 1	2nd-level syndrome

=> syndrome 11
=> Hamming code 000|101|101|11

(d) encode 000101 with two-level syndromes which have three and two bits, respectively

Fig.2. Examples of Hamming function.

$= \eta(\eta(101,3),3) = \eta(101101,3) = 101101010$. In Fig.2(d), the Hamming code of 000101 with two-level syndromes which contain 3 and 2 bits, respectively, can be calculated by $\eta(000101,3,2) = \eta(000101101,2) = 00010110111$.

Lemma 2: Given any binary number, u and its *neighbor* $v \in \underline{u}$. If $\eta(u,p)=uP_u$ and $\eta(v,p)=vP_v$ where p is large enough to encode full binary digits of u, it can be inferred that $\Psi(P_u,P_v)\geq 2$ and $\Psi(\underline{P_u},P_v)\geq 1$.

Proof: According to the theory of Hamming code, $\Psi(uP_u, vP_v)\geq 3$. Since $\Psi(u, v)=1$, it can be derived that $\Psi(P_u,P_v)\geq 2$. Also knowing that $\Psi(P_u, \underline{P_u})=1$, therefore we have $\Psi(\underline{P_u},P_v)\geq 1$. ♦

In the following section we will present the methodology of applying recursive Hamming code in specifying recipient nodes during the broadcasting operations.

3. The proposed method

Previous research by Mckinley *et al*[4] depicted that his algorithm can reduce communication latency by exploiting the distance insensitivity of wormhole-routing and relaxing the assumption that all messages must be send between neighboring nodes. However their result is far from optimal. In this paper, we propose a new broadcasting method to achieve a nearly optimal solution. In an n-dimensional all-port wormhole-routed hypercube, every processor has n links directly connected to its neighbors. If every node could send messages to as many as n nodes in each broadcasting step, the broadcasting can be completed in only $\lceil n/\log_2(n+1) \rceil$ steps[1], which is the theoretical optimal.

According to the above idea, our fundamental method is to (1) view the original hypercube as a k-ary broadcasting tree instead, where k is chosen as close to $n+1$ as possible and (2) broadcast messages down this k-ary broadcasting tree. Here we draw the concept of *subcube* that any n-dimensional hypercube can be regarded as 2^d $(n\text{-}d)$-dimensional smaller hypercubes. Any node in the broadcasting tree is a hypercube and its branch nodes are its corresponding subcubes. At each broadcasting step, the source node in a hypercube sends a copy of the message to each of its subcubes.

A larger breadth reduces the height of a broadcasting tree, thus inclines reduction in the number of broadcasting steps. Therefore, bounded by value $n+1$, k should be chosen as large as possible. We properly choose $k=2^p$, here $p=\lfloor \log_2(n+1) \rfloor$. The nodes in each level of the broadcasting tree would represent n, $n\text{-}p$, $n\text{-}2p$, ..., $n\text{-}(\lceil n/p \rceil -1)*p$ and 0-dimensional subcubes, respectively, from top level 0 to bottom level $\lceil n/p \rceil$.

As to naming the subcubes as stated above, if all the addresses of nodes in a subcube have leading binary digits $b_{n-1}b_{n-2}...b_{n-p}$ in common, we say that those nodes belong to $b_{n-1}b_{n-2}...b_{n-p}$-subcube. For example, $(n\text{-}p)$-dimensional subcubes in the first level of the broadcasting tree are composed of $b_{n-1}b_{n-2}...b_{n-p}$-subcubes, ranged from 0 to 2^p-1. So are $(n\text{-}2p)$-dimensional $b_{n-1}b_{n-2}...b_{n-2p}$-subcubes in the second level of broadcasting tree, ranged from 0 to $2^{2p}-1$; and so on. Note that addressing bits now are divided into $\lceil n/p \rceil$ p-bit partitions from left to right, each of which represents the index of each level's subcubes. The lowest partition would be possibly consisted of less than p bits, but we will show later that this will not affect the generality of our routing algorithm.

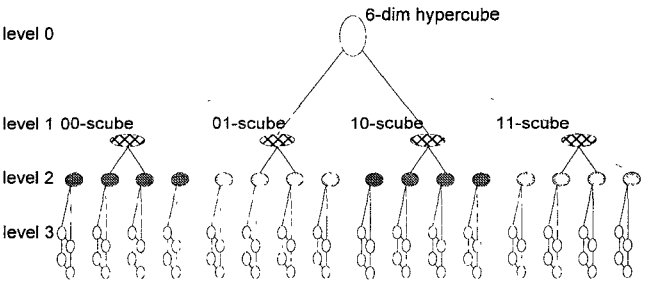

Fig.3. A 4-ary broadcasting tree of 6-cube

After constructing the k-ary broadcasting tree ($k=2^p$), we do the broadcasting as follows. By sending out 2^p-1 messages, a source node could make totally 2^p destination nodes, including itself, get a copy of the message. Then, each recipient will turn into the next-step's source node in each child subcube and be in charge of broadcasting to other 2^p destination nodes. After i steps of broadcasting, the message would reach 2^{i*p} nodes. At the last step (i.e. step $\lceil n/p \rceil$), all source nodes send out $2^{(n-(\lceil n/p \rceil -1)*p)}-1$ messages to complete the broadcasting.

Fig.3 illustrates broadcasting occurred within a 6-dimensional hypercube. The number of messages sent per step is $2^p-1=3$ for $p=\lfloor \log_2(6+1) \rfloor=2$. Hence broadcasting will be undertaken within a three-step broadcasting tree. At the first step, source node 000000 sends out three messages into subcubes (with leading bits 01, 10, and 11) and hence makes four nodes (000000 and three recipients within 01, 10, and 11-subcubes, respectively) get a copy of the message. These four nodes would become next-step's source nodes of the four 4-dim subcubes. Next, each source node in the four subcubes (with leading bits 00, 01, 10, and 11) will send three messages again. It would make totally sixteen nodes within the subcubes (with leading bits 0000, 0001, ..., 1111) get the message. At last, these sixteen nodes within each subcube will send three messages again and make every node get a copy of the message.

From the above example, the framework of broadcasting has been drafted. However, channel contention within each broadcasting step is happened to be resolved. Let us back to the above example again. At the third step of the broadcasting, source node 000000 send messages to three nodes within its 2-dim subcube (with leading bits 0000), i.e. 000001, 000010, and 000011. Fig.4(a) shows that message directed towards 000011 will compete channels with

[1] Let it takes α steps to complete optimal broadcasting, thus α should satisfy $(n+1)^{\alpha} \geq 2^n$. Hence we get $\alpha = \lceil n/\log_2(n+1) \rceil$.

message directed towards 000001 or 000010, unless it is emitted from another source node in other subcube, as shown in Fig.4(b). Actually this situation would happen at any level of the broadcasting tree. To avoid any contention like this, we contribute an alternative broadcasting tree that could adroitly deploy channels in the hypercube space.

A systematic approach using Hamming code is depicted to prevent possible channel contention. Now we illustrate the broadcasting of step i as an example without lost of generality. When the broadcasting undertaken within A-subcube of level i-1, all its child subcubes(say AP_{all}-subcubes, here P_{all} is the set of all numbers ranged from 0 to 2^p-1) will get a copy of the message after the broadcasting. First, every source node of level i-1, which will initiate the step-i broadcasting, always conforms the format of AZ, where

$A=(b_{n-1}b_{n-2}..b_{n-(i-1)*p})$;
$Z=(b_{n-(i-1)*p-1}...b_1b_0)$ is recursive Hamming syndrome of A, computed by $\eta(A, p_1, \ldots, p_{j-1}, p_j)$ with $p_1=....=p_{j-1}=p=\lfloor\log_2(n+1)\rfloor$ and $p_j=n-a-p*(j-1)$, here $j=\lceil(n-a)/p\rceil$ and the bit number of partition A, a, is equal to $(i-1)*p$;

Second, the broadcasting utilizes three types of routes to deliver 2^p-1 messages to the destinations. The routes will also ensure the specified destinations always conform the format as stated above to fit for next step's broadcasting. We regard AZ as APR instead by letting P be the leftmost p bits of partition Z and R be the trailing bits. Now AP_{all}-subcubes' broadcasting are completed by the routes as follows:

$$\boldsymbol{APR} \xrightarrow{A-R} A\underline{P}R \xrightarrow{\{A\underline{P}-\}} \boldsymbol{A\underline{P}R}_1 \quad \ldots\ldots ①$$

$$\boldsymbol{\underline{A}P_2R'} \xrightarrow{-PR'} AP_2R' \xrightarrow{\{AP_2-\}} \boldsymbol{AP_2R_2} \quad \ldots\ldots ②$$

$$\boldsymbol{APR} \xrightarrow{AP-} AP\underline{R} \xrightarrow{\{A-\underline{R}\}} AP_3\underline{R} \xrightarrow{\{AP_3-\}} \boldsymbol{AP_3R_3} \quad \ldots\ldots ③$$

Here P, P_2 are p-bit syndromes of A and $\underline{A}$, respectively. P_3 are p-bit binary digits chosen such that it satisfies $P_3\cap(\{P\}\cup\underline{P}\cup P_2)=\varnothing$. The trailing bit blocks, R, R_1, R_2, R_3 and R' are *recursive Hamming syndromes* of AP, $A\underline{P}$, AP_2, AP_3, and $\underline{A}P_2$, respectively. Terms in italic bold are source and destination nodes, while others in italic are routers of some intermediate nodes which play a key role in the proof of contention-free. The following details how three different types of routes play their roles in the broadcasting.

These rules are classified in the basis of the dimension of the first used link. (Type 1) Crossing direct links within dimensions P, messages from APR are routed to its child subcubes with leading bits equal to $A\underline{P}$. Then they will head to destination $A\underline{P}R_1$ through links in dimensional order. (Type 2) This type of routing is to cooperate with source nodes in other subcubes. For those AP_2-subcubes, the copies of the message are coming from $\underline{A}$-subcubes. Messages emitted from $\underline{A}P_2R'$ in neighboring $\underline{A}$-subcubes cross direct links (within dimensions of A) into AP_2-subcubes. Then the messages head to destinations AP_2R_2 through links in dimensional order. Similarly, source node APR also help broadcasting to its neighboring $\underline{A}$-subcubes. We can ensure that AP_2 is disjoint with $A\underline{P}$ appeared in type 1 according to Lemma 2. Therefore the destination subcubes will not be duplicate in both types. (Type 3) If the broadcasting in both types 1 and 2 could not cover all the child subcubes, the rest subcubes, say AP_3-subcubes, will be handled by broadcasting through lower dimensional channels within dimensions of R. For every P_3, we choose a unique value $\underline{R}$. Thus message emitted from APR will be forced through lower dimensional channel to router $AP\underline{R}$ and then heading to AP_3R_3 in dimensional order. Obviously the routes by our routing methods could be handled by e-cube routing except the first links. This significantly simplifies the implementation of the routing algorithm in the router and make it feasible in practical machines.

Let us take rule 1 as an example. Message emitted from source node APR are specified to traverses neighboring router $A\underline{P}R$ via the direct link first. Next, it traverses the *path*, which consists of links and routers, between $A\underline{P}R$ and $A\underline{P}R_1$ in *dimensional order* and arrives at destination $A\underline{P}R_1$. Every time the destination node, e.g. $A\underline{P}R_1$, receives the message, it will turn into the source node of $A\underline{P}$-subcube at the next step to continue the broadcasting.

Theorem 1: *The broadcasting algorithm would correctly make all child subcubes at every step get one copy of the message.*

Proof: Suppose that the broadcasting is taking place at any source node APR within A-subcube. It is known that partitions A, P and R are a, p, r bits, respectively, where $a+p+r=n$. According to Lemma 1, a p-bit syndrome can encode at most s bits that satisfies $2^p=s+p+1$. Now we want to send out 2^p-1 messages at this step. If $a\geq s$, rules 1 and 2 would have p and s messages emitted, respectively. Since $2^p-1=s+p$, the broadcasting could be handled by only these two rules. Otherwise, i.e. $a<s$, $(s-a)$ messages should be emitted through lower dimensional channels by rule 3. We can show that r dimensions of partition R are enough for routing since $(s-a)=2^p-1-p-a\leq 2^{\lfloor\log_2(n+1)\rfloor}-1-p-a\leq n-p-a=r$. Therefore, we complete the proof of this theorem. ♦

Theorem 2: *The proposed broadcasting algorithm is contention-free in each step.*

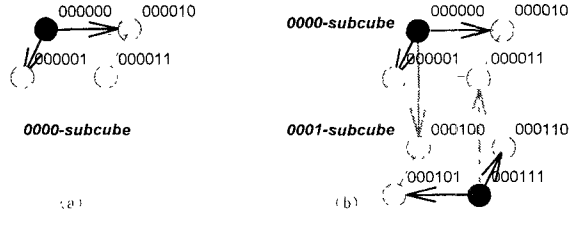

Fig 4 (a) contention in a 2-dim subcube when data routed to 000011 is blocked by message routed to 000010 (b) an alternative route without contention

Proof: To ensure the above routing algorithm is contention-free, we should prove that channels among the routes are different. For message sent from node $b_{n-1}b_{n-2}...b_m...b_1b_0$ to neighbor $b_{n-1}b_{n-2}...\overline{b_m}...b_1b_0$ via the direct link, it means the channel used is of the m-th dimensional channel. Those channels/paths used by the above three rules can be classified by the dimensions they fall into. Consequently, we divide all paths into three disjoint groups, $\{A\text{-}R, \{A\text{-}\underline{R}\}\}$, $\{A\underline{P}\text{-}, \{AP_2\text{-}\}, AP\text{-}, \{AP_3\text{-}\}\}$ and $\{\text{-}P_2R'\}$. Paths/channels in the first set must be arc-disjoint for $Hamming(R, \underline{R}) = 1$. In the second set, since P_2 is the syndrome of $\underline{A}$, $\{P\}$, $\underline{P}$ and P_2 must be disjoint according to Lemma 2. Since P_3 is chosen such that it satisfies $P_3 \cap (\{P\} \cup \underline{P} \cup P_2) = \varnothing$, all channels in this group are mutually disjoint. In the third set, channel - P_2R' between $\underline{A}P_2R'$ and AP_2R' is uniquely used because there is unique corresponding $\underline{A}P_2$ using the channel, where P_2 is the syndrome of $\underline{A}$. Consequently, we can conclude the routing algorithm is contention-free at each broadcasting step within the hypercube. ♦

Theorem 3: *The near-optimal broadcasting algorithm requires* $\lceil n/\lfloor \log_2(n+1) \rfloor \rceil$ *steps to complete the broadcasting.*

Fig.5 illustrates broadcasting in a 4-dimensional hypercube started from node 0000. In our method, the hypercube can be regarded as a two-step broadcasting tree. Fig.5(b) show the first-step broadcasting; By rule 1, copies of the message are forced to cross single link to neighboring nodes 01|00 and 10|00 which belong to 01-subcube and 10-subcube, respectively. Then they are forwarded to the destinations 01|11 and 10|00 by ways within their own subcubes. Rule 2 is not applicable at current level. Rule 3 forces a copy of message crossing lower dimensional link to get another destination, i.e. 11|11. After completing the first step, the recipient nodes start the next-step's broadcasting as shown in Fig.5(c). Say, in 00-scube, source node 00|00 sends messages to 00|01, and 00|10 by rule 1. Meanwhile source node 01|11 would help sending message to 00|11 according to rule 2. Rule 3 is not used at this moment. By now broadcasting within 00-scube is completed.

Fig. 6 compares the performance of our NOB(Near-Optimal Broadcasting) algorithm, two other methods -- the standard SBT algorithm and DT algorithm, and the theoretical optimal value. From Fig. 6, it can be observed that NOB is superior to DT in terms of steps as the dimension of hypercubes is more than six. Also note that the NOB algorithm is optimal in most cases. This makes it be a promising approach for broadcasting.

dimension	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
SBT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DT	1	2	2	2	3	3	4	4	5	5	6	6	7	7	8	8
NOB	1	2	2	2	3	3	3	3	3	4	4	4	5	5	4	4
Optimal	1	2	2	2	2	3	3	3	3	3	4	4	4	4	4	4

Table I. Comparison in terms of steps.

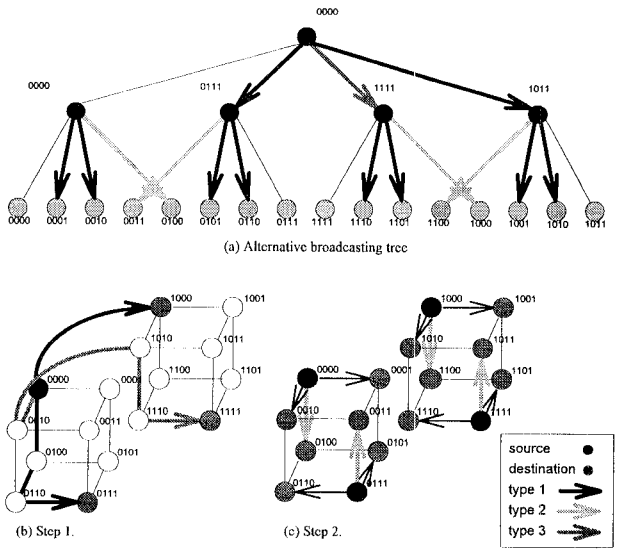

Fig.5 The proposed routes in a 4-dim cube.

Appendix gives the routing program in each node. Since the destination and step index are carried in the header of the message, nodes received the message could know how to follow up the broadcasting. Besides, the Hamming encoding and the destination nodes could be statically computed and stored in the nodes, thus the time complexity takes less effect on the whole operation.

4. Conclusions

In this paper, we adroitly explored a near-optimal method to broadcast messages in an all-port wormhole-routed hypercube. By taking the advantage of the distance insensitivity of wormhole routing and relaxing the assumption that all messages must be sent between neighboring nodes, we have stridden in reducing the number of steps in broadcasting operations nearly to optimal. We have also proved that this routing algorithm is contention-free in each broadcasting step. The exhibited advantage of this method would make it a promising approach to overcome the bottleneck in broadcasting.

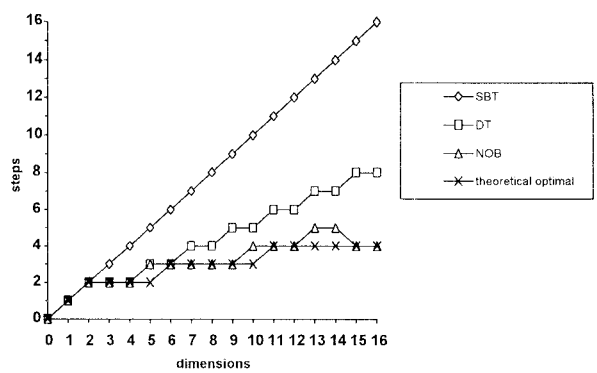

Fig. 6 Performance comparison of three algorithms.

Acknowledgments

The authors wish to thanks the reviewers of the 1995 ACM International Conference on Supercomputing for reading an earlier version of this manuscript and making many suggestions to improve the contents. This work was supported in part by the National Science Council of the Republic of China under Contract NSC 84-2213-E-001-003.

References

1. W. J. DALLY AND C. L. SEITZ, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. on Computers*, vol.C-36, no.5, pp.547-553, 1987.
2. R. W. HAMMING, *Coding and Information Theory*, Chap. 3, Prentice-Hall 1986.
3. S. L. JOHNSON AND C. T. HO, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. on Computers*, vol.C-38, pp.1249-1268, Sept. 1989.
4. P. K. MCKINLEY AND C. TREFFTZ, "Efficient broadcast in all-port wormhole-routed hypercubes," *Proc. of the 1993 International Conference on Parallel Processing*, part II, pp.228-291, 1993.
5. L. M. NI AND P. K. MCKINLEY, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, vol. 26, pp. 62-76, Feb. 1993.
6. S. RANKA AND S. SAHNI, Hypercube Algorithms with Applications to Image Processing and Pattern Recognition, Springer-Verlag New York Inc., 1990.
7. Y. SHIN, AND J. FIER, Hypercube systems and key applications. In K. HWANG, AND D. DEGROOT (Eds.), *Parallel Processing for Supercomputing and Artificial Intelligence*, Chap. 6, McGraw-Hill, New York, 1988.
8. Y. J. TSAI AND P. K. MCKINLEY, "A dominating set model for broadcast in all-port wormhole-routed 2D mesh networks," *Proc. of 1994 Int. Conf. on Supercomputing*, pp. 126-135, July, 1994.
9. H. XU, P. K. MCKINLEY, AND L. M. NI, "Efficient implementation of barrier synchronization in wormhole-routed hypercube multicomputers," *Journal of Parallel Distributed Computing*, vol. 16, pp.172-184, 1992.

Appendix

Algorithm: Node Routing Program
Input: step index i and node *address*.
Output: /* 2^p-1 copies of messages emitted */
Procedure
/* variables a, p and r stand for bit number of partitions
A, P and R, respectively. */
begin
let $p = \lfloor \log_2(n+1) \rfloor$; /* dimension in each level */
if ($i > \lceil n/p \rceil$) **return**;
/* set values of a, p, and r. */
$a = p*(i-1)$;
$p = (i < \lceil n/p \rceil)$? p:$n-a$;
$r = n-a-p$;

/* set parameters of recursive Hamming syndrome */
$j = \lceil (n-a-p)/\lfloor \log_2(n+1) \rfloor \rceil$;
$p_1 = ... = p_{j-1} = \lfloor \log_2(n+1) \rfloor$; $p_j = r-(j-1)*\lfloor \log_2(n+1) \rfloor$;
Partition binary digits address into A, P, and R.

Call Rule 1(*address*, a, p, r, i);
Call Rule 2(*address*, a, p, r, i);
Call Rule 3(*address*, a, p, r, i);
end

Function: Rule 1 /* routing path: $APR \rightarrow A\underline{P}R \Rightarrow A\underline{P}R_1$ */
Parameters: (*address*, a, p, r, i)
begin
for every $\underline{P}$ is neighbor(P) **do**
begin
$R_1 = \eta(A\underline{P}, p_1, p_2, ..., p_j)$;
Construct the message flits with destination address $A\underline{P}R_1$ and step index i+1
Send the message out through the channel between APR and $A\underline{P}R$.
end
end

Function: Rule 2 /* routing path: $APR \rightarrow \underline{A}PR \Rightarrow \underline{A}P\ R_2'$ */
Parameters: (*address*, a, p, r, i)
begin
if ($a = 0 \,||\, p < 2$) **then** return; /* not applicable */
$s = 2^p - p - 1$;
$k = 1$; $s = \min(s, a)$; /* s is bounded by a */
for $\underline{A}$ differ from A at bit k-1 **and** $k \le s$ **do**
begin
$R_2' = \eta(\underline{A}P, p_1, p_2, ..., p_j)$;
Construct the message flits with destination address $\underline{A}P\ R_2'$ and step index i+1.
Send the message out through the channel between APR and $\underline{A}P\ R_2'$.
$k = k+1$;
end
end

Function: Rule 3 /* path. $APR \rightarrow AP\underline{R} \rightarrow AP_3\underline{R} \Rightarrow AP_3R_3$ */
Parameters: (*address*, a, p, r, i)
begin
if (r < 1) **then** return; /* not applicable */
for every P_3 satisfy $P_3 \cap (\{P\} \cup \underline{P} \cup P_2) = \varnothing$ **do**
begin
Choose an unique $\underline{R}$ that is neighbor(R);
$R_3 = \eta(AP_3, p_1, p_2, ..., p_j)$;
Construct the message flits with destination address AP_3R_3 and step index i+1.
Send the message out through the channel between APR and $AP\underline{R}$.
end
end