Efficient Broadcasting in Wormhole-Routed Multicomputers: A Network-Partitioning Approach

Yu-Chee Tseng, *Member, IEEE Computer Society*, San-Yuan Wang,
and Chin-Wen Ho, *Member, IEEE Computer Society*

Abstract—In this paper, a *network-partitioning* approach for one-to-all broadcasting on wormhole-routed networks is proposed. To broadcast a message, the scheme works in three phases. First, a number of *data-distributing networks (DDNs)*, which can work independently, are constructed. Then the message is evenly divided into submessages, each being sent to a representative node in one DDN. Second, the submessages are broadcast on the DDNs concurrently. Finally, a number of *data-collecting networks (DCNs)*, which can work independently too, are constructed. Then, concurrently on each DCN, the submessages are collected and combined into the original message. Our approach, especially designed for wormhole-routed networks, is conceptually similar but fundamentally very different from the traditional approach (e.g., [4], [13], [18], [31]) of using multiple *edge-disjoint spanning trees* in parallel for broadcasting in store-and-forward networks. One interesting issue is on the definition of *independent* DDNs and DCNs, in the sense of wormhole routing. We show how to apply this approach to tori, meshes, and hypercubes. Thorough analyses and comparisons based on different system parameters and configurations are conducted. The results do confirm the advantage of our scheme, under various system parameters and conditions, over other existing broadcasting algorithms.

Index Terms—Collective communication, hypercube, interconnection network, mesh, one-to-all broadcast, parallel processing, torus, wormhole routing.

1 INTRODUCTION

IN a multicomputer network, processors often need to communicate with each other for various reasons, such as data exchange or event synchronization. Efficient communication has been recognized to be critical for high-performance computing. One essential communication operator is the *one-to-all broadcast*, where a source node needs to send a message to every other node in the network. Broadcast has many applications, such as linear algebra algorithms [12], barrier synchronization [34], parallel graph algorithms, parallel matrix algorithms, distributed table-lookup, fast Fourier transformation, and cache coherence. The one-to-all broadcast, together with other operators such as *all-to-all broadcast*, *personalized broadcast*, and *data reduction*, are termed as *collective communication* and have received intensive attention recently [1], [2], [15], [17], [28], [29], [30].

We consider the communication network using *wormhole routing* switching technology [7], [19], which is characterized with low communication latency and is quite insensitive to routing distance in the absence of link contention. Such technology has been adopted by many new-generation parallel machines, such as the Intel Touchstone DELTA [11], Intel Paragon, MIT J-machine [20], Caltech MOSAIC, nCUBE 3 [8], and Cray T3D and T3E [14], [6].

In this paper, a *network-partitioning* approach for one-to-all broadcasting on wormhole-routed networks is proposed. To broadcast a message, the scheme works in three phases, as follows: First, a number of *data-distributing networks (DDNs)*, which can work independently, are constructed. Then, the message is evenly divided into submessages, each being sent to a representative node in one DDN. Second, the submessages are broadcast on the DDNs concurrently. Finally, a number of *data-collecting networks (DCNs)*, which can work independently too, are constructed. Then, concurrently on each DCN, the submessages are collected and combined into the original message. One interesting issue in this approach is on how to define two subnetworks to be *independent* in the sense of wormhole routing—independent networks should be able to work independently without interferences. Formal definitions can be seen in Section 2.

One typical approach to the broadcast problem is to utilize multiple spanning trees in parallel for transmission. For instance, Johnson and Ho [13] show how to use n edge-disjoint spanning trees in an n -cube for various versions of broadcast problems. While Bermond et al. [4] show how to construct two edge-disjoint spanning trees in a 2D torus, Michallon and Trystram [18] further use four disjoint trees to facilitate broadcasting. On the side of star graphs, Tseng and Sheu [31] have used $n - 1$ congestion-2 edge-disjoint spanning trees for broadcasting in an n -star. Note that in all the above work [4], [13], [18], [31], the number of spanning trees is a fixed number, given a fixed network of a fixed dimension. Furthermore, the spanning trees are “real” graphs in

• The authors are with the Department of Computer Science and Information Engineering, National Central University, Chung-Li, 32054, Taiwan.
E-mail: {yctseng, sywang, hocw}@csie.ncu.edu.tw.

Manuscript received 21 Oct. 1996; revised 8 Jan. 1998.

For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 100329.

Fig. 1. A 4×4 torus and the generic node architecture for wormhole routing. The gray nodes and dotted links constitute subnetwork G_1 , while black nodes and solid links constitute subnetwork G_2 .

the sense that, for any edge of a tree, its two endpoints must exist in the tree. Such spanning trees are suitable for *store-and-forward* types of networks, but may not be suitable for wormhole-routed ones.

Specially designed for wormhole-routed networks, our approach is conceptually similar but fundamentally very different from the approach used in [4], [13], [18], [31]. First, the DDNs and DCNs used in our scheme may not be spanning trees. Second, the numbers of DDNs and DCNs are adjustable parameters, which can be used to optimize the performance. Third, a subnetwork may not be a “graph” in standard graph-theoretic terminology—an edge may exist in a subnetwork without both of its endpoints existing. This in fact carries special meaning in a wormhole-routed network, in which a message may pass the router of a node without interfering the computation in the node. Thus, in our work, a network may contain a path of edges with only two nodes existing at the start and end of the path (we call such a network a *dilated network*). Due to the distance-insensitive characteristic of wormhole routing, the transmission along the path is expected to be pretty fast.

We show how to apply our scheme to 2D tori, 2D meshes, and hypercubes. Thorough analyses and comparisons based on different system parameters and configurations are conducted. The results do confirm the advantage of our scheme, under various system parameters and conditions, over other existing broadcasting algorithms (e.g., the U-torus scheme for one-port tori by Robinson et al. [22], the U-mesh scheme for one-port meshes by McKinley et al. [16], the Scatter-Collect and the Edge-Disjoint-Spanning-Fences schemes for one-port tori/meshes by Barnett et al. [3], the Postal model when applied to one-port tori/meshes [5], the dominating-set approaches for all-port meshes/tori by Tsai and McKinley [24], [25], [26], [27], and the schemes for all-port hypercubes by Ho and Kao [10] and Wang and Ku [32]). In particular, in our schemes, because mesh/torus are partitioned into DDNs that are dilated mesh/torus, broadcasting on these DDNs can be easily done by directly applying existing broadcasting algorithms [3], [16], [22], [26], [27] for ordinary mesh/torus. Following most standard terminology in the literature [19], we divide the communication latency into two parts: startup cost (for initializing a communication on a communication link) and

transmission cost (for sending data on the links). The latter is further divided into the cost for the header flit (which needs the router to make a routing decision) and that for the follow-up flits (which do not). For a quick overview and comparison, based on these three factors, of existing and our algorithms, refer to Table 1, Table 2, Table 4, Table 5, Table 6, Table 7, Fig. 8, Fig. 14, Fig. 15, Fig. 16, and Fig. 18. As can be observed, under one-port model, the U-torus, U-mesh, and Postal schemes perform better with fairly small messages; the Scatter-Collect and Edge-Disjoint-Spanning-Fences schemes are better when broadcast message is impractically large; our network-partitioning schemes are useful when the broadcast message is of a reasonable size. Under all-port model, the dominating-set approaches (for meshes/tori) and Ho-Kao scheme (for hypercubes) perform better than ours at short messages, but are worse than ours at larger messages. In addition, we also use an nCUBE/2 to emulate a torus and conduct some experiments. More details are in Section 3.1.5 and the obtained results do conform to our analysis.

The rest of this paper is organized as follows. Preliminaries are given in Section 2. Sections 3 and 4 present our routing algorithms for 2D tori and meshes, respectively, under various system configurations. Our algorithms for hypercubes are presented in Section 5. Finally, conclusions are drawn in Section 6.

2 PRELIMINARIES

2.1 System Model

In a wormhole-routed network, each node contains a separate *router* to handle the communication tasks. The architecture of a generic node is shown in Fig. 1 [19]. Each router supports some numbers of *internal channels* (connecting to the local processor/memory) and *external channels* (connecting to other routers). Each of these channels consists of a pair of *input* and *output channels*. From the connectivity between routers, we can define the topology of a wormhole-routed network as $G = (V, C)$, where V is the node set and C specifies the channel connectivity. Throughout this paper, we assume that the channel connectivity is *bi-directional*, in the sense that if there is a connection from router x to router y , then the reverse connection also exists.

Fig. 2. (a) 4×4 mesh network G , (b) a subnetwork G_1 of G , and (c) a subnetwork G_2 of G .

Both the *one-port* and *all-port* models will be considered in this paper, where under the former model a node can only send and simultaneously receive one message at a time, while under the latter a node can simultaneously send and receive messages along all channels. Reflecting on the architecture in Fig. 1, the router will have one (four) *internal* channels under the one-port (all-port) model. We note that these models, however, do not limit the routing capability of routers—a router can still concurrently send and receive flits along all *external* channels.

In a wormhole-routed network, a message is partitioned into *flits*. The *header* flit governs the routing, and the remaining flits follow the header in a pipelined fashion. In the contention-free case, the communication latency for sending a message of L bytes from a source node to a destination node with distance d can be formulated as $T_s + d \frac{L_f}{B} + \frac{L}{B}$, where T_s is the startup time, L_f is the length of each flit, and B is the bandwidth of the channel. In this paper, as a convention, the broadcast latency of an algorithm will be written in three factors as $\alpha T_s + \beta T_f + \gamma L T_c$, where $T_f = \frac{L_f}{B}$ can be regarded as the cost to transmit a header flit, and $T_c = 1/B$ as that for a nonheader flit. We will compare the α , β , and γ values of our algorithm against others.

Throughout the paper, we will consider the network G as mesh, torus, or hypercube. Routing on these networks are assumed to be the commonly used *dimensional-ordered* routing [7], [19]. For example, the unicast path shown in Fig. 1 follows such routing.

2.2 Independent Subnetworks

Given a wormhole-routed network $G = (V, C)$, a subnetwork $G' = (V', C')$ of G is one such that $V' \subseteq V$ and $C' \subseteq C$. However, a subnetwork is not necessarily a “graph” in standard graph-theoretic terms. Specifically, suppose channel $(x, y) \in C'$. Then the vertices x and y incident by c are not necessarily in the vertex set V' . This carries special meanings to us: In subnetwork G' , the local processors of x and y are not allowed to send and receive messages in G' , but the routers in them can help propagating flits in G' . That is, worms can pass through x and y , but should not be initiated from or destined to x and y . This leads to the following definitions:

DEFINITION 1. Two subnetworks $G_1 = (V_1, C_1)$ and $G_2 = (V_2, C_2)$ of G are said to be independent under the one-port model if $V_1 \cap V_2 = \emptyset$ and $C_1 \cap C_2 = \emptyset$.

DEFINITION 2. Two subnetworks $G_1 = (V_1, C_1)$ and $G_2 = (V_2, C_2)$ of G are said to be independent under the all-port model if $C_1 \cap C_2 = \emptyset$.

There is some subtlety in the above definitions. For instance, Fig. 1 contains two subnetworks G_1 and G_2 , which are independent under both one-port and all-port models. However, the two subnetworks G_1 and G_2 of network G shown in Fig. 2 are only independent under the all-port model, but not one-port model, because they share common nodes. Hence, one can simultaneously use G_1 and G_2 to perform communication safely under the all-port model, but not so under the one-port model.

2.3 A General Broadcasting Scheme

Given any network G , here we propose a general broadcasting scheme. From G we first construct two kinds of subnetworks: *data-distributing networks* (DDNs) and *data-collecting networks* (DCNs). Suppose we have h DDNs, $DDN_0, DDN_1, \dots, DDN_{h-1}$, and k DCNs, $DCN_0, DCN_1, \dots, DCN_{k-1}$. We require the following properties in our model:

- P1.** $DDN_0, DDN_1, \dots, DDN_{h-1}$ are mutually independent (under the given port model).
- P2.** $DCN_0, DCN_1, \dots, DCN_{k-1}$ are mutually independent (under the given port model) and they together contain all nodes of G .
- P3.** DDN_i and DCN_j intersect in at least one node, for all $0 \leq i < h$ and $0 \leq j < k$.

With the above properties, our broadcast scheme works in three phases as follows. Here, the source node is x and the broadcast message is M .

- PHASE 1.** Node x evenly partitions M into h submessages $M_0, M_1, \dots, M_{h-1}$ and distributes each M_i to one representative node r_i of network DDN_i , $i = 0..h-1$.
- PHASE 2.** Concurrently in each DDN_i , node r_i broadcasts the submessage M_i to the rest of the nodes in DDN_i .
- PHASE 3.** Concurrently in each network DCN_i , $i = 0..k-1$, each node collects submessages $M_0, M_1, \dots, M_{h-1}$ from those that have received submessages in Phase 2 and combines them into the original M .

Clearly, **P1** and **P2** ensure the concurrent execution in Phases 2 and 3, respectively. The correctness of broadcast is guaranteed by **P2** and **P3**. The following properties are not a necessity, but would offer regularity in designing Phases 2 and 3.

Fig. 3. Independent subnetworks in a 16×16 torus (under both one- and all-port models): (a) four dilated-4 undirected 4×4 tori, and (b) eight dilated-4 directed 4×4 tori.

P4. $DDN_0, DDN_1, \dots, DDN_{h-1}$ are isomorphic.

P5. $DCN_0, DCN_1, \dots, DCN_{h-1}$ are isomorphic.

In the following three sections, we discuss how to define the DDNs and DCNs in tori, meshes, and hypercubes.

2.4 Independent Subnetworks in Tori

A 2D torus $T_{s \times t}$ consists of $s \times t$ nodes each denoted as $p_{x,y}$ where $0 \leq x < s$ and $0 \leq y < t$. A node p_{x_0,y_0} is connected to p_{x_1,y_1} if and only if either $x_1 = (x_0 \pm 1) \bmod s$ and $y_1 = y_0$, or $y_1 = (y_0 \pm 1) \bmod t$ and $x_1 = x_0$. (Hereafter, we will omit saying “mod” whenever the context is clear.) The torus is arranged on a plane as shown in Fig. 3 and we will use “row” and “column” as in standard algebra to refer to a set of components in the network.

DEFINITION 3. Given a torus $T_{s \times t}$ and any integer h that divides both s and t , the data distribution network $DDN_k = (V_k, C_k)$, where $0 \leq k < h$, is defined as follows:

$$V_k = \{p_{x,y} \mid x = ah + k, y = bh + k, \text{ for all } a = 0..(s/h) - 1 \text{ and } b = 0..(t/h) - 1\}$$

$$C_k = \{\text{all channels at rows } ah + k \text{ and at columns } bh + k\}.$$

Intuitively, each DDN is a “dilated- h ” torus of size $(s/h) \times (t/h)$, in the sense that each edge is dilated by a path of h edges. An example is shown in Fig. 3a with four dilated-4 4×4 tori embedded in a 16×16 torus. One can easily verify

that the above defined h DDNs are mutual independent under both one-port and all-port models.

DEFINITION 4. Given a torus $T_{s \times t}$ and an integer h which divides both s and t , the data collecting network $DCN_{a,b} = (V_{a,b}, C_{a,b})$, $0 \leq a < s/h$ and $0 \leq b < t/h$, consists of vertex set $V_{a,b} = \{p_{x,y} \mid x = a \times h + i, y = b \times h + j \text{ for all } i, j = 0..h - 1 \text{ and channel set } C_{a,b} \text{ which is defined to be the set of edges induced by } V_{a,b} \text{ in } T_{s \times t}$.

Intuitively, these DDNs are obtained by evenly slicing the torus into st/h^2 blocks, each being a square $h \times h$ mesh (see the example in Fig. 3a). We can also show that each DDN_k must intersect with each $DCN_{a,b}$ by exactly one node. So these DDNs and DCNs satisfy all properties **P1-P5**.

With brief observation on Fig. 3a, one can easily devise eight new (directed) DDNs by partitioning each (undirected) DDNs further into two directed subnetworks, one using only positive channels (i.e., in positive- x and positive- y directions), and the other negative channels only (i.e., in negative- x and negative- y directions). Now, each directed DDN is a directed torus. However, these eight DDNs are only mutually independent under the all-port model since some DDNs share common nodes. The following definition shows how to conquer this problem:

DEFINITION 5. Given a torus $T_{s \times t}$ and any integer h that divides both s and t , the positive data distribution network $DDN_k^+ = (V_k^+, C_k^+)$, where $0 \leq k < h$, is defined as follows:

$$V_k^+ = \{p_{x,y} \mid x = ah + k, y = bh + k, \text{ for all } a = 0..(s/h) - 1 \\ \text{and } b = 0..(t/h) - 1\}$$

$$C_k^+ = \{\text{all positive channels at row } ag + k \text{ and columns } bh + k\}, \\ \text{and the negative data distribution network}$$

$$DDN_k^- = (V_k^-, C_k^-)$$

is defined as follows:

$$V_k^- = \{p_{x,y} \mid x = ah + k + 1, y = bh + k, \text{ for all } a = 0..(s/h) - 1 \\ \text{and } b = 0..(t/h) - 1\}$$

$$C_k^- = \{\text{all negative channels at row } ag + k \\ \text{and columns } bh + k + 1\}.$$

Intuitively, DDN_k^+ is the same as DDN_k defined earlier except that it only uses the positive channels, and DDN_k^- is obtained by cyclically shifting DDN_k to the right by one position and it only uses negative channels. Each DDN is now a dilated- h directed $(s/h) \times (t/h)$ torus. An example is shown in Fig. 3b with eight dilated-4 directed 4×4 tori embedded in a 16×16 torus (for visual clarity, only typical channels are shown). Clearly, these positive and negative DDNs are isomorphic and mutually independent under both one-port and all-port models.

2.5 Independent Subnetworks in Meshes

A 2D $s \times t$ mesh, denoted as $M_{s \times t}$, is defined similar to $T_{s \times t}$ except that there are no wrapped-around links. The same notations used for tori will be used here, too.

One fundamental difference between meshes and tori is that meshes are not node-symmetric. Thus, it is easy to “shift” a DDN in a torus to any desired location. However, this is less obvious for the case of meshes. The following definition provides a way to find a set of DDNs *with respect to* a given mesh node.

DEFINITION 6. Given a mesh $M_{s \times t}$, a node $p_{i,j}$ in the mesh, and any integer h that divides both s and t , the data-distribution network $DDN_k = (V_k^-, C_k^-)$ with respect to $p_{i,j}$, $0 \leq k < h$, is defined as follows:

$$V_k = \{p_{x,y} \mid x = (ah + k + i) \bmod s, y = (bh + k + j) \bmod t, \\ \text{for all } a = 0..(s/h) - 1, b = 0..(t/h) - 1\}$$

$$C_k = \{\text{all channels at rows } (ah + k + j) \bmod s \text{ and at columns } \\ (bh + k + i) \bmod t\}.$$

Intuitively, these DDNs are obtained from the DDNs of a torus (in Definition 3) by shifting them horizontally and vertically by i and j positions, respectively. Each DDN is a dilated- h mesh of size $(s/h) \times (t/h)$. For instance, the DDNs in Fig. 4 are defined with respect to node $p_{3,1}$. These DDNs are isomorphic and mutually independent. It is also straightforward to extend Definition 5 to obtain $2h$ independent directed DDNs; we leave this part to the reader.

Fig. 4. Four dilated-4 independent 4×4 mesh in a 16×16 mesh defined with respect to node $p_{3,1}$.

The DCNs are defined exactly the same as in tori (Definition 4). Each DDN still intersects with any DCN at one node. So properties **P1–P5** still hold true under our definitions.

2.6 Independent Subnetworks in Hypercubes

A binary n -cube is an undirected graph with 2^n nodes each labeled with a distinct binary string $b_1 b_2 \dots b_n$. Node $b_1 \dots b_i \dots b_n$ and node $b_1 \dots \bar{b}_i \dots b_n$ are joined by an edge *along* dimension i , where $\bar{b}_i$ is the one's complement of b_i . It is well-known that an n -cube has a recursive structure that can be partitioned into many *subcubes*. A subcube of dimension $d \leq n$ can be denoted by a ternary string $x_1 x_2 \dots x_n$, where $x_i \in \{0, 1, *\}$, with exactly d $*$ s, where $*$ means a “don't-care.” This subcube consists of all nodes obtained from $x_1 x_2 \dots x_n$ by arbitrarily replacing each $*$ -symbol with 0 or 1. For instance, $*01*0$ is a 2-cube in a 5-cube consisting of nodes $\{00100, 00110, 10100, 10110\}$.

DEFINITION 7. Given an n -cube and an integer d , $1 \leq d \leq n$, we define 2^{n-d} DDNs each being an $(n-d)$ -cube of the form $b_1 b_2 \dots b_d *^{n-d}$ ($*^{n-d}$ denotes a sequence of $n-d$ $*$ s). Also, we define 2^{n-d} DCNs, each being a d -cube of the format $*^d a_{d+1} a_{d+2} \dots a_n$.

LEMMA 1. For any pair of DDN and DCN defined in Definition 7, they intersect in exactly one node.

So properties **P1–P5** all hold true.

3 ONE-TO-ALL BROADCAST IN A TORUS

In this section, we apply the DDNs and DCNs of a torus defined in Section 2.4 to our general broadcast scheme. We

Fig. 5. Broadcasting in a one-port torus: (a) Phase 1 happening in the upper-left 4×4 submesh of the torus, (b) row-broadcasting of Phase 3, and (c) column-collecting of Phase 3.

consider both one-port and all-port communication models. Although our scheme can be applied to nonsquare tori, below we develop the scheme based on a torus $T_{2^n \times 2^n}$. This is solely for the purpose of simplifying the analysis and comparison part. So the possible value of h (which will define the numbers of DDNs and DCNs) is $h = 2^d$, $d = 1..n$. Without loss of generality, we let the source node be $p_{0,0}$. The message to be broadcast is M of length L bytes.

3.1 One-Port Model

Below, we first develop our schemes based on the (undirected) DDNs defined in Definition 3. Then we will discuss the alternative if we use the (directed) DDNs in Definition 5.

3.1.1 Phase 1: Scattering to Representative Nodes of Undirected DDNs

There are h undirected DDNs: $DDN_0, DDN_1, \dots, DDN_{h-1}$. We can let $r_i = p_{i,i}$ be the representative node of DDN_i , $i = 1..h-1$. So the source node $p_{0,0}$ should evenly partition M into submessages $M_0, M_2, \dots, M_{h-1}$ and then distribute each M_i to r_i . This can be done efficiently by a simple recursive doubling as follows. Let $r_0 = p_{0,0}$. Node r_0 first sends submessages $M_{h/2}, M_{h/2+1}, \dots, M_{h-1}$ to $r_{h/2}$ using any shortest path. Then, r_0 and $r_{h/2}$ each plays as the source of $M_0, M_1, \dots, M_{h/2-1}$ and $M_{h/2}, M_{h/2+1}, \dots, M_{h-1}$, respectively, and recursively distributes these submessages to nodes $r_0, r_1, \dots, r_{h/2-1}$ and nodes $r_{h/2}, r_{h/2+1}, \dots, r_{h-1}$. Fig. 5a illustrates this process in the upper-left 4×4 submesh of the torus when $h = 4$. The execution time of this phase is

$$T_1 = \sum_{i=1}^d \left(T_s + 2(2^{d-i})T_f + \frac{L}{2^i}T_c \right) \\ = dT_s + 2(2^d - 1)T_f + \left(1 - \frac{1}{2^d}\right)LT_c. \quad (1)$$

3.1.2 Phase 2: Broadcasting in Undirected DDNs

In Phase 2, every representative node r_i , $i = 0..h-1$, should broadcast submessage M_i on DDN_i in parallel. Recall that each of our DDNs is in fact a dilated- h tori. So, any existing broadcasting algorithm for tori should be able to apply to our DDNs. In the literature, there are three such schemes: recursive doubling (RD) [22], scatter-collect (SC) [3], and edge-disjoint spanning fences (EDSF) [3]. Below, we will

introduce these alternatives and analyze the costs when applying them to our dilated DDNs. Our analyses do take the dilation of our DDNs into consideration, though it only causes a little penalty due to the distance-insensitive characteristic of wormhole routing.

Our first alternative, the RD scheme [22], was originally designed for multicasting on a torus, but can be used for broadcasting here. Recursive doubling is in fact a commonly used technique in parallel processing and, thus, we omit the details. In an ordinary (undilated) $2^n \times 2^n$ torus, the RD scheme takes time

$$T_{RD}^t = 2 \sum_{i=1}^n (T_s + 2^{n-i}T_f + LT_c) = 2nT_s + (2^{n+1} - 2)T_f + 2nLT_c.$$

Applied to our dilated- h DDNs (of size $2^{n-d} \times 2^{n-d}$), it should be rewritten as

$$T_2^{RD} = 2(n-d)T_s + (2^{n+1} - 2^{d+1})T_f + \frac{2(n-d)}{2^d}LT_c. \quad (2)$$

Our second alternative is the SC scheme [3]. In a $2^n \times 2^n$ torus, the scheme works in four stages as follows:

- 1) *Column Scattering*: The source node slices the broadcast message evenly into 2^n submessages and, then, scatters them across the column where the source resides;
- 2) *Row Scattering*: Each node receiving a submessage in step 1 further slices the submessage into 2^n smaller submessages and scatters them across the row where it resides;
- 3) *Row Collecting*: Each row independently forms a logical ring and every node circulates the submessage it received in step 2 around the ring; and
- 4) *Column Collecting*: Each column independently forms a logical ring and every node circulates the submessage it received in step 3 around the ring.

In an ordinary $2^n \times 2^n$ torus, the SC algorithm takes time [3]:

$$T_{SC}^t = \sum_{i=1}^n \left(T_s + 2^{n-i}T_f + \frac{1}{2^i}LT_c \right) + \sum_{i=1}^n \left(T_s + 2^{n-i}T_f + \frac{1}{2^{i+n}}LT_c \right) \\ + \sum_{i=1}^{2^n-1} \left(T_s + T_f + \frac{1}{2^{2n}}LT_c \right) + \sum_{i=1}^{2^n-1} \left(T_s + T_f + \frac{1}{2^n}LT_c \right) \\ = (2^{n+1} + 2n - 2)T_s + (2^{n+2} - 4)T_f + 2 \left(1 - \frac{1}{2^{2n}}\right)LT_c.$$

In our dilated- h DDNs (of size $2^{n-d} \times 2^{n-d}$), the cost is

$$T_2^{SC} = 2(2^{n-d} + n - d - 1)T_s + (2^{n+2} - 2^{d+2})T_f \\ + 2\left(\frac{1}{2^d} - \frac{1}{2^{2n-d}}\right)LT_c.$$

Our third alternative is the EDSF scheme [3]. First, two edge-disjoint (but not node-disjoint) spanning fences are constructed from the torus.¹ The source node then partitions the broadcast message into k_1 submessages and *alternately* injects them into the two fences. The way submessages are injected will ensure that no node needs to propagate submessages for both fences at the same step (and, thus, the one-port assumption is not violated).

The execution time of the EDSF scheme on an ordinary $T_{2^n \times 2^n}$ is

$$T_{EDSF}^t = (k_1 + 2^n + 2^n) \left(T_s + T_f + \frac{L}{k_1} T_c \right),$$

where k_1 should be determined for best performance as follows:

$$k_1 = \begin{cases} 1 & \text{if } \sqrt{2^{n+1} LT_c / (T_s + T_f)} < 1 \\ L & \text{if } \sqrt{2^{n+1} LT_c / (T_s + T_f)} > L. \\ \sqrt{2^{n+1} LT_c / (T_s + T_f)} & \text{otherwise} \end{cases} \quad (3)$$

In our dilated DDNs, the T_{EDSF}^t should be translated into

$$T_2^{EDSF} = (k_2 + 2^{n-d} + 2^{n-d}) \left(T_s + 2^d T_f + \frac{L}{2^d k_2} T_c \right),$$

where k_2 , the best number of sub-messages, is

$$k_2 = \begin{cases} 1 & \text{if } \sqrt{2^{n-2d+1} LT_c / (T_s + 2^d T_f)} < 1 \\ L/2^d & \text{if } \sqrt{2^{n-2d+1} LT_c / (T_s + 2^d T_f)} < L/2^d. \\ \sqrt{2^{n-2d+1} LT_c / (T_s + 2^d T_f)} & \text{otherwise} \end{cases} \quad (4)$$

3.1.3 Phase 3: Data Collecting in DCNs

After Phase 2, in each DCN (which is an $h \times h$ mesh), the diagonal nodes have each received one of $M_0, M_1, \dots, M_{h-1}$. These submessages should be distributed to every node of the DCN. This is implemented in two stages: *row broadcasting* followed by *column collecting*.

In the row broadcasting stage, each node holding a submessage broadcasts along its own row in a recursive doubling manner. An example is shown in Fig. 5b in a 4×4 DCN. This takes d communication phases and incurs cost

$$T_{3-1}^{t1} - \sum_{i=1}^d (T_s + 2^{d-i} T_f + 2^{-d} LT_c). \quad (5)$$

In the column collecting stage, each node collects the submessages from the other nodes located on the same column. We first embed a logical (directed) ring on each

Fig. 6. A dilation-2 embedding of a logical ring into a linear path.

column of the DCN. This is done by first visiting even nodes downward the column and then odd nodes upward the column. An example of such embedding is shown in Fig. 6. Clearly, this gives a dilation-2 embedding. With this embedding, every node then pipelines its submessage following the direction of the ring $h-1$ steps, after which the broadcasting is done. An example is shown in Fig. 5c in a 4×4 DCN. The column collecting stage runs in time

$$T_{3-2}^{t1} = (h-1)(T_s + 2T_f + 2^{-d} LT_c), \quad (6)$$

where the constant 2 comes from the embedding dilation.

Summing the above, the total cost of this phase is

$$T_3 = T_{3-1}^{t1} + T_{3-2}^{t1} \\ = (2^d + d - 1)T_s + 3(2^d - 1)T_f + \left(1 + \frac{d-1}{2^d}\right)LT_c. \quad (7)$$

3.1.4 Performance Analysis and Comparison

The total costs of our algorithm, based on the SC, EDSF, and RD schemes in Phase 2 are, respectively as follows ("NP" stands for network partitioning):

$$T_{NP-RD}^t = T_1 + T_2^{RD} + T_3 \\ = (2^d + 2n - 1)T_s + (2^{n+1} + 3 \times 2^{2d} + 2^d - 5)T_f \\ + \left(2 + \frac{2n - d - 2}{2^d}\right)LT_c$$

$$T_{NP-SC}^t = T_1 + T_2^{SC} + T_3 \\ = (2^{n-d+1} + 2^d + 2n - 3)T_s + (3 \times 2^{n+1} - 2^d - 5)T_f \\ + \left(2 + \frac{d}{2^d} - \frac{2}{2^{2n-d}}\right)LT_c$$

$$T_{NP-EDSF}^t = T_1 + T_2^{EDSF} + T_3 \\ = (k_2 + 2^{n-d+1} + 2^d + 2d - 1)T_s \\ + ((k_2 + 5)2^d + 2^{n+1} - 5)T_f \\ + \left(2 + \frac{d-1}{2^d} + \frac{2^{n-2d+1}}{k_2}\right)LT_c.$$

For ease of comparison, we summarize the costs of RD, SC, and EDSF schemes (for a same-size torus) and ours in Table 1. Note that the cost of each algorithm should be $\alpha T_s + \beta T_f + \gamma LT_c$. There are several parameters interacting in Table 1. To understand how much performance gain can be obtained by using our schemes (by applying SC, EDSF, or RD in Phase 2) against the original SC, EDSF, and RD algorithms, we draw Fig. 7, which shows the broadcast latency versus various message lengths in a $2^5 \times 2^5$ torus with communication parameters $T_s = 150 \mu\text{sec}$, $T_f = 2 \mu\text{sec}$, and

1. Thus, in our terminology, these spanning fences are independent under the all-port model, but not so under the one-port model.

TABLE 1
COMPARISON OF BROADCAST COSTS IN A $2^n \times 2^n$ TORUS UNDER THE ONE-PORT MODEL

Algorithms	α	β	γ
RD [22]	$2n$	$2^{n+1} - 2$	$2n$
SC [3]	$2n + 2^{n+1} - 2$	$2^{n+2} - 4$	$2 - \frac{1}{2^{2n}}$
EDSF [3]	$k_1 + 2^{n+1}$	$k_1 + 2^{n+1}$	$1 + \frac{2^{n+1}}{k_1}$
NP-RD	$2n + 2^d - 1$	$2^{n+1} + 3 \times 2^d - 5$	$2 + \frac{2^{n-d-2}}{2^d}$
NP-SC	$2n + 2^{n-d+1} + 2^d - 3$	$2^{n+2} + 2^d - 5$	$2 + \frac{d}{2^d} - \frac{2}{2^{2n-d}}$
NP-EDSF	$k_2 + 2^{n-d+1} + 2^d + 2d - 1$	$(k_2 + 5)2^d + 2^{n+1} - 5$	$2 + \frac{d-1}{2^d} + \frac{2^{n-2d+1}}{k_2}$

(a)

(b)

(c)

Fig. 7. Broadcast latency vs. message size in a $2^5 \times 2^5$ torus with $T_s = 150 \mu\text{sec}$, $T_f = 2 \mu\text{sec}$, and $T_c = 0.5 \mu\text{sec}$. (a) RD vs. NP-RD, (b) SC vs. NP-SC, (c) EDSF vs. NP-EDSF.

$T_c = 0.5 \mu\text{sec}$ (note that the *relative* values of these parameters, instead of their *absolute* values, are what really matter; this should hold true throughout the analysis of this paper). From Fig. 7a, we see that NP-RD is much better than RD in most range of message sizes. Larger messages usually result in more performance gain. Also, the larger the value of d is, the more the performance gains. This is due to the larger number of DDNs performing broadcast at the same time. This can be verified from Table 1, where RD has smaller α value but much larger γ value than NP-RD.

From Fig. 7b, we observe that SC outperforms NP-SC only when $d \leq 3$ and L is fairly large ($\geq 20\text{K}$ bytes). When $d = 4$ or 5 , NP-SC outperforms SC in all ranges of L . This can be verified from Table 1, which indicates that SC has larger α but slightly smaller γ value than NP-SC. A similar phenomenon can be observed in Fig. 7c, and it does show in Table 1 that EDSF has larger α but slightly smaller γ than NP-EDSF.

To give an overall comparison, Fig. 8 shows the performance of RD, SC, EDSF, NP-RD, NP-SC, and NP-EDSF

Fig. 8. The effect of T_s/T_c ratio in a $2^5 \times 2^5$ torus with $h = 2^3$, $T_f = 2 \mu\text{sec}$ and $T_c = 0.5 \mu\text{sec}$. (a) $T_s = 10 \mu\text{sec}$ and $T_s/T_c = 20$, (b) $T_s = 150 \mu\text{sec}$ and $T_s/T_c = 300$.

in a $2^5 \times 2^5$ torus with T_s/T_c set to 20 and 300. We used $d = 3$ in our NP-based schemes. One referee also suggested us to include the *Postal model*² into our comparison. As can be seen, RD performs the best with fairly small messages ($L \leq 32$ bytes when $T_s/T_c = 20$, and $L \leq 256$ bytes when $T_s/T_c = 300$). Under $T_s/T_c = 20$, SC is the best when L is $3K \sim 5K$, and under $T_s/T_c = 300$, SC is the best when $L \geq 40K$. EDSF is only useful when $T_s/T_c = 20$ and $L > 5K$ (from the figure we conjecture that when $T_s/T_c = 300$, EDSF may become the best when L is impractically large). In all other cases, it is beneficial to use our NP-based schemes. In other words, our NP-based schemes are useful when the broadcast message is of a reasonable size, and this justifies the practical value of our result.

It is also worth comparing Figs. 8a and 8b to observe the effect of T_s/T_c on latency (in current parallel machines, this ratio ranges largely between 10 to 1000). We observe that a lower ratio T_s/T_c tends to give our algorithms more advantage over RD, while a higher ratio T_s/T_c tends to give our algorithms more advantage over SC and EDSF. The trend should remain the same as the ratio T_s/T_c changes.

3.1.5 An Emulation Experiment on Hypercube

The above comparisons are based on mathematical analyses. In order to justify that our analyses are correct, we have conducted an emulation experiment on a four-dimensional nCUBE/2 with 16 nodes. (The reason is that we do not have access to mesh/torus machines at the time of writing.) It is well-known that an n -cube has a subgraph which is a $2^i \times 2^j$ torus such that $i + j \leq n$. So we can use the 4-cube to emulate a 4×4 torus.

2. The Postal model [5] is designed for one-to-all broadcast in a complete graph. Based on factors such as network size, message size, and the machine's communication parameters, the model will determine an appropriate spanning tree to propagate the broadcast message. Although there are many factors to determine a good tree, as proposed in [5], it is possible to simplify the construction of the tree by using a divide-and-conquer technique. In case of tori/meshes (which are not complete graphs), it is also possible to schedule congestion-free communication. The performance results presented in Fig. 8 are obtained by varying the dividing factor (referred to as α in [5]) and choosing the one giving the least latency.

Fig. 9a shows the scenario of our emulation of a 4×4 torus using a 4-cube to perform the RD scheme. Fig. 9b shows the scenario to perform NP-RD when the emulated 4×4 torus is partitioned into two DDNs which are dilation-2 2×2 torus (i.e., $d = 1$). Because nCUBE/2 also uses wormhole and dimensional-ordered routing, the cube can precisely emulate the message-passing paths that we expect in an ordinary 4×4 torus. So the emulation experiment should be quite reliable. In the same manner, we also ran SC and NP-SC schemes on this emulated torus.

The obtained emulation results are shown in Fig. 10a. As reported in [23], the communication parameters of nCUBE/2 are: $T_s = 150 \mu\text{sec}$, $T_f = 2 \mu\text{sec}$, and $T_c = 0.46 \mu\text{sec}$. We apply these parameters to the predicted communication latency formulae developed in Table 1 and draw Fig. 10b. Comparing these two figures, we see that the predicted performance for RD, NP-RD, and NP-SC does conform closely to our emulation results. However, the emulated SC scheme is worse than our prediction, probably because the broadcast message is partitioned into too many pieces. Due to the size limitation of communication buffers in nCUBE/2, we could only simulate messages as large as 32K bytes. RD performs the best with fairly small messages ($L \leq 256$ bytes). When $256 \leq L \leq 4K$, NP-RD outperforms others. With $L > 4K$, NP-SC is the best. SC is the worst in our simulation, but we expect that it becomes the best when L is much larger.

3.1.6 NP-RD2h: Applying the NP Approach Based on Directed DDNs and the RD Scheme

Below, we briefly discuss how to apply our network-partitioning approach based on the directed DDNs in Definition 5. We use the RD scheme in Phase 2. As before, we set $h = 2^d$, where $1 \leq d \leq n$ and, thus, there are $2h$ DDNs: DDN_k^+ and DDN_k^- , $k = 0..h-1$.

In Phase 1, we let $r_k^+ = p_{k,k}$ (resp., $r_k^- = p_{k+1,k}$) be the representative node of DDN_k^+ (resp., DDN_k^-), $k = 0..h-1$. We first perform the same Phase 1 for undirected DDNs (refer

Fig. 9. Emulation of a 4×4 torus using a 4-cube: (a) the scenario of running the RD scheme and (b) the scenario of running the NP-RD scheme.

Fig. 10. Comparison of RD, NP-RD, SC, and NP-SC in a 4×4 torus: (a) emulation results using a four-dimensional nCUBE/2 and (b) predicted latency from analysis using $T_s = 150 \mu\text{sec}$, $T_f = 2 \mu\text{sec}$ and $T_c = 0.46 \mu\text{sec}$.

to Section 3.1.1). At the end, each r_k^+ sends half of its submessage to r_k^- (which is on r_k^+ 's right-hand side).

In Phase 2, we apply the RD scheme [22] on each directed DDN. This causes no problem because the RD scheme only uses a directed torus. However, only a submessage of half length is broadcast, as compared to the NP-RD scheme.

After Phase 2, the diagonal nodes on each DCN have each received a submessage. Further, by Definition 5, nodes of the right-hand side of these diagonal nodes have each also received a submessage. We first let the former nodes send their submessages to the latter on their left-hand side and, then, execute the same Phase 3 as in Section 3.1.3.

Let's call this scheme NP-RD2h. NP-RD2h only incurs two extra submessage transmissions (of size $\frac{L}{2h}$, one-hop away) in Phases 1 and 3 as opposed to NP-RD, but offers the benefit of broadcasting a smaller submessage (of $\frac{L}{2h}$ bytes) in

Phase 2. It is not hard to derive the exact cost of NP-RD2h. See Fig. 8, which also contains a draw for NP-RD2h (with $d = 3$). Generally speaking, NP-RD2h performs similarly to NP-RD. In some range of L , it is indeed worthwhile to use this scheme.

As a final note, the SC and EDSF schemes also use only one direction of the torus. So we can similarly develop a NP-SC2h and a NP-EDSF2h based on the $2h$ directed DDNs. However, our simulation tests showed no benefit in using them in most ranges of L . So the details, which are easy to develop, are omitted here.

3.2 All-Port Model

Now we consider broadcasting under the all-port model. For ease of making comparison, we use a square torus $T_{2^n \times 2^n}$. We will only use the undirected DDNs in Definition 3, but we do not use the directed DDNs in Definition 5, as we know of no scheme for broadcasting on a directed all-port torus (to

Fig. 11. Communication patterns in stage 1 of Phase 1 in an all-port torus: (a) the first step and (b) the second step.

be used in Phase 2). The possible number of DDNs will be $h = 2^d$, $d = 1..n$.

3.2.1 Phase 1: Scattering to Representative Nodes

Without loss of generality, let $p_{m,m}$, $m = \frac{h}{2}$ be the source node. Also, let $r_i = p_{i,i}$ be the representative node of DDN_i , $0 \leq i \leq h$. Observe that these $p_{i,i}$ s form a diagonal in $DCN_{0,0}$. The problem now becomes distributing submessages $M_0, M_1, \dots, M_{h-1}$ from the center of an $h \times h$ mesh to the diagonal nodes.

The problem can be solved efficiently in two stages. Let $D(p_{x,y}, k)$ denote the sequence of k nodes $[p_{x,y}, p_{x+1,y+1}, p_{x+2,y+2}, \dots, p_{x+k-1,y+k-1}]$. (For instance, $D(p_{0,0}, h)$ is the main diagonal of $DCN_{0,0}$.) As shown below, in the first stage, each M_i will be sent to one node in the i th row of the $h \times h$ mesh, $i = 0..h-1$. In the second stage, M_i will be sent to $p_{i,i}$.

STAGE 1. We divide h evenly into five integers t_i , $i = 0..4$, such that $h = \sum_{i=0}^4 t_i$. This can be done by letting $t_i = \lceil \frac{h-i}{5} \rceil$ (see [9, chapter 3]). Also, let $s_i = \sum_{k=0}^i t_k$. We define five node sequences Q_0, Q_1, Q_2, Q_3, Q_4 as follows: $Q_0 = D(p_{s_0,0}, t_0)$, $Q_1 = D(p_{s_0,s_0}, t_1)$, $Q_2 = D(p_{s_1,s_1}, t_2)$, $Q_3 = D(p_{s_2,s_2}, t_3)$, and $Q_4 = D(p_{s_3,s_3}, t_4)$. (See Fig. 11a for an illustration.) Let the center nodes of Q_0, Q_1, Q_2, Q_3, Q_4 be: C_0 = intersection node of column m and Q_0 , C_1 = center node of Q_1 , $C_2 = p_{m,m}$, C_3 = center node of Q_3 , and C_4 = intersection node of column m and Q_4 . First, node $p_{m,m}$ concurrently sends t_i submessages to nodes C_i , $i = 0, 1, 3, 4$, following the dimension-ordered routing. Then, C_i acts as the source node to scatter the t_i submessages it received to rows covered

by Q_i . In Figs. 11a and 11b, we illustrate the routing in the first and second steps, respectively. The about recursion is repeated until the length of each of sequences Q_0, Q_1, Q_2, Q_3 , and Q_4 reduces to one. After the recursion completes, one node in each row of $DCN_{0,0}$, will have received a submessage.

STAGE 2. Every node that received a submessage in stage 1 concurrently sends the submessage to $r_i = p_{i,i}$, $i = 0..h-1$.

With an arbitrary value of h , the execution time will be

$$T_1^A = \sum_{i=1}^{\lceil \log_5 h \rceil} \left(T_s + 2 \left\lceil \frac{h}{5^i} \right\rceil T_f + \left\lceil \frac{LT_c}{5^i} \right\rceil \right) + \left(T_s + \frac{hT_f}{2} + \left\lceil \frac{L}{h} \right\rceil T_c \right) \\ \leq (\lceil \log_5 h \rceil + 1)T_s + hT_f + (1/4 + 1/h)LT_c.$$

Plugging in $h = 2^d$, the cost is upper-bounded by

$$T_1^{At} \leq (\lceil d \log_5 2 \rceil + 1)T_s + 2^d T_f + (1/4 + 1/2^d)LT_c.$$

3.2.2 Phase 2: Broadcasting in DDNs

Next, we need to perform broadcasting in each DDN. We will use the *extended dominating node (EDN)* scheme proposed by [26], [27] designed for broadcasting in an all-port torus. The idea is illustrated in Fig. 12. In one step, the source node S sends the broadcast message to three nodes A, B, C . In another step with a T-pattern, 16 nodes will have received the broadcast message and these 16 nodes already form a regular pattern. Clearly, we can now partition the network into 16 submeshes and recursively perform the above steps. This works for tori of size $4^k \times 4^k$. It is also possible to extend the scheme for tori of size $(2 \times 4^k) \times (2 \times 4^k)$ [26], [27].

On an ordinary $T_{2^n \times 2^n}$ torus, the EDN scheme incurs a cost of

$$T_{EDN} = nT_s + \frac{2(2^{n+1} - 2 + \text{mod}(n, 2))}{3} T_f + nLT_c.$$

 Fig. 12. Two broadcasting steps of the EDN scheme in a 16×16 torus.

Translated into our dilated DDNs, the cost is

$$T_2^{EDN} = (n-d)T_s + \frac{2^{d+1}(2^{n-d+1} - 2 + \text{mod}(n-d, 2))}{3} T_f + \frac{n-d}{2^d} LT_c.$$

3.2.3 Phase 3: Data Collecting in DCNs

Similar to the one-port case (Section 3.1.3), the collecting is implemented in two stages: row broadcasting and column collecting.

The row broadcasting stage can be done by applying a recursive tripling technique as follows. We consider a list of consecutive h nodes x_i , $i = \frac{-h}{2} + 1, \frac{-h}{2} + 2, \dots, \frac{h}{2}$, located in a same row of the torus such that x_0 is the node which received a submessage in Phase 2. Note that the h consecutive nodes may be located in two consecutive DCNs. Then we partition the list evenly into three sublists, the first of length $\lceil \frac{h}{3} \rceil$, the second $\lceil \frac{h-1}{3} \rceil$, and the third $\lceil \frac{h-2}{3} \rceil$. Also, let x' and x'' be the middles of the first and third sublists, respectively. In one phase, x_0 concurrently sends its submessage to x' and x'' . Then x' , x_0 , x'' recursively act as the sources of the first, second, and third sublists, respectively. The idea is illustrated in Fig. 13. So, the number of nodes receiving the submessages will be tripled after each step. This gives a cost of

$$T_{3-1}^A = \sum_{i=1}^{\lceil \log_3 h \rceil} \left(T_s + \left\lceil \frac{h}{3^i} \right\rceil T_f + \left\lceil \frac{L}{h} \right\rceil T_c \right) \leq \lceil \log_3 h \rceil T_s + \left(\left\lceil \frac{h}{2} \right\rceil + \lceil \log_3 h \rceil \right) T_f + \lceil \log_3 h \rceil \left\lceil \frac{L}{h} \right\rceil T_c.$$

The column collecting stage can be done by invoking the same steps as in Section 3.1.3. The cost is the same T_{3-2}^{t1} in (6). So the cost of this phase is upper-bounded by

$$T_3^A = T_{3-1}^{At} + T_{3-2}^{t1} = (\lceil \log_3 h \rceil + h - 1) T_s + \left(\lceil \log_3 h \rceil + 2h - 2 + \left\lceil \frac{h}{2} \right\rceil \right) T_f + (\lceil \log_3 h \rceil + h - 1) \left\lceil \frac{L}{h} \right\rceil T_c. \quad (8)$$

Letting $h = 2^d$, the cost of this phase is bounded by

Fig. 13. Row broadcasting of Phase 3 in an all-port torus.

$$T_3^{At} = (\lceil d \log_3 2 \rceil + 2^d - 1) T_s + \left(\lceil d \log_3 2 \rceil + \frac{5}{2} 2^d - 2 \right) T_f + (\lceil d \log_3 2 \rceil + 2^d - 1) \left\lceil \frac{L}{2^d} \right\rceil T_c.$$

3.2.4 Performance Analysis and Comparison

Summing all three costs, the broadcast latency of our algorithm (termed as NP-EDN) in a $T_{2^n \times 2^n}$ is

$$T_{NP-EDN} = T_1^{At} + T_2^{EDN} + T_3^{At} \leq (2n + 2^d - d \lceil d \log_3 2 \rceil + \lceil d \log_5 2 \rceil) T_s + \left(\frac{2^{n+2}}{3} + \frac{13}{3} 2^{d-1} + \frac{\text{mod}(n-d, 2) 2^{d+1}}{3} \right) T_f + \left(\frac{5}{4} + \frac{n-d + \lceil d \log_3 2 \rceil}{2^d} \right) LT_c.$$

We summarize the broadcast costs of the EDN and our NP-EDN in Table 2. Our α and β factors are larger than those of the EDN scheme; the amount of difference depends on the parameter d . However, our γ factor is close to a constant, while that of EDN is linear to n . This is because no message partitioning is used in the EDN scheme. So we should expect much performance improvement over EDN when the message is large.

To understand the interaction among the α , β , γ factors, in Fig. 14 we depict the costs of EDN and NP-EDN in a $T_{2^5 \times 2^5}$

with $T_s = (10 \mu\text{sec or } 150 \mu\text{sec})$, $T_f = 2 \mu\text{sec}$, and $T_c = 0.5 \mu\text{sec}$ for various message sizes. Only with fairly small messages ($L \leq 64$ bytes when $T_s/T_c = 20$ and $L \leq 512$ bytes when $T_s/T_c = 300$), will EDN perform better. As the message sizes increase, significant gain can be obtained by using NP-EDN. Although a larger ratio T_s/T_c gives EDN more advantage, by comparing Figs. 14a and 14b, the effect seems to be very limited. Also, to get the best performance of NP-EDN, a median value of $d (= 2 \text{ or } 3)$ would be appropriate.

4 ONE-TO-ALL BROADCAST IN A MESH

Next, we apply our network-partitioning approach to a $2^n \times 2^n$ mesh. Again, the possible numbers of DDNs are still $h = 2^d$ for $d = 1..n$. Let the source node be $p_{x,y}$. We use the DDNs defined in Definition 6 with respect to the source $p_{x,y}$. Also, let $p_{x,y}$ reside in $DCN_{a,b}$ for some a and b . We choose the intersection of DDN_i and $DCN_{a,b}$ as the representative node r_i of DDN_i (there is only one such node).

4.1 Phases 1, 2, and 3

The schemes for meshes are similar to those for tori. However, there do exist some differences, mainly due to nonsymmetric structure of meshes. We summarize our schemes, as well as

TABLE 2
COMPARISON OF BROADCAST COSTS IN A $2^n \times 2^n$ TORUS UNDER THE ALL-PORT MODEL

Torus	α	β	γ
EDN[27]	n	$\frac{2(2^{n+1}-2+mod(n,2))}{3}$	n
NP-EDN	$n + 2^d - d + \lceil d \log_3 2 \rceil + \lceil d \log_5 2 \rceil$	$\frac{2^{n+2}+13 \times 2^{d-1}+mod(n-d,2)2^{d+1}}{3}$	$\frac{5}{4} + \frac{n-d+\lceil d \log_3 2 \rceil}{2^d}$

(a)

(b)

Fig. 14. Performance comparison of EDN and NP-EDN schemes in a $2^5 \times 2^5$ torus with $T_f = 2 \mu\text{sec}$ and $T_c = 0.5 \mu\text{sec}$. (a) $T_s = 10 \mu\text{sec}$ and $T_s/T_c = 20$, (b) $T_s = 150 \mu\text{sec}$ and $T_s/T_c = 300$.

the associated costs, for meshes in Table 3, under both one-port and all-port models. As before, in phase 2, we have choices of using the RD, SC, and EDSF schemes under the one-port case, and using the dominating (termed as D-node) scheme under the all-port case. So our NP-based schemes are named accordingly as NP-RDM, NP-SCM, NP-EDSFM, and NP-D. Note that when applying the SC scheme [3], we have modified its scatter phases as follows. We embed on each column and row of our DDNs (which are meshes) a dilated-2 directed ring, as illustrated in Fig. 6. On these columns/rows the column/row-scattering is then performed in a recursive-doubling manner. On the contrary, [3] suggests to traverse the whole column/row of the mesh along the positive direction and then connect the two boundary nodes by the negative channels. The long path connecting the boundary nodes will incur larger β value as opposed to our dilated-2 embedding.

4.2 Performance Analysis and Comparison

Comparisons on the broadcast latency, under one-port and all-port models, are summarized in Table 4 and Table 5, respectively. Note that the best values of k_3 and k_4 for EDSF and NP-EDSFM in Table 4 are, respectively,

$$k_3 = \begin{cases} 1 & \text{if } \sqrt{(2^{n+1})LT_c/(T_s + 2^n T_f)} < 1 \\ L & \text{if } \sqrt{(2^{n+1})LT_c/(T_s + 2^n T_f)} > L \\ \sqrt{(2^{n+1})LT_c/(T_s + 2^n T_f)} & \text{otherwise.} \end{cases} \quad (9)$$

$$k_4 =$$

$$\begin{cases} 1 & \text{if } \sqrt{2^{n-2d+1}LT_c/(T_s + (2^n - 2^d)T_f)} < 1 \\ L/2^d & \text{if } \sqrt{2^{n-2d+1}LT_c/(T_s + (2^n - 2^d)T_f)} > L/2^d \\ \sqrt{2^{n-2d+1}LT_c/(T_s + (2^n - 2^d)T_f)} & \text{otherwise.} \end{cases} \quad (10)$$

Fig. 15 draws the latency of all schemes in a $2^5 \times 2^5$ mesh under various values of L . The general trend is similar to that in tori—that RD works the best under fairly small messages, ours work the best under median-size messages, SC outperforms ours at larger messages ($\geq 15K$ when $T_s/T_c = 20$ and $\geq 40K$ when $T_s/T_c = 300$), and EDSF is only useful when L is extremely large.

In Table 5, it shows that our NP-D has higher α and β factors. But the $\gamma = O(n)$ of the D-node scheme is reduced by our NP-D to almost a constant if d is set to $O(\log n)$. So NP-D should be much faster D-node as L is large. Fig. 16 shows the comparison under various situations. As can be seen, only under small messages (≤ 64 bytes when $T_s/T_c = 20$ and $\leq 1K$ when $T_s/T_c = 300$) will the D-node scheme work better. As the message becomes larger the benefit of using our NP-D algorithm enlarges. Also, by comparing Figs. 16a and 16b, it indicates that a smaller ratio of T_s/T_c tends to give our NP-D more advantage.

5 ONE-TO-ALL BROADCAST IN AN ALL-PORT HYPERCUBE

In this section, we apply our network-partitioning approach to an all-port n -cube. Without loss of generality, let

TABLE 3
SUMMARY OF HOW OUR NP-BASED SCHEMES WORK IN A $2^n \times 2^n$ MESH

Mesheres	One-Port Model	All-Port Model
Phase 1: Scattering to Representative Nodes	<p>Multicast sub-message M_i to r_i, $i = 0..h-1$, in $DCN_{a,b}$. We apply the U-mesh scheme [16], which is for non-personalized multicast, and modify it for personalized multicast.</p> $T_1^m \leq dT_s + d(2^d - 1)T_f + \left(1 - \frac{1}{2^d}\right)LT_c$	<p>Source $p_{x,y}$ sends its message to the center of $DCN_{a,b}$. Then the center node executes the same stage 1 as in Section 3.2.1 on $DCN_{a,b}$. Each node received a sub-message in stage 1 sends the sub-message to r_i.</p> $T_1^{Am} = (T_s + 2^d T_f + LT_c) + T_1^{At}$ $= (\lceil d \log_5 2 \rceil + 2)T_s + 2^{d+1}T_f + \left(\frac{5}{4} + \frac{1}{2^d}\right)LT_c$
Phase 2: Broadcasting in DDNs	<p>Apply the RD scheme [16]. The cost is Eq. (2)</p> <p>Apply the modified SC scheme [3] as noted in the text.</p> $T_{2m}^{sc} = 2(2^{n-d} + n - d - 1)T_s + 6(2^n - 2^d)T_f + 2\left(\frac{1}{2^d} - \frac{1}{2^{2n-d}}\right)LT_c$ <p>Apply the EDSF scheme [3] for a mesh.</p> $T_{2m}^{edsf} = (k_4 + 2^{n-d+1})Y$ $Y = T_s + (2^n - 2^d)T_f + \frac{LT_c}{2^d k_4}$	<p>Apply the D-node scheme [24].</p> $T_2^D = (n - d + 2)T_s + 3(2^n - 2^d)T_f + \frac{n - d + 2}{2^d}LT_c$
Phase 3: Data Collecting in DCNs	<p>Same as Phase 3 for tori (see Section 3.1.3). The cost is Eq. (7).</p>	<p>Row-broadcasting on each DCN row by applying the technique in [21] followed by column collecting on each DCN column.</p> $T_3^{Am} = (\lceil (d+1) \log_3 2 \rceil + 2^d - 1)T_s + 3(2^d - 1)T_f + \left(1 + \frac{\lceil (d+1) \log_3 2 \rceil - 1}{2^d}\right)T_c$

TABLE 4
COMPARISON OF BROADCAST COSTS IN A $2^n \times 2^n$ MESH UNDER THE ONE-PORT MODEL

Mesheres	α	β	γ
RD [16]	$2n$	$2^{n+1} - 2$	$2n$
SC [3]	$2n + 2^{n+1} - 2$	$2(2^{2n} - 2^n)$	$2 - \frac{1}{2^{2n}}$
EDSF [3]	$k_3 + 2^{n+1}$	$(k_3 + 2^{n+1})(2^n - 1)$	$1 + \frac{2^{n+1}}{k_3}$
NP-RDM	$2n + 2^d - 1$	$2^{n+1} + (d+1) \times 2^d - d - 3$	$2 + \frac{2n-d-2}{2^d}$
NP-SCM	$2n + 2^{n-d+1} + 2^d - 3$	$3 \times 2^{n+1} + (d-3)2^d - d - 3$	$2 + \frac{d}{2^d} - \frac{2}{2^{2n-d}}$
NP-EDSFM	$k_4 + 2^{n-d+1} + 2^d + 2d - 1$	$(2^n - 2^d)(k_4 + 2^{n-d+1}) + (d+3)(2^d - 1)$	$2 + \frac{d-1}{2^d} + \frac{2^{n-2d+1}}{k_4}$

TABLE 5
COMPARISON OF BROADCAST COSTS IN A $2^n \times 2^n$ MESH UNDER THE ALL-PORT MODEL

Mesh	α	β	γ
D-node [24]	$n + 2$	3×2^n	$n + 2$
NP-D	$n - d + 2^d + 3 + \lceil (d+1) \log_3 2 \rceil + \lceil d \log_5 2 \rceil$	$3 \times 2^n + 2^{d+1} - 3$	$\frac{9}{4} + \frac{n-d+2+\lceil (d+1) \log_3 2 \rceil}{2^d}$

the source node be $00 \dots 0$. The DDNs and DCNs are as defined in Definition 7. The possible numbers of DDNs and DCNs are 2^d and 2^{n-d} , respectively, $d = 1..n$. The message to be broadcast is M of size L bytes. The representative node of DDN of the form $b_1 b_2 \dots b_d \dots 0^{n-d}$ is $b_1 b_2 \dots b_d 0^{n-d}$.

5.1 Phases 1, 2, and 3

Note that all representative nodes together form a subcube $*0^{n-d}$. In Phase 1, we construct the well-known *binomial spanning tree* [13] in subcube $*0^{n-d}$ rooted at $00 \dots 0$. Then the root scatters the submessage through the tree. The cost can be easily found to be

Fig. 15. Broadcast latency at various message sizes in a one-port $2^5 \times 2^5$ mesh with $h = 2^3$, $T_f = 2 \mu\text{sec}$, and $T_c = 0.5 \mu\text{sec}$. (a) $T_s = 10 \mu\text{sec}$ and $T_s/T_c = 20$, (b) $T_s = 150 \mu\text{sec}$ and $T_s/T_c = 300$.

Fig. 16. Broadcast latency in an all-port $2^5 \times 2^5$ mesh with $T_f = 2 \mu\text{sec}$ and $T_c = 0.5 \mu\text{sec}$. (a) $T_s = 10 \mu\text{sec}$ and $T_s/T_c = 20$, (b) $T_s = 150 \mu\text{sec}$ and $T_s/T_c = 20$.

$$T_1^c = dT_s + dT_f + \left(1 - \frac{1}{2^d}\right)LT_c.$$

In Phase 2, all DDNs simultaneously perform the broadcast scheme by Ho and Kao [10] for an all-port hypercube. This scheme (termed as HK) utilizes a near-optimal number of start-ups, but may incur high transmission cost (as will be seen later, the latter cost can be reduced by applying our network-partitioning approach). In an n -cube, the HK scheme takes time $T_{HK}(n) = \alpha_n T_s + \beta_n T_f + \gamma_n LT_c$, where the parameters α_n , β_n , γ_n can be derived recursively as follows:

$$\begin{aligned} \alpha_n &= \gamma_n = 1 + \alpha_{n-\lfloor \log_2 n + 1 \rfloor}, \quad n \geq 5 \\ \beta_n &= n + \beta_{n-\lfloor \log_2 n + 1 \rfloor}, \quad n \geq 5 \\ \alpha_1 &= 1, \alpha_i = \gamma_i = 2, \quad 2 \leq i \leq 4 \\ \beta_1 &= 1, \beta_2 = 2, \beta_3 = 4, \beta_4 = 5. \end{aligned}$$

As our DDNs are $(n-d)$ -cubes, the cost becomes

$$T_2^{hk} = \alpha_{n-d} T_s + \beta_{n-d} T_f + \frac{\gamma_{n-d}}{2^d} LT_c.$$

In Phase 3, in each DCN, each node has a submessage to be broadcast. This is in fact an all-to-all broadcast and can be done using the standard *exchange* algorithm in [13]. The cost can be easily derived to be

$$T_3^c = dT_s + dT_f + \left(1 + \frac{1}{2^d}\right)LT_c.$$

Fig. 17 shows an example how broadcasting from node 0000 is done in a 4-cube with d set to two.

5.2 Performance Analysis and Comparison

Table 6 compares the HK and our NP-HK schemes. Our α value is larger than the HK's (in fact, the HK's α value is very close to optimum). On the contrary, our γ value is much less than the HK's. For ease of comparison, we list the α , β , and γ values of HK and NP-HK when the network is a 5-cube, 10-cube, or 15-cube in Table 7.

Fig. 17. An example of broadcasting in an all-port 4-cube: (a) Phase 1, (b) Phase 2, (c) first step of Phase 3, (d) second step of Phase 3.

 TABLE 6
 COMPARISON OF BROADCAST COSTS IN AN ALL-PORT n -CUBE

Hypercube	α	β	γ
HK [10]	α_n	β_n	γ_n
NP-HK	$2d + \alpha_{n-d}$	$2d + \beta_{n-d}$	$2 + \frac{\gamma_{n-d}-2}{2^d}$

 TABLE 7
 COMPARISON OF α , β , γ VALUES IN A 5-CUBE, 10-CUBE, AND 15-CUBE AT VARIOUS d VALUES

Dimension	parameter	Ho-Kao	NP-HK Broadcast algorithm				
			$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
$n = 5$	α	3	4	6	8	9	10
	β	9	7	8	8	9	10
	γ	3	2	2	2	1.975	1.975
$n = 10$	α	4	6	8	10	11	13
	β	22	22	21	18	19	19
	γ	4	3	2.5	2.125	2.0625	2.03125
$n = 15$	α	6	8	9	11	13	13
	β	43	44	39	38	34	32
	γ	6	4	2.75	2.375	2.1785	2.0625

 Fig. 18. Performance comparison of HK and NP-HK in a 10-cube with $T_f = 2 \mu\text{sec}$ and $T_c = 0.5 \mu\text{sec}$. (a) $T_s = 10 \mu\text{sec}$ and $T_s/T_c = 20$, (b) $T_s = 150 \mu\text{sec}$ and $T_s/T_c = 20$.

To understand the interaction among the α , β , γ factors, in Fig. 18, we depict the costs of HK and NP-HK in a 10-cube with $T_s = (20 \mu\text{sec or } 150 \mu\text{sec})$, $T_f = 2 \mu\text{sec}$, and $T_c = 0.5 \mu\text{sec}$ for various message sizes. The results indicate that HK is better when L is small (≤ 64 bytes when $T_s/T_c = 20$, and $\leq 0.75K$ when $T_s/T_c = 300$). The larger ratio of T_s/T_c does give HK more advantage, due to the fact that our α value is larger. But as L increases, significant gain can be obtained by using NP-HK.

As a final comment, we note that the recently proposed scheme for broadcasting on all-port hypercubes by Wang and Ku [32] uses a more-near-to-optimal number of start-ups than that of the HK scheme. We can plug this scheme into Phase 2 easily. However, only very limited numbers of phases (typically one or two for reasonable ns) will be saved and the routing is not dimensional-ordered. Our tests also showed a similar scenario as discussed above, so the details are omitted here.

6 CONCLUSIONS

In this paper, we have presented a network-partitioning approach for one-to-all broadcast in wormhole networks. The approach works based on constructing multiple independent subnetworks which can work concurrently to increase the parallelism in communication. The *network-partitioning-based* approach distinguishes itself from the traditional *edge-disjoint-spanning-trees-based* approach (e.g., [4], [13], [18], [31], [5]) in many fundamental aspects. First, instead of trees, the subnetworks could be of any topology. Second, instead of a fixed constant, the number of independent subnetworks could be an adjustable parameter. Last, instead of a standard graph, a subnetwork could be a "diluted" graph and, thus, the special distance-insensitive characteristic of wormhole routing can be better utilized. As to future research, we feel that these fundamental issues may be used as a guideline in designing other collective communication patterns in wormhole networks.

We have also shown how to apply this network-partitioning approach to tori, meshes, and hypercubes. One interesting phenomenon is that many existing algorithms designed for these networks can be easily plugged into our schemes and be used by our diluted subnetworks. Extensive analyses, comparisons, and simulations have been performed when plugging in these alternatives and the results do confirm the advantage of using our network-partitioning approach in certain, usually reasonable, situations and configurations. All these have strongly justified the practical value of this approach.

Finally, the analyses in this work are based on a synchronous model, i.e., the communication steps are performed one after without interference. So the *depth-contention* problem [16] can be further studied when the communication steps are performed without synchronization.

ACKNOWLEDGMENTS

This research was supported in part by the National Science Council of the Republic of China under Grant no. NSC87-

2213-E-008-012 and Grant no. NSC87-2213-E-008-016. A preliminary version of this paper appeared in the *Proceedings of the Symposium on Parallel and Distributed Processing*, 1996 [33].

REFERENCES

- [1] V. Bala, J. Bruck, R. Cypher, P. Elustondo, A. Ho, C.T. Ho, S. Kipnis, and M. Snir, "CCL: A Portable and Tunable Collective Communication Library for Scalable Parallel Computer," *Proc. Int'l Parallel Processing Symp.*, pp. 835-843, Cancun, Mexico, Apr. 1994.
- [2] M. Barnett, S. Gupta, D.G. Payne, L. Shuler, R. van de Geijn, and J. Watts, "Interprocessor Collective Communication Library (Inter-Com)," *Proc. Scalable High Performance Computing Conf.*, pp. 357-364, 1994.
- [3] M. Barnett, D.G. Payne, R. van de Geijn, and J. Watts, "Broadcasting on Meshes with Worm-Hole Routing," *J. Parallel and Distributed Computing*, vol. 35, pp. 111-121, 1996.
- [4] J.-C. Bermond, P. Michallon, and D. Trystram, "Broadcasting in Wraparound Meshes with Parallel Monodirectional Link," *Parallel Computing*, vol. 18, pp. 639-648, 1992.
- [5] J. Bruck, L.D. Coster, N. Dewulf, C.-T. Ho, and R. Lauwereins, "On the Design and Implementation of Broadcast and Global Combine Operations Using Postal Model," *IEEE Trans. Parallel and Distributed Systems*, vol. 7, no. 3, pp. 256-265, Mar. 1996.
- [6] *Cray T3E Scalable Parallel Processing System*. Cray Research Inc., 1995.
- [7] W. Dally and C. Seitz, "The Torus Routing Chip," *J. Distributed Computing*, vol. 1, no. 3, pp. 187-196, 1986.
- [8] B. Duzett and R. Buck, "An Overview of the nCUBE3 Supercomputer," *Proc. Symp. Frontiers of Massively Parallel Computation*, pp. 458-464, 1992.
- [9] R.L. Graham, D.E. Knuth, and O. Patashnik, *Concrete Mathematics*. Addison-Wesley, 1994.
- [10] C.-T. Ho and M.-Y. Kao, "Optimal Broadcasting in All-Port Wormhole-Routed Hypercubes," *IEEE Trans. Parallel and Distributed Systems*, vol. 6, no. 2, pp. 200-204, Feb. 1995.
- [11] *A Touchstone DELTA System Description*. Intel Corp., 1990.
- [12] S.L. Johnson, "Communication Efficient Basic Linear Algebra Computations on Hypercubes Architectures," *J. Parallel and Distributed Computing*, vol. 4, no. 2, pp. 133-172, 1991.
- [13] S.L. Johnson and C.-T. Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes," *IEEE Trans. Computers*, vol. 39, no. 9, pp. 1,249-1,268, Sept. 1989.
- [14] R.E. Kessler and J.L. Schwazmeier, "CRAY T3D: A New Dimension for Cray Research," *Proc. COMPCON '93*, pp. 176-182, 1993.
- [15] P.K. McKinley, Y.-J. Tsai, and D.F. Robinson, "Collective Communication in Wormhole-Routed Massively Parallel Computers," *Computer*, vol. 28, no. 12, pp. 39-50, Dec. 1995.
- [16] P.K. McKinley, H. Xu, A.-H. Esfahanian, and L.M. Ni, "Unicast-Based Multicast Communication in Wormhole-Routed Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 12, pp. 1,252-1,265, Dec. 1994.
- [17] "Document for Standard Message-Passing Interface," *Message Passing Interface Forum*, Nov. 1993.
- [18] P. Michallon and D. Trystram, "Minimum Depth Arcs-Disjoint Spanning Trees for Broadcasting on Wrap-Around Meshes," *Proc. Int'l Conf. Parallel Processing*, vol. 1, pp. 80-83, 1995.
- [19] L.M. Ni and P.K. McKinley, "A Survey of Wormhole Routing Techniques in Directed Network," *Computer*, vol. 26, no. 2, pp. 62-76, Feb. 1993.
- [20] P.R. Nuth and W.J. Dally, "The J-Machine Network," *Proc. IEEE Int'l Conf. Computer Design: VLSI in Computer and Processors*, pp. 420-423, 1992.
- [21] J. Park, H.G. Kim, S. Hwang, J. Kim, I. Jang, H. Yoon, and J.W. Cho, "An Efficient Unicast-Based Multicast Algorithm in Two-Port Wormhole-Routed 2D Mesh Networks," *Proc. IEEE Int'l Conf. Algorithms and Architecture for Parallel Processing*, pp. 326-331, 1996.
- [22] D.F. Robinson, P.K. McKinley, and B.H.C. Cheng, "Optimal Multicast Communication in Wormhole-Routed Torus Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 6, no. 10, pp. 1,029-1,042, Oct. 1995.
- [23] m. Schmidt-Voigt, "Efficient Parallel Communication with the nCUBE 2S Processor," *Parallel Computing*, vol. 20, pp. 509-530, 1994.

- [24] Y.-J. Tsai and P.K. McKinley, "A Dominating Set Model for Broadcasting in All-Port Wormhole-Routed 2D Mesh Networks," *Proc. ACM Int'l Conf. Supercomputing*, pp. 126-135, 1994.
- [25] Y.-J. Tsai and P.K. McKinley, "An Extended Dominating Node Approach to Collective Communication in All-Port Wormhole-Routed 2D Meshes," *Proc. Scalable High Performance Conf.*, pp. 199-206, Knoxville, Tenn., May 1994.
- [26] Y.-J. Tsai and P.K. McKinley, "A Broadcasting Algorithm for All-Port Wormhole-Routed Torus Networks," *Proc. Symp. Frontiers of Massively Parallel Computation*, pp. 529-536, 1995.
- [27] Y.-J. Tsai and P.K. McKinley, "A Broadcasting Algorithm for All-Port Wormhole-Routed Torus Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 7, no. 8, pp. 876-885, Aug. 1996.
- [28] Y.-C. Tseng and S.K.S. Gupta, "All-To-All Personalized Communication in a Wormhole-Routed Torus," *IEEE Trans. Parallel and Distributed Systems*, vol. 7, no. 5, pp. 498-505, May 1996.
- [29] Y.-C. Tseng, T.-H. Lin, S.K.S. Gupta, and D.K. Panda, "Bandwidth-Optimal Complete Exchange on Wormhole-Routed 2D/3D Torus Networks: A Diagonal-Propagation Approach," *IEEE Trans. Parallel and Distributed Systems*, vol. 8, no. 4, pp. 380-396, Apr. 1997.
- [30] Y.-C. Tseng, D.K. Panda, and T.-H. Lai, "A Trip-Based Multicasting Model in Wormhole-Routed Networks with Virtual Channels," *IEEE Trans. Parallel and Distributed Systems*, vol. 7, no. 2, pp. 138-150, Feb. 1996.
- [31] Y.-C. Tseng and J.-P. Sheu, "Toward Optimal Broadcast in Star Graph Using Multiple Spanning Trees," *IEEE Trans. Computers*, vol. 46, no. 5, pp. 593-599, May 1997.
- [32] C.-M. Wang and C.-Y. Ku, "A Near-Optimal Broadcasting Algorithm in All-Port Wormhole-Routed Hypercubes," *Proc. ACM Int'l Conf. Supercomputing*, pp. 147-153, 1995.
- [33] S.-Y. Wang, Y.-C. Tseng, and C.-W. Ho, "Efficient Single-Node Broadcast in Wormhole-Routed Multicomputers: A Network-Partitioning Approach," *Proc. Symp. Parallel and Distributed Processing*, 1996.
- [34] H. Xu, P.K. McKinley, and L.M. Ni, "Efficient Implementation of Barrier Synchronization in Wormhole-Routed Hypercubes Multicomputers," *J. Parallel and Distributed Computing*, vol. 16, pp. 172-184, 1992.

Yu-Chee Tseng received his BS and MS degrees in computer science from the National Taiwan University and the National Tsing-Hua University in 1985 and 1987, respectively. From 1989 to 1990, he worked for the WANG Laboratory and D-LINK Inc. as a software engineer. He obtained his PhD in computer and information science from Ohio State University in January of 1994. From February 1994 to July 1996, he was with the Department of Computer Science, Chung-Hua University, Taiwan. Since August 1996, he has been an associate professor in the Department of Computer Science and Information Engineering, National Central University, Chung-Li, Taiwan. Dr. Tseng has served on the program committee of the International Conference of Parallel and Distributed Systems, 1996, and on the program committee of the International Conference on Parallel Processing, 1998. His research interests include parallel and distributed computing, fault-tolerant computing, parallel computer architecture, wireless network, and mobile computing. Dr. Tseng is a member of the IEEE Computer Society and the ACM.

San-Yuan Wang received his BS and PhD degrees from the Department of Computer Science and Information Engineering of the Tamkang University and the National Central University in 1991 and 1998, respectively. His research interests include parallel and distributed computing, parallel computer architecture, and mobile computing.

Chin-Wen Ho received the BS in mathematics from National Taiwan University in 1979, and MS and PhD degrees in computer sciences from National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1988, respectively. He is an associate professor in the Department Of Computer Sciences and Information Engineering at National Central University, Chung-Li, Taiwan. He is a member of the IEEE Computer Society. His research interests include algorithm design and analysis, graph theory, and parallel processing.