Multi-Node Broadcasting in All-Ported 3-D Wormhole-Routed Torus Using an Aggregation-then-Distribution Strategy

Yuh-Shyan Chen [a] Chao-Yu Chiang [b] Che-Yi Chen [c]

[a] *Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan, R.O.C.*

[b] *Division of Research and Developement, Computer Center, National Chung Cheng University, Chiayi, Taiwan, R.O.C.*

[c] *Department of Computer Science, National Tsing Hua University, Hsingchu, Taiwan, R.O.C.*

Abstract

In this paper, we investigate the *multi-node broadcasting* problem in a 3-D torus, where there are an unknown number of s source nodes located at unknown positions each intending to broadcast a message of size m bytes to the rest of the network. The torus is assumed to use the *all-port* model and the popular *dimension-ordered* routing. Existing congestion-free results are derived based on finding multiple edge-disjoint spanning trees in the network. This paper shows how to efficiently perform multi-node broadcasting in a 3-D torus. The main technique used in this paper is an *aggregation-then-distribution* strategy, which is characterized by the following features: (i) the broadcast messages are *aggregated* into some positions on the 3-D torus, then a number of *independent subnetworks* are constructed from the 3-D torus; and (ii) these subnetworks, which are responsible for *distributing* the messages, fully exploit the communication parallelism and the characteristic of wormhole routing. It is shown that such an approach is more appropriate than those using edge-disjoint trees for fixed-connection networks such as tori. Extensive simulations are conducted to evaluate this multi-broadcasting algorithm.

Key words: Broadcasting, collective communication, dimension-order routing, interconnection network, many-to-all broadcasting, parallel processing, torus, wormhole-routed.

1 Introduction

A massively parallel computer (MPC) consists of a large number of identical processing elements interconnected by a network. One basic communication operation which uses such a machine is broadcasting. Two commonly discussed instances are: *one-to-all* broadcasts and *all-to-all* broadcasts, where one or all nodes need to broadcast messages to the rest of the nodes [2]. A more complicated instance is the *many-to-all* (or *multi-node*) broadcast, where there is an unknown number of nodes located at unknown positions each intending to perform a broadcast operation. The focus of this paper is the multi-node broadcast problem, which has been applied to parallel graph algorithms, parallel matrix algorithms, fast Fourier transformations, and cache coherence [3], [5], [7], [12], [14]. This is especially true for those *collective communication patterns*, such as *broadcasting* and *multicasting*, which involve more than one source. Moreover, it is an important primitive communication operation for data-redistribution communication of a parallelizing compiler [16].

The *wormhole routing technology* [13], [15] has been s adopted by many new-generation parallel computers, such as the Intel Touchstone DELTA, Intel Paragon, MIT J-machine, Caltech MOSAIC, and Cray T3D. In such networks, a packet is partitioned into a sequence of *flits*, which are sent in a worm-like (or pipelined) manner. In the absence of congestion, the communication latency in such networks is proportional to the *additive* factor of message length and routing distance (while on the contrary the latency in a store-and-forward network is proportional to the *multiplicative* factor of message length and routing distance). It is for this reason that communication latency for network with wormhole routing is recognized to be quite distance-insensitive. Many routing algorithms have been proposed to utilize this property. For instance, [15] shows how to perform one-to-all broadcast in wormhole-routed all-port tori using as least communication phases as possible, where a *phase* consists of a set of congestion-free communication paths; paths of different lengths can co-exists in a phase, but the corresponding communications are expected to complete in about the same time due to wormhole routing's distance-insensitive property.

Email addresses: `yschen@cs.ccu.edu.tw` (Yuh-Shyan Chen), `cychiang@so-net.net.tw` (Chao-Yu Chiang), `jerry@cs.nthu.edu.tw` (Che-Yi Chen).

[1] The preliminary version of this paper is presented at The Third International Symposium on High Performance Computing, Oct. 2000, Tokyo, JAPAN.

[2] Yuh-Shyan Chen is with Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan, R.O.C., (Email: yschen@cs.ccu.edu.tw). This work was supported by the Ministry of Education, the the Republic of China, under grant 91A-H-FA07-1-4 (Learning Technology) and National Science Council of the Republic of China under grant #NSC-89-2218-E-305-002.

Multi-node broadcast problems have been studied in a variety of interconnection networks [5], [9], [10], [11], [12], [14]. Saad and Schultz [9], [10] initially defined this problem and proposed a simple routing algorithm for hypercubes. Stamoulis and Tsitsiklis [11] proposed a method of using n *edge-disjoint spanning trees* in an n-dimensional hypercube to solve this problem. A distributed approach to improve the load imbalance problem in [11] was presented by Tseng [12] for hypercubes and star graphs. Efforts were made by Varvarigos [14] to solve the more complicated problem where each source node may have several messages (of the same length) to broadcast. Susanne *et al.* [5] proposed a scheme called *s-to-p broadcasting*, where the authors try to align broadcast messages into a regular pattern before the broadcasting. Recently, Kesavan and Panda [6] investigated a multiple multicast with minimized node contention on wormhole k-ary n-cube networks. However, their approach attempted to reduce the node-contention problem, which does not produce a congestion-free result.

The aforementioned congestion-free results are all based on finding edge-disjoint spanning trees in a network and are appropriate for *non-fixed connection networks* [3]. One problem with this is that the number of edge-disjoint trees that can be offered by a network is fixed [3]. The other problem is that the characteristic of wormhole routing, which is assumed in this paper, is not well exploited [13]. In this paper, we consider multi-dimensional tori, which have been adopted by Cray T3D and T3E and are *fixed-connection* networks. The currently popular wormhole routing technology is assumed. In the literature, sending a packet involves two costs: *start-up time* and *transmission time*. Attempts to minimize both of these costs are made.

This paper addresses the multi-node broadcasting problem in wormhole-routed 3-D tori. Our approach is designed based on a proposed *aggregation-then-distribution* strategy. The major contribution of this paper is to present a way to develop a multi-node broadcasting using an *aggregation-then-distribution* strategy in wormhole-routed 3-D tori, which is characterized by the following features: (i) broadcast messages are *aggregated* into some positions of the 3-D torus, then a number of *independent subnetworks* are constructed from the 3-D torus, and (ii) these subnetworks, which are responsible for *distributing* the messages, can fully exploit communication parallelism and the characteristics of wormhole routing. We adopt an algebraic foundation [15] to develop our multi-node broadcasting algorithm. Using algebraic presentation allows us to develop an efficient multi-node broadcast algorithm in wormhole-routed 3-D tori. Our *aggregation-then-distribution* strategy is divided into two phases. First, *network-partitioning* techniques proposed in [13] are used to obtain multiple independent subnetworks (which differ from edge-disjoint spanning trees) in a torus. The number of independent subnetworks is actually an adjustable parameter. For a multi-node broadcast problem with an unknown number of s source nodes located at unknown positions in an torus each intending to broadcast an m-byte message, our approach can solve it efficiently in time $O(\max(\lceil \log_7 n \rceil, h)T_s + \max(\lceil \log_7 \lceil \frac{n}{h} \rceil \rceil \widetilde{m}, hm')T_c)$, where h is the number of independent subnetworks, $\widetilde{m} = \frac{n}{h^2}m$, and $m' = \frac{n}{h}m$. It is shown that this number has outperformed the aforementioned congestion-free scheme using edge-disjoint spanning-trees.

The rest of this paper is organized as follows. The basic ideas are given in Section 2. Section 3 presents our multi-node broadcasting in a 3-D torus. Timing analyses and comparisons are in Section 4, and conclusions are drawn in Section 5.

2 Basic Ideas

2.1 System Model

A *massively parallel computer* (MPC) is formally represented as $G = (V, C)$, where V denotes the node set and C specifies the channel connectivity. Each node contains a separate router to handle its communication tasks. In this paper, we consider G as a 3-D torus $T_{n_1 \times n_2 \times n_3}$ with $n_1 \times n_2 \times n_3$ nodes. In 3-D tori, each node is denoted as $P_{i,j,k}$, $1 \leq i \leq n_1$, $1 \leq j \leq n_2$, $1 \leq k \leq n_3$, and P_{i_1,i_2,i_3} has an edge connected to $P_{(i_1 \pm 1) \bmod n_1, i_2, i_3}$ along dimension one, an edge to $P_{i_1,(i_2 \pm 1) \bmod n_2, n_3}$ along dimension two, and an edge to $P_{i_1,i_2,(i_3 \pm 1) \bmod n_3}$ along dimension three. Each edge is considered to consist of two directed communication links pointing in opposite directions.

The *wormhole routing* model is assumed [7]. Under such a model, each packet is partitioned into smaller units called *flits*, which are sent in a pipelined manner. In the absence of congestion, the communication latency in the networks is proportional to the factor of the sum of the message length and the routing distance. Specifically, the time required to deliver a packet of L bytes from a source node to a destination node can be formulated as $T_s + LT_c$, where T_s is the start-up time containing the channel setup and software overhead, and T_c represents the transmission time per data byte. In this paper, attempts are made to maximize the trade-off between the start-up and transmission costs. In addition, we adopt the *all-port* model, that a node can simultaneously send and receive messages along all outgoing and incoming links, and *dimension-ordered routing* [12], that is every message must traverse links in a strictly increasing order.

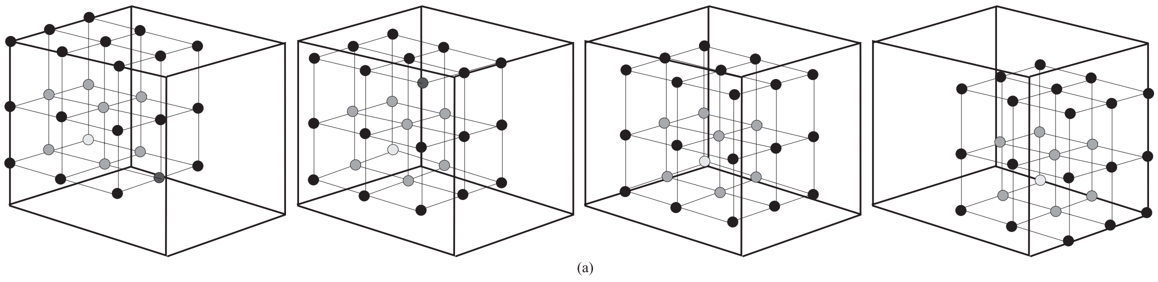
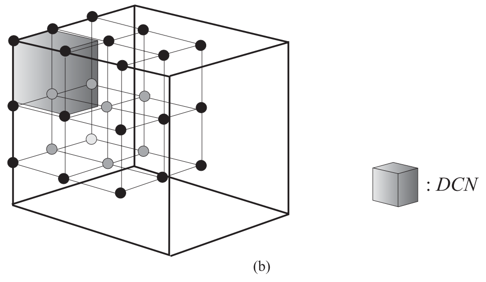

Fig. 1. (a) An example of *DDNs* and (b) *DCN* in a 3D torus.

2.2 Network Partitioning Scheme on a 3-D Torus

We expand the network partitioning scheme in 2-D tori [13] into 3-D tori as follows. Consider a 3-D torus $T_{n_1 \times n_2 \times n_3}$. Suppose that an integer h exists such that n_1, n_2, and n_3 are divisible by h. We define an $h \times h$ *data-distribution network* $DDN_{u,v}$ $=(V_{u,v}$,$C_u)$=DDN_i, $u, v = 0...h-1$ and $i = u * h + v$, as follows:

$$\begin{aligned} V_{u,v} &= \begin{array}{l} \{P_{i,j,l} | i = ah + ((u+v) \bmod h), j = bh + v, l = ch + u, \\ \text{for all } a = 0...\lceil \frac{n_1}{h} \rceil - 1, b = 0...\lceil \frac{n_2}{h} \rceil - 1, c = 0...\lceil \frac{n_3}{h} \rceil - 1\}; \end{array} \\ C_{u,v} &= \{\text{all channels of the } x\text{-axis } ah + u + v \text{ , } y\text{-axis } bh + v, \text{ and } z\text{-axis } ch + u\}. \end{aligned}$$

Each *DDN* is a *dilation-h* 3-D torus of size $\lceil \frac{n_1}{h} \rceil \times \lceil \frac{n_2}{h} \rceil \times \lceil \frac{n_3}{h} \rceil$, such that each edge is dilated by a path of h edges. Fig. 1 illustrates an example where the block nodes denote *DDN* and the gray zone represents *DCN*. The 3-D torus $T_{n_1 \times n_2 \times n_3}$ is partitioned into a $\frac{n_1 \times n_2 \times n_3}{h^3}$ *data collecting network* $DCN_k = (V_{a,b,c}, C_{a,b,c})$, $a = 0...n_1 - 1$, $b = 0...n_2 - 1$, $c = 0...n_3 - 1$, and $1 \leq k \leq \frac{n_1 \times n_2 \times n_3}{h^3}$, as follows:

$$\begin{aligned} V_{a,b,c} &= \{P_{i,j,l} | i = ah + x, j = bh + y, l = ch + z, \text{ for all } x, y, z = 0..h-1\}; \\ C_{a,b,c} &= \{\text{the set of edges induced by } V_{a,b,c} \text{ in } T_{n_1 \times n_2 \times n_3}\}. \end{aligned}$$

These DDNs and DCNs have the following properties.

1. DDN_0, DDN_1,..., and DDN_{h^2-1} are mutually independent (under the given port model).
2. DCN_0, DCN_1,..., and DCN_{k-1} are mutually independent (under the given port model), and together they contain all nodes of G.
3. DDN_i and DCN_j intersect in at least one node, for all $0 \leq i < h^2$ and $0 \leq j < \frac{n_1 \times n_2 \times n_3}{h^3}$.
4. DDN_0, DDN_1,..., and DDN_{h^2-1} are isomorphic.
5. DCN_0, DCN_1,..., and DCN_{k-1} are isomorphic.

2.3 Algebraic Notation

In the following, we adopt algebraic notation defined in [15] to represent the routing algorithm. The torus of size n is an undirected graph. Each node is denoted as $P_{x_1,x_2,...,x_k}$, $0 \leq x_i \leq n$, $1 \leq i \leq k$. Our routing algorithm is based on the concept of a "span of vector spaces" in linear algebra. The algebraic notation is used to represent the k-D torus from other perspectives. The torus is mapped into a Euclidean integer space Z^k, where Z is in the domain $\{0, 1, ..., n-1\}$. Conveniently, the i-th positive (resp., negative) elementary vector is denoted as $\vec{e}_i$ (resp., $\vec{e}_{-i}$) of Z^k, $i = 1..k$. We may rewrite $\vec{e}_{i_1} + \vec{e}_{i_2}$ as $\vec{e}_{i_1,i_2}$, $\vec{e}_{i_1} - \vec{e}_{i_2}$ $(=\vec{e}_{i_1} + \vec{e}_{-i_2})$ as $\vec{e}_{i_1,-i_2}$, and $\vec{e}_{i_1}$ +...+ $\vec{e}_{i_m}$ as $\vec{e}_{i_1,...,i_m}$. For instance, $\vec{e}_{1,3}$=$\vec{e}_1 + \vec{e}_3$ and $\vec{e}_{1,-3}$=$\vec{e}_1 - \vec{e}_3$.

Lemma 1 *In Z^k, given node x, a q-tuple of vectors B=$(\vec{b}_1, \vec{b}_2, ..., \vec{b}_q)$, and a q-tuple of integer N=$(n_1$,n_2, ..., $n_q)$, the span of x by vectors B and distances N is defined as*

$$SPAN(x, B, N) = \{x + \sum_{i=1}^{q} a_i \vec{b}_i \,|0 \leq a_i \leq n_i\}.$$

This notation is used to represent some subgroups in a tori. For instance, the main diagonal line of torus $T_{n \times n}$ is represented as $SPAN(P_{0,0}, (\vec{e}_{1,2}), n)$; an XY-plane passing through node $P_{0,0,i}$ in $T_{n \times n \times n}$ is denoted as $SPAN(P_{0,0,i}, (\vec{e}_1, \vec{e}_2), (n, n))$. A 3-D torus is viewed as $SPAN(P_{0,0,0}, (\vec{e}_1, \vec{e}_2, \vec{e}_3), (n, n, n))$.

We should introduce some notations to facilitate our routing algorithm. Considering a 3-D torus, we introduce the two matrices of delivery routing and distance. A *delivery routing matrix* $R = [r_{i,j}]_{3 \times 3}$ is a matrix with entries $-1, 0, 1$ such that each row indicates a message delivery; if $r_{i,j} = 1$ (resp. -1), the corresponding message will travel along the positive (resp. negative) direction of dimension j; if $r_{i,j} = 0$, the message will not travel along dimension j. A *distance matrix* $D = [d_{i,j}]_{3 \times 3}$ is an integer diagonal matrix (all non-diagonal

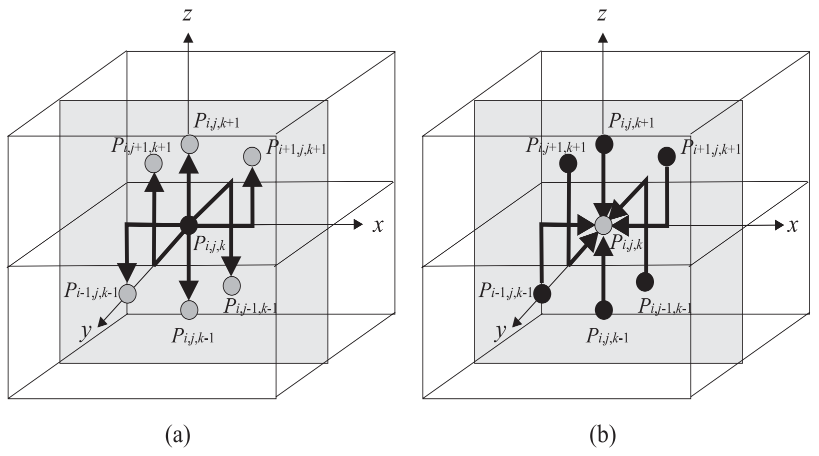

Fig. 2. Examples of (a) a routing matrix and (b) collection routing matrix.

elements are 0); $d_{i,i}$ represents the distance to be traveled by the i-th message described in R along each dimension.

For instance, the six message deliveries in Fig. 2(a) have three directions and thus can be represented by a delivery routing matrix:

$$R = \begin{bmatrix} \vec{e}_{1,3} \\ \vec{e}_{2,3} \\ \vec{e}_3 \end{bmatrix} = \begin{bmatrix} 1\;0\;1 \\ 0\;1\;1 \\ 0\;0\;1 \end{bmatrix}.$$

In general, the node $p_{i,j,k}$ sends M to the six nodes $p_{i+\alpha_1,j,k+\alpha_1}$, $p_{i,j+\alpha_2,k+\alpha_2}$, $p_{i,j,k+\alpha_3}$, $p_{i+\alpha_{-1},j,k+\alpha_{-1}}$, $p_{i,j+\alpha_{-2},k+\alpha_{-2}}$, and $p_{i,j,k+\alpha_{-3}}$, (note that $\alpha_{\pm i} \approx \pm\frac{it}{7}$, where t is block size; see Section 3 for details on deriving t). So two distance matrices can be used:

$$D^+ = \begin{bmatrix} \alpha_1 & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \alpha_3 \end{bmatrix} \text{ and } D^- = \begin{bmatrix} \alpha_{-1} & 0 & 0 \\ 0 & \alpha_{-2} & 0 \\ 0 & 0 & \alpha_{-3} \end{bmatrix}$$

and the 6 message deliveries in Fig. 2(a) are represented by matrix multiplication:

$$D^+ \times R = \begin{bmatrix} \alpha_1 & 0 & \alpha_1 \\ 0 & \alpha_2 & \alpha_2 \\ 0 & 0 & \alpha_3 \end{bmatrix} \text{ and } D^- \times R = \begin{bmatrix} \alpha_{-1} & 0 & \alpha_{-1} \\ 0 & \alpha_{-2} & \alpha_{-2} \\ 0 & 0 & \alpha_{-3} \end{bmatrix}.$$

For instance, given

$$D^+ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } D^- = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \text{ and}$$

$$D^+ \times R = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } D^- \times R = \begin{bmatrix} -1 & 0 & -1 \\ 0 & -1 & -1 \\ 0 & 0 & -1 \end{bmatrix},$$

then node $p_{i,j,k}$ may send M to the six nodes $p_{i+1,j,k+1}$, $p_{i,j+1,k+1}$, $p_{i,j,k+1}$, $p_{i-1,j,k-1}$, $p_{i,j-1,k-1}$, and $p_{i,j,k-1}$.

Further we define a similar routing matrix, namely the collection routing matrix C. A *collection routing matrix* $C = [c_{i,j}]_{3\times 3}$ is a matrix with entries $-1, 0, 1$ such that each row indicates the path of a collected message; if $c_{i,j} = 1$ (resp. -1), the corresponding message will be collected from neighboring nodes along the positive (resp. negative) direction of dimension j; if $c_{i,j} = 0$, the message will not be collected from neighbors along dimension j. Normally, if matrices C and R have the same entries, their representative routing path denotes the opposite direction. For instance as shown in Fig. 2(b), given a collection routing matrix

$$C = \begin{bmatrix} \vec{e}_{1,3} \\ \vec{e}_{2,3} \\ \vec{e}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix},$$

then matrix multiplication,

$$D^+ \times C = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } D^- \times C = \begin{bmatrix} -1 & 0 & -1 \\ 0 & -1 & -1 \\ 0 & 0 & -1 \end{bmatrix},$$

indicates that node $p_{i,j,k}$ collects six distinct messages from nodes $p_{i+1,j,k+1}$, $p_{i,j+1,k+1}$, $p_{i,j,k+1}$, $p_{i-1,j,k-1}$, $p_{i,j-1,k-1}$, and $p_{i,j,k-1}$. Observe that the collection routing matrix C is always used in the aggregation phase, and delivery routing matrix R is adopted in the distribution phase.

3 Multi-Node Broadcasting in 3-D Tori

Now we introduce our multi-node broadcasting algorithm. The network partition scheme is applied to a 3-D torus, so that h^2 number of DDNs and $\lceil \frac{n}{h} \rceil \times \lceil \frac{n}{h} \rceil \times \lceil \frac{n}{h} \rceil$ number of DCNs exist.

3.1 The Aggregation-then-Distribution Strategy

Existing multi-node broadcasting results are based on the construction of multiple spanning trees [11], [12], [14] such that all source nodes are evenly distributed to root nodes in all trees. As mentioned in Section 1, this approach is suitable for non-fixed connection networks [4], [11], [12], [14]. However, limiting the maximum number of multiple spanning trees is not suitable for fixed-connection networks [1], [8], [13]. The largest number of spanning trees in multi-dimensional tori is as many as the *degree*. It is worth mentioning that there are only six edge-disjoint spanning trees in 3-D tori. This motivates us to develop a truly efficient scheme to improve the limination.

Our proposed scheme, namely the *aggregation-then-distribution* scheme, is based on the network partitioning scheme; a torus network is partitioned into numbers of DDNs and $DCNs$. The main function of the *aggregation-then-distribution* scheme is outlined.

1. **Aggregation phase:** There are many source nodes located at unknown positions, and each one intends to broadcast its message. Source messages are aggregated into some regular positions of $h \times h$ DDNs. The purpose of the aggregation operation is to regularize the data pattern. Unfortunately, this operation causes load-imbalance problem. A tuning operation is presented for the purpose of load balancing.
2. **Distribution phase:** A number of *independent subnetworks* ($h \times h$ DDNs) are constructed from the 3-D torus. These subnetworks, which are responsible for distributing messages, can fully exploit the communication parallelism. Multi-node broadcasting is accomplished by means of these independent subnetworks.

In the following, the aggregation and distribution phases of multi-node broadcasting are presented, respectively.

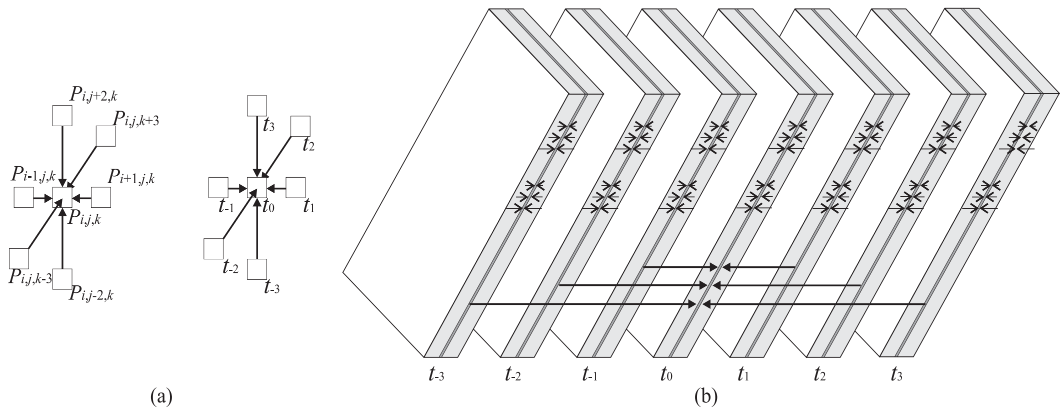

Fig. 3. (a) Data aggregation pattern when $h = 7$ and (b) data aggregation pattern when $h > 7$.

3.2 Aggregation Phase

Assume that there are s source nodes, and each one intends to broadcast its message. Initially, messages in all source nodes attempt to be aggregate to DDN_0, DDN_1,..., DDN_{h^2-1}. This operation is very efficient for the many-to-all communication pattern because the communication pattern is regulated in advance. The aggregation phase is divided into two steps:

- Step 1: diagonal-based data-aggregation operation; and
- Step 2: balancing-load operation.

3.2.1 Step 1: Diagonal-Based Data-Aggregation Operation

The main function of the data-aggregation operation is to standardize the communication pattern before multi-node broadcasting. The sizes of DDN_0, DDN_1,..., DDN_{h^2-1} and DCN_0, DCN_1,..., DCN_{k-1} are initially determined, where $k = \frac{n^3}{h^3}$.

Let a 3-D torus be represented as $SPAN(P_{0,0,0}, (\vec{e}_{1,3}, \vec{e}_{1,2}, \vec{e}_1), (n, n, n))$. Each DCN is viewed as $SPAN(P_{x,y,z}, (\vec{e}_{1,3}, \vec{e}_{1,2}, \vec{e}_1), (h, h, h))$, where $0 \leq x = ih, y = jh, z = kh < n$. The data-aggregation operation attempts to aggregate all possible messages to a special plane in each DCN which is denoted as a diagonal plane represented by $SPAN(P_{x,y,z}, (\vec{e}_{1,3}, \vec{e}_{1,2}), (h, h))$. That is, all nodes aggregate messages into the diagonal plane $SPAN(P_{x,y,z}, (\vec{e}_{1,3}, \vec{e}_{1,2}), (h, h))$. Initially, we let $h = 7$; then every node $P_{i,j,k}$ in the diagonal plane of each DCN aggregates messages from nodes $P_{i-1,j,k}$, $P_{i+1,j,k}$, $P_{i,j-2,k}$, $P_{i,j+2,k}$,

$P_{i,j,k-3}$, and $P_{i,j,k+3}$ as shown in Fig. 3(a). This operation is represented by

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, D^{+} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, D^{-} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix},$$

and

$$D^{+} \times C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \text{ and } D^{-} \times C = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix}.$$

Obviously, this communication pattern is congestion free. If $h > 7$, each distinct group can aggregate messages from seven different nodes, and then each of the seven groups can aggregate messages into a diagonal plane. Finally, all messages are aggregated into one diagonal plane, which is executed in $\lceil \log_7 h \rceil$ steps. Fig. 3(b) illustrates an example when $h = 49$; six messages from nodes $P_{i-t,j,k}$, $P_{i+t,j,k}$, $P_{i,j-2t,k}$, $P_{i,j+2t,k}$, $P_{i,j,k-3t}$, and $P_{i,j,k+3t}$ are aggregated to node $P_{i,j,k}$, where $t = 1$ and 7. This operation is represented by

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ D^{+} = \begin{bmatrix} t & 0 & 0 \\ 0 & 2t & 0 \\ 0 & 0 & 3t \end{bmatrix}, \ D^{-} = \begin{bmatrix} -t & 0 & 0 \\ 0 & -2t & 0 \\ 0 & 0 & -3t \end{bmatrix},$$

and

$$D^{+} \times C = \begin{bmatrix} t & 0 & 0 \\ 0 & 2t & 0 \\ 0 & 0 & 3t \end{bmatrix} \text{ and } D^{-} \times C = \begin{bmatrix} -t & 0 & 0 \\ 0 & -2t & 0 \\ 0 & 0 & -3t \end{bmatrix}.$$

Lemma 2 *Diagonal-based data-aggregation operations can be recursively performed on a torus $T_{n\times n\times n}$ in $\lceil \log_7 h \rceil T_s + \sum_{i=0}^{\lceil \log_7 h \rceil} 7^i m T_c = \lceil \log_7 h \rceil T_s + \frac{7^{\lceil \log_7 h \rceil} - 1}{6} m T_c$.*

3.2.2 Step 2: Balancing-Load Operation

After applying the data-aggregation operation, each DDN_0, DDN_1,..., and DDN_{h^2-1} has a different number of messages. This is load imbalance, so a

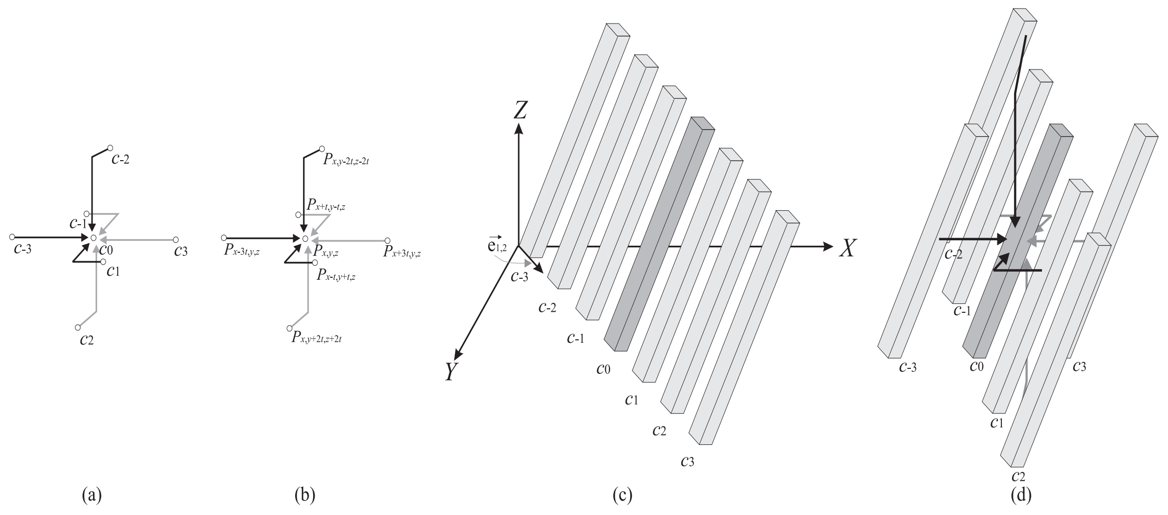

Fig. 4. (a) Plane to line if $h > 7$, (b) plane to line if $h = 7$, (c) diagonal plane pattern, (d) prefix-sum collection into a diagonal plane.

data tuning procedure is presented in order to achieve load balance. This part is divided into two procedures:

- Prefix-Sum Procedure: Each message-retaining node calculates a prefix-sum value; and
- Data Tuning Procedure: A data tuning operation is performed to achieve the load balance.

3.2.2.1 Prefix-Sum Procedure: The prefix-sum procedure exchanges information for calculating the prefix-sum value. After the data-aggregation operation, all source nodes' messages are aggregated to regular positions, which are in diagonal plane $SPAN(P_{x,y,z}, (\vec{e}_{1,3}, \vec{e}_{1,2}), (h, h))$, where $0 \le x = ih, y = jh, z = kh < n$. All those planes constitute a special cube $SPAN(P_{0,0,0}, (\vec{e}_{1,3}, \vec{e}_{1,2}, \vec{e}_{1}), (n, n, \lceil \frac{n}{h} \rceil))$.

In the following, we describe a simple diagonal-based recursive prefix-sum procedure. Our diagonal-based recursive prefix-sum procedure calculates a prefix-sum value for each message-retaining node in $SPAN(P_{0,0,0}, (\vec{e}_{1,3}, \vec{e}_{1,2}, \vec{e}_{1}), (n, n, \lceil \frac{n}{h} \rceil))$. The diagonal-based prefix-sum procedure is divided into forward and backward stages. In the forward stage, information on the number of messages is aggregated from a cube to a plane, and then from a plane to a line, and then from a line to one node. After the forward stage, the total number of whole source messages is kept in one node. In the backward stage, the partial prefix-sum value is returned from the node to a line, and then from a line to a plane, and eventually from a plane to a cube. Herein we omit the detailed operations. Two examples of plane-to-line and line-to-node

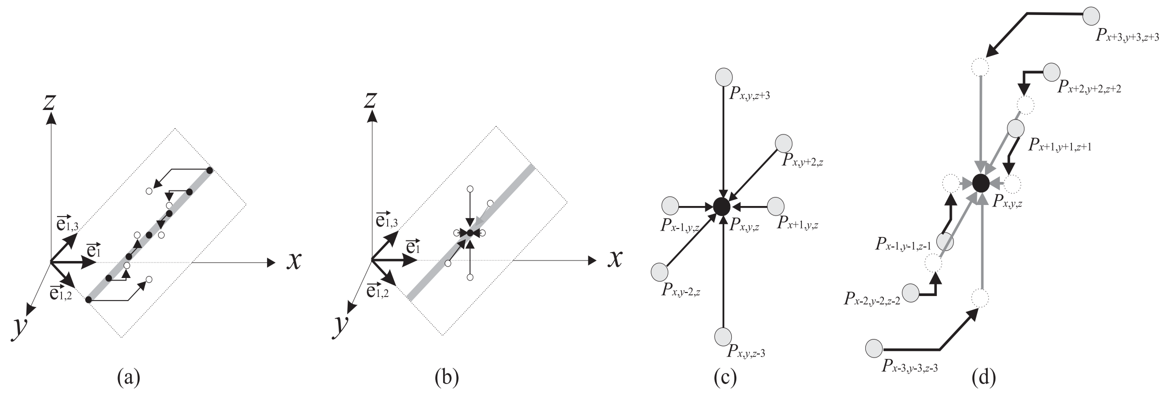

Fig. 5. (a) First stage of line to node, (b) second stage of line to node, (c) first stage collection pattern, (d) second stage collection pattern

operations are illustrated in Figs. 4 and 5. After the backward stage, every node in $SPAN(P_{0,0,0}, (\vec{e}_{1,3}, \vec{e}_{1,2}, \vec{e}_1), (n, n, \lceil \frac{n}{h} \rceil))$ has its own prefix-sum value for calculating a unique ranking number. Using ranking numbers allows each node in $SPAN(P_{0,0,0}, (\vec{e}_{1,3}, \vec{e}_{1,2}, \vec{e}_1), (n, n, \lceil \frac{n}{h} \rceil))$ to move extra messages to other *DDNs* to satisfy the load balance. Consequently, the total time cost of the prefix-sum procedure is $(8\lceil \log_7 n \rceil + 2\lceil \log_7 \lceil \frac{n}{h} \rceil \rceil + 1)(T_s + T_c)$.

3.2.2.2 Data Tuning Procedure: This task is divided into two parts: (1) finding a destination list and (2) performing a data tuning operation. Two important values are needed: one is the prefix-sum value and the other is the number of retained-messages. The destination list is calculated based on these two values. Assume that node x is located in $DDN_{i,j}$, with a destination list. The information on the destination list indicates that node x should move a message to certain neighboring nodes. To satisfy this purpose, for node x, if $(k, l) \in$ destination list, one message from $DDN_{i,j}$ (node x) is moved to $DDN_{k,l}$. That is, every node x performs the following operation.

S1. Finding a destination list: Having a prefix-sum value α and number of retained-messages β, then the destination list is $\digamma = \{\alpha \bmod h^2, (\alpha + 1) \bmod h^2, ..., (\alpha + \beta) \bmod h^2\}$ if the number of *DDN*s is h^2. Two communication steps are needed if one intends to move message from $DDN_{i,j}$ to $DDN_{k,l}$. One is for moving message from $DDN_{i,j}$ to $DDN_{k,j}$ (called *row tuning operation*) and other one is to move message from $DDN_{k,j}$ to $DDN_{k,l}$ (called *column tuning operation*). Finally, a destination sequence $\digamma'$ is constructed to find the destination $DDN_{k,l}$ for $DDN_{i,j}$. Note that $\digamma'$ is a sequence of pairs which is constructed as follows. For every $t \in \digamma$, let $(i = t \bmod h, j = t/h) \in \digamma'$, where i, j indicate the offset value of row and column tuning operations in the data tuning operation. For instance, if $h^2 = 49$, if $\alpha = 47$ and $\beta = 6$, then $\digamma = \{47, 48, 0, 1, 2, 3\}$. If a node is in $DDN_{5,5}$, then

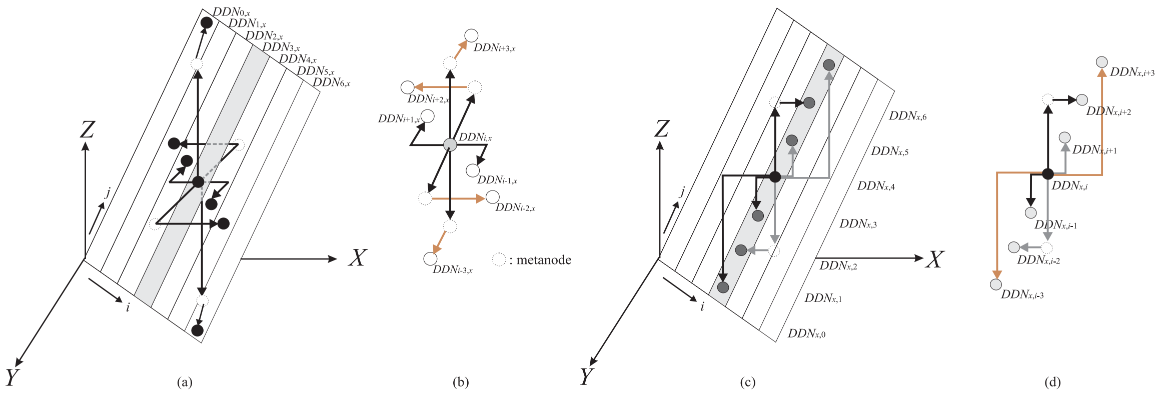

Fig. 6. (a),(b) Row tuning actions, and (c),(d) column tuning actions.

$F = \{47, 48, 0, 1, 2, 3\}$ and $F' = \{(5, 6), (6, 6), (0, 0), (1, 0), (2, 0), (3, 0)\}$.

S2. Data tuning operation: The data tuning operation is divided into row tuning and column tuning operations which are formally described below.

T1. Row tuning operation ($DDN_{i,j} \rightarrow DDN_{k,j}$): An extra alignment operation is executed due to the dimension-order routing. If $|i - k| \leq 3$, then we allow $DDN_{i,j} \rightarrow DDN_{i\pm1,j}$, $DDN_{i\pm2,j}$, and $DDN_{i\pm3,j}$ within two communication steps. For each node in the diagonal plane of $DDN_{i,j}$, we first align $DDN_{i\pm1,j}$ along dimension X with distance ±1, $DDN_{i\pm2,j}$ along dimension Y with distance ±2, and $DDN_{i\pm3,j}$ along dimension Z with distance ±3 to six meta-nodes, as shown in Fig. 6(a). Every node $P_{x,y,z}$ in diagonal plane $DDN_{i,j}$ distributes its messages to six nodes $P_{x-1,y-1,z}$, $P_{x+1,y+1,z}$, $P_{x,y-2,z}$, $P_{x,y+2,z}$, $P_{x,y,z-3}$, and $P_{x,y,z+3}$, which are represented by

$$R = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \text{ where } D^{+} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } D^{-} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

Let nodes $P_{x,y-2,z}$, $P_{x,y+2,z}$, $P_{x,y,z-3}$, and $P_{x,y,z+3}$ act as meta-nodes. Second, meta-nodes $P_{x,y-2,z}$ and $P_{x,y+2,z}$ forward messages to $P_{x-2,y-2,z}$ and $P_{x+2,y+2,z}$ along dimension X, and meta-nodes $P_{x,y,z-3}$, and $P_{x,y,z+3}$ forward messages to $P_{x,y-3,z-3}$ and $P_{x,y+3,z+3}$ along dimension Y as shown in Fig. 6(b). Further if $|i - k| > 3$, then this procedure can be generally performed in time $2\lceil \log_7 h \rceil (T_s + mT_c)$.

T2: Column tuning operation ($DDN_{k,j} \rightarrow DDN_{k,l}$): Due to dimension-order routing, an extra alignment operation is performed. If $|j - l| \leq 3$, then we allow $DDN_{k,j} \rightarrow DDN_{k,j\pm1}$, $DDN_{k,j\pm2}$, and $DDN_{k,j\pm3}$ within two communication steps. In the first step, a node in $DDN_{k,j}$ sends a message to $DDN_{k,j-1}$ and $DDN_{k,j+1}$ and aligns the message along dimension Z with distance ±2 to two meta-nodes. That is, every node $P_{x,y,z}$ directly sends a message to $P_{x-1,y,z-1}$ and $P_{x+1,y,z+1}$ and sends a message to two meta-nodes $P_{x,y,z-2}$ and $P_{x,y,z+2}$, which are represented by

$$R = \begin{bmatrix} 1\ 0\ 1 \\ 0\ 0\ 2 \\ 0\ 0\ 0 \end{bmatrix}, \text{ where } D^{+} = \begin{bmatrix} 1\ 0\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 1 \end{bmatrix} \text{ and } D^{-} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

In the second step, $DDN_{k,j}$ directly moves a message to $DDN_{k,j\pm 3}$ and, at the same time, two meta-nodes forward messages to destination $DDN_{x,j\pm 2}$. That is, every node $P_{x,y,z}$ sends a message to $P_{x-3,y,z-3}$ and $P_{x+3,y,z+3}$. Simultaneously, meta-nodes $P_{x,y,z-2}$ and $P_{x,y,z+2}$ send messages to $P_{x-2,y,z-2}$ and $P_{x+2,y,z+2}$ along dimension X. Further, if $|i-k| > 3$, then this task is generally completed in time $2\lceil \log_7 h \rceil (T_s + mT_c)$.

The total time cost of the data tuning operation is $4\lceil \log_7 h \rceil (T_s + mT_c)$. Note that the communication pattern of the row and column tuning operations is congestion free.

3.3 Distribution Phase

After the aggregation phase, each of the *data distribution networks* DDN_0, DDN_1,..., DDN_{h^2-1} has the same number of messages. These subnetworks, which are responsible for distributing messages can fully exploit the communication parallelism of wormhole routing. The distribution phase is divided into three steps.

- Step 1: (*Alignment Operation*) For every *DDN*, messages of nodes are aligned to the diagonal plane, and then an all-to-all broadcast operation is executed such that all nodes in each diagonal plane has the same number of broadcast messages. Note that different diagonal plane would have distinct broadcast messages.
- Step 2: (*Broadcast Operation*) Every *DDN* performs a diagonal plane broadcast scheme [15], such that all *DDN*s have the same copies of the broadcast messages.
- Step 3: (*Data-Collection Operation*) Each node in every *DCN* collects messages from all other nodes in the same *DCN*.

3.3.1 Step 1: Alignment Operation

Each *DDN* retains the same number of messages which indicates that each *DDN* has equal communication latency. The original 3-D torus is virtually partitioned into $h \times h$ subtorus *DDN*s. Each pair of *DDN*s is mutually disjoint.

The alignment operation, which is divided into two stages, allows congestion-free linking.

S1. Alignment to the diagonal plane: All possible messages are aligned into the diagonal plane. This task can be easily achieved by performing the diagonal-based data aggregation operation as introduced in Section 3.1, which takes time $\lceil \log_7(\lceil \frac{n}{h} \rceil) \rceil (T_s + \widetilde{m} T_c)$, where $\widetilde{m} = \frac{s}{h^2} m$.

S2. All-to-all broadcasting procedure on the diagonal plane: This procedure collects messages of each node in the diagonal plane $SPAN(P_{x,y,z}, (\vec{e}_{1,3}, \vec{e}_{1,2}), (\lceil \frac{n}{h} \rceil, \lceil \frac{n}{h} \rceil))$ from other nodes located in the same diagonal plane. The plane can be viewed as having $\lceil \frac{n}{h} \rceil$ rows or $\lceil \frac{n}{h} \rceil$ columns. Two broadcasting operations are needed. Basically, this procedure is the row and column tuning operations with different distance matrices of D^+ and D^-.

B1. Row broadcasting operation: This procedure is the same as the row tuning (**T1**) operation with the modification of

$$R = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \text{ where } D^+ = \begin{bmatrix} h & 0 & 0 \\ 0 & h & 0 \\ 0 & 0 & h \end{bmatrix} \text{ and } D^- = \begin{bmatrix} -h & 0 & 0 \\ 0 & -h & 0 \\ 0 & 0 & -h \end{bmatrix}.$$

B2. Column broadcasting operation: This procedure is the same as the column tuning (**T2**) operation with modification of

$$R = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix}, \text{ where } D^+ = \begin{bmatrix} h & 0 & 0 \\ 0 & h & 0 \\ 0 & 0 & h \end{bmatrix} \text{ and } D^- = \begin{bmatrix} -h & 0 & 0 \\ 0 & -h & 0 \\ 0 & 0 & -h \end{bmatrix}.$$

Further, if $\lceil \frac{n}{h} \rceil > 7$, the diagonal plane is partitioned into $\lceil \frac{n}{7h} \rceil \times \lceil \frac{n}{7h} \rceil$ groups. All-to-all broadcasting operations are executed in $\lceil \frac{n}{7h} \rceil \times \lceil \frac{n}{7h} \rceil$ groups in time $4 \lceil \log_7(\lceil \frac{n}{h} \rceil) \rceil (T_s + \widetilde{m} T_c)$, where $\widetilde{m} = \frac{s}{h^2} m$. Each communication step of row and column broadcasting operations is obviously congestion free. This is because messages travel along distinct dimensions during broadcasting operations.

3.3.2 Step 2: Broadcast Operation

After the alignment operation, every node in the diagonal plane of each *DDN* contains the same broadcast messages. The next step is to perform a well-known result, the diagonal broadcast scheme in a 3-D torus [15], on each *DDN* in parallel. The diagonal plane $SPAN(P_{x,y,z}, (\vec{e}_{1,3}, \vec{e}_{1,2}), (\lceil \frac{n}{h} \rceil, \lceil \frac{n}{h} \rceil))$ has partial source messages, and the broadcasting is based on recursively sending messages from a diagonal plane to six planes. Note that the operation is executed in time $\lceil \log_7 \lceil \frac{n}{h} \rceil \rceil (T_s + \widetilde{m} T_c)$, where $\widetilde{m} = \frac{s}{h^2} m$.

3.3.3 Step 3: Data Collection Operation

For each *data collecting network* (which is an $h \times h \times h$ mesh), each diagonal plane receives messages M_0, M_1, ..., and M_{h^2-1}. Each received message contains all messages of one *DDN*. These messages should be propagated to every node of the *DCN*. This is implemented in three stages: *row broadcasting* followed by *column* and *horizontal broadcasting*.

In the row broadcasting stage, we use a recursive scheme. Nodes located in the diagonal plane send messages to two nodes with distance $\pm\frac{1}{3}h$ and recursively propagate the message. This requires $\lceil \log_3 h \rceil$ communication phases and incurs the time cost $T_1 = \lceil \log_3 h \rceil (T_s + \widetilde{m} T_c)$, where $\widetilde{m} = \frac{s}{h^2} m$.

Every node collects partial messages from the row broadcasting stage. The messages belong to its column nodes; every node concurrently sends separate messages to other nodes in a pipelined scheme. We first embed a logical (directed) ring in each column of the *DCN*. This is done by first visiting even nodes down the column and then odd nodes up the column. This produces dilation-2 embedding. With this embedding, every node then pipeline-propagates its own messages following the ring. The same scheme is executed in horizontal broadcasting, i.e., pipeline-propagates messages in the Z-axis direction of the $h \times h \times h$ *DCN*. The column broadcasting stage runs in time $T_2 = (h-1)(T_s + \widetilde{m} T_c)$, where $\widetilde{m} = \frac{s}{h^2} m$, and horizontal broadcasting runs in time $T_3 = (h-1)(T_s + m' T_c)$, where $m' = \frac{s}{h} m$. The total time cost of the data collecting operation is $T = (\lceil \log_3 h \rceil + 2h - 2) T_s + [(\lceil \log_3 h \rceil + h - 1)\widetilde{m} + (h-1)m'] T_c$, where $\widetilde{m} = \frac{s}{h^2} m$ and $m' = \frac{s}{h} m$.

4 Performance Comparison

We begin with a discussion of time complexity, and then a simulation result is given to verify the effectiveness of our proposed scheme.

4.1 Performance Analysis

The time complexity of our proposed scheme is given herein.

Lemma 3 *The aggregation phase can be executed in a $T_{n \times n \times n}$ torus within*

$$((8\lceil \log_7 n \rceil + 5\lceil \log_7 h \rceil + 2\lceil \log_7 \frac{n}{h} \rceil + 1) T_s$$

Table 1
Comparison of the communication latency of multi-node broadcasting using various schemes.

Strategy	Start-up computation	Trans. computation
Edge-disjoint spanning trees (3D) [14]	$O(3\lfloor\frac{n}{2}\rfloor T_s)$	$O(3\lfloor\frac{n}{2}\rfloor \cdot \frac{sm}{6} T_c)$
Aggregation-based (3D)	$O(\max(\lceil\log_7 n\rceil, h)T_s)$	$O(\max(\lceil\log_7 \lceil\frac{n}{h}\rceil\rceil \frac{s}{h^2}m, sm)T_c)$

$$+\left[(4\lceil\log_7 h\rceil + \frac{7^{\lceil\log_7 h\rceil} - 1}{6})m + (8\lceil\log_7 n\rceil + 2\lceil\log_7 \frac{n}{h}\rceil + 1)\right] T_c,$$

where m denotes the size of a unit message.

Lemma 4 *The distribution phase can be executed in a $T_{n\times n\times n}$ torus within*

$$(6\left\lceil\log_7\left\lceil\frac{n}{h}\right\rceil\right\rceil + \lceil\log_3 h\rceil + 2h - 2)T_s$$
$$+[(6\left\lceil\log_7\left\lceil\frac{n}{h}\right\rceil\right\rceil + \lceil\log_3 h\rceil + h - 1)\frac{s}{h^2}m + (h-1)\frac{s}{h}m]T_c.$$

Theorem 5 *The multi-node broadcasting algorithm with the aggregation-then-distribution strategy can be performed in a $T_{n\times n\times n}$ torus within*

$$O(\max(\lceil\log_7 n\rceil, h)T_s + \max(\left\lceil\log_7\left\lceil\frac{n}{h}\right\rceil\right\rceil \frac{s}{h^2}m, sm)T_c).$$

A simple comparison of the time complexity is given. Since there are six *edge-disjoint spanning trees* [3] in a 3-D tori, where the height of a spanning tree is D+2, and D=$3\lfloor\frac{n}{2}\rfloor$ is the diameter in $T_{n\times n\times n}$. Therefore, the time complexity of the scheme using edge-disjoint spanning trees is $O(3\lfloor\frac{n}{2}\rfloor T_s + (3\lfloor\frac{n}{2}\rfloor \cdot \frac{sm}{6})T_c)$. Table 1 shows that our scheme is more efficient than the *edge-disjoint spanning-tree* scheme due to the fact that $O(\lceil\log_7\left\lceil\frac{n}{h}\right\rceil\rceil\frac{s}{h^2}mT_c) < O(3\lfloor\frac{n}{2}\rfloor \cdot \frac{sm}{6})T_c)$.

4.2 Simulation Results

We have developed a simulator to study the performance issue. We mainly compared our scheme against the multiple-spanning-tree scheme [11] under various situations. Parameters used in our simulations are listed below.

- The torus size is $16 \times 16 \times 16$.
- Startup time $T_s = 30\ \mu s$ and $T_c = 1\ \mu s$.

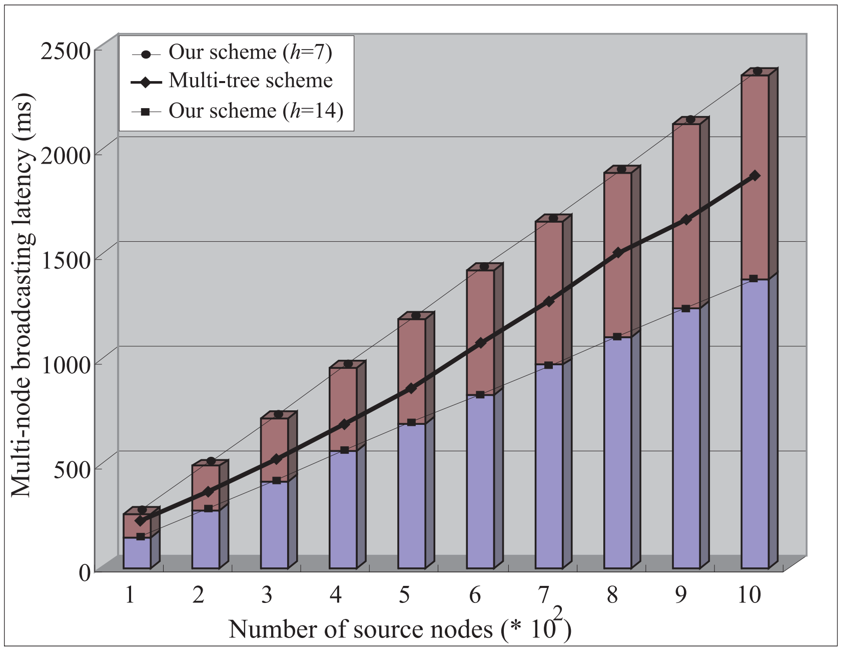

Fig. 7. Multi-node broadcast latency in a $16 \times 16 \times 16$ torus for various numbers of source nodes.

- Dilation $h = 7$ or 14.
- The message size rangs from 2 to $10k$.

Below, we show our simulation results from three prospects.

A) Effects of the Number of Sources: Fig. 7 shows the multi-node broadcast latency when $T_s = 30\ \mu s$ and $T_c = 1\ \mu s$ for various numbers of sources. Our scheme when $h = 7$ incurs higher latency than multiple-spanning-tree scheme, while our scheme when $h = 14$ has a lower latency than the multiple-spanning-tree scheme. This reflects the fact that our scheme performs better than the multiple-spanning-tree scheme with various number of sources.

B) Effects of Message Length: Fig. 8 shows the multi-node broadcast latency when $T_s = 30\ \mu s$ and $T_c = 1\ \mu s$ at various message lengths. Our scheme when $h = 7$ incurs a higher latency than that the multiple-spanning-tree scheme, while our scheme when $h = 14$ has a lower latency than the multiple-spanning-tree scheme. Our scheme truely has better performance than the multiple-spanning-tree scheme for various message length. Furthermore, Fig. 8 also illustrates that our scheme has better performance as more message are generated in the MPC.

C) Effects of Value of h: The value of h reflects the number of subnetworks, and thus the level of communication parallelism. So a larger h generally delivers a better performance. Figs. 7 and 8 compare multi-node broadcast latency

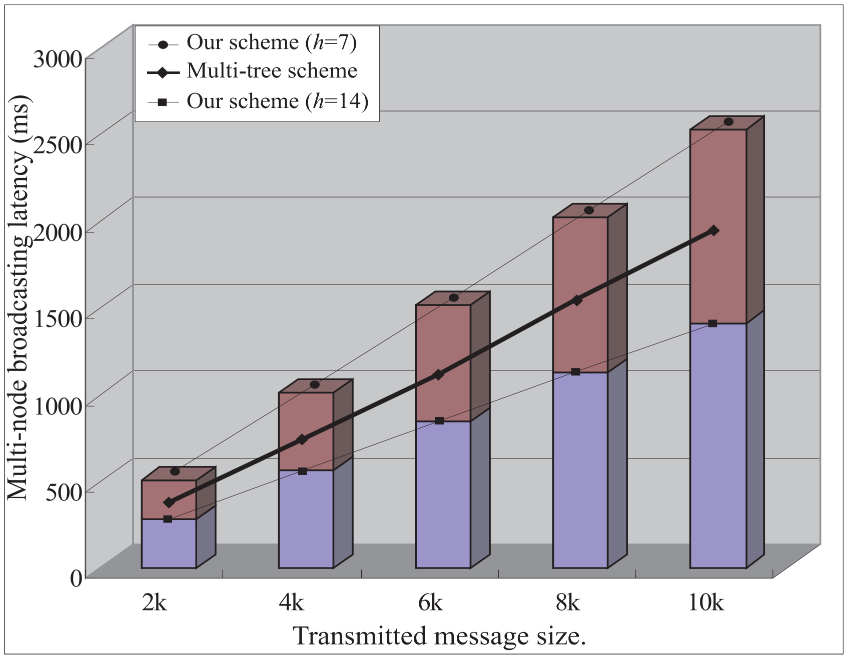

Fig. 8. Multi-node broadcast latency in a $16 \times 16 \times 16$ torus for various numbers of transmitted messages.

when $h = 7$ and 14. Observe that our scheme has a lower latency when $h = 14$ than when $h = 7$. This verifies that the higher the level of communication parallelism is, the better the performance will be.

By comparing Figs. 7 and 8, our scheme can exploit a higher level of communication parallelism than can the multiple-spanning-tree scheme. Generally speaking, it is worth mentioning that there is significant improvement in communication latency due to our scheme being able to exploit higher communication parallelism.

5 Conclusions

In this paper, we have shown how to solve the multi-node broadcast problem in a 3-D torus using a proposed *aggregation-then-distribution* strategy. The underlying assumptions are wormhole and dimension-ordered routing, which are currently in general use. The main technique is to partition the torus into a certain number of independent subnetworks such that all messages can be transmitted in parallel. This aggregation-then-distribution scheme is proposed for fixed-degree interconnection networks. Timing analysis has shown that this scheme is promising over conventional schemes using multiple spanning trees.

References

[1] J. Bruck, R. Cypher, and C. T. Ho. Fault-tolerant de bruijn and shuffle-exchange networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(5):548–553, 1994.

[2] Y. S. Chen, T. Y. Juang, and E. H. Tseng. Efficient broadcasting in an arrangement graph using multiple spanning trees. *IEICE Transactions on Fundamentals of Electronic, Communications, and Computer Science*, E83-A(1):139–149, Jan. 2000.

[3] M. Cosnard and D. Trystram. *Parallel Algorithms and Architectures.* Thomaon Computer Press, Boston MA, 1995.

[4] K. Day and A. Tripathi. Arrangement graphs: a class of generalized star graphs. *Information Processing Letter*, 42(5):235–241, 1992.

[5] S. E. Hambrusch, A. A. Khokhar, and Y. Lin. Scalable s-to-p broadcasting on message-passing mpps. *IEEE Transactions on Parallel and Distributed Systems*, 9(8):758–768, Aug. 1998.

[6] R. Kesavan and D. K. Panda. Multiple multicast with minimized node contention on wormhole k-ary n-cube networks. *IEEE Transactions on Parallel and Distributed Systems*, 10(4):371–393, 1999.

[7] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes.* Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[8] V. Lo, S. Rajopadhy, J. A. Telle, and X. Zhong. Parallel divide and comquer on meshes. *IEEE Transactions on Parallel and Distributed Systems*, 7(10), 1996.

[9] Y. Saad and M. Schultz. Data communication in hypercubes. *Journal of Parallel and Distributed Computing*, 6(1):115–135, Feb. 1989.

[10] Y. Saad and M. Schultz. Data communication in parallel architectures. *Parallel Computing*, 11:131–150, 1989.

[11] G. D. Stamoulis and J. N. Tsitsiklis. An efficient algorithm for multiple simultaneous broadcasts in the hypercube. *Information Processing Letter*, 46:219–224, Jul. 1989.

[12] Y. C. Tseng. Multi-node broadcasting in hypercubes and star graph. *Journal of Information Science and Engineering*, 14(4):809–820, 1998.

[13] Y. C. Tseng, S. Y. Wang, and C. W. Ho. Efficient broadcasting in wormhole-routed multicomputers: a network-partitioning approach. *IEEE Transactions on Parallel and Distributed Systems*, 10(1):44–61, Jan. 1999.

[14] E. A. Varvarigos and D. P. Bertsekas. Pratial multinode broadcast and partial exchange algorithms for d-dimension meshes. *Journal of Parallel and Distributed Computing*, 23:177–189, 1994.

[15] S. Y. Wang and Y. C. Tseng. Algebraic foundations and broadcasting algorithms for wormhole-routed all-port tori. *IEEE Transactions on Computers*, 49(3):246–258, Mar. 2000.

[16] M. Wolfe. *High Performance Compilers for Parallel Computing.* Addison-Wesley, USA, 1996.

Yuh-Shyan Chen received the B.S. degree in computer science from Tamkang University, Taiwan, Republic of China, in June 1988 and the M.S. and Ph.D. degrees in Computer Science and Information Engineering from the National Central University, Taiwan, Republic of China, in June 1991 and January 1996, respectively. He joined the faculty of Department of Computer Science and Information Engineering at Chung-Hua University, Taiwan, Republic of China, as an associate professor in February 1996. He joined the Department of Statistic, National Taipei University in August 2000, and then joined the Department of Computer Science and Information Engineering, National Chung Cheng University in August 2002. Dr. Chen served as IASTED Technical Committee on Telecommunications for 2002 2005, Program Committee Member of IEEE ICPADS'2001, IASTED CCN'2002-CCN2004, MSEAT'2003, IEEE ICCCN'2001 2003, ICPP'2003, ICDCS'2004, and IASTED CSA'2004. He was a Workshop Co-Chair of the 2001 Mobile Computing Workshop, and Guest Editor of Journal of Internet technology, special issue on "Wireless Internet Applications and Systems" (2002), special issue on "Wireless Ad Hoc Network and Sensor Networks" (2004), and Telecommunication Systems, special issue on "Wireless Sensor Networks" (2004). His paper wins the 2001 IEEE 15th ICOIN-15 Best Paper Award. His recent research include WLAN, mobile computing, mobile ad-hoc network, wireless sensor network, and mobile P2P computing. Dr. Chen is a member of the IEEE Computer Society, IEICE Society, and Phi Tau Phi Society.

Chao-Yu Chiang received the B.S. degree in computer science and Information Engineering from Tamkang University, Taiwan, Republic of China, in June 2001 and the M.S. degrees in Information Management from the National Taipei University, Taiwan, Republic of China, In June 2003. Since Aug. 2003, he serve as the Research Assistant at the Computer Center of National Chung Cheng University. His research interests include wireless network and mobile computing.

Che-Yi Chen received his B.S. degree in Computer Science and the M.S. degree in Electrical Engineering from the Chung-Hua University in 1995 and 1999, respectively. He worked for the Department of Computer Center, Chung-Hua University as a research assistant in 2001. He is now a PhD candidate in computer science in National Tsing-Hua University, Hsin-Chu, Taiwan. His research interests include collective communication, computer networks, and network security.