All-to-All Personalized Communication in Multidimensional Torus and Mesh Networks

Young-Joo Suh, *Member*, *IEEE*, and Kang G. Shin, *Fellow*, *IEEE*

Abstract—All-to-all personalized communication commonly occurs in many important parallel algorithms, such as FFT and matrix transpose. This paper presents new algorithms for all-to-all personalized communication or complete exchange in multidimensional torus- or mesh-connected multiprocessors. For an $R \times C$ torus or mesh where $R \leq C$, the proposed algorithms have time complexities of $O(C)$ message startups and $O(RC^2)$ message transmissions. The algorithms for three- or higher-dimensional tori or meshes follow a similar structure. Unlike other existing message-combining algorithms in which the number of nodes in each dimension should be a power-of-two and square, the proposed algorithms accommodate non-power-of-two tori or meshes where the number of nodes in each dimension need not be power-of-two and square. In addition, destinations remain fixed over a larger number of steps in the proposed algorithms, thus making them amenable to optimizations. Finally, the data structures used are simple, hence making substantial savings of message-rearrangement time.

Index Terms—Collective communication, all-to-all personalized communication, complete exchange, direct exchange, message-combining, interprocessor communication.

1 Introduction

Distributed-memory multiprocessors are known to be an attractive candidate architecture for scalable, massively parallel applications. Since memory is distributed among the processors, interprocessor communication is realized by passing messages through an interconnection network. However, interprocessor communication overhead is one of the major factors that limit the performance of parallel systems, and can become a bottleneck to scalable parallel implementations of computationally-intensive applications. This has resulted in the development of efficient, high-speed network architectures and innovative algorithms for scheduling interprocessor communication to minimize message latency.

A specific class of communication patterns that has received considerable attention is *collective communication* [7], [8], [10]. Collective communication is defined as a communication pattern involving a group of processes and it is supported by the Message Passing Interface (MPI), which is a portable, efficient, and flexible standard for message passing programs [20]. Commonly-used collective communication patterns are broadcast, scatter, gather, all-to-all broadcast, and all-to-all personalized communication. Collective communication is well-known for its high demand for network bandwidth and the resultant high algorithm-execution time.

- *Y.-J. Suh is with the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), San 31, Hyoja-Dong, Pohang, 790-784, Korea. E-mail: yjsuh@postech.edu.*
- *K.G. Shin is with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, 1301 Beal Ave., Ann Arbor, MI 48109-2122. E-mail: kgshin@eecs.umich.edu.*

Manuscript received 27 July 1998; revised 3 Jan. 2000; accepted 2 June 2000. For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 107212.

Among these collective communication operations, *all-to-all personalized communication* is the most demanding communication pattern [2], [3], [5], [6], in which every node communicates a distinct message to every other node in the system. In an N-node system, each node P_i, $1 \leq i \leq N$, has N blocks of data $B[i,1], B[i,2], \ldots, B[i,N]$, with a distinct block for each other node. After the operation, each node P_i has N blocks of data, $B[1,i], B[2,i], \ldots, B[N,i]$, one from each other node. In this operation, each node acts as the source of a scatter operation as well as the destination of a gather operation. As a result, this communication operation is also referred to as *all-to-all scatter-gather*, *complete exchange*, or *all-to-all personalized exchange*. All-to-all personalized communication is used in many scientific parallel algorithms, such as matrix transpose and fast Fourier transform (FFT).

In general, there are two paradigms for performing all-to-all personalized communication: direct and message-combining. In *direct* algorithms, every pair of processors exchange data directly. For an N-processor architecture equipped with a single port to the network interface and full duplex channels, such algorithms require at least $N-1$ steps. However, if the network is not fully-connected, then conflicts for use of links in the network may increase the number of communication steps. When the startup cost of a message transmission dominates, an alternative paradigm for all-to-all personalized communication is to use *message-combining*. Blocks destined for each processor are combined in messages in successive steps of operation. In this latter approach, message-combining generally results in longer messages and a reduced number of message initiations. Fig. 1 compares the two approaches for an all-to-all personalized communication in a 2×4 mesh. A direct algorithm proposed in [17] is applied to a 2×4 mesh, as illustrated in Fig. 1a. Using this algorithm,

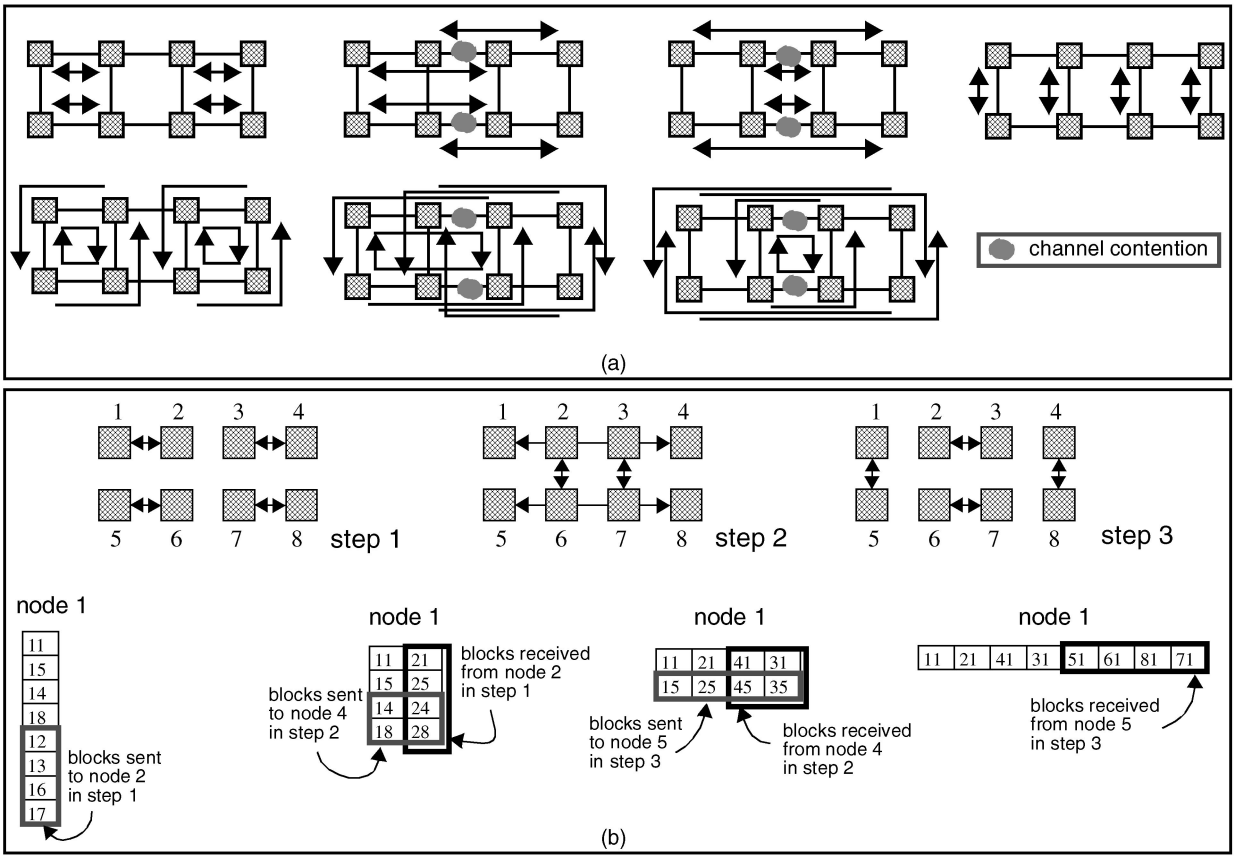

Fig. 1. Direct and message-combining approaches: (a) direct approach, (b) message-combining approach.

the all-to-all personalized communication requires at least seven communication steps since there are eight processors. However, as shown in Fig. 1a, channel contention occurs in four of those steps. As a result, at least 11 contention-free communication steps are required. In direct algorithms, each processor exchanges one block per step. Fig. 1b illustrates the same operation with message-combining. With this approach, only three contention-free communication steps are required. Fig. 1b also illustrates message blocks transmitted or received by processor 1 in each step, where a block is identified by the combination of source node ID and destination ID (e.g., a block which is originated from node 1 and destined for node 2 is identified by block 12). As shown in the figure, four message blocks are transmitted in each step. Thus, the message-combining algorithm requires three communication steps and 12 units of message transmission time, while the direct algorithm requires 11 communication steps and 11 units of message transmission time. It would be desirable if a model could be developed that would provide a guideline in formulating tradeoffs between message startup costs and message sizes based on the physical properties of the communication mechanisms in the target parallel architecture.

For all-to-all personalized communication using direct exchange, Scott [11] has shown that at least $a^3/4$ steps are required in an $a \times a$ mesh. Thakur and Choudhary [17] proposed direct algorithms for power-of-two 2D meshes. In Step i, $1 \leq i \leq N-1$, each node exchanges messages with the node that is determined by taking exclusive-OR of the node number with i. Therefore, the entire communi–cation pattern is decomposed into a sequence of pairwise exchanges. However, in some steps, link contentions exist among the exchange pairs. In machines such as Intel Paragon, the software overhead, not the channel contention, is the main concern. This algorithm [17] is based on the observation that channel contention is not a matter to be concerned about. Tseng and Gupta [18] proposed a direct algorithm for multidimensional tori. For an n-dimensional torus of $a_1 \times a_2 \times \ldots \times a_{n-1} \times a_n$, where $a_1 \geq a_2 \geq \ldots \geq a_{n-1} \geq a_n$, the algorithm requires $O({a_1}^2 \times a_2 \times \ldots \times a_{n-1} \times a_n)$ time complexity due to message startups.

In [1], Bokhari and Berryman developed two message-combining algorithms referred to as binary exchange and quadrant exchange for a circuit-switched $2^d \times 2^d$ mesh. In the binary exchange, the mesh is recursively halved and nodes symmetrically located with respect to each cut exchange block. The quadrant exchange treats the mesh as groups of 2×2 submeshes and exchanges blocks among the nodes in each submesh. Successive groups of 2×2 submeshes are interleaved until all blocks are exchanged. Sundar et al. [16] presented an algorithm called *cyclic exchange* in a power-of-two 2D square mesh. In each phase of the cyclic exchange, every node communicates in two steps with two other nodes, one in the same row and one in the same column. In a step of a phase, some pairs of nodes perform the horizontal exchange first, while

others perform the vertical exchange first. Subsequent steps reverse the order. While the cyclic exchange requires the same number of phases as the quadrant exchange, its performance in an asynchronous environment is shown to be superior. In [19], Tseng et al. present an algorithm with $O(2^d)$ time complexity for message startups in a $2^d \times 2^d$ 2D torus using message-combining. Nodes are partitioned into four groups, and messages are exchanged among nodes in the same group. After some data rearrangement, each node exchanges messages with a node in a different node group. In [12], [13], Suh and Yalamanchili proposed algorithms using message-combining in $2^d \times 2^d$ and $2^d \times 2^d \times 2^d$ tori or meshes with time complexities of $O(d)$ due to message startups and $O(2^{3d})$ (in 2D) or $O(2^{4d})$ (in 3D) due to message transmissions. In [15], Suh and Yalamanchili proposed a set of configurable algorithms for complete exchange for 2D meshes, which can be tuned to trade message-initiation or startup overhead against message-transmission time. These message-combining algorithms differ primarily in the manner in which pairwise exchange operations are scheduled. However, they have all been defined for meshes or tori where the number of processors in each dimension is an integer power-of-two and square.

In this paper, we present new algorithms for all-to-all personalized communication for multidimensional tori or meshes. The algorithms utilize message-combining to reduce the time associated with message startups. They are suitable for a wide range of torus or mesh topologies. The salient features of the proposed algorithms are:

1. Unlike existing message-combining algorithms, in which the number of nodes in each dimension should be power-of-two and square, they accommodate non-power-of-two and nonsquare tori or meshes.
2. They are simple in that destinations remain fixed over a larger number of steps, and are thus amenable to optimizations, e.g., caching of message buffers and locality optimizations.
3. They can be extended to higher-dimensional networks.

The following section presents the system model, performance parameters, and notation used in this paper. We propose algorithms for tori and meshes in Sections 3 and 4, respectively. Section 5 evaluates the performance of the proposed algorithms. Our results are summarized in Section 6.

2 Performance Model and System Architecture

The time to communicate data from one node to another node is a key source of overhead when executing parallel programs. A common metric used to evaluate the performance of collective communication algorithms is *completion time* or *communication time*.

In general, the completion time is comprised of two time components: start-up time and network time. *Start-up time* is the time required for both the source and destination nodes to handle the packet. It includes the time to prepare the message (e.g., adding header, tail, and error detection/correction information), manage buffers, and establish an interface between the local processor and the router. It is dominated by the software overhead in modern message-passing multiprocessors. *Network time* is the elapsed time from the first byte entering the network until the last byte arrives at the destination. It includes *message-transmission time* and *propagation time*. The message-transmission time is the per-byte transmission time multiplied by the message size in bytes. The time taken by the header of a message to travel from a node to a neighbor node is called the *per-hop delay*. The propagation delay is the per-hop delay multiplied by the number of links traversed by a message. In addition to the communication time, some collective communication algorithms (especially, all-to-all personalized communication algorithms using message-combining) require data rearrangement between communication steps. The data-rearrangement operations occur within a single node to prepare for the next step, i.e., to transmit the correct blocks to correct destinations. Furthermore, barrier synchronization is also required between successive communication steps. In this paper, we will consider these terms in analyzing the performance of all-to-all communication algorithms.

The following is a summary of the notation used in this paper.

- t_s: Startup time per message.
- t_c: Message transmission time per flit.
- t_l: Time for a flit to cross a link during path setup.
- t_b: Barrier synchronization time per step.
- ρ: Data rearrangement time per byte.
- m: Block size in flits.
- T: Completion time.

The target architecture is torus- or mesh-connected, wormhole-switched [9] multiprocessors. The proposed algorithms applies equally well to networks using virtual cut-through or packet switching. Each packet is partitioned into a number of *flits*. We assume that each processor has N distinct m-byte message blocks for all-to-all personalized communication. We also assume that the channel width is one flit, the flit size is one byte, and each processor has one pair of injection/consumption buffers for the internal processor-router channel (i.e., one-port architecture). All links are full duplex channels. In this paper, a *step* is the basic unit of a contention-free communication and a *phase* is a sequence of steps. The completion time for one communication step can be expressed as $T = t_s + m \cdot t_c + h \cdot t_l$ if one message block is transmitted to the destination across h hops in a contention-free manner using wormhole switching. This time does not include the data-rearrangement time between steps.

The logical data structure in each node is a 2D array (in 2D networks) or nD array (in n-dimensional networks). We assume that these arrays are arranged in column-major order. We also assume that if physically noncontiguous blocks are transmitted from this array, a message-rearrangement step must take place prior to transmission.

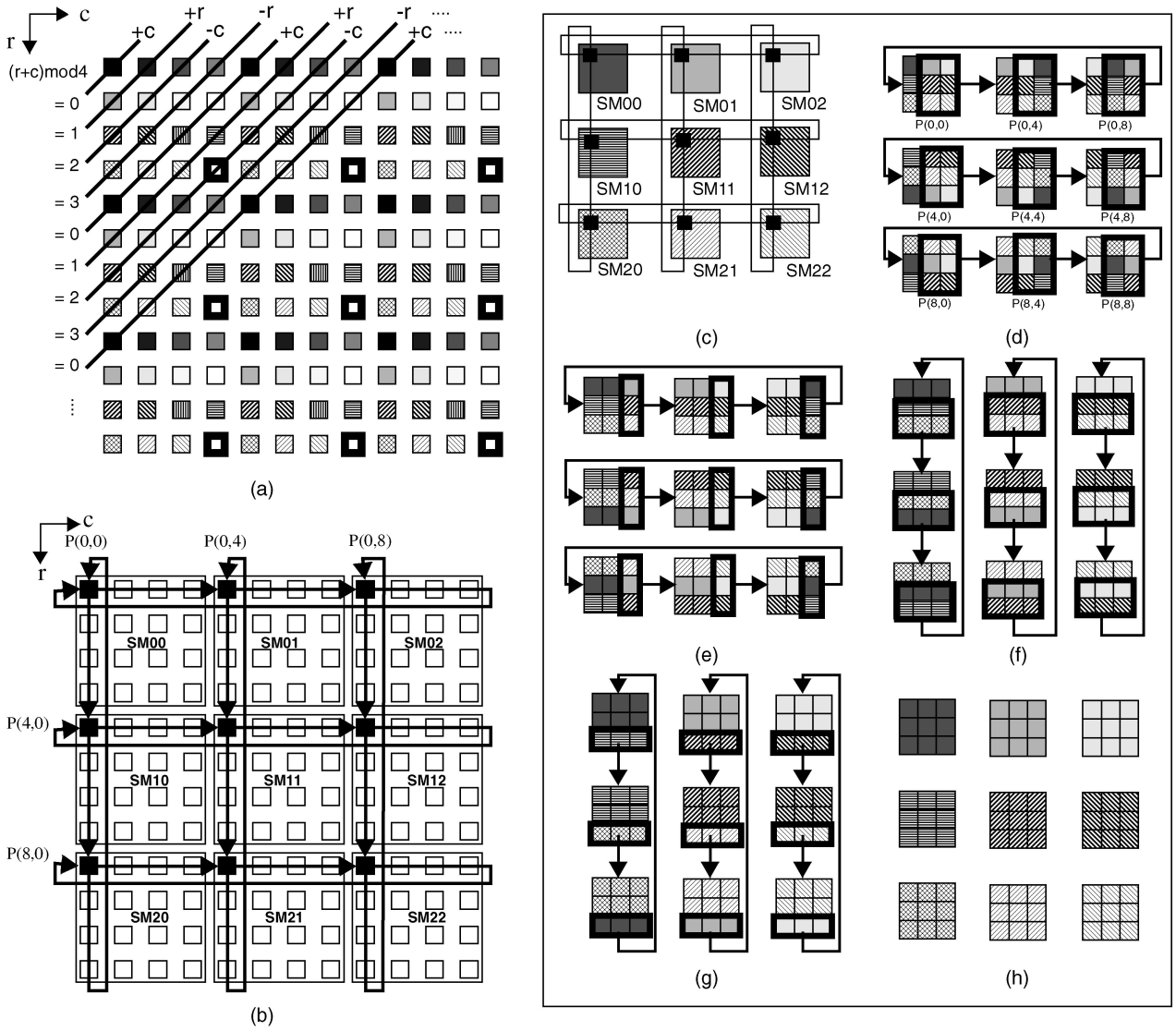

Fig. 2. Node groups, a 3×3 subtorus formed by a node group, and all-to-all personalized communication operation among nodes in a subtorus. (a) 16 distinct node groups and directions taken by each node in Phase 1, (b) 3×3 subtorus formed by group 00, (c) 4×4 submeshes and nodes in group 00, (d) Phase 1 Step 1, (e) Phase 1 Step 2, (f) Phase 2 Step 1, (g) Phase 2 Step 2, and (h) after Phase 2.

3 Algorithms for Tori

3.1 2D Tori

For an $R \times C$ torus, where R and C are multiples of four and $R \leq C$, each node is identified by a label $P(r, c)$, $0 \leq r \leq R-1$ and $0 \leq c \leq C-1$. Each node is included in one of 16 node groups according to the following rule:

IF *r mod 4 = i* and *c mod 4 = j*, *then* P(r,c) is included in group ij.

For example, in a 12×12 torus shown in Fig. 2a, nine nodes of identical marking are included in the same group. The nodes in a group form an $\frac{R}{4} \times \frac{C}{4}$ subtorus. Fig. 2b illustrates the 3×3 subtorus formed by group 00 to which nine nodes, P(0,0), P(0,4), P(0,8), P(4,0), P(4,4), P(4,8), P(8,0), P(8,4), and P(8,8) belong. In addition, if an $R \times C$ torus is divided into 4×4 contiguous submeshes (SMs), each node in a SM is included in one of 16 distinct groups.

3.1.1 An Overview

The proposed 2D algorithm consists of four phases. In Phases 1 and 2, messages are exchanged, performing all-to-all personalized communication, among the nodes in the same group. For an illustrative purpose, we consider all-to-all personalized communication in a 12×12 torus. Fig. 2c is a simplified representation of Fig. 2b, where only SMs and nodes in group 00 are shown. Each node has 144 blocks to scatter, and the blocks are divided into nine 4×4 block groups (BGs) considering 9 SMs (SM00, SM01, SM02, SM10, SM11, SM12, SM20, SM21, and SM22) and 16 nodes in each SM. In Fig. 2d, each node in group 00 has 9 BGs to scatter with distinct markings, where each BG is destined for the SM which has the same marking as the BG in Fig. 2c. Thus, BGs of identical marking will be gathered in one node in the SM that has the same marking as the BGs, upon completion of all-to-all personalized communication. Before starting transmission, the BGs are stored in a 2D array and they are arranged by considering the following Steps (to be

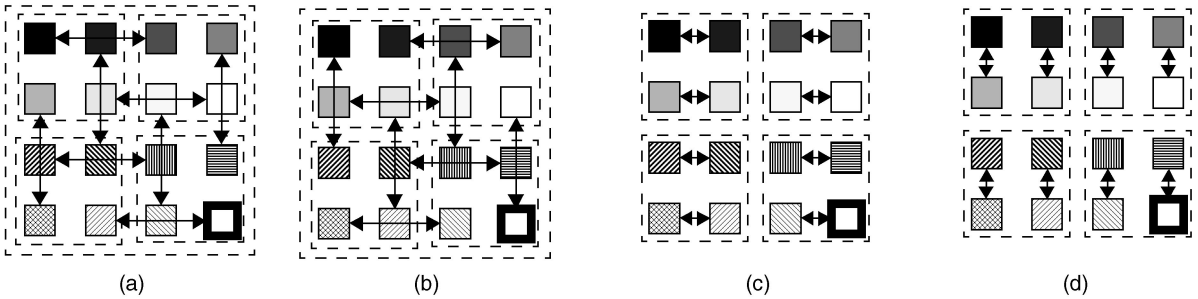

Fig. 3. Communication pattern in Phases 3 and 4 for a 12×12 torus. (a) Phase 3 Step 1, (b) Phase 3 Step 2, (c) Phase 4 Step 1, and (d) Phase 4 Step 2.

described in Section 3.1.3). In Step 1 of Phase 1, each node transmits the BGs in the second and third columns while receiving the same number of BGs along a row, as illustrated in Fig. 2d. The data arrays after Step 1 of Phase 1 are illustrated in Fig. 2e. In Step 2 of Phase 1, each node transmits the BGs in the third column while receiving the same number of BGs (see Fig. 2e). After Step 2 of Phase 1, BGs in each node are those destined for nodes in its SM and SMs in the same column as shown in Fig. 2f. Now, Phase 2 starts and each node changes dimensions and transmits BGs along a column. In Step 1 (Step 2) of Phase 2, each node transmits the BGs in the second and third rows (third row) while receiving the same number of BGs along a column, as shown in Fig. 2f (Fig. 2g). After Step 2 of Phase 2, all BGs gathered in each node have the same marking (see Fig. 2h), which indicates that all-to-all personalized communication among nodes in group 00 is achieved successfully.

In Phases 1 and 2, nodes in the same group perform all-to-all personalized communication among them, as described above. However, since nodes in 16 distinct groups perform the operations in parallel, we should schedule links to avoid channel contention. If we consider a row (or column), each node in the row (or column) is included in one of four node groups (see Fig. 2a). Since nodes in four groups cannot transmit message blocks along two directions in the row (or column) in parallel without channel contention, two node groups should be assigned to two directions in the other dimension for contention-free transmissions. Since there are four directions, positive row (+r), negative row (-r), positive column (+c), and negative column (-c), four node groups share distinct directions according to the result of $(r+c) mod 4$ operation (see Fig. 2a). In Phase 2, each node changes dimensions then performs transmission along the new dimension.

After Phase 2, each node in a SM has blocks originated from nodes in the same node group *and* destined for the 16 nodes in the same SM to which the node belongs. In the next two phases (Phases 3 and 4), message transmissions are performed among nodes in distinct groups and in the same SM. Each SM can be divided into four 2×2 submeshes. In each 2×2 submesh, there are four nodes in upper left, upper right, lower left, and lower right. In the two Steps of Phase 3, four nodes in the same position in 2×2 submeshes exchange blocks (see Figs. 3a and 3b, where only one SM is shown). In a step, each node transmits blocks destined for the receiver node as well as blocks destined for the other three nodes in the 2×2 submesh to which the receiver node belongs. After Phase 3, each node in a 2×2 submesh has blocks originated from nodes in four distinct groups and destined for nodes in the same 2×2 submesh to which the node belongs. In the two steps of Phase 4, four nodes in each 2×2 submesh exchange blocks to complete all-to-all personalized communication (see Figs. 3c and 3d). The following sections describe the algorithm in detail.

3.1.2 Communication Pattern

We now describe the communication pattern of the proposed 2D algorithm, which consists of four phases. In Phase 1, the following operations are performed:

Phase 1:
IF $(r+c) \ mod \ 4 = 0$, P(r,c) $\rightarrow$ P(r, (c+4) $mod \ C$). (1)
IF $(r+c) \ mod \ 4 = 1$, P(r,c) $\rightarrow$ P((r+4) $mod \ R$, c). (2)
IF $(r+c) \ mod \ 4 = 2$, P(r,c) $\rightarrow$ P(r, (c-4) $mod \ C$). (3)
IF $(r+c) \ mod \ 4 = 3$, P(r,c) $\rightarrow$ P((r-4) $mod \ R$, c). (4)

Phase 1 requires $\frac{C}{4} - 1$ steps. Throughout these $\frac{C}{4} - 1$ steps of Phase 1, each node transmits message blocks to a fixed destination node along the direction selected by the node. Consider blocks of a node, say node A, to be scattered to all other nodes. In Step 1, node A transmits all of its blocks except those to be transmitted by itself in Phases 2, 3, and 4, to the next node, say node B, along the direction selected by the nodes. In Step 2, node B extracts blocks to be transmitted by itself in Phases 2, 3, and 4, then transmits the remaining blocks to the next node, say node C, along the direction selected by the nodes. This procedure repeats and in the last step in Phase 1, the last node, say node L, along the direction receives only the blocks to be transmitted by the node in Phases 2, 3, and 4. Likewise, the other nodes scatter their blocks to all nodes in the same node group and in the same column or row. If $R \neq C$, then each node that satisfies the above Conditions 2 and 4 finishes the operations in Phase 1 in $\frac{R}{4} - 1$ steps, and idle or send empty messages during the remaining $\frac{C-R}{4}$ steps.

In Phase 2, all nodes change dimensions then transmit message blocks along the new dimension. In Phase 2, the following operations are performed:

Phase 2:
IF $(r+c) \ mod \ 4 = 0$, P(r,c) $\rightarrow$ P((r+4) $mod \ R$, c). (5)
IF $(r+c) \ mod \ 4 = 1$, P(r,c) $\rightarrow$ P(r, (c+4) $mod \ C$). (6)
IF $(r+c) \ mod \ 4 = 2$, P(r,c) $\rightarrow$ P((r-4) $mod \ R$, c). (7)
IF $(r+c) \ mod \ 4 = 3$, P(r,c) $\rightarrow$ P(r, (c-4) $mod \ C$). (8)

Phase 2 also requires $\frac{C}{4}-1$ steps and the communication pattern is the same as that in Phase 1. Each node in a row or column of Phase 1 (e.g., node A, B, C, ..., L) transmits blocks along a column or row in its new dimension in parallel. In a step, each node extracts blocks for itself and blocks to be transmitted by itself in Phases 3 and 4, then transmits the remaining blocks to the next destination node. Thus, after $\frac{C}{4}-1$ steps of Phase 2, each node has blocks originated from nodes in the same group, destined for itself and to be transmitted by the node in Phases 3 and 4. As in Phase 1, if $R \neq C$ then each node that satisfies the above Conditions 5 and 7 finish the operations in Phase 2 in $\frac{R}{4}-1$ steps and idle or send empty messages during the remaining $\frac{C-R}{4}$ steps.

Now, the network can be divided into $\frac{RC}{16}$ 4×4 submeshes. All nodes in a 4×4 submesh are included in distinct node groups and have blocks originated from nodes in their respective groups. In the next two phases, all-to-all personalized communication operation is performed among nodes within each 4×4 submesh. In Phase 3, the following operations are performed:

Step 1 of Phase 3:
IF $(r+c)\ mod\ 4 = even$ AND $c\ mod\ 4 = 0\ or\ 1$,
P(r,c) → P(r, c+2).
IF $(r+c)\ mod\ 4 = even$ AND $c\ mod\ 4 = 2\ or\ 3$,
P(r,c) → P(r, c-2).
IF $(r+c)\ mod\ 4 = odd$ AND $r\ mod\ 4 = 0\ or\ 1$,
P(r,c) → P(r+2, c).
IF $(r+c)\ mod\ 4 = odd$ AND $r\ mod\ 4 = 2\ or\ 3$,
P(r,c) → P(r-2, c).
Step 2 of Phase 3:
IF $(r+c)\ mod\ 4 = even$ AND $r\ mod\ 4 = 0\ or\ 1$,
P(r,c) → P(r+2, c).
IF $(r+c)\ mod\ 4 = even$ AND $r\ mod\ 4 = 2\ or\ 3$,
P(r,c) → P(r-2, c).
IF $(r+c)\ mod\ 4 = odd$ AND $c\ mod\ 4 = 0\ or\ 1$,
P(r,c) → P(r, c+2).
IF $(r+c)\ mod\ 4 = odd$ AND $c\ mod\ 4 = 2\ or\ 3$,
P(r,c) → P(r, c-2).

In Phase 4, the network is further divided into 2×2 submeshes and two Steps are required as follows:

Step 1 of Phase 4:
IF $c\ mod\ 2 = 0$, P(r,c) → P(r, c+1).
IF $c\ mod\ 2 = 1$, P(r,c) → P(r, c-1).
Step 2 of Phase 4:
IF $r\ mod\ 2 = 0$, P(r,c) → P(r+1, c).
IF $r\ mod\ 2 = 1$, P(r,c) → P(r-1, c).

The next subsection describes the contents of transmitted blocks and the array structure in each communication step.

3.1.3 Data Array

Initially, $P(r,c)$ has RC distinct blocks to distribute to other nodes in 2D array B[u,v], where $0 \leq u \leq R-1$ and $0 \leq v \leq C-1$ if $(r+c)\ mod\ 4 = 0\ or\ 2$ (i.e., nodes that transmit blocks along a row and a column in Phases 1 and 2, respectively), or $0 \leq u \leq C-1$ and $0 \leq v \leq R-1$ if $(r+c)\ mod\ 4 = 1\ or\ 3$ (i.e., nodes that transmit blocks along a column and a row in Phases 1 and 2, respectively). We assume that the array is ordered in column major, and if blocks to be transmitted are not contiguous, then they should be rearranged before transmission. The initial data structure of a node is dependent upon the communication pattern in Phases 1 and 2. A block destined for the node that is u hops away from the node along the direction that the node takes in Phase 1 is located in B[u,0]. In B[u,v], a block destined for the node that is v hops away from the node in B[u,0] along the direction the node takes in Phase 2 is located.

In Step i, $1 \leq i \leq \frac{C}{4}-1$, of Phase 1, each node transmits blocks in columns $4i$ through $C-1$ to its destination node, while receiving the same number of blocks: In Step 1, each node transmits all blocks except those to be transmitted by itself in Phases 2, 3, and 4 (i.e., blocks in the first four columns). Among the blocks received in Step 1, each node extracts the blocks to be transmitted by itself in the following phases (i.e., blocks in the fifth through eighth columns), then transmits the remaining blocks to its destination node in Step 2. This procedure repeats until the last step of Phase 1.

In Step j, $1 \leq j \leq \frac{C}{4}-1$, of Phase 2, each node transmits blocks in rows $4j$ through $C-1$ to its destination node in Phase 2, while receiving the same number of blocks from its source node in Phase 2: In Step 1, each node transmits all blocks except those will be transmitted by itself in Phases 3 and 4 (i.e., blocks in the first four rows). Among the blocks received in Step 1, each node extracts the blocks to be transmitted by itself in Phases 3 and 4 (i.e., blocks in the fifth through eighth rows), then transmits the remaining blocks to its destination node in Step 2. This procedure repeats until the last step of Phase 2.

After Phase 2, each node in a 4×4 submesh has RC blocks originated from all nodes in the same group ($\frac{RC}{16}$ nodes) destined for 16 nodes in the 4×4 submesh to which the node belongs. But blocks destined for each node in the 4×4 submesh are distributed. Thus, before Phase 3, the blocks are rearranged: If we divide a 4×4 submesh into 2×2 submeshes, there are four 2×2 submeshes—one includes a node P (e.g., S0), another includes the partner node in Step1 of Phase 3 (e.g., S1), another includes the partner node in Step 2 of Phase 3 (e.g., S2), and the other submesh (e.g., S3). Blocks destined for S0, S1, S3, and S2 (e.g., B0, B1, B3, and B2, respectively) are arranged in that order in data array of node P. In Step 1 of Phase 3, node P sends blocks destined for S1 and S3 (i.e., B1 and B3) while receiving the same number of blocks, B0 and B2, from the partner node in Step1 of Phase 3. Now, blocks in node P's data array are B0, B0, B2, and B2, in that order. In the next step, node P sends B2's while receiving B0's.

After Phase 3, each node in a 2×2 submesh has RC blocks originated from all nodes in four node groups destined for four nodes in the submesh to which the node belongs, and the blocks are distributed. Thus, before Phase 4, the blocks are rearranged: blocks destined for the node itself (e.g., N0), partner node in Step 1 of Phase 4 (e.g., N1), partner node in Step 2 of Phase 4 (e.g., N2), and the other node (e.g., N3). Blocks destined for N0, N1, N3, and N2 are arranged in that order in the data array of node N0, and the block transmissions in Phase 4 are performed in the

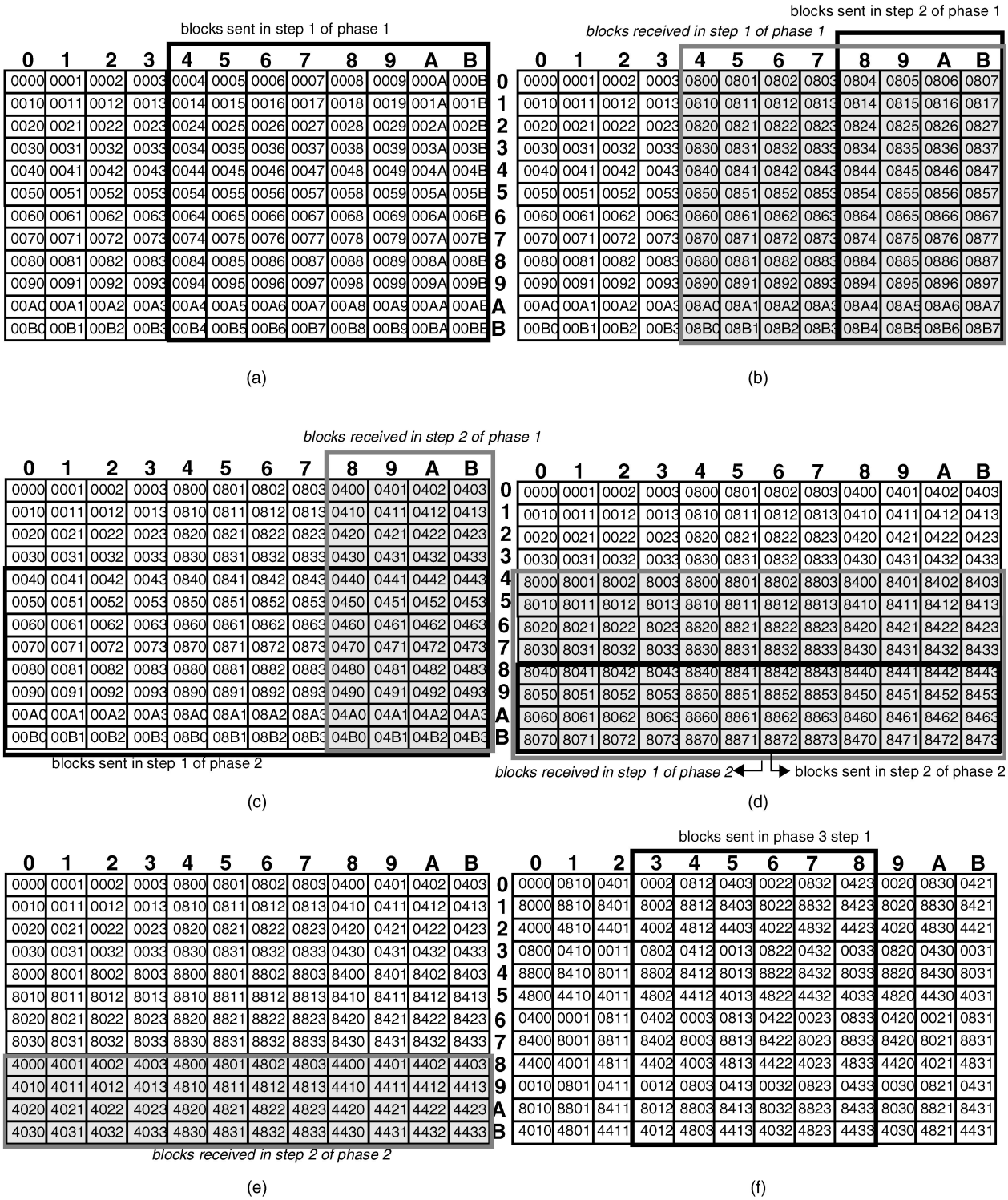

Fig. 4. Data structure of node (0,0) in a 12×12 torus. (a) Initial state, (b) after Step1 of Phase 1, (c) after Step 2 of Phase 1, (d) after Step1 of Phase 2, (e) after Step 2 of Phase 2, and (f) after rearrangement (see the next page for (g)-(k)).

same way as those in Phase 3. After Phase 4, every node has RC blocks, one block from every node in the network.

As an example for the 12×12 torus shown in Fig. 2, let us examine the communication requirements for node P(0,0). The initial block distribution of node P(0,0) is shown Fig. 4a, where each block is identified by the combination of source ID and destination ID. For example, a block that is originated from node P(1,2) and destined for node P(3,4) is identified by block 1234. P(0,0) is included in group 00 and nodes in the group are assigned +c and +r directions in Phases 1 and 2, respectively. Thus, the first row of the data array includes blocks destined for nodes in the same row along the +c direction from P(0,0) in that order, and the other blocks in each column of the data array

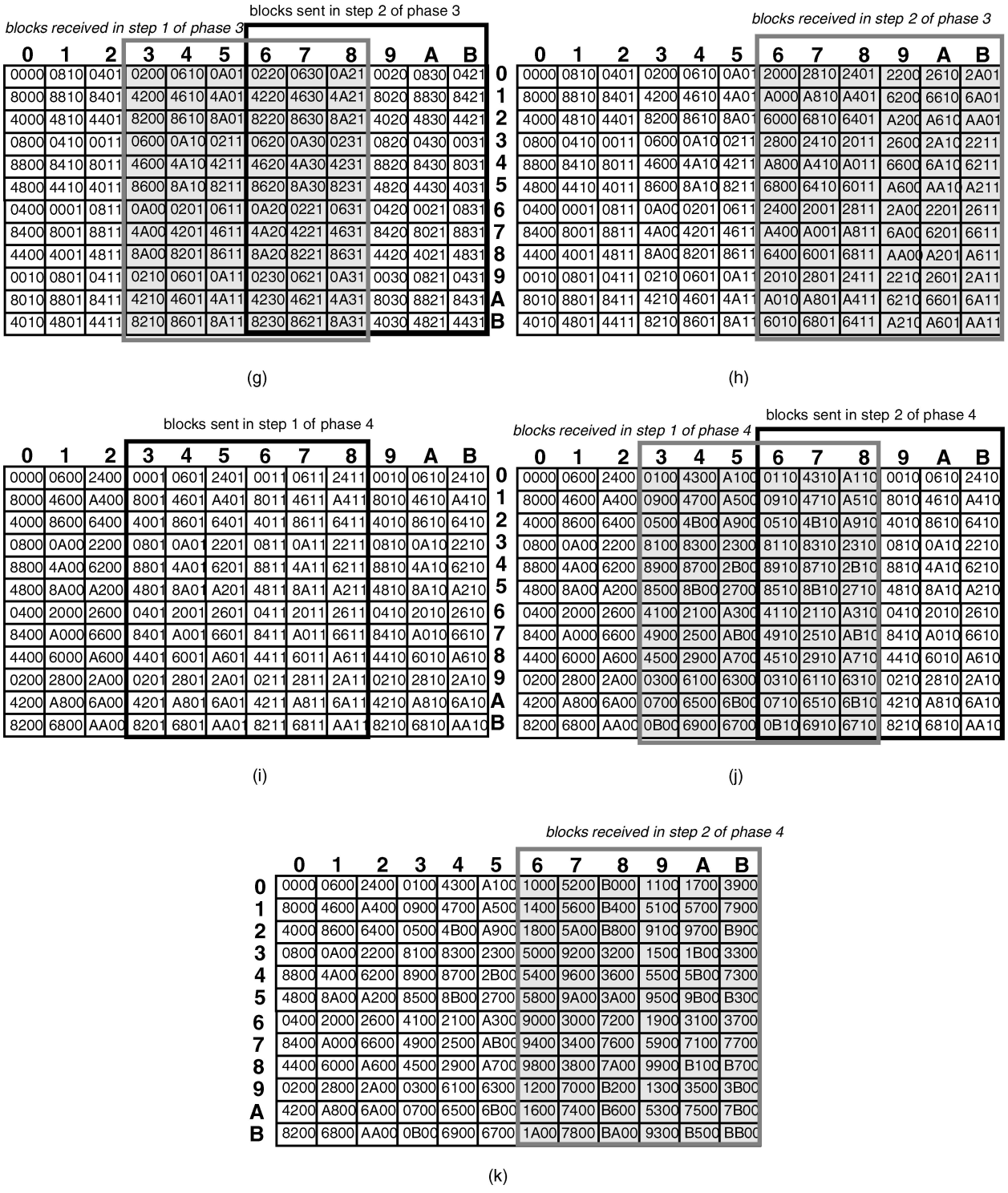

Fig. 4. (Continued from the previous page.) Data structure of node (0,0) in a 12×12 torus. (g) After Step1 of Phase 3, (h) after Step 2 of Phase 3, (i) after rearrangement, (j) after Step1 of Phase 4, and (k) after Step 2 of Phase 4.

include blocks destined for nodes in the same column along the +r direction from each node in the first row, in that order. It is exactly the same configuration as the network itself. It may be easier to understand the initial data structure in each node as follows: The network is reconfigured by placing each node in the origin (0,0) and by making its message transmission operation in Phases 1 and 2 performed along +c and +r, respectively. Then, the initial data structure follows exactly the same configuration as that of the reconfigured network. In Step 1 of Phase 1, P(0,0) sends blocks in columns 4 through B to P(0,4) while receiving the same number of blocks from P(0,8) (Fig. 4b). In Step 2 of Phase 1, P(0,0) sends blocks in the last four columns (Fig. 4b) to P(0,4) while receiving the same number of blocks (Fig. 4c) from P(0,8). In Phase 2, every node in group 00 transmits blocks along +r direction. In Step 1 of

Phase 2, blocks in rows 4 through B are transmitted (Fig. 4c) to P(4,0), while receiving blocks (Fig. 4d) from P(8,0). In Step 2 of Phase 2, blocks in the last four rows (Fig. 4d) are transmitted to P(4,0), while receiving blocks (Fig. 4e) from P(8,0). After Phase 2, all blocks in the array are those destined for nodes in the 4×4 submesh (i.e., P(0,0) through P(3,3)) to which P(0,0) belongs, as shown in Fig. 4e (where the last two digits of block IDs are $00, 01, 02, 03, 10, 11, \ldots, 32, 33$). But blocks destined for nodes in the 4×4 submesh are distributed. Thus, before Phase 3, the blocks are rearranged (considering communication pattern in Phase 3), as shown in Fig. 4f. In Phase 3 Step 1, blocks in the middle six columns (Fig. 4f) are transmitted to P(0,2), while receiving the same number of blocks (Fig. 4g) from P(0,2). In the next step, blocks in the last six columns (Fig. 4g) are transmitted to P(2,0) while receiving the same number of blocks (Fig. 4h). After Phase 3, all blocks in the array are those destined for nodes in the 2×2 submesh (i.e., P(0,0), P(0,1), P(1,0), and P(1,1)) to which P(0,0) belongs as shown in Fig. 4h (the last two digits of block IDs are 00, 01, 10, and 11), but they are distributed. Thus, before Phase 4, the blocks are rearranged (considering the communication pattern in Phase 4), as shown in Fig. 4i. Then, Phase 4 is initiated and the operations are very similar to those in Phase 3. The two steps in Phase 4 are shown in Figs. 4j and 4k. As shown in Fig. 4k, P(0,0) now has all blocks destined for itself from all nodes (the first two digits of block IDs are all distinct, while the last two digits are 00).

As shown above, the proposed 2D algorithm requires data rearrangement steps after Phases 2 and 3. However, a data rearrangement step is also required after Phase 1. After Phase 1, blocks in each node are ordered in row-major order. Logically they are contiguous in row by row, but physically they are not contiguous since we assumed that the array is physically arranged in column-major order. Thus, before the start of Phase 2, the blocks should be rearranged in column-major order.

Formally, the proposed algorithm for all-to-all personalized communication in an $R \times C$ torus can be expressed as shown in Fig. 5.

3.1.4 Complexity Analysis

We now analyze the complexity of the proposed algorithm in terms of startup cost, message-transmission cost, data-rearrangement cost, message propagation cost, and barrier cost.

1. *Startup cost*. For an $R \times C$ 2D torus, $\frac{C}{4} - 1$ steps per phase are required in Phases 1 and 2, and two steps per phase are required in Phases 3 and 4. Thus, a total of $\frac{C}{2} + 2$ steps is required.
2. *Message-transmission cost*. In Step p of Phase 1, where $1 \leq p \leq \frac{C}{4} - 1$, $R(C - 4p)$ blocks (since $R \leq C$) are transmitted. In Step q of Phase 2, where $1 \leq q \leq \frac{C}{4} - 1$, $R(C - 4q)$ blocks are transmitted. In Phases 3 and 4, there are four steps and $\frac{RC}{2}$ blocks are transmitted in each step. Thus, the total number of transmitted blocks is $\frac{RC}{4}(C + 4)$.
3. *Data-rearrangement cost*. At the end of each phase, blocks are rearranged to prepare for the next phase. Since there are four phases, three data-rearrangement steps are required. Thus, the total data-rearrangement cost is $3(RC)m\rho$.
4. *Message propagation cost*. In Phases 1 and 2, there are $\frac{C}{2} - 2$ steps. In each step, the number of hops to the destination is four. In each of two steps in Phases 3 and 4, the number of hops to the destination is two and one, respectively. Thus, the total number of hops is $2C - 2$ and the message propagation cost is expressed as $2(C - 1)t_l$.
5. *Barrier synchronization cost*. Since the proposed algorithm requires $\frac{C}{2} + 2$ steps, the total overhead by barrier synchronization is $\left(\frac{C}{2} + 1\right)t_b$.

In an $N \times N$ torus, we have $\frac{N^2}{2}$ nodes on each side of the network bisection transmitting a block to every node in the other half for a total of $\frac{N^4}{4}$ message blocks crossing the bisection. The bisection consists of $2N$ links. Therefore, we can arrive at a lower bound on message transmission of $O(N^3)$. We see that the proposed algorithm has a time complexity of $O(N^3)$ in terms of message transmission time, assuming $R = C = N$. With respect to startup costs, consider the broadcast of a message to N^2 nodes. In the absence of contention, the optimal number of steps is $\lceil \log_2 N \rceil$ per dimension. Thus, a lower bound on the number of steps to reach all nodes is $O(\log_2 N^2)$. This lower bound is difficult to achieve due to channel contention in one-port architecture. Most all-to-all personalized communication algorithms show a time complexity of $O(N)$ in terms of startup cost, except [13], which shows a time complexity of $O(\log_2 N^2)$.

3.1.5 Correctness of the Proposed Algorithm

Now, we discuss the correctness of the proposed algorithm. First, consider a bidirectional ring of nodes shown in Fig. 6a. If the nodes are divided into two groups, each group can form a unidirectional ring choosing one of the two (positive and negative) directions, and nodes in each unidirectional subring can transmit message blocks to nodes in the same group along the direction without channel contention (see Fig. 6b). If there are an even number (B) of nodes in the ring and each node in a unidirectional subring passes the message that is received in the previous step, then a node's message is distributed to all nodes in the subring (i.e., nodes in the same group) in $\frac{B}{2} - 1$ steps, as shown in Fig. 6c.

Now, consider a 2D torus. In Phases 1 and 2 of the proposed algorithm, there are 16 node groups according to the following rule:

IF $r \bmod 4 = i$ and $c \bmod 4 = j$, P(r,c) is included in group ij.

If we divide the torus into 4×4 contiguous submeshes, each node in a 4×4 submesh is included in one of 16 distinct groups. In Phases 1 and 2, each node in a 4×4 submesh transmits messages to nodes of the same group located in other submeshes along a row or column. Since there are 16 groups and four directions (+r, -r, +c, and -c), four groups should choose the same direction, which may cause channel contention. But, if four nodes that choose the same direction are not located in the same row and column, no channel contention occurs. In the proposed algorithm, it is achieved by Conditions 1 through 4 in Phase 1 and Conditions 5

```
# 2D Algorithm for All-to-All personalized communication for Node P(row_id, col_id)

START
/* Begin Phase 1 */
SWITCH (row_id+col_id)mod 4
    CASE 0: For s = 1 to C/4-1, transmit B[0..R-1,4s..C-1] to node P(row_id, (col_id+4)mod C)
    CASE 1: For s = 1 to C/4-1, transmit B[0..R-1,4s..C-1] to node P((row_id+4)mod R, col_id)
    CASE 2: For s = 1 to C/4-1, transmit B[0..R-1,4s..C-1] to node P(row_id, (col_id-4)mod C)
    CASE 3: For s = 1 to C/4-1, transmit B[0..R-1,4s..C-1] to node P((row_id-4)mod R, col_id)
DATA REARRANGEMENT

/* Begin Phase 2*/
SWITCH (row_id+col_id)mod 4
    CASE 0: For s = 1 to C/4-1, transmit B[0..R-1,4s..C-1] to node P((row_id+4)mod R, col_id)
    CASE 1: For s = 1 to C/4-1, transmit B[0..R-1,4s..C-1] to node P(row_id, (col_id+4)mod C)
    CASE 2: For s = 1 to C/4-1, transmit B[0..R-1,4s..C-1] to node P((row_id-4)mod R, col_id)
    CASE 3: For s = 1 to C/4-1, transmit B[0..R-1,4s..C-1] to node P(row_id, (col_id-4)mod C)
DATA REARRANGEMENT

/* Begin Phase 3 Step 1 */
IF {(row_id+col_id)mod 4 is even} AND {(col_id)mod 4 is 0 OR 1}
    transmit blocks B[o..R-1,C/4..3C/4-1] to node P(row_id, col_id+2)
ELSE IF {(row_id+col_id)mod 4 is even} AND {(col_id)mod 4 is 2 OR 3}
    transmit blocks B[o..R-1,C/4..3C/4-1] to node P(row_id, col_id -2)
ELSE IF {(row_id+col_id)mod 4 is odd} AND {(col_id)mod 4 is 0 OR 1}
    transmit blocks B[o..R-1,C/4..3C/4-1] to node P(row_id+2, col_id)
ELSE {(row_id+col_id)mod 4 is odd} AND {(col_id)mod 4 is 2 or 3}
    transmit blocks B[o..R-1,C/4..3C/4-1] to node P(row_id-2, col_id)

/* Begin Phase 3 Step 2*/
IF {(row_id+col_id)mod 4 is even} AND {(col_id)mod 4 is 0 OR 1}
    transmit blocks B[0..R-1,C/2..C-1] to node P(row_id+2, col_id)
ELSE IF {(row_id+col_id)mod 4 is even} AND {(col_id)mod 4 is 2 OR 3}
    transmit blocks B[0..R-1,C/2..C-1] to node P(row_id-2, col_id)
ELSE IF {(row_id+col_id)mod 4 is odd} AND {(col_id)mod 4 is 0 OR 1}
    transmit blocks B[0..R-1,C/2..C-1] to node P(row_id, col_id+2)
ELSE {(row_id+col_id)mod 4 is odd} AND {(col_id)mod 4 is 2 or 3}
    transmit blocks B[0..R-1,C/2..C-1] to node P(row_id, col_id-2)

DATA REARRANGEMENT

/* Begin Phase 4 Step 1*/
IF my_column_id is even
    transmit blocks B[o..R-1,C/4..3C/4-1] to node P(row_id, col_id+1)
ELSE
    transmit blocks B[o..R-1,C/4..3C/4-1] to node P(row_id, col_id-1)

/* Begin Phase 4 Step 2*/
IF my_row_id is even
    transmit blocks B[0..R-1,C/2..C-1] to node P(row_id+1, col_id)
ELSE
    transmit blocks B[0..R-1,C/2..C-1] to node P(row_id-1, col_id)

END
```

Fig. 5. A formal description of a 2D algorithm for an $R \times C$ torus.

through 8 in Phase 2. Thus, every node can transmit messages without channel contention in Phases 1 and 2.

Now, consider data received by an arbitrary node $P(x_p \dots x_0, y_q \dots y_0)$. Let the data that originated from node $P(x_p \dots x_0, y_q \dots y_0)$ be $D[x_p \dots x_0, y_q \dots y_0]$. In Phase 1, nodes in the same group and in the same row or column transmit message blocks along a subring, just as shown in Fig. 6c. Since there are $\frac{C}{4}$ or $\frac{R}{4}$ nodes in a subring, a message

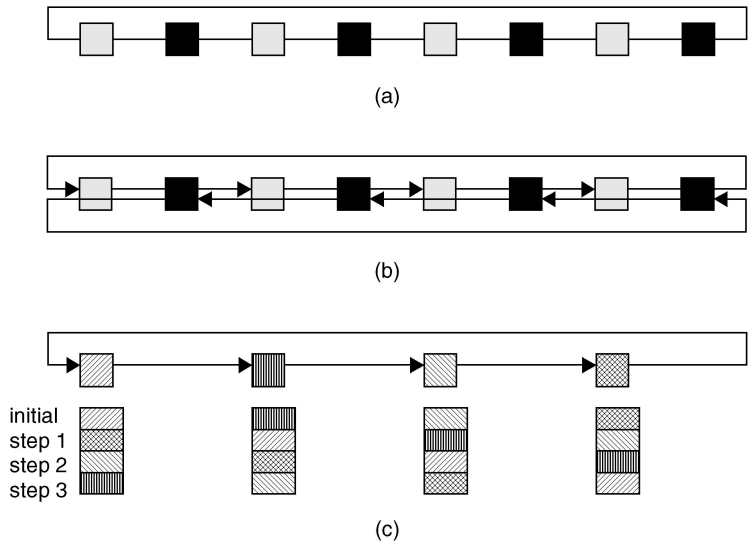

Fig. 6. Message propagation in a linear ring. (a) Bidirectional ring, (b) two unidirectional subrings, and (c) message propagation.

from a node is scattered to all nodes in the same subring in $\frac{C}{4}-1$ steps (recall $C \geq R$). After Phase 1, there are two possibilities as follows:

IF $(r+c)\ mod\ 4 = 0\ or\ 2$, then node P has data $D[x_p \ldots x_0, F \ldots F y_1 y_0]$, where F indicates don't care.
IF $(r+c)\ mod\ 4 = 1\ or\ 3$, then node P has data $D[F \ldots F x_1 x_0, y_q \ldots y_0]$.

In Phase 2, each node changes dimensions and repeats the same operation. After Phase 2, node P has data $D[F \ldots F x_1 x_0, F \ldots F y_1 y_0]$.

In the next four steps of Phases 3 and 4, node P exchanges message blocks with nodes in distinct node groups. After Phase 3 and Phase 4, node P has data $D[F \ldots F F x_0, F \ldots F F y_0]$ and $D[F \ldots F F F, F \ldots F F F]$, respectively.

3.2 *n*-Dimensional Tori

The algorithm for 2D tori can be extended to *n*-dimensional tori in a straightforward manner. Before describing the general *n*-dimensional algorithm, it is helpful to describe a 3D algorithm.

3.2.1 Algorithm for 3D Tori

For an $N_1 \times N_2 \times N_3$ 3D torus, where N_1, N_2, and N_3 are a multiple of four and $N_1 \geq N_2 \geq N_3$, each node is identified by a label $P(x, y, z)$, where $0 \leq x \leq N_1 - 1$, $0 \leq y \leq N_2 - 1$, and $0 \leq z \leq N_3 - 1$. Each node is included in one of 64 node groups, according to the following rule:

IF $x\ mod\ 4 = i, y\ mod\ 4 = j, and\ z\ mod\ 4 = k$, node P(x,y,z) is included in group ijk.

Communication Pattern. The proposed algorithm requires five phases. In Phase 1, the following operations are performed:

Phase 1:
IF $(x+y)\ mod\ 4 = 0$ and $z\ mod\ 4 = 0\ or\ 2$, then P(x,y,z) transmits to P((x+4) $mod\ N_1$,y,z).
IF $(x+y)\ mod\ 4 = 1$ and $z\ mod\ 4 = 0\ or\ 2$, then P(x,y,z) transmits to P(x,(y+4) $mod\ N_2$,z).
IF $(x+y)\ mod\ 4 = 2$ and $z\ mod\ 4 = 0\ or\ 2$, then P(x,y,z) transmits to P((x-4) $mod\ N_1$,y,z).
IF $(x+y)\ mod\ 4 = 3$ and $z\ mod\ 4 = 0\ or\ 2$, then P(x,y,z) transmits to P(x,(y-4) $mod\ N_2$,z).
IF $z\ mod\ 4 = 1$, then P(x,y,z) transmits to P(x,y,(z+4) $mod\ N_3$).
IF $z\ mod\ 4 = 3$, then P(x,y,z) transmits to P(x,y,(z-4) $mod\ N_3$).

The communication pattern of Phase 1 in a 2D torus (*pattern A*) is performed in even numbered X-Y planes, while interplane communication operations (*pattern C*) are performed among nodes in odd numbered planes. For example, the communication pattern in Phase 1 for a $12 \times 12 \times 12$ torus is illustrated in Fig. 7a. There are $\frac{N_1}{4} - 1$ steps in Phase 1.

In Phase 2, the following operations are performed:

Phase 2:
IF $(x+y)\ mod\ 4 = 0$, then P(x,y,z) transmits to P(x,(y+4) $mod\ N_2$,z).
IF $(x+y)\ mod\ 4 = 1$, then P(x,y,z) transmits to P((x+4) $mod\ N_1$,y,z).
IF $(x+y)\ mod\ 4 = 2$, then P(x,y,z) transmits to P(x,(y-4) $mod\ N_2$,z).
IF $(x+y)\ mod\ 4 = 3$, then P(x,y,z) transmits to P((x-4) $mod\ N_1$,y,z).

In Phase 2, the communication pattern of Phase 2 in a 2D torus (*pattern B*) is performed in even numbered X-Y planes, while the communication pattern of Phase 1 in a 2D torus (*pattern A*) is performed among nodes in odd numbered planes. Fig. 7b shows the communication pattern in Phase 2 for a 12×12 torus. There are also $\frac{N_1}{4} - 1$ steps in Phase 2.

In Phase 3, the following operations are performed:

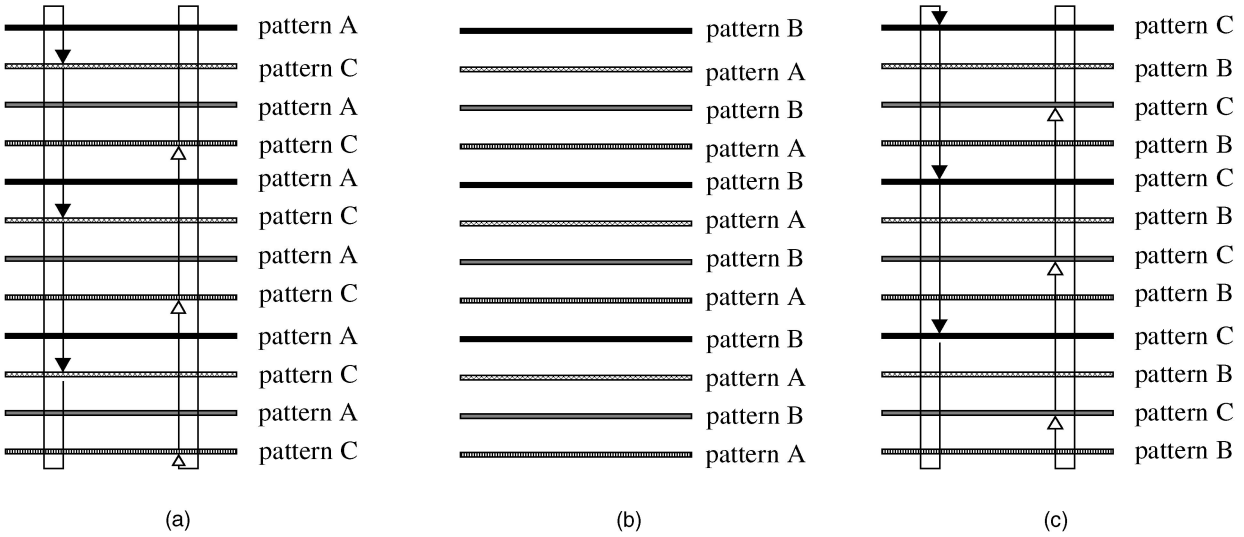

Fig. 7. Communication pattern in Phases 1, 2, and 3 for a $12 \times 12 \times 12$ torus. (a) Phase 1, (b) Phase 2, and (c) Phase 3.

Phase 3:
IF $(x+y)\ mod\ 4 = 0$ and $z\ mod\ 4 = 1\ or\ 3$, then P(x,y,z) transmits to P((x+4) $mod\ N_1$,y,z).
IF $(x+y)\ mod\ 4 = 1$ and $z\ mod\ 4 = 1\ or\ 3$, then P(x,y,z) transmits to P(x,(y+4) $mod\ N_2$,z).
IF $(x+y)\ mod\ 4 = 2$ and $z\ mod\ 4 = 1\ or\ 3$, then P(x,y,z) transmits to P((x-4) $mod\ N_1$,y,z).
IF $(x+y)\ mod\ 4 = 3$ and $z\ mod\ 4 = 1\ or\ 3$, then P(x,y,z) transmits to P(x,(y-4) $mod\ N_2$,z).
IF $z\ mod\ 4 = 0$, then P(x,y,z) transmits to P(x,y, (z+4) $mod\ N_3$).
IF $z\ mod\ 4 = 2$, then P(x,y,z) transmits to P(x,y, (z-4) $mod\ N_3$).

In Phase 3, nodes in even numbered planes follow *pattern C* while nodes in the other planes follow *pattern B*, as illustrated in Fig. 7c. In Phase 3, there are also $\frac{N_1}{4} - 1$ steps.

After Phase 3, the network is divided into $\frac{N_1 N_2 N_3}{4^3}$ $4 \times 4 \times 4$ submeshes. Now, Phase 4 starts and consists of three steps: The following operations are performed in each step of Phase 4 (see Figs. 8a, 8b, and 8c for a $12 \times 12 \times 12$ torus, where only one $4 \times 4 \times 4$ submesh is shown):

Step 1 of Phase 4:
IF $(x+y)\ mod\ 2 = 0$, $y\ mod\ 4 = 0\ or\ 1$, and $z\ mod\ 2 = 0$, then P(x,y,z) transmits to P(x+2,y,z).
IF $(x+y)\ mod\ 2 = 0$, $y\ mod\ 4 = 2\ or\ 3$, and $z\ mod\ 2 = 0$, then P(x,y,z) transmits to P(x-2,y,z).
IF $(x+y)\ mod\ 2 = 1$, $x\ mod\ 4 = 0\ or\ 1$, and $z\ mod\ 2 = 0$, then P(x,y,z) transmits to P(x,y+2,z).
IF $(x+y)\ mod\ 2 = 1$, $x\ mod\ 4 = 2\ or\ 3$, and $z\ mod\ 2 = 0$, then P(x,y,z) transmits to P(x,y-2,z).
IF $z\ mod\ 4 = 1$, then P(x,y,z) transmits to P(x,y,z+2).
IF $z\ mod\ 4 = 3$, then P(x,y,z) transmits to P(x,y,z-2).

Step 2 of Phase 4:
IF $(x+y)\ mod\ 2 = 0$ and $x\ mod\ 4 = 0\ or\ 1$, then P(x,y,z) transmits to P(x,y+2,z).
IF $(x+y)\ mod\ 2 = 0$ and $x\ mod\ 4 = 2\ or\ 3$, then P(x,y,z) transmits to P(x,y-2,z).
IF $(x+y)\ mod\ 2 = 1$ and $y\ mod\ 4 = 0\ or\ 1$, then P(x,y,z) transmits to P(x+2,y,z).
IF $(x+y)\ mod\ 2 = 1$ and $y\ mod\ 4 = 2\ or\ 3$, then P(x,y,z) transmits to P(x-2,y,z).

Step 3 of Phase 4:
IF $(x+y)\ mod\ 2 = 0$, $y\ mod\ 4 = 0\ or\ 1$, and $z\ mod\ 2 = 1$, then P(x,y,z) transmits to P(x+2,y,z).
IF $(x+y)\ mod\ 2 = 0$, $y\ mod\ 4 = 2\ or\ 3$, and $z\ mod\ 2 = 1$, then P(x,y,z) transmits to P(x-2,y,z).
IF $(x+y)\ mod\ 2 = 1$, $x\ mod\ 4 = 0\ or\ 1$, and $z\ mod\ 2 = 1$, then P(x,y,z) transmits to P(x,y+2,z).
IF $(x+y)\ mod\ 2 = 1$, $x\ mod\ 4 = 2\ or\ 3$, and $z\ mod\ 2 = 1$, then P(x,y,z) transmits to P(x,y-2,z).
IF $z\ mod\ 4 = 0$, then P(x,y,z) transmits to P(x,y,z+2).
IF $z\ mod\ 4 = 2$, then P(x,y,z) transmits to P(x,y,z-2).

After Phase 4, the network is further divided into $\frac{N_1 N_2 N_3}{8}$ $2 \times 2 \times 2$ submeshes. Now, Phase 5 is initiated and there are three steps. In each step, every node exchanges messages along X-, Y-, and Z-dimensions, respectively (see Figs. 8d, 8e, and 8f for a $12 \times 12 \times 12$ torus, where only one $2 \times 2 \times 2$ submesh is shown). That is, the following operations are performed in each step of Phase 5:

Step 1 of Phase 5:
IF $x\ mod\ 2 = 0$, then P(x,y,z) transmits to P(x+1,y,z).
IF $x\ mod\ 2 = 1$, then P(x,y,z) transmits to P(x-1,y,z).

Step 2 of Phase 5:
IF $y\ mod\ 2 = 0$, then P(x,y,z) transmits to P(x,y+1,z).
IF $y\ mod\ 2 = 1$, then P(x,y,z) transmits to P(x,y-1,z).

Step 3 of Phase 5:
IF $z\ mod\ 2 = 0$, then P(x,y,z) transmits to P(x,y,z+1).
IF $z\ mod\ 2 = 1$, then P(x,y,z) transmits to P(x,y,z-1).

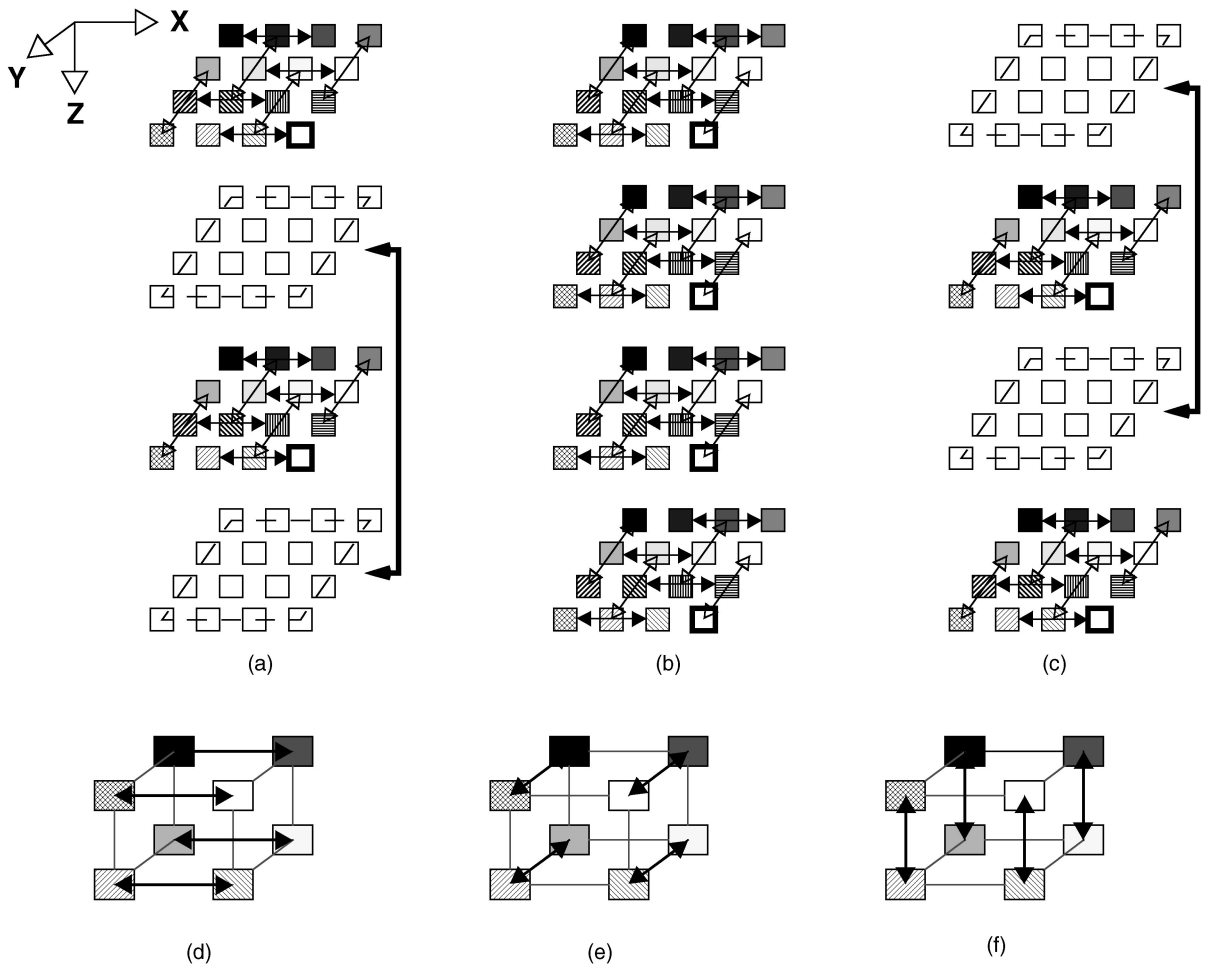

Fig. 8. Communication pattern in Phases 4 and 5 for a $12 \times 12 \times 12$ torus. (a) Phase 4 Step 1, (b) Phase 4 Step 2, (c) Phase 4 Step 3, (d) Phase 5 Step 1, (e) Phase 5 Step 2, and (f) Phase 5 Step 3.

Data Array. Now, consider the data array of each node. Initially, each node has $N_1N_2N_3$ distinct blocks in a 3D array B[u,v,w], where $0 \leq u \leq N_1 - 1$, $0 \leq v \leq N_2 - 1$, and $0 \leq w \leq N_3 - 1$. Since the data array structure in 3D tori is very similar to that in 2D tori and can be easily extended, we just examine the communication requirements in node P(0,0,0). In Step 1 of Phase 1, P(0,0,0) sends to P(4,0,0) blocks B[4..N_1–1, $\star$, $\star$], while receiving the same number of blocks from node P($N_1 - 4$,0,0). The notation B[4..N_1–1, $\star$, $\star$] identifies all blocks from B[4,0,0] to B[N_1–1, N_2–1, N_3–1]. In the next Step, P(0,0,0) transmits blocks B[8..N_1–1, $\star$, $\star$] to P(4,0,0). In general, in Step s_1 of Phase 1, $1 \leq s_1 \leq \frac{N_1}{4} - 1$, P(0,0,0) transmits blocks B[4s_1..N_1–1, $\star$, $\star$]. In Step s_2 of Phase 2, $1 \leq s_2 \leq \frac{N_2}{4} - 1$, P(0,0,0) transmits blocks B[$\star$, 4s_2..N_2–1,$\star$] to P(0,4,0). In Step s_3 of Phase 3, $1 \leq s_3 \leq \frac{N_3}{4} - 1$, P(0,0,0) transmits blocks B[$\star$, $\star$, 4s_3..N_3–1] to P(0,0,4). The blocks transmitted by node P(0,0,0) in each step of Phases 1, 2, and 3 in a $12 \times 12 \times 12$ torus are shown in Fig. 9. After Phase 3, blocks originated from nodes in the same group destined for nodes in the $4 \times 4 \times 4$ submesh which includes P(0,0,0) are gathered in P(0,0,0). Thus, in six steps in Phases 4 and 5, the blocks destined for the other nodes in the $4 \times 4 \times 4$ submesh are transmitted.

3.2.2 Extension to n-Dimensional Tori

Now, we describe a general n-dimensional algorithm. Since the extension for n-dimensional tori can be made similarly to the 2D-to-3D extension, we describe the n-dimensional algorithm briefly in this section.

For an $N_1 \times \ldots \times N_n$ n-dimensional tori, where $N_1, \ldots, N_n$ are a multiple of four and $N_1 \geq \ldots \geq N_n$, there are $n + 2$ phases. In the first n phases, messages are transmitted among nodes in the same group which form an $\frac{N_1}{4} \times \ldots \times \frac{N_n}{4}$ subtorus. To avoid channel contention, the dimensions in which messages are transmitted are distributed in each phase. In general, for n-dimensional tori, nodes in the even-numbered unit along dimension n follow the communication patterns of $(n-1)$-dimensional networks during the first $n - 1$ phases and then perform communications along the last dimension (i.e., dimension n) in phase n, while the other nodes perform the communications along the dimension n in Phase 1 and then follow the communications of $(n-1)$-dimensional networks

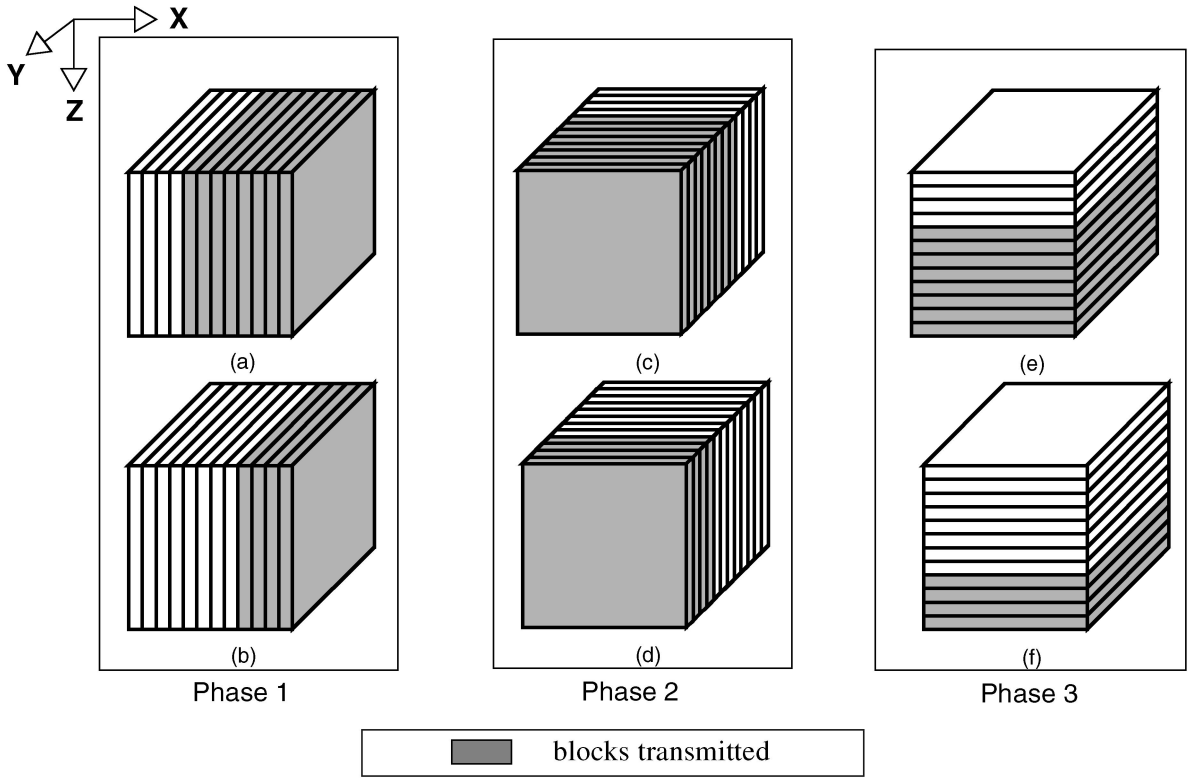

Fig. 9. Blocks transmitted in each step of Phases 1, 2, and 3 for a $12 \times 12 \times 12$ torus. (a) Step 1 Phase 1, (b) Step 2 Phase 1, (c) Step 1 Phase 2, (d) Step 2 Phase 2, (e) Step 1 Phase 3, and (f) Step 2 Phase 3.

during the remaining $n-1$ phases. In Phases $n+1$ and $n+2$, message exchanges are performed among nodes in $4 \times \ldots \times 4$ and $2 \times \ldots \times 2$ n-dimensional submeshes, respectively.

3.2.3 Complexity Analysis

Just as with 2D tori, we analyze the time costs required by the proposed n-dimensional algorithm in terms of startup cost, message-transmission cost, data-rearrangement cost, message propagation cost, and barrier cost.

1. *Startup cost*. For an $N_1 \times \ldots \times N_n$ n-dimensional torus, $N_1 \geq \ldots \geq N_n$, there are $n+2$ phases. In the first n phases, $\frac{N_1}{4}-1$ steps per phase are required. In Phases $n+1$ and $n+2$, n steps are required in each phase. Thus, a total of $n\left(\frac{N_1}{4}+1\right)$ steps is required.
2. *Message-transmission cost*. In Step s, $1 \leq s \leq \frac{N_1}{4}-1$, in each of the first n phases, $(N_1-4s)(N_2 \ldots N_n)$ blocks are transmitted (since $N_1 \geq \ldots \geq N_n$). In each step of Phases $n+1$ and $n+2$, $\frac{1}{2}(N_1 \ldots N_n)$ blocks are transmitted. Thus, the total number of transmitted blocks is $\frac{n}{8}(N_1+4)(N_1 \ldots N_n)$.
3. *Data-rearrangement cost*. At the end of each phase, blocks are rearranged to prepare for the next phase. Since there are $n+2$ phases, $n+1$ data-rearrangement steps are required. Thus, the total data-rearrangement cost is $(n+1)(N_1 \ldots N_n)m\rho$.
4. *Message propagation cost*. In the first n phases, there are $\frac{N_1}{4}-1$ steps per phase. In each step, the number of hops to the destination is four. In Phases $n+1$ and $n+2$, n steps are required in each phase and the number of hops to the destination is two and one, respectively. Thus, the total message propagation cost is $n(N_1-1)t_l$.
5. *Barrier synchronization cost*. Since the proposed algorithm requires $n\left(\frac{N_1}{4}+1\right)$ steps, the total overhead by barrier synchronization is $\left(\frac{nN_1}{4}+n-1\right)t_b$.

4 Algorithms for Meshes

In the previous section, all-to-all personalized communication algorithms for torus-connected machines were described. Since tori are meshes with wrap-around channels, the all-to-all personalized communication algorithms for meshes are very similar to those for tori and can be easily derived. Since the basic idea is very similar to that of tori, this section briefly describes all-to-all personalized communication algorithms for multidimensional meshes.

4.1 2D Meshes

For an $R \times C$ mesh, where R and C are even numbers and $R \leq C$, each node is identified by a label $P(r,c)$, $0 \leq r \leq R-1$ and $0 \leq c \leq C-1$. Each node is included in one of four node groups according to the following rules:

IF r and c are even, P(r,c) is included in group EE.
IF r is odd and c is even, P(r,c) is included in group OE.
IF r is even and c is odd, P(r,c) is included in group EO.
IF r and c are odd, P(r,c) is included in group OO.

The nodes in a group form an $\frac{R}{2} \times \frac{C}{2}$ submesh. For example, in the 6×6 mesh shown in Fig. 10, nine nodes of

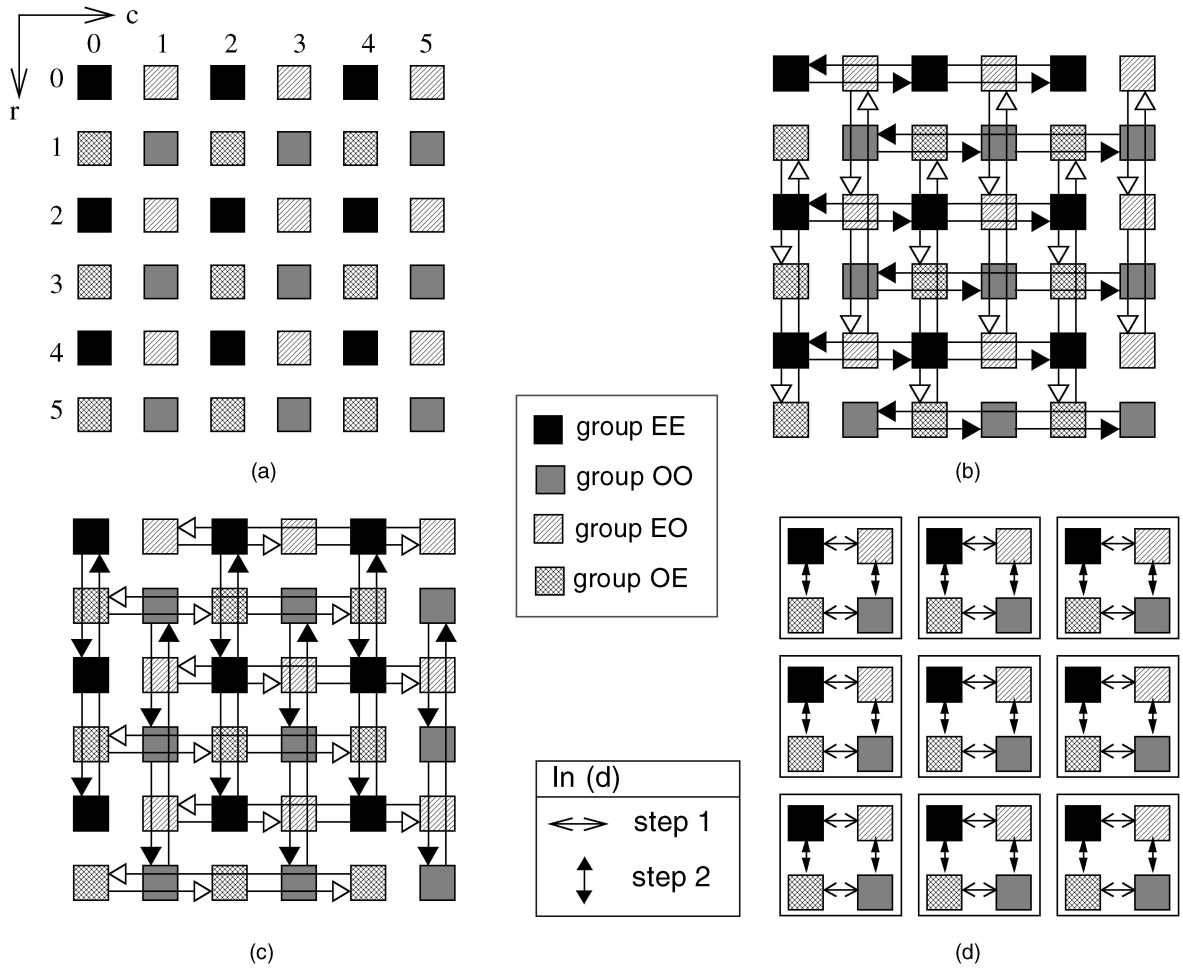

Fig. 10. Communication pattern in a 6×6 mesh. (a) Four node groups, (b) Phase 1, (c) Phase 2, and (d) Phase 3.

identical marking form a 3×3 submesh. The proposed algorithm consists of three phases. In Phases 1 and 2, messages are exchanged among nodes in the same group. Nodes in each group and in the same row/column form a logical ring. For example, in the 6×6 mesh shown in Figs. 10b and 10c, three nodes in each row/column form a logical ring. Phases 1 and 2 require $\frac{C}{2} - 1$ steps per phase and the following operations are performed in each step of Phases 1 and 2:

Phase 1:
IF $P(r, c) \in EE \; or \; OO$, then P(r,c) transmits to P(r, (c+2) $mod \; C$).
IF $P(r, c) \in EO \; or \; OE$, then P(r,c) transmits to P((r+2) $mod \; R, \; c$).
Phase 2:
IF $P(r, c) \in EE \; or \; OO$, then P(r,c) transmits to P((r+2) $mod \; R, \; c$).
IF $P(r, c) \in EO \; or \; OE$, then P(r,c) transmits to P(r, (c+2) $mod \; C$).

After Phase 2, the network is divided into $\frac{RC}{4}$ contiguous 2×2 submeshes and all of the four nodes in a 2×2 submesh are included in distinct node groups (see Fig. 10d). In Phase 3, messages are transmitted by nodes in distinct groups and in the same 2×2 submesh to complete the all-to-all personalized communication, as shown in Fig. 10d. In Phase 3, two steps are required and the following operations are performed in each step:

Step 1 of Phase 3:
IF $P(r, c) \in EE \; or \; OE$, then P(r,c) transmits to P(r, c+1).
IF $P(r, c) \in OO \; or \; EO$, then P(r,c) transmits to P(r, c-1).
Step 2 of Phase 3:
IF $P(r, c) \in EE \; or \; OE$, then P(r,c) transmits to P(r+1, c).
IF $P(r, c) \in OO \; or \; EO$, then P(r,c) transmits to P(r-1, c).

After Phase 3, each node has RC blocks originated from all nodes in the network to complete the all-to-all personalized communication.

Figs. 10 and 11 show the communication pattern and data array of the proposed 2D algorithm in a 6×6 mesh, respectively.

Complexity Analysis.

1. *Start-up cost*. For an $R \times C$ 2D mesh, $R \leq C$, there are $\frac{C}{2} - 1$ steps in Phase 1, $\frac{C}{2} - 1$ steps in Phase 2, and two steps in Phase 3. Thus, a total of C steps are required.

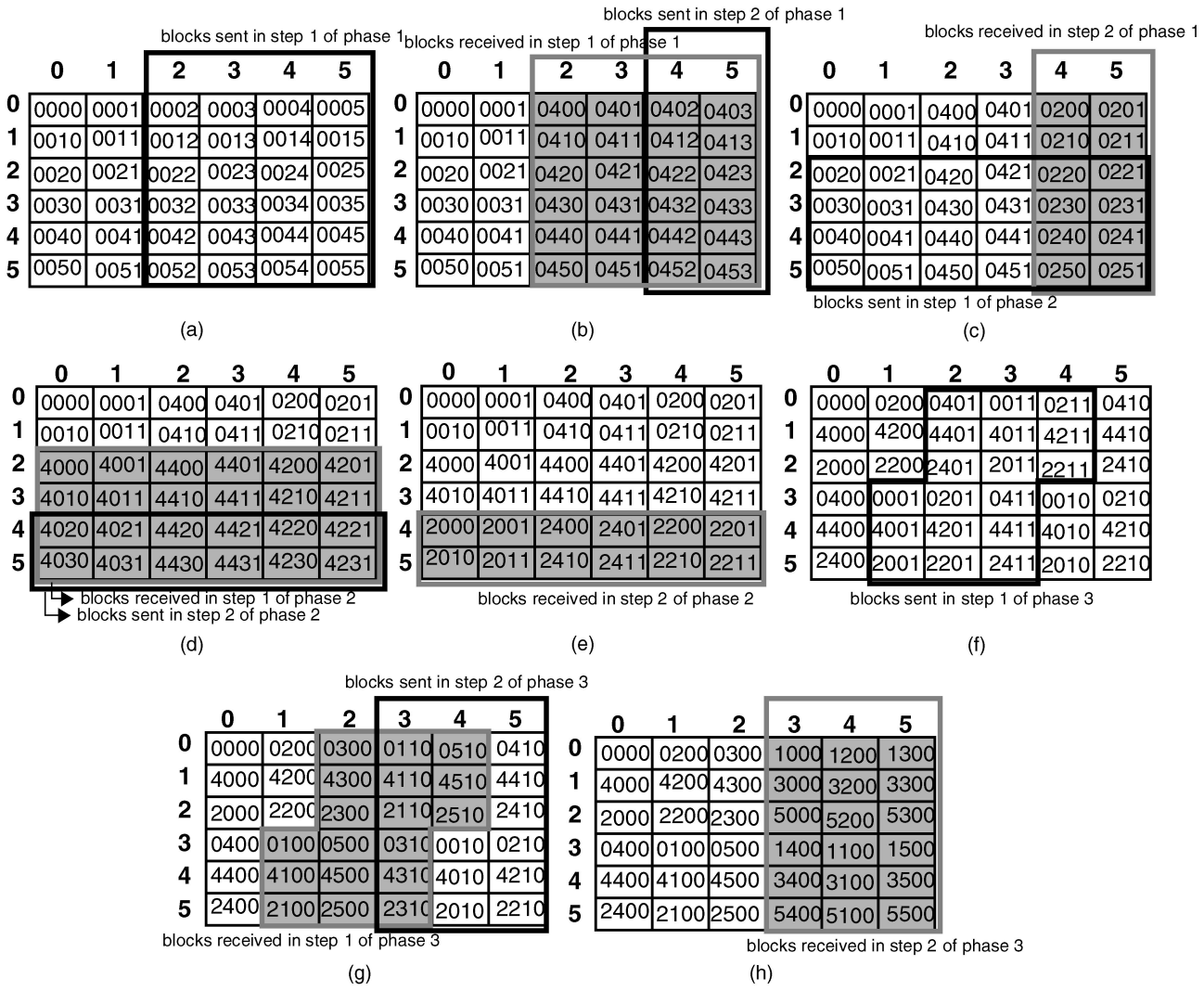

Fig. 11. Data structure of node (0,0) in a 6×6 mesh. (a) Initial state, (b) after Step 1 of Phase 1, (c) after Step 2 of Phase 1, (d) after Step1 of Phase 2, (e) after Step 2 of Phase 2, (f) after rearrangement, (g) after Step 1 of Phase 3, and (h) after Step 2 of Phase 3.

2. *Message transmission cost*. In step p of Phase 1, where $1 \leq p \leq \frac{C}{2} - 1$, $R(C - 2p)$ blocks (since $R \leq C$) are transmitted. In Step q of Phase 2, where $1 \leq q \leq \frac{C}{2} - 1$, $R(C - 2q)$ blocks are transmitted. In Phase 3, there are two steps and $\frac{RC}{2}$ blocks are transmitted in each step. Thus, the total number of transmitted blocks is $\frac{RC^2}{2}$.
3. *Data rearrangement cost*. Since there are three phases, two message-rearrangement steps are required. Thus, the total data rearrangement cost is $2(RC)m\rho$.
4. *Message propagation cost*. In the first two phases, there are $\frac{C}{2} - 1$ steps per phase. In each step, the largest number of hops to the destination is $C - 2$. In Phase 3, two steps are required and the number of hops to the destination is one in each step. Thus, message propagation cost is expressed as $\{(C-2)^2 + 2\}t_l$.
5. *Barrier synchronization cost*. Since the proposed algorithm requires C steps, the total overhead by barrier synchronization is $(C - 1)t_b$.

4.2 Extension to Multidimensional Meshes

Just as tori, the algorithm for 2D meshes can be extended to $N_1 \times \ldots \times N_n$ n-dimensional meshes in a straightforward manner. In this case, there are $n + 1$ phases and the message transmissions are performed along logical rings in each dimension in each of the first n phases. In the last phase (phase $n + 1$), message exchange operations are performed in each $2 \times 2 \times \ldots \times 2$ n-dimensional submesh.

Fig. 12 illustrates the communication pattern in a $6 \times 6 \times 6$ mesh.

Complexity Analysis.

1. *Startup cost*. For an $N_1 \times \ldots \times N_n$ n-dimensional mesh, $N_1 \geq \ldots \geq N_n$, there are $n + 1$ phases. In the first n phases, $\frac{N_1}{2} - 1$ steps per phase are required. In phases $n + 1$, n steps are required. Thus, a total of $\frac{nN_1}{2}$ steps is required.
2. *Message-transmission cost*. In Step s, $1 \leq s \leq \frac{N_1}{2} - 1$, in each of the first n phases, $(N_1 - 2s)(N_2 \ldots N_n)$ blocks

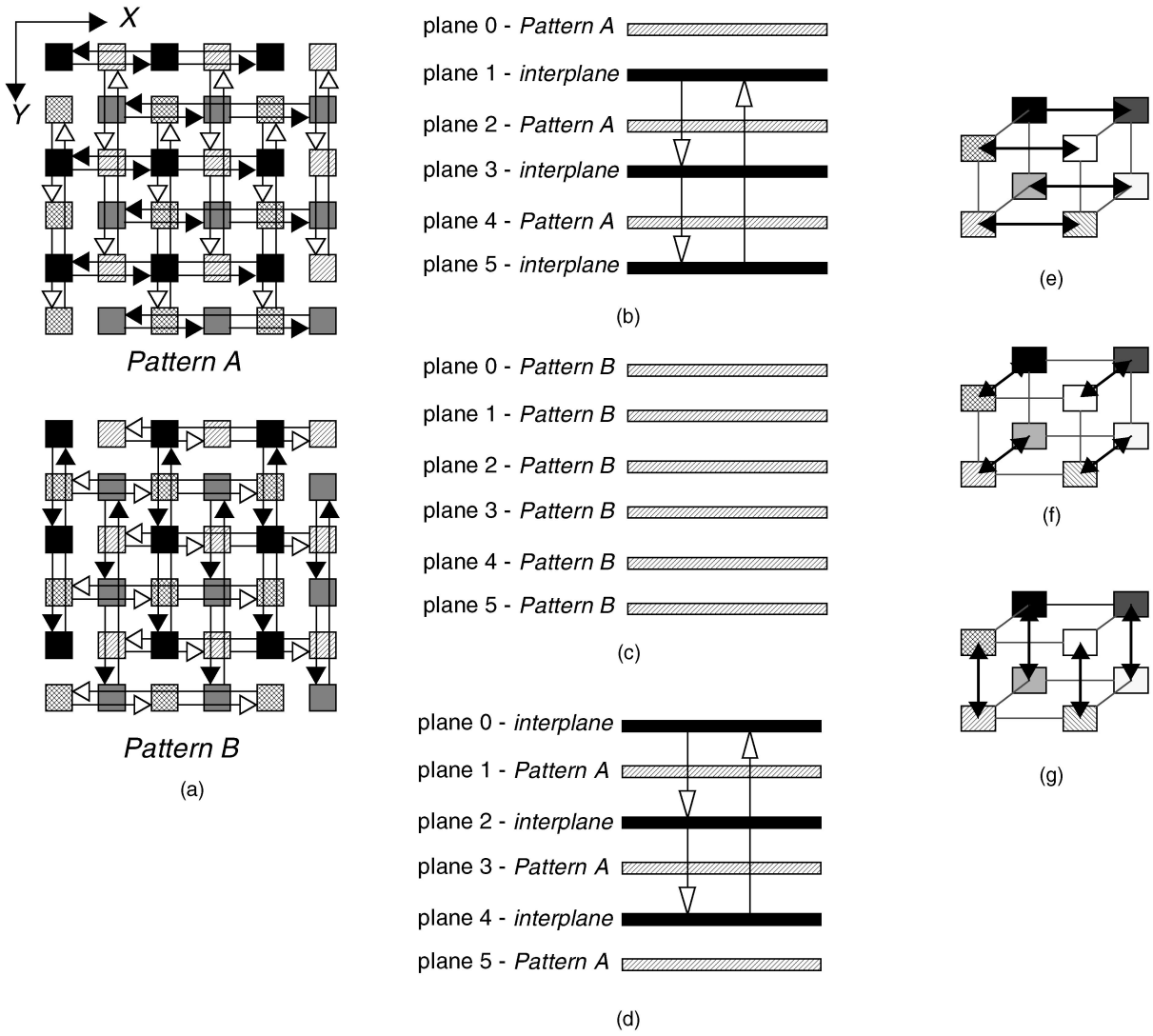

Fig. 12. Communication pattern in a $6 \times 6 \times 6$ mesh. (a) Communication patterns in X-Y planes, (b) Phase 1, (c) Phase 2, (d) Phase 3, (e) Phase 4 Step 1, (f) Phase 4 Step 2, and (g) Phase 4 Step 3.

are transmitted. In each step of Phase $n+1$, $\frac{1}{2}(N_1 N_2 \ldots N_n)$ blocks are transmitted. Thus, the total number of transmitted blocks is $\frac{n}{4}(N_1^2 N_2..N_n)$.

3. *Data-rearrangement cost*. At the end of each phase blocks are rearranged to prepare for the next phase. Since there are $n+1$ phases, n data-rearrangement steps are required. Thus, the total data-rearrangement cost is $n(N_1 N_2 \ldots N_n)m\rho$.
4. *Message propagation cost*. In the first n phases, there are $\frac{N_1}{2} - 1$ steps per phase. In each step, the number of hops to the destination is $N_1 - 2$. In phase $n+1$, n steps are required and the number of hops to the destination is one in each step. Thus, message propagation cost is expressed as $\frac{n}{2}\{(N_1 - 2)^2 + 2\}t_l$.
5. *Barrier synchronization cost*. Since the proposed algorithm requires $\frac{nN_1}{2}$ steps, the total overhead by barrier synchronization is $\left(\frac{nN_1}{2} - 1\right)t_b$.

5 Performance Evaluation

Thus far, we analyzed the time cost required by the proposed algorithm in terms of dominant components in completion time: startup cost, message transmission cost, data rearrangement cost, message propagation cost, and barrier synchronization cost. In this section, the performance of the proposed algorithms is evaluated and compared with that of existing algorithms.

The time complexities of the proposed algorithms are summarized in Table 1. We are not aware of any existing message-combining algorithms for n-dimensional tori or meshes, where the number of nodes in each dimension is non-power-of-two. For 2D tori or meshes, Tseng et al. [19], Sundar et al. [16], Suh and Yalamanchili [12], [13] proposed all-to-all personalized communication algorithms using message-combining. In these algorithms, networks are assumed to be power-of-two and square, that is, there are 2^d nodes in both dimensions. If we apply the proposed 2D tori algorithm to power-of-two square tori, the startup time and message transmission time are equivalent to those in [19] (see Table 2). But, the proposed algorithm is advantageous in terms of data rearrangement time and message propagation time. In the proposed 2D tori algorithm, data rearrangement is required between phases

TABLE 1
Performance Summary of the Proposed Algorithms

Network	Startup Cost	Message Transmission Cost	Propagation Cost	Data Rearrangement Cost	Barrier Cost
$R \times C$ 2D torus	$\left(\frac{C}{2}+2\right)t_s$	$\frac{RC}{4}(C+4)mt_c$	$2(C-1)t_l$	$3(RC)m\rho$	$\left(\frac{C}{2}+1\right)t_b$
$N_1 \times \ldots \times N_n$ nD torus	$n\left(\frac{N_1}{4}+1\right)t_s$	$\frac{n}{8}(N_1+4)(N_1N_2\ldots N_n)mt_c$	$n(N_1-1)t_l$	$(n+1)(N_1N_2\ldots N_n)m\rho$	$\left(\frac{nN_1}{4}+n-1\right)t_b$
$R \times C$ 2D mesh	$(C)t_s$	$\left(\frac{RC^2}{2}\right)mt_c$	$\{(C-2)^2+2\}t_l$	$2(RC)m\rho$	$(C-1)t_b$
$N_1 \times \ldots \times N_n$ nD mesh	$\left(\frac{n}{2}N_1\right)t_s$	$\frac{n}{4}(N_1^2N_2\ldots N_n)mt_c$	$\frac{n}{2}\{(N_1-2)^2+2\}t_l$	$n(N_1N_2\ldots N_n)m\rho$	$\left(\frac{n}{2}N_1-1\right)t_b$

TABLE 2
Comparison of Completion Time for a $2^d \times 2^d$ Torus or Mesh

Topology	Algorithm	Startup Cost	Message Transmission Cost	Propagation Cost	Rearrangement Cost	Barrier Cost
2^dx2^d Tori	[19]	$(2^{d-1}+2)t_s$	$(2^{3d-2}+2^{2d})mt_c$	$\frac{1}{3}(2^{2d-1}+10)t_l$	$(2^{d-1}+1)2^{2d}m\rho$	$(2^{d-1}+1)t_b$
	[13]	$(3d-3)t_s$	$\{9\cdot 2^{3d-4}+(d^2-5d+3)2^{2d-1}\}m\cdot t_c$	$(13\cdot 2^{d-2}-3d-3)t_l$	$\{9\cdot 2^{3d-4}+(d^2-5d+3)2^{2d-1}\}m\rho$	$(3d-4)t_b$
	Proposed	$(2^{d-1}+2)t_s$	$(2^{3d-2}+2^{2d})mt_c$	$(2^{d+1}-2)t_l$	$(3)2^{2d}m\rho$	$(2^{d-1}+1)t_b$
2^dx2^d Mesh	[16]	$(2^d)t_s$	$(2^{3d-1})mt_c$	$\frac{1}{3}(2^{2d}+2)t_l$	$(2^d-1)2^{2d}m\rho$	$(2^d-1)t_b$
	[13]	$(3d-2)t_s$	$\{7\cdot 2^{3d-3}+(d^2-3d-1)2^{2d-1}\}m\cdot t_c$	$(9\cdot 2^{d-1}-3d-4)t_l$	$\{7\cdot 2^{3d-3}+(d^2-3d-1)2^{2d-1}\}m\rho$	$(3d-3)t_b$
	Proposed	$(2^d)t_s$	$(2^{3d-1})mt_c$	$(2^{2d}-2^{d+2}+6)t_l$	$(2)2^{2d}m\rho$	$(2^d-1)t_b$

to prepare for the next phase. In a $2^d \times 2^d$ torus, there are four phases in the proposed algorithm, thus only three rearrangement steps are required, regardless of the network size. However, in the algorithm in [19], data rearrangement is required between steps rather than phases (in our physical model of data array: if noncontiguous blocks are transmitted, the blocks should be rearranged or copied). Since the algorithm in [19] requires $2^{d-1}+1$ data rearrangement steps, the time complexity due to data rearrangement is $O(2^{3d})$, while that of the proposed algorithm is $O(2^{2d})$. With respect to the total propagation time, the proposed algorithm requires four hops (in Phases 1 and 2), two hops (in Phase 3), and one hop (in Phase 4) per step, regardless of the network size. Thus, this algorithm which exhibits a time complexity of $O(2^d)$ compares favorably to the algorithm in [19], which exhibits a time complexity of $O(2^{2d})$ due to propagation time. Thus, the proposed algorithm exhibits better performance than existing algorithms in power-of-two square tori, even though the proposed algorithm is targeted for the networks whose size of each dimension need not be power-of-two and square. If we compare the performance of 2D tori algorithm with that of the algorithm in [13] for a $2^d \times 2^d$ torus, message startup cost is $O(d)$ for the algorithm [13], while it is $O(2^d)$ for the proposed algorithm. The message-transmission cost of the proposed algorithm is $O(2^{3d})$ as the algorithm [13], but lower than that of the algorithm [13]. The time complexity due to data rearrangement for the algorithm [13] is $O(2^{3d})$, while that of the proposed algorithm is $O(2^{2d})$. With respect to the total propagation time, the proposed algorithm has time complexity of $O(2^d)$ as the algorithm [13], but a little lower than that of the algorithm [13]. Thus, the proposed algorithm is advantageous over the algorithm [13] in all parameters except the startup cost. Compared with existing algorithms for meshes [13], [16], the proposed mesh algorithm shows very similar characteristics.

Ideally, we would base our performance evaluation on the implementation of commercial parallel supercomputers. However, analysis of the scalability of these algorithms across a range of systems sizes is hampered by the lack of availability of a range of large system sizes. Moreover, the systems we could access did not permit control of the shape of allocated subpartition, i.e., we could not guarantee that they would be square. What we desired was a more flexible methodology that would yield reliable estimates of execution time across a broader range of system sizes. Therefore, our studies are based on the analytic models of execution time, but using values of parameters measured on one of commercial machines, Paragon, i.e., $t_s = 75\mu s$, $t_c = 0.011\mu s$, $\rho = 0.014\mu s$, $t_b = (126d - 113)\mu s$ [2], and $t_l = 0.02\mu s$ (measured by us).

Fig. 13 shows the performance of the proposed 2D tori algorithm and existing algorithms [19] and [13] in 16×16, 32×32, 64×64, and 128×128 tori, where Figs. 13b and 13d

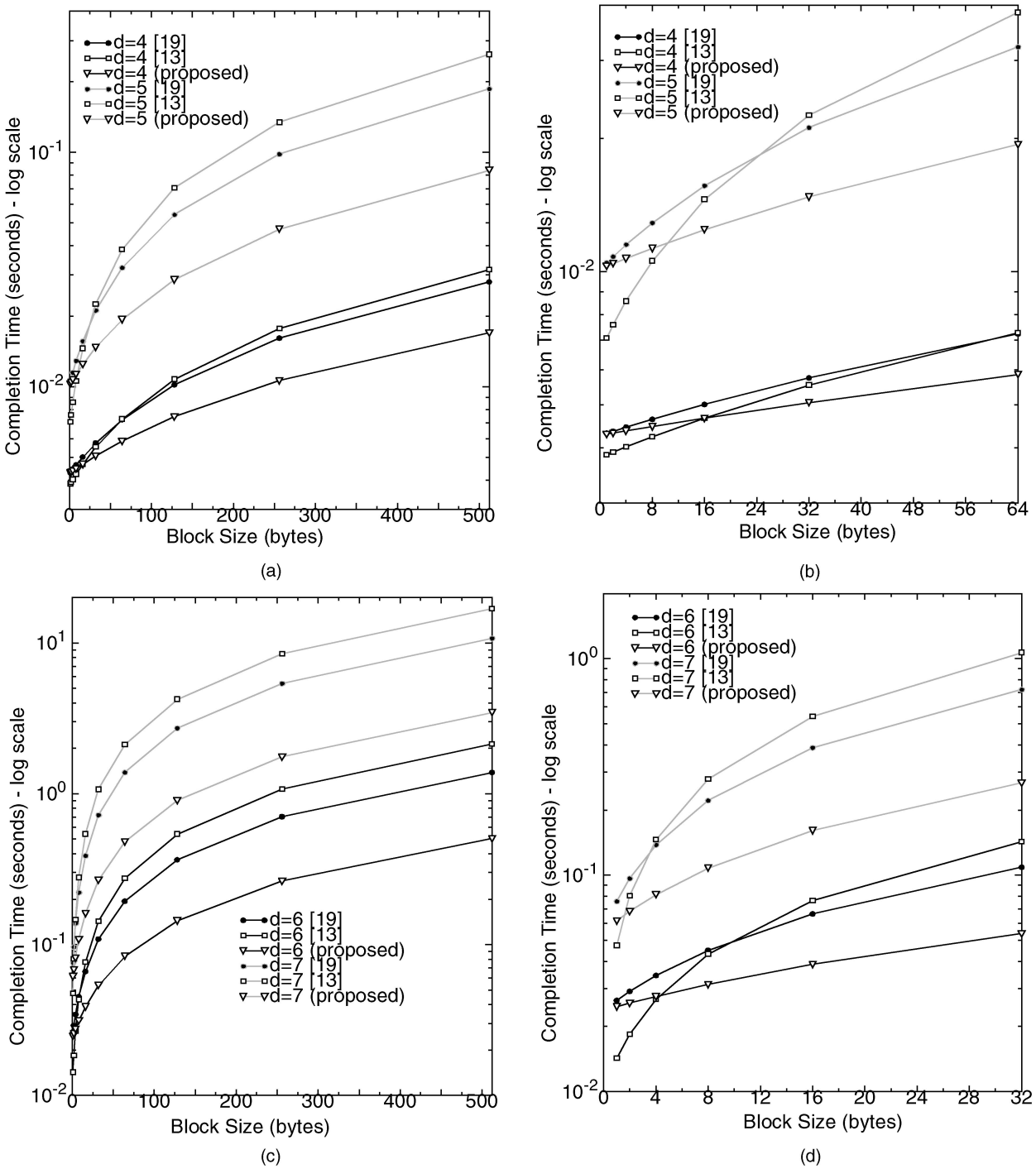

Fig. 13. Estimated performance of algorithms in 16×16, 32×32, 64×64, and 128×128 tori.

illustrate performance of the algorithms when block sizes are small. As shown in the figures, algorithm [13] outperforms the other algorithms for small block sizes, while the proposed algorithm is superior for larger block sizes. As network size increases, the cross-over points appear in small block sizes (see Figs. 13b and 13d). These indicate that startup cost is the most dominant factor when block size is small, while message transmission cost becomes an important factor as network size and/or block size increase, since message transmission cost increases significantly with $O(2^{3d}m)$ while start-up cost increases with $O(2^d)$ (in [19] and the proposed algorithm) or $O(d)$ (in [13]). Interestingly, the proposed algorithm shows much better performance than the algorithm [19], even though both of them exhibit the same startup cost and message transmission cost. This is because the proposed algorithm shows lower data rearrangement and propagation costs, data rearrangement cost becomes more and more important as block size increases, and the time complexity due to data rearrangement for the proposed algorithm is $O(2^{2d})$, while that of the algorithms [19] and [13] is $O(2^{3d})$. Fig. 14 shows the performance of the proposed algorithm for 2D meshes and the algorithm in [13]

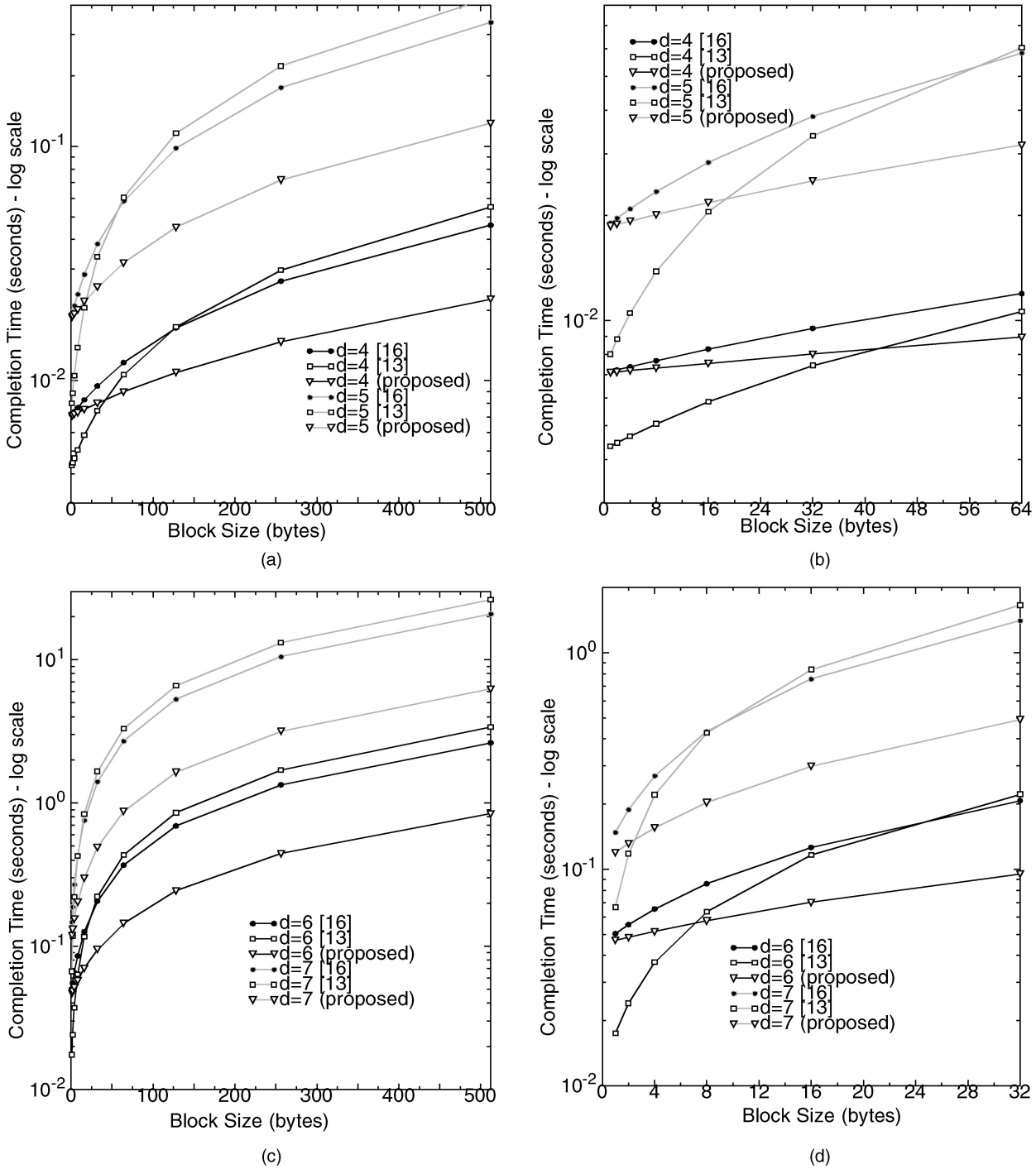

Fig. 14. Estimated performance of algorithms in 16×16, 32×32, 64×64, and 128×128 meshes.

and [16] for all-to-all personalized communication in 16×16, 32×32, 64×64, and 128×128 meshes, using the same parameter values, and we obtained similar results.

Now, to examine the performance of the proposed algorithm and the low startup cost algorithm [13] when the startup cost is large, we consider performance of the proposed and existing algorithms when the ratio of two dominant parameters, start-up cost and message transmission cost (i.e., t_s/t_c) is high. For this study, we used the parameter values listed above, except for start-up cost (t_s). Since the ratio t_s/t_c in the above is about 7,000, we examine a large (20,000) value of t_s/t_c. Fig. 15 illustrates the performance of algorithms in 16×16 and 64×64 tori. The performance characteristics are very similar to those seen in Fig. 13, but the cross-over points appear in large block sizes when t_s/t_c becomes larger. For meshes, we used the same parameter values that are used in tori. Fig. 16 compares the performance of the proposed mesh algorithm and the algorithms [16] and [13] for all-to-all personalized communication in 16×16 and 64×64 meshes, and we obtain similar results.

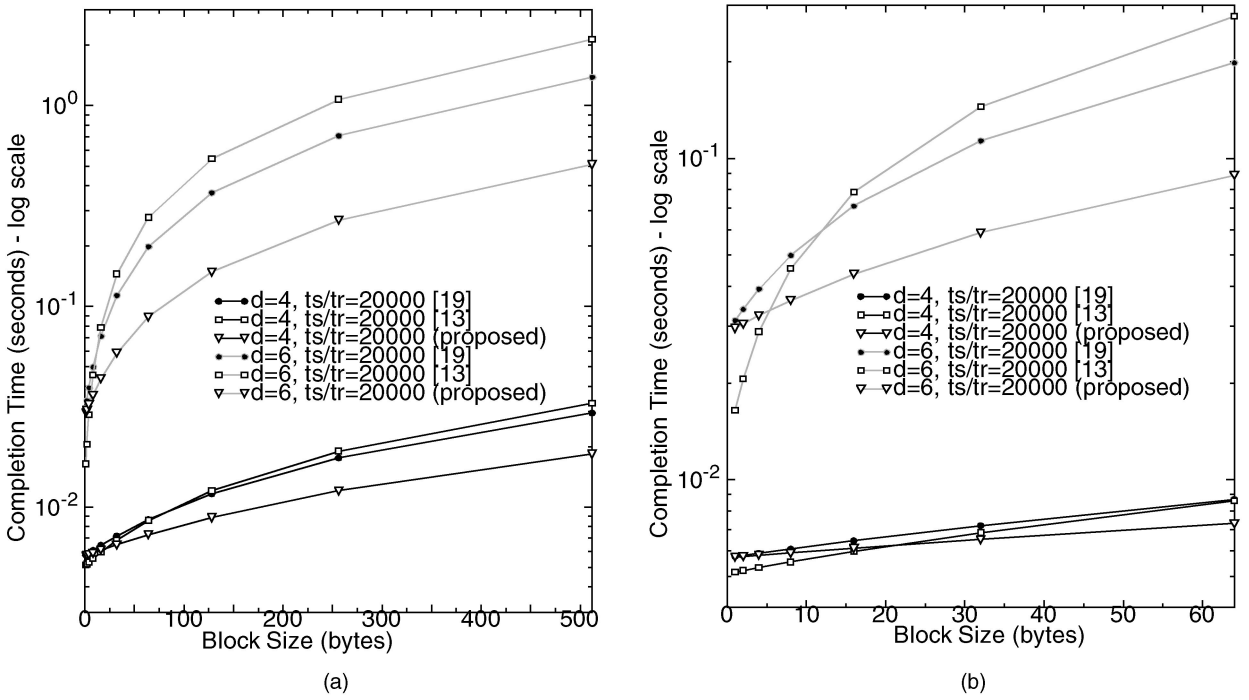

Fig. 15. Estimated performance of algorithms in 16×16 and 64×64 tori when $t_s/t_r = 20,000$.

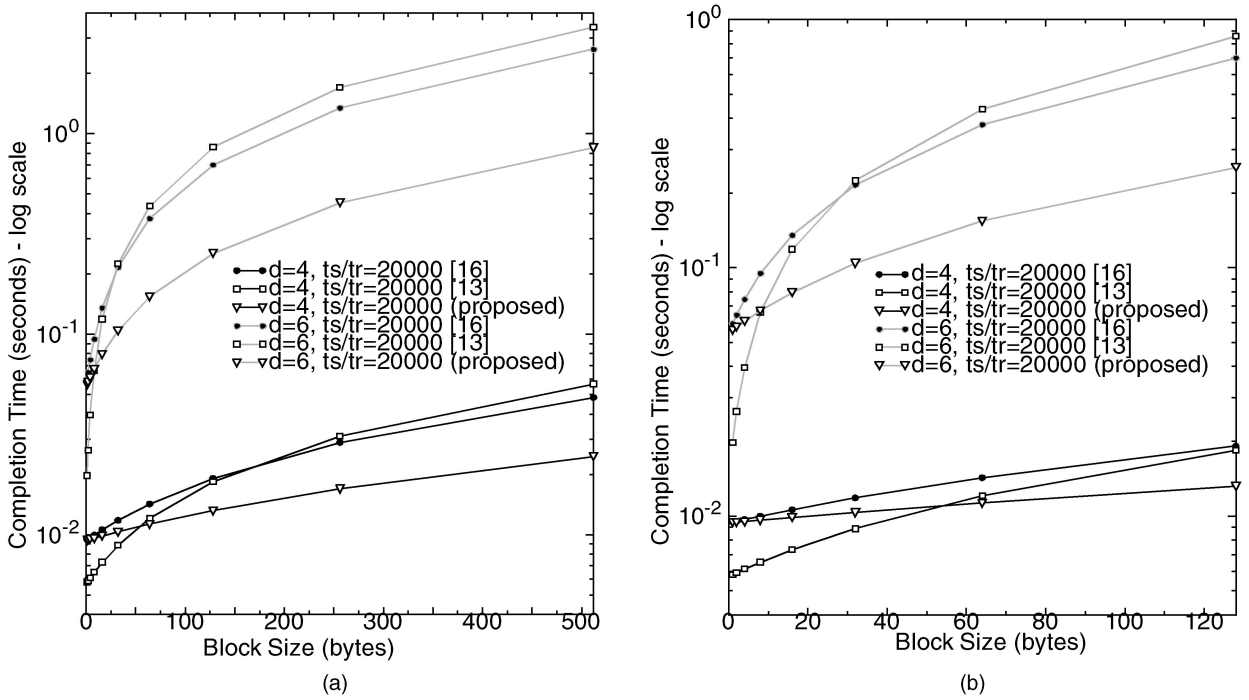

Fig. 16. Estimated performance of algorithms in 16×16 and 64×64 meshes when $t_s/t_r = 20,000$.

6 Conclusions

We proposed new algorithms for all-to-all personalized communication for multidimensional torus- or mesh-connected networks. Although the algorithms targeted at wormhole-switched networks, they can be efficiently used in virtual cut-through or circuit-switched networks. The proposed algorithms utilize message-combining to reduce the time complexity of message startups. Unlike existing message-combining algorithms, the proposed algorithms accommodate non-power-of-two networks of arbitrary dimensions. In addition, destinations remain fixed over a larger number of steps in the proposed algorithms, thus making them amenable to optimizations. Finally, the data structures used are simple, and hence, make substantial saving of message-rearrangement time.

Although we assumed that the number of nodes in each dimension is a multiple of four (in tori) or even (in meshes), the proposed algorithms can be used in tori or meshes with an arbitrary number of nodes in each dimension. If the

number of nodes in each dimension is not a multiple of four (in tori) or odd (in meshes), the proposed algorithms can be used by adding virtual nodes, then having every node perform communication steps as proposed in this paper.

When applied to power-of-two and square tori or meshes, the proposed algorithms showed better performance than the algorithms [19] and [16], but the algorithm [13] outperformed ours when block sizes are small. However, if the network size is not power-of-two and/or the block size is large, the algorithm [13] may suffer performance degradation. The proposed algorithms perform best when the network is not power-of-two and/or the block size is large.

ACKNOWLEDGMENTS

The work reported in this paper was supported in part by the Office of Naval Research under Grant N00014-99-1-0465 and by the Ministry of Education of Korea through the BK-21 program.

REFERENCES

[1] S.H. Bokhari and H. Berryman, "Complete Exchange on a Circuit Switched Mesh," *Scalable High Performance Computing Conf.*, pp. 300-306, 1992.
[2] S. H. Bokhari, "Multiphase Complete Exchange on Paragon, SP2, and CS-2," *IEEE Parallel & Distributed Technology*, pp. 45-59, Fall 1996.
[3] J. Bruck, C.T. Ho, S. Kipnis, and D. Weathersby, "Efficient Algorithms for All-to-All Communications in Multi-Port Message-Passing Systems," *Symp. Parallel Algorithms and Architectures*, pp. 298-309, 1994
[4] W. J. Dally, "Performance Analysis of k-ary n-cube Interconnection Networks," *IEEE Trans. Computers*, vol. 39, no. 6, pp. 775-785, June 1992.
[5] S. Hinrichs, C. Kosak, and D.R. O'Hallaron, T.M. Sticker, and R. Take, "An Architecture for Optimal All-to-All Personalized Communication," *Symp. Parallel Algorithms and Architectures*, pp. 310-319, 1994.
[6] S.L. Johnsson and C.T. Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes," *IEEE Trans. Computers*, vol. 38, no. 9, pp. 1249-1268, Sept. 1989.
[7] P.K. McKinley, Y.-J. Tsai, and D.F. Robinson, "Collective Communication Trees in Wormhole-Routed Massively Parallel Computers," Technical Report MSU-CPS-95-6, Michigan State Univ., Mar. 1995.
[8] P. K. McKinley, Y.-J. Tsai, D. Robinson, "Collective Communication in Wormhole-Routed Massively Parallel Computers," *Computer*, pp. 39-50, Dec. 1995.
[9] L.M. Ni and P.K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *Computer*, vol. 26, pp. 62-76, Feb. 1993.
[10] D.K. Panda, "Issues in Designing Efficient and Practical Algorithms for Collective Communication on Wormhole-Routed Systems," *Proc. ICPP Workshop on Challenges for Parallel Processing*, pp. 8-15, 1995.
[11] D.S. Scott, "Efficient All-to-All Communication Patterns in Hypercube and Mesh Topologies," *Proc. Sixth Conf. Distributed Memory Concurrent Computers*, pp. 398-403, 1991.
[12] Y.J. Suh and S. Yalamanchili, "Algorithms for All-to-All Personalized Exchange in 2D and 3D Tori," *Proc. 10th Int'l Parallel Processing Symp.*, pp. 808-814, Apr. 1996.
[13] Y.J. Suh and S. Yalamanchili, "All-to-All Communication with Minimum Start-Up Costs in 2D/3D Tori and Meshes," *IEEE Trans. Parallel and Distributed Systems*, vol. 9, no. 5, pp. 442-458, May 1998.
[14] Y. J. Suh and K.G. Shin, "Efficient All-to-All Personalized Exchange in Multidimensional Torus Networks," *Proc. 27th Int'l Conf. Parallel Processing*, Aug. 1998.
[15] Y.J. Suh and S. Yalamanchili, "Configurable Algorithms for Complete Exchange in 2D Meshes," *IEEE Trans. Parallel and Distributed Systems*, vol. 11, no. 4, pp. 337-356, Apr. 2000.
[16] N.S. Sundar, D.N. Jayasimha, D.K. Panda, and P. Sadayappan, "Complete Exchange in 2D Meshes," *Scalable High Performance Computing Conf.*, pp. 406-413, 1994.
[17] R. Thakur and A. Choudhary, "All-to-All Communication on Meshes with Wormhole Routing," *Proc. Eighth Int'l Parallel Processing Symp.*, pp. 561-565, Apr. 1994.
[18] Y.-C. Tseng and 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.
[19] Y.-C. Tseng, S. Gupta, and D. Panda, "An Efficient Scheme for Complete Exchange in 2D Tori," *Proc. Int'l Parallel Processing Symp.*, pp. 532-536, 1995.
[20] Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard,"Technical Report CS-93-214, Univ. of Tenn., Apr. 1994.
[21] Cray T3D, "System Architecture Overview," 1994.

Young-Joo Suh received the BS and MS degrees in electronics engineering from Hanyang University, Seoul, Korea, in 1985 and 1987, respectively, and the PhD degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, Georgia, in 1996. From 1988 to 1990, Dr. Suh was a research engineer at the Central Research Center of LG Electronics Inc., Seoul, Korea. From 1990 to 1993, he was an assistant professor in the Department of Computer Science and Engineering at the Chung-Cheong College, Korea. After receiving the PhD degree, he worked as a postdoctoral researcher in the Computer Systems Research Laboratory in the School of Electrical and Computer Engineering at the Georgia Institute of Technology from 1996 to 1997. From 1997 to 1998, he was a research fellow of the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science at the University of Michigan. He is currently an assistant professor in the Department of Computer Science and Engineering at the Pohang University of Science and Technology (POSTECH), Pohang, Korea.

Dr. Suh's current research interests include collective communication, interconnection networks, networking protocols, and mobile computing. Dr. Suh is a member of the IEEE, the IEEE Computer Society, and the IEEE Communications Society.

Kang G. Shin received the BS degree in electronics engineering from Seoul National University, Seoul, Korea in 1970, and both the MS and PhD degrees in electrical engineering from Cornell University, Ithaca, New York in 1976 and 1978, respectively. From 1978 to 1982, he was on the faculty of Rensselaer Polytechnic Institute, Troy, New York. He has held visiting positions at the U.S. Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, The Computer Science Division within the Department of Electrical Engineering and Computer Science at UC Berkeley, the International Computer Science Institute, Berkeley, California, the IBM T.J. Watson Research Center, and Software Engineering Institute at Carnegie Mellon University. He also chaired the Computer Science and Engineering Division, EECS Department, at the University of Michigan for three years beginning January 1991.

Dr. Shin is the professor and founding director of the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, Michigan. He is an IEEE fellow and member of the Korean Academy of Engineering. He was the general chair of the 2000 IEEE Real-Time Technlogy and Applications Symposium and has authored or coauthored more than 600 technical papers and book chapters in the areas of distributed real-time computing and control, computer networking, fault-tolerant computing, and intelligent manufacturing. His current research focuses on Quality of Service (QoS) sensitive computing and networking with emphases on timeliness and dependability. He has also been applying the basic research results to telecommunication and multimedia systems, embedded systems, and manufacturing applications.