Toward Optimal Complete Exchange on Wormhole-Routed Tori

Yu-Chee Tseng, *Member*, *IEEE Computer Society*,
Sze-Yao Ni, *Student Member*, *IEEE Computer Society*, and
Jang-Ping Sheu, *Senior Member*, *IEEE*

Abstract—In this paper, we propose new routing schemes to perform *all-to-all personalized communication* (or known as *complete exchange*) in wormhole-routed, one-port tori. On tori of equal size along each dimension, our algorithms use both asymptotically optimal startup and transmission time. The results are characterized by several interesting features: 1) the use of *gather-scatter* tree to achieve optimality in startup time, 2) enforcement of shortest paths in routing messages to achieve optimality in transmission time, 3) application of *network-partitioning* techniques to reduce the constant associated with the transmission time, and 4) the dimension-by-dimension and gather-scatter-tree approach to make possible applying the results to nonsquare, any-size tori. In the literature, some algorithms are optimal in only one of startup and transmission costs, while some, although asymptotically optimal in both costs, will incur much larger constants associated with the costs. Numerical analysis and experiment both show that significant improvement can be obtained by our scheme on total communication latency over existing results.

Index Terms—All-to-all personalized communication, broadcast, complete exchange, gossiping, multicomputer network, torus, wormhole routing.

1 Introduction

Advances in technology have made possible multicomputers of large scale. In a multicomputer network, fast and efficient interprocessor communication is crucial to unleashing the aggregated computing power. The most basic communication pattern is *one-to-one* (*unicast*). Recent research has put much attention on the *collective communication*, which incurs denser and heavier traffic on the network. Examples include *one-to-all* (*broadcast*), *one-to-many* (*multicast*), and *all-to-all* communications and a large amount of work can be found in [2], [4], [5], [7], [11], [15], [16], [17], [18], [20], [22], [30]. Messages to be sent can be further classified as *nonpersonalized* (wherein all receivers will receive a *same* message from a same source) and *personalized* (wherein each receiver will receive a *different* message from a same source). Some of these communication patterns have also been implemented in PVM [8] and MPI [19] as communication libraries.

In this paper, we study the *all-to-all personalized* communication, or known as *complete exchange* or *gossiping*, wherein each node needs to send a distinct message to each of the rest of the nodes. This represents the densest communication pattern among what is identified above. Applications of complete exchange include matrix algorithms, fast Fourier transformation (FFT), graph algorithms, and data redistribution in HPF [13]. It can also be used to evaluate the quality of an interconnection network. Previous work for complete exchange can be found in [3], [9], [23], [27], [28], [29] for meshes and [6], [10], [25], [26], [31], [32] for tori.

Here, the torus network is considered, which architecture has been adopted by commercial machines such as Cray T3D/T3E. The switching model under consideration is *wormhole routing*, which has been widely used in existing machines such as Caltech MOSAIC, Cray T3D/T3E, IBM SP2, Intel Touchstone Delta, Intel Paragon, MIT J-machine, and nCUBE3.

Works related to the problem considered in this paper include [1], [6], [10], [14], [25], [26], [31], [32]. The results in [1], [6], [14] are based on a torus/mesh using packet switching (or store-and-forward). Such schemes are inappropriate for wormhole-routed networks as the distance-insensitive property is hardly exploited. Communication in a wormhole-routed network typically incurs two kinds of costs: *startup time* and *transmission time*.[1]

Both schemes in [10], [31] use the optimal transmission time to achieve complete exchange in a torus. However, the startup cost is pretty high—$O(n^3)$ in a 2D $n \times n$ torus and $O(n^4)$ in a 3D $n \times n \times n$ torus. To relieve this problem, reference [32] proposes a diagonal-propagation scheme which uses asymptotically optimal transmission time, but incurs a much lower $O(n)$ startup time (for both 2D and 3D tori). This startup time is still relatively higher than the theoretical lower bound of $O(\lg n)$. The first scheme that is known to use both asymptotically optimal startup time and transmission time is proposed in [25], [26]. However, the constant associated with the transmission time is relatively high and the effect of this is significant as the amount of

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

Manuscript received 14 July 1998.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 107147.

1. In a wormhole-routed network, the communication latency to deliver a worm of m bytes is typically modeled as $t_s + mt_x$ [21]. The former cost is termed as *startup* time and the latter, the *transmission* time.

data sent in complete exchange is fairly large (see the comparison in Section 6).

We comment that the complete-exchange algorithms developed for meshes [3], [9], [23], [27], [28], [29] may be directly applied on tori. However, such algorithms may fail in using the additional bandwidth provided by tori (a torus has twice the bisection bandwidth that of a mesh of the same size) and, thus, are inherently slower than good torus algorithms, as has been observed by [32].

In this paper, we also present a complete exchange scheme which uses asymptotically optimal startup and transmission time. For a brief overview, refer to Table 3 and Table 4. Our 2D and 3D schemes both incur transmission time of $\frac{65}{48}$ times the lower bound, as opposed to that of $\frac{9}{2}$ and 10 times, respectively, the lower bound in [25], [26]. According to our numerical evaluation, significant gain can be achieved by our schemes (refer to Fig. 9 and Fig. 11 for a quick overview).

In addition to performance gain, our schemes also possess some features which are worth of pointing out. First, inspired by [25], we also use a "gather-then-scatter" (or called bottom-up in [25]) technique to achieve asymptotically optimal startup time. Second, we try to send messages along shortest paths as much as possible. This turns out to be important to achieve optimality in transmission time. On the contrary, references [25], [26], [31] use nonminimal paths to deliver messages. Third, inspired by [34], [35], we adopt the *network-partitioning* technique to divide a torus into multiple logical subtori. This turns out to be helpful for our schemes to fully utilize the communication bandwidth and to conform to the *one-port* model, wherein a node can only send, and simultaneously receive, one worm at a time. Last, we take a dimension-by-dimension and gather-scatter-tree approach, which makes easy extending our schemes to any-dimensional, nonsquare, non-power-of-2 tori (which seems to be difficult, if not impossible, for the approaches adopted by [25], [26], [32]).

The rest of this paper is organized as follows. As a basic construct, Section 2 develops a complete exchange scheme on a 1D ring. Based on this construct, we present our complete exchange schemes for 2D and 3D tori in Section 3 and Section 4, respectively. The extensions to nonsquare, non-power-of-2 tori are discussed in Section 5. Some numerical analysis and evaluation are shown in Section 6 to demonstrate the strength of our result. In Section 7, issues of synchronization in our schemes are discussed. Conclusions are drawn in Section 8.

2 Basic Construct: Complete Exchange on a Ring

In this section, we consider the complete exchange problem on a ring of length $n = 2^d$. Nodes on the ring are denoted as v_i, $i = 0..(n-1)$. Between v_i and $v_{(i+1) \bmod n}$, there is a positive link from v_i to $v_{(i+1) \bmod n}$ and a negative link along the reverse direction. The *positive distance from v_i to v_j*, denoted as $dist^+(v_i, v_j)$, equals $(j-i) \bmod n$ and the *negative distance from v_i to v_j*, denoted as $dist^-(v_i, v_j)$, is $n - dist^+(v_i, v_j)$. Below, we omit saying "mod" whenever the context is clear. On the ring, a transmission from v_i to v_j along the positive direction will be denoted as $v_i \xrightarrow{+} v_j$, while that along the negative direction will be denoted as $v_i \xrightarrow{-} v_j$.

In the problem of complete exchange, each node v_s has a *message block* (or simply *block*) denoted as b_s^t, which is aimed at node v_t. We use $b_s^{i \triangleright j}$ to denote the set of blocks $\{b_s^i, b_s^{i+1}, \ldots, b_s^j\}$ and $b_s^{i \triangleleft j}$ the set of blocks $\{b_s^i, b_s^{i-1}, \ldots, b_s^j\}$. Symbols $\triangleright$ and $\triangleleft$ are used in the incremental and decremental senses, respectively. Likewise, we define $b_{i \triangleright j}^d = \{b_i^d, b_{i+1}^d, \ldots, b_j^d\}$ and $b_{i \triangleleft j}^d = \{b_i^d, b_{i-1}^d, \ldots, b_j^d\}$.

2.1 The Gather-Scatter Tree

Our scheme consists of a sequence of *gathering* phases followed by a sequence of *scattering* phases. In the beginning, all nodes will join the communication. After each gathering phase, the blocks are concentrated into a smaller number of nodes. On the contrary, blocks are distributed to more nodes after each scattering phase. At the end, it is guaranteed that every block arrives at its destination. The communication patterns of these phases are defined as follows:

Definition 1. *Given any l, $0 \le l \le d-2$, define the communication phases GP_l^+ and SP_l^+ as follows:*

$$GP_l^+ = SP_l^+ = \{v_i \xrightarrow{+} v_{i+2^l} \mid i \bmod 2^l = 0\}.$$

In the definition, GP stands for "gathering phase," SP for "scattering phase," and $+$ for "positive" direction. Note that although GP_l^+ and SP_l^+ have the same communication pattern, as yet to be shown, different blocks are delivered in them.

The concept of the so-called *gather-scatter* tree is best described by putting together a sequence of positive phases,

$$GP_0^+ \to GP_1^+ \to \ldots \to GP_{d-2}^+ \to SP_{d-2}^+ \\ \to SP_{d-3}^+ \to \ldots \to SP_0^+.$$

An example is shown in Fig. 1 with $d = 4$. The gathering phases are time-spread vertically from the bottom, while the scattering phases are time-spread similarly from the top. We will call such a tree the *positive gather-scatter tree* (though, precisely speaking, it is a graph).

The height of the tree is $d-1$. The tree is very helpful in determining how to route a block from one node to another, by taking some gathering phases followed by some scattering phases. For instance, three routes exist from v_2 to v_4: 1) $v_2 \xrightarrow{+} v_4$ in GP_1^+, 2) $v_2 \xrightarrow{+} v_4$ in SP_1^+, and 3) $v_2 \xrightarrow{+} v_3$ in GP_0^+ followed by $v_3 \xrightarrow{+} v_4$ in SP_0^+.

Definition 2. *For each integer l, $0 \le l \le d-2$, define $V_l = \{v_i \mid i \bmod 2^l = 0\}$. For each integer l, $1 \le l \le d-1$, define $\hat{V}_l = V_{l-1} - V_l$ except that $\hat{V}_{d-1} = V_{d-2}$. For all values of l unspecified, $V_l = \emptyset$ and $\hat{V}_l = \emptyset$.*

Intuitively, if v_i belongs to V_l, it will join the communication in GP_l^+. However, v_i will not join the next gathering phase GP_{l+1}^+ if $v_i \in \hat{V}_{l+1}$. For example, v_2 belongs to V_1, implying that v_2 will communicate in GP_1^+, but not in GP_2^+ because $v_2 \in \hat{V}_2$. On the contrary, v_4 is in V_1 and V_2, so

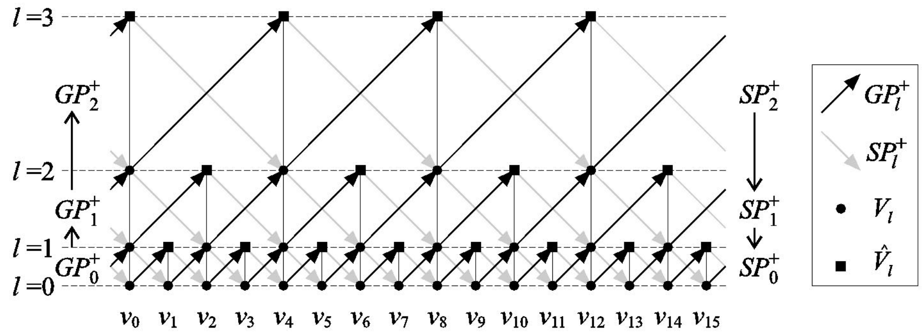


Fig. 1. The positive gather-scatter tree on a ring of length $n = 16$.

it will communicate in both GP_1^+ and GP_2^+. Similar phenomena hold true for scattering phases.

The gather-scatter tree can also be used in determining the set of nodes from/to which a node can gather/scatter blocks. Specifically, if $v_i \in V_l \cup \hat{V}_l$, the set of nodes from which v_i can collect blocks in the gathering phases is ("C" means "coverage")

$$GC_l^+(v_i) = \{v_i, v_{i-1}, \ldots, v_{i-(2^l-1)}\}.$$

Similarly, the set of nodes to which v_i can forward blocks in scattering phases is

$$SC_l^+(v_i) = \{v_i, v_{i+1}, \ldots, v_{i+2^l-1}\}.$$

This leads to the following lemma.

Lemma 1. *For any v_s and v_t such that $dist^+(v_s, v_t) \leq \frac{n}{2}$, there exists a path leading from v_s to v_t on the positive gather-scatter tree.*

Proof. This can be validated since there always exists a v_m between v_s and v_t such that $v_m \in V_h \cup \hat{V}_h$ satisfying $v_s \in GC_h^+(v_m)$ and $v_t \in SC_h^+(v_m)$ for some $h, 0 \leq h \leq (d-1)$.□

Definition 3. *Given any l, $0 \leq l \leq d-2$, define the communication phases GP_l^- and SP_l^- as follows:*

$$GP_l^- = SP_l^- = \{v_i \xrightarrow{-} v_{i-2^l} \mid i \bmod 2^l = 0\}.$$

Definition 3 is simply rewritten from Definition 1 by using links in the negative direction. It is easy to generalize to the concept of the *negative gather-scatter tree* (by reversing the directions of all transmissions in Fig. 1) and further prove a reachability property similar to Lemma 1. It should be understood that the extension to the negative tree is straightforward.

In Sections 2.2 and 2.3, we will develop a complete exchange scheme using the positive and negative gather-scatter trees. For each v_i, it will deliver $n/2$ blocks $b_i^{i+1 \triangleright i+2^{d-1}}$ on the positive tree, and $n/2 - 1$ blocks $b_i^{i-1 \triangleleft i-(2^{d-1}-1)}$ on the negative tree. However, as the negative tree is symmetric to the positive one, we will concentrate our discussion on the positive tree.

2.2 The Path Selection Strategy for Blocks

As shown earlier, there may exist multiple paths between a pair of source and destination nodes on the positive gather-scatter tree. How to choose from these paths to reduce the communication latency is a difficult problem. Mainly, we need a good heuristics to balance the communication load (number of transmitted blocks) on each link in a phase.

The following observation is used as a guideline in designing our scheme:

Observation 1. *For two nodes $v_i \in V_h$ and $v_j \in \hat{V}_h$, the traffic in v_i tends to be busier than that in v_j as v_i needs to join more communication phases than v_j does.*

We next discuss a strategy for routing blocks in the gathering phases. Consider the gathering phase GP_l^+, $0 \leq l \leq (d-2)$. Suppose that, right before GP_l^+, a block b_s^t has arrived at node $v_i \in V_l$. We need to decide, in the communication $v_i \xrightarrow{+} v_{i+2^l}$ in GP_l^+, whether b_s^t should be sent to v_{i+2^l} or not. There are two cases:

Case 1: $v_i \in \hat{V}_{l+1}$. This implies that v_i will be prohibited from communicating in the subsequent gathering phases $GP_{l'}^+$ and scattering phases $SP_{l'}^+$, $l' > l$, which in turn implies that the scattering coverage of v_i is at most as large as $SC_{l+1}^+(v_i)$. Now, consider the location of v_t (see Fig. 2a for an illustration):

1. $dist^+(v_i, v_t) < 2^l$, i.e., $v_t \in SC_l^+(v_i)$. Apparently, b_s^t should not be sent to v_{i+2^l}; otherwise, the block will go too far beyond its destination.
2. $2^l \leq dist^+(v_i, v_t) < 2^{l+1}$. If so, v_t is in both $SC_{l+1}^+(v_i)$ and $SC_{l+1}^+(v_{i+2^l})$. That is, v_t can be reached from both v_i and v_{i+2^l} using later scattering phases. Because $v_i \in \hat{V}_{l+1}$ and $v_{i+2^l} \in V_{l+1}$, according to Observation 1, v_{i+2^l} tends to be busier than v_i and, thus, b_s^t should not be sent in this phase.
3. $dist^+(v_i, v_t) \geq 2^{l+1}$. This implies $v_t \notin SC_{l+1}^+(v_i)$ and, thus, b_s^t must be sent in $v_i \xrightarrow{+} v_{i+2^l}$ so as to reach v_t.

Case 2: $v_i \in V_{l+1}$. This implies that the receiving node $v_{i+2^l} \in \hat{V}_{l+1}$ and will not communicate in the subsequent gathering phases $GP_{l'}^+$ and scattering phases $SP_{l'}^+$, $l' > l$, which in turn implies that the scattering coverage of v_{i+2^l} is at most as large as $SC_{l+1}^+(v_{i+2^l})$. Now, consider the location of v_t (see Fig. 2b for an illustration):

1. $dist^+(v_i, v_t) < 2^l$, i.e., $v_t \in SC_l^+(v_i)$. Apparently, b_s^t should not be sent to v_{i+2^l}.
2. $2^l \leq dist^+(v_i, v_t) < 2^l + 2^{l+1}$, i.e., $v_t \in SC_{l+1}^+(v_{i+2^l})$. However, it is also possible that $v_t \in SC_{l'}^+(v_j)$ such

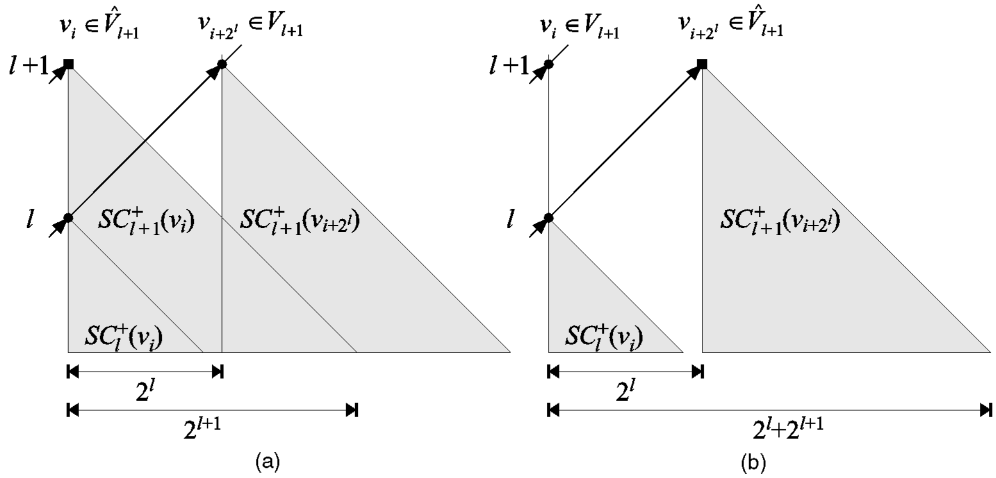


Fig. 2. Node coverage in the communication $v_i \xrightarrow{+} v_{i+2^l}$ of gathering phase GP^+_l: (a) Case 1: $v_i \in \hat{V}_{l+1}$. (b) Case 2: $v_i \in V_{l+1}$.

that $v_j \in V_{l'}$ or $\hat{V}_{l'}$, $l' \geq l+1$. Observation 1 indicates that v_{i+2^l} will be less busy and, thus, b^t_s should be sent in $v_i \xrightarrow{+} v_{i+2^l}$.

3. $dist^+(v_i, v_t) \geq 2^l + 2^{l+1}$. This implies $v_t \notin SC^+_{l+1}(v_{i+2^l})$ and, thus, b^t_s should not be sent in this phase (it will join later phases for wider coverage).

Routing in the scattering phases is simpler. Consider the scattering phase SP^+_l. Suppose that, right before SP^+_l, a block b^t_s has arrived at $v_i \in V_{l+1} \cup \hat{V}_{l+1}$ (i.e., V_l). If $dist^+(v_i, v_t) < 2^l$, i.e., $v_t \in SC^+_l(v_i)$, apparently b^t_s should not be sent to v_{i+2^l} (or it will go too far beyond its destination). Otherwise, $v_t \in SC^+_l(v_{i+2^l})$ and, thus, b^t_s should be sent in $v_i \xrightarrow{+} v_{i+2^l}$.

Example 1. Fig. 3 shows the transmission patterns from the point of view of sources v_0, v_1, v_2, and v_3 to some destinations on the positive gather-scatter tree. For instance, consider the source v_1 in Fig. 3b. In GP^+_0, because $v_1 \in \hat{V}_1$, Case 1 should be applied to $v_1 \xrightarrow{+} v_2$. As destination v_2 satisfies $2^0 \leq dist^+(v_1, v_2) < 2^1$, subcase 2 should be applied and, thus, b^2_1 should remain in v_1. As destination $v_t, t = 3..9$, satisfies $dist^+(v_1, v_t) \geq 2^1$, subcase 3 should be applied and, thus, b^t_1 ($b^{3 \triangleright 9}_1$) should be sent to v_2. In GP^+_1, consider $b^{3 \triangleright 9}_1$ that have been moved to v_2. Because destination $v_2 \in \hat{V}_2$, again Case 1 should be applied to $v_2 \xrightarrow{+} v_4$. As destination v_3 satisfies $dist^+(v_2, v_3) < 2$, subcase 1 should be applied and, thus, b^3_1 should remain in v_2. As destination $v_t, t = 4..5$, satisfies $2^1 \leq dist^+(v_2, v_t) < 2^2$, subcase 2 should be applied and, thus, $b^{4 \triangleright 5}_1$ should remain in v_2. As destination $v_t, t = 6..9$, satisfies $dist^+(v_2, v_t) \geq 2^2$, subcase 3 should be applied and, thus, $b^{6 \triangleright 9}_1$ should be sent to v_4.

Now, consider the source v_3, as shown in Fig. 3d. Similar to the decisions for source v_1, in GP^+_0 $b^{5 \triangleright 11}_3$ should be moved to v_4 (Case 1). In GP^+_1, Case 2 should be applied to $v_4 \xrightarrow{+} v_6$. As v_5 satisfies $dist^+(v_4, v_5) < 2$, subcase 1 should be applied and, thus, b^5_3 should remain in v_4. As $v_t, t = 6..9$, satisfies $2^1 \leq dist^+(v_4, v_t) < 2^1 + 2^2$, subcase 2 should be applied and, thus, $b^{6 \triangleright 9}_3$ should be sent to v_6. As $v_t, t = 10..11$, satisfies $dist^+(v_4, v_t) \geq 2^1 + 2^2$, subcase 3 should be applied and, thus, $b^{10 \triangleright 11}_3$ should remain in v_6.

2.3 The Routing Algorithm

We now reorganize the algorithm in a formal way. Routing on the positive tree consists of $2d-2$ phases:

$$GP^+_0 \rightarrow GP^+_1 \rightarrow \ldots \rightarrow GP^+_{d-2} \rightarrow SP^+_{d-2} \rightarrow SP^+_{d-3} \rightarrow \ldots \rightarrow SP^+_0.$$

Initially, each v_i has a pool of blocks $B_i = b^{i+1 \triangleright i+2^{d-1}}_i$. Note that B_i will change by time. At the end of the algorithm, the B_i in each v_i contains $b^i_{i-2^{d-1} \triangleright i-1}$. Every v_i executes the following phases synchronously.

Phase GP^+_l: $//l = 0, 1, ..., (d-2)$.
  if $v_i \in V_l$ **then**
    if $v_i \in V_{l+1}$ **then** //Case 2.
      $M = \{b^t_s |\ b^t_s \in B_i \text{ and } v_t \in SC^+_{l+1}(v_{i+2^l})\}$.
    else //Case 1.
      $M = \{b^t_s |\ b^t_s \in B_i \text{ and } v_t \notin SC^+_{l+1}(v_i)\}$.
    end if
    $B_i = B_i - M$.
    Send M to v_{i+2^l}.
    Receive blocks from v_{i-2^l} and add these blocks to B_i.
  end if
Phase SP^+_l: $//l = (d-2), (d-3), ..., 0$.
  if $v_i \in V_l$ **then**
    $M = \{b^t_s |\ b^t_s \in B_i \text{ and } v_t \in SC^+_l(v_{i+2^l})\}$.
    $B_i = B_i - M$.
    Send M to v_{i+2^l}.
    Receive blocks from v_{i-2^l} and add these blocks to B_i.
  end if

The correctness of the routing algorithm can be seen as follows: Case 1 of the algorithm GP^+_l describes that a block b^t_s will stay in v_i if $v_t \in SC^+_{l+1}(v_i)$ or be moved to v_{i+2^l} for joining later phases. Case 2 of the algorithm GP^+_l describes that b^t_s will be moved from v_i to v_{i+2^l} if $v_t \in SC^+_{l+1}(v_{i+2^l})$. Otherwise, b^t_s stays in v_i because $v_t \in SC^+_l(v_i)$ (Fig. 2b) or it will join later phases. Therefore, after the gathering phases

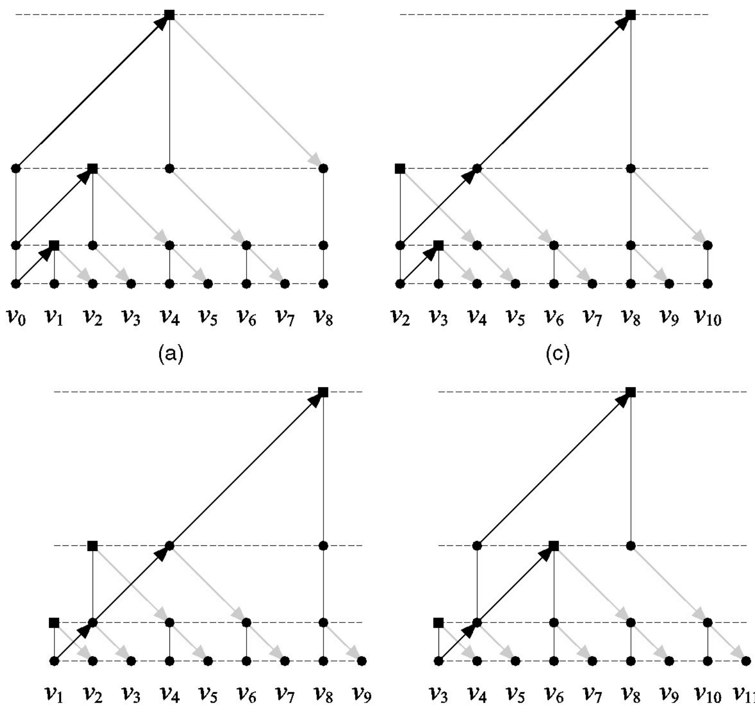


Fig. 3. Path selection from the point of view of sources v_0, v_1, v_2, and v_3 to some destinations.

$GP_0^+ \to GP_1^+ \to \ldots \to GP_{d-2}^+$ are completed, the block b_s^t will be located in some v_m such that $v_t \in SC_h^+(v_m)$ and $v_m \in V_h \cup \hat{V}_h$, for some $h, 0 \le h \le (d-1)$. Then, the scattering phases $SP_{d-2}^+ \to SP_{d-3}^+ \to \ldots \to SP_0^+$ will ensure that b_s^t can reach its destination v_t.

2.4 Overlapping Positive Phases and Negative Phases

We have derived the routing on the positive gather-scatter tree; routing on the negative tree can be similarly obtained. To perform complete exchange, one naive solution is to sequentially perform the positive phases followed by the negative phases. Apparently, this is inefficient as half of the links will be unused in each phase. A better solution is to overlap positive phases and negative phases:

$$\begin{aligned}(GP_0^+ \cup GP_0^-) &\to (GP_1^+ \cup GP_1^-) \to \ldots (GP_{d-2}^+ \cup GP_{d-2}^-)\\ &\to (SP_{d-2}^+ \cup SP_{d-2}^-) \to (SP_{d-3}^+ \cup SP_{d-3}^-)\\ &\to \ldots \to (SP_0^+ \cup SP_0^-).\end{aligned}$$

However, problems may arise because some nodes may need to send/receive more than one message in one phase, thus violating the one-port model. Below we show how to modify our algorithm to solve this problem.

First, we shift the communication patterns in all negative phases, except GP_0^- and SP_0^-, along the positive direction by one position. That is, we redefine the following negative phases:

$$GP_l^- = SP_l^- = \{v_i \xrightarrow{-} v_{i-2^l} \mid (i-1) \bmod 2^l = 0\},\quad 1 \le l \le (d-2).$$

This will relieve the necessity for a node to send/receive more than one message in phase $(GP_l^+ \cup GP_l^-)$ and phase $(SP_l^+ \cup SP_l^-)$, $1 \le l \le (d-2)$. For example, see the second to fifth phases in Fig. 4. Note that V_l and $\hat{V}_l$ for the negative tree need to be adjusted accordingly to adapt to those changes.

However, since all transmissions of GP_0^+, GP_0^-, SP_0^+, and SP_0^- are of distance one, the above shifting technique does not help to satisfy the one-port constraint. Therefore, we redefine GP_0^+, GP_0^-, SP_0^+, and SP_0^- by removing some transmissions from them as follows:

$$\begin{aligned}GP_0^+ &= \{v_i \xrightarrow{+} v_{i+1} \mid i \bmod 2 = 1\},\\ SP_0^+ &= \{v_i \xrightarrow{+} v_{i+1} \mid i \bmod 2 = 0\},\\ GP_0^- &= \{v_i \xrightarrow{-} v_{i-1} \mid i \bmod 2 = 0\},\\ SP_0^- &= \{v_i \xrightarrow{-} v_{i-1} \mid i \bmod 2 = 1\}.\end{aligned}$$

Now, $GP_0^+ \cup GP_0^-$, as well as $SP_0^+ \cup SP_0^-$, will conform to the 1-port constraint, as shown in Fig. 4 (the first and last phases).

With these changes, we need to modify the routing of some b_s^t, $dist(v_s, v_t) \le 2$, too. Taking source v_0 as an example, blocks b_0^1 and b_0^2 that would have been sent in the original GP_0^+ will be left undelivered (since $v_0 \xrightarrow{+} v_1$ is removed) and, thus, kept in v_0. Fortunately, this can be taken care of by delivering b_0^1 in $v_0 \xrightarrow{+} v_1$ of SP_0^+ and delivering b_0^2 in $v_0 \xrightarrow{+} v_2$ of GP_1^+. Similarly, the block b_1^2 of v_1 that would have been sent in the original SP_0^+ will be undeliverable since $v_1 \xrightarrow{+} v_2$ is removed. Still, this can be

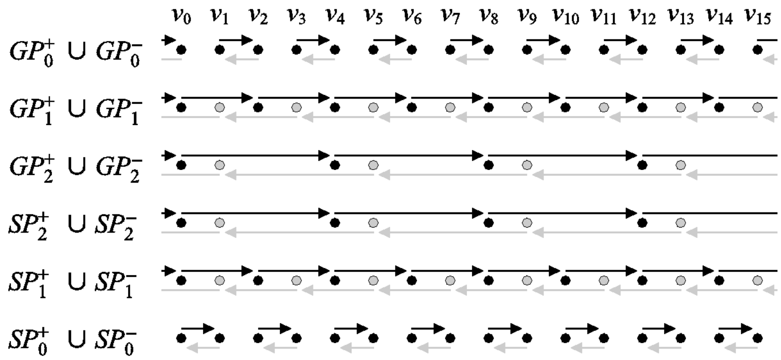


Fig. 4. Adjusting positive and negative phases to conform to the one-port model.

solved by sending b_1^2 in $v_1 \xrightarrow{+} v_2$ of GP_0^+. We can accommodate these changes in the routing rules of GP_l^+, GP_l^-, SP_l^+, and SP_l^-, $l = 0, 1$. All other phases remain the same.

2.5 Performance Analysis

In the following, we analyze the total latency of our complete exchange scheme on a ring of length 2^d with message block size b, startup time t_s, and transmission time t_x. Lemmas 2 to 6 show the communication latency of the *original* positive phases (i.e., *without* considering the modification in Section 2.4). Finally, Theorem 1 gives the total communication latency incurred by the phases *with* the changes in Section 2.4. The proofs of Lemmas 2 to 6 and Theorem 1 can be found in the appendix.

Lemma 2. *The latency of* $GP_l^+, 0 \leq l \leq d-3, d \geq 3$, *is*

$$T_{1D,GP_l^+}(d,b) = \\ t_s + max\{(2^{d+l-1} - 5 \cdot 2^{2l-1} + 3 \cdot 2^{l-1}), (7 \cdot 2^{2l-2})\} \cdot b \cdot t_x. \tag{1}$$

Lemma 3. *The latency of* $GP_{d-2}^+, d \geq 3$ *is*

$$T_{1D,GP_{d-2}^+}(d,b) = t_s + (2^{2d-6} + 3 \cdot 2^{d-3}) \cdot b \cdot t_x.$$

Lemma 4. *The latency of* $SP_{d-2}^+, d \geq 3$, *is*

$$T_{1D,SP_{d-2}^+}(d,b) = t_s + b \cdot t_x.$$

Lemma 5. *The latency of* $SP_l^+, 0 \leq l \leq d-3, d \geq 3$, *is*

$$T_{1D,SP_l^+}(d,b) = \\ t_s + max\{(2^{d+l-1} - 5 \cdot 2^{2l-1} + 3 \cdot 2^{l-1}), (7 \cdot 2^{2l-2})\} \cdot b \cdot t_x. \tag{2}$$

From 2 and 5, we find the interesting coincidence that $T_{1D,GP_l^+}(d,b) = T_{1D,SP_l^+}(d,b)$, $0 \leq l \leq (d-3)$. Also, there are two terms in the max function in (1) or (2). The following corollary resolves the max function when $d \geq 4$.

Corollary 1. *The latency of a positive phase* GP_l^+ *or* SP_l^+, $0 \leq l \leq d-4, d \geq 4$, *is*

$$T_{1D,GP_l^+}(d,b) = T_{1D,SP_l^+}(d,b) \\ = t_s + (2^{d+l-1} - 5 \cdot 2^{2l-1} + 3 \cdot 2^{l-1}) \cdot b \cdot t_x,$$

and, if $l = d-3$, *is*

$$T_{1D,GP_{d-3}^+(d,b)} = T_{1D,SP_{d-3}^+(d,b)} \\ = t_s + \begin{cases} (3 \cdot 2^{2d-7} + 3 \cdot 2^{d-4}) \cdot b \cdot t_x & if\ 3 \leq d \leq 5, \\ (7 \cdot 2^{2d-8}) \cdot b \cdot t_x & if\ d \geq 6. \end{cases}$$

Lemma 6. *The total communication time of all positive phases on a ring of length* 2^d, $d \geq 3$, *is*

$$T_{1D}^+(d,b) = \\ \begin{cases} (2d-2)t_s + (\frac{1}{3} \cdot \frac{31}{32} \cdot 2^{2d-1} + 2^{d-3} - \frac{1}{3}) \cdot b \cdot t_x & if\ 3 \leq d \leq 5, \\ (2d-2)t_s + (\frac{1}{3} \cdot \frac{65}{64} \cdot 2^{2d-1} - 2^{d-2} - \frac{1}{3}) \cdot b \cdot t_x & if\ d \geq 6. \end{cases}$$

Theorem 1. *The total communication time of our complete exchange scheme on a ring of length* 2^d, $d \geq 3$, *is*

$$T_{1D}(d,b) = \begin{cases} T_{1D}^+(d,b) + 3 \cdot b \cdot t_x & if\ d = 3, \\ T_{1D}^+(d,b) + 2 \cdot b \cdot t_x & if\ d \geq 4. \end{cases}$$

3 Complete Exchange on a 2D Torus

In this section, we consider the complete exchange on a 2D $n \times n$ torus, where $n = 2^d$. Nodes in the torus are denoted as $v_{(i,j)}$, $i = 0..(n-1)$ and $j = 0..(n-1)$. Each $v_{(i,j)}$ has a block $b_{(i,j)}^{(x,y)}$ aimed at $v_{(x,y)}$.

In Section 3.1, we first present a naive scheme, based on which we then develop a more efficient one in Section 3.2.

3.1 A Naive Scheme: Algorithm T1

One obvious approach to perform complete exchange on a 2D torus is to regard the torus as the graph product of two rings and directly apply the ring complete exchange in Section 2, first along the x-axis and then along the y-axis. This is summarized below.

X-Stage: For each $j = 0..(n-1)$, regard $v_{(0,j)}, v_{(1,j)}, \ldots, v_{(n-1,j)}$ as a ring and perform the ring complete exchange. In the complete exchange, instead of

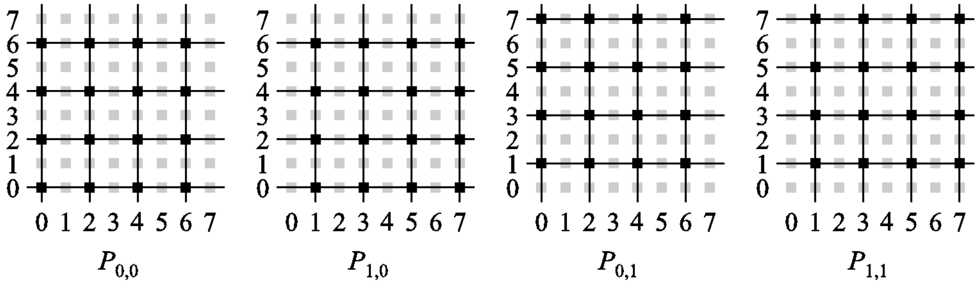


Fig. 5. Four logical tori in an 8×8 torus.

sending one block to each destination, $v_{(i,j)}$ should forward n blocks $b_{(i,j)}^{(x,*)}$ to destination $v_{(x,j)}$, $x = 0..(n-1)$, where $* = 0..(n-1)$.

Y-Stage: For each $i = 0..(n-1)$, regard $v_{(i,0)}, v_{(i,1)}, \ldots, v_{(i,n-1)}$ as a ring and perform the ring complete exchange. In the complete exchange, each $v_{(i,j)}$ forwards n blocks $b_{(*,j)}^{(i,y)}$ to $v_{(i,y)}$, $y = 0..(n-1)$.

Lemma 7. *On a $2^d \times 2^d$ torus, $d \geq 3$, the latency incurred by algorithm T1 is*

$$T_{2D,T1}(d, b) = 2 \cdot T_{1D}(d, 2^d \cdot b).$$

Proof. By Theorem 1, the cost of the X-stage is $T_{1D}(d, 2^d \cdot b)$ as the data to be forwarded from a node to any other node contains 2^d blocks. This is the same for the Y-stage. □

3.2 A Network-Partitioning Approach: Algorithm T4

The obvious deficiency of algorithm T1 is that transmissions always happen along either the X dimension or Y dimension, but not both. The implication here is that at least half of communication bandwidth is waste. To fix this problem, we propose a new scheme called T4, which is so named because *four* copies of T1 will be running simultaneously.

The idea is similar to the *network-partitioning* approach proposed in [34], [35]. We will construct four logical tori, $P_{i,j}$, $0 \leq i, j < 2$, each of size $\frac{n}{2} \times \frac{n}{2}$. The logical torus $P_{i,j}$ consists of nodes

$$\{v_{(x,y)} | \ (x \bmod \sqrt{4}) = i \text{ and } (y \bmod \sqrt{4}) = j\}.$$

In $P_{i,j}$, a node $v_{(x,y)}$ is considered to have a logical link (which is physically dilated by two) to each of nodes $v_{(x\pm2,y)}$ and $v_{(x,y\pm2)}$. For instance, Fig. 5 shows four logical tori in an 8×8 torus. However, communication can be performed in a dilated torus as fast as it can in an ordinary torus due to the distance-insensitive property of wormhole routing. Two important properties offered by such logical partitioning are:

P1. The four logical tori $P_{i,j}$, $0 \leq i, j < 2$, are node-disjoint.

P2. Tori $P_{0,0}$ and $P_{1,1}$ are link-disjoint, and $P_{1,0}$ and $P_{0,1}$ are link-disjoint.

Next, we need to schedule complete exchange on these four logical tori. Property **P1** guarantees that we can freely use these tori without violating the 1-port constraint. **P2** guarantees that we can simultaneously run algorithm T1 on tori $P_{0,0}$ and $P_{1,1}$ without any link contention. We observe that more saving can be obtained by running algorithm T1 on $P_{1,0}$ and $P_{0,1}$ by swapping the execution order to first running Y-stage and then X-stage. The communication directions are summarized in Table 1.

Note that there is no link contention among all these four tori. Also note that, although the above scheduling does utilize all links in every phase, blocks may not reach some of their destinations since the logical tori are node-disjoint. So, some preparation phases shown below are necessary. We schedule every node (say, $v_{(x,y)} \in P_{i,j}$) to forward its blocks aimed at nodes in the other three tori $P_{i+1,j}$, $P_{i,j+1}$, and $P_{i+1,j+1}$ (note that, here, "mod 2" is necessary for subscripts larger than one) before performing the above two stages. This can be done in two phases:

Pre1. Node $v_{(x,y)}$ sends to $v_{(x+1,y)}$ all blocks aimed at $P_{i+1,j}$ and $P_{i+1,j+1}$.

Pre2. Node $v_{(x,y)}$ sends to $v_{(x,y+1)}$ all blocks, together with the blocks received from $v_{(x-1,y)}$ (in **Pre1**), aimed at $P_{i,j+1}$.

The result is that each $v_{(x,y)} \in P_{i,j}$ has collected blocks from $v_{(x-1,y)}$, $v_{(x,y-1)}$, and $v_{(x-1,y-1)}$ aimed at nodes in $P_{i,j}$ and will deliver these blocks in place of these three nodes. In both phases **Pre1** and **Pre2**, $n^2/2$ blocks are sent.

Theorem 2. *On a $2^d \times 2^d$ torus, $d \geq 4$, the communication latency incurred by algorithm T4 is*

$$T_{2D,T4}(d, b) = 2t_s + 2^{2d} \cdot b \cdot t_x + T_{2D,T1}(d - 1, 4b).$$

Proof. The first two terms are incurred by the two preprocessing phases. The last term is by algorithm T1, which is run concurrently on all four logical tori (each of size $2^{d-1} \times 2^{d-1}$). As each node needs to represent three other neighbors, the latency is $T_{2D,T1}(d - 1, 4b)$. □

TABLE 1
The Scheduling of Communication Directions of Algorithm T4

	$P_{0,0}$	$P_{1,1}$	$P_{0,1}$	$P_{1,0}$
Stage 1	X-stage		Y-stage	
Stage 2	Y-stage		X-stage	

TABLE 2
Scheduling of Communication Directions for the Logical Torus Groups in a 3D Torus

	G_0	G_1	G_2	G_3
Stage 1	X-stage	Y-stage	Z-stage	
Stage 2		X-stage	Y-stage	Z-stage
Stage 3	Z-stage		X-stage	Y-stage
Stage 4	Y-stage	Z-stage		X-stage

4 Extension to 3D Tori

In the following, we show how our approaches is extended for a 3D $n \times n \times n$ torus. Nodes in the torus will be denoted as $v_{(x,y,z)}$, $0 \leq x, y, z \leq n-1$.

Similar to Section 3, we also develop a naive 3-stage scheme, called C1, by first performing an X-stage on each ring along the x-axis, then a Y-stage on each ring along the y-axis, and then a Z-stage on each ring along the z-axis. Apparently, each stage will take time $T_{1D}(d, n^2 \cdot b)$, so the total time is $3 \cdot T_{1D}(d, n^2 \cdot b)$.

As before, to better utilize the communication bandwidth, we need to partition the network into a number of smaller 3D logical tori. The notions behind constructing these logical tori are as follows: First, as there are three stages X, Y, and Z (in the naive scheme), nodes along each axis should be divided into at least three logical tori to fully utilize all communication links. Second, these logical tori should be cubic (of equal sizes along all dimensions); therefore, no matter which stage (X, Y, or Z) a torus is scheduled to execute, the communication time is about the same (thus, no torus needs to wait for others to complete).

Based on these observations, one possibility is to partition each axis into four logical rings. Thus, we define 4^3 dilation-4 logical tori $C_{i,j,k}$, $0 \leq i, j, k < 4$, such that $C_{i,j,k}$ consists of nodes

$$\{v_{(x,y,z)} \mid (x \bmod 4) = i, (y \bmod 4) = j, \text{and } (z \bmod 4) = k\}.$$

In $C_{i,j,k}$, node $v_{(x,y,z)}$ is considered to be logically adjacent to six nodes $v_{(x\pm4,y,z)}$, $v_{(x,y\pm4,z)}$, and $v_{(x,y,z\pm4)}$. The logical connection is physically dilated by four links. A property similar to **P1** in Section 3.2 is:

P1′: The 64 logical tori $C_{i,j,k}$, $0 \leq i, j, k < 4$, are mutually node-disjoint.

However, some logical tori do share common links. So, we classify the tori, according to their link sets, into four groups, $s = 0..3$,

$$G_s = \{C_{i,j,k} \mid (i + j + k) \bmod 4 = s\}.$$

Each G_s contains 16 logical tori. A property similar to **P2** in Section 3.2 is:

P2′: All 16 logical tori in each G_s are link-disjoint.

Similar to the development in Section 3.2, we can schedule any communication on these 64 logical tori without violating the 1-port constraint (**P1′**). Simultaneously performing any communication stage (X-, Y-, or Z-stage) is free from contention in all tori of G_s (**P2′**). One possible arrangement is shown in Table 2. Intuitively, the scheduling of each stage is obtained by cyclically shifting that in the previous stage. Every torus group will perform one X-stage, one Y-stage, and one Z-stage in some order. One nice property is that all n^2 axes along each dimension are busy at each stage. The scheme is named C64 because it is featured by having 64 logical tori running C1 simultaneously.

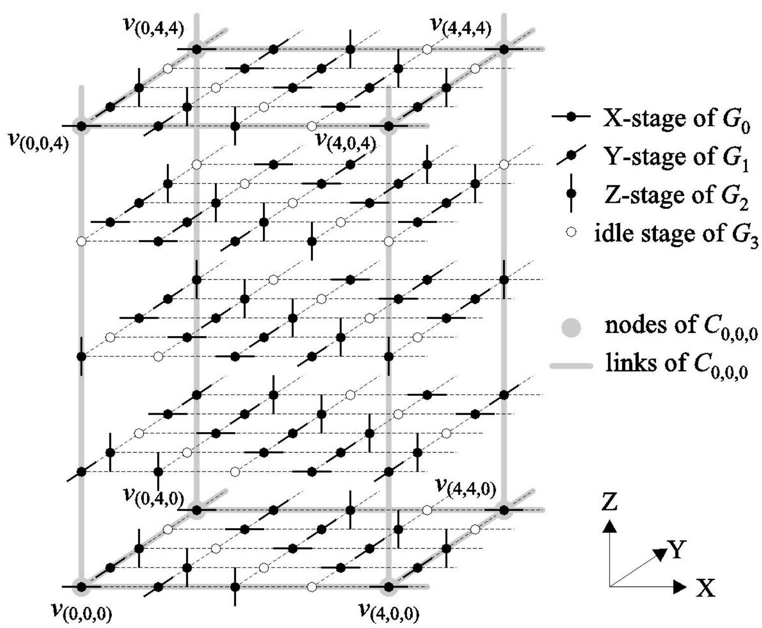


Fig. 6. Illustration of communication pattern in stage 1 in a 3D torus.

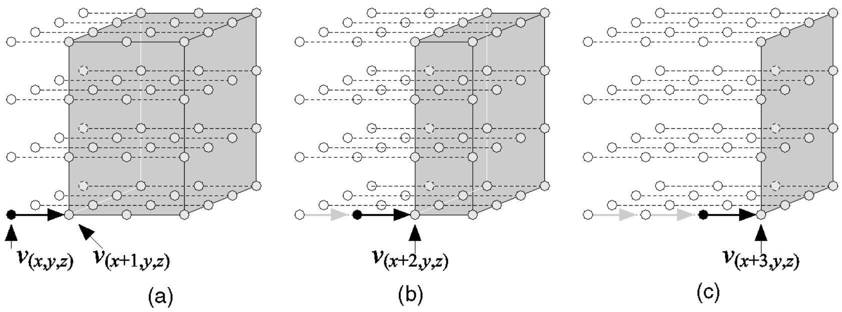


Fig. 7. The first three preparation phases.

Fig. 6 demonstrates the communication in Stage 1 following Table 2 (for clarity, only a portion of the torus is shown). Observe that all physical links are utilized. The similar phenomena will occur in other stages, too.

It remains to describe the preparation phases. We can let each node $v_{(x,y,z)}$ forward $n^3/64$ blocks to each $v_{(x+\delta_x,y+\delta_y,z+\delta_z)}$, $0 \le \delta_x, \delta_y, \delta_z \le 3$. The latter nodes will be responsible for delivering the received blocks to other nodes in their logical tori. It suffices to use nine phases as follows:

1. three phases each of the pattern $v_{(x,y,z)} \xrightarrow{+} v_{(x+1,y,z)}$,
2. three phases each of the pattern $v_{(x,y,z)} \xrightarrow{+} v_{(x,y+1,z)}$, and
3. three phases each of the pattern $v_{(x,y,z)} \xrightarrow{+} v_{(x,y,z+1)}$.

The three phases in 1 are illustrated in Fig. 7, where the nodes in the gray area are the target to which blocks are expected to be delivered. Thus, in total, $48 \cdot \frac{n^3}{64}$, $32 \cdot \frac{n^3}{64}$, and $16 \cdot \frac{n^3}{64}$ blocks will be sent in these three phases, respectively. Note that, as a node needs to forward blocks for other nodes in phases of 2 and 3, the numbers of blocks sent in them will be exactly the same as that in 1.

Theorem 3. *On a $2^d \times 2^d \times 2^d$ torus, $d \ge 5$, the communication latency incurred by our algorithm C64 is*

$$T_{3D}(d,b) = 9t_s + 9 \cdot 2^{3d-1} \cdot b \cdot t_x + 4 \cdot T_{1D}(d-2, 64 \cdot 2^{2(d-2)} \cdot b).$$

Proof. The preprocessing cost is $3 \cdot (3t_s + (48+32+16)\frac{n^3}{64} \cdot b \cdot t_x)$, which gives the first two terms. The last term is from the cost of complete exchange on a ring multiplied by four stages. □

5 Extension to Nonsquare, Non-Power-of-2 Tori

Up to this point, it seems that our approach can only be applied to tori whose side lengths are equal and power-of-2. Recall that, in Section 2, we developed a (*perfect*) gather-scatter tree on a ring of length $n = 2^d$. In fact, if some irregularity is allowed, on a ring of *any* size n, a positive gather-scatter tree can be obtained by slightly modifying Definition 1 as follows:

$$GP_l^+ = SP_l^+ = \{v_i \xrightarrow{+} v_j \mid i \bmod 2^l = 0\},$$
$$\text{where } j = \begin{cases} i + 2^l & \text{if } i + 2^l < n, \\ 0 & \text{otherwise.} \end{cases}$$

In $GP_l^+ = SP_l^+$, whenever the destination node v_{i+2^l} does not exist, we "wraparound" the destination to node v_0.

Let $d = \lceil \lg n \rceil$. The *positive gather-scatter tree* is still defined based on the $2(d-1)$ phases:

$$GP_0^+ \to GP_1^+ \to \ldots \to GP_{d-2}^+ \to SP_{d-2}^+ \to SP_{d-3}^+ \to \ldots \to SP_0^+.$$

For instance, Fig. 8 shows the positive gather-scatter trees on rings of lengths $n = 10$ and $n = 13$.

The gathering coverage and scattering coverage should be changed accordingly, for instance, the gathering coverage $GC_3^+(v_0) = \{v_0, v_9, v_8, \ldots, v_5\}$ in Fig. 8a and the scattering coverage $SC_3^+(v_{12}) = \{v_{12}, v_0, v_1, \ldots, v_3\}$ in Fig. 8b. A reachability property similar to Lemma 1 will still hold true. We conjecture that the communication time of our complete exchange on a ring of length n may be upper-bounded by $T_{1D}(\lceil \lg n \rceil, b)$. However, the problem of deriving the exact formula of the performance of complete exchange on a non-power-of-2 ring is an open problem to us.

With the availability of complete exchange on a ring of any size, the extension to higher-dimensional, non-power-of-2 tori can be obtained following the line of development in earlier sections. It is also straightforward to extend our scheme to a nonsquare torus since we take a dimension-by-dimension approach.

6 Performance Comparison

In this section, we compare our algorithms (T4 and C64) against those by [25](T-2D1, T-2D2, T-3D1, and T-3D2)[2] and [32] (DP-2D and DP-3D). The following lemma will be used for analyzing these schemes.

Lemma 8. *To perform complete exchange on a 2D $n \times n$ torus (resp., 3D $n \times n \times n$ torus), a lower bound on the startup time is $\lg(n^2)t_s$ (resp., $\lg(n^3)t_s$) and a lower bound on the transmission time is $\frac{n^3}{8} b \cdot t_x$ (resp., $\frac{n^4}{8} b \cdot t_x$).*

2. In [26], algorithms T-2D1 and T-3D1 are referred to as T2D and T3D, respectively.

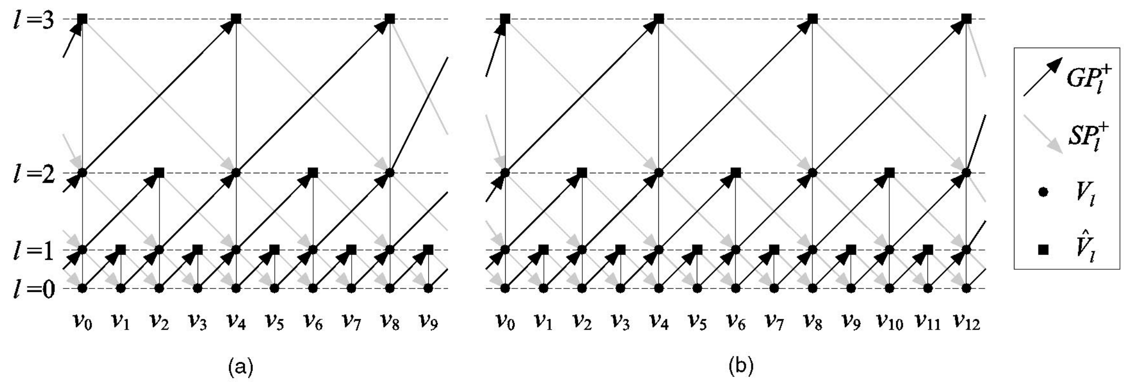


Fig. 8. The positive gather-scatter trees on rings of lengths (a) $n = 10$ and (b) $n = 13$.

Proof. The lower bounds on the startup time are obtained by taking logarithm of the network size. The bounds on transmission time are established in [32]. □

Table 3 shows the startup and transmission costs of our and others' algorithms on a 2D $2^d \times 2^d$ torus. In terms of startup cost, both T-2D1 and our T4 have an order of $O(d)$, while T-2D2 and DP-2D have an exponential order of $O(2^d)$. Our T4 incurs a slightly higher startup time, but, as will be shown later, this will be offset by the transmission time, which is relatively more significant. In terms of transmission cost, T-2D1, T-2D2, and DP-2D are about $\frac{9}{2}$, 3, and 2 times the lower bound, respectively. Our T4 requires the least transmission time, about $\frac{65}{48} \approx 1.35$ times the lower bound.

Table 4 shows the startup and transmission costs of our and others' algorithms on a 3D $2^d \times 2^d \times 2^d$ torus. In terms of transmission cost, T-3D1, T-3D2, and DP-3D are about 10, 3, and 2 times the lower bound, respectively. Our C64 requires transmission time of about $\frac{65}{48} \approx 1.35$ times the lower bound.

From Table 3 and Table 4, we establish in Table 5 the theoretical speedups on transmission time of our T4 over T-2D1, T-2D2, DP-2D, and our C64 over T-3D1, T-3D2, DP-3D, when d approaches infinity. These speedups can be used as a reference in our following analyses when we compare other algorithms to our algorithms.

Next, we further study the impact of ratio $\frac{t_s}{b \cdot t_x}$. We plot Fig. 9 using different ratios of $\frac{t_s}{b \cdot t_x} =$ 2,500, 500, 100, and 1 at different torus sizes. The plots are obtained by dividing the latency of other algorithms by that of ours (thus, a speedup value larger than 1 indicates the advantage of our algorithm). The largest ratio, 2,500, is chosen for the following reason: In Intel Paragon, $t_s = 216us$ and $t_x = 0.0226us/\text{byte}$ [12]; letting $b = 4$ we have $\frac{t_s}{b \cdot t_x} \approx 2,500$. We observe that, in most cases, the speedup is larger than 1 for all ratios of $\frac{t_s}{b \cdot t_x}$. Only when d is small ($d = 4 \sim 6$ in Fig. 9a and $d = 4 \sim 5$ in Fig. 9b, c, d, the speedups are less significant because our algorithms take more steps (startup times) than others. After $d \geq 6$, the speedups will approach the values in Table 5 because the transmission costs will become the dominating factor.

In Fig. 10, we take a closer look at the relationship between the speedup and the message block size b by fixing the ratio $\frac{t_s}{t_x}$ at 10,000, 2,000, and 400. Fig. 10a shows the speedups obtained by our T4 when $d = 4$ (a 16×16 torus). The speedup of T4 over T-2D1 is less significant when b is small ($b = 4 \sim 16$) because T-2D1 has the lowest startup cost. Thus, a higher ratio of $\frac{t_s}{t_x}$ will lead to a lower speedup. When b gets larger, the transmission time will dominate the overall cost, and the speedup will reach a stable value of 1.89 for T-2D1/T4, 1.33 for T-2D2/T4, and a stable value of 1.11 for DP-2D/T4. Fig. 10b shows a similar trend when the network size is $d = 5$ (a 32×32 torus), but with a higher stable speedup value of 2.94 for T-2D1/T4, 1.81 for T-2D2/T4, and of 1.36 for DP-2D/T4. In Fig. 10c, the

TABLE 3
Comparison of Startup and Transmission Costs of Complete Exchange Schemes on a 2D $2^d \times 2^d$ Torus

Algorithm		Startup time ($\times t_s$)	Transmission time ($\times b \cdot t_x$)
T-2D1	($d \geq 4$)	$3(d-1)$	$9 \cdot 2^{3d-4} + (d^2 - 5d + 3)2^{2d-1}$
T-2D2	($d \geq 4$)	$3 \cdot 2^{d-2}$	$3 \cdot 2^{3(d-1)}$
DP-2D		$(2^{d-1} + 2)$	$2^{3d-2} + 2^{2d}$
T4	($d = 4$)	$4d - 6$	$\frac{1}{3} \cdot \frac{31}{32} 2^{3d-1} + 5 \cdot 2^{2d-2} + \frac{1}{3} 2^{d+2}$
	($5 \leq d \leq 6$)		$\frac{1}{3} \cdot \frac{31}{32} 2^{3d-1} + 5 \cdot 2^{2d-2} + \frac{5}{3} 2^{d+2}$
	($d \geq 7$)		$\frac{1}{3} \cdot \frac{65}{64} 2^{3d-1} + 2^{2d-1} + \frac{5}{3} 2^{d+2}$

TABLE 4
Comparison of Startup and Transmission Costs of Complete Exchange Schemes on a 3D $2^d \times 2^d \times 2^d$ Torus

Algorithm		Startup time ($\times t_s$)	Transmission time ($\times b \cdot t_x$)
T-3D1	($d \geq 3$)	$4d-2$	$5 \cdot 2^{4d-2} + (d^2 - 3d - 2)2^{3d-1}$
T-3D2	($d \geq 4$)	$3 \cdot 2^{d-2} + d$	$3 \cdot 2^{4d-3} + d \cdot 2^{3d-1}$
DP-3D		$(2^{d-1} + 5)$	$2^{4d-2} + 5 \cdot 2^{3d-1}$
C64	($d = 5$)	$8d - 25$	$\frac{1}{3} \cdot \frac{31}{32} 2^{4d-1} + 5 \cdot 2^{3d} + \frac{1}{3} 2^{2d+7}$
	($6 \leq d \leq 7$)		$\frac{1}{3} \cdot \frac{31}{32} 2^{4d-1} + 5 \cdot 2^{3d} + \frac{5}{3} 2^{2d+4}$
	($d \geq 8$)		$\frac{1}{3} \cdot \frac{65}{64} 2^{4d-1} + 7 \cdot 2^{3d-1} + \frac{5}{3} 2^{2d+4}$

TABLE 5
Theoretical Speedups on Transmission Time of Our Schemes T4 and C64 over Other Schemes When $d \to \infty$

Compared algorithm		Theoretical speedups on transmission time
2D	T-2D1/T4	$\frac{216}{65} \approx 3.32$
	T-2D2/T4	$\frac{144}{65} \approx 2.22$
	DP-2D/T4	$\frac{96}{65} \approx 1.48$
3D	T-3D1/C64	$\frac{480}{65} \approx 7.38$
	T-3D2/C64	$\frac{144}{65} \approx 2.22$
	DP-3D/C64	$\frac{96}{65} \approx 1.48$

speedup approaches 3.47 for T-2D1/T4, 2.05 for T-2D2/T4, and approaches 1.46 for DP-2D/T4 when b increases. Note that the speedup is slightly going down for T-2D2/T4 and DP-2D/T4 when b is small (at $b = 4 \sim 16$) in Fig. 10c. This is because both T-2D2 and DP-2D have a higher $O(2^d)$ startup costs and, thus, when $d = 6 \sim 7$, the save on startup cost is a dominating factor. When b gets larger, the speedup is flattened out because the transmission time becomes the dominating factor again.

To study the speedups of algorithm C64, we show only the cases of $\frac{t_s}{b \cdot t_x} =$ 2,500 and 1 in Fig. 11a and b, respectively, because the trend under different ratios of $\frac{t_s}{b \cdot t_x}$ are similar. Our algorithm C64 can provide significant speedups in most cases.

Finally, we comment on possible inclusion of synchronization and local data movement costs into the performance evaluation. Suppose a global barrier is inserted between each two consecutive phases to synchronize the communication. Since the cost of a barrier operation is fixed for a given torus, a convenient way is to regard this cost as a part of the startup cost (t_s). The local data movement cost refers to the costs of disassembling and reassembling a message to be delivered on a link. This is proportional to the size of the message. Since we already count the size of the message into the communication cost, a convenient way is to regard the local data movement cost per byte as a part of the transmission cost (t_x). To summarize, our timing model can reasonably approximate the communication latency as well as the barrier and local data movement costs if we can properly determine the values of t_s and t_x. This is why we used a wide range of $\frac{t_s}{b \cdot t_x}$ in the above comparison.

7 Concerns of Synchronization

In previous analyses, we have assumed that the communication phases happen perfectly synchronized in a stepwise manner, which is not necessarily true. Consider Fig. 12. After $v_0 \xrightarrow{+} v_2$ is finished in GP_1^+, v_2 does not participate in any of GP_2^+ and SP_2^+, so it will proceed to carry out $v_2 \xrightarrow{+} v_4$ that is supposed to be performed later in SP_1^+. This transmission ($v_2 \xrightarrow{+} v_4$) may contend with $v_0 \xrightarrow{+} v_4$ in GP_2^+ and even $v_0 \xrightarrow{+} v_4$ in SP_2^+ for the physical links between v_2 and v_4. Such contention may disrupt the transmissions in the gather-scatter tree and prolong the overall latency. Below, we discuss three possible approaches, which require different levels of hardware support, to synchronize these phases in our gather-scatter-tree schemes.

7.1 Global Barrier

Apparently, we can add barriers into the following sequence of phases

$$GP_0^+ \to GP_1^+ \to \ldots \to GP_{d-2}^+ \to SP_{d-2}^+ \to SP_{d-3}^+ \\ \to \ldots \to SP_0^+.$$

This requires $2d-3$ barriers (counting the arrows). However, after careful examination, the two sequences $GP_{d-2}^+ \to SP_{d-2}^+$ and $SP_1^+ \to SP_0^+$ are self-synchronized and, thus, the corresponding barriers are unnecessary. So, only $2d-5$ global barriers are required. This approach is appropriate for systems that support hardware barrier synchronization.

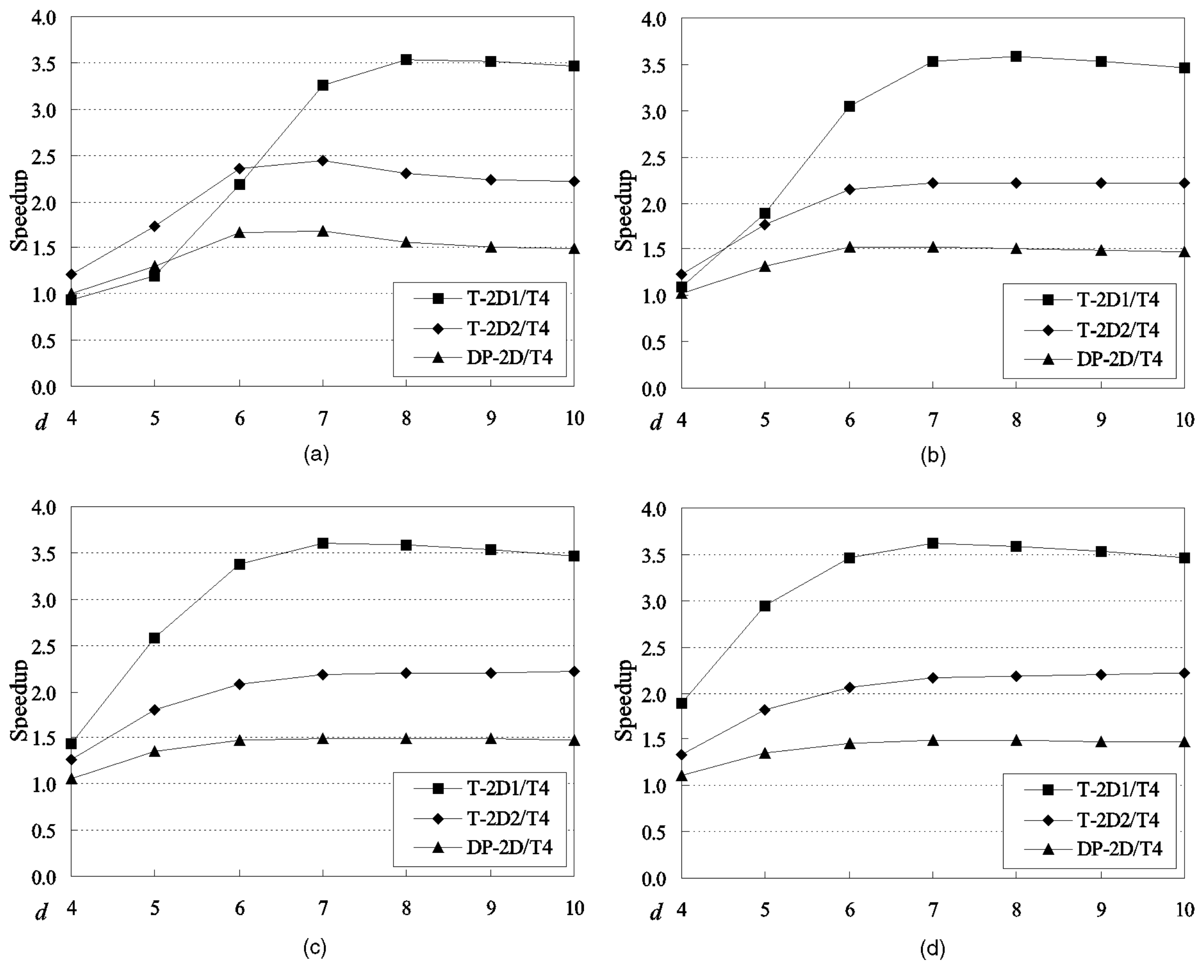


Fig. 9. Speedup of complete exchange obtained by our algorithm T4 at different values of d. (a) $\frac{t_s}{b \cdot t_x} = 2,500$. (b) $\frac{t_s}{b \cdot t_x} = 500$. (c) $\frac{t_s}{b \cdot t_x} = 100$. (d) $\frac{t_s}{b \cdot t_x} = 1$.

7.2 Local Synchronization

The previous approach has assumed that efficient hardware-supported barrier synchronization is available. If not so, software-implemented global barriers may be used. However, software barriers will be much more costly. Fortunately, as shown in Fig. 12, these global barriers can be replaced by *local synchronization* between some pairs of nodes. For instance, we only require that v_0 wait for a signal from v_4 after GP_1^+ to prevent $v_0 \xrightarrow{+} v_4$ of GP_2^+ from contending with $v_2 \xrightarrow{+} v_4$ of GP_1^+. Similarly, we only require that v_2 wait for a signal from v_0 after SP_2^+ to prevent $v_2 \xrightarrow{+} v_4$ of SP_1^+ from starting before $v_0 \xrightarrow{+} v_4$ of SP_2^+ has finished.

The above signaling can be easily implemented by sending a null message. Thus, the cost should be much less than that of doing global software barriers. In general, after each $v_{i+2^{l-1}} \xrightarrow{+} v_{i+2^l}$ of GP_{l-1}^+, $l = 1, \dots, d-2$, we need to add a signaling $v_{i+2^l} \xrightarrow{-} v_i$ for each $v_i \in V_l$. This will eliminate the contention between $v_{i+2^{l-1}} \xrightarrow{+} v_{i+2^l}$ of GP_{l-1}^+ and $v_i \xrightarrow{+} v_{i+2^l}$ of GP_l^+. Also, after each $v_i \xrightarrow{+} v_{i+2^l}$ of SP_l^+, $l = d-2, \dots, 2$, we need to add a signaling $v_i \xrightarrow{+} v_{i+2^{l-1}}$ for each $v_i \in V_l$. This will eliminate the contention between $v_i \xrightarrow{+} v_{i+2^l}$ of SP_l^+ and $v_{i+2^{l-1}} \xrightarrow{+} v_{i+2^l}$ of SP_{l-1}^+.

In total, $2d - 5$ times of local synchronization will be used.

7.3 Systems which Support Prioritized Messages

Hardware support of prioritized message delivery [24] in wormhole networks has received some attention recently. In such systems, messages with higher priorities can preempt the network resources (input buffer and output channel) of lower priority messages. In [24], a throttle mechanism is proposed to preserve input buffers to ensure that, when the header flit with a higher priority arrives, the wormhole router can accept this header and, then, this header (and, thus, those flits following this header) can preempt the output channel if it is currently used by flits with a lower priority. After the tail flit of the higher priority worm leaves the router, the preempted channel is returned to the original worm.

If the above prioritized message delivery is supported by the underlying system, we may assign higher priorities to messages in earlier phases to prevent them from being blocked by messages of later phases. In this way, the overall latency will not be degraded even if some transmissions of posterior phases start before prior phases. To support our schemes, the system should provide at least a number of priority levels equal to the number of phases.

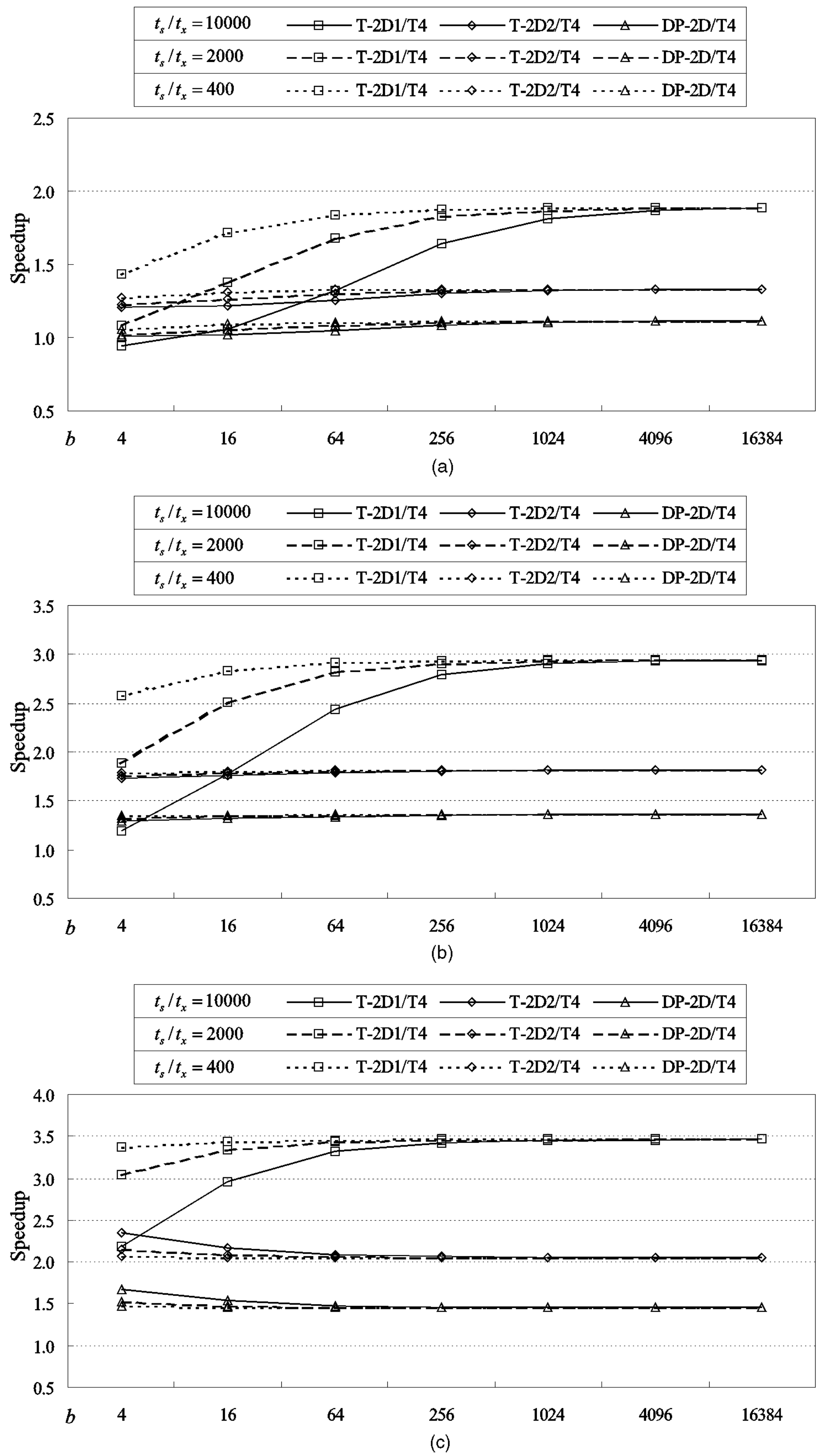


Fig. 10. Speedup of complete exchange obtained by our algorithm T4 vs. block size b on a $2^d \times 2^d$ torus with $\frac{t_s}{t_x} = 10,000$, $\frac{t_s}{t_x} = 2,000$, and $\frac{t_s}{t_x} = 400$. (a) $d = 4$. (b) $d = 5$. (c) $d = 6$.

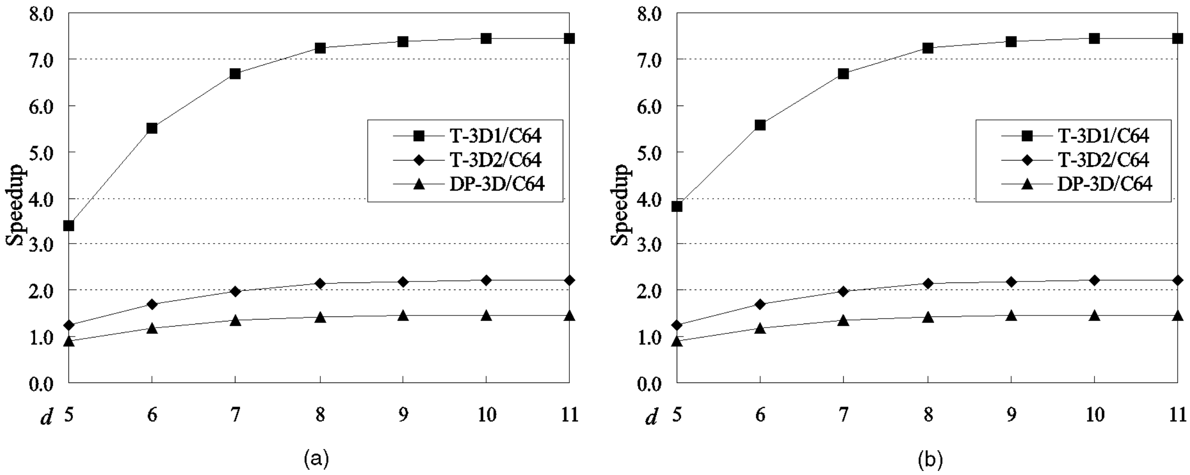


Fig. 11. Speedup of complete exchange obtained by our algorithm C64 at different values of d. (a) $\frac{t_s}{b \cdot t_x} = 2,500$. (b) $\frac{t_s}{b \cdot t_x} = 1$.

8 Conclusions

In this paper, we have presented a systematic solution to perform complete exchange in a torus network using wormhole routing. The solution can be used on nonsquare, non-power-of-2, any-dimensional tori, and this is the first result known to us with such generality in the literature. Interesting techniques used in this paper include the gather-scatter tree structure, network-partitioning approach, and dimension-by-dimension strategy to optimize the startup and transmission costs subject to wormhole routing. Performance-wise, when the torus is square, our 2D and 3D schemes incur asymptotically optimal startup and transmission time (about 2 and $\frac{8}{3}$ times the startup lower bound and both 1.35 times the transmission lower bound for 2D and 3D schemes, respectively). Numerical evaluation has shown significant speedup of these schemes over existing schemes at various communication parameters. Future research may be on reducing the constants associated with the startup and transmission complexities.

Appendix

For convenience, we define $\Phi^+ = 2^{d-1} + 1$. Intuitively, this is the number of destination nodes a node needs to cover in the positive tree plus one (by "plus one," we imagine that the node itself is also a destination).

Proof of Lemma 2. Consider the transmission $v_i \xrightarrow{+} v_{i+2^l}$ of GP_l^+. There are two cases (recall the discussion in Section 2.2).

Case 1: $v_i \in \hat{V}_{l+1}$. Since $v_i \in V_l$, blocks in B_i are gathered from 2^l nodes in $GC_l^+(v_i)$. According to the algorithm, any $b_s^t \in B_i$ that $v_t \notin SC_{l+1}^+(v_i)$ will be sent (refer to Fig. 13a). This implies that any block in B_i whose destination is farther than $v_{i+2^{l+1}-1}$ will be sent. Thus, for first source $v_{i-(2^l-1)} \in GC_l^+(v_i)$, blocks $b_{i-(2^l-1)}^{(i+2^{l+1}) \triangleright (i-(2^l-1)+(\Phi^+-1))}$ will be sent in $v_i \xrightarrow{+} v_{i+2^l}$, where $v_{i-(2^l-1)+(\Phi^+-1)}$ is the farthest destination of the source in the positive tree. This includes $\Phi^+ - 3 \cdot 2^l + 1$ blocks. Similar calculation can be done for other sources in $GC_l^+(v_i)$ (e.g., for $v_{i-(2^l-1)+1} \in GC_l^+(v_i)$, in total $\Phi^+ - 3 \cdot 2^l + 2$ blocks $b_{i-(2^l-1)+1}^{(i+2^{l+1}) \triangleright (i-(2^l-1)+1+(\Phi^+-1))}$ will be sent). Summing these together, node v_i needs to send

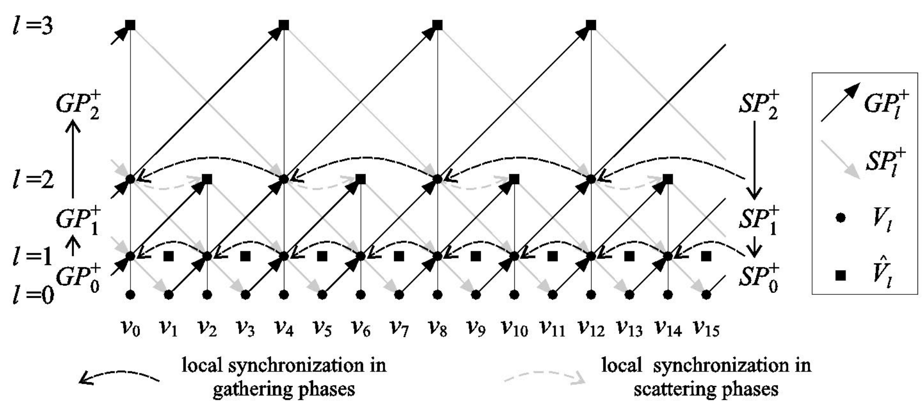


Fig. 12. Additional barriers/synchronization in the positive gather-scatter tree of a 16-node ring to avoid link contention.

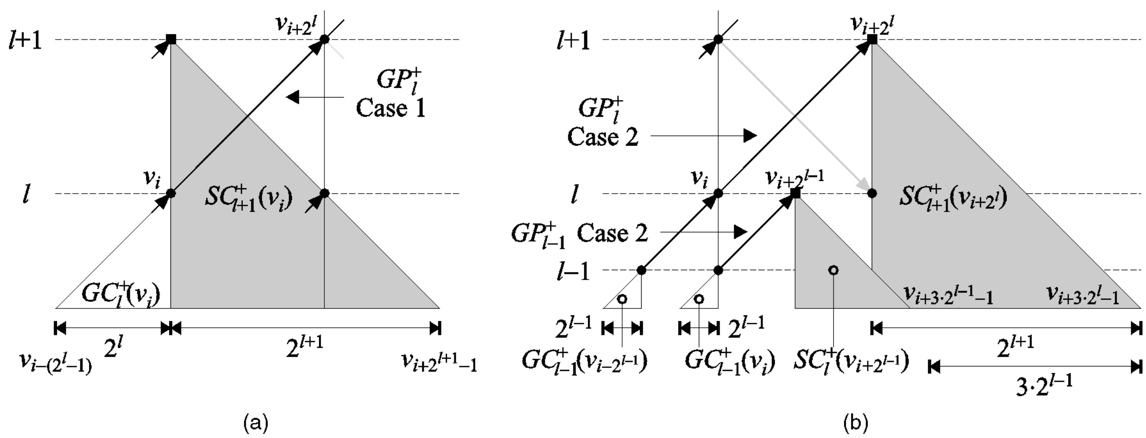


Fig. 13. The source and destination nodes involved in (a) Case 1 and (b) Case 2 transmissions in GP_l^+.

$$\sum_{j=0}^{2^l-1}(\Phi^+ - 3\cdot 2^l + 1 + j) = 2^{d+l-1} - 5\cdot 2^{2l-1} + 3\cdot 2^{l-1}$$

blocks. Note that the above analysis is correct since all terms in $\sum$ are positive numbers.

Case 2: $v_i \in V_{l+1}$. First, consider $1 \le l \le d-3$. According to the algorithm, any $b_s^t \in B_i$ that $v_t \in SC_{l+1}^+(v_{i+2^l})$ will be sent to v_{i+2^l}. However, there is some intersection between $SC_l^+(v_{i+2^{l-1}})$ and $SC_{l+1}^+(v_{i+2^l})$ (see the illustration in Fig. 13b). So, the analysis depends on the location of v_s.

1. $v_s \in GC_{l-1}^+(v_{i-2^{l-1}})$ (first half of $GC_l^+(v_i)$): Any b_s^t that $v_t \in SC_{l+1}^+(v_{i+2^l})$ will be sent in $v_i \xrightarrow{+} v_{i+2^l}$. So, in total, $2^{l-1}\cdot 2^{l+1} = 2^{2l}$ blocks $(b_s^{i+2^l \triangleright i+2^l+2^{l+1}-1})$ will be sent.
2. $v_s \in GC_{l-1}^+(v_i)$ (second half of $GC_l^+(v_i)$): Any b_s^t that $v_t \in SC_l^+(v_{i+2^{l-1}}) \cap SC_{l+1}^+(v_{i+2^l})$ has been sent in $v_i \xrightarrow{+} v_{i+2^{l-1}}$ of the previous GP_{l-1}^+. So, only the remaining b_s^t that $v_t \in SC_{l+1}^+(v_{i+2^l}) - SC_l^+(v_{i+2^{l-1}})$ needs to be sent. Therefore, in total, $2^{l-1}\cdot(3\cdot 2^{l-1}) = 3\cdot 2^{2l-2}$ blocks $(b_s^{i+3\cdot 2^{l-1} \triangleright i+3\cdot 2^l-1})$ will be sent.

As a result, $2^{2l} + 3\cdot 2^{2l-2} = 7\cdot 2^{2l-2}$ blocks are sent in this case. Note that the above analysis is valid as every v_s considered has one block for the farthest destination $v_{i+3\cdot 2^l-1}$ when $1 \le l \le d-3$. As for $l=0$, exactly two blocks $b_i^{i+1 \triangleright i+2}$ will be sent.

The lemma then follows by taking the maximum of the costs incurred by the above two cases. Note that, when $l=0$, the cost of Case 2 (two blocks) is overwhelmed by that of Case 1. □

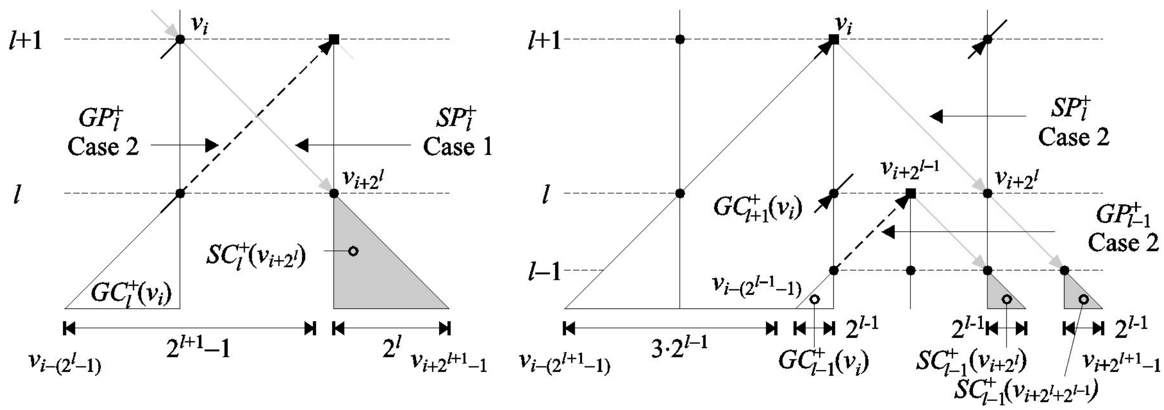
Proof of Lemma 3. The proof is similar to that of Lemma 2. Consider the transmission $v_i \xrightarrow{+} v_{i+2^{d-2}}$ of GP_{d-2}^+. As v_i must be in $\hat{V}_{d-1}$, only Case 2 in the proof of Lemma 2 sustains. According to the algorithm, any $b_s^t \in B_i$ that $v_t \in SC_{d-1}^+(v_{i+2^{d-2}})$ will be sent to $v_{i+2^{d-2}}$. However, there is an intersection between $SC_{d-2}^+(v_{i+2^{d-3}})$ and $SC_{d-1}^+(v_{i+2^{d-2}})$. So, the analysis depends on the location of v_s.

1. $v_s \in GC_{d-3}^+(v_{i-2^{d-3}})$ (first half of $GC_{d-2}^+(v_i)$): Any b_s^t that $v_t \in SC_{d-1}^+(v_{i+2^{d-2}})$ will be sent. For source $v_{i-(2^{d-2}-1)} \in GC_{d-3}^+(v_{i-2^{d-3}})$, there are $\Phi^+ - 2^{d-1} + 1$ blocks, i.e., $b_{i-(2^{d-2}-1)}^{i+2^{d-2} \triangleright i-(2^{d-2}-1)+(\Phi^+-1)}$, to be sent. Similar calculation can be done for other sources (e.g., for $v_{i-(2^{d-2}-1)+1}$, $\Phi^+ - 2^{d-1} + 2$ blocks will be sent). Summing these together, $\sum_{j=0}^{2^{d-3}-1}(\Phi^+ - 2^{d-1} + 1 + j) = 2^{2d-7} + 3\cdot 2^{d-4}$ blocks will be sent.
2. $v_s \in GC_{d-3}^+(v_i)$ (second half of $GC_{d-2}^+(v_i)$): The blocks to be sent are those b_s^t that $v_t \in SC_{d-1}^+(v_{i+2^{d-2}}) - SC_{d-2}^+(v_{i+2^{d-3}})$. For source $v_{i-(2^{d-3}-1)} \in GC_{d-3}^+(v_i)$, there are $\Phi^+ - 2^{d-1} + 1$ blocks, i.e., $b_{i-(2^{d-3}-1)}^{i+3\cdot 2^{d-3} \triangleright i-(2^{d-3}-1)+(\Phi^+-1)}$, to be sent. Similar calculation can be done for other sources (e.g., for $v_{i-(2^{d-3}-1)+1}$, $\Phi^+ - 2^{d-1} + 2$ blocks will be sent). Summing these together, $\sum_{j=0}^{2^{d-3}-1}(\Phi^+ - 2^{d-1} + 1 + j) = 2^{2d-7} + 3\cdot 2^{d-4}$ blocks will be sent.

Therefore, we need to send $2(2^{2d-7} + 3\cdot 2^{d-4}) = 2^{2d-6} + 3\cdot 2^{d-3}$ blocks in $v_i \xrightarrow{+} v_{i+2^{d-2}}$ of GP_{d-2}^+. The above analysis is valid for $d \ge 3$. □

Proof of Lemma 4. Consider the transmission $v_i \xrightarrow{+} v_{i+2^{d-2}}$ of SP_{d-2}^+. Blocks in B_i are gathered from all nodes of $GC_{d-1}^+(v_i)$. According to the algorithm, any b_s^t that $v_t \in SC_{d-2}^+(v_{i+2^{d-2}})$ will be sent. However, $SC_{d-2}^+(v_{i+2^{d-2}}) \subset SC_{d-1}^+(v_{i+2^{d-2}})$, where the latter is considered in previous $v_i \xrightarrow{+} v_{i+2^{d-2}}$ of GP_{d-2}^+. So, the analysis depends on the location of v_s.

1. $v_s \in GC_{d-2}^+(v_{i-2^{d-2}})$ (first half of $GC_{d-1}^+(v_i)$): For the first destination $v_{i+2^{d-2}}$, any b_s^t that $v_s \in GC_{d-2}^+(v_{i-2^{d-2}})$ and $dist^+(v_s, v_{i+2^{d-2}}) < \Phi^+$ will be sent. The only block satisfied this condition is $b_{i-2^{d-2}}^{i+2^{d-2}}$. There is no block to be sent to other destinations in $SC_{d-2}^+(v_{i+2^{d-2}})$.
2. $v_s \in GC_{d-2}^+(v_i)$ (second half of $GC_{d-1}^+(v_i)$): In GP_{d-2}^+, any b_s^t that $v_t \in SC_{d-1}^+(v_{i+2^{d-2}})$ has already



(a) (b)

Fig. 14. Source and destination nodes involved in (a) Case 1 and (b) Case 2 transmissions in SP_l^+.

been sent to $v_{i+2^{d-2}}$. Thus, there is no block left to be sent for these sources.

Consequently, one block will be sent in $v_i \xrightarrow{+} v_{i+2^{d-2}}$ of SP^+_{d-2}. (An instance of this lemma can be observed in Fig. 3, where v_0 is the source and v_8 is the destination.) □

Proof of Lemma 5. Consider the transmission $v_i \xrightarrow{+} v_{i+2^l}$ of SP_l^+. There are two cases.

Case 1: $v_i \in V_{l+1}$ $(v_{i+2^l} \in \hat{V}_{l+1})$. Any $b_s^t \in B_i$ that $v_t \in SC_l^+(v_{i+2^l})$ will be sent. As shown in Fig. 14a, the transmission $v_i \xrightarrow{+} v_{i+2^l}$ in GP_l^+ has already moved blocks from sources in $GC_l^+(v_i)$ to v_{i+2^l} for destinations in $SC_{l+1}^+(v_{i+2^l})$. Since $SC_l^+(v_{i+2^l}) \subset SC_{l+1}^+(v_{i+2^l})$, any b_s^t which remains in B_i such that $v_t \in SC_l^+(v_{i+2^l})$ must have v_s farther than $v_{i-(2^l-1)}$. Thus, for the first destination v_{i+2^l}, $2^{d-1} - 2^{l+1} + 1$ blocks $(b^{i+2^l}_{i+2^l-(\Phi^+-1) \triangleright i-2^l})$ will be sent, where $v_{i+2^l-(\Phi^+-1)}$ is the farthest source that v_{i+2^l} could involve. Similar calculation can be done for other destinations in $SC_l^+(v_{i+2^l})$ (e.g., for v_{i+2^l+1}, $(2^{d-1} - 2^{l+1} + 1) - 1$ blocks are sent). Summing these together,

$$\sum_{j=0}^{2^l-1} (2^{d-1} - 2^{l+1} + 1 - j) = 2^{d+l-1} - 5 \cdot 2^{2l-1} + 3 \cdot 2^{l-1}$$

blocks will be sent. The analysis is valid since all terms in $\sum$ are positive.

Case 2: $v_i \in \hat{V}_{l+1}$ $(v_{i+2^l} \in V_{l+1})$. First, consider $1 \le l \le d-3$. Since $v_i \in \hat{V}_{l+1}$, blocks in B_i are gathered from 2^{l+1} nodes $(GC_{l+1}^+(v_i))$. Any $b_s^t \in B_i$ that $v_t \in SC_l^+(v_{i+2^l})$ will be sent. Some blocks heading for destinations in $SC_{l-1}^+(v_{i+2^l}) \subset SC_l^+(v_{i+2^l})$ have already been sent in GP_{l-1}^+ (Fig. 14b). So, the analysis depends on the location of v_t.

1. $v_t \in SC_{l-1}^+(v_{i+2^l})$ (first half of $SC_l^+(v_{i+2^l})$): Source nodes in $GC_{l-1}^+(v_i)$ already have Case 2 $v_i \xrightarrow{+} v_{i+2^{l-1}}$ in GP_{l-1}^+ move blocks whose destinations include those in $SC_{l-1}^+(v_{i+2^l})$. Thus, any b_s^t that $v_s \in GC_{l+1}^+(v_i) - GC_{l-1}^+(v_i)$ will be sent. That is, $3 \cdot 2^{l-1}$ blocks $b^t_{i-(2^{l+1}-1) \triangleright i-(2^{l-1}-1)-1}$ will be sent to v_{i+2^l} for each v_t. So, in total, $2^{l-1} \cdot 3 \cdot 2^{l-1} = 3 \cdot 2^{2l-2}$ blocks will be sent.
2. $v_t \in SC_{l-1}^+(v_{i+2^l+2^{l-1}})$ (second half of $SC_l^+(v_{i+2^l})$): Any b_s^t that $v_s \in GC_{l+1}^+(v_i)$ will be sent. That is, 2^{l+1} blocks $b^t_{i-(2^{l+1}-1) \triangleright i}$ will be sent for each v_t. So, in total, $2^{l-1} \cdot 2^{l+1} = 2^{2l}$ blocks are sent.

As a result, $2^{2l} + 3 \cdot 2^{2l-2} = 7 \cdot 2^{2l-2}$ blocks are sent in a Case 2 transmission of SP_l^+. Note that the above analysis is valid as the farthest distance $dist^+(v_{i-(2^{l+1}-1)}, v_{i+2^{l+1}-1}) < \Phi^+$ when $1 \le l \le d-3$. As for SP_0^+, exactly two blocks, i.e., $b^{i+1}_{i-1 \triangleright i}$, will be sent.

The lemma then follows by taking the maximum of the costs incurred by the above two cases. Note that, when $l = 0$, the cost of Case 2 (two blocks) is overwhelmed by that of Case 1. □

Proof of Corollary 1. Note that the analyzed result of GP_l^+ (Lemma 2) is identical to that of SP_l^+ (Lemma 5). In each of GP_l^+ or SP_l^+, two message sizes are compared for finding the transmission time of that phase. If $l \le d-4$, the message size of Case 1 is always larger than that of Case 2. Therefore, the transmission time is $(2^{d+l-1} - 5 \cdot 2^{2l-1} + 3 \cdot 2^{l-1}) \cdot b \cdot t_x$. In case of $l = d-3$, the message size of Case 1 transmission is larger if $3 \le d \le 5$, but, if $d \ge 6$, that of Case 2 is larger. Thus, by substituting l with $d-3$, we have $3 \cdot 2^{2d-7} + 3 \cdot 2^{d-4}$ for $3 \le d \le 5$ and $7 \cdot 2^{2(d-3)-2} = 7 \cdot 2^{2d-8}$ for $d \ge 6$. □

Proof of Lemma 6. The total communication time is

$$T_{1D}^+(d,b) = \sum_{l=0}^{d-3} T_{1D,GP_l^+}(d,b) + T_{1D,GP_{d-2}^+}(d,b) + T_{1D,SP_{d-2}^+}(d,b) + \sum_{l=0}^{d-3} T_{1D,SP_l^+}(d,b).$$

□

Proof of Theorem 1. Since $\Phi^+ > \Phi^- = 2^{d-1}$, the latency of a negative phase is smaller than that of a corresponding positive phase. Therefore, we need to consider only the latency of positive part plus the additional cost for overlapping. We adjust the latency for positive phases GP_0^+ and GP_1^+ according to Section 2.4. Depending on the location of v_s, we have following four cases:

1. $v_s \in \hat{V}_2$: $v_s \xrightarrow{+} v_{s+1}$ of GP_0^+ and $v_{s+1} \xrightarrow{+} v_{s+2}$ of SP_0^+ are removed. Thus, b_s^{s+1} is sent in SP_0^+ (Case 1), and b_s^{s+2} is sent in SP_1^+ (Case 2).
2. $v_{s+1} \in \hat{V}_2$: $v_s \xrightarrow{+} v_{s+1}$ of SP_0^+ is removed. Thus, b_s^{s+1} is sent in GP_0^+ (Case 1).
3. $v_{s+2} \in \hat{V}_2$: $v_s \xrightarrow{+} v_{s+1}$ of GP_0^+ and $v_{s+1} \xrightarrow{+} v_{s+2}$ of SP_0^+ are removed. Thus, b_s^{s+1} is sent in SP_0^+ (Case 1), and b_s^{s+2} is sent in GP_1^+ (Case 2).
4. $v_{s+3} \in \hat{V}_2$: $v_s \xrightarrow{+} v_{s+1}$ of SP_0^+ is removed. Thus, b_s^{s+1} is sent in GP_0^+ (Case 1).

Because the transmission time of each of GP_0^+, SP_0^+, GP_1^+, and SP_1^+ is dominated by Case 1 transmissions, the total transmission time is increased by only two blocks, one added to GP_0^+ and another to SP_0^+, for $d \geq 4$. If $d = 3$, GP_1^+ is the special case GP_{d-2}^+ in which every transmission is considered as Case 2. Thus, the additional one block transmitted in GP_1^+ should be taken into account, where no additional block is transmitted in SP_1^+ since SP_1^+ is the special case SP_{d-2}^+. Consequently, three blocks are added to the total transmission time for $d = 3$. □

ACKNOWLEDGMENTS

This work was supported in part by the National Science Council of the Republic of China under Grants NSC88-2213-E-008-020, NSC88-2213-E-008-027, and NSC89-2213-E-008-024. A preliminary version of this paper appeared in the *Proceedings of the 1997 International Conference on Parallel and Distributed Systems* [33].

REFERENCES

[1] A. Bagchi, E.F. Schmeichel, and S.L. Hakimi, "Parallel Information Dissemination by Packets," *SIAM J. Computing,* vol. 23, no. 2, pp. 355-372, Apr. 1994.

[2] P.E. Berman, L. Gravano, G.D. Pifarre, and J.C. Sanz, "Adaptive Deadlock- and Livelock-Free Routing with All Minimal Paths in Torus Networks," *Proc. ACM Symp. Parallel Algorithms and Architecture,* pp. 3-12, 1992.

[3] S.H. Bokhari and H. Berryman, "Complete Exchange on a Circuit Switched Mesh," *Proc. Scalable High Performance Computing Conf.,* 1992.

[4] R. Cypher and L. Gravano, "Adaptive, Deadlock-Free Packet Routing in Torus Networks with Minimal Storage," *Proc. Int'l Conf. Parallel Processing,* vol. III, pp. 204-211, 1992.

[5] W.J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels," *IEEE Trans. Parallel and Distributed Systems,* vol. 4, no. 4, pp. 466-475, Apr. 1993.

[6] V.V. Dimakopoulos and N.J. Dimopoulos, "A Theory for Total Exchange in Multidimensional Interconnection Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 7, pp. 639-649, July 1998.

[7] S. Fujita and M. Yamashita, "Fast Gossiping on Mesh-Bus Computers," *IEEE Trans. Computers,* vol. 45, no. 11, pp. 1,326-1,330, Nov. 1996.

[8] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM 3 Users Guide and Reference Manual.* Oak Ridge, Tenn.: Oak Ridge Nat'l Laboratory, May 1994.

[9] S. Gupta, S. Hawkinson, and B. Baxter, "A Binary Interleaved Algorithm for Complete Exchange on a Mesh Architecture," technical report, Intel Corp., 1994.

[10] S. Hinrichs, C. Kosak, D.R. O'Hallaron, T.M. Stricker, and R. Take, "An Architecture for Optimal All-to-All Personalized Communication," Technical Report CMU-CS-94-140, School of Computer Science, Carnegie Mellon Univ., Sept. 1994.

[11] C. Ho and M. Kao, "Optimal Broadcast on Hypercubes with Wormhole and e-Cube Routings," *Proc. Int'l Conf. Parallel and Distributed Systems,* pp. 694-697, Taipei, Taiwan, 1993,

[12] B.H. Juurlink, "Experimental Validation of Parallel Computation Models on the Intel Paragon," *Proc. Int'l Parallel Processing Symp.,* 1998.

[13] S.D. Kaushik, C.-H. Huang, J. Ramanujam, and P. Sadayappan, "Multi-Phase Array Redistribution: Modeling and Evaluation," *Proc. Int'l Parallel Processing Symp.,* pp. 441-445, 1995.

[14] D.W. Krumme, G. Cybenko, and K.N. Venkataraman, "Gossiping in Minimal Time," *SIAM J. Computing,* vol. 21, no. 1, pp. 111-139, Feb. 1992.

[15] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms.* Redwood City, Calif.: Benjamin/Cummings, 1994.

[16] X. Lin, P.K. McKinley, and L.M. Ni, "Deadlock-Free Multicast Wormhole Routing in 2-D Mesh Multicomputers," *IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 8, pp. 793-804, Aug. 1994.

[17] P.K. McKinley, Y.-J. Tsai, and D.F. Robinson, "Collective Communication in Wormhole-Routed Massively Parallel Computers," *Computer,* vol. 28, no. 12, pp. 39-50, Dec. 1995.

[18] P.K. McKinley, H. Xu, A.-H. Esfahanian, and L.M. Ni, "Unicast-Based Multicast Communication in Wormhole-Routed Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 12, pp. 1,252-1,265, Dec. 1994.

[19] Message Passing Interface Forum, "Document for Standard Message-Passing Interface," Technical Report CS-93-214, Univ. of Tennessee, Nov. 1993.

[20] P. Michallon and D. Trystram, "Minimum Depth Arcs-Disjoint Spanning Trees for Broadcasting on Wrap-Around Meshes," *Proc. Int'l Conf. Parallel Processing,* vol. 1, pp. 80-83, 1995.

[21] L.M. Ni and P.K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *Computer,* vol. 26, no. 2, pp. 62-76, Feb. 1993.

[22] D.F. Robinson, P.K. McKinley, and B.H.C. Cheng, "Optimal Multicast Communication in Torus Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 6, no. 10, pp. 1,029-1,042, Oct. 1995.

[23] 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.

[24] H. Song, B. Kwon, and H. Yoon, "Throttle and Preempt: A New Flow Control for Real-Time Communications in Wormhole Networks," *Proc. Int'l Conf. Parallel Processing,* pp. 198-202, 1997.

[25] Y.-J. Suh and S. Yalamanchili, "Algorithms for All-to-All Personalized Exchange in 2D and 3D Tori," *Proc. Int'l Parallel Processing Symp.,* 1996.

[26] 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.

[27] N.S. Sundar, D.N. Jayasimha, D.K. Panda, and P. Sadayappan, "Hybrid Algorithms for Complete Exchange in 2D Meshes," *Proc. Int'l Conf. Supercomputing,* pp. 181-188, 1996.

[28] S. Takkella and S. Seidel, "Complete Exchange and Broadcast Algorithm for Meshes," *Proc. Scalable High-Performance Computing Conf.,* pp. 422-428, 1994.

[29] R. Thakur, A. Choudhary, and G. Fox, "Complete Exchange on a Wormhole Routed Mesh," *Proc. Int'l Workshop Modeling, Analysis, and Simulation of Computer and Telecomm. System,* pp. 131-135, 1994.

[30] Y.-J. Tsai and P.K. Mckinley, "A Broadcasting Algorithm for All-Port Wormhole-Routed Torus Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 8, pp. 876-885, Aug. 1996.

[31] Y.-C. Tseng and S.K.S. Gupta, "All-to-All Personalized Communication in a Wormhole-Routed Torus," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 5, pp. 498-505, May 1996.

[32] Y.-C. Tseng, T.-H. Lin, S.K.S. Gupta, and D.K. Panda, "Bandwidth-Optimal Complete Exchange on Wormhole-Routed 2D/3D torus Networks: A Diagonal-Propagation Approach," *IEEE Trans. Parallel and Distributed Systems,* vol. 8, no. 4, pp. 380-396, Apr. 1997.
[33] Y.-C. Tseng, S.-Y. Ni, and J.-P. Sheu, "Toward Optimal Complete Exchange on Wormhole-Routed Tori," *Proc. Int'l Conf. Parallel and Distributed Systems,* pp. 96-103, 1997.
[34] Y.-C. Tseng, S.-Y. Wang, and C.-W. Ho, "Efficient Broadcasting in Wormhole-Routed Multicomputers: A Network-Partitioning Approach," *IEEE Trans. Parallel and Distributed Systems,* vol. 10, no. 1, pp. 44-61, Jan. 1999.
[35] S.-Y. Wang, Y.-C. Tseng, and C.-W. Ho, "Efficient Multicast in Wormhole-Routed 2D Mesh/Torus Multicomputers: A Network-Partitioning Approach," *Proc. Symp. Frontiers of Massively Parallel Computation,* pp. 42-49, 1996.

Yu-Chee Tseng received his BS and MS degrees in computer science from the National Taiwan University and the National Tsing-Hua University in 1985 and 1987, respectively. He worked for D-LINK Inc. as an engineer in 1990. He obtained his PhD in computer and information science from the Ohio State University in January of 1994. From 1994 to 1996, he was an associate professor in the Department of Computer Science, Chung-Hua University. He joined the Department of Computer Science and Information Engineering, National Central University in 1996, and became a professor there in 1999. Dr. Tseng served as a program committee member for the International Conference on Parallel and Distributed Systems, 1996, the Internationall Conference on Parallel Processing, 1998, and the Internationall Conference on Distributed Computing Systems, 2000. He was a workshop co-chair for the National Computer Symposium, 1999. His research interests include mobile computing, wireless communication, network security, parallel and distributed computing, and computer architecture.

Dr. Tseng is a member of the IEEE Computer Society and the ACM.

Sze-Yao Ni received his BS and MS degrees in electrical engineering from the National Central University, Taiwan, Republic of China, in 1991 and 1993, respectively. He is expected to receive his PhD in computer science and information engineering from the National Central University in the spring of 2000. His research interests include computer architecture, parallel and distributed computing, and mobile computing. He is a student member of the IEEE Computer Society and a member of the ACM and the Phi Tau Phi Society.

Jang-Ping Sheu received the BS degree in computer science from Tamkang University, Taiwan, Republic of China, in 1981, and the MS and PhD degrees in computer science from the National Tsing Hua University, Taiwan, Republic of China, in 1983 and 1987, respectively.

He joined the faculty of the Department of Electrical Engineering, National Central University, Taiwan, Republic of China, as an associate professor in 1987. He is currently a professor and chair of the Department of Computer Science and Information Engineering, National Central University. From March to June of 1995, he was a visiting researcher at the IBM T.J. Watson Research Center, New York. His current research interests include parallelizing compilers, interconnection networks, and mobile computing.

Dr. Sheu is a senior member of the IEEE and a member of the ACM and Phi Tau Phi Society. He is an associate editor of the *Journal of Information Science and Engineering*, *Journal of the Chinese Institute of Electrical Engineering*, and *Journal of the Chinese Institute of Engineers*. He received the Distinguished Research Awards of the National Science Council of the Republic of China in 1993-1994, 1995-1996, and 1997-1998.