Fast Gossiping in Square Meshes/Tori with Bounded-Size Packets

Francis C.M. Lau, *Member*, *IEEE*, and Shi-Heng Zhang

Abstract—Gossiping is the communication problem in which each node has a unique message (token) to be transmitted to every other node. The nodes exchange their tokens by packets. A solution to the problem is judged by how many rounds of packet sending it requires. In this paper, we consider the version of the problem in which small-size packets (each carrying exactly one token) are used, the links (edges) of the network are half-duplex (only one packet can flow through a link at a time), and the nodes are all-port (a node's incident edges can all be active at the same time). This is also known as the H* model. We study the 2D square mesh and the 2D square torus. An improved, asymptotically optimal algorithm for the mesh and an optimal algorithm for the torus are presented.

Index Terms—Gossiping, all-to-all broadcast, total exchange, collective communication, parallel algorithms, interconnection networks, communication optimization, scheduling.

1 Introduction

In parallel and distributed computing, communication among processors is an important issue. Gossiping, also known as total exchange and all-to-all (nonpersonalized) broadcast, is the communication problem in which each processor (or node) has a unique message (or token) to be transmitted to every other processor. Because of its rich communication pattern, gossiping is a useful benchmark for evaluating the communication capability of an interconnection structure. The gossiping problem has been studied extensively during the last two decades; a summary of the results can be found in [12], [10], [13]. Gossiping as an embedded operation is needed in many real computations, such as matrix multiplication, LU-factorization, Householder transformation, direct N-body computation, global processor synchronization, and load balancing. Juurlink et al. [14] cited two specific applications that require gossiping as a subroutine: splitter-based sorting [17] and parallel block predictor-corrector methods to solving ordinary differential equations [21].

Krumme et al. have suggested that the gossiping problem can be studied under four different communication models, which have different restrictions on the use of the links, as well as the ability of a node in handling its incident links [16]. The four models are:

1. the full-duplex, all-port model,
2. the full-duplex, one-port model,
3. the half-duplex, all-port model, and
4. the half-duplex, one-port model, which can be identified by the labels F*, F1, H*, and H1, respectively.

A full-duplex link allows both ends to send/receive a message at the same time; a half-duplex link allows only one end to do so at a time. In the one-port mode, only one of the incident links of a node may be active at a time; all the incident links may be active at the same time in the all-port mode. The four models, therefore, form a spectrum, with F* being the strongest in communication capability and H1 the weakest. Krumme et al. studied the problem for a number of well-known topologies under the H1 model [16] and for the hypercube under both the H* and the H1 model [15].

Bermond et al. [3] have added another dimension to the problem. They suggested that a packet carrying tokens cannot be of infinite size, which a great majority of previous work had assumed. In reality, indeed, a packet's delay is somewhat dependent on its contents, especially in tightly coupled multiprocessors. They studied the gossiping problem under this hypothesis and under the F1 model, deriving results for the complete graph, hypercube, cycle, and path [2]. Bagchi et al. have considered the same, but under the H1 model [4], [5].

In this paper, we adopt the bounded packet size restriction. We can use the parameter p to denote the size of a packet: $p = 1$ means that a packet can carry up to one token, $p = 2$ two tokens, etc. In this paper, we consider only the case of $p = 1$.

The gossiping process advances by rounds (or timesteps) in a lock-step fashion. In each round, a packet can only travel across one edge. We refer to the sending of a packet across an edge a *packet move*. A packet move translates into a unit of communication load that the gossiping algorithm introduces into the network. In general, there are two measures by which a gossiping algorithm may be evaluated: the number of timesteps to complete the gossip and the communication load in terms of packet moves the gossip generates, also referred to as *calls* in some studies (e.g., [1]). In the domain of parallel and distributed computing, the former is by far the more dominant, which is also the measure we are interested in

• *The authors are with the Department of Computer Science and Information Systems, University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: fcmlau@csis.hku.hk.*

Manuscript received 27 Aug. 1997; revised 8 June 2000; accepted 20 Sept. 2001.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 105553.

1045-9219/02/$17.00 © 2002 IEEE

here. Czumaj et al. have studied the time and communication load trade-offs in gossiping under the F1 model [6].

We define $g_p(N)$ to be the time required to complete a gossip for the interconnection network N if the packet size is p. Since only one value of p ($p = 1$) is being considered in this paper, we use $g(N)$ for the gossiping time.

In this paper, our focus is on the 2D mesh and the 2D torus which are important communication fabric for modern parallel machines. Parallel machines that use 2D meshes include the MIT J-Machine [7], the Symult 2010 [22], and the Intel Touchstone [19]. For simplicity, we consider square meshes and tori. Our algorithms can be easily extended to cover the nonsquare ones as well, but the results would be nonoptimal (see discussion in Conclusion). Under the restriction of bounded packet size, the following are the existing results for the mesh and the torus.

- For the F* model and $p = 1$, Soch and Tvrdík obtained the optimal result, $\lceil (mn-1)/2 \rceil$, for the $m \times n$ mesh with the restriction that at least one of m and n must be even [24]; as well as the optimal result, $\lceil (mn-1)/4 \rceil$, for the torus (for $n = 2$ and m odd, one extra step is needed) [29]. Their solutions are also buffer-optimal, requiring auxiliary buffers for at most three packets for tori and buffers for $O(m+n)$ packets for meshes [25].
- For the F1 model and $p = 1$, Bermond et al. gave an algorithm that can solve the problem for the $n \times m$ mesh in $2mn - 3 - \max\{m, n\}$ steps, m and n odd [3].
- For the H* model and $p = 1$, Fujita and Yamashita gave an algorithm that can solve the problem for the $n \times n$ (square) mesh in $n^2/2 + 2n - 3$ and $n^2/2 + 2n - 5/2$ steps for even and odd n, respectively [11].
- For the H1 model and any value of p, Bagchi et al. derived the result, $2mn/p + O(m+n)$, for the $m \times n$ mesh [5].

In this paper, we assume the H* model and $p = 1$. Many of the modern designs of routers use separate controllers to manage the links, which can operate simultaneously in parallel. Both the F* and the H* model are realistic models for router implementation. There are pros and cons to operating a link in half- or full-duplex mode (see the discussion in [9]). One well-known example of H* router is the Network Design Frame [8]. H* can be a special mode of an F* router, where the full-duplex links are set to operate by software in half-duplex mode. This can be the preferred mode of operation if it gives algorithmic advantages while the links' capacity can still be fully utilized. Many F* routers (e.g., the C104 router for the transputer [20]), when operating in full-duplex mode, offer only a fraction of the maximum per-link bandwidth in each direction and, so, the gain in total communication bandwidth versus the half-duplex mode is minimal.

We present an improved algorithm for the square 2D mesh and an algorithm for the square 2D torus; the latter is optimal for the case of even n and two steps away from the optimal for the case of odd n. This work is meant to serve as a starting point for the study of all-to-all communications based on the H* model and under the bounded packet size restriction and, hence, we have avoided such practical issues as start-up times and store-and-forward versus wormhole routing. Start-up times are real and could be substantial especially when the packets are relatively small [14], [26]. Wormhole routing is the preferred routing mode in modern router designs and its distance-insensitivity has prompted many interesting solutions to collective communication problems [23], [27], [28], [30]. These issues need to be addressed in an extended study for which the algorithms and results in this paper may serve as a basis.

2 Preliminaries

An $n \times n$ square mesh or torus has n rows and n columns, both indexed from 0 to $n-1$. A node is uniquely identified by (i, j), where i and j are the node's row and column positions, respectively. For convenience of discussion, we use $v_0, v_1, \ldots, v_{n-1}$ to represent the n nodes in any row or column, with v_c being the *center node*, $c = \lfloor n/2 \rfloor$ for the odd-n case, and $n/2$ for the even-n case. Initially, each node has a token, which is to be sent to every other node.

The following are two simple lower bounds on the gossiping time for the mesh and the torus, respectively.

Lemma 1. *For a mesh of size $n \times n$, M, the lower bound on $g(M)$, is $n^2/2 + n/2$ [11].*

Proof. There are n^2 nodes. Each node's token has to reach $n^2 - 1$ nodes. Hence, the total number of token moves is at least $n^2(n^2-1)$. M has $2n(n-1)$ edges. Therefore, there exists an edge which has to accommodate at least $n^2(n^2-1)/(2n(n-1)) = n^2/2 + n/2$ moves. □

Note that, for the case of $n = 1$, no movement of token is necessary. So, the above is good for $n > 1$.

Lemma 2. *For a torus of size $n \times n$, T, the lower bound on $g(T)$ is $(n^2+1)/2 - 1$ or $n^2/2$ for even n.*

Proof. T has $2n^2$ edges. Hence, the lower bound is equal to $n^2 \times (n^2-1)/2n^2 = n^2/2 - 1/2$, or $n^2/2$ for even n. □

In the next several sections, we will present our solutions to the problem. Note that, if we consider an edge to be a series of "instances" of the edge being at different timesteps, then the above lower bounds correspond to filling up all the instances of all the edges over a period of time. Algorithms that attempt to match these bounds would have to try to occupy every edge at every timestep. Moreover, there should not be any duplication of packets—that is, the same token should not reach the same node more than once.

The two-phase algorithm by Fujita and Yamashita was the first attempt at solving the problem for the square mesh. Initially, the nodes are labeled as *even* or *odd* according to their positions: Node (i, j) is even if $i + j$ is even and odd otherwise. Phase 1 then operates as outlined in Fig. 1. Fig. 2 shows the detailed operations of Phase 1 for a single row (or column) for various mesh cases, where a dot corresponds to an edge and one row of dots corresponds to the entire row or column of edges at a particular timestep. A black dot represents an edge in use (i.e., a packet is being transmitted through the edge), and a white dot an idle edge instance. The dissemination of a token from one node to the next and

▷ **Phase 1.** All even nodes broadcast their tokens to all the nodes in their respective rows and all odd nodes broadcast their tokens to all the nodes in their respective columns.

▷ **Phase 2.** Every row and every column performs a gossip with the tokens collected during Phase 1 along the row/column itself.

Fig. 1. Outline of the two phases.

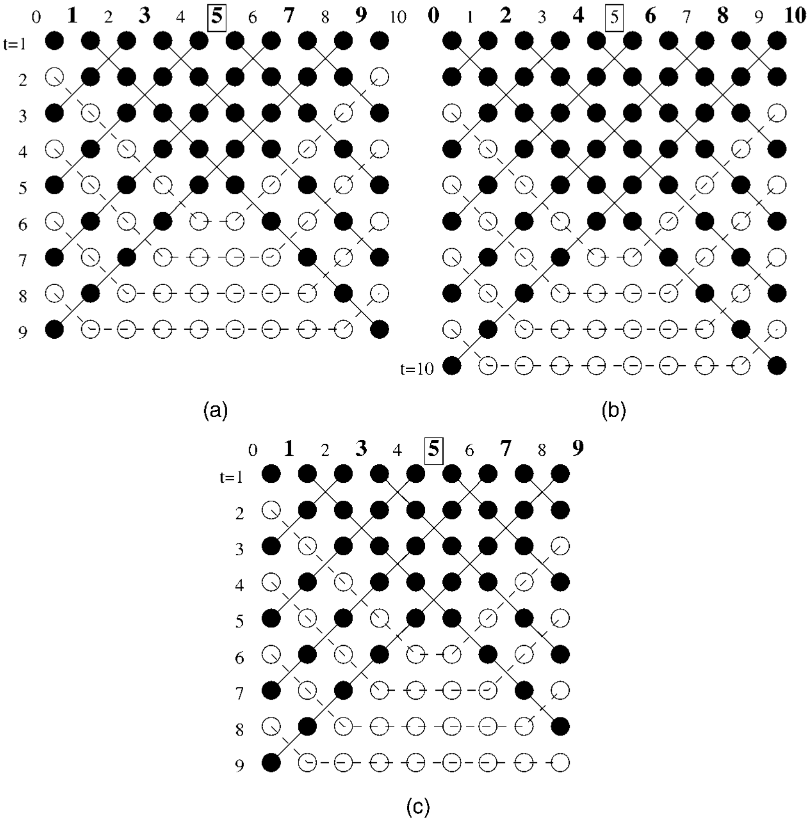

Fig. 2. Dot-wire diagrams: Examples of Phase-1 execution. (a) $n = 11$, (b) $n = 11$, and (c) $n = 10$.

so on is represented by a "wire" of black dots. We call these pictures of dots and wires "dot-wire diagrams."

We can call the Phase-1 algorithm a "half-gossip" algorithm because only half ($n/2$, and ±1 for odd n) of the nodes in a row or column broadcast their tokens. And we call the tokens broadcast by the even nodes during Phase 1 *even tokens* and those by odd nodes *odd tokens*. After Phase 1, each node in a row will have collected $n/2$ (±1 for odd n) column tokens and each node in a column will have collected the same number of row tokens. Then in Phase 2, every row and every column performs a (full) gossip with the tokens collected during Phase 1 along the row/column itself using a gossip algorithm for path.

The Fujita-Yamashita algorithm turned out to be asymptotically optimal ($O(n^2/2)$). We found, however, that this algorithm has two major problems: First, the execution of the Phase-1 algorithm would leave a number of edge instances idle; second, the Phase-2 algorithm, which is a path gossiping algorithm, is not optimal. As a result, there is a gap of $O(3n/2)$ between the upper and the lower bound. Our task is to try to close this gap.

In the following, we adopt the two-phase strategy, but with the following important changes. For the mesh,

- we use a different Phase-2 algorithm, which is an optimal gossiping algorithm for the path, and
- we try to overlap the two phases as much as possible so that the idle edge instances in Phase 1 can be used by Phase 2.

For the torus, we use the original Phase-1 algorithm for the even-n case and a modified version for the odd-n case. For Phase 2, we use a simple but optimal algorithm for gossiping in a cycle.

Some examples of Phase-1 execution are shown in Fig. 2. Note that in a dot-wire diagram, a wire can only go downward and no two wires may cross at a black dot; if they do, that means there is a clash on the use of the same edge.

3 The Gossip Algorithm for a Square Mesh

The algorithm we propose here for the mesh uses the first phase of the Fujita-Yamashita algorithm as its first phase.

```
▷ $R_i = L_i = \{\text{tokens from Phase 1}\}$ for $0 < i < n-1$;
  $R_0 = \{\text{tokens from Phase 1}\}$; $L_{n-1} = \{\text{tokens from Phase 1}\}$;
▷ node $v_i$ (on the left of the center node):
  repeat
      (* the following are in parallel *)
      • if $R_i \neq \emptyset$
            extract the first token from $R_i$ and send it to $v_{i+1}$
        else
            receive a token from $v_{i+1}$ ($L_{i+1}$) and append it to $L_i$ (if $i > 0$);
      • if there is a token from $v_{i-1}$
            append it to $R_i$;
  until done;
▷ node $v_i$ (on the right of the center node):
  repeat
      (* similarly, except all the directions are reversed *)
  until done;
▷ center node $v_c$:
      (* the following are in parallel *)
      • repeat
            receive a token from the left and append it to $R_c$;
        until no more token;
        repeat
            extract the first token from $R_c$ and send it to $v_{c+1}$;
        until $R_c = \emptyset$;
      • (* similarly for the other side *)
```

Fig. 3. The Phase-2 algorithm.

For the second phase, we adopt the path gossiping algorithm introduced in [18], with the modification that each node can hold multiple tokens initially. By proving a lower bound on the gossiping time, we show that this path algorithm is optimal for the second phase. An important feature of the combined algorithm is that it starts both phases at the same time, which allows the second phase to make use of the idle edge instances of the first phase. The resulting algorithm is faster than the Fujita-Yamashita algorithm by approximately n steps.

3.1 Phase 1

We examine the half-gossip algorithm (Fig. 1) which is Phase 1 of the combined algorithm. Our goal is to identify idle edge instances (corresponding to white dots in the dot-wire diagram) arising during the execution of this algorithm. The Phase-2 algorithm will try to fill up these idle instances. There are three cases, two for the case of odd n and one for the case of even n, as shown in Fig. 2. Because only every other node is active, there is no conflict over the use of the edges and, hence, all all the wires are straight lines. We define a *white wire* to be one that comprises an idle instance (or white dot) of every one of the $n-1$ edges. For each of the three cases in Fig. 2, there are exactly $\lfloor n/2 \rfloor - 1$ (for odd n) or $n/2 - 1$ (for even n) such white wires, as indicated by the dashed lines in the figure. These white wires are "concave" when viewed from the top and their "bottom" touches the center node. For the odd-n case, the first white wire occurs at either $t = 2$ (Fig. 2a) or $t = 3$ (Fig. 2b) and, for the even-n case, it occurs at $t = 2$ (Fig. 2c). The time at which the first white wire occurs is crucial: The second phase must be ready by then in order to make use of all the white wires.

3.2 Phase 2—Odd n

Consider, without loss of generality, a row of the mesh. For the odd-n case, because the number of even nodes and the number of odd nodes in a column are not the same, the result of Phase 1 would be that the row has an uneven load for Phase 2 to distribute: Either an even node has one token more than that of an odd node or the other way around. Let's assume the former (the other case is similar), where an even node has $\lceil n/2 \rceil$ tokens and an odd node has $\lfloor n/2 \rfloor$ tokens.

We use the path algorithm introduced in [18] here, which divides the row into two equal halves, with a center node, v_c, in the middle, where $c = \lfloor n/2 \rfloor$. The algorithm is as shown in Fig. 3. Instead of one token per node, it now handles multiple tokens per node. Each node is equipped with two queue variables, L_i and R_i, which are used to hold

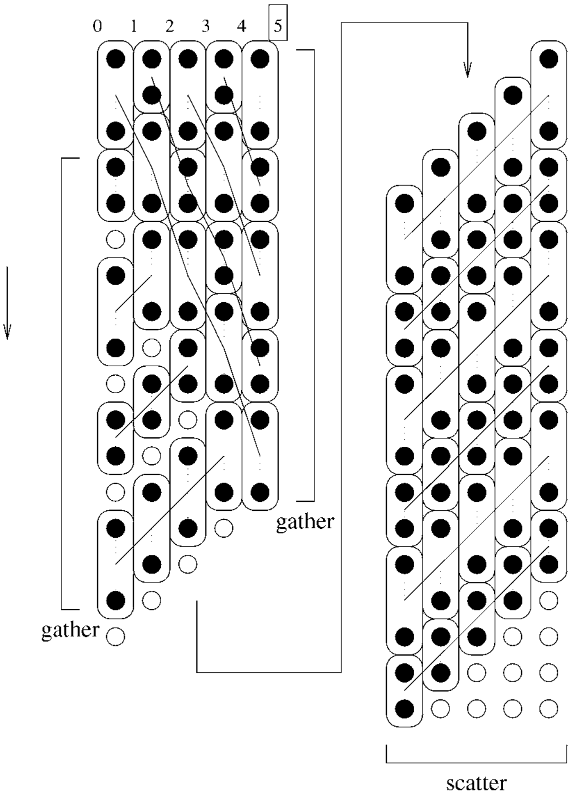

Fig. 4. Phase-2 execution for $n = 11$ (showing only the left half).

tokens to be sent to the left and right hand side, respectively. Initially, these two variables would each have the set of tokens generated from Phase 1.

We focus on the left half; what goes on in the right half is identical. The operation of the algorithm can be seen as being composed of two gather suboperations, followed by a scatter suboperation. In the first gather suboperation, tokens of nodes $v_0, v_1, \ldots, v_{c-1}$ all go to the center node (and all the intermediate nodes); in the second gather suboperation, tokens of nodes $v_{c-1}, \ldots, v_1$ go to node v_0 (and all the intermediate nodes); in the scatter suboperation, tokens belonging to nodes $v_c, v_{c+1}, \ldots, v_{n-1}$, which have been gathered at node v_c, are scattered to nodes $v_0, v_1, \ldots, v_{c-1}$. Fig. 4 shows the execution of Phase-2 algorithm for $n = 11$; only the left half is shown. Instead of wires of dots, we now have wires of ovals. The bigger oval contains $\lceil n/2 \rceil$ tokens that an even node has collected from Phase 1; the smaller oval contains $\lfloor n/2 \rfloor$ tokens from Phase 1.

It is interesting to see that, if we treat an oval as a dot, the pattern in Fig. 4 is exactly the same as the case where each node has exactly one token to start with, as shown in Fig. 5. In particular, the number of idle edge instances is the same.[1] In fact, the path algorithm, which is optimal for the one–token-per-node case, is also optimal for the case here. Specifically, the number of steps for the case here is equal to (refer to the edge $\langle v_0, v_1 \rangle$)

$$\lceil n/2 \rceil^2 + \lfloor n/2 \rfloor^2 + (\lfloor n/2 \rfloor - 1) = (n^2 + n - 2)/2 = S'_{odd},$$

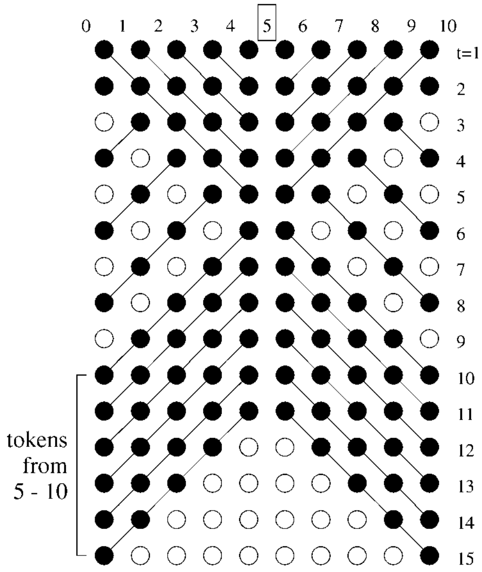

Fig. 5. The Phase-2 algorithm for a path of length n if every node has one token to distribute.

where the first term comes from the big ovals, the second term from the small ovals, and the third term from the white dots. S'_{odd} matches the lower bound which is given below.

Lemma 3. *Given odd integer n and a path of n nodes, the lower bound on the gossiping time is $(n^2 + n - 2)/2$, where each even node initially has $\lceil n/2 \rceil$ tokens and each odd node has $\lfloor n/2 \rfloor$ tokens.*

Proof. Consider the edge $\langle v_{c-1}, v_c \rangle$, where $c = \lfloor n/2 \rfloor$. To gossip, there are $((c-1)/2 + 1) \times \lceil n/2 \rceil + (c-1)/2 \times \lfloor n/2 \rfloor$ tokens, belonging to nodes $v_0, v_1, \ldots, v_{c-1}$, that have to go through the edge in question from left to right, and there are $((c-1)/2 + 1) \times (\lceil n/2 \rceil + \lfloor n/2 \rfloor)$ tokens that have to go through this edge from right to left. Therefore, to transport all these tokens through the edge requires at least $(c+1)n - c$ steps. The last token that goes through this edge requires at least $c - 1$ steps to reach the last node in the path. Hence, the minimum number of steps to complete the gossip is

$$(c+1)n - c + (c-1) = (n^2 + n - 2)/2.$$

□

For the other odd-n case, where an odd node in the row has one token more than an even node at the beginning of Phase 2, the number of steps would be equal to

$$2(\lceil n/2 \rceil \times \lfloor n/2 \rfloor) + \lfloor n/2 \rfloor - 1 = (n^2 + n - 4)/2 = S''_{odd}.$$

Using similar arguments as in Lemma 3, we can prove that the Phase-2 algorithm is optimal for this case as well and for the even-n case below. Obviously, $S'_{odd} > S''_{odd}$; hence, the number of steps for the execution of the Phase-2 algorithm over all the rows and columns is equal to $(n^2 + n - 2)/2 = S'_{odd}$.

[1] This number, as can be inferred from the example in Fig. 4 or Fig. 5, is equal to $4 \times \sum_{i=1}^{\lfloor n/2 \rfloor - 1} i = (n^2 - 4n + 3)/2$.

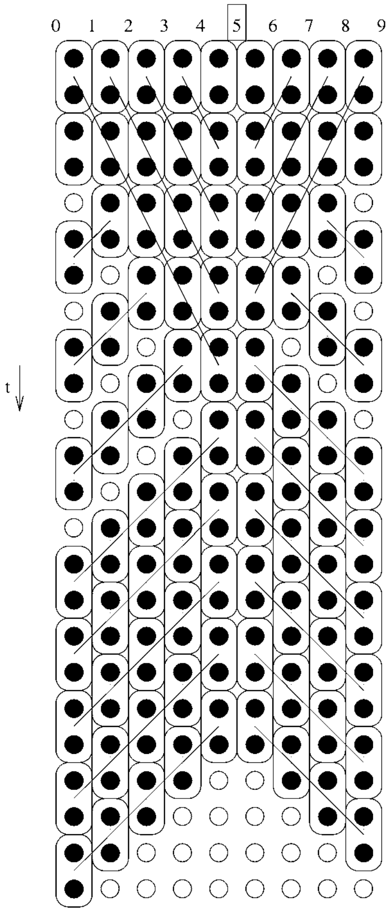

Fig. 6. Phase-2 execution for $n = 10$.

3.3 Phase 2—Even n

For this case, the ovals are all of the same size ($n/2$). An example of the execution of the Phase-2 algorithm is shown in Fig. 6, where the center node is v_c, $c = n/2$. The two halves are not exactly symmetrical. The number of steps for the execution of the algorithm is equal to

$$n \times n/2 + (n/2 - 1) = (n^2 + n - 2)/2 = S_{even}.$$

3.4 Combining Phase 1 and Phase 2

Instead of starting Phase 2 after Phase 1 is completely over, we let Phase 2 "cut into" Phase 1. As has been shown in Section 3.1, there are $\lfloor n/2 \rfloor - 1$ (respectively, $n/2 - 1$) white wires (a series of n idle edges) in the execution of the Phase-1 algorithm for the odd-n (respectively, even-n) case. The goal is to have the Phase-2 algorithm make use of all these white wires. The strategy is simple: to map the first step of the execution of the Phase-2 algorithm to the first white wire, the second step to the second white wire, etc. The following lemma guarantees that Phase 2 can make use of all the white wires in Phase 1.

Lemma 4. *By the time the kth white wire occurs, the tokens needed by the kth step of Phase 2 would all be ready,* $k = 1, 2, \ldots$

Proof. We refer to Fig. 2 and Fig. 7 without loss of generality. In Fig. 2, we see that the first white wires for the various cases occur at $t_1 \geq 2$ and the other white wires at every other time step—i.e., $t_1 + 2, t_1 + 4, \ldots$

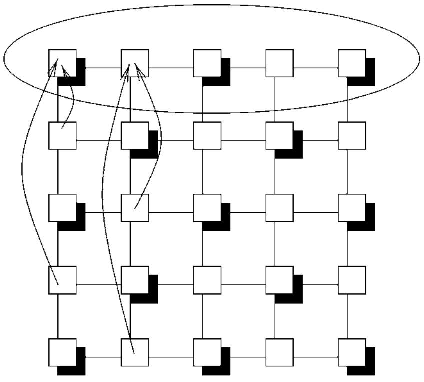

Fig. 7. Phase-1 tokens reaching their destined nodes at every other timestep.

Now consider Fig. 7, where we focus on the first row of the mesh. Since the Phase-1 algorithm is all nonblocking throughout, the tokens needed by Phase 2 will arrive at their destined nodes exactly one every two timesteps and the first one of these tokens arrives at $t_2 \leq 2$. Since $t_1 \geq t_2$ and there are less than $n/2$ white wires in Phase 1, all the tokens needed to fill any of the white wires would reach their destined nodes before that white wire occurs. □

By letting Phase 2 cut into Phase 1, the number of steps to execute Phase 2 is reduced by $\lfloor n/2 \rfloor - 1$ and $n/2 - 1$ for the odd-n case and the even-n case, respectively. We therefore have the following results for the combined algorithm.

Theorem 1. *The number of timesteps required to finish gossiping for the* $n \times n$ *mesh is* $n^2 + n - 1/2$ *for odd* n *and* $n^2 + n - 1$ *for even* n.

Proof. Refer to Fig. 2 without loss of generality. For the odd n case, Phase 1 will be dominated by the case in Fig. 2b, and, hence, the time to execute Phase 1 is $n - 1$ and, that, for the even case, is also $n - 1$ (Fig. 2c). Hence, for the odd-n case, we have the total time equal to $(n - 1) + S'_{odd} - (\lfloor n/2 \rfloor - 1) = n^2/2 + n - 1/2$ and, for the even-n case,

$$(n - 1) + S_{even} - (n/2 - 1) = n^2/2 + n - 1.$$

□

Our algorithm is better than the Fujita-Yamashita algorithm by n step, and is about $n/2$ steps away from the lower bound.

Finally, we need to point out the modifications that should be made to the Phase-2 algorithm so that the Phase-2 algorithm would not misuse some of the edges which should be used by the Phase-1 algorithm during the overlapped period. The modifications are as follows:

- The Phase-2 algorithm starts at the same time as the Phase-1 algorithm.
- The Phase-2 algorithm would give priority to Phase-1 tokens. It will send out a Phase-2 token only when there is no Phase-1 token on hand to be sent. A single bit is enough to mark a token as belonging to Phase 1 or Phase 2.

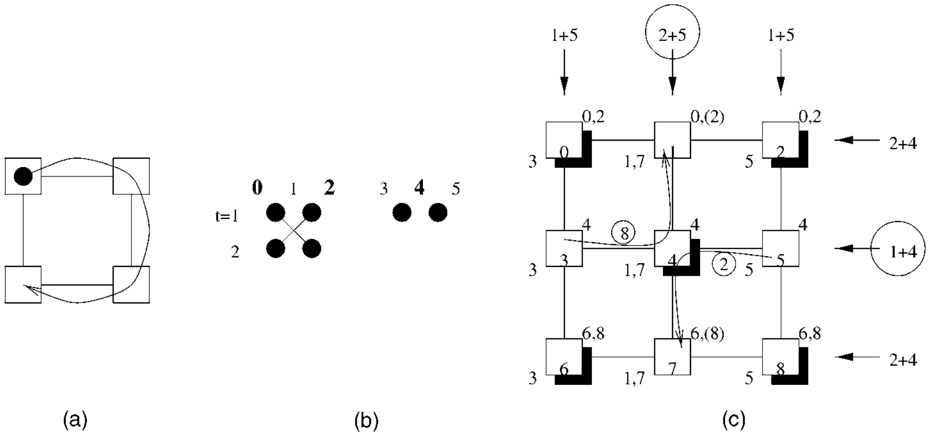

Fig. 8. (a) Movement of a token in a 2×2 mesh. (b) The two half-gossip patterns for Phase 1 of the 3×3 mesh. (c) Optimal gossiping for the 3×3 mesh.

3.5 Optimal Solutions for $n \leq 3$

There exists a gap of $n/2$ between the lower and the upper bound. The trivial proof for the lower bound would prompt us to doubt the tightness of the lower bound. This section shows that, for $n \leq 3$, the lower bound is indeed tight. For $n = 2$, the mesh is a cycle with four nodes, as shown in Fig. 8a. All the nodes need to do is simply send their tokens along the same agreed direction; the gossip would finish in three steps, which is optimal. For $n = 3$, the situation is trickier; a modification is needed for our mesh algorithm.

In the following, we assume that initially each node holds a token which has the same identity (an integer) as the node; $\langle a \rangle$ denotes a token and $\langle a, b, \ldots \rangle$ a collection of tokens (from Phase 1) held in a node, where a, b, etc. are token identities; v_i denotes the node with identity i.

For $n = 3$, our mesh algorithm finishes in 7 steps, which is one step more than the lower bound. We now show how we could reduce the seven timesteps to 6 by modifying the algorithm slightly. Fig. 8c shows a 3×3 mesh after Phase 1. During Phase 1, two different half-gossip patterns would occur, as shown in Fig. 8b, one taking one timestep and the other taking two timesteps. Then, for Phase 2, for the rows, the gossip will take four timesteps because there are four tokens ($\langle 3 \rangle - \langle 1, 7 \rangle - \langle 5 \rangle$) from Phase 1 that need to be broadcast, and five timesteps for the columns ($\langle 0, 2 \rangle - \langle 4 \rangle - \langle 6, 8 \rangle$). Combining the two phases, we have the total timesteps for each row and column as indicated in the Fig. 8c. All except one row and one column would spend six timesteps: the middle row needs only five timesteps, while the middle column needs seven. Our modification to the algorithm is to cut short the Phase 2 execution for the middle column by one step, but use the unused timestep in the middle row to make up for it, as follows:

For the middle column, for Phase 2, instead of $\langle 0, 2 \rangle - \langle 4 \rangle - \langle 6, 8 \rangle$, we do the gossip for just $\langle 0 \rangle - \langle 4 \rangle - \langle 6 \rangle$. This takes three timesteps instead of the original five, but the result would be that v_1 will not receive $\langle 8 \rangle$, v_4 will not receive $\langle 2 \rangle$ and $\langle 8 \rangle$, and v_7 will not receive $\langle 2 \rangle$. Now, we have both the middle row as well as the middle column that have one unused timestep (they take $1 + 4$ and $2 + 3$ timesteps, respectively). We can make use of these unused timesteps to deliver $\langle 2 \rangle$ and $\langle 8 \rangle$ to v_1, v_4, and v_7, as shown in the figure. $\langle 8 \rangle$ is available in v_3 as early as $t = 3$ (the second step of Phase 2) and, so, v_3 can send $\langle 8 \rangle$ to v_4 any time before $t = 6$, using the unused timestep. The situation for $\langle 2 \rangle$ is similar. Node v_4 can wait until as late as $t = 6$ to send $\langle 2 \rangle$ and $\langle 8 \rangle$ to v_7 and v_1, respectively, using the unused timestep.

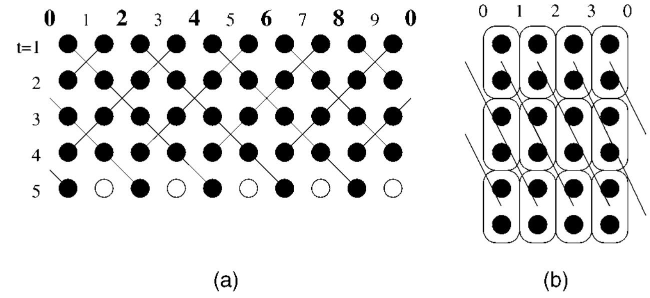

Fig. 9. Phase 1 and 2 for even-n torus. (a) Phase 1; $n = 10$. (b) Phase 2; $n = 4$.

With the above modification, the gossip for the 3×3 mesh can finish in six timesteps, matching the lower bound.

4 The Gossip Algorithm for a Square Torus

With additional wrap-around edges, an $n \times n$ torus is expected to have better performance in terms of gossiping time than its nontoroidal counterpart of the same size. In the following, we show that, using a two-phase strategy similar to the one we used for the mesh, we can achieve optimality for the even-n case and very nearly so for the odd-n case.

4.1 Even n

As in the mesh algorithm, we label the nodes as even or odd. Consider a row, where there are $n/2$ even and $n/2$ odd nodes. An even node of the row needs to broadcast its token to the other $n - 1$ nodes. Since $n - 1$ is an odd number, we impose on the token so that it would go to $n/2$ nodes on the right and $n/2 - 1$ nodes on the left.[2] Note that v_{n-1} is v_0's

[2] The left-right orientation is with respect to the "Linear view" of a row, as shown in Fig. 9a.

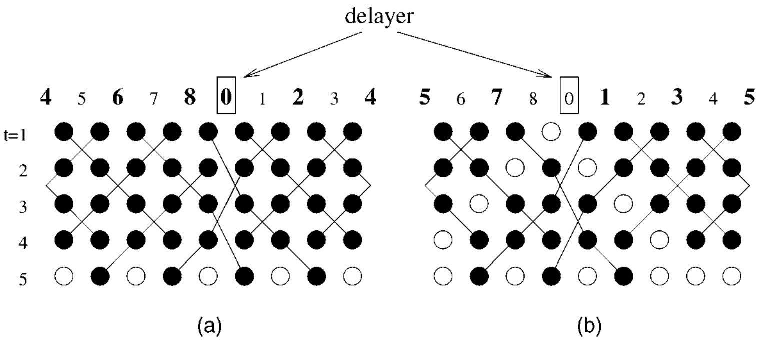

Fig. 10. Phase-1 execution for $n = 9$.

left-hand neighbor and v_0 is v_{n-1}'s right-hand neighbor. To implement this rule, every token will need to carry a counter, which is set to either $n/2$ or $n/2 - 1$ initially, depending on its orientation; after every move, the counter is decremented and a token with a zero counter value will not be propagated any further. Alternatively, if tokens carry the identity of their source, the distance that a token has traveled can be easily derived, which can be used to terminate the propagation of a token.

Fig. 9a gives the dot-wire diagram of the Phase-1 algorithm for $n = 10$. The number of idle edge instances is equal to $n/2$ because the wires that stretch towards the left are one step shorter than those that stretch toward the right. This number of idle instances, however, is not enough to form a white wire for Phase 2 to take advantage of. Nonetheless, as we will soon see, if Phase 2 can fill up all its edge instances (or leave no more than $n/2 - 1$ of them idle), the combined algorithm would still be optimal. In fact, Phase 2 can, using the following simple algorithm.

- A counter is attached to every token generated from Phase 1 and initialized to the value $n - 1$.
- Every node maintains a single queue of tokens, which contains the $n/2$ tokens from Phase 1 initially.
- Each node extracts and sends the tokens in its queue to the right-hand neighbor one by one until the queue becomes empty.
- When a token comes to a node from the node's left-hand neighbor, the counter of the token is decremented; the token will not propagate any further if the counter value reaches zero; otherwise, the token joins the queue.

Fig. 9b shows an example of the execution of this algorithm for one row, where $n = 4$. After Phase 1, each node in the row has $n/2 = 2$ tokens to distribute. It is easy to see that this is the best one can do for Phase 2.

Theorem 2. *Given a torus R of size n × n, where n is even. The combined algorithm—Phase 1 followed by Phase 2 with no overlapping—can finish the gossip in time $g(R) = n^2/2$, which is optimal.*

Proof. Referring to Fig. 9a without loss of generality, Phase 1 takes $n/2$ steps and Phase 2 takes $n/2 \times (n - 1) = n^2/2 - n/2$ steps. Hence, $g(T) = n^2/2$, which is optimal according to Lemma 3. □

4.2 Odd n

Consider a row. With an odd number of nodes, the Phase-1 algorithm, which we have used before, cannot be applied directly. This is clear by looking at Fig. 10a where two even nodes, v_0 and v_8, become each other's neighbor. Note that we have rotated the row for easier viewing. If we use the previous Phase-1 algorithm as it is, v_0's packet and v_8's packet would collide at $t = 1$. To solve the problem, we pretend that there is an odd node between v_8 and v_0 and, hence, every token that goes through the edge $\langle v_8, v_0 \rangle$ will take two steps instead of one. This can be achieved by designating one of v_0 and v_8 to be a "delayer" node. The delayer node would execute the algorithm with the following modification.

- For every token (either the node's own token or one just received from a neighbor) to be forwarded to the

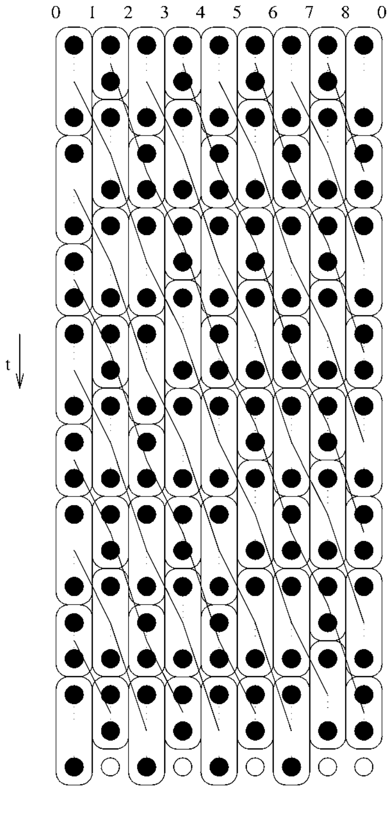

Fig. 11. Phase-2 execution for $n = 9$.

TABLE 1
Summary of Results

		Lower bound	Upper bound (this paper)	Upper bound ([11])
Mesh	odd n	$n^2/2+n/2$	$n^2/2+n-1/2$	$n^2/2+2n-5/2$
	even n	$n^2/2+n/2$	$n^2/2+n-1$	$n^2/2+2n-3$
	$n=2,3$	$3,6$	$3,6$	N.A.
Torus	odd n	$n^2/2-1/2$	$n^2/2+3/2$	N.A.
	even n	$n^2/2$	$n^2/2$	N.A.

next node, delay the token by one timestep before forwarding it.

All other nodes execute the Phase-1 algorithm as it is, without the modification. A row has either $\lceil n/2 \rceil$ (Fig. 10a) or $\lfloor n/2 \rfloor$ (Fig. 10b) even nodes. For the latter case, we assign one of the two consecutive *odd* nodes to be the delayer. As shown in Fig. 10, both cases take the same number of timesteps—$\lfloor n/2 \rfloor + 1$.

For Phase 2, the same algorithm as which was used in the even case can be used. For odd n, however, the load is not even across a row or column at the start of Phase 2. We consider the dominating case, where there are $\lceil n/2 \rceil$ nodes having one token more than the other nodes. An example of the execution of the Phase-2 algorithm is given in Fig. 11 for $n = 9$. The even nodes have $\lceil n/2 \rceil$ tokens and the odd nodes have $\lfloor n/2 \rfloor$ tokens initially. Referring to the figure, the height of the dot-wire diagram is equal to

$$((n-1)/2+1) \times \lceil n/2 \rceil + ((n-1)/2-1) \times \lfloor n/2 \rfloor \\ = n^2/2 - n/2 + 1.$$

By adding this to Phase 1′s time, we have the following:

Theorem 3. *Given a torus, T, of size $n \times n$, where n is even. Using the combined algorithm (Phase 1 followed by Phase 2) as described above, $g(T) = (n^2+1)/2 + 1$.*

5 Conclusion

Table 1 summarizes the results derived in this paper. Based on the results, we consider the problem for the case of the 2D square torus to be settled. For the mesh, the performance of our algorithm is better than that of the Fujita-Yamashita algorithm by n steps. There exists a gap of $n/2$ between the lower and the upper bounds derived. The case of $n = 3$ seems to hint that any attempt to close the gap should be targeted at the upper bound. It should also be interesting to try to solve the problem for any value of p, as well as for higher-dimensional meshes/tori.

The two-phase strategy works best for square meshes/tori because of the division of work between rows and columns and their parallel operations. For $m \times n$ meshes/tori, where $m \neq n$, the strategy will end up with nonoptimal results as the time will be dominated by operations along the longer dimension and there would be much idleness in the the other dimension. For these structures, we should look for a different strategy.

Acknowledgments

This work is supported by the Hong Kong RGC Grant HKU7028/98E. The authors wish to thank the referees for their constructive comments.

References

[1] J.-C. Bermond, L. Gargano, and S. Perennes, "Optimal Sequential Gossiping by Short Messages," *Discrete Applied Mathematics,* vol. 86, pp. 145-155, 1998.

[2] J.-C. Bermond, L. Gargano, A.A. Rescigno, and U. Vaccaro, "Fast Gossiping by Short Messages," *SIAM J. Computing,* vol. 27, pp. 917-941, 1998.

[3] J.-C. Bermond, L. Gargano, A.A. Rescigno, and U. Vaccaro, "Fast Gossiping by Short Messages," *Proc. Int'l Colloquium Automata, Languages, and Processing '95,* pp. 135-146, 1995.

[4] A. Bagchi, E.F. Schmeichel, and S.L. Hakimi, "Sequential Information Dissemination by Packets," *Networks,* vol. 22, pp. 317-333, 1992.

[5] A. Bagchi, E.F. Schmeichel, and S.L. Hakimi, "Parallel Information Dissemination by Packets," *SIAM J. Computing,* vol. 23, pp. 355-372, 1994.

[6] A. Czumaj, L. Gasieniec, and A. Pelc, "Time and Cost Trade-Uffs in Gossiping," *SIAM J. Discrete Math.,* vol. 11, pp. 400-413, 1998.

[7] W.J. Dally, "The J-Machine: System Support for Actors," *Actors: Knowledge-Based Concurrent Computing,* Hewitt and Agha, eds., 1989.

[8] W.J. Dally and P. Song, "Design of Self-Timed VLSI Multicomputer Communication Controller," *Proc. Int'l Conf. Computer Design,* pp. 230-234, 1987.

[9] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach.* Los Alamitos, Calif.: IEEE CS Press, 1997.

[10] P. Fraigniaud and E. Lazard, "Methods and Problems of Communication in Usual Networks," *Discrete Applied Math.,* vol. 53, pp. 79-134, 1994.

[11] S. Fujita and M. Yamashita, "Fast Gossiping on Square Mesh Computers," *Information Processing Letters,* vol. 48, pp. 127-130, 1993.

[12] S.M. Hedetniemi, S.T. Hedetniemi, and A. Liestman, "A Survey of Gossiping and Broadcasting in Communication Networks," *Networks,* vol. 18, pp. 319-349, 1988.

[13] J. Hromkovic, R. Klasing, B. Monien, and R. Peine, "Dissemination of Information in Interconnection Networks (Broadcasting & Gossiping)," *Combinatorial Network Theory,* pp. 125-212, D.-Z. Du and D.F. Hsu, eds., 1996.

[14] B.H.H. Juurlink, J.F. Sibeyn, and P.S. Rao, "Gossiping on Meshes and Tori," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 6, pp. 513-525, June 1998.

[15] D.W. Krumme, "Fast Gossiping for the Hypercube," *SIAM J. Computing,* vol. 21, no. 2, pp. 365-380, Apr. 1992.

[16] 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.

[17] M. Kaufmann and J.F. Sibeyn, "Randomized Multipacket Routing and Sorting on Meshes," *Algorithmica,* vol. 17, pp. 224-244, 1997.

[18] F.C.M. Lau and S.H. Zhang, "Optimal Gossiping in Paths and Cycles," Technical Report TR-97-10, Dept. of Computer Science, Univ. of Hong Kong, July 1997.

[19] S.L. Lillevik, "The Touchstone 30-Gigaflop DELTA Prototype," *Proc. Sixth Distributed Memory Computing Conf.,* pp. 671-677, 1991.

[20] *Networks, Routers and Transputers,* M.D. May, P.W. Thompson, and P.H. Welch, eds.Amsterdam: IOS Press, 1993.

[21] P.S. Rao and G. Mouney, "Data Communications in Parallel Block Predictor-Corrector Methods for Solving ODEs," technical report, LAAS-CNRS, France, 1995.

[22] C.L. Seitz, "The Architecture and Programming of the Ametek Series 2010 Multicomputer," *Proc. Third Conf. Hypercube Concurrent Computers and Applications*, pp. 33-36, Jan. 1988.

[23] Y.-J. Suh and K.G. Shin, "Efficient All-to-All Personalized Exchange in Multidimensional Torus Networks," *Proc. Int'l Conf. Parallel Processing '98,* pp. 468-475, Aug. 1998.

[24] M. Soch and P. Tvrdík, "Optimal Gossip in Noncombining 2D-Meshes," *Proc. Fourth Int'l Colloquium Structural Information Comm. Complexity (SIROCCO '97),* 1997.

[25] M. Soch and P. Tvrdík, "Time-Optimal Gossip of Large Packets in Noncombining 2D Tori and Meshes," *IEEE Trans. Parallel and Distributed Systems,* vol. 10, no. 12, pp. 1252-1261, Dec. 1999.

[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] Y.-j Tsai and P.K. McKinley, "An Extended Dominating Node Approach to Broadcast and Global Combine in Multiport Wormhole-Routed Mesh Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 8, no. 1, pp. 41-58, Jan. 1997.

[28] 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.

[29] P. Tvrdík and M. Soch, "Optimal Gossip in Store-and-Forward Noncombining 2D Tori," *Proc. EUROPAR '97,* 1997.

[30] Y. Yang and J. Wang, "Efficient All-to-All Broadcast in All-Port Mesh and Torus Networks," *Proc. Int'l Symp. High Performance Computer Architecture (HPCA-5),* pp. 290-299, Jan. 1999.

Francis C.M. Lau (S '78-M '87) received the BSc degree from Acadia University, Canada, and the MMath and the PhD degrees from the University of Waterloo. He joined the Department of Computer Science and Information Systems at The University of Hong Kong in 1987, where he is now the head of the department. His research interests are in parallel and distributed computing, object-oriented programming, operating systems, and Web and Internet computing. He is a member of the IEEE.

Shi-heng Zhang received the BSc degree from Hefei Polytechnic University, China, in 1985, and the MSc degree from the University of Science and Technology of China in 1988. He pursued PhD study in computer science at The University of Hong Kong during 1995-1998. His research interest is in parallel and distributed computing.