Fast Gossiping in Meshes/Tori with Bounded-Size Packets

F.C.M. Lau* and S.H. Zhang
Department of Computer Science
The University of Hong Kong
{fcmlau,shzhang}@cs.hku.hk

August 1997

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, complete exchange, 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 complete exchange and all-to-all commu-

*Correspondence: F.C.M. Lau, Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong / Email: fcmlau@cs.hku.hk / Fax: (+852) 2559 8447.

nication, 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; it is also useful in many real applications, such as matrix transposition, fast Fourier transform algorithms, global processor synchronization, and load balancing. The problem has been studied extensively during the last two decades; a summary of the results can be found in [6, 4, 7].

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 [8]. 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.

Bagchi *et al.* [2], as well as Bermond *et al.* [1], have added another dimension to the problem. They suggested that a packet (for carrying tokens from one node to the next) cannot be of infinite size which a great majority of previous work had assumed [1]. In reality, indeed, a packet's delay is somewhat dependent on its contents, especially in tightly coupled multiprocessors. In this paper, we refer to this as the *bounded packet size restriction.* We 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 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 (*i.e.*, the "time" measure), and the communication load in terms of packet moves the gossip generates. 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 in this paper. We define $g_p(N)$ to be the time required to complete a gossip under some given p value for the interconnection network N. Since only one p value ($p = 1$) 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 among the most popular topologies for processors interconnection. For simplicity, we consider square meshes and tori. Our algorithms can be easily extended to cover the non-square ones as well, but the results would be less satisfactory (see discussions 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 $p = 1$, Šoch 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 [10]; 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) [11].

- 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 [1].

- 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 [5].

- For the H1 model and any p value, Bagchi *et al.* derived the result, $2mn/p + O(m + n)$, for the $m \times n$ mesh [3].

In this paper, we assume the H* model and $p = 1$. We present an improved algorithm for the square mesh, and an algorithm for the square torus; the latter is optimal for the case of even n, and two steps away from the optimal for the case of odd n.

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, we use $v_0, v_1, \ldots, v_{n-1}$ to represent the n nodes in any row or column. 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.

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

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 $n^2 \times (n^2 - 1)$. G has $2n \times (n-1)$ edges. Therefore, there exists an edge which has to accommodate $n^2 \times (n^2 - 1)/(2n \times (n-1)) = n^2/2 + n/2$ moves. □

Theorem 2 *For a torus of size $n \times n$, R, the lower bound on $g(R)$ is $(n^2 + 1)/2 - 1$ for odd n and $n^2/2$ 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 $(n^2 + 1)/2 - 1$ for odd n and $n^2/2$ 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 lower bounds come from filling up all the instances of all the edges over a period of time. To achieve a matching upper bound, we need an algorithm that would do the same: to try to use 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 in Fig. 1.

We give only the outline for this trivial algorithm; the detailed algorithm can be easily inferred from the examples in Fig. 2.

▷ 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.

Figure 1: The Phase-1 (half-gossip) algorithm

We can call the Phase-1 algorithm a "half-gossip" algorithm because only half or about half of the nodes in a row or column broadcast their tokens. And we call the tokens broadcast by the even nodes along their respective rows *even tokens* and those by odd nodes along their respective columns *odd tokens*. After Phase 1, each node in a row will have collected $n/2$ (or ± 1 in case of 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 along the row/column itself using a gossip algorithm for paths.

Although it turns out to be asymptotically optimal, the Fujita-Yamashita algorithm suffers from two major problems: first, their Phase 1 would result in a number of idle edge instances, and second, their Phase-2 algorithm (for paths) is not optimal.

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 as much as possible the two phases so that the idle instances of edges in Phase 1 can be made use of 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.

For better viewing, we model the packet moves over the edges of a row or column as a two-dimensional array of "dots". A dot corresponds to an edge, and one row of dots corresponds to the entire row or column of edges at a particular timestep. This two-dimensional array in fact is the entire execution profile of an algorithm applied to the row or column in question. An example is shown in Fig. 2, where a black dot corresponds to an edge in use (*i.e.*, a packet is being transmitted through the edge), and a white dot to an idle instance of an edge. The dissemination of a token from one node to the next and so on is represented by a "wire" of black dots. Note that a wire can only go downward in the execution profile, and no two wires may cross at a black dot; if they cross, that means there is a clash on the use of the same edge.

3 The Mesh

The algorithm we propose here for the mesh uses the first phase of the Fujita-Yamashita algorithm as its first phase (*i.e.*, Fig. 1). For the second phase, the path gossiping algorithm introduced in [9] is used. By proving a lower bound on the gossiping time, we show that this path algorithm is optimal for the second phase here where a node has more than one token to distribute at the start of 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. It turns out that our algorithm is faster than the Fujita-Yamashita algorithm by approximately n steps.

3.1 Phase 1

Let's examine the half-gossip algorithm (Fig. 1) which is Phase 1 of the combined algorithm. Our goal is to identify instances of idle edges (corresponding to white dots in the dot-wire diagram) arising during the execution of this algorithm. The Phase-2 algorithm will then 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 the pattern of communication is regular (*i.e.*, all the wires are straight lines). Let's 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, indicated by the dashed lines. These white wires are "concave" when viewed from the top ($t = 0$) and their "bottom" touches the center node, v_c, of the path, where $c = \lfloor n/2 \rfloor$ for the odd-n case, and $n/2$ for the even-n case. For the odd-n case, the first white wire occurs at either $t = 2$ (Fig. 2(a)) or $t = 3$ (Fig. 2(b)), and that for the even-n case occurs at $t = 2$ (Fig. 2(c)). 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.

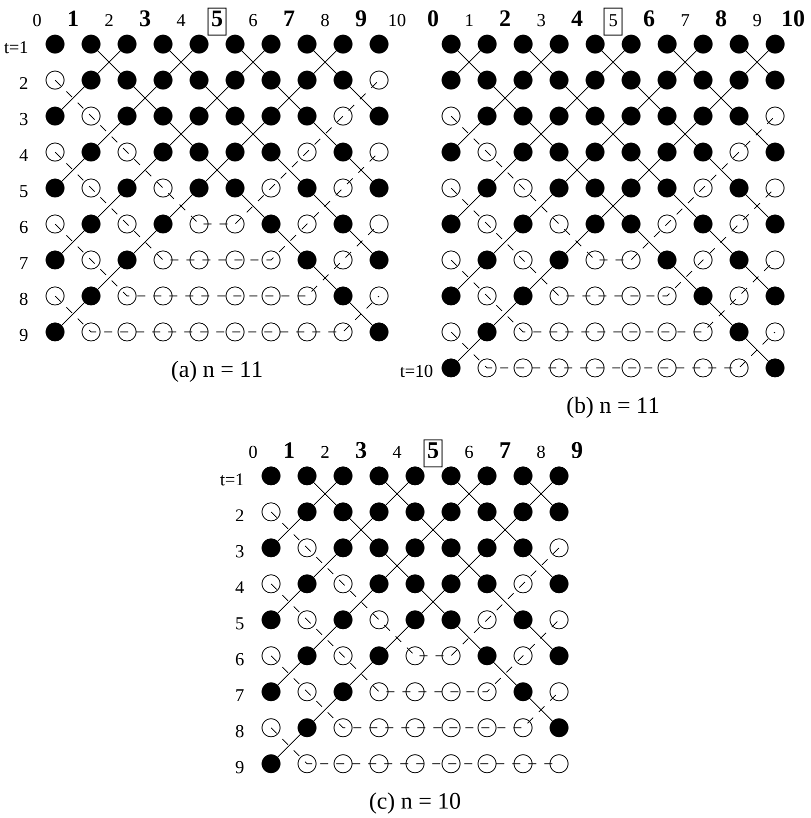

Figure 2: Examples of Phase-1 execution

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 is that this 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)—that is, an even node has $\lceil n/2 \rceil$ tokens, and an odd node has $\lfloor n/2 \rfloor$ tokens.

$\triangleright$ $R_i = L_i = \{Token_i\}$ for $0 < i < n-1$; $R_0 = \{Token_0\}$; $L_0 = \{Token_{n-1}\}$;
    (* the following are in parallel *)
$\triangleright$ each node v_i on the left of the center node repeat
        if $R_i \neq \emptyset$
            select an arbitrary token from R_i and send it to v_{i+1}
            (the latter then puts it in R_{i+1});
        else
            obtain an arbitrary token from v_{i+1} (L_{i+1})
            and store it in L_i (if $i > 0$) ;
    until done;
$\triangleright$ each node v_i on the right of the center node repeat
        (* similarly, except all the directions are reversed *)
    until done;

Figure 3: The Phase-2 algorithm (for path)

We use the path algorithm introduced in [9] 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. Let's focus on the left half; what goes on in the right half is symmetric. 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; in the second gather suboperation, tokens of nodes $v_{c-1}, \ldots, v_1$ go to the node v_0; 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 the nodes $v_0, v_1, \ldots, v_{c-1}$. Fig. 4 shows the execution of this 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 corresponds to the $\lceil n/2 \rceil$ tokens that are in an even node of the row at $t = 0$; and the small oval the $\lfloor n/2 \rfloor$ tokens in an odd node.

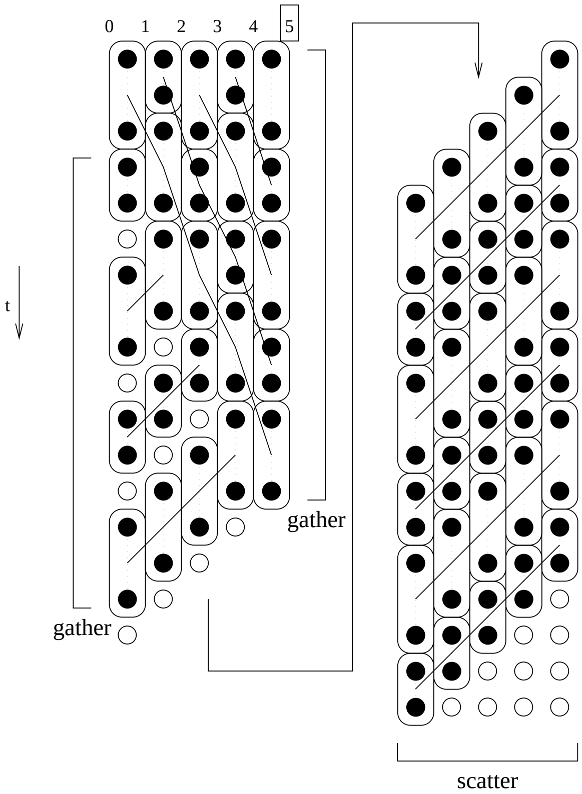

Figure 4: Phase-2 execution for $n = 11$ (showing only the left half)

It is interesting to see that, if we treat an oval a dot, the pattern as exhibited in Fig. 4 is exactly the same as that of the case where each node has exactly one token to start with (if we focus on the wires and the idle edge instances), as shown in Fig. 5. In particular, the number of idle edge instances is the same. This number, as can be inferred from the example in Fig. 4 or Fig. 5, is equal to

$$4 \times \sum_{i=1}^{\lfloor n/2 \rfloor - 1} i = (n^2 - 4n + 3)/2 = I_o.$$

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 $\langle v_0, v_1 \rangle$)

$$\lceil n/2 \rceil^2 + \lfloor n/2 \rfloor^2 + \lfloor n/2 \rfloor - 1 = (n^2 + n - 2)/2 = T_{o'}$$

which matches the lower bound, given below.

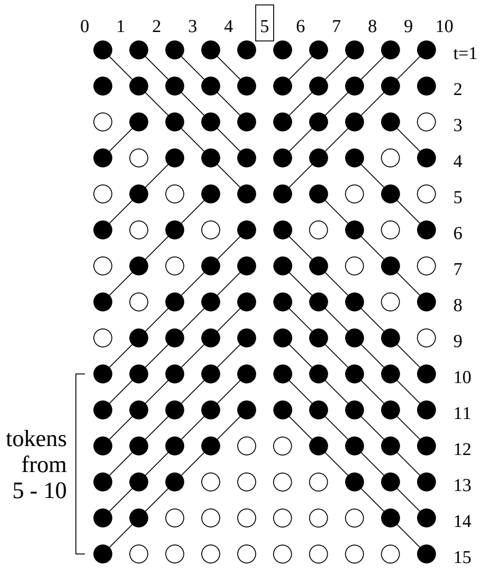

Figure 5: The case of each node having one token to distribute

Theorem 3 *For a path of n nodes, the lower bound on the gossiping time is $(n^2 + n - 2)/2$ for odd n and 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 $(c+1)n - c + (c-1) = (n^2 + n - 2)/2$. $\square$

Using similar arguments, we can easily prove that the algorithm is optimal also for the following case as well as the even-n case.

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

$$(\lceil n/2 \rceil \times \lfloor n/2 \rfloor)^2 + \lfloor n/2 \rfloor - 1 = (n^2 + 2n - 7)/4 = T_{o''}.$$

Obviously, $T_{o'} > T_{o''}$; hence, the number of steps for the execution of the Phase-2 algorithm over all the rows and columns is $T_{o'}$.

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 symmetric, but close. The number of idle instances of edges is equal to

$$3 \times \sum_{i=1}^{n/2-1} i + \sum_{i=1}^{n/2-2} i = (n^2 - 3n + 2)/2 = I_e.$$

And the number of steps for the execution of the algorithm is equal to

$$n \times n/2 + (n/2 - 1) = (n^2 + n - 2)/2 = T_e.$$

3.4 Combining Phase 1 and Phase 2

Instead of starting Phase 2 only after Phase 1 is completely over, we let Phase 2 "cuts into" Phase 1. As shown in Section 3.1, there are $\lfloor n/2 \rfloor - 1$ (resp. $n/2 - 1$) white wires (a series of n idle edges) in the execution of the Phase-1 algorithm for the odd-n (resp. even-n) case. The goal here is to have the Phase-2 algorithm make use of all these white wires, thus cutting down on the number of steps for the execution of the Phase-2 algorithm. The strategy is simply: 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, and so on. This is possible because of the following facts (let's assume that we are looking at a row of the mesh).

- The first white wire occurs at $t = 2$ or 3; by this time, all the nodes in the row should have received their first column-token from their neighbors along their respective columns.

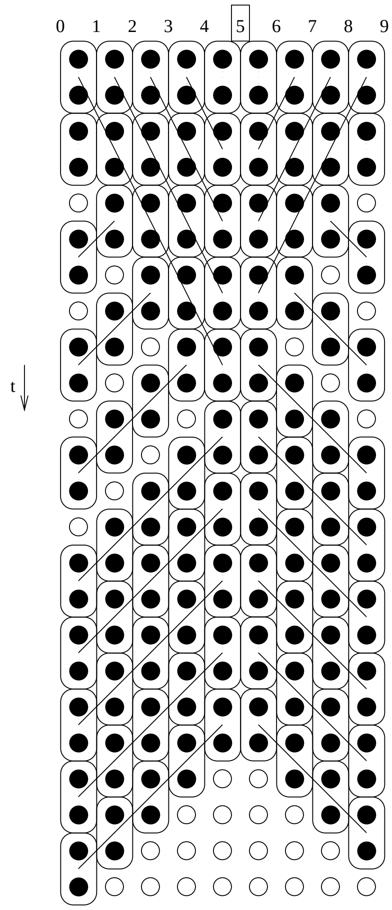

Figure 6: Phase-2 execution for $n = 10$

- The remaining white wires occur after every other timestep while the nodes in the row receive their column-tokens at every timestep.

- The white wires are concave and touch the center node of the row, while the flow of the column-tokens in the execution profile of the Phase-2 algorithm (for example, refer to Fig. 4) is from top to bottom and goes through the center node, and the latter has a steeper slope than either side of the former.

- There are $\lfloor n/2 \rfloor$ (or $n/2$) white wires, and the first $\lfloor n/2 \rfloor$ (or $n/2$) steps of the execution profile of the Phase-2 algorithm are all full (*i.e.*, there are no idle instances of edges).

Therefore, by letting Phase 2 to 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 4 *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. For the odd n case, Phase 1 will be dominated by the case in Fig. 2(b), and hence the time to execute Phase 1 is $n - 1$; and that for the even case is also $n - 1$ (Fig. 2(c)). Hence, for the odd-n case, we have the total time equal to $(n-1) + T_{o'} - (\lfloor n/2 \rfloor - 1) = n^2/2 + n - 1/2$, and for the even-$n$ case, $(n-1) + T_e - (n/2 - 1) = n^2 + n - 1$. □

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.

▷ For a row, the Phase-2 algorithm gives priority to row-tokens (*i.e.*, those belonging to Phase 1) over the use of any edge of the row: it will send out a column-token only when there is no row-token on hand to be sent; similarly, for a column, the Phase-2 algorithm gives priority to column-tokens.

4 The Torus

With additional wrap-around edges, an $n \times n$ torus is expected to have a better performance in terms gossiping time than its non-toroidal 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 a rule on the token so that it would go to $n/2$ nodes on the right and $n/2-1$ nodes on the left.[1] Note that v_{n-1} is v_0's 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.

Fig. 7(a) gives the execution profile 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 towards the right. This number of idle instances, however, is not enough to form a white wire for Phase 2 to take advantage of. If Phase 2 can fill up all its edge instances (or leave no more than $n/2-1$ of them idle), the combined algorithm should still be optimal. Indeed Phase 2 can, using the following simple algorithm.

- ▷ Every token is attached a counter with the value $n-1$.
- ▷ Every node keeps receiving tokens from its left-hand neighbor and keeps sending tokens to its right-hand neighbor until there are no more tokens; every time a token is sent to the next node, its counter is decremented; a token with a zero counter value is destroyed.

[1] The left-right orientation is with respect to the "linear view" of a row, as shown in Fig. 7(a).

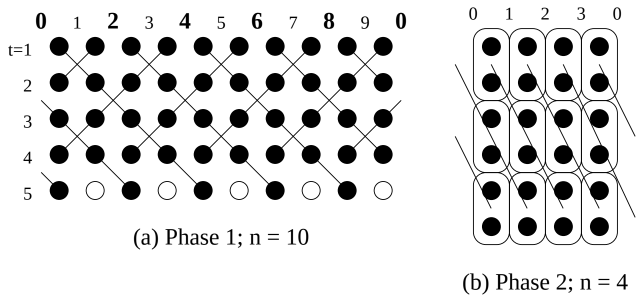

Figure 7: Phase 1 and 2 for even-n torus

Fig. 7(b) shows an example of the execution of this algorithm for one row, where $n = 4$. After Phase 1, each node in the row has $n/2 = 2$ column-tokens to distribute. It is easy to see that this simple algorithm is optimal as far as gossiping in a row (a cycle, more exactly) is concerned.

Theorem 5 *Given a torus, R, of size $n \times n$, where n is even. Using the combined algorithm (Phase 1 + Phase 2, as described above), $g(R) = n^2/2$ which is optimal.*

Proof: Referring to Fig. 7(a) without loss of generality, Phase 1 takes $n/2$ steps. Referring to Fig. 7, 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 Theorem 3. □

4.2 Odd n

Consider a row. With an odd number of nodes, the Phase-1 algorithm as we have used before cannot be applied directly. The reason is clear by looking at Fig. 8(a) where there are two consecutive even nodes, v_0 and v_8. 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 initially 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 next node, delay the token by one timestep

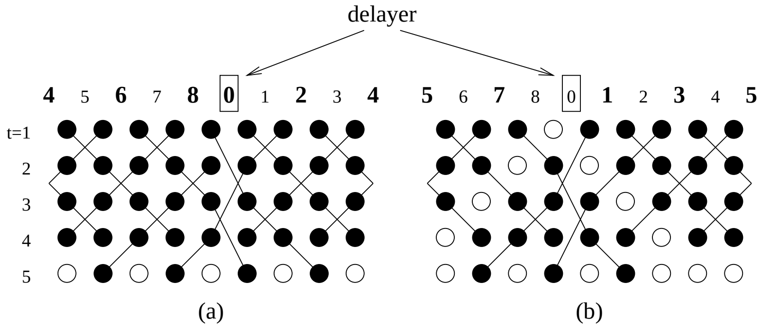

Figure 8: Phase-1 execution for $n = 9$

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. 8(a)) or $\lfloor n/2 \rfloor$ (Fig. 8(b)) even nodes. For the latter case, we designate one of the two consecutive *odd* nodes to be the delayer. As shown in Fig. 8, both cases take the same number of timesteps—$\lfloor n/2 \rfloor + 1$.

For Phase 2, the same algorithm as has been 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. 9 for $n = 9$. The even-numbered nodes have $\lceil n/2 \rceil$ tokens, and the odd-numbered nodes have $\lfloor n/2 \rfloor$ tokens initially. Referring to the figure, the height of the execution profile is equal to

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

Add this to Phase 1's time we have the following.

Theorem 6 *Given a torus, R, of size $n \times n$, where n is even. Using the combined algorithm (Phase 1 + Phase 2, as described above), $g(R) = (n^2+1)/2 + 1$.*

5 Conclusion

Table 5 summarizes the results derived in this paper. Based on the results, we consider the problem for the case of the tori to be settled. For the mesh, the

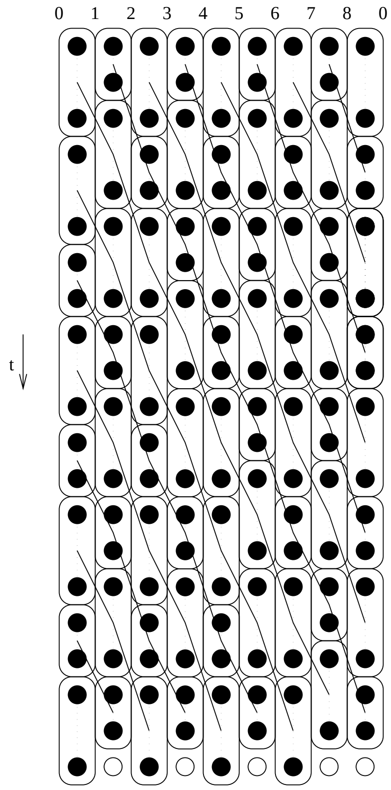

Figure 9: Phase-2 execution for $n = 9$

performance of our algorithm is better than that of the Fujita-Yamashita algorithm by n steps; further improvements might still be possible since there is a gap of $n/2$ between the lower and 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.

		Lower Bound	**Upper Bound**
Mesh	odd n	$n^2/2 + n/2$	$n^2 + n - 1/2$
	even n	$n^2/2 + n/2$	$n^2 + n - 1$
Torus	odd n	$(n^2 + 1)/2 - 1$	$(n^2 + 1)/2 + 1$
	even n	$n^2/2$	$n^2/2$

Table 1: Summary of results

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 lead to non-optimal results as the time will be dominated by operations along the longer dimension and there could be much idleness in the the other dimension. For these structures, we should look for a different strategy.

References

[1] J.-C. Bermond, L. Gargano, A.A. Rescigno, and U. Vaccaro, "Fast Gossiping by Short Messages," *Proc. ICALP'95*, pp. 135–146, Szeged, Hungary, 1995.

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

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

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

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

[6] 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.

[7] J. Hromkovič, R. Klasing, B. Monien, and R. Peine, "Dissemination of Information in Interconnection Networks (Broadcasting & Gossiping)," in *Combinatorial Network Theory*, pp. 125–212, D.-Z. Du and D.F. Hsu (Eds.), Kluwer Academic Publishers, Netherlands, 1996.

[8] D.W. Krumme, G. Cybenko, and K.N. Venkataraman, "Gossiping in Minimal Time," *SIAM J. on Computing*, Vol. 21, No. 1, pp. 111–139, February 1992.

[9] F.C.M. Lau and S.H. Zhang, Optimal Gossiping in Paths and Cycles, Technical Report TR-97-10, Department of Computer Science, The University of Hong Kong, July 1997.

[10] M. Šoch and P. Tvrdík, "Optimal Gossip in Noncombining 2D-Meshes," *Proc. of 4th International Colloquium on Structural Information & Communication Complexity (SIROCCO'97)*, 1997.

[11] P. Tvrdík and M. Šoch, "Optimal Gossip in Store-and-Forward noncombining 2D Tori," *Proc. of EUROPAR'97*, 1997.