Journal of Algorithms **42,** 1–19 (2002)
doi:10.1006/jagm.2001.1200, available online at http://www.idealibrary.com on IDEAL®

Oblivious Gossiping on Tori[1]

Ulrich Meyer

Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany
E-mail: umeyer@mpi-sb.mpg.de; http://www.mpi-sb.mpg.de/~umeyer/

and

Jop F. Sibeyn

Department of Computing Science, Faculty of Science and Technology, Umeå University, 90187 Umeå, Sweden
E-mail: jopsi@cs.umu.se; http://www.cs.umu.se/~jopsi/

Received August 5, 1998

Near-optimal gossiping algorithms are given for two-dimensional and higher dimensional tori, assuming the full-port store-and-forward communication model. For two-dimensional tori, a previous algorithm achieved optimality in an intricate way, with an adaptive routing pattern. In contrast, the processing units in our algorithm forward the received packets always in the same way. We thus achieve almost the same performance with patterns that might be hardwired. © 2002 Elsevier Science

1. INTRODUCTION

Meshes and Tori. One of the most thoroughly investigated interconnection schemes for parallel computation is the $n \times n$ *mesh*, in which n^2 processing units, *PUs*, are connected by a two-dimensional grid of communication links. Its immediate generalizations are d-dimensional $n \times \cdots \times n$ meshes. Despite their large diameter, meshes are of great importance due to their simple structure and efficient layout.

Tori are the variant of meshes in which the PUs on the outside are connected with "wraparound links" to the corresponding PUs at the other

[1] Partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

0196-6774/02 $35.00

end of the mesh. Tori are node symmetric. Furthermore, the bisection width for tori is twice as large as that for meshes, whereas their diameter is smaller by a factor of two. Numerous parallel machines, such as the Intel Paragon, Cray T3D, and Cray T3E, have been built with two- and three-dimensional mesh and torus topologies. Tori can be embedded in meshes with load 1, dilation 2, and congestion 2. That is, for all those algorithms that run twice as fast on tori as on meshes, one can assume without loss of performance that the network is a torus.

Gossiping. *Gossiping* is a fundamental communication problem: Initially each of the P PUs knows some data of size s bytes, which must be routed so that in the end all PUs have the complete set of information of size $s \cdot P$ bytes (this problem is also called *all-to-all broadcast*).

Gossiping appears as a subroutine in many important problems. For example, if M numbers are to be sorted on P PUs, then a good approach is to select a set of m splitters [13, 18, 20] which must be made available in every PU. This means that we have to perform a gossip in which each of the P PUs contributes $s = m/P$ numbers. Gossiping is also used in parallel implementations of matrix-multiplication, LU-factorization, Householder transformations, and basic linear algebra operations [8, 9]. Another application of gossiping appears in algorithms for solving ordinary differential equations using parallel block predictor–corrector methods [19]. In each application of the block method, computations corresponding to the prediction are carried out by different PUs and these values are required by all other PUs. More information about gossiping can be found in [6, 10, 11, 15].

Earlier Work. A substantial amount of research has been carried out on variants of the gossiping problem on meshes and tori [3, 4, 7, 12, 16, 17]. Recently Šoch and Tvrdík [21] have analyzed the following variant of the gossiping problem:

- Packets of size s can be transferred in one *step* between adjacent PUs (store-and-forward model).
- In each step a PU can exchange packets with all its neighbors (full-port model).

They show that on a two-dimensional $n_1 \times n_2$ torus, the gossiping can be solved in $\lceil (n_1 \cdot n_2 - 1)/4 \rceil$ steps if $n_1, n_2 \geq 3$. This is optimal for fixed packet size s. The algorithm is based on time-arc-disjoint broadcast trees. The algorithm is *time-dependent* in the sense that the pattern according to which the PUs are forwarding the received packets is changing from step to step. Thus, for every routing decision, a PU has to perform some non-trivial computation. Precomputing all routing decisions once and for all and storing them in tables to facilitate fast lookup is practically not feasible as the gossiping may be needed in many different partitions. Another drawback of

this algorithm is its requirement for a non-constant buffer space where the PUs store packets which cannot be forwarded immediately. In subsequent papers [23–26] the approach has been extended to three-dimensional tori and the buffer size per PU has been reduced. However, the algorithm of [21] already requires the distinction of many cases, and the description is mainly in the form of pictures. The subsequent improvements resulted in even more complicated algorithms. We think these weaknesses are inherent to the time-dependent approach.

Alternatively, Yang and Wang [28] propose another optimal but simpler scheme for d dimensional tori and meshes. They also use broadcast trees but time-arc disjointness is circumvented by applying FIFO queues. Unfortunately, the queues may have to store $\Omega(P/d)$ packets. In a setting where huge packets are immediately processed upon receipt in their target PUs this might be a considerable disadvantage.

Practical Considerations. Modern parallel computers mostly apply *worm-hole routing* instead of the store-and-forward routing introduced above. With worm-hole routing, packets are transferred between their origin and destination as a stream of "flits." As long as there is no congestion, the time for such a transfer is more or less independent of the distance. For a packet of size s, it can be written as $t_{\text{startup}} + s \cdot t_{\text{feed/byte}}$ where t_{startup} denotes the "startup time" which is independent of the packet size, and $t_{\text{feed/byte}}$ denotes the "feeding time" which is needed to transfer one byte of a packet between a PU and the network.

Since routing patterns designed for the store-and-forward model are obviously congestion-free (communication is restricted to neighboring PUs), their time analysis using full-ported worm-hole routing is straightforward: if during each of T steps each PU transfers at most one packet of size s to each of its adjacent PUs, then the routing takes at most $T \cdot (t_{\text{startup}} + s \cdot t_{\text{feed/byte}})$ time.

For the gossiping problem with initially s bytes in each PU it is not required to have fixed packet size s. However, by splitting the initial information sets of the PUs and transmitting packets of size $s' = s/k, k > 1$ will require at least $\lceil (P-1) \cdot k/(2 \cdot d) \rceil$ steps. Hence, due to the increased number of startups this will never result in an improved routing time. Still, it may allow much simpler routing schemes. The performance loss depends on the ratios of s/s' and $t_{\text{startup}}/t_{\text{feed/byte}}$.

New Results. In this paper, we analyze the same problem as Šoch and Tvrdík did. Clearly, we cannot improve their optimality. Instead, we try to determine the minimal concessions that must be made to obtain an algorithm that is *time-independent* (also called *oblivious*) in the sense that the pattern according to which the PUs are forwarding the received packets is the same in every step of the algorithm. That is, in our gossiping algorithms,

after some $\mathcal{O}(d)$ precomputation on a d-dimensional torus, a PU knows once and for all that packets coming from direction x_i have to be forwarded in direction x_j, $1 \leq i, j \leq d$. No buffers are needed to store packets until they can eventually be forwarded. Time-independence ensures that the routing can be performed with minimal delay and PU internal memory. For a fixed size network the pattern might even be built into the hardware. This is also advantageous on a system in which the connections must be somehow switched.

Our first approach for d-dimensional tori splits the s bytes of information initially residing in each PU into d packets of size s/d each and routes them along d fixed edge-disjoint Hamiltonian cycles. More concretely, the ith packet of a PU is routed along both directions of the ith cycle. For packet size s/d this achieves the optimal number of steps, $\lceil (P-1)/2 \rceil$. One might ask whether the d edge-disjoint cycles could also be chosen so that there is no need to split the packets and that routing the entire packets of size s bytes along all d cycles in parallel for $\lceil (P-1)/(2 \cdot d) \rceil$ steps would solve the gossiping problem as well. In Section 3 we will show that this is not possible: even for $d = 2$, $\Omega(P)$ extra steps are needed.

Our second approach is to accept that an oblivious gossiping for packet size s takes a non-optimal number of routing steps. However, we replace the edge-disjoint Hamiltonian cycles by an improved routing scheme that limits the number of extra steps to $o(P)$. For $d = 2$, the algorithm is particularly simple and needs $n_1 \cdot n_2/4 + n_1/2 + n_2/2 + 2$ steps, which is just slightly more than optimal. Therefore, this algorithm might be preferable over the one from [21]. Generally, we aim to perform the gossiping with packets of size s in $P/(2 \cdot d) + o(P)$ steps on a d-dimensional torus by constructing partial Hamiltonian cycles: on a d-dimensional torus, we construct d cycles, which each cover $P/d + o(P)$ PUs. The layout of the cycles must fulfill the demand that whenever a PU is not covered by some cycles then those cycles are present in its adjacent PUs. Thus, a PU receives its packets either via the cycles it takes part in or with a one time step delay via a neighboring PU. For the practically relevant case $d = 3$, this gives the first simple and explicit construction achieving close to optimally: on an $n_1 \times n_2 \times n_3$ torus, our schedule requires $n_1 \cdot n_2 \cdot n_3/6 + \mathcal{O}(n_1 \cdot n_2 + n_1 \cdot n_3)$ steps for packet size s.

Contents. In Section 2, we describe the optimal time gossiping approach that routes packets of size s/d, $d \geq 2$, along d edge-disjoint Hamiltonian cycles. In Section 3, we prove that this approach cannot give the optimal number of steps for packets of size s on a two-dimensional torus, no matter how the Hamiltonian cycles are chosen. Then, in Section 4, we describe improved near-optimal time algorithms using packets of size s on two- and three-dimensional tori. Finally, in Section 5, we numerically compare the

performance of the gossiping algorithms of this paper and determine their range of optimality.

2. STEP-OPTIMAL ALGORITHMS

We assume that each PU initially holds some data of size s. In this section we present optimal gossiping algorithms for two-dimensional $n_1 \times n_2$ tori using packets of size $s/2$; i.e., conceptually the input of each PU is split into two packets, one for each cycle. The transfer of a packet between two adjacent PUs takes one step. We show that the gossiping can be performed in $n_1 \cdot n_2/2$ steps. We distinguish a few cases, depending on the parity of n_1 and n_2. Finally, we indicate how this idea can be generalized for higher dimensional tori.

2.1. *Two-Dimensional Tori, n_1, and n_2 Even*

First we settle the indexing of the torus. The PU with index i will be designated PU i. PU $(0, 0)$ lies in the upper-left corner. PU (i, j) lies in row i and column j. n_1 gives the number of rows and n_2 gives the number of columns. The routing rules are very simple: PU (i, j) determines whether j is odd or even; then it sets its routing rules as

$$j < n_2 - 1,\ j \text{ even: } T \leftrightarrow R; B \leftrightarrow L.$$

$$j < n_2 - 1,\ j \text{ odd: } T \leftrightarrow L; B \leftrightarrow R.$$

Here T, B, L, and R designate the directions *top, bottom, left*, and *right*, respectively. By $T \leftrightarrow R$, we mean that the packets coming from above should be routed on to the right and vice versa. The other $\leftrightarrow$ symbols are to be interpreted analogously. Only in the special case $j = n_2 - 1$ do we have the rule

$$j = n_2 - 1\text{: } T \leftrightarrow R; B \leftrightarrow L.$$

The resulting routing scheme is illustrated in Fig. 1.

THEOREM 1. *If every PU of an $n_1 \times n_2$ torus, with n_1 and n_2 even, holds s bytes of information then gossiping can be performed in $n_1 \cdot n_2/2$ steps using packets of size $s/2$.*

Proof. After splitting the initial s bytes of information every PU knows two packets of size $s/2$, one for each cycle. In each step, it receives four new packets, two from each cycle. In the last step only, a PU receives twice the same two packets. ∎

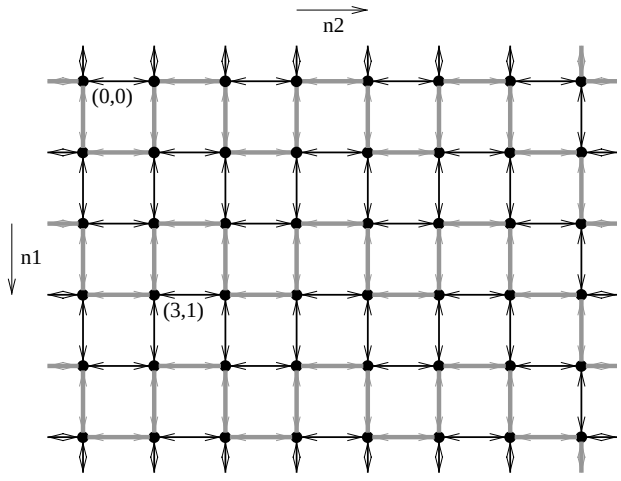


FIG. 1. A Hamiltonian cycle on a 6×8 torus (black), whose complement (drawn with gray lines) also gives a Hamiltonian cycle.

2.2. *Two-Dimensional Tori, n_1 Odd, and n_2 Even*

If n_1 or n_2 is odd, we must use a slightly modified schedule. In this section we consider only the case n_1 odd and n_2 even. The case n_1 even and n_2 odd can be solved in a similar way. Here we do not construct complete Hamiltonian cycles, but cycles that visit most PUs and pass within distance one from the remaining PUs. The PUs in column $n_1 - 2$ will store just one packet instead of two. This situation can be reached within one step: By assuming that each PU initially holds s bytes then for $0 \le i < n_1, 1 \le j < n_2 - 1$, PU (i, j) transmits $j \cdot s/(2 \cdot n_2)$ bytes of its own data to PU $(i, j-1)$. Furthermore, each PU in column $n_1 - 2$ sends another $s/(2 \cdot n_2)$ bytes to its right neighbor. Now the PUs can reorganize their data in packets of size at most $(1 + 1/n_2) \cdot s/2$, each according to our requirements. Except for column $n_1 - 2$, the rules governing how to pass on the packets are the same as in the basic case. In column $n_1 - 2$ we perform

$$i = n_1 - 2\text{: } L \leftrightarrow R; L \rightarrow T; L \rightarrow B.$$

PUs which do not lie on a given cycle, out-of-cycle-PUs, abbreviated *OOC-PUs*, are provided with the packets which are transferred along this cycle by their neighboring on-cycle-PUs, *OC-PUs*. With respect to different cycles, a PU can be both out-of-cycle and on-cycle. The packets received by an OOC-PU are not forwarded. This can be achieved in such a way that a connection has to transfer only one packet in each step. The resulting routing scheme is illustrated in Fig. 2.

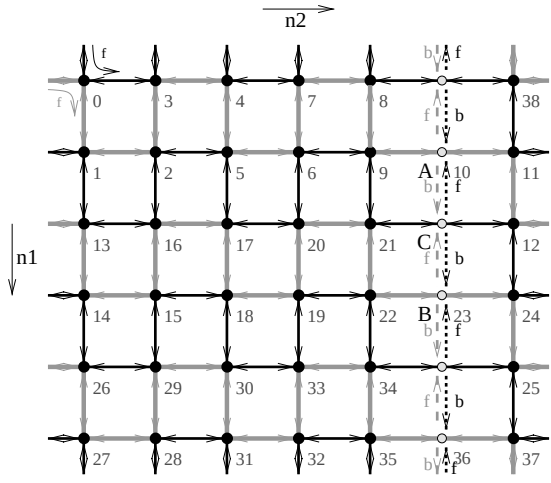


FIG. 2. Partial Hamiltonian cycles on a 6×7 torus. The PUs in column 5 lie on only one cycle. Such a PU passes the packets that are running forward (i.e., enter from left) on this cycle to the PU above it, and those that are running backward (i.e., enter from right) to the PU below it.

The optimal performance in the basic case (n_1, n_2 even) was achieved by using both directions on every cycle, thus halving the circulation time. The inclusion of OOC-PUs complicates this approach: We need two different OC-PUs, A and B, in order to provide an OOC-PU C with packets from both directions of that cycle. Let A transfer the packets that are walking backward whereas B is responsible for the packets going forward. If m denotes the cycle length, and A and B are not adjacent on the cycle, then $\lceil m/2 \rceil$ circulation steps are not enough: some packets are received from both directions, while others pass by without notice. In the example of Fig. 2 we have $m = 39$ and the OOC-PU C receives the following packets from the gray cycle (double-receives are marked with a star):

Step	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Rec. via A	10	11	12	13	14	15	16	17^*	18^*	19^*	20^*	21^*	22^*	23^*	24	25	26	27	28	29	30	31	32	33	34	35	36^*
Rec. via B	23	22	21	20	19	18	17	16^*	15^*	14^*	13^*	12^*	11^*	10^*	9	8	7	6	5	4	3	2	1	0	38	37	36

Let l be the number of OC-PUs from A to B in the forward direction. Then during the first $\lfloor l/2 \rfloor$ steps, C receives two new packets in each step. In the course of the next $\lceil l/2 \rceil$ steps at most one new packet is received. Finally, the remaining packets are received during another $\lceil m/2 - l/2 \rceil$ step. The cycles are chosen such that $m = N - n_1/2$ and $l = 2 \cdot n_2$. Thus, the whole algorithm with two packets per PU needs $n_1 \cdot n_2/2 + 1 + n_2$ steps. It should also be taken into account that the packet size compared to the basic

case has been increased by a factor of $(1 + 1/n_2)$ after the first step. Under our assumption that s is large enough to make the feeding time dominant over the startup costs, the increase in packet size also results in longer feeding times. However, the time increase is bounded by an equivalent of $\mathcal{O}(n_1 \cdot n_2 \cdot (1 + 1/n_2)) = \mathcal{O}(n_1)$ steps.

If both n_1 and n_2 are odd, optimal gossiping for two packets per PU can be achieved with a slightly modified schedule. Details are given in [14].

For d-dimensional tori, one should first construct d edge-disjoint Hamiltonian cycles. Such constructions are described in [1, 2, 5]. Then, if every PU holds d packets of size s/d, each packet is forwarded for $P/2$ steps in both directions along one of the Hamiltonian paths. In this way, after $P/2$ steps, each PU has received all packets which are optimal for packet size s/d. The routing is trivial, but the precomputation may be quite involved. In particular, it is non-trivial to compute the local routing scheme for each PU individually without computing the whole paths.

In the following section we will show that the approach above essentially needs to split the initial data into smaller packets in order to achieve the optimal bound for the number of steps. Otherwise, $\Omega(P)$ extra steps are needed. Therefore, in Section 4 we give an alternative generic construction that in principle works for arbitrary dimensions and which yields a routing that can be performed in $P/(2 \cdot d) + o(P)$ steps.

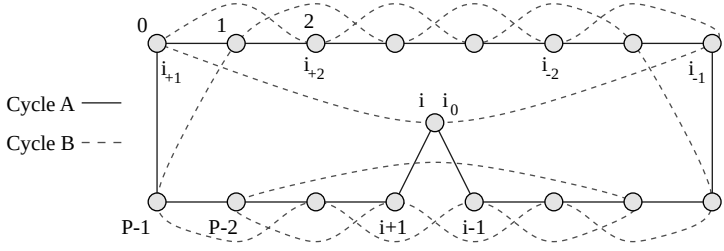
3. A LOWER BOUND FOR GOSSIPING VIA EDGE-DISJOINT HAMILTONIAN CYCLES

The communication network at hand imposes limits on the way one can choose edge-disjoint Hamiltonian cycles; sometimes they do not exist at all. In contrast to Section 2 we now consider the case that the s bytes of initial information in each PU are not split into two packets but are always transferred as one packet of size s. We prove a lower bound for the number of steps needed if the packets are circulated in both directions on both cycles in parallel. The result holds for any network whose underlying graph is 4-regular; thus it holds in particular for two-dimensional tori.

THEOREM 2. *Consider a gossiping on a* 4*-regular graph network with P PUs and one non-modifiable packet per PU such that each packet is bidirectionally circulated via two edge-disjoint Hamiltonian cycles in parallel. Any choice of the two cycles requires at least* $(11/40) \cdot P - \mathcal{O}(1)$ *steps until all PUs have seen all packets.*

Proof. Let us assume that there exist two edge-disjoint Hamiltonian cycles. Furthermore, we may assume that the PUs are indexed so that the first cycle, A, visits the PUs in consecutive order: $0, 1, \ldots, P-2, P-1$.

Thus, during the routing PU i will receive the packets from PU $(i - j) \bmod P$ and PU $(i + j) \bmod P$ in Step j, $0 < i, j < P$. In the same step, PU i will also receive along the second cycle, B, the packets from PU i_{-j} and PU i_{+j} where $\ldots, i_{-2}, i_{-1}, i_0, i_{+1}, i_{+2}, \ldots$ denotes the order of the second cycle relative to PU i.

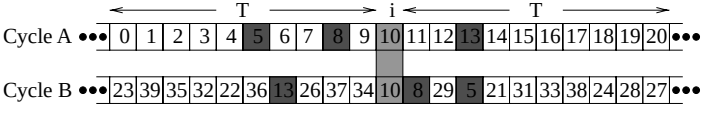


In the best case every PU should receive two new packets from each cycle in every step unless all $P - 1$ packets are present. Then clearly $T = \lceil (P - 1)/4 \rceil$ steps would suffice to complete the gossiping. However, due to a non-optimal choice of the cycles or limitations of the network *double-receives* may occur: this means that during the first T steps a PU i receives k_i packets via *both* Hamiltonian cycles. These double-receives will delay the gossiping by at least $\max_i \lfloor k_i/4 \rfloor$ steps since during each subsequent step PU i can receive at most four of the so far missing packets. Using the above notation we have $k_i = |\mathscr{A}_i \cap \mathscr{B}_i \backslash \{i\}|$, where $\mathscr{A}_i = \{(i - T) \bmod P, \ldots, (i + T) \bmod P\}$ and $\mathscr{B}_i = \{i_{-T}, \ldots, i_{+T}\}$. In order to avoid tedious rounding we will use $T = P/4 - \mathscr{O}(1)$.

In the following we will prove that a PU i with $k_i \geq P/10 - \mathscr{O}(1)$ exists, causing at least $P/40 - \mathscr{O}(1)$ extra receive steps. Let us first choose $i = \lfloor P/4 \rfloor$. If $k_i \geq P/10$, we are done. Otherwise at least $2 \cdot T - P/10 = 2/5 \cdot P - \mathscr{O}(1)$ elements from $\mathscr{B}_i$ have indices in $\overline{\mathscr{A}_i} = \{0, \ldots, P - 1\} \backslash \mathscr{A}_i$. Let $\mathscr{C}_i = \overline{\mathscr{A}_i} \cap \mathscr{B}_i$. The new PU index i' is set to the median of the elements in $\mathscr{C}_{\lfloor P/4 \rfloor}$. We will show that at least $P/10 - \mathscr{O}(1)$ elements from $\mathscr{C}_{\lfloor P/4 \rfloor}$ are double-receives for PU i'.

Before we go on, we illustrate the above definitions with an example.

EXAMPLE 1. Consider the following two cycles for $P = 41$ and $T = \lceil (P - 1)/4 \rceil = 10$.

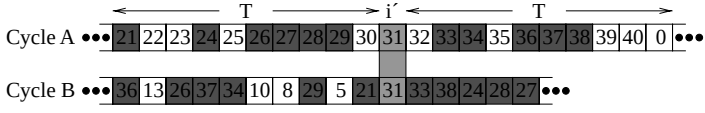


We start by choosing $i = \lfloor P/4 \rfloor = 10$. Then $\mathscr{A}_{10} = \{0, \ldots, 20\}$, $\overline{\mathscr{A}_{10}} = \{21, \ldots, 40\}$ and $\mathscr{B}_{10} = \{5, 8, 10, 13, 21, 22, 23, 24, 26, 27, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39\}$.

The set of double-receives is given by $\mathscr{A}_{10} \cap \mathscr{B}_{10} \backslash \{10\} = \{5, 8, 13\}$, and thus $k_{10} = 3 < P/10$. We must choose another PU i'. For $\mathscr{C}_{10} = \overline{\mathscr{A}_{10}} \cap \mathscr{B}_{10}$ we have

$$\mathscr{C}_{10} = \{21, 22, 23, 24, 26, 27, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39\}.$$

The median of the elements in $\mathscr{C}_{10}$ is $i' = 31$. After realigning the cycles according to PU i' we have the following situation:

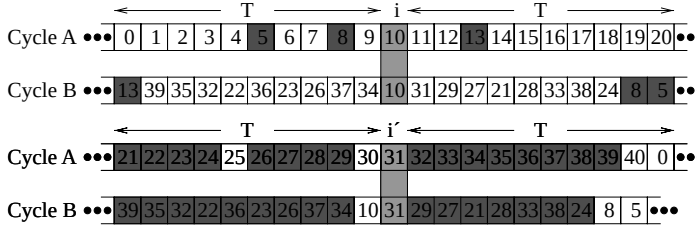


Now $\mathscr{A}_{31} = \{21, \ldots, 40, 0\}$. Due to the incomplete specification of the second cycle, $\mathscr{B}_{31}$ is not fully known. However, in any case

$$\mathscr{B}_{31} \supset \{5, 8, 10, 13, 21, 24, 26, 27, 28, 29, 31, 33, 34, 36, 37, 38\}.$$

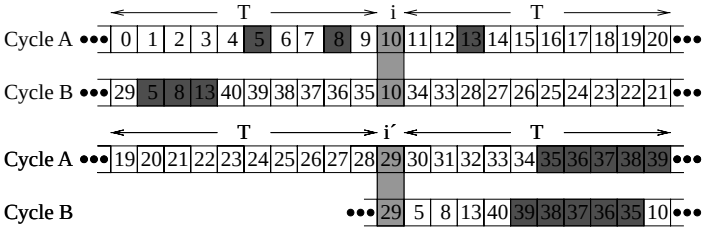
Hence, $\mathscr{A}_{31} \cap \mathscr{B}_{31} \backslash \{31\} \supset \{21, 24, 25, 26, 27, 28, 29, 33, 34, 36, 37, 38\}$, and therefore $k_{31} \geq 12 > P/10$.

We only have to deal with the case that a new PU $i' \neq \lfloor P/4 \rfloor$ has to be chosen. As i' is set to the median of at least $2/5 \cdot P - \mathscr{O}(1)$ elements out of $\overline{\mathscr{A}_{\lfloor P/4 \rfloor}}$, $i' \in \{P/2 + P/5 - \mathscr{O}(1), \ldots, P - 1 - P/5 + \mathscr{O}(1)\}$. The relative position of i' on the cycle B compared to $i = \lfloor P/4 \rfloor$ and the other elements of $\mathscr{C}_{\lfloor P/4 \rfloor}$ determines how many elements of $\mathscr{C}_{\lfloor P/4 \rfloor}$ belong to $\mathscr{A}_{i'}$ and are therefore double-receives in PU i': in the worst case, i' is a neighbor of $\lfloor P/4 \rfloor$ on B and all other elements of $\mathscr{C}_{\lfloor P/4 \rfloor}$ are positioned compactly around these two indices. Then $|\mathscr{C}_{\lfloor P/4 \rfloor}| - \mathscr{O}(1) \geq 2/5 \cdot P - \mathscr{O}(1)$ double-receives occur. Here is an appropriate modification of our previous example:



In the best case, PU i' is followed on one side of the B cycle by at least $P/4$ PUs storing elements not belonging to $\mathscr{A}_{i'}$. Let us assume that this is the case on the left side; for example, $\{i'_{-P/4}, \ldots, i'_{-1}\} \cap \mathscr{A}_{i'} = \varnothing$. This situation might occur, for example, when i' was located just at the left end of $\mathscr{B}_i$,

$i' = i_{-P/4}$. Then superfluous receives for PU i' may only arrive from the right side. Let us again look at an example:



We have $|\mathscr{C}_i| \geq 2/5 \cdot P - \mathcal{O}(1)$; thus in the best case the first $P/2 - 2/5 \cdot P - \mathcal{O}(1) = P/10 + \mathcal{O}(1)$ elements received from the right side of the B cycle may not belong to $\mathscr{A}_{i'}$ as well. After that, the number of double-receives is minimized if $\mathscr{C}_i$ consists of two chunks of elements: $\lceil |\mathscr{C}_i|/2 \rceil$ small elements so that the median i' is as small too and $\lfloor |\mathscr{C}_i|/2 \rfloor$ large elements so that the biggest of them will not belong to $\mathscr{A}_{i'}$ and therefore will not cause double-receives. Those big elements should float into PU i' prior to the smaller ones. See the example above. However, as observed before, $i' \geq P/2 + P/5 - \mathcal{O}(1)$. Consequently, $\mathscr{A}_{i'}$ spans large elements up to at least $P/2 + P/5 + P/4 - \mathcal{O}(1) = 19/20 \cdot P - \mathcal{O}(1)$. Thus, at most $P/20 + \mathcal{O}(1)$ big elements from $\mathscr{C}_i$ which do not belong to $\mathscr{A}_{i'}$ are received from the right side.

Altogether, during the first $P/4$ steps PU i' receives on the B cycle at least $P/4 - P/10 - P/20 - \mathcal{O}(1) = P/10 - \mathcal{O}(1)$ elements fitting into $\mathscr{A}_{i'}$. ∎

4. ONE-PACKET ALGORITHMS

In this section we give another practical alternative to the approach in [21]: we present algorithms which require only one packet per PU; i.e., having s bytes of initial information in each PU, they operate with packets of size s. They are optimal to within $o(P)$ steps. The simple idea is based on *partial* Hamiltonian cycles that include only a faction of all PUs: we construct d edge-disjoint cycles of length $P/d + o(P)$. Each of the cycles must have the following special property: if a PU does not lie on a cycle, then this PU must be adjacent to two PUs that do lie on the cycle. Each of these two PUs will transmit the packets from one direction of the cycle to the OOC-PU.

4.1. *Two-Dimensional Tori*

The construction of two partial Hamiltonian cycles with the desired properties is easy for two-dimensional tori. At the same time, this clearly

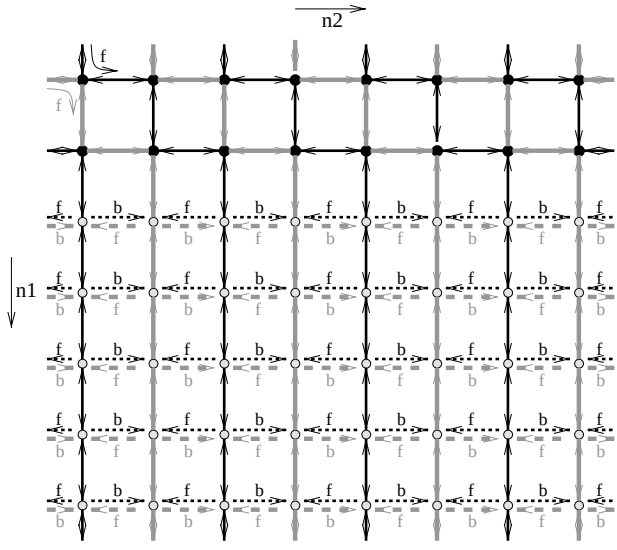


FIG. 3. Two edge-disjoint cycles, each of which comes within distance at most one from all PUs.

illustrates our intentions. There are two axes: the x_1-axis, running horizontally and the x_2-axis running vertically. PU $(0, 0)$ is assumed to lie in the upper-left corner. n_1 and n_2 denote the size of the torus in the x_1 and x_2 directions, respectively. We assume that n_1 is even and that $n_2 \geq 2$.

The construction is somewhat similar to the approach presented in Section 2.2. The two edge-disjoint cycles, which are illustrated in Fig. 3, can be described as follows: In the lower part of all columns, they only consist of vertical connections. In the highest two rows, there is a "zigzag" pattern, consisting of a horizontal connection, followed by a vertical connection, followed by a horizontal connection. Thus, starting in the PUs of row 0, the cycles make positive moves along axes x_1, x_2, and x_1 and then $n_2 - 1$ moves along the x_2-axis. One more move along the x_2-axis brings us over the wraparound connection to the next zigzag. Both cycles have total length $n_1 \cdot n_2/2 + n_1$. Binding of the OOC-PUs is done exactly as in the algorithm in Section 2.2; the case of odd n_1 can be treated by inserting one special column.

THEOREM 3. *If every PU of an $n_1 \times n_2$ torus holds s bytes of initial data, then gossiping can be performed in $n_1 \cdot n_2/4 + n_1/2 + n_2/2 + 2$ steps using packets of size s.*

Proof. Each of the cycles consists of $n_1 \cdot n_2/2 + n_1$ PUs, so using both directions in parallel one needs $n_1 \cdot n_2/4 + n_1/2 + 1$ steps to spread all the

packets within the cycle. For every OOC-PU, the two supplying OC-PUs are separated by $n_2 + 1$ other PUs on their cycle. According to the discussion of double-receives in Section 2.2 this accounts for additional $\lceil (n_2 + 1)/2 \rceil$ steps. ∎

4.2. *Three-Dimensional Tori*

For three-dimensional tori, we generalize the scheme of Section 4.1, showing more abstractly the underlying approach. PU $(0, 0, 0)$ is assumed to lie in the upper-back-left corner. The three axes are denoted as x_1, x_2, and x_3. They run left-, front-, and downward, respectively. We consider an $n_1 \times n_2 \times n_3$ torus, for suitable n_1, n_2, and n_3: we assume that n_1 is a multiple of 3, that n_2 is a multiple of n_1, and that $n_3 \geq 3$. As a generalization of the two-dimensional pattern, we construct a pattern that is similar to the bundle of rods in a nuclear power plant.

For two-dimensional tori, there are two cycles, each a concatenation of $n_1/2$ *laps*. Each lap consists of a zigzag followed by a long move along the x_2-axis. The zigzags are needed to bring us two positions further, connecting the laps of a cycle. For three-dimensional tori, there are three cycles. These are isomorphic, except that they start in different positions: Cycle j, $0 \leq j \leq 2$, starts in PU $(j, 0, 0)$. Here, the zigzags bring us three positions further. The first type of zigzag consists of positive moves along the following sequence of axes: $(1, 3, 1, 3, 1)$. A zigzag is followed by $n_3 - 2$ moves along the x_3-axis, the last move traversing a wraparound connection. In this way we can fill up a plane, but in order to get to the next plane, there must be a second type of zigzag consisting of moves along the following sequence of axes: $(1, 3, 1, 3, 2)$. Thus, a complete cycle is the concatenation of $n_1/3 \cdot n_2$ laps. After each $n_1/3 - 1$ laps using a zigzag of the first type, one lap with a zigzag of the second type follows. The cycles are illustrated in Fig. 4 and Fig. 5.

That the constructed cycles have all the desired properties can be tested with the help of the following reduction.

LEMMA 1. *PU* $(x_1 + 3 \cdot k_1, x_2 + 3 \cdot k_2, x_3)$, $k_1, k_2 \geq 0$, *lies on the same cycle as PU* (x_1, x_2, x_3).

Proof. The $(1, 3, 1, 3, 1)$ zigzag brings us three positions further along the x_1-axis, This implies that the cycle in all positions $(x_1 + 3 \cdot k, x_2, x_3)$ is the same as that in (x_1, x_2, x_3). The zigzag $(1, 3, 1, 3, 2)$ brings us two positions further along the x_1-axis and one position along the x_2-axis. Thus, in position $(x_1 + 6 \cdot k, x_2 + 3 \cdot k, x_3)$ we are on the same cycles as in position (x_1, x_2, x_3). But, according to the first rule we may shift along the x_1-axis over multiples of three. This gives the lemma. ∎

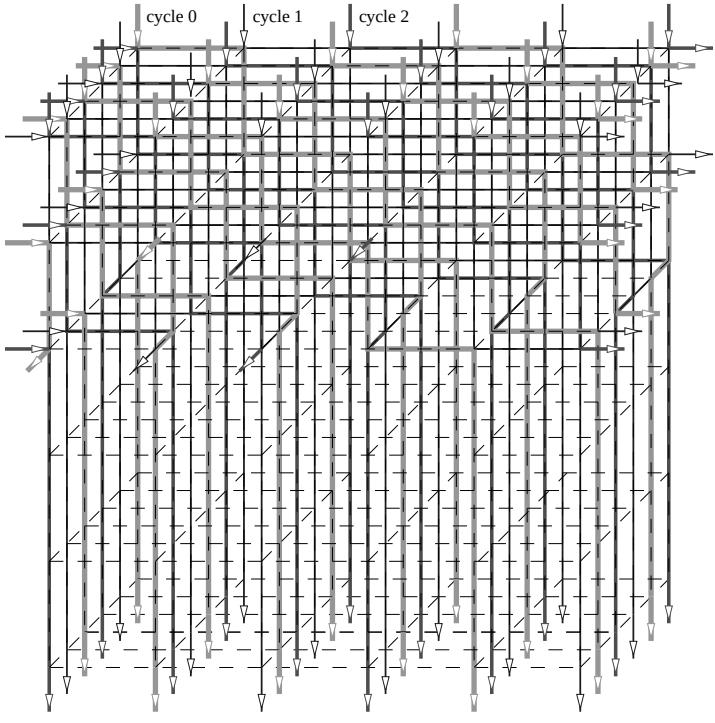


FIG. 4. One edge-disjoint cycle used in the one-packet algorithm on a three-dimensional torus.

Combining this with the fact that for any given value of $x_3 > 3$, the structure of the rods is the same, gives

COROLLARY 1. *For testing that the cycles are indeed cycles and edge-disjoint and for testing that the locality structure is as desired, it is sufficient to test these properties for a* $3 \times 3 \times 4$ *torus.*

The $3 \times 3 \times 4$ torus is so small that it is easy to verify that the cycles are edge-disjoint cycles. Cutting through the rods for $x_3 = 4$ gives the following pattern of cycle numbers:

$$\begin{matrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{matrix}$$

In this way, PU (x_1, x_2, x_3), $x_3 > 3$, on Cycle j can be supplied with packets which do not lie on its own cycle: it receives packets running forward on Cycle $(j+1) \bmod 3$ from the PU $((x_1 - 1) \bmod n_1, x_2, x_3)$ and packets running backward from PU $((x_1, (x_2 - 1) \bmod n_2, x_3)$. Similarly,

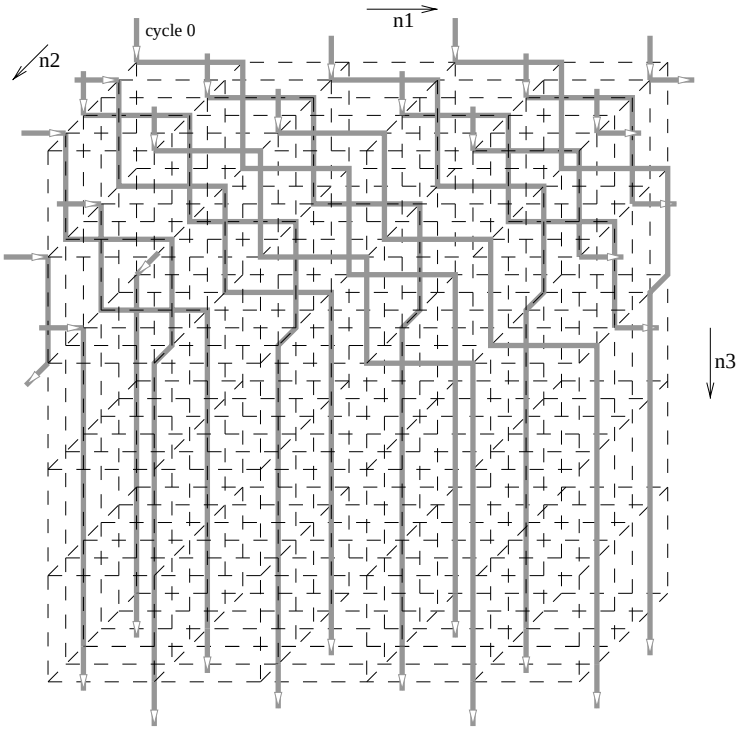


FIG. 5. Three edge-disjoint cycles used in the one-packet algorithm on a three-dimensional torus.

it is supplied with packets running forward on Cycle $(j-1) \bmod 3$ from PU $((x_1+1) \bmod n_1, x_2, x_3)$ and with backward-running packets from PU $((x_1, (x_2+1) \bmod n_2, x_3)$. Each pair of supporting OC-PUs is separated by $(n_1/3+1) \cdot n_3 - 1$ other PUs.

In the upper part of our reactor, exactly two cycles pass through each PU P, and the shifts are so that the connections that are not used by cycle traffic lead to PUs that lie on the third cycle. These facts can be easily established for the $3 \times 3 \times 4$ torus. At most $2 \cdot (n_1/3+1) \cdot n_3 - 1$ PUs lie between two supporting OC-PUs.

THEOREM 4. *If every PU of an $n_1 \times n_2 \times n_3$ torus, with n_1 a multiple of 3, n_2 a multiple of n_1, and $n_3 \geq 3$, holds s bytes of initial data, then gossiping can be performed in $n_1 \cdot n_2 \cdot n_3/6 + \mathcal{O}(n_1 \cdot n_2 + n_1 \cdot n_3)$ steps using packets of size s.*

Proof. Each lap has length n_3+3. There are $n_1/3 \cdot n_2$ laps, so the cycles have length $n_1 \cdot n_2 \cdot n_3/3 + n_1 \cdot n_2$. Packets are routed in both directions along them, so after $n_1 \cdot n_2 \cdot n_3/6 + n_1 \cdot n_2/2$ steps, a PU has received all

the packets that have started in a PU that lies on the cycle(s) in which the first-mentioned PU lies. The remaining packets running on the other cycles are received from the PU's neighbors one step after they received those packets. Each pair of supporting OC-PUs is separated by $\mathcal{O}(n_1 \cdot n_3)$ other PUs. Therefore, double-receives in the OOC-PUs can cause at most $\mathcal{O}(n_1 \cdot n_3)$ extra steps (compare Section 2.2). ∎

The above idea can be extended to perform gossiping on d-dimensional $n_1 \times n_2 \times \cdots \times n_d$ tori in $(1 + o(1)) \cdot n_1 \cdot n_2 \cdots n_d/(2 \cdot d)$ steps. In order to achieve this, one must construct a more complex zigzag pattern, so that the right locality structure is achieved among the rods (every rod on a given cycle is adjacent to two rods that belong to any of the other $d - 1$ cycles) and so that in the head every PU is adjacent to any cycle it is not lying on. It is very hard though to come us with a convincing proof that such constructions are correct, and because the practical importance of them is limitted, we have omitted them here.

5. COMPARISON OF PERFORMANCES

In this section we numerically compare the performance of the gossiping algorithms of this paper and determine their range of optimality. As already pointed out in Section 1, the cost for a step of a store-and-forward algorithm is determined by the start-up time, the packet size, and the speed of the connections to adjacent neighbors: if during a step each PU transfers at most one packet of size s' to each of its adjacent PUs, then the step takes at most $t_{\text{startup}} + s' \cdot t_{\text{feed/byte}}$ time.

In the gossiping problem every PU initially holds s bytes of data. On a d-dimensional $n \times \cdots \times n$ torus, the algorithms from Section 2 require $n^d/2$ steps, in each of which a PU sends packets of size $s' = s/d$ to all its neighbors. This implies a routing time

$$T_{\text{opt}}(n, d) = n^d/2 \cdot (t_{\text{startup}} + s/d \cdot t_{\text{feed/byte}}).$$

The algorithms from Section 4 require $n^d/(2 \cdot d) + n^d/(2 \cdot n) + 2$ steps, in each of which a PU sends packets of size $s' = s$ to all its neighbors. Thus, for the corresponding routing time T_{one} we find

$$T_{\text{one}}(n, d) \simeq n^{d-1}/2 \cdot (n/d + 1) \cdot (t_{\text{startup}} + s \cdot t_{\text{feed/byte}}).$$

The routing time T_{stv} for the algorithm from [21] is given by

$$T_{\text{stv}}(n, d) \simeq n^d/(2 \cdot d) \cdot (t_{\text{startup}} + s \cdot t_{\text{feed/byte}}).$$

How much performance is lost by applying our simple time-independent algorithms instead of the complicated approach of [21] depends on the value of the ratio

$$r = t_{\text{startup}}/(s \cdot t_{\text{feed/byte}}).$$

For different parallel computers and values of s, the value of r can be large (up to 10^4) or small (less than 1). We give a few examples: on a cluster of workstations we found $t_{\text{startup}} \simeq 10^{-3}$ s and $t_{\text{feed/byte}} \simeq 10^{-7}$ s; hence $r \simeq 10{,}000/s$. On a Cray T3E, we measured $t_{\text{startup}} \simeq 5 \cdot 10^{-6}$ s and $t_{\text{feed/byte}} \simeq 10^{-8}$ s, yielding $r \simeq 5{,}000/s$.

For $r = 1$, the time consumption for a step with packet size s is equally split between the startup time and the feeding time. Hence, by replacing such a step by two steps with packet size $s/2$, the total time will increase by a factor of 1.5. The smaller r is, the less important is the number of steps, provided that the same total amount of data is transferred.

Comparing $T_{\text{opt}}(n, d)$ and $T_{\text{stv}}(n, d)$ we find

$$\frac{T_{\text{opt}}(n, d)}{T_{\text{stv}}(n, d)} \simeq \frac{1 + d \cdot r}{1 + r}.$$

Thus, for example, on the T3E with its three-dimensional torus structure our step-optimal algorithm with packet size $s/3$ needs $s \geq 5000 \cdot 1.9/0.1 = 95{,}000$ in order to be at most 10% slower than the time-dependent approach of [21]. In contrast, our algorithms from Section 4 use the same packet sizes as those in [21]. The performance loss due to time-independence is bounded by a factor of $1 + \mathcal{O}(d/P)$; for the T3E with 512 PUs, this is less than 1% loss.

TABLE I
Comparison between Normalized Times Taken by the Algorithms from [22] (Top), from Section 2 (Middle), and from Section 4 (Bottom)

	r					
n	0.01		0.1		1	
	4	11	4	12	8	21
4	4	11	5	14	12	43
	6	19	7	21	12	37
	65	689	70	751	128	1365
16	65	702	77	887	192	2730
	73	818	79	891	144	1620
	1034	44110	1126	48041	2048	87346
64	1044	44958	1229	56754	3072	174719
	1067	46152	1162	50264	2112	91390

Note. In each cell we give the result for $d = 2$ (left) and $d = 3$ (right).

Normalized numerical results for some characteristic values of n and r are given in Table I. The actual routing times can be obtained by multiplication with the factor $s \cdot t_{\text{feed/byte}}$. For $r > 1$, none of these approaches makes sense, because then one should better apply a strategy that requires non-optimal routing volume but substantially fewer startups (see [4, 12, 16, 26, 27]). We see that mostly the one-packet algorithm from Section 4 is hardly slower than the algorithm from [21], but it is much simpler and far easier to generalize for higher dimensions. Therefore, we think that in most cases it may constitute a practical alternative. The algorithm from Section 2 may be attractive for $d = 2$ and small r: for that case it is very simple, while it achieves almost optimal performance.

6. CONCLUSION

We have completed the analysis of the gossiping problem on full-port store-and-forward tori. In [21] only one interesting aspect of this problem was considered. We have shown that an almost equally good performance can be achieved by simpler time-independent algorithms.

REFERENCES

1. B. Alspach, J.-C. Bermond, and D. Sotteau, Decomposition into cycles. I. Hamilton decompositions, *in* "Proceedings of the Workshop on Cycles and Rays, Montreal, 1990."
2. J. Aubert and B. Schneider, Decomposition de la Somme Cartesienne d'un Cycle et de l'Union de Deux Cycles Hamiltoniens en Cycles Hamiltonien, *Discrete Math.* **38** (1982), 7–16.
3. M. Barnett, R. Littlefield, D. G. Payne, and R. van de Geijn, Global combine on mesh architectures with wormhole routing, *in* "Proceedings of the 7th International Parallel Processing Symposium," pp. 13–16, IEEE, 1993.
4. O. Delmas and S. Perennes, Circuit-switched gossiping in 3-dimensional torus networks, *in* "Proceedings of the 2nd International Euro-Par Conference," Lecture Notes in Computer Science, Vol. 1123, pp. 370–373, Springer-Verlag, Berlin/New York, 1996.
5. M. F. Foregger, Hamiltonian decomposition of products of cycles, *Discrete Math.* **24** (1978), 251–260.
6. P. Fraigniaud and E. Lazard, Methods and problems of communication in usual networks, *DAMATH Discrete Appl. Math. Combin. Oper. Res. Comput. Sci.* **53** (1994), 79–133.
7. P. Fraigniaud and J. G. Peters, Structured communication in torus networks, *in* "Proceedings of the 28th Hawaii Conference on System Science, 1995," pp. 584–593.
8. G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker, "Solving Problems on Concurrent Processors, Vol. 1, General Techniques and Regular Problems," Prentice-Hall International, Englewood Cliffs, NJ, 1988.
9. D. Gannon and J. Van Rosendale, On the impact of communication complexity on the design of parallel numerical algorithms, *IEEE Trans. Compu.* **C-33** (1984), 1180–1194.
10. S. M. Hedetniemi, T. Hedetniemi, and A. L. Liestman, A survey of gossiping and broadcasting in communication networks, *Networks* **18** (1988), 319–349.

11. J. Hromkovič, R. Klasing, B. Monien, and R. Peine, Dissemination of information in interconnection networks (broadcasting and gossiping), *in* "Combinatorial Network Theory," (F. Hsu and D. Z. Du, Eds.), pp. 125–212, Kluwer Academic, Dordrecht/Norwell, MA, 1996.
12. B. Juurlink, J. F. Sibeyn, and P. S. Rao, Gossiping on meshes and tori, *IEEE Trans. Parallel and Distributed Systems* **9**, No. 6 (1998), 513–525.
13. M. Kaufmann and J. F. Sibeyn, Randomized multipacket routing and sorting on meshes, *Algorithmica* **17** (1997), 224–244.
14. U. Meyer and J. F. Sibeyn, "Time-Independent Gossiping on Full-Port Tori," Technical Report MPI-I-98-1014, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1998.
15. A. Pelc, Fault-tolerant broadcasting and gossiping in communication networks, *Networks* **28** (1996), 143–156.
16. J. G. Peters and M. Syska, Circuit-switched broadcasting in torus networks, *IEEE Trans. Parallel and Distributed Systems* **7** (1996), 246–255.
17. B. Plateau and D. Trystam, Optimal total exchange for a 3-D torus of processors, *Inform. Process. Lett.* **42** (1992), 95–102.
18. S. Rajasekaran, k-k routing, k-k sorting, and cut-through routing on the mesh, *J. Algorithms* **19** (1995), 361–382.
19. P. S. Rao and G. Mouney, "Data Communications in Parallel Block Predictor–Corrector Methods for solving ODEs," Technical Report 95399, LAAS-CNRS, France, 1995.
20. J. Reif and L. G. Valiant, A logarithmic time sort for linear size networks, *J. ACM* **34**, No. 1 (1987), 68–76.
21. M. Šoch and P. Tvrdík, Optimal gossip in store-and-forward noncombining 2-D tori, *in* "Proceedings of the 3rd International Euro-Par Conference," Lecture Notes in Computer Science, Vol. 1300, pp. 234–241, Springer-Verlag, Berlin/New York, 1997.
22. M. Šoch and P. Tvrdík, Time-optimal gossip in noncombing 3-D tori, *in* "Proceedings of the 5th International Colloquium on Structural Information and Communication Complexity," pp. 259–271, Carleton Scientific, Waterloo, Ontario, 1998.
23. M. Šoch and P. Tvrdík, Time-optimal gossip in noncombing 2-D tori with constant buffers, *in* "Proceedings of the 4th International Euro-Par Conference," Lecture Notes in Computer Science, Vol. 1470, pp. 1047–1050, Springer-Verlag, Berlin/New York, 1998.
24. M. Šoch and P. Tvrdík, Time-optimal gossip in noncombining 3-D tori with constant buffers, *in* Proceedings of the 2nd IASTED International Conference on Parallel and Distributed Computing and Networks, pp. 666–671, Acta Press, Calgary, 1998.
25. M. Šoch and P. Tvrdík, "Bufferless Gossip of Large Packets in Noncombining 2-D Tori," Research Report DC-99-03, Department of CS&E, Czech Technical University, Prague, 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* **9**, No. 5 (1988), 442–458.
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* **8**, No. 1 (1997), 41–58.
28. Y. Yang and J. Wang, Efficient all-to-all broadcast in all-port mesh and torus networks, *in* "Proceedings of the Fifth International Symposium on High-Performance Computer Architecture," pp. 290–299, IEEE, 1999.