# Gossiping on Meshes and Tori

Ben H.H. Juurlink, Jop F. Sibeyn, *Member*, *IEEE Computer Society*,
and P.S. Rao, *Member*, *IEEE*

**Abstract**—Algorithms for performing gossiping on one- and higher-dimensional meshes are presented. As a routing model, the practically important wormhole routing is assumed. We especially focus on the trade-off between the start-up time and the transmission time. For one-dimensional arrays and rings, we give a novel lower bound and an asymptotically optimal gossiping algorithm for all choices of the parameters involved. For two-dimensional meshes and tori, a simple algorithm composed of one-dimensional phases is presented. For an important range of packet and mesh sizes, it gives clear improvements upon previously developed algorithms. The algorithm is analyzed theoretically and the achieved improvements are also convincingly demonstrated by simulations, as well as an implementation on the Paragon. On the Paragon, our algorithm even outperforms the gossiping routine provided in the NX message-passing library. For higher-dimensional meshes, we give algorithms which are based on an interesting generalization of the notion of a diagonal. These algorithms are analyzed theoretically, as well as by simulation.

**Index Terms**—Gossip, global communication, wormhole routing, mesh networks, torus networks.

## 1 Introduction

### 1.1 Meshes and Tori

One of the most thoroughly investigated interconnection schemes for parallel computers is the $n \times n$ *mesh*, in which $n^2$ processing units (**PUs**) are connected by a two-dimensional grid of communication links. A *torus* is a mesh with wrap-around connections. Their immediate generalizations are $d$-dimensional $n \times \cdots \times n$ meshes and tori. Although these networks have a large diameter in comparison to the various hypercubic networks, they are nevertheless of great importance due to their simple structure and efficient layout. Numerous parallel machines with mesh and torus topologies have been built, and various algorithmic problems have been analyzed on theoretical models of the mesh.

### 1.2 Wormhole Routing

Traditionally, algorithms for the mesh have been developed using a store-and-forward routing model in which a packet is treated as an atomic unit that can be transferred between two adjacent PUs in unit time. However, many modern parallel architectures employ *wormhole routing* instead. Briefly, in this model, a packet consists of a number of atomic data units called *flits* which are routed through the network in a pipelined fashion. As long as there is no congestion in the network, the time to send a packet consisting of $l$ flits between two arbitrary PUs is well approximated by $t_s + l \cdot t_l$, where $t_s$ is the **start-up** time (the time needed to initiate the message transmission) and $t_l$ is the **flit-transfer** time (the time required for actually transferring the data). Usually, $t_s \gg t_l$ so that it is important to minimize the number of startups when the packet size is small, whereas it is important to minimize the transmission time when the packet size is large. These two goals may conflict, and then trade-offs must be made.

### 1.3 Gossiping

Collective communication operations occur frequently in parallel computing, and their performance often determines the overall running time of an application. One of the fundamental communication problems is *gossiping* (also called total exchange or all-to-all nonpersonalized communication). Gossiping is the problem in which every PU wants to send the same packet to every other PU. Said differently, initially, each of the $N$ PUs contains an amount of data of size $L$ and, finally, all PUs know the complete data of size $N \cdot L$. This is a very communication intensive operation. On a $d$-dimensional store-and-forward mesh, it can be performed trivially in $N/d$ steps, but for wormhole-routed meshes, it is less obvious how to organize the routing so that the total cost is minimal.

Gossiping appears as a subroutine in many important problems in parallel computation. We just mention two of them. If $M$ keys need to be sorted on $N$ PUs ($M \gg N$), then a good approach is to select a set of $m$ splitters [15], [14], [8] which must be made available in all PUs. This means that we have to perform a gossip in which every PU contributes $m/N$ keys. In this case, the cost of gossiping (provided it is performed efficiently) will not dominate the overall sorting time when the input size is large, because the splitters constitute only a small fraction of the data. A second application of gossiping appears in algorithms for solving ordinary differential equations using parallel block predictor-corrector methods [16]. In each application of the method, block point computations corresponding to the prediction are carried out by different PUs, and these values are needed by all PUs for the correction phase, requiring a gossiping of the data.

- *B.H.H. Juurlink is with the Heinz Nixdorf Institute, Paderborn University, Fürstenallee 11, 33102 Paderborn, Germany. E-mail: benj@uni-paderborn.de.*
- *J.F. Sibeyn is with the Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. E-mail: jopsi@mpi-sb.mpg.de.*
- *P.S. Rao is with Dow Jones Markets. E-mail: srini@tts.dowjones.com.*

*Manuscript received 6 Dec. 1996; revised 15 Aug. 1997.*
*For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 100355.*

### 1.4 Previous Work

A substantial amount of research has been performed on finding efficient algorithms for collective communication operations on wormhole-routed systems (see, e.g., [1], [4], [13], [3], [18]). However, most papers either deal with very small packets or with very large packets. Both these extreme cases require algorithms optimizing only one parameter.

If the packets are small, then the number of start-ups should be minimized. Peters and Syska [13] considered the broadcasting problem on two-dimensional tori and showed that it can be performed in the optimal $2 \cdot \lceil \log_5 n \rceil$ steps. Their ideas have been generalized to three-dimensional tori in [3]. The algorithms described in these papers can be adapted for the gossiping problem by first concentrating all data into one PU and then performing a broadcast. However, such an approach leads to a prohibitively large transmission time. Another drawback of both approaches is that it is assumed that the routing paths may be selected by the algorithm. The algorithms presented in this paper can also be used if the network only supports dimension-ordered routing.

If the packets are large, a store-and-forward approach yields the best results. As mentioned before, on a $d$-dimensional $n \times \cdots \times n$ mesh it can be performed trivially in $n^d/d$ packet steps. Gossiping in a store-and-forward hypercube model was studied in [9].

There are many other papers on collective communication operations on wormhole-routed meshes and tori. Although these papers do not deal with the same problem, there are some similarities. For example, Sundar et al. [17] propose a hybrid algorithm for performing *personalized* all-to-all communication (complete exchange) on wormhole-routed meshes. Briefly, they employ a logarithmic step algorithm until the packet size becomes large, at which point they switch to a linear step algorithm.

### 1.5 Our Results

In this paper, we focus on the trade-off between the start-up time and the transmission time. This is useful because there is a large range of mesh sizes, packet sizes, and start-up costs in which neither of the two contributions is negligible. We would like to emphasize that we are not proposing a hybrid algorithm that simply uses the fastest of the gather/broadcast approach and the store-and-forward approach. In an intermediate range of packet sizes, our algorithm is asymptotically better than the *best* of the two extreme approaches.

A nontrivial lower bound shows that our algorithms are close to optimal for all possible values of the parameters involved. For the efficiency of the two-dimensional algorithm, it is essential that data is concentrated in PUs that lie on diagonals. For higher-dimensional meshes we give an interesting generalization of the notion of a diagonal which may be of independent interest. We remark that Tseng et al. [18] also used diagonals in their complete exchange algorithm. However, the generalization of a diagonal given there for three-dimensional tori is rather straightforward. Hyperspaces are used that, when projected, give back a diagonal in two-dimensional space. We generalize the diagonal in a different way, that gives better performance, and which allows us to formulate a generic algorithm that works for arbitrary dimensions (not only dimension three) without problem.

We also compare the value of several strategies by substituting parameters in the formulas for their time consumptions. Furthermore, our theoretical results for two-dimensional meshes are completed with measurements of an implementation on the Intel Paragon. The assumed and the real hardware model do not completely coincide, but still we believe that these measurements support our claims in most important points.

### 1.6 Contents

This paper is organized as follows. In the next section, the model of computation is described. Thereupon, in Section 3, we present several lower bounds for the gossiping problem. The problem of gossiping on a one-dimensional torus (circular array) is analyzed in Section 4. After that, in Section 5 and Section 6, we extend the algorithm to two- and higher-dimensional meshes and tori. Finally, in Section 7, experimental results gathered on the Intel Paragon are presented.

## 2 Model of Computation

A $d$-dimensional **mesh** consists of $N = n^d$ processing units (**PUs**) laid out in a $d$-dimensional grid of side length $n$. Every PU is connected to each of its (at most) $2 \cdot d$ immediate neighbors by a bidirectional communication link. A torus is a mesh with wrap-around connections. We concentrate on the communication complexity and assume that a PU can perform an unbounded amount of internal computation in a step. It is also assumed that a PU can simultaneously send and receive data over all its connections. This is sometimes called the full-port or all-port model. With minor modifications, the presented algorithms can also be implemented on one-port architectures.

For the communication, we assume the much considered wormhole routing model (see [6], [12], [5] for some recent surveys). In this model, a packet consists of a number of atomic data units called **flits**. During routing, the header flit governs the route of the packet and the other flits follow it in a pipelined fashion. Initially, all flits reside in the source PU and, finally, all flits should reside in the destination PU. At intermediate stages, all flits of a packet reside in adjacent PUs. The packets should be "expanded" and "contracted" only once. That is, two or more flits should reside in the same PU only at the source and destination PU. Wormhole routing is likely to produce deadlock unless special care is taken.

The reasons to consider wormhole routing instead of the more traditional store-and-forward routing are of a practical nature. On modern MIMD computers (such as the Intel Paragon and the Cray T3D), the time to initiate a packet transmission is considerably larger than the time needed to traverse a connection. Wormhole routing has been developed in response to this fact. The time for sending a packet consisting of $l$ flits over a distance of $d$ connections is given by

$$t(d, l) = t_s + d \cdot t_d + l \cdot t_l. \tag{1}$$

We refer to $t_s$ as the **start-up time**, $t_d$ as the **hop time**, and $t_l$ as the **flit-transfer time**.

Equation (1) is only correct if there is no link contention (in other words, as long as the paths of the packets do not overlap). If paths of various packets overlap, then the transfer time increases. All our algorithms are overlap-free.

## 3 Lower Bounds

We start with a trivial but general lower bound. Thereupon, we give a more detailed analysis, proving a stronger lower bound for special cases.

LEMMA 1. *In any network with N PUs, degree* $\Delta$*, and diameter D, the time* $T_{con}(N, \Delta, D)$ *needed to concentrate all information in a single PU satisfies*:

$$T_{con}(N, \Delta, D) \geq \max\{(N/\Delta) \cdot l \cdot t_l, (D/2) \cdot t_d, \log_{\Delta+1} N \cdot t_s\}.$$

PROOF. The terms are motivated as follows: $N \cdot l$ flits have to be transferred over at most $\Delta$ connections to the PU in which the data is concentrated; one packet must travel over a distance of at least $(D/2)$ to reach the concentration PU; after $t$ steps, a PU can hold at most $(\Delta + 1)^t$ data items by induction. □

Of course, a lower bound for $T_{con}$ immediately implies the same lower bound for the gossiping problem. The degree of a $d$-dimensional $n \times \cdots \times n$ mesh is $2 \cdot d$, and the diameter equals $d \cdot (n - 1)$.

Usually, $t_d$ is comparable to $t_l$, while $D < (N/\Delta) \cdot l$. Thus, we can omit the term $(D/2) \cdot t_d$ from the lower bound without sacrificing too much accuracy. By dividing both remaining terms by $l \cdot t_l$, and by setting $T' = T/(l \cdot t_l)$ and $r = t_s/(l \cdot t_l)$, we obtain the following simplified lower bound for concentrating all data in one PU:

$$T'_{con}(N, \Delta) \geq \max\left\{N/\Delta, \left(\log N/\log(\Delta + 1)\right) \cdot r\right\}. \quad (2)$$

In Section 4, gossiping algorithms are presented that match this lower bound up to constant factors for all $r \leq n^{1-\epsilon}$, $\epsilon > 0$, as well as for all $r = \Omega(n)$. For the intermediate range, there is a considerable deviation from (2). Therefore, these values of $r$ are considered in more detail. Let $T'_{\text{gos}}(n)$ denote the number of time units (of duration $l \cdot t_l$ each) required for gossiping on a circular array with $n$ PUs.

THEOREM 1. *Let* $r = t_s/(l \cdot t_l) \leq (n - 1)/e$*, then*

$$T'_{gos}(n) = \Omega\left(n \cdot \ln n/\ln(n/r)\right).$$

Theorem 1 will be proven by two lemmas. Notice that it establishes a smooth transition from the range of small $r$ values ($r \leq n^{1-\epsilon}$, $\epsilon > 0$) to the range of large $r$ values ($r = \Omega(n)$), for which (2) already gives sharp results.

First, we show that for proving lower bounds, one can concentrate on the *dissemination problem*: the problem of broadcasting the information that is concentrated in one PU to all other PUs. The number of time units required for this problem is denoted by $T'_{\text{dis}}$.

LEMMA 2.

$$T'_{dis} - T'_{con} \leq T'_{gos} \leq T'_{dis} + T'_{con}.$$

PROOF. Starting with all data concentrated in a single PU, the initial situation can be established in time $T'_{con}$ by reversing a concentration. On the other hand, gossiping can be performed by concentrating and subsequently disseminating. □

As in our case, we will prove a dissemination time that is of larger order than the concentration time (e.g., for $r = n/\log n$, $T'_{\text{con}}(n) = \mathcal{O}(n)$), we have $T'_{\text{dis}} = \Theta\left(T'_{\text{gos}}\right)$.

For the dissemination problem with certain $r$, it is easy to see that having full freedom of choosing the size of the packets can be at most a factor two cheaper than when the data are bundled into fixed messages of size $r$. That is, we may focus on the problem of disseminating $n/r$ messages, residing in PU 0, while sending a message takes $2 \cdot r \cdot l \cdot t_l = 2 \cdot t_s$ time. At most another constant factor difference is introduced if we assume that dissemination has to be performed on a circular array with only rightward connections. By the above argumentation, the proof of Theorem 1 is completed by

LEMMA 3. *Consider a circular array with only rightward connections and with n PUs. Initially, PU* 0 *contains n/r messages of size r. In one step, the messages may be sent rightwards arbitrarily far, but the paths of the messages should be disjoint. If* $r \leq (n - 1)/e$*, then dissemination takes at least* $n/r \cdot \ln n/\ln(n/r)$ *steps.*

PROOF. We speak of the original $n/r$ messages as *colors*, and the task is to make all colors available in all PUs. We define a cost function $F(t)$ for the distribution of colors after $t$ steps. Consider a PU $i$ and a color $c$, and let $j$ be the rightmost PU to the left of $i$ holding color $c$. The contribution of PU $i$ to $F(t)$ by color $c$ is $\ln(i - j)$ if PU $i$ does not contain color $c$, and 0 otherwise. The initial cost is given by

$$F(0) = n/r \cdot \sum_{i=1}^{n-1} \ln i \simeq n^2/r \cdot \ln n.$$

We consider how much the cost function can be reduced after a step is performed. It is essential that the paths must be disjoint. One large "jump" by a message of some color $c$ gives a strong reduction of the contribution by color $c$, but the following claim shows that the total reduction is at most $n \cdot \ln(n/r)$.

CLAIM 1. *If* $r \leq (n - 1)/e$*, then, after one step, the cost function is reduced by at most* $n \cdot \ln(n/r)$*. Moreover, this occurs if we make a jump over distance r with one message from each color.*

From this, the result of the lemma follows, because then the number of steps required for dissemination is at least

$$\frac{F(0)}{\Delta F} \geq \frac{n^2/r \cdot \ln n}{n \cdot \ln(n/r)} = \frac{n \cdot \ln n}{r \cdot \ln(n/r)}.$$

In order to prove the claim, let $d_c$ be the maximum jump made by a message of color $c$, $0 \leq c \leq n/r - 1$. Obviously, we must have $\sum_c d_c \leq n$, since the paths of the messages must be disjoint. The reduction of the contribution to $F(t)$ by color $c$ is at most

$$\sum_{i=n-d_c}^{n-1} \ln i - \sum_{i=1}^{d_c-1} \ln i \simeq d_c \cdot \ln n - d_c \cdot \ln d_c = d_c \cdot \ln(n/d_c). \quad (3)$$

This can be seen as follows. The initial contribution by color $c$ is at most $\sum_{i=1}^{n} \ln i$. After a step over distance $d_c$, the contribution of the PUs which are within distance $d_c$ remains unchanged. This contribution is $\sum_{i=1}^{d_c-1} \ln i$. Furthermore, the contribution of the other PUs becomes $\sum_{i=d_c+1}^{n-1} \ln(i-d_c)$. The reduction due to the step made by color $c$ is, therefore, at most

$$\sum_{i=1}^{n} \ln i - \sum_{i=d_c+1}^{n-1} \ln(i-d_c) - \sum_{i=1}^{d_c-1} \ln i = \sum_{i=n-d_c}^{n-1} \ln i - \sum_{i=1}^{d_c-1} \ln i.$$

From (3), it follows that the total reduction (due to all steps made by all colors) is bounded by

$$\Delta F \le \sum_{c=0}^{n/r-1} d_c \cdot \ln(n/d_c).$$

It needs to be shown that this expression is at most $n \cdot \ln(n/r)$. "Powering," we obtain

$$e^{\Delta F} \le e^{\sum d_c \cdot \ln(n/d_c)} = \left(\frac{n}{d_0}\right)^{d_0} \cdot \left(\frac{n}{d_1}\right)^{d_1} \cdots \left(\frac{n}{d_{n/r-1}}\right)^{d_{n/r-1}} = \frac{n^{\sum d_c}}{{d_0}^{d_0} \cdot {d_1}^{d_1} \cdots {d_{n/r-1}}^{d_{n/r-1}}}.$$

The Lagrange multiplicator theorem (see, e.g., [11, Section 4.3]) gives that the product of factors with a fixed sum is maximal if all factors are equal. Therefore,

$${d_0}^{d_0} \cdot {d_1}^{d_1} \cdots {d_{n/r-1}}^{d_{n/r-1}} \ge \left(\frac{\sum d_c}{n/r}\right)^{\sum d_c}.$$

It follows that

$$\frac{n^{\sum d_c}}{{d_0}^{d_0} \cdot {d_1}^{d_1} \cdots {d_{n/r-1}}^{d_{n/r-1}}} \le \left(\frac{n^2}{r \cdot \sum d_c}\right)^{\sum d_c}.$$

Let $a_k = \left(\frac{n^2}{r \cdot k}\right)^k$ for $k = 1, 2, \ldots, n$. We need to show that $a_k$ is maximal if $k$ is fixed at its maximum legal value, which is $n$. Consider the ratio $a_{k+1}/a_k$. We have

$$\frac{a_{k+1}}{a_k} = \frac{n^2 \cdot k^k}{r \cdot (k+1)^{k+1}}.$$

It needs to be shown that $a_{k+1}/a_k > 1$. Hence, we must have

$$\frac{n^2 \cdot k^k}{r \cdot (k+1)^{k+1}} > 1$$
$$\Leftrightarrow \frac{(k+1)^{k+1}}{k^k} < \frac{n^2}{r}$$
$$\Leftrightarrow \left(1+\frac{1}{k}\right)^k \cdot (k+1) < \frac{n^2}{r}$$
$$\Leftarrow e \cdot (n+1) < \frac{n^2}{r},$$

which holds because $r \le (n-1)/e$. It follows that the total reduction in cost is at most

$$\ln(e^{\Delta F}) \le \ln((n/r)^n) = n \cdot \ln(n/r). \quad \square$$

## 4 Linear and Circular Arrays

In this section, we analyze gossiping on one-dimensional processor arrays. It is assumed that the time for routing a packet is given by (1) as long as the paths of the packets do not overlap. As in the previous section, the distance term, which is of minor importance anyway, is neglected in the rest of this paper. Furthermore, we write $r = t_s/(l \cdot t_l)$ and express the time needed for gossiping in units of duration $l \cdot t_l$. We only present algorithms for circular arrays. Due to their more regular structure, these are slightly "cleaner," but with minor modifications all results carry on for linear arrays.

### 4.1 Basic Approaches

For gossiping on a circular array with $n$ PUs, there are two trivial approaches. Each of them is good in an extreme case.

1) Every PU sends a packet containing its data to the left and right. The packets are sent on for $\lfloor n/2 \rfloor$ steps.
2) Recursively concentrate the data packets into a selected PU. After that, disseminate the information to all other PUs by reversing the process.

Let $T_1'(n,r)$ and $T_2'(n,r)$ denote the time taken by Approach 1 and Approach 2, respectively. A simple analysis gives

Lemma 4.

$$T_1'(n,r) = \lfloor n/2 \rfloor \cdot (1+r),$$
$$T_2'(n,r) \simeq \log_3 n \cdot (n + 2 \cdot r).$$

Proof. Approach 1 consists of $\lfloor n/2 \rfloor$ steps and in each step, every packet consists of $l$ flits.

The time taken by Approach 2 is determined as follows. During the concentration phase, the packets get three times as heavy in every step:

$$T_{\text{conc}}(n,r) = \sum_{i=0}^{\log_3 n-1} \left(3^i + r\right) < n/2 + \log_3 n \cdot r.$$

The expression for the dissemination phase is similar, but in this case the packets consist of $n \cdot l$ flits in every step:

$$T_{\text{dis}}(n,r) = \sum_{i=0}^{\log_3 n-1} (n+r) = \log_3 n \cdot (n+r).$$

Adding the two contributions and neglecting the lower-order term $n/2$ gives the stated result. $\square$

Approach 1 is good when $r$ is small. Comparing it with the lower bound given in (2) shows that it is exactly optimal when $r = 0$. When $r$ goes to infinity, Approach 2 becomes optimal to within a constant factor. It will outperform Approach 1 for many practical values of $r$. Still, in principle, the time consumption of Approach 2 is not even linear in $n$.

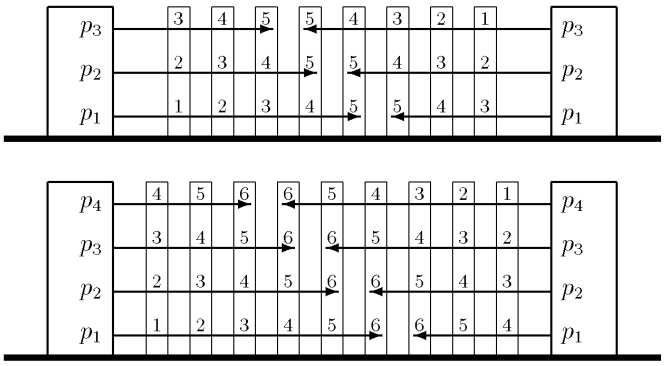



Fig. 1. Behavior of one round of Phase 3 of Algorithm CIRCGOS. Every arrow is labeled with the time steps at which the corresponding packet reaches the PUs. The top figure illustrates the case $a = 9$ and $b = 5$. In this case, three packets are routed from the bridgeheads. The bottom figure illustrates the case $a = 10$ and $b = 6$. In this case, the data is partitioned into four packets.

### 4.2 Intermixed Approach

For $r = \log n$, both approaches require $\Theta(n \cdot \log n)$ time units. This is a factor of $\log n$ more than given by the lower bound. For this intermediate range of $r$-values, we present an algorithm that establishes a better trade-off between the start-up and the transfer time. The algorithm consists of three phases and works with parameters $a$ and $b$:

Algorithm CIRCGOS($n$, $a$, $b$)

1) Concentrate $n/a$ packets in $a$ evenly interspaced PUs, called **bridgeheads**.
2) For $\lfloor a/2 \rfloor$ steps, send the packets of size $n/a$ among the bridgeheads in both directions such that, afterwards, every bridgehead contains the complete data.
3) In $\lceil \log_a n - 1 \rceil$ rounds, repeatedly increase the number of bridgeheads by a factor of $a$. This will be done as follows. Let $b \geq \lfloor a/2 \rfloor$ denote the number of steps allowed in one round, and let $k = 2 \cdot b - a + 2$. Every bridgehead partitions the data into $k$ packets $p_1, \ldots, p_k$ of size $n/k$ each. Thereupon, the packets are broadcast to the new bridgeheads in a pipelined fashion. The packets to the right are sent in order, whereas the packets to the left are sent in reverse order.

In Phase 2, the packets are circulated around. The description is pleasant because of the circular structure. In Phase 3, two oppositely directed packet streams are sustained between the bridgeheads. In order to fully exploit the bidirectional communication links, a bridgehead should not send the same packets to the left and right. Rightwards the packets should be sent in order, whereas leftwards they should be sent in reverse order. Fig. 1 shows two examples.

The total time consumption of algorithm CIRCGOS is given in the following lemma. We do not consider all rounding details.

LEMMA 5. *Let $T'_{cg,f}$ denote the time taken by Phase $f$ ($f$ = 1, 2, 3) of* CIRCGOS($n$, $a$, $b$). *Then,*

$$T'_{cg,1} = n/(2 \cdot a) + \log_3(n/a) \cdot r,$$
$$T'_{cg,2} = \lfloor a/2 \rfloor \cdot (n/a + r),$$
$$T'_{cg,3} = (\log_a n - 1) \cdot b \cdot (n/(2 \cdot b - a + 2) + r).$$

PROOF. Phase 1 corresponds to a concentration step on linear arrays of size $n/a$ instead of $n$. The time needed for Phase 2 follows by multiplying the number of steps by the time taken by each of them. In order to prove the time consumption of Phase 3, it needs to be shown that after $b$ steps every new bridgehead contains the complete data. Consider a new bridgehead and assume it is at distance $d \leq \lfloor a/2 \rfloor$ to the closest old bridgehead. This new bridgehead receives the first packet after $d$ steps. After that, it receives one more packet in Step $d + 1$ through Step $a - d - 1$. From that point onwards, it receives two packets in every step. After Step $a - d + x$, it contains $a + 2 \cdot (x - d) + 2$ packets. By setting $b = a - d + x$, it follows that after $b$ steps the new bridgehead contains $a + 2 \cdot (b - a) + 2 = 2 \cdot b - a + 2$ packets, as required. □

At first glance, it is not clear what the result of Lemma 5 means. In particular, it is not immediately clear which $a$ and $b$ should be chosen. In order to obtain an impression, we have written a small program which searches for the optimal values. Table 1 lists some typical results. From these results we conclude that

- For realistic values of $n$ and $r$, CIRCGOS may be several times faster than the best of Approach 1 and Approach 2. Furthermore, it never performs worse.
- The range of $r$ values for which algorithm CIRCGOS is the fastest increases with $n$.
- The best choices for $a$ and $b$ increase with $n$ and decrease with $r$. In this range of $n$ and $r$, $b$ is approximately given by $b = \lfloor 1.1 \cdot (a - 2) \rfloor$.

Notice that when $a = n$, CIRCGOS is identical to Approach 1. Furthermore, CIRCGOS($n$, 3, 1) behaves almost identically to Approach 2, but after $\log_3 n$ steps the three bridgeheads contain the complete data. The dissemination phase therefore requires one routing step fewer than in Approach 2. This explains why CIRCGOS($n$, 3, 1) is always faster than Approach 2, which does not profit from the wrap-around connections.

Although the exact choice for the parameters is essential for obtaining the best performance (as shown in Table 1), the *asymptotic* analysis remains unchanged if we take $b = a$. The reason for this is that using different parameters can reduce the amount of transferred data by at most a factor of two. For proving asymptotic results this is fine, but for practical applications this is highly undesirable. On the other hand, we might have used more parameters: The factor $a$ by which the number of bridgeheads is increased in every round of Phase 3 might have been chosen differently, together with its corresponding optimal choice of $b$.

THEOREM 2. *Let $r < n$. The number of time units needed by* CIRCGOS($n$, $n/r$, $n/r$) *is given by*

$$T'_{gos}(n, r) = \mathcal{O}\left(n \cdot \ln n / \ln(n/r)\right).$$

PROOF. From Lemma 5, it follows that

$$T'_{gos}(n, r) = \mathcal{O}\left(n + r \cdot \log r + n \cdot \log_{n/r} r\right)$$
$$= \mathcal{O}\left(n + r \cdot \ln r + n \cdot \ln r / \ln(n/r)\right).$$

TABLE 1
COMPARISON BETWEEN THE TIME TAKEN BY APPROACH 1 (TOP), APPROACH 2 (MIDDLE), AND CIRCGOS (BOTTOM)

| $n \setminus r$ | 2 | | 10 | | 50 | | 250 | |
|---|---|---|---|---|---|---|---|---|
| | 39 | | 143 | | 663 | | 3,263 | |
| 27 | 92 | | 140 | | 380 | | 1,580 | |
| | 39 | (27, -) | 115 | (8, 5) | 347 | (3, 1) | 1,447 | (3, 1) |
| | 120 | | 440 | | 2,040 | | 10,040 | |
| 81 | 339 | | 403 | | 723 | | 2,323 | |
| | 113 | (37, 36) | 260 | (11, 9) | 672 | (3, 1) | 2,172 | (3, 1) |
| | 363 | | 1,331 | | 6,171 | | 30,371 | |
| 243 | 1,234 | | 1,314 | | 1,714 | | 3,714 | |
| | 304 | (68, 78) | 601 | (19, 19) | 1,374 | (7, 5) | 3,509 | (3, 1) |
| | 1,092 | | 4,004 | | 18,564 | | 91,364 | |
| 729 | 4,397 | | 4,493 | | 4,973 | | 7,373 | |
| | 828 | (138, 179) | 1,449 | (36, 42) | 2,895 | (12, 11) | 6,755 | (4, 2) |

*The values of the parameters a and b for which the result for* CIRCGOS *was obtained are given behind its time consumption. The cost unit is* $l \cdot t_l$.

When $r = n/x$, where $x > 1$, the second term never dominates because $r = n/x \le n/\ln x = n/\ln(n/r)$. Replacing the factor of $\ln r$ in the third term by $\ln n$ gives the theorem. □

Thus, Algorithm CIRCGOS gives a continuous transition from gossiping times $\mathcal{O}(n)$ (as achieved by Approach 1 when $r = \mathcal{O}(1)$) to gossiping times $\mathcal{O}(n \cdot \log n)$ (as achieved by Approach 2 when $r = n$). For intermediate $r$ values, CIRCGOS may be substantially faster:

COROLLARY 1. *Algorithm* CIRCGOS *is asymptotically optimal for all values of r. For all* $\log n \le r \le n^{1-\epsilon}$, $\epsilon > 0$, CIRCGOS *is* $\Omega(\log n)$ *times faster than Approach 1 and Approach 2.*

PROOF. The optimality claim follows by comparing the result of Theorem 2 with the lower bound given in Theorem 1. For $r \ge n$, optimality was already established before.

For $r \ge \log n$, Approach 1 and Approach 2 both take $\Omega(n \cdot \log n)$ time units. On the other hand, when $r \le n^{1-\epsilon}$, CIRCGOS has a time consumption of at most $\mathcal{O}(n \cdot \log n/(1-\epsilon) \cdot \log n) = \mathcal{O}(n)$. □

### 4.3 Generalization

In the previous section, we presented a gossiping algorithm for linear and circular arrays which is optimal to within a constant factor for all values of $r$. In this section, we show that this immediately implies an asymptotically optimal algorithm for two- and three-dimensional meshes and tori. In fact, it immediately gives an asymptotically optimal algorithm for $d$-dimensional meshes, as long as $d$ is constant. The reason to develop gossiping algorithms for two- and higher-dimensional meshes (as will be done in subsequent sections) is, therefore, to obtain algorithms with good practical behavior, paying attention to the constants.

The algorithm for gossiping on a $d$-dimensional mesh consists of $d$ phases. In Phase $f$, $0 \le f \le d-1$, the packets participate in a gossip along axis $f$. For each of these one-dimensional gossips, the most efficient algorithm is taken. As the size of the packets increases in each phase (in Phase $f$, the packets consist of $l \cdot n^f/d$ flits each), this is not necessarily the same algorithm in all phases. The described algorithm will be denoted by HIGH-DIM-GOS.

THEOREM 3. *For constant d,* HIGH-DIM-GOS *has asymptotically optimal performance.*

PROOF. Denote the time taken by Phase $f$ of HIGH-DIM-GOS by $T_{\mathrm{hdg},f}$, and the time taken by the optimal gossiping algorithm by $T_{\mathrm{opt}}$. Clearly, $T_{\mathrm{opt}}$ exceeds the time required for making all information available in all PUs, starting with the situation at the beginning of Phase $d-1$. In Phase $d-1$, only a fraction of $1/d$ of the connections is used. Using all connections would make the algorithm faster by at most a factor of $d$. Thus, $T_{\mathrm{hdg},d-1} \le d \cdot T_{\mathrm{opt}}$. Since $T_{\mathrm{hdg},f} \le T_{\mathrm{hdg},d-1}$ for all $0 \le f < d-1$, it follows that $\sum_f T_{\mathrm{hdg},f} \le d^2 \cdot T_{\mathrm{opt}}$. □

Thus, once again, we would like to emphasize that achieving asymptotically optimal performance is not the real issue, but constructing algorithms with good practical behavior.

Algorithm HIGH-DIM-GOS does not take advantage of the all port communication capability. To do better, the $l$ flits in each PU are colored with $d$ colors: Flit $\lfloor c \cdot l/d \rfloor$ to Flit $\lfloor (c+1) \cdot l/d - 1 \rfloor$ are given Color $c$, $0 \le c < d$. After that, we perform $d$ independent gossiping operations with parameter $r' = t_s/(l' \cdot t_l)$, where $l' = \lceil l/d \rceil$. In Phase $f$, $0 \le f \le d-1$, the packets with Color $c$ participate in an operation along axis $(f + c) \bmod d$. This algorithm will be denoted by HIGH-DIM-GOS′. It has the same start-up time as HIGH-DIM-GOS, but the transfer time is reduced by a factor of $d$.

## 5 TWO-DIMENSIONAL ARRAYS

In this section, we analyze the gossiping problem on two-dimensional $n \times n$ tori. First, we investigate what can be obtained by overlapping two one-dimensional gossiping algorithms, one along the rows and one along the columns, as sketched in Section 4.3. After that, a truly two-dimensional algorithm is presented which, for some values of $n$ and $r$, performs significantly better.

### 5.1 Basic Approaches

The simplest idea is to apply HIGH-DIM-GOS′ with a choice from the presented one-dimensional gossiping algorithms in each phase. Let Approach *i-j* denote the algorithm in

TABLE 2
COMPARISON OF THE RESULTS OBTAINED FOR GOSSIPING ON AN $n \times n$ TORUS

| $n \setminus r$ | 10 | | 50 | | 200 | | 500 | |
|---|---|---|---|---|---|---|---|---|
| | 442 | | 1,482 | | 5,382 | | 13,182 | |
| | 409 | | 1,189 | | 4,114 | | 9,964 | |
| 27 | 1,153 | | 1,733 | | 3,533 | | 7,133 | |
| | 415 | (27, 13) | 1,123 | (14, 11) | 3,164 | (4, 2) | 6,508 | (3, 1) |
| | 409 | (9, 8) | 929 | (4, 2) | 2,458 | (3, 1) | 5,458 | (3, 1) |
| | 2,440 | | 5,640 | | 17,640 | | 81,640 | |
| | 2264 | | 4,204 | | 11,479 | | 50,279 | |
| 81 | 13,443 | | 14,083 | | 16,483 | | 29,283 | |
| | 2,260 | (81, 40) | 4,157 | (88, 88) | 8,826 | (18, 16) | 25,507 | (6, 4) |
| | 2,232 | (23, 24) | 3,375 | (10, 8) | 8,826 | (18, 16) | 17,607 | (3, 1) |
| | 17,182 | | 26,862 | | 63,162 | | 135,762 | |
| | 16,621 | | 21,881 | | 41,606 | | 81,056 | |
| 243 | 148,429 | | 149,229 | | 152,229 | | 158,229 | |
| | 16,459 | (243, 127) | 21,917 | (243, 121) | 36,776 | (82, 94) | 54,689 | (37, 40) |
| | 16,391 | (63, 78) | 19,190 | (27, 27) | 25,586 | (13, 11) | 34,858 | (8, 6) |

*For every pair of values (n, r), the number of time steps is given (from top to bottom) for Approach 1-1, Approach 2-1, Approach 2-2,* HIGH-DIM-GOS′ *that* CIRCGOS, *and* TORGOS. *For the latter two, the values of (the second) a and b for which the result was obtained are indicated.*

which first Approach $i$ is applied, and then Approach $j$. Approach 1-2 can be excluded, since it will never outperform the best of the other approaches. Let $T'_{i,j}$ denote the number of time steps taken by Approach *i-j*. Using the results from Section 4, we find

LEMMA 6.

$$T'_{1,1} \simeq \lfloor n/2 \rfloor \cdot (n/2 + 1/2 + 2 \cdot r),$$
$$T'_{2,1} \simeq \lfloor n/2 \rfloor \cdot (n/2 + \log_3 n) + (n/2 + 2 \cdot \log_3 n) \cdot r,$$
$$T'_{2,2} \simeq \log_3 n \cdot (n^2/2 + n/2 + 4 \cdot r).$$

The time consumption for applying the best version of CIRCGOS in both phases cannot be fitted in a simple formula, but it is better than the best of the above algorithms by almost the same factors as those found before. In Table 2, some numerical results are given. Because the packets have size $n \cdot l$ during Phase 2 (which dominates the total time consumption), the transition between the various algorithm now occurs for much larger $r$ than in Table 1.

### 5.2 Intermixed Approach

In this section, we present a two-dimensional analogue of algorithm CIRCGOS. The algorithm as described below does not use the horizontal and vertical connections simultaneously. Such an algorithm is called **uniaxial**. By applying the coloring technique of HIGH-DIM-GOS′, the transfer time is halved.

The algorithm first creates a situation comparable to the one we find after Phase 2 of CIRCGOS. For this, three routing phases are required:

Algorithm TORGOS($n$, $a$, $b$)

1) Each $PU_{i,j}$ where $(j - i) \bmod (n/a) = 0$ is designated as a bridgehead. Note that there are $a$ bridgeheads in every row. In each bridgehead, concentrate $n/a$ packets from its row.
2) For $\lfloor a/2 \rfloor$ steps, send packets of size $n/a$ along the rows among the bridgeheads in both directions, such that afterwards, every bridgehead contains the complete data of its row.
3) For $\lfloor a/2 \rfloor$ steps, send packets of size $n$ along the columns among the bridgeheads in both directions, such that afterwards, every bridgehead contains $a \cdot n$ data.

Now, each bridgehead in Row $i$, $0 \le i < n$, holds all data from every Row $i'$ where $(i - i') \bmod (n/a) = 0$. This is the result of the diagonal way in which the bridgeheads were chosen. An example is given in Fig. 2. Thus, all data are available on the diagonals of every $n/a \times n/a$ submesh. The algorithm proceeds in $\log_a n - 1$ rounds. In each round, the number of bridgeheads is increased by a factor of $a$.

INVARIANT 1. *At the beginning of Round t, $1 \le t \le \log_a n$, every bridgehead holds $n \cdot a^t$ data, and all data are available on the diagonals of all $n/a^t \times n/a^t$ submeshes.*

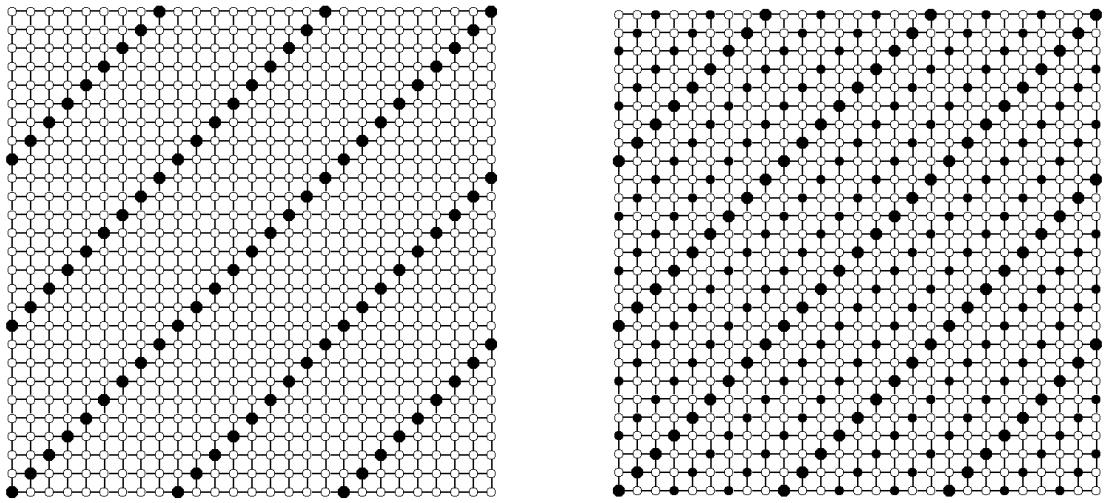
This implies that the gossiping has been completed when $t = \log_a n$. A more formal description of the last phase is given below:

Algorithm TORGOS($n$, $a$, $b$) (continued)

4) For $t = 1, \ldots, \lceil \log_a n - 1 \rceil$, repeatedly increase the number of bridgeheads by a factor of $a$ by inserting $a - 1$ new bridgeheads between any pair of two consecutive bridgeheads in every row.
   a) The information from the old bridgeheads in a row is passed to the $a - 1$ new bridgeheads in $b \ge \lfloor a/2 \rfloor$ steps with packets of size $m/(2 \cdot b - a + 2)$, where $m = n \cdot a^t$.
   b) For $\lfloor a/2 \rfloor$ steps, send packets of size $m$ along the columns among the bridgeheads (old as well as new) in both directions, so that afterwards, every bridgehead contains $a \cdot m$ data.

The three phases that operate along the rows are identical to those of CIRCGOS. Only Phase 3 and Phase 4b, which add the information of a row to $a$ other rows, are new. The following analogue of Lemma 5 is straightforward:

LEMMA 7. *Let $T'_{tg,f}$ denote the number of time units needed for Phase f of* TORGOS($n$, $a$, $b$). *Then,*



(a) (b)

Fig. 2. The bridgeheads in a two-dimensional torus for the case $n = 27$ and $a = 3$. (a) After Phase 3, each bridgehead in Row $i$, $0 \le i < 27$, knows all data from every Row $i'$, where $(i - i') \bmod 9 = 0$. The bridgeheads from nine consecutive rows, therefore, know the complete data. (b) The bridgeheads (new ones are drawn smaller) during the first round of Phase 4. Hereafter, each bridgehead in Row $i$ knows all data from every Row $i'$, where $(i - i') \bmod 3 = 0$.

$$T'_{tg,1} = n/(2 \cdot a) + \log_3(n/a) \cdot r,$$

$$T'_{tg,4 \cdot a} = \frac{b \cdot n^2}{(a-1) \cdot (2 \cdot b - a + 2)} + (\log_a n - 1) \cdot b \cdot r,$$

$$T'_{tg,2+3+4 \cdot b} = \lfloor a/2 \rfloor \cdot \left(n^2/(a-1) + (\log_a n + 1) \cdot r\right).$$

PROOF. The time taken by Phase 1 is given in the proof of Lemma 5. In Phase 2, 3, and 4b, there are $\log_a n + 1$ rounds in total, and each round consists of $\lfloor a/2 \rfloor$ routing steps. The size of the packets increases from $n/a$ until $n^2/a$ over the rounds. The transfer time is, therefore, bounded by $\lfloor a/2 \rfloor \cdot \sum_{t \ge 1} n^2/a^t = \lfloor a/2 \rfloor \, n^2/(1 - a)$. The time taken by Phase 4a is determined analogously. □

Just as CIRCGOS, TORGOS constitutes a compromise between simplicity and performance. The routing time will be somewhat smaller if the $a$-values and the corresponding $b$-values are chosen in dependency of the growing size of the packets. But even the presented basic version of TORGOS performs fairly well. In Table 2, we compare the performance of all two-dimensional algorithms. It can be seen that TORGOS is always the most efficient algorithm, but for small $r$-values the difference with Approach 2-1 is marginal. The performance of the variation of HIGH-DIM-GOS′ that utilizes CIRCGOS in both phases is better than that of the simple approaches but, nevertheless, slightly disappointing, particularly if one considers that this algorithm can choose its $a$ and $b$ values in each phase independently. Generally, we can conclude that if one aims for simplicity, one should utilize Approach 2-1. If a slightly more involved algorithm is acceptable, one should use TORGOS, which may be more than twice as fast.

## 6 HIGHER-DIMENSIONAL ARRAYS

For the success of TORGOS, it was essential that the packets were concentrated on diagonals at all times, as formulated in Invariant 1. Starting in such a situation, the invariant could be efficiently reestablished by copying horizontally (Phase 4a), and adding together vertically (Phase 4b).

The main problem in the construction of a gossiping algorithm for $d$-dimensional meshes is that it is unclear how the concept of a diagonal can be generalizes. Once we have such a "diagonal," we can perform an analogue of TORGOS. In the following section we describe the appropriate notion of $d$-dimensional diagonals. After that, we specify and analyze the gossiping algorithm for $d \ge 3$.

### 6.1 Generalized Diagonals

The property of a two-dimensional diagonal that must be generalized is the possibility of "seeing" a full and nonoverlapping hyperplane, when looking along any of the coordinate axes. We will try to explain what this means.

Let the **unit-cube** $I^d \in \mathbb{R}^d$ be defined as $I^d = [0, 1\rangle \times \cdots \times [0, 1\rangle$. When projecting the diagonal of $I^2$ orthogonally on the $x_0$-axes, we obtain the set $[0, 1\rangle \times 0$; when projecting on the $x_1$-axes, we obtain $0 \times [0, 1\rangle$. These projections are bijections (one-to-one mappings) from the diagonal to the sides of $I^2$. For algorithm TORGOS, this means that the information from diagonals can be replicated efficiently. A diagonal behaves like a magical mirror: Data received along one axis can be reflected along the other axis, not only in one direction, but in *both* directions. This requirement of problem-free copying between diagonals in adjacent submeshes along all coordinate axes leads to the following:

DEFINITION 1. *A subset $D$ of $I^d$ is called a $d$-dimensional diagonal if the orthogonal projections of $D$ onto any of the bounding hyperplanes of $I^d$ are bijective.*

We will prove that the union of the following $d$ sets satisfies the property of a $d$-dimensional diagonal:

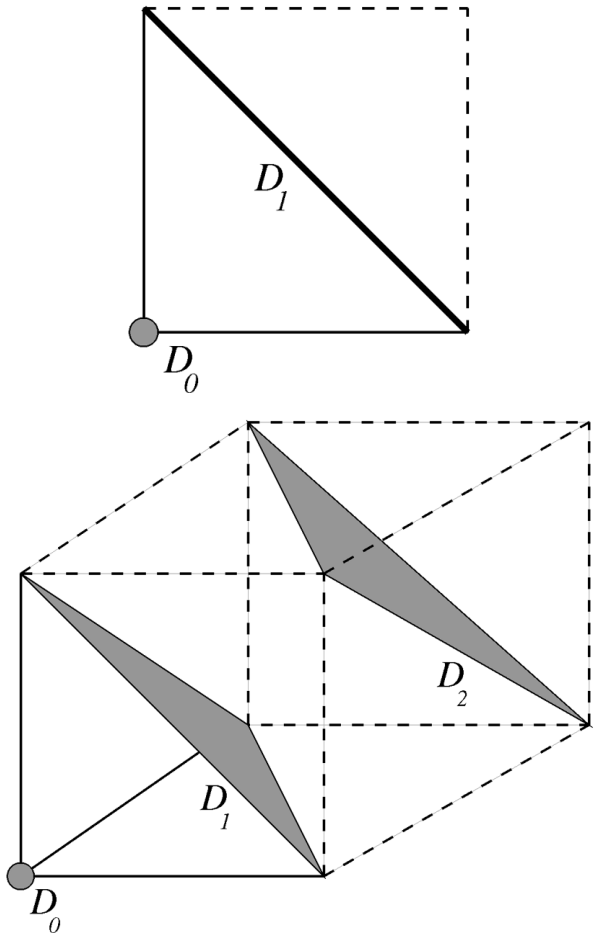



Fig. 3. Diagonals of $I^2$ and $I^3$. The bounding lines that belong (do not belong) to the considered sets are drawn solid (dashed). The corner points of $\mathcal{D}_1$ are no elements of it. Projecting $I^3$ downwards maps $\mathcal{D}_0 \cup \mathcal{D}_1 \cup \mathcal{D}_2$ bijectively on the ground plane.

$$\begin{cases} \mathcal{D}_0 &= \left\{0 \le x_0, x_1, \ldots, x_{d-1} < 1 \,\middle|\, x_0 + x_1 + \cdots + x_{d-1} = 0\right\} \\ \mathcal{D}_1 &= \left\{0 \le x_0, x_1, \ldots, x_{d-1} < 1 \,\middle|\, x_0 + x_1 + \cdots + x_{d-1} = 1\right\} \\ &\vdots \\ \mathcal{D}_{d-1} &= \left\{0 \le x_0, x_1, \ldots, x_{d-1} < 1 \,\middle|\, x_0 + x_1 + \cdots + x_{d-1} = d-1\right\}. \end{cases}$$

Notice that $\mathcal{D}_0 = \{(0, 0, \ldots, 0)\}$. On $I^2$, the diagonal consists of $\{(0, 0)\}$ as well as the points in $\{0 \le x_0, x_1 < 1 \mid x_0 + x_1 = 1\}$. The diagonals of $I^2$ and $I^3$ are illustrated in Fig. 3. On a torus, the $d$ partial hyperplanes are connected in a topologically interesting way.[1]

LEMMA 8. *$\bigcup_{i=0}^{d-1} \mathcal{D}_i$ gives a diagonal of $I^d$ in the sense of Definition 1.*

PROOF. As the $\mathcal{D}_i$ are completely symmetric, we can concentrate on the projection $\Pi_0$ along the $x_0$-axis. It is easy to check that for all $i$, $0 \le i \le d-1$,

1. It is easy to see that all $d$-subsets together form a closed $d-1$-dimensional subspace of the $d$-dimensional torus. On two-dimensional tori, they constitute a circle and on three-dimensional tori a two-dimensional torus. Generally, the diagonal of a $d$-dimensional torus is a $d-1$-dimensional torus, but a proof of this is beyond the scope of the paper.

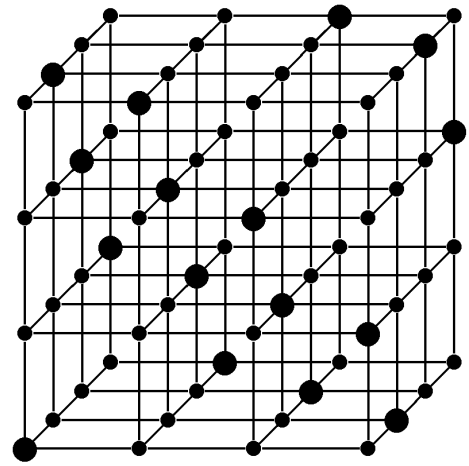


Fig. 4. The diagonal of a $4 \times 4 \times 4$ grid: 16 points, such that if they were occupied by towers in a three-dimensional chess game, none of them could capture an other.

$$\Pi_0(\mathcal{D}_i) = 0 \times \{0 \le x_1, \ldots, x_{d-1} < 1 \mid i - 1 < x_1 + \cdots + x_{d-1} \le i\}.$$

Clearly, these sets are all disjoint, so $\Pi_0$ is injective. On the other hand,

$$\begin{aligned} \bigcup_{i=0}^{d-1} \Pi_0(\mathcal{D}_i) = \\ 0 \times \left\{0 \le x_1, \ldots, x_{d-1} < 1 \,\middle|\, -1 < x_1 + \cdots + x_{d-1} \le d-1\right\} \\ = 0 \times \left\{0 \le x_1, \ldots, x_{d-1} < 1\right\}. \end{aligned}$$

which implies the surjectivity of $\Pi_0$. □

In order to extend the definitions to $d$-dimensional $n \times \cdots \times n$ cubes, one has to multiply all bounds on every $x_i$ by $n$. Thus, the diagonal $D_n$ can be defined concisely as

$$D_n = \left\{0 \le x_0, \ldots, x_{d-1} < n \,\middle|\, \left(\sum_{i=0}^{d-1} x_i\right) \bmod n = 0\right\}. \qquad (4)$$

On grids, only the points with integral coordinates should be taken. An example is given in Fig. 4.

So, we successfully defined $d$-dimensional diagonals. The reader is advised to obtain a full understanding of the case $d = 3$. For us it was helpful to construct a model of paper (cardboard would have been even better). Such a model makes it easy to convince oneself that the required property, that looking along a coordinate axis indeed gives a full but nonoverlapping view of the hyperplanes, is satisfied. Though we are not aware of any result in this direction, we are not sure that we are the first to define this concept. Still, we are very pleased with the utmost simplicity of (4) and the elegance of the proof of Lemma 8.

### 6.2 Details of the Algorithm

With the defined diagonals, we can now generalize TORGOS for gossiping on tori of arbitrary dimensions. The algorithm is almost the same as before. With a few extra routing steps, the algorithm can also be applied for meshes. Again, the presented algorithm is uni-axial. By applying the coloring technique of HIGH-DIM-GOS′, the transfer time can be reduced by a factor of $d$. By **rows**, we mean one-dimensional subspaces parallel to the $x_0$-axis.

TABLE 3
COMPARISON OF THE RESULTS OBTAINED FOR GOSSIPING ON A THREE-DIMENSIONAL $n \times n \times n$ TORUS

| $n \backslash r$ | 10 | | 50 | | 250 | | 1,000 | |
|---|---|---|---|---|---|---|---|---|
| | 289 | 111 | 769 | 111 | 3,219 | 111 | 12,119 | 111 |
| 9 | 280 | (9, 4) | 760 | (9, 4) | 3,036 | (6, 3) | 10,955 | (3, 1) |
| | 269 | (6, 4) | 688 | (3, 1) | 2,588 | (3, 1) | 9,713 | (3, 1) |
| | 3,917 | 211 | 5,197 | 211 | 10,459 | 221 | 28,909 | 221 |
| 27 | 3,873 | (27, 13) | 5,074 | (27, 13) | 10,195 | (27, 13) | 27,291 | (13, 10) |
| | 3,806 | (21, 19) | 4,881 | (10, 7) | 8,695 | (6, 3) | 20,237 | (3, 1) |
| | 91,857 | 211 | 95,977 | 211 | 110,238 | 221 | 152,238 | 221 |
| 81 | 91,658 | (81, 40) | 95,008 | (81, 40) | 108,390 | (81, 40) | 152,861 | (81, 40) |
| | 90,824 | (81,161) | 94,605 | (41, 30) | 105,506 | (19, 13) | 132,439 | (10, 6) |

*For every pair of values $(n, r)$, the number of time steps are given (from top to bottom) for application of* HIGH-DIM-GOS′ *with the best choice from Approach 1 and Approach 2 (indicated), for* HIGH-DIM-GOS′ *that utilizes* CIRCGOS *in every phase, and for* CUBGOS′. *For the latter two, the values of (the last) $a$ and $b$ are also indicated.*

Algorithm CUBGOS($n, d, a, b$)

1) In each row, $a$ PUs are designated as bridgeheads, namely the PUs which lie on the diagonal of their $n/a \times \cdots \times n/a$ submesh. Concentrate in each bridgehead $n/a$ packets from their rows.
2) For $\lfloor a/2 \rfloor$ steps, send packets of size $n/a$ along the rows among the bridgeheads in both directions, such that afterwards, every bridgehead contains the complete data of its row.
3) Perform $d-1$ round each consisting of $\lfloor a/2 \rfloor$ routing steps. In Round $i$, $1 \le i < d$, packets of size $a^{i-1} \cdot n$ are routed along the $x_i$-axis among the bridgeheads in both directions, such that afterwards, every bridgehead contains $a^i \cdot n$ data.

Now, each bridgehead in Row $(x_1, \ldots, x_{d-1})$, $0 \le x_1, \ldots, x_{d-1} < n$, holds all data from every Row $(x'_1, \ldots, x'_{d-1})$, where $\left(\sum_{i=1}^{d-1} \left((x_i - x'_i) \bmod (n/a)\right)\right) = 0$. Thus, all data are available on the diagonal of every $n/a \times n/a$ submesh. Hereafter, $\log_a n - 1$ further rounds are performed. In each round, the number of bridgeheads is increased by a factor of $a$.

INVARIANT 2. *At the beginning of Round $t$, $1 \le t \le \log_a n$, every bridgehead holds $n \cdot a^{(d-1)\cdot t}$ data, and all data are available on the diagonal of every $n/a^t \times \cdots \times n/a^t$ submesh.*

When $t = \log_a n$, this implies that the gossiping has been completed. A more formal description of the last phase is given below.

Algorithm CUBGOS($n, d, a, b$) (continued)

4) For $t = 1, \ldots, \lceil \log_a n - 1 \rceil$, repeatedly increase the number of bridgeheads by a factor of $a$ by inserting $a-1$ new bridgeheads between any pair of consecutive bridgeheads in every row.
   a) The information from the old bridgeheads in a row is passed to the $a-1$ new bridgeheads in $b \ge \lfloor a/2 \rfloor$ steps with packets of size $m/(2 \cdot b - a + 2)$, where $m = n \cdot a^{(d-1)\cdot t}$.
   b) Perform $d-1$ subphases each consisting of $\lfloor a/2 \rfloor$ routing steps. In Subphase $i$, $1 \le i < d$, packets of size $a^{i-1} \cdot m$ are routed along the $x_i$-axis among the bridgeheads (old as well as new) in both directions. Afterwards, every bridgehead contains $a^i \cdot m$ data.

The following analogue of Lemma 7 is straightforward:

LEMMA 9. *Let $T'_{cg,f}$ denote the number of time units needed for Phase $f$ of* CUBGOS($n, d, a, b$). *Then,*

$$T'_{cg,1} = n/(2 \cdot a) + \log_3(n/a) \cdot r,$$

$$T'_{cg,4\cdot a} = \frac{b \cdot n^d}{\left(a^{d-1} - 1\right) \cdot (2 \cdot b - a + 2)} + \left(\log_a n - 1\right) \cdot b \cdot r,$$

$$T'_{cg,2+3+4\cdot b} = \lfloor a/2 \rfloor \cdot \left(n^d/(a-1) + \left((d-1) \cdot \log_a n + 1\right) \cdot r\right).$$

Denote the version of the algorithm that utilizes coloring of the packets in order to fully exploit the all-port communication capability by CUBGOS′. Then we get

THEOREM 4. *Let $T'_{cg'}$ denote the number of time units taken by* CUBGOS′($n, d, a, b$). *Then,*

$$T'_{cg'} \simeq \frac{a}{a-1} \cdot n^d/(2 \cdot d) + d \cdot a/2 \cdot \log_a n \cdot r.$$

PROOF. In Lemma 9, the third expression dominates the other two by far. □

Thus, the transfer time is within a factor of $a/(a-1)$ from optimality for all $d$, and the start-up time is within a factor of $\frac{d \cdot a/2 \cdot \log_a n}{\log_{2\cdot d+1} n^d} \simeq \frac{a \cdot (\log d + 1)}{2 \cdot \log a}$ from optimality. This appears to be a really strong result. From Table 3 it can be seen that, for some $n$ and $r$, CUBGOS′ is substantially faster than HIGH-DIM-GOS′, even though the latter has much more freedom of choosing its parameters. Actually, if one is going to apply HIGH-DIM-GOS′, then one can just as well take the best of Approach 1 and Approach 2 in each of the phases.

## 7 EXPERIMENTS

To validate the efficiency of the developed algorithms, we implemented them on the Intel Paragon [2]. In this section, the experimental results are presented.

### 7.1 System Description

The Paragon system used for the experimentation consists of 140 PUs, each consisting of two 50MHz i860 XP microprocessors. One processor, called the **message processor**, is dedicated to communication, so that the compute processor is released from message-passing operations. Every PU is connected to a Mesh Routing Chip (MRC), and the MRCs are arranged in a two-dimensional mesh which is 14 nodes high and 10 nodes wide. The links can transfer data at a rate up to 175 MB/s in both directions simultaneously.

The algorithms were implemented using the NX message-passing library. NX is the programming interface supplied by Intel. Other communication layers for the Paragon, such as SUNMOS [10], achieve higher bandwidth and lower latency than NX, but were not available.

Some features of the Paragon are particularly important in order to understand the performance of the implemented algorithms, namely:

- The MRCs implement dimension order wormhole routing, i.e., packets are first routed along the rows to their destination columns and from there along the columns to their destinations. We employed this fact to embed a circular array into the mesh topology of the Paragon.
- When a message enters its destination before the receive is posted, the OSF/1 operating system buffers the message in a system buffer. When the corresponding receive is issued, the message is copied from the system buffer to the application buffer. This buffering is very expensive and can be avoided if the recipient first sends a zero-length synchronization message to the sender indicating that it has posted the receive. All implementations make use of this mechanism.
- In previous experiments on the Paragon [7], we determined that the startup cost of a message transmission under NX is about 150 $\mu$s. Short messages incur a somewhat lower startup cost than long messages, because they are sent immediately, whereas long messages wait until sufficient space is available at the destination processor. The experiments also showed that the unidirectional transfer rate from PU to MRC under NX is about 87 MB/s (11.5 ns per byte), whereas the bidirectional transfer rate is approximately 44 MB/s. Furthermore, the bidirectional transfer rate between two MRCs is 175 MB/s. Because of this, the topology of the Paragon can be viewed as a torus.

### 7.2 Modifications to the Algorithms

The implemented algorithms deviate slightly from the algorithms described in the previous sections. This was done for two reasons. First, because every PU of the Paragon is connected to an MRC and not directly to its (up to) four neighbors, we cannot assume the full-port model in which a PU can send and receive a message in all four wind directions simultaneously. Second, as mentioned above, the unidirectional transfer rate of the Paragon using NX is about 87 MB/s, whereas the bidirectional transfer rate is approximately 44 MB/s. This shows that it is more accurate to assume that a PU cannot send and receive simultaneously, although this is not a feature of the Paragon architecture but a feature of NX.

We give two examples of how (the analyses of) the algorithms need to be modified in order to reflect these communication characteristics of the Paragon. First, Approach 1 for gossiping on a circular array of $n$ PUs now consists of $n - 1$ steps, and in each step every PU must send a message to one of its neighbors and receive a message from its other neighbor. Since this cannot happen simultaneously, the time consumption of Approach 1 is given by

$$T_1(n, r) = (n - 1) \cdot (2 + 2 \cdot r).$$

Similarly, under the modified model it takes $\lceil \log_2 n \rceil$ instead of $\lceil \log_3 n \rceil$ steps to concentrate all data into one PU and another $\lceil \log_2 n \rceil$ steps to broadcast the data to every other PU. The time taken by Approach 2 is, therefore, approximately given by

$$T_2(n, r) \simeq \lceil \log_2 n \rceil \cdot (n + 2 \cdot r).$$

A detailed analysis of all algorithms under this model is omitted because the modifications are rather straightforward. Furthermore, the main purpose of this section is to show that the developed techniques actually work in practice, and not that the performance model is accurate. A detailed performance model should incorporate that short messages incur a lower startup cost than long messages, that the send and receive overheads can differ significantly, etc. This is beyond the scope of this paper.

An additional remark concerns the implementations of algorithm CIRCGOS and TORGOS. In the implementations, we used three parameters: $a_1$, $a_2$, and $k$, where $a_1$ is the number of bridgeheads, $a_2$ is the factor by which the number of bridgeheads is increased in every round, and $n/k$ is the packet size during Phase 3 (4a) of CIRCGOS (TORGOS). In the descriptions of CIRCGOS and TORGOS, we have set $a_1 = a_2 = a$ and $k = 2 \cdot b - a + 2$. For asymptotic analysis this is fine, but on such a moderate size platform rounding errors may be introduced which can affect the execution times significantly.

### 7.3 Experimental Results

The circular array algorithms were implemented on the Paragon by embedding a circular array into the mesh. Fig. 5 compares the performance achieved by algorithm CIRCGOS($a_1$, $a_2$, $k$) on a circular array with 64 PUs with the performance achieved by Approach 1 and the performance attained by Approach 2. Every data point measured for the implementation of algorithm CIRCGOS is labeled with the values of $a_1$, $a_2$, and $k$ for which the result was obtained. In order to place the data on a common scale, we divided the time taken by each algorithm by the message length $m$. The total execution time is obtained by multiplying the time per byte by the message length.

It can be seen that algorithm CIRCGOS is always faster than Approach 2. For messages up to 256 bytes, the best

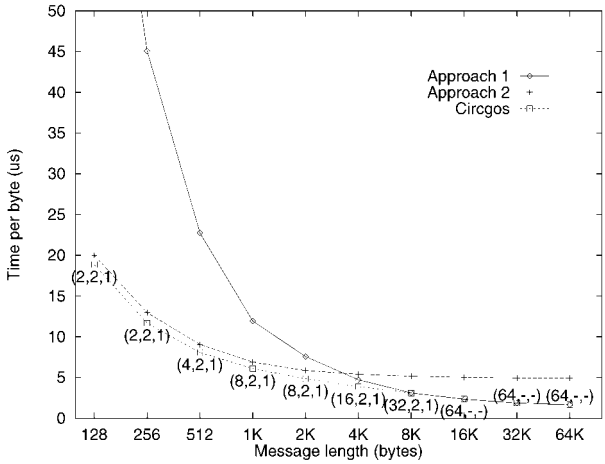



Fig. 5. Performance of the gossiping algorithms on a circular array with 64 PUs.

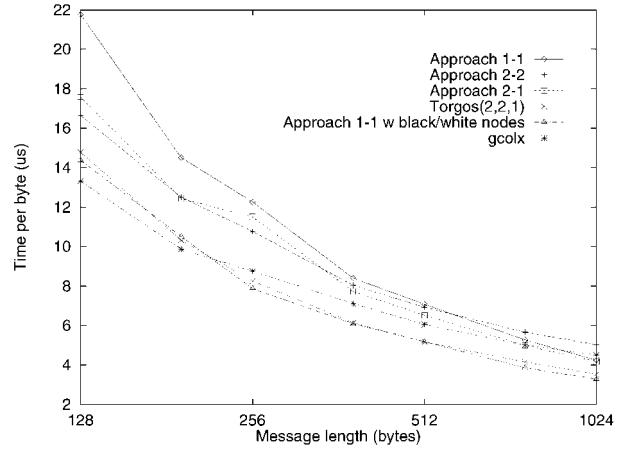
results are obtained with $a_1 = a_2 = 2$ and $k = 1$. With this set of parameters, the behavior of CIRCGOS is almost the same as the behavior of Approach 2, except that it saves one startup and the transmission of a packet of size $l \cdot n/2$ at the end of the concentration phase. When the message size increases, the fastest results are obtained when the number of bridgeheads $a_1$ also increases, but $a_2$ and $k$ remain fixed. With these parameters, Algorithm CIRCGOS first concentrates data in a few selected nodes as in Approach 2, after that it circulates the packets around as in Approach 1 and, finally, it broadcasts the data to all nonbridgeheads, again as in Approach 2. Other values for $a_2$ and $k$ always performed worse than $a_2 = 2$ and $k = 1$. When the message size increases beyond 16 KB, the best results are obtained when $a_1 = n = 64$. For this value of $a$, Algorithm CIRCGOS and Approach 1 behave identically, as can be seen since the data points coincide.

Fig. 6 shows the performance of six gossiping routines on an $8 \times 8$ configuration of the Paragon:

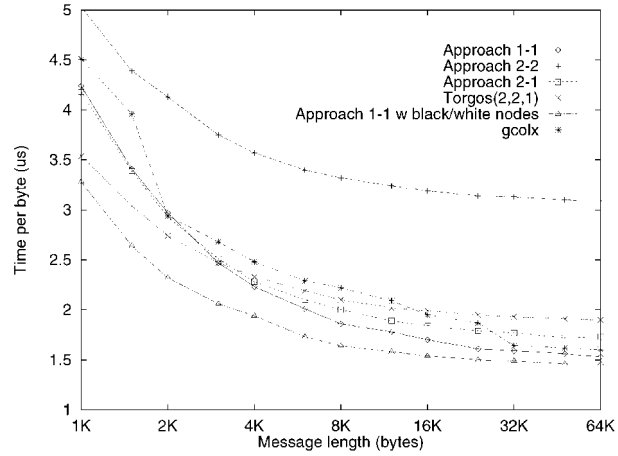
1) Approach 1-1,
2) Approach 2-2,
3) Approach 2-1,
4) Algorithm TORGOS with parameters $a_1 = a_2 = 2$ and $k = 1$,
5) Approach 1-1 using black/white packets, and
6) The gossiping routine **gcolx** provided by the NX message-passing library.

The fifth algorithm implementation does not partition the packet in every PU into a white and a black packet, but first performs a gossip in every $2 \times 2$ submesh, after which each $PU_{i,j}$ where $i + j$ is odd colors its packet white, and each $PU_{i,j}$ where $i + j$ is even colors its packet black. This was done because of the one-port restriction. Furthermore, the first four algorithm implementations do *not* employ the technique of interleaving horizontal and vertical messages. This was done because on such a moderate size network one does not save many startups by using a concentrate/broadcast approach instead of a store-and-forward approach. Moreover, if the PUs were divided into black and white PUs, the differences would almost vanish.

Because the differences between the various gossiping algorithms are rather small on this moderate size machine, we divided the experimental data into results for




(a)




(b)

Fig. 6. Performance of the gossiping algorithms on a two-dimensional mesh with 64 PUs. (a) Messages smaller than 1KB. (b) Messages larger than 1KB.

messages smaller than 1KB and messages larger than 1KB. Comparing Approaches 1-1, 2-2, 2-1, and TORGOS (2, 2, 1), we find that TORGOS is the fastest algorithm for messages up to 3KB. For larger messages, Approach 1-1 yields the best results. For a message of 3KB, the ratio between the startup cost of the message transmission and the transmission time of the message is about 4.2, and for such a small ratio Approach 1-1 turns out to be the fastest gossiping algorithm. Furthermore, as was indicated in Section 6, Approaches 2-2 and 2-1 have become obsolete; they never outperform the fastest of Approach 1-1 and TORGOS (2, 2, 1).

Comparing Approach 1-1 and TORGOS (2, 2, 1) with the gossiping routine **gcolx** supplied by NX, one can see that **gcolx** only yields the best results when the message size is very small. For messages larger than about 200 bytes, the fastest of our algorithm implementations always outperforms the vendor supplied routine. The largest relative difference was measured for messages of 1.5KB. For this message length, **gcolx** requires 3.96 $\mu$s/byte, whereas TORGOS (2, 2, 1) needs 3.04 $\mu$s/byte, which corresponds to a performance improvement by a factor of about 1.3. We

believe that this supports our claim that the developed algorithms have practical relevance.

Also included in Fig. 6 are the results obtained for an implementation of Approach 1-1 in which the nodes are divided into white and black nodes, and in which the white nodes route their packets at all times orthogonally to the black ones. It can be seen that, except for very small packets, this implementation always produces the best results. As stated before, this is due to the fact that on this moderate size machine one does not save many startups by using a concentrate/broadcast approach instead of a store-and-forward approach, especially when the nodes are split into white and black nodes. The results for this algorithm are mainly included here to show that the idea of interleaving horizontal and vertical packets can be used advantageously.

## 8 Conclusion

We presented gossiping algorithms for meshes of arbitrary dimensions. We optimized the trade-off between contributions due to start-ups and those due to the bounded capacity of the connections. This enabled us to reduce the time for gossiping in theory as well as practice for an important range of the involved parameters. Furthermore, we presented an interesting generalization of a diagonal, which can be applied to arbitrary dimensions. This seems to have wider applicability.

## Acknowledgments

Part of this research was performed during a stay by P.S. Rao at the Max-Planck-Institute. Ben H.H. Juurlink was supported in part by DFG-SFB 376 "Massive Parallelität" and by EU ESPRIT Long Term Research Project 20244 (ALCOM-IT). Ben H.H. Juurlink was affiliated with Leiden University, The Netherlands, when this work was commenced. P.S. Rao is on a one year leave from the Department of Computer Information Systems, University of Hyderabad, India. Computational support was provided by KFA Jülich, Germany.

## References

[1] M. Barnett, R. Littlefield, D.G. Payne, and R. van de Geijn, "On the Efficiency of Global Combine Algorithms for 2D Meshes with Wormhole Routing," *J. Parallel and Distributed Computing*, vol. 24, pp. 191–201, 1995.

[2] R. Berrendorf, H.C. Burg, U. Detert, R. Esser, M. Gerndt, and R. Knecht, "Intel Paragon XP/S—Architecture, Software Environment, and Performance," Technical Report KFA-ZAM-IB-9409, Forschungszentrum Jülich GmbH, 1994.

[3] O. Delmas and S. Perennes, "Circuit-Switched Gossiping in 3-Dimensional Torus Networks," *Proc. Second Euro-Par Conf., LNCS* vol. 1123, pp. 370–373, 1996.

[4] P. Fraigniaud and J.G. Peters, "Structured Communication in Torus Networks," *Proc. 28th Hawaii Conf. System Science*, vol. 2, pp. 584–593, 1995.

[5] Y. Huang and P.K. McKinley, "An Adaptive Global Reduction Algorithm for Wormhole-Routed 2D Mesh Networks," *Proc. Seventh Symp. Parallel and Distributed Processing*, pp. 114–121, 1995.

[6] K. Hwang, *Advanced Computer Architecture; Parallelism, Scalability, Programmability*. McGraw-Hill, 1993.

[7] B.H.H. Juurlink, "Experimental Validation of Parallel Computation Models on the Intel Paragon," *Proc. 12th Int'l Parallel Processing Symp. and Ninth Symp. Parallel and Distributed Processing*, pp. 492–497, 1998.

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

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

[10] A.B. Maccabe, K.S. McCurley, R. Riesen, and S.R. Wheat, "SUNMOS for the Intel Paragon: A Brief User's Guide," *Proc. Intel Supercomputer Users Group Ann. North Am. Users Conf.*, 1993.

[11] J.E. Marsden and A.J. Tromba, *Vector Calculus*. W.H. Freeman and Company, 1981.

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

[13] J.G. Peters and M. Syska, "Circuit-Switched Broadcasting in Torus Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 7, pp. 246–255, 1996.

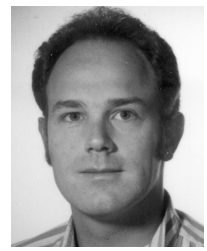
[14] S. Rajasekaran, "*k-k* Routing, *k-k* Sorting, and Cut-Through Routing on the Mesh," *J. Algorithms*, vol. 19, no. 3, pp. 361–382, 1995.

[15] J. Reif and L.G. Valiant, "A Logarithmic Time Sort for Linear Size Networks," *J. ACM*, vol. 34, no. 1, pp. 68–76, 1987.

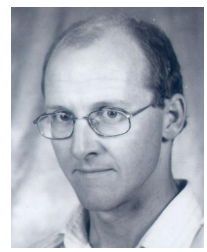
[16] P.S. Rao and G. Mouney, "Data Communications in Parallel Block Predictor-Corrector Methods for solving ODEs," Technical Report LAAS-CNRS, France, 1995.

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

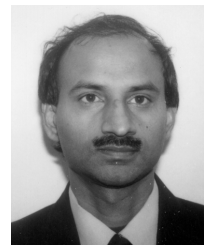
[18] Y.-C. Tseng, T-H. Lin, S.K.S. Gupta, and D.K. Panda, "Bandwidth-Optimal Complete Exchange on Wormhole-Routed 2D/3D Torus Networks: A Diagonal-Propagation Approach," *IEEE Trans. Parallel and Distributed Systems*, vol. 8, pp. 380–396, 1997.



**Ben H.H. Juurlink** received an MSc degree in computer science from Utrecht University, The Netherlands, in 1992, and a PhD degree in computer science from Leiden University, The Netherlands, in 1997. Since January 1997, he has been working as a post-doctoral researcher at the Heinz Nixdorf Institute and with the Department of Computer Science, Paderborn University, Germany. His research interests include parallel computation models, parallel algorithms and architectures, optimizing compilers, and performance evaluation.



**Jop F. Sibeyn** turned to computer science after studying physics and mathematics at Utrecht University. In 1992, he received his PhD. Since then, he has been working in Saarbrücken at the MPI für Informatik as a researcher. His main interest lies in practical and theoretical aspects of parallel algorithms for interconnection networks. Recently, he started working on external-memory algorithms and plans to do more research in this direction in the future. He is a member of the IEEE Computer Society.



**P.S. Rao** received the BSc degree from Bangalore University, India, in 1979, the MSc degree in applied mathematics from REC Warangal in 1981, and the PhD degree in numerical algorithms from the Indian Institute of Technology, New Delhi, India, in 1986. He is currently an assistant professor in the Department of Computer/Information sciences at the University of Hyderabad, India, and presently working as a consultant for Dow Jones Markets, on leave from the University of Hyderabad. Dr. Rao's research interests include parallel algorithms, distributed systems, routing techniques, and numerical algorithms. He has published several papers in these areas in refereed journals, such as Parallel Algorithms, BIT, JCAM, IMAJNA, and also in the proceedings of several international conferences. He was an IEEE student branch counselor, instrumental in starting an IEEE student branch and branch library at the university. He is a member of the IEEE.