Gossiping on Meshes and Tori*

Ben H.H. Juurlink
Dept. of Computer Science, Leiden University
P.O. Box 9512, 2300 RA Leiden, The Netherlands
Email: benj@cs.LeidenUniv.nl

P. S. Rao
Dept. of Computer Science, University of Hyderabad
Hyderabad - 500 046, India
Email: psraocs@uohyd.ernet.in

Jop F. Sibeyn
Max-Planck-Institut für Informatik
Im Stadtwald, 66123 Saarbrücken, Germany
Email: jopsi@mpi-sb.mpg.de

December 2, 1996

Abstract

Algorithms for performing gossiping on one- and higher dimensional meshes are presented. As a routing model, we assume the practically important worm-hole routing.

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, several simple algorithms composed of one-dimensional phases are 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 and by an implementation on the Paragon. For example, on a Paragon with 81 processors and messages of size 32 KB, relying on the built-in router requires 716 milliseconds, while our algorithm requires only 79 milliseconds.

For higher dimensional meshes, we give algorithms which are based on a generalized notion of a diagonal. These are analyzed theoretically and by simulation.

Index Terms: Parallel Algorithms, Interconnection Networks, Grids, Communication, Worm-hole Routing, Gossiping, Generalized Diagonals, Simulations, Experiments, Intel Paragon.

1 Introduction

Meshes. 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. While meshes have a large diameter in comparison to the various hypercubic networks, they are nonetheless of great importance due to their simple structure and efficient layout. Numerous parallel machines with mesh topologies have been built, and various algorithmic problems have been analyzed on theoretical models of the mesh.

Gossiping. Gossiping (also called complete exchange) is one of the fundamental communication problems. It appears in many contexts, both theoretical and practical. Gossiping is the problem in which

*Part of this research was performed during a stay of the second author at the Max-Planck Institute every PU of a network wants to send a packet of a given size to every other PU. Said differently, initially each of the N PUs of a network holds an amount of data of size L, and finally all PUs must 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/d steps. For meshes with worm-hole routing it is less obvious how to organize the routing such that the total cost is minimal.

Gossiping appears as a subroutine in many important problems in parallel computation. We just mention two. If M numbers are sorted on N PUs for some $M \gg N$, then a good approach is to select randomizedly or deterministically a set of m splitters [10, 6, 13]. These splitters must be made available in every PU. This means that we have to perform a gossip in which every PU contributes m/N numbers. Gossiping is an expensive operation, but in this case the amount of data to gossip is a lower-order fraction of the total amount of data. Thus, for large inputs the gossiping time becomes negligible. The efficiency of the gossiping algorithm determines the minimal problem size for which such algorithms can be used effectively.

A second application of gossiping appears in algorithms for solving ordinary differential equations using parallel block predictor-corrector methods [11, 12]. In each application of the block method, block point computations corresponding to the prediction are carried out by different PUs and these values are needed by the other PUs for the correction phase. At the end of correction phase, the corrected values are again needed by all PUs for advancing the computation, and thus needing *two* gossiping steps in each application of the method.

Earlier Work. A substantial amount of research has been performed on (variants of) the gossiping problem (see, e.g., [1, 3, 9, 2, 14]). Some of these papers address the same problem as considered by us: gossiping on meshes with worm-hole routing.

The algorithms in [9, 2] are very sophisticated and focus on minimizing the number of routing operations. Theoretically, this is an interesting issue, but practically, such an approach is appropriate only for very small packets. In this case the gossiping time is almost entirely determined by the start-up latencies (see Section 2). However, even for very small packets, the complicated structure of these algorithms may impair their implementation.

In [14] the main goal is to minimize the load on the connections. For two-dimensional tori, an algorithm is presented, in which the connections of the bisection have to transmit twice as many packets as given by a simple lower bound. This is achieved by some 'diagonal-propagation' approach. The approach is generalized for three-dimensional tori.

Our Results. In this paper in some sense we turn back to basics. Rather than to design an even more sophisticated algorithm, along the lines of [9, 2], we present a fairly simple algorithm and show that it actually works in practice.

An essential point is that we achieve an optimal trade-off between start-up time and data-transmission time. For relatively large messages, it is not enough to focus on the number of start-ups. A non-trivial lower bound shows that our algorithms are close to optimal for *all* possible values of the involved parameters. On two-dimensional meshes, the information is concentrated on diagonals. For higher dimensional meshes we give an interesting generalization of the notion of a diagonal, which may be of independent interest.

Our approach is somewhat similar to the approach of Tseng et al. [14], but there are important differences as well:

- In [14] trade-offs are not considered. Instead, a rather good general behavior is established. For small packets our approach is much better (requiring a logarithmic instead of a linear number of start-ups). For very large packets our approach is twice as fast.

- The generalization of a diagonal given in [14] for three-dimensional tori is rather straight-forward. Hyperspaces are used that when projected give back a diagonal in two-dimensional space. We generalize the diagonal in a more interesting way, and with reason: the performance is better, and a generic algorithm that works for arbitrary dimensions can be formulated without problem.

Furthermore, our theoretical results for two-dimensional meshes are completed with measurements of an actual implementation on the Intel Paragon. These measurements support our claims in most important points.

Contents. After a presentation of the model of computation and the lower bounds, the problem of gossiping on a one-dimensional mesh is analyzed in Section 4. Then we proceed with an extension to two- and higher-dimensional meshes. 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 links. We concentrate on the communication complexity, and assume that in a step a PU can perform an unbounded amount of internal computation. It is also assumed that a PU can send data to all adjacent PUs simultaneously. This is sometimes called the full-port model.

For the communication we assume the much considered worm-hole routing model (see [5, 8, 4] for some recent surveys). In this model a packet consists of a number of indivisible data units called *flits*. Furthermore, a packet has a header which contains the necessary routing information such as its destination. During routing the header governs the route of the packet and the other flits follow in a pipeline 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. In general worm-hole routing is likely to produce deadlock unless special care is taken.

The reasons to consider worm-hole 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 issue a packet is considerably larger than the time needed to traverse a connection. Worm-hole routing amortizes this 'start-up time', such that for sufficiently large packets it is hardly noticeable.

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*.

(1) is correct only if the packets do not contend for a single connection (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. 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 *For any network of N PUs with degree deg and diameter D, the time $T_{con}(N, deg, D)$ for concentrating all information in a single node satisfies:*

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

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

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

Usually t_d is comparable to t_l, while $D < N/deg \cdot l$. Thus we can omit the term $D \cdot t_d$ from the lower bound without sacrificing too much accuracy. Dividing all terms by $l \cdot t_l$, and 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:

$$T'_{\mathrm{con}}(N, deg, D) \geq \max\{N/deg, (\log N / \log(deg + 1)) \cdot r\}. \tag{2}$$

In Section 4, gossiping algorithms for $n \times \cdots \times n$ meshes are presented that match the lower bound of (2) up to a constant for all $r \leq n^{1-\epsilon}$, $\epsilon > 0$, and 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.

Theorem 1 *Let $r = t_s/(l \cdot t_l)$ and $r \leq (n-1)/e$. For gossiping on a linear processor array with n PUs, the number $T'_{gos}(n,1)$ of time units (of duration $l \cdot t_l$ each) satisfies*

$$T'_{gos}(n,1) = \Omega(n \cdot \ln n / \ln(n/r)).$$

Notice that Theorem 1 constitutes a smooth transition between the range of small r values, $r \leq n^{1-\epsilon}$, $\epsilon > 0$, and the range of large r values, $r = \Omega(n)$, for which (2) already gives sharp results. It is proven by several lemmas.

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.

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 of the gossiping can be established in T_{con} by reversing a concentration. On the other hand, a 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 $r = n/\log n$, $T'_{\text{con}}(n,1) = \mathcal{O}(n)$), we have $T_{\text{dis}} = \Theta(T_{\text{gos}})$.

For the dissemination problem with certain r, it is easy to see that having full freedom to choose 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 costs $2 \cdot r$. At most another constant factor difference is introduced if we assume that the gossiping 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 rightward connections only 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 gossiping all messages takes at least $\frac{n \cdot \ln n}{r \cdot \ln(n/r)}$ 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 (considered cyclically). The contribution of PU i to $F(t)$ by color c is $\ln((i-j) \bmod n)$ 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. Of essential importance is 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 $n/\ln(n/r)$.

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

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

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

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

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

We need to show that this expression is at most $n/\ln(n/r)$. 'Powering,' we obtain

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

By a well-known result[1] we have

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

It follows that

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

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

$$\begin{aligned} &\frac{n^2 \cdot k^k}{r \cdot (k+1)^{k+1}} > 1 \\ \Rightarrow\ & \frac{(k+1)^{k+1}}{k^k} < \frac{n^2}{r} \\ \Rightarrow\ & \left(1 + \frac{1}{k}\right)^k \cdot (k+1) < \frac{n^2}{r} \\ \Rightarrow\ & e \cdot (n+1) < \frac{n^2}{r}, \end{aligned}$$

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

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

So, we conclude that the number of steps required for dissemination is at least

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

□

[1] The Lagrange multiplicator theorem (given in any book on calculus, e.g., [7, Section 4.3]), immediately gives that the product of factors with a fixed sum is maximal if all factors are equal.

4 Linear and Circular Arrays

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. We only present algorithms for circular arrays. Because of the more regular structure, these are slightly 'cleaner', but with minor modifications all of them carry on for linear arrays.

4.1 Basic Approaches

For gossiping on a circular array consisting of 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. Perform a recursive concentration of the data packets into a selected PU, then reverse the process, to disseminate the information to all other PUs.

Lemma 4 *If the packets consist of l flits each, then Approach 1 takes*

$$T_1(n, l) = \lfloor n/2 \rfloor \cdot (t_s + t_d + l \cdot t_l),$$

for gossiping on a circular array with n PUs.

Proof: The algorithm consists of $\lfloor n/2 \rfloor$ trivial steps. □

Lemma 5 *If the packets consist of l flits each, then the time consumption for Approach 2 can be estimated on*

$$T_2(n, l) \simeq \log_3 n \cdot (2 \cdot t_s + n \cdot l \cdot t_l). \tag{4}$$

Proof: During the concentration, the number of 'active' PUs is reduced by a factor three in every step. The packets get three times as heavy in every step, and the distance over which the packets have to be sent, is multiplied by three. For the concentration this gives

$$T_{\text{conc}} = \sum_{i=0}^{\log_3 n - 1} (t_s + 3^i \cdot (t_d + l \cdot t_l)) < \log_3 n \cdot t_s + n/2 \cdot (t_d + l \cdot t_l).$$

The expression for the dissemination phase is similar, but here the packets consist of $n \cdot l$ flits in all steps. Thus,

$$T_{\text{dis}} = \sum_{i=0}^{\log_3 n - 1} (t_s + 3^i \cdot t_d + l \cdot n \cdot t_l) < \log_3 n \cdot (t_s + n \cdot l \cdot t_l) + n/2 \cdot t_d.$$

t_d is of the same order of magnitude as t_l. So, the term $n/2 \cdot t_d$ is small in comparison to $\log_3 n \cdot n \cdot l \cdot t_l$. Adding the two contributions gives the lemma. □

Approach 1 is good when t_s is small: according to (2), it is exactly optimal for $r = t_s/(l \cdot t_l) = 0$. In the other extreme, that is, for $r \to \infty$, Approach 2 becomes optimal to within a constant factor by Lemma 1. It will be better than Approach 1 for many practical values. Still, in principle the time consumption of Approach 2 is not even linear in n. For large n or l, the term $\log_3 n \cdot n \cdot l \cdot t_l$ dominates, even though t_l is much smaller than t_s.

4.2 Intermixed Approach

We henceforth neglect the distance term, which is of minor importance anyway, and write $t_l' = l \cdot t_l$ instead of t_l. If $t_s \gg t_l'$, then for all reasonable values of n, the result of Lemma 5 cannot be improved. However, there are good reasons to assume that for certain problems, t_s and t_l' may be of comparable magnitude:

Observation 1 *Any ratio t_s/t_l' is possible.*

For example, if a large sorting problem is solved on a relatively small system, then the packets that have to be sent are large, consisting of many flits. In that case it may even happen that $t_l' = l \cdot t_l > t_s$. For such instances, we propose an approach which has features of both basic approaches.

The algorithm consists of three phases, and works with parameters a and b. Later we discuss the right choices for them.

Algorithm CIRCGOS(a, b)

1. Concentrate n/a (rounding!) data in a evenly interspaced PUs, called *bridge heads* or *concentration points*.

2. For $\lfloor a/2 \rfloor$ steps, send packets of size n/a among the concentration points in both directions, such that afterwards all data are known in all concentration points.

3. In $\lceil \log_a n - 1 \rceil$ further rounds, repeatedly increase the number of bridgeheads by a factor of a until n. The information is passed to the $a-1$ new points between any two existing bridgeheads in $b \geq \lfloor a/2 \rfloor$ steps with packets of size $n/(2 \cdot b - a + 2)$.

In Phase 2, the packets are circulated around. The description is pleasant because of the circular structure. A small improvement (of some importance only for small meshes), may be obtained, by exploiting in Phase 3 that already part of the data are known in the points to which the data are transferred. Notice that the algorithm 'degenerates' into Approach 1 for $a = n$.

It is easy to compute the total time for CIRCGOS. Remind that the distance term is neglected, and that we do not consider all details of rounding.

Lemma 6 *The three phases of* CIRCGOS*(a, b) take*

$$\begin{aligned} T_{cg,\,1} &= \log_3(n/a) \cdot t_s + n/(2 \cdot a) \cdot t_l', \\ T_{cg,\,2} &= \lfloor a/2 \rfloor \cdot (t_s + n/a \cdot t_l'), \\ T_{cg,\,3} &= (\log_a n - 1) \cdot b \cdot (t_s + \lceil n/(2 \cdot b - a + 2) \rceil \cdot t_l'). \end{aligned}$$

Proof: Phase 1 is a simple concentration step, on linear arrays of size n/a instead of n. The time for Phase 2 follows by multiplying the number of steps by the time for each of them. Phase 3 consists of $\log_a n - 1$ rounds of b steps each. $\square$

At a first glance, it is not clear what the result of Lemma 6 means. Particularly, it is not immediately clear which a and b should be chosen. To obtain an impression, we have written a small program which searches for the optimal values. In Table 1 we have listed some typical results. There are several interesting conclusions that can be derived:

- For realistic values of n and t_s/t_l', CIRCGOS may be several times faster than the best of Approach 1 and Approach 2.
- At worst, CIRCGOS is hardly slower than Approach 2 (actually, for $a = 3$ and $b = 1$, it becomes equal to Approach 2, except for the fact that the presence of knowledge is not exploited).
- The range of t_s/t_l' values for which a non-degenerate instance of CIRCGOS is the best increases with n.
- The best choices of a and b increase with n and decrease with t_s/t_l'.

Though the detailed choice of the parameters is essential for the performance of CIRCGOS, the analysis of what happens is *qualitatively* unchanged if we take $a = b = n \cdot t_l'/t_s$. For this choice the performance is asymptotically optimal (cf. Theorem 1):

$n \diagdown t_s/t'_l$	2		10		50		250	
	40		144		664		3264	
27	64		104		304		1304	
	40	(27, -)	**100**	(3, 1)	318	(3, 1)	1318	(3, 1)
	120		440		2040		10040	
81	257		319		593		1993	
	120	(81, -)	**239**	(5, 4)	594	(3, 1)	2013	(3, 1)
	364		1332		6172		30372	
243	990		1062		1422		3222	
	337	(4, 8)	**565**	(7, 7)	**1251**	(3, 2)	3248	(3, 1)
	1092		4004		18564		91364	
729	3667		3755		4195		6395	
	936	(10, 20)	**1377**	(13, 17)	**2707**	(7, 7)	**6264**	(4, 2)

Table 1: Comparison of the results obtained for gossiping on a circular arrays with n PUs applying Approach 1 (top), Approach 2 (middle) and CIRCGOS (bottom). The instances for which CIRCGOS is better are printed bold. Behind the results for CIRCGOS, the values of the parameters a and b for which the result was obtained are indicated. The cost unit is t'_l.

Theorem 2 *Let $r = t_s/(l \cdot t_l)$, $r < n$. We consider gossiping on a linear processor array with n PUs. When applying* CIRCGOS*$(n/r, n/r)$, the number $T'_{gos}(n, 1)$ of time units (of duration $l \cdot t_l$ each) satisfies*

$$T'_{gos}(n, 1) = \mathcal{O}(n \cdot \ln n / \ln(n/r)).$$

Proof: From Lemma 6 we obtain

$$T'_{\mathrm{gos}}(n, 1) = \mathcal{O}(n + r \cdot \log_3 r + n \cdot \log_{n/r} r) = \mathcal{O}(n + r \cdot \ln r + n \cdot \ln r / \ln(n/r)).$$

For $r = n/x$ with $x > 1$, the second term never dominates, because $r = n/x \leq n/\ln x = n/\ln(n/r)$. Replacing the factor $\ln r$ in the third term by $\ln n$ gives the theorem. $\square$

Thus, CIRCGOS gives a natural continuous transition from gossiping times $\mathcal{O}(n)$, as achieved by Approach 1 for $r = \mathcal{O}(1)$, to gossiping times $\mathcal{O}(n \cdot \log n)$, as achieved by Approach 2 for $r = n$. Though we have not analyzed the constants hidden in the expressions of Theorem 1 and Theorem 2, probably they do not lie far apart: for $r \leq 1$ and $r \geq n$, CIRCGOS degenerates to Approach 1 and Approach 2, respectively, and these are provably close to optimal (see the remark at the end of Section 4.1). For intermediate r values, CIRCGOS may be substantially faster:

Corollary 1 *Let $r = t_s/t'_l$. For all $\log n \leq r \leq n^{1-\epsilon}$, $\epsilon > 0$,* CIRCGOS *is $\Omega(\log n)$ times faster than Approach 1 and 2.*

Proof: For $r \geq \log n$, Approach 1 and Approach 2 both take $\Omega(n \cdot \log n)$ time units. On the other hand, for $r = n^{1-\epsilon}$, CIRCGOS has a time consumption bounded by $\mathcal{O}(n \cdot \log n/(1 - \epsilon) \cdot \log n) = \mathcal{O}(n)$. $\square$

4.3 Extension to Higher Dimensions

We sketch a very simple algorithm for gossiping on d-dimensional meshes, $d > 1$.

The algorithm consists of d phases: in Phase f, $0 \leq f \leq d-1$, the packets participate in a gossip along axis f. For each of the one-dimensional gossips, we take the most efficient algorithm from Section 4. As the size of the packets increases with the number of phases performed (in Phase f, the packets consist of $l \cdot n^f/d$ flits, if they initially consisted of l flits), this is not necessarily the same algorithm in all phases. Call the described algorithm HIGH-DIM-GOS.

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

Proof: Denote the time for Phase f of HIGH-DIM-GOS by $T_{\text{hdg},\ f}$, and the optimal gossiping time by T_{opt}. Clearly, T_{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 $1/d$ of the connections is used. Using all connections would make the algorithm faster by at most a factor d. Thus, $T_{\text{hdg},\ d-1} \leq d \cdot T_{\text{opt}}$. As $T_{\text{hdg},\ f} \leq T_{\text{hdg},\ d-1}$, for all $0 \leq f < d-1$, $\sum_f T_{\text{hdg},\ f} \leq d^2 \cdot T_{\text{opt}}$. □

It is easy to give a considerably better algorithm, exploiting the full routing power of a higher dimensional mesh. First the packets are colored with d colors. The packet in PU $(i_0, \ldots, i_{d-1})$ is given color $(i_0 + \cdots + i_{d-1}) \bmod d$. Now, in Phase f, $0 \leq f \leq d-1$, the packets with color c participate in a gossip along axis $(f + c) \bmod d$. To this algorithm we refer by HIGH-DIM-GOS$'$.

The most important conclusion from this section is that for higher dimensional meshes, achieving asymptotically optimal results is *not* an important issue. The real issue is of a more practical nature: constructing algorithms with minimal routing time for a given choice of parameters, paying attention to the constants involved. This will be the main goal of the subsequent sections.

5 Two-Dimensional Arrays

In this section we extend the approach of the previous section, to obtain simple but effective algorithms for gossiping on two-dimensional meshes. In principle, more sophisticated approaches, e.g. as presented in [9], give better performance. However, they are so complicated, that we do not believe that they might be implemented effectively.

5.1 Basic Approaches

The simplest idea is to apply HIGH-DIM-GOS$'$, with the best of Approach 1 and Approach 2 in each of the two phases. By Approach i-j, we denote the algorithm in which first Approach i is applied and then Approach j. Approach 1-2 can be excluded. The time for gossiping with Approach i-j, is denoted $T_{i,j}$. Using the results from Section 4, we find

Lemma 7

$$\begin{aligned} T_{1,1} &\simeq 3/4 \cdot n \cdot t_s + n/4 \cdot (n+1) \cdot t_l', \\ T_{2,1} &\simeq (2 \cdot \log_3(n/2) + n/2) \cdot t_s + n/2 \cdot (\log_3(n/2) + n/2) \cdot t_l', \\ T_{2,2} &\simeq (4 \cdot \log_3 n - 3) \cdot t_s + (2 \cdot \log_3 n - 2) \cdot n/2 \cdot (n/2 + 1) \cdot t_l'. \end{aligned}$$

Proof: Applying Approach 1 in Phase 1 takes $\lfloor \lceil n/2 \rceil / 2 \rfloor \cdot (t_s + t_l')$, because there are at most $\lceil n/2 \rceil$ packets on the one dimensional subarrays (rings). According to Lemma 5, applying Approach 2 instead, takes about $(2 \cdot \log_3 \lceil n/2 \rceil - 1) \cdot t_s + (\log_3 \lceil n/2 \rceil - 1) \cdot \lceil n/2 \rceil \cdot t_l'$. In Phase 2, we start with packets of size at most $\lceil n/2 \rceil$ in all PUs of a column or row. This gives $\lfloor n/2 \rfloor \cdot (t_s + \lceil n/2 \rceil \cdot t_l')$ for Approach 1, and $(2 \cdot \log_3 n - 1) \cdot t_s + (\log_3 n - 1) \cdot n^2/2 \cdot t_l'$ for Approach 2. □

Substituting some values for n and $r = t_s / t_l'$ (numerical results are given in Table 2), shows that

- Approach 1-1 is the best if $r \lesssim 10$.
- Approach 2-1 is the best if $10 \lesssim r \lesssim (\log_3 n - 1) \cdot n$. It may be up to 25% faster than the best of the other two approaches.
- Approach 2-2 is the best if $(\log_3 n - 1) \cdot n \lesssim r$.

5.2 Intermixed Approach

The described approaches are competitive for many choices of n, t_s and t_l', but for an important range, they are not. In this section we describe a more truly two-dimensional approach. It gives results that are better by a constant factor for intermediate r values.

The algorithm is a two-dimensional analogue of CIRCGOS. We concentrate on the 'white' packets, initially residing in the PUs $P_{i,j}$ with $i+j$ even. The 'black' packets are handled analogously, being routed at all times orthogonally to the white ones.

Algorithm TORGOS

1. Concentrate all white packets in a concentration points of their rows: in row i, in the PUs $P_{i,j}$ with $(j-i) \bmod (n/a) = 0$. Each concentration point now holds $n/(2 \cdot a)$ white packets.

2. Route the data residing in each concentration point in $\lfloor a/2 \rfloor$ steps to all other concentration points in the same row. Now every concentration point holds $n/2$ white packets.

3. Route the data residing in each concentration point in $\lfloor a/2 \rfloor$ steps to all other concentration points in the same column. Now every concentration point holds $a \cdot n/2$ white packets.

4. Determine suitable b_i, $0 \leq i < t$, such that $\prod_i b_i = n/a$, and $x_i \geq \lfloor b_i/2 \rfloor$. Perform t rounds of further concentration. At the beginning of round j, $0 \leq j < t$, the concentration points all hold $S_j = a \cdot \prod_{i=0}^{j-1} b_i \cdot n/2$ white packets.

- **a.** The data are divided into packets of size $S_j/(2 \cdot x_j - b_j + 2)$. Route these packets during x_j steps along the rows, to $b_j - 1$ points equally interspaced between any two concentration points.
- **b.** Perform $\lfloor b_j/2 \rfloor$ steps of vertical routing with packets of size S_j.

Phase 1 is performed by a repeated concentration in $\log_3(n/a)$ steps. In Phase 2 and 3, the packets are circulated around. As for CIRCGOS, the description is simple because of the wrap-around connections, but further the algorithm immediately carries over to meshes. After Phase 1, all data are present on each of a diagonals. After Phase 3, all data are present on each section of length n/a of these diagonals. In Phase 4, new diagonals are created. First the data are copied to them (4.a), then they are made available in all sections (4.b).

We compute the time consumption of TORGOS. The distance term is neglected, and we do not consider all details of rounding.

Lemma 8 *The phases of* TORGOS(a, b_i, x_i) *take*

$$\begin{aligned}
T_{tg,\ 1} &= \log_3(n/(2 \cdot a)) \cdot t_s + n/(4 \cdot a) \cdot t_l', \\
T_{tg,\ 2} &= \lfloor a/2 \rfloor \cdot (t_s + n/(2 \cdot a) \cdot t_l'), \\
T_{tg,\ 3} &= \lfloor a/2 \rfloor \cdot (t_s + n/2 \cdot t_l'), \\
T_{tg,\ 4.a,\ j} &= x_j \cdot (t_s + (\frac{a \cdot \prod_{i=0}^{j-1} b_i \cdot n/2}{2 \cdot x_j - b_j + 2}) \cdot t_l'), \\
T_{tg,\ 4.b,\ j} &= \lfloor b_j/2 \rfloor \cdot (t_s + a \cdot \prod_{i=0}^{j-1} b_i \cdot n/2 \cdot t_l').
\end{aligned}$$

Proof: The arguments are analogous to those in the proof of Lemma 6. □

Actually, we defined a class of algorithms, leaving the parameters unspecified. Because the weight of the packets increases after every round in Phase 4, the optimal choice of the b_i satisfies

$$2 \leq b_0 \leq b_1 \leq \ldots \leq b_{t-1}.$$

For all realistic values of n, we have to choose only a few b_i, and it turns out that excellent performance is even obtained when all $b_i = b$ and all $x_i = x$, for certain b and x. This considerably reduces the effort of choosing the optimal parameters, and simplifies the expressions for the total time consumptions during Phase 4.a and 4.b:

$$\begin{aligned}
T_{\text{tg, 4.a}} &\simeq x \cdot (\log_b(n/a) \cdot t_s + n^2/(2 \cdot (b-1) \cdot (2 \cdot x_j - b_j + 2)) \cdot t_l'), && (5) \\
T_{\text{tg, 4.b}} &= \lfloor b_j/2 \rfloor \cdot (\log_b(n/a) \cdot t_s + n^2/(2 \cdot (b-1)) \cdot t_l'). && (6)
\end{aligned}$$

In Table 2 we give some results of simulations. The optimal choices of a, b and x were found by a simple

$n \diagdown t_s/t_l'$	8		30		100		250	
	351		796		2214		5252	
	360		761		2038		4774	
27	855		1053		1683		3033	
	444		606		1122		2227	
	363	(3, 9, 7)	**605**	(3, 3, 2)	1122	(3, 3, 1)	2227	(3, 3, 1)
	2146		3483		7736		16848	
	2155		3194		6501		13568	
81	10188		10474		11384		13334	
	3424		3652		4378		5934	
	2162	(3, 27, 22)	**2828**	(9, 9, 8)	**3982**	(3, 5, 3)	5934	(3, 3, 1)
	16281		20290		33048		60386	
	16335		19200		28317		47853	
243	119206		119580		120770		123320	
	29814		30108		31044		33049	
	16288	(5, 49, 54)	**17808**	(7, 35, 32)	**21101**	(3, 9, 9)	**25477**	(3, 9, 7)
	137416		149445		187718		269730	
	137819		146074		172341		228627	
729	1332416		1332878		1334348		1337498	
	266398		266758		267904		270360	
	137398	(3, 243, 211)	**141693**	(9, 81, 86)	**149888**	(15, 49, 49)	**162239**	(17, 43, 37)

Table 2: Comparison of the results obtained for gossiping on a $n \times n$ tori. Results are given for Approach 1-1 (top rows), Approach 2-1 (second rows) and Approach 2-2 (third rows). In the fourth rows we give the results for TORGOS, with $a = b = 3$ and $x = 1$. In the lowest rows, we give the results for TORGOS, with the corresponding optimal choices of a, b and x indicated in brackets. Where these results are better than any of the others, they are printed bold. The cost unit is t_l'.

computer program. We also added the results for an analogue Approach 2: the version of TORGOS with $a = b = 3$ and $x = 1$.

As claimed before, Approach 2-1 may be substantially better than either Approach 1-1 or Approach 2-2. The smallest problem instance for which this occurs is $n = 27, k = 50$. But, looking at the complete list of results, we see that Approach 2-1 and Approach 2-2 have become obsolete: in all cases TORGOS performs better. Only for small $r = t_s/t_l'$, it may happen that Approach 1-1 is the best of all. Trying shows that this happens if $r \leq 8$ (except for very large n). From these observations, it is easy to derive a good gossiping strategy for two-dimensional tori. Qualitatively one can say that

- The range of r values for which a non-trivial instance of TORGOS is the best increases with n.
- The best choices of a, b and x increase with n. The best choices of b and x decrease with r. a is always rather small; only for relatively large r, b is smaller than n/a. x is of the same order as b.

6 Higher Dimensions

For the success of the two-dimensional algorithm it was essential that the packets were concentrated on diagonals at all times: after round j of Phase 4, all data is known on a diagonal consisting of $n/(a \cdot b^j)$ PUs, each PU holding $n \cdot a \cdot b^j$ data. Starting in such a situation, this property could be efficiently established for $j + 1$: copying horizontally (Phase 4.a), and wiping together vertically (Phase 4.b).

The main problem in the construction of a gossiping algorithm for d-dimensional meshes is, that it is not clear how to generalize the concept of a diagonal. 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. Thereafter, we specify and analyze the gossiping algorithm for $d \geq 3$.

6.1 Generalized Diagonals

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

In $[0,1] \times [0,1] \subset \mathbb{R}^2$, when projecting its diagonal, the set $\{x + y = 1 | 0 \leq x, y \leq 1\}$, perpendicularly on the y-axes, we obtain the set $0 \times [0,1]$; when projecting on the x-axes, we obtain $[0,1] \times 0$. The projection is not only surjective, but also injective. For our gossiping algorithm TORGOS, this means that the information from diagonals in adjacent submeshes can be copied without problem onto each other. 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 us to the following

Definition 1 *A subset of a d-dimensional cube is called a d-dimensional diagonal, if the following two conditions are satisfied:*

1. *The perpendicular projection of the diagonal of a subcube onto any of the bounding hyperplanes of this cube is surjective.*
2. *The perpendicular projection of the diagonal of a subcube onto any of the bounding hyperplanes of this cube is injective except for subsets of total measure zero.*

For the *unit-cube* $I^d = [0,1] \times \cdots \times [0,1]$, the union of the intersections of the following set hyperplanes with the I^d gives a diagonal:

$$\left\{ \begin{array}{rcl} \mathcal{D}_1 & = & \{x_0, x_1, \ldots, x_{d-1} | x_0 + x_1 + \cdots + x_{d-1} = 1\} \\ \mathcal{D}_2 & = & \{x_0, x_1, \ldots, x_{d-1} | x_0 + x_1 + \cdots + x_{d-1} = 2\} \\ & \vdots & \\ \mathcal{D}_{d-1} & = & \{x_0, x_1, \ldots, x_{d-1} | x_0 + x_1 + \cdots + x_{d-1} = d-1\} \end{array} \right.$$

Lemma 9 *A diagonal of I^d is given by*

$$\bigcup_{i=1}^{d-1} \mathcal{D}_i \cap I^d.$$

Proof: As the D_i are completely symmetric, we can concentrate on the projection along the x_0-axis. Denote the projection of a set $\mathcal{S}$ along the x_0-axis by $\Pi_0(\mathcal{S})$. It is easy to check that for all $1 \leq i \leq d$,

$$\Pi_0(\mathcal{D}_i) = \{x_0, x_1, \ldots, x_{d-1} | x_0 = 0, i - 1 \leq x_1 + \cdots + x_{d-1} \leq i\}. \tag{7}$$

So, for any $i < j$,

$$\Pi_0(\mathcal{D}_i) \cap \Pi_0(\mathcal{D}_j) = \left\{ \begin{array}{ll} \{x_0, x_1, \ldots, x_{d-1} | x_0 = 0, x_1 + \cdots x_{d-1} = i\} & \text{if } j = i+1, \\ \emptyset & \text{if } j > i+1. \end{array} \right.$$

Hence, the projection is almost injective, as required by point 2 of Definition 1. On the other hand, (7) gives that

$$\begin{array}{rcl} \bigcup_{i=1}^{d-1} \Pi_0(\mathcal{D}_i) & = & \bigcup_{i=1}^{d-1} \{x_0, x_1, \ldots, x_{d-1} | x_0 = 0, i-1 \leq x_1 + \cdots + x_{d-1} \leq i\} \\ & = & \{x_0, x_1, \ldots, x_{d-1} | x_0 = 0, 0 \leq x_1 + \cdots + x_{d-1} \leq d-1\} \end{array}$$

Hence, the projection is surjective as well. □

So, we successfully defined d-dimensional diagonals. The reader is advised to obtain a full understanding of the case $d = 3$, as illustrated in Figure 1. 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 non-overlapping view of the hyperplanes.

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 the defined diagonal and the elegance of the proof of Lemma 9.

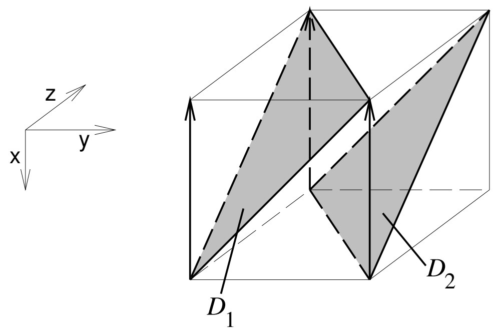


Figure 1: Diagonal of the unit cube; the projection along the x-axis is the set $0 \times [0,1] \times [0,1]$.

6.2 Details of the Algorithm

On a d-dimensional mesh or torus, the packets are colored with a color from $\{0, 1, \ldots, d-1\}$: the packet residing in position $(x_0, x_1, \ldots, x_{d-1})$ is given color $(\sum_i x_i) \bmod d$. In this way, there are n/d packets with the same color on any 1-dimensional submesh. The packets of each color are treated independently by d orthogonally operating gossiping procedures. In order to get a clear exposition, we present the algorithm for tori. With some extra routing steps, meshes can be treated similarly. We concentrate on the color 0 packets. The packets of color i, $1 \leq i \leq d-1$, are handled analogously. In order not to complicate the algorithm, we give a generalization of TORGOS(3, 3, 1):

Algorithm CUBGOS(n, d)

1. In every PU, compute the nearest intersection of its row with the set of diagonals of all $n/3 \times \cdots \times n/3$ subtori. In each row, concentrate all color 0 packets in $\log_3 n - 1$ steps in the three computed concentration points. Each concentration point now holds $n/(3 \cdot d)$ color 0 packets.

2. Perform d steps of data spreading: in Step i, $0 \leq i \leq d-1$, the color 0 data residing in a concentration point are routed along axis i to both other concentration points on this same axis. After Step i, every concentration point holds $n/d \cdot 3^i$ color 0 data.

3. Perform $\log_3 n - 1$ rounds of further concentration. At the beginning of round j, $1 \leq j \leq \log_3 n - 1$, the concentration points all hold $n/d \cdot 3^{j \cdot (d-1)}$ color 0 data. All data are known within every $n/3^j \times \cdots \times n/3^j$ subtorus. In round j, two diagonals are added between any two existing diagonals. Then we perform

a. Route a copy of the color 0 data in each direction along axis 0, from every PU on an old diagonal to the PUs on the two adjacent new diagonals.

b. Perform $d-1$ steps of data spreading: in Step i, $1 \leq i \leq d-1$, the color 0 data residing in a concentration point are routed along axis i to the concentration points on adjacent diagonals. After Step i, every concentration point holds $n/d \cdot 3^{j \cdot (d-1)+i}$ color 0 data.

Notice that the situation after Phase 1 is similar to the situation after Phase 3.a, and that Phase 2 is analogous to Phase 3.b. We compute the time consumption of CUBGOS.

Lemma 10 *The phases of* CUBGOS(n, d) *take*

$$\begin{aligned} T_{cg,\ 1} &= (\log_3 n - 1) \cdot t_s + n/(6 \cdot d) \cdot t_l', \\ T_{cg,\ 3.a} &= (\log_3 n - 1) \cdot t_s + n^d/(d \cdot (3^{d-1} - 1)) \cdot t_l', \\ T_{cg,\ 2} + T_{cg,\ 3.b} &= \log_3 n \cdot (d-1) \cdot t_s + n^d/(2 \cdot d \cdot (1 - 1/3^{d-1})) \cdot t_l', \end{aligned}$$

Proof: We consider the last equation. Clearly, in Phase 2 and Phase 3.b, there are performed $\log_3 n$ rounds of $d-1$ steps each. In Step i, $i \leq i \leq d-1$, of Round j, $0 \leq j \log_3 n - 1$ (where Round 0 corresponds to Phase 2), the packets that are sent have weight $n/d \cdot 3^{j \cdot (d-1)+i-1}$. Summing over i and j and using the estimate $\sum_{i=1}^{d-1} 3^{i-1} \leq 3^{d-1}/2$, gives the result. $\square$

Adding together all important contributions gives

Theorem 4 *For gossiping on a d-dimensional torus* CUBGOS *requires about*

$$(d+1) \cdot \log_3 n \cdot t_s + \frac{(3^{d-1}+2) \cdot n^d}{2 \cdot d \cdot (3^{d-1}-1)} \cdot t_l'.$$

This is a very strong and general result, the algorithm is close to optimal for all n, t_s/t_l' and $d \geq 2$:

Corollary 2 *For gossiping on d-dimensional tori,* CUBGOS *is*

$$\max\{1 + 1/d, (1 + 2/3^{d-1})/(1 - 1/3^{d-1})\}\text{-}optimal.$$

Proof: Because, for given d and n, $d \cdot \log_3 n \cdot t_s + n^d/(2 \cdot d) \cdot t_l'$, is a trivial lower bound for just concentrating all data in a single PU, the worst performance ratio is the maximum over all $r = t_s/t_l' > 0$ of

$$\frac{(d+1) \cdot \log_3 n \cdot r + (1 + 2/3^{d-1})/(1 - 1/3^{d-1}) \cdot n^d/(2 \cdot d)}{d \cdot \log_3 n \cdot r + n^d/(2 \cdot d)}.$$

Differentiating for r, gives that the extremal values are assumed for $r = 0$ and $r = \infty$. Substituting these values gives the stated result. □

For $d = 3$, we have $1^3/8$-optimality, and for large d, CUBGOS even almost achieves one-optimality.

7 Experiments

In this section we present experimental results collected on an Intel Paragon XP/S 10 consisting of 138 processing nodes.

System Description. The Paragon system used for the experimentation consists of 138 PUs, each containing a i860 microprocessor and 32 MB of local memory. The mesh is 14 nodes high and 10 nodes wide, with 2 nodes missing in the lower right corner. All experiments were conducted on a configuration of size 9×9.

Some features of the Paragon system used in this study are particularly important in order to understand the performance of our algorithms. Especially

- The Paragon router employs dimension order routing, i.e., packets are first routed along the rows to their destination columns, and from there along the columns to their destinations.
- As shown in Figure 2, the PUs are connected to the communication network by uni-directional, 80 MB/s links. Router chips in the communication network are connected by bi-directional, 120 MB/s links.
- The NX message-passing library supports so-called "synchronous" (blocking) and "asynchronous" (non-blocking) sends and receives. In principle, asynchronous calls allow communication in horizontal and vertical directions simultaneously. However, we found that in many cases asynchronous sends are dispatched one after the other.
- When a message enters its destination before the corresponding receive is posted, the operating system has to buffer the incoming message in a system buffer. When the corresponding receive instruction is issued, the message is copied from the system buffer to the application buffer. This buffering is very expensive and should be avoided whenever possible. Message buffering can be avoided if the recipient first sends a zero-length message to the sender indicating that it has posted the receive. All implementations make use of this synchronization mechanism.

Performance Model. In total eight algorithms were implemented: four one-dimensional (1D) gossiping algorithms and four two-dimensional (2D) variations. The 1D algorithms are obtained by first performing Approach 1 or 2 in the rows, then in the columns. The 2D variants divide the packet initially residing

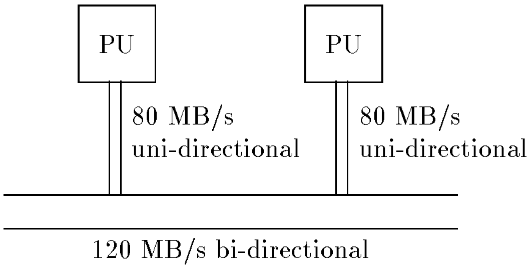


Figure 2: Communication hardware of the Intel Paragon.

in each PU into a white packet and a black packet, and route the white packets orthogonally (and simultaneously) to the black ones.

It is necessary to refine the performance model used in the previous sections. In our performance model it is assumed that each algorithm consists of a sequence of synchronous communication steps. Furthermore, the time taken by a step with messages of m bytes is modeled by the formula

$$T_{\text{step}}(m) = a_{\text{step}} + b_{\text{step}} \cdot m,$$

where a_{step} and b_{step} are constants that depend on the exact step executed. In theory these constants are independent of which step is executed, but in practice they can vary substantially. The reason is that in some steps a node needs to send or receive only one message, whereas in others it may be required to send or receive multiple messages. A similar performance model was used in [1] for the Intel Delta.

The implementations make use of the following basic steps:

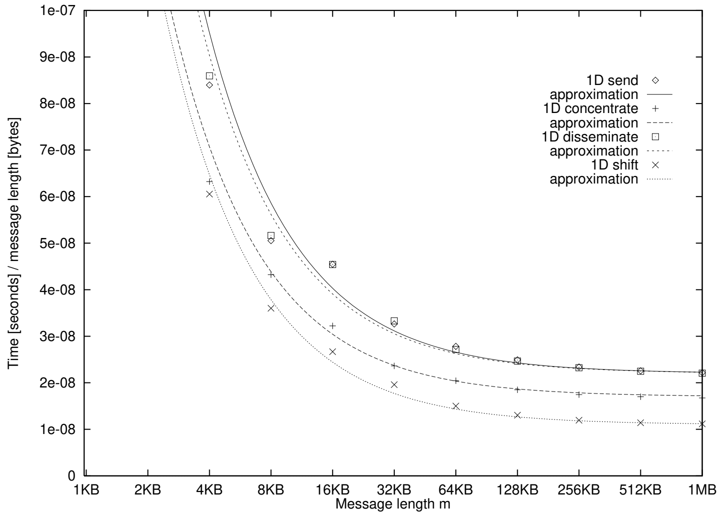
- A *1D-send* step: each node sends a message to its east or south neighbor, but not simultaneously. Nodes in the last row/column send messages to the nodes in the first row/column. This is the basic step used in Approach 1-1.
- A *1D-concentrate* step: data is concentrated in one node. This basic step is used in Approach 2-2 during the concentrate phase.
- A *1D-disseminate* step: a selected node simultaneously sends a messages to two other nodes. This is the basic step during the disseminate phase.
- A *1D-shift* step. In algorithm TORGOS, it is necessary that the packets are concentrated on diagonals. However, because the PUs in the first column do not have an east neighbor, an extra step sending the packets to the selected PUs is required.

These steps differ in the number of messages sent or received by each node. For example, in a 1D-send step each PU sends one message and receives one message. On the other hand, in a 1D-shift step, one PU in each row sends one message and another receives one message. Each step also has a 2-dimensional variant in which communication takes place in both horizontal and vertical directions simultaneously. These steps will be denoted by 2D-send, 2D-concentrate, 2D-disseminate and 2D-shift, respectively.

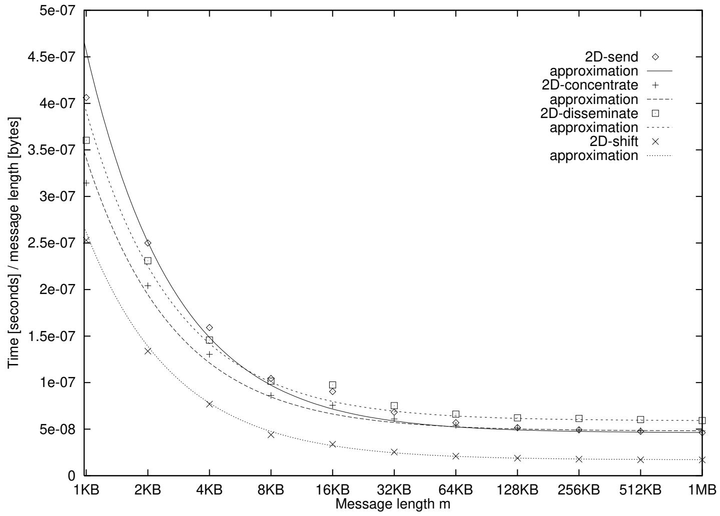
Experimental Results. Figure 3(a) plots the time per byte taken by each 1D basic step as a function of the message length. The total execution time is obtained by multiplying the time per byte by the message length m. By fitting a straight line through the measured data points, we found that the time (in seconds) taken by each 1D basic step with messages of m bytes is approximately given by:

$$\begin{aligned}
T_{\text{1D-send}}(m) &\simeq 3.0 \cdot 10^{-4} + 2.2 \cdot 10^{-8} \cdot m \\
T_{\text{1D-conc}}(m) &\simeq 2.2 \cdot 10^{-4} + 1.7 \cdot 10^{-8} \cdot m \\
T_{\text{1D-diss}}(m) &\simeq 2.8 \cdot 10^{-4} + 2.2 \cdot 10^{-8} \cdot m \\
T_{\text{1D-shift}}(m) &\simeq 2.2 \cdot 10^{-4} + 1.1 \cdot 10^{-8} \cdot m.
\end{aligned}$$

These approximations are also depicted in the figure. It can be seen that the formulae are reasonably accurate (the maximum error is 15%). For the 2D variants, the time taken by each basic step is approximately given by:



(a)



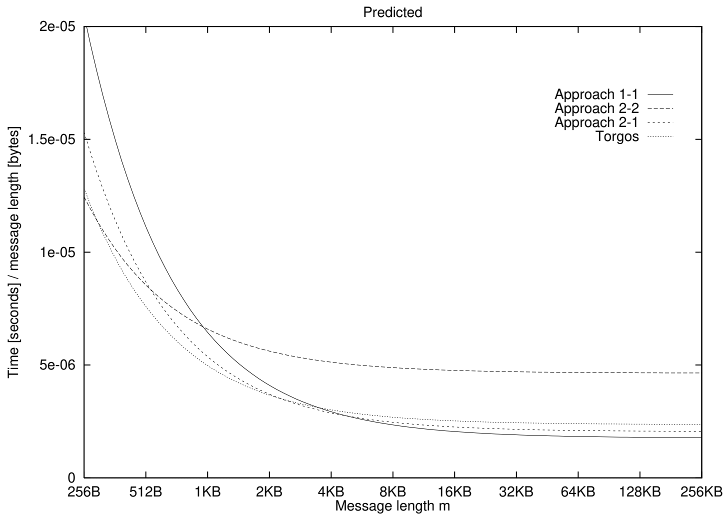
(b)

Figure 3: Time taken by every basic step: 1D operations at the top; 2D operations at the bottom.

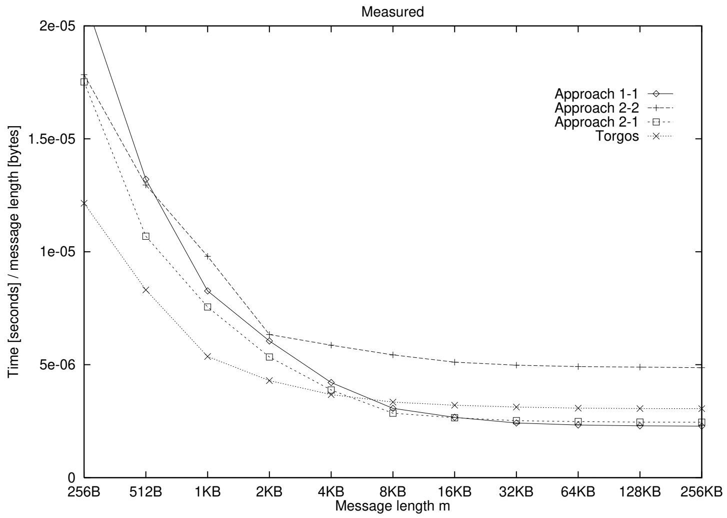
$$
\begin{aligned}
T_{\text{2D-send}}(m) &\simeq 4.2 \cdot 10^{-4} + 4.6 \cdot 10^{-8} \cdot m \\
T_{\text{2D-conc}}(m) &\simeq 3.0 \cdot 10^{-4} + 4.8 \cdot 10^{-8} \cdot m \\
T_{\text{2D-diss}}(m) &\simeq 3.4 \cdot 10^{-4} + 5.9 \cdot 10^{-8} \cdot m \\
T_{\text{2D-shift}}(m) &\simeq 2.5 \cdot 10^{-4} + 1.7 \cdot 10^{-8} \cdot m.
\end{aligned}
$$

Note that the time taken by each 2D basic step more than doubles compared to the 1-dimensional variants, except for the shift basic step. As noted before, it appears that at any given time a processor can be sending a message *or* receiving a message, but not both. However, we do not have an explanation for the fact that the time taken by a 2D basic step is more than twice the amount of time required by the corresponding 1D basic step.

We are now ready to discuss the performance of our gossiping algorithms. The predicted (top figure) and measured (bottom figure) execution times of the 1D implementations are depicted in Figure 4. Again, the total execution times are divided by the message length m in order to capture them in one picture. Every experiment was repeated 25 times and the average was taken. The maximum observed deviation



(a)



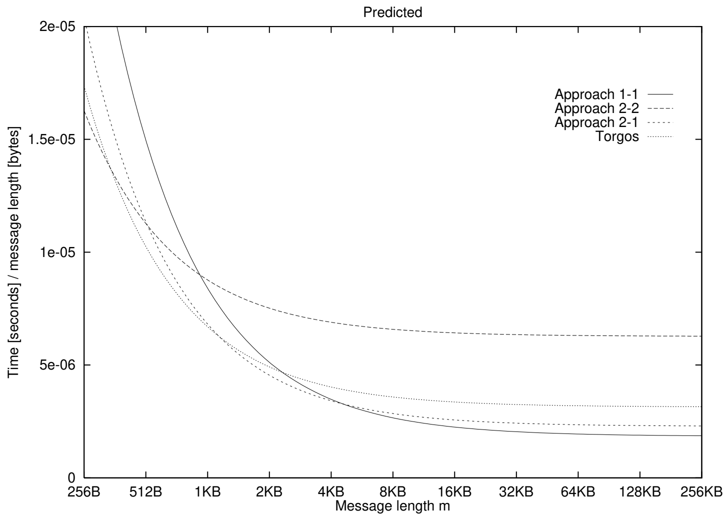
(b)

Figure 4: Predicted and measured execution times of the four one-dimensional gossiping algorithms on a 9×9 mesh.

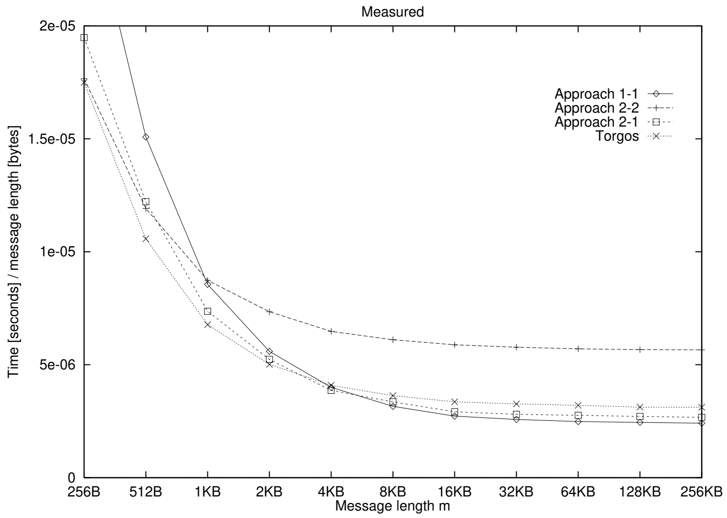
from the mean was about 16%.

Comparing the predicted and actual performance, we find that the implementations run within 20% of the expected execution times. This can be called acceptable. Moreover, the relative performance of the for gossiping algorithms is quite accurately predicted. The model predicts that for very small messages Approach 2-2 is the fastest. However, this is not observed in practice. For messages up to 4 KB, the TORGOS$(3, 3, 1)$ algorithm is faster than any other algorithm. For very long vectors ($>$ 16 KB), Approach 1-1 is the best and Approach 2-2 performs significantly worse than the other approaches, which is in agreement with the predictions. For messages of moderate size (4–16 KB), Approach 2-1 is faster, which is also predicted by the model. Figure 4 clearly demonstrates that in order to obtain a gossiping routine that is efficient for all message sizes, a trade-off has to be made between the number of start-ups and the communication volume. Even on this relatively small network, it is not enough to focus on one goal only.

We also compared the performance of our implementations with the performance of an implementation that uses the global communication routine `gcolx` supplied by Intel. Somewhat surprisingly, our implementations were much faster than the `gcolx` communication routine. For example, for messages of 32 KB, the `gcolx` routine requires 716 milliseconds, while Approach 1-1 requires only 79 milliseconds.



(a)



(b)

Figure 5: Predicted and measured execution times of the four two-dimensional gossiping algorithms on a 9×9 mesh.

We do not have an explanation for the poor performance of the vendor-supplied routine.

The predicted and measured execution times of the two-dimensional gossiping algorithms are shown in Figure 5. Again, we find that the relative performance of the four implemented algorithms is quite accurately predicted. As expected, the 2D implementations do not outperform the 1D algorithms because PUs cannot be sending and receiving data simultaneously. Under ideal circumstances, the time taken by each 1D basic step and its corresponding 2D basic step would be about the same, but this is not observed in practice. We also implemented a 2D variation of Approach 1-1 in which the nodes were divided into white and black nodes (instead of the packets). However, due to rounding effects, this implementation did not outperform the 2D algorithms described here on a configuration of size 9×9. On a 10×10 mesh it was about a factor of 1.4 faster than the corresponding 1D algorithm.

Because the size of the mesh used for experimentation is relatively small, the difference between the various gossiping algorithms is not very large. For larger values of n, these differences will become more apparent. Nevertheless, the experimental results presented in this section support our theoretical claims in most important points.

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 *and* practice for an important range of the involved parameters.

Acknowledgments

Computational support was provided by KFA Jülich, Germany.

References

[1] Barnett, M., R. Littlefield, D.G. Payne, R. van de Geijn, 'Global Combine on Mesh Architectures with Wormhole Routing,' *Proc. 7th International Parallel Processing Symposium*, pp. 13–16, IEEE, 1993.

[2] Delmas, O., S. Perennes, 'Circuit-Switched Gossiping in 3-Dimensional Torus Networks,' *Proc. 2nd International Euro-Par Conference*, LNCS, Springer-Verlag, pp. 370–373, 1996.

[3] Fraignaud, P., J.G. Peters, 'Structured Communication in Torus Networks,' *Proc. 28th Hawai Conference on System Science*, 1995.

[4] Huang, Y., Ph. K. McKinley, 'An Adaptive Global Reduction Algorithm for Wormhole-Routed 2D Mesh Networks,' *Proc. 7th Symposium on Parallel and Distributed Processing*, IEEE, 1995.

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

[6] Kaufmann, M., S. Rajasekaran, J.F. Sibeyn, 'Matching the Bisection Bound for Routing and Sorting on the Mesh,' *Proc. 4th Symposium on Parallel Algorithms and Architectures*, pp. 31–40, ACM, 1992.

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

[8] Ni, L.M., Ph.K. McKinley, 'A Survey of Wormhole Routing Techniques in Direct Networks,' *IEEE Computer*, 26(2), pp. 62–76, 1993.

[9] Peters, J.G., M. Syska, 'Circuit-Switched Broadcasting in Torus Networks,' *IEEE Transactions on Parallel and Distributed Systems*, to appear.

[10] Reif, J., L.G. Valiant, 'A Logarithmic Time Sort for Linear Size Networks,' *Journal of the ACM*, 34(1), pp. 68–76, 1987.

[11] Rao, P.S., G. Mouney, 'Data Communications in Parallel Block Predictor-Corrector Methods for solving ODEs,' *Techn. Rep.*, LAAS-CNRS, France, 1995.

[12] Rao, P.S., D. Trystram, 'Block Predictor(low order) Corrector Methods to solve Ordinary Differential Equations on Parallel Computers,' *Research Rep. IMAG-RR-757-M-*, TIM3, Institute IMAG, France, December 1988.

[13] Sibeyn, J.F., 'Sample Sort on Meshes,' *Techn. Rep. MPI-I-95-1012*, Max-Planck Institut für Informatik, Saarbrücken, Germany, 1995.

[14] Tseng, Y-C., T-H. Lin, S.K.S. Gupta, D.K. Panda, 'Bandwidth-Optimal Complete Exchange on Wormhole-Routed 2D/3D Torus Networks: A Diagonal-Propagation Approach,' submitted to *IEEE Transactions on Parallel and Distributed Systems*. Preliminary version appeared in *Proc. 9th International Parallel Processing Symposium*, pp. 532–536, IEEE, 1995.