Worm-Hole Gossiping on Meshes[*]

Ben H.H. Juurlink[†] P.S. Rao[‡] Jop F. Sibeyn[§]

Abstract

Several 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, we give a novel lower bound and an asymptotically optimal gossiping algorithm. For two-dimensional meshes, we present a simple algorithm composed of one-dimensional phases. For an important range of packet and mesh sizes, it gives clear improvements. The algorithm is analyzed theoretically, but, the achieved improvements are also convincingly demonstrated by simulations and by an implementation on the Paragon. For higher dimensional meshes, we give algorithms which are based on a generalized notion of a diagonal.

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. Numerous parallel machines with mesh topologies have been built.

Gossiping. Gossiping is a fundamental communication problem. It appears in many contexts, both theoretical and practical. Gossiping is the problem in which each of the N PUs needs to send data to every other PU. Finally, all PUs must know the complete data of size $N \cdot L$. This is a very communication intensive operation.

Gossiping appears as a subroutine in many important problems. For example, if M numbers are to be sorted on N PUs, then a good approach is to select a set of m splitters [8, 5] which must be made available in every PU. This means that we have to perform a gossip in which every PU contributes m/N numbers. In this case the amount of data is small and, hence, the gossiping time can be made negligible with efficient gossiping algorithms. A second application of gossiping appears in algorithms for solving ordinary differential equations using parallel block predictor-corrector methods [9]. In each application of the block method, computations corresponding to the prediction are carried out by different PUs and these values are needed by all other PUs.

Earlier Work. A substantial amount of research has been performed on (variants of) the gossiping problem [1, 2, 7]. In some sense, we turn back to basics. Rather than to design an even more sophisticated algorithm, along the lines of [7], 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 and routing time. For relatively large messages, it is not enough to focus on the number of start-ups only. A non-trivial lower bound shows that our algorithms are close to optimal for *all* 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.

[*] Part of this research was performed during a stay of the second author at the Max-Planck-Institute

[†] Dept. of Computer Science, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands. Email: benj@cs.LeidenUniv.nl

[‡] Dept. of Computer Science, University of Hyderabad, Hyderabad - 500 046, India. Email: psraocs@uohyd.ernet.in

[§] Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. Email: jopsi@mpi-sb.mpg.de

2 Preliminaries and Lower Bounds

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. We assume the full-port model in which a PU can transmit data to all of its neighbors simultaneously.

For the communication we assume the much considered worm-hole routing model (see [6, 3] for some recent surveys). In this model a packet consists of **flits** and has a header which contains the necessary routing information. The other flits just follow the header. Initially all flits reside in the source PU. Finally all flits should reside in the destination PU. Furthermore, two or more flits may reside in the same PU only at the source and the destination. 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, the time to issue a packet is considerably larger than the time needed to traverse a connection. The time to send a packet consisting of l flits over a distance of c connections is given by

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

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 as long as the paths of various packets do not overlap. Our algorithms are overlap-free.

We start with a trivial but general lower bound.

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:*

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

Of course, T_{con} immediately gives a lower bound for the gossiping problem. A stronger lower bound is given in the following theorem. The proof of this theorem is given in [4].

Theorem 1 *Let $r = t_s/(l \cdot t_l)$ where $r \leq n/e^2$. The time for gossiping on a linear array with n PUs satisfies*

$$T_{gos} = \Omega(n \cdot \ln n/\ln(n/r) \cdot l \cdot t_l).$$

3 Linear and Circular Arrays

We analyze gossiping on one-dimensional processor arrays. We assume 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 the algorithms for circular arrays. With minor modifications, all of them carry on for linear arrays.

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. Recursively concentrate the data into a selected PU. Then, reverse the process to disseminate the information to all other PUs.

Lemma 2 *If the packets consist of l flits each, then Approach 1 takes $T_1(n, l) = \lfloor n/2 \rfloor \cdot (t_s + t_d + l \cdot t_l)$ time.*

Lemma 3 *If the packets consist of l flits each, then the time consumption of Approach 2 can be estimated on $T_2(n, l) \simeq \log_3 n \cdot (2 \cdot t_s + n \cdot l \cdot t_l)$.*

Proof: During the concentration phase, the number of 'active' PUs is reduced by a factor of three in every step, the packets get three times as heavy and the distance over which the packets have to be sent increases by a factor of three. This gives $T_{\text{conc}} = \sum_{i=0}^{\log_3 n-1}(t_s + 3^i \cdot (t_d + l \cdot t_l)) < \log_3 n \cdot t_s + n/2 \cdot (t_d + l \cdot t_l)$. In all steps of the dissemination phase, the packets consist of $n \cdot l$ flits each, giving $T_{\text{dis}} = \sum_{i=0}^{\log_3 n-1}(t_s + 3^i \cdot t_d + l \cdot n \cdot t_l) < \log_3 n \cdot (t_s + n \cdot l \cdot t_l) + n/2 \cdot t_d$. Since t_d is of the same order as t_l, the term $n/2 \cdot t_d$ can be ignored. $\square$

$n \diagdown t_s/t_l'$	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'.

If $t_s \gg l \cdot t_l$, then for all reasonable values of n, the result of Lemma 3 cannot be improved. However, note that *any ratio* $t_s/(l \cdot t_l)$ *is possible*. For example, if a large sorting problem is solved on a relatively small system, then the packets consist of many flits. In that case it may even happen that $l \cdot t_l > t_s$. For such instances, we propose an approach which has features of both basic approaches.

We henceforth neglect the distance term, which is of minor importance anyway, and write $t_l' = l \cdot t_l$. The algorithm consists of three phases, and works with parameters a and b.

Algorithm CIRCGOS(a, b)

1. Concentrate n/a data in a evenly interspaced PUs, called *bridgeheads* 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 every concentration point.

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. Notice that the algorithm becomes equal to Approach 1 for $a = n$.

Lemma 4 *The three phases of* CIRCGOS(a, b) *take*

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

The best choices for a and b have been found by a simple computer program. Table 1 lists 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 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 that this knowledge is not exploited).

- The range of t_s/t_l' values for which CIRCGOS is the best increases with n. The best choices of a and b increase with n and decrease with t_s/t_l'.

Theorem 2 *Let $r = t_s/t'_l$ where $r < n$. The time consumption of* CIRCGOS*($n/r, n/r$) is given by*

$$T_{cg}(n/r, n/r) = \mathcal{O}(n \cdot \ln n / \ln(n/r) \cdot t'_l).$$

Thus, CIRCGOS($n/r, n/r$) is asymptotically optimal (cf. Theorem 1), and 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 $r = n$.

Corollary 1 *Let $r = t_s/t'_l$ and $0 < \epsilon < 1$. For all r, $\log n \leq r \leq n^\epsilon$,* CIRCGOS *is about a factor of $\log n$ faster than Approach 1 and 2.*

Proof: For $r \geq \log n$, Approach 1 and Approach 2 both take $\Omega(n \cdot \log n \cdot t'_l)$ time. On the other hand, for $r = n^\epsilon$, CIRCGOS takes $\mathcal{O}(n \cdot \log n / \log(n^{1-\epsilon}) \cdot t'_l) = \mathcal{O}(n \cdot t'_l)$ time. $\square$

4 Two-Dimensional Arrays

The simplest idea for gossiping on two-dimensional (2D) tori is to send the packets first along the rows and then along the columns, choosing the best of Approach 1 and Approach 2 in each phase. A factor of two is gained when the packets in PUs (x, y) with $x + y$ even are colored 'white' and 'black' otherwise, and by routing the black packets orthogonally to the white ones. Let Approach i-j denote the algorithm in which first Approach i is applied and then Approach j, and let $T_{i,j}$ denote the time taken by Approach i-j. Approach 1-2 can be excluded.

Lemma 5 *The time consumption of the three gossiping algorithms is given by*

$$\begin{aligned} T_{1,1} &\simeq 3/4 \cdot n \cdot t_s + n/4 \cdot (n+1) \cdot t'_l, \\ T_{2,1} &\simeq (2 \cdot \log_3(n/2) + n/2) \cdot t_s + n/2 \cdot (\log_3(n/2) + n/2) \cdot t'_l, \\ T_{2,2} &\simeq (4 \cdot \log_3 n - 3) \cdot t_s + (2 \cdot \log_3 n - 2) \cdot n/2 \cdot (n/2 + 1) \cdot t'_l. \end{aligned}$$

The described approaches are competitive for many choices of n, t_s and t'_l, but a more truly 2D approach gives considerably better results for intermediate r values. The algorithm is a 2D analogue of CIRCGOS. We may concentrate on the white packets. The black packets are routed orthogonally to the white ones.

Algorithm TORGOS(a, b, x)

1. Concentrate all white packets in a concentration points of their rows; in row i, the PUs (i, j) with $(j - i) \bmod (n/a) = 0$. After this phase, each concentration point holds $n/(2 \cdot a)$ white packets.

2. Route the data 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 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, x and t such that $b^t = n/a$ and $x \geq \lfloor b/2 \rfloor$. Perform t rounds of further concentration. At the beginning of round j, $0 \leq j < t$, the concentration points contain $S_j = a \cdot b^j \cdot n/2$ white packets.

  a. Divide the data into packets of size $S_j/(2 \cdot x - b + 2)$. Route these for x steps along the rows, to $b - 1$ points equally interspaced between any two concentration points.

  b. Perform $\lfloor b/2 \rfloor$ steps of vertical routing with packets of size S_j.

Phase 1 is performed by a repeated concentration in $\log_3(n/a)$ steps. After this phase, 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).

n	$t_s/t_l = 8$		$t_s/t_l = 30$		$t_s/t_l = 100$		$t_s/t_l = 250$	
	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 four gossiping algorithms on $n \times n$ 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$(3, 3, 1)$. In the last 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'.

Lemma 6 *The phases of* TORGOS(a, b, x) *take time*

$$\begin{aligned}
T_{\textit{tg, 1}} &= \log_3(n/(2\cdot a))\cdot t_s + n/(4\cdot a)\cdot t_l', \\
T_{\textit{tg, 2}} &= \lfloor a/2\rfloor \cdot (t_s + n/(2\cdot a)\cdot t_l'), \\
T_{\textit{tg, 3}} &= \lfloor a/2\rfloor \cdot (t_s + n/2\cdot t_l'), \\
T_{\textit{tg, 4.a}} &\simeq x\cdot(\log_b(n/a)\cdot t_s + n^2/(2\cdot(b-1)\cdot(2\cdot x - b + 2))\cdot t_l'), \\
T_{\textit{tg, 4.b}} &= \lfloor b/2\rfloor \cdot (\log_b(n/a)\cdot t_s + n^2/(2\cdot(b-1))\cdot t_l').
\end{aligned}$$

In Table 2 we give some numerical results. 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.

5 Higher Dimensions

For the success of the two-dimensional algorithm it was essential that the packets were concentrated on diagonals at all times. 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.

The property of a two-dimensional diagonal that must be generalized, is the possibility of ‘seeing’ a full 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 $[0, 1] \times 0$. For 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:*

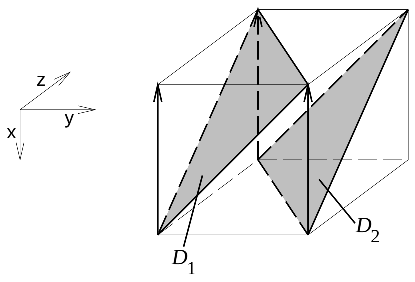

Figure 1: Diagonal of the unit cube; the projection along the x-axis is the set $0 \times [0,1] \times [0,1]$.

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 $i = 1, 2, \ldots, d-1$, let $\mathcal{D}_i = \{x_0, x_1, \ldots, x_{d-1} | x_0 + x_1 + \cdots + x_{d-1} = i\}$. For the **unit-cube** $I^d = [0,1] \times \cdots \times [0,1]$, the union of the intersections of the $\mathcal{D}_i$-hyperplanes with I^d gives a diagonal:

Lemma 7 *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 $\Pi_0(\mathcal{S})$. Then, for all $1 \leq i \leq d$,

$$\Pi_0(\mathcal{D}_i) = \{x_0, x_1, \ldots, x_{d-1} | x_0 = 0, i-1 \leq x_1 + \cdots + x_{d-1} \leq i\}. \tag{2}$$

So, for any $i < j$,

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

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

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

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 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 7.

We now describe the algorithm for d-dimensional tori. Each packet is given a color from $\{0, 1, \ldots, d-1\}$; the packets in PU $(x_0, x_1, \ldots, x_{d-1})$ are 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.

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. For $d = 2$, CUBGOS is identical to TORGOS(3, 3, 1).

Lemma 8 *The phases of* CUBGOS*(n, d) take time*

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

Proof: We consider the last equation. Clearly, in Phase 2 and Phase 3.b, $\log_3 n$ rounds of $d-1$ steps each are performed. In Step i, $0 \leq i \leq d-1$, of Round j, $0 \leq j \leq \log_3 n - 1$ (where Round 0 corresponds to Phase 2), the packets 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 all important contributions together gives

Theorem 3 *The time consumption of* CUBGOS *is given by*

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

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

For $d = 2$, we have $5/2$-optimality, and for large d, CUBGOS almost achieves one-optimality.

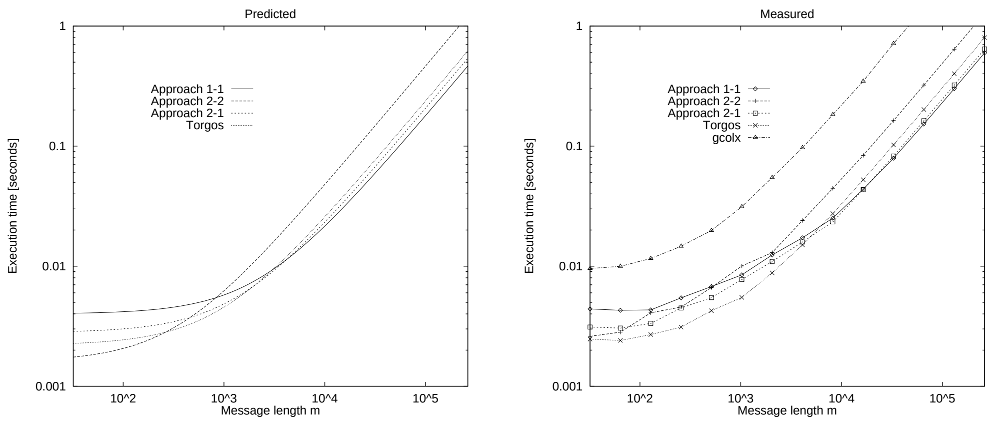

Figure 2: Predicted and measured execution times of the four one-dimensional gossiping algorithms on a 9×9 mesh.

6 Experiments

In this section, we present experimental results collected on an Intel Paragon XP/S 10 configuration of 9 nodes high and 9 nodes wide. A fuller account of the experimental data presented in this section is provided in [4].

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

Note that because the experimental platform is a mesh, not a torus, the runtime analysis slightly changes. For example, the time taken by Approach 1-1 for the 1D case is given by $T_{1,1} = (n-1) \cdot (t_s + t_l') + (n-1) \cdot (t_s + n \cdot t_l')$. Furthermore, it is necessary to refine the performance model used in the previous sections. In particular, we need to distinguish between several basic steps. The reason is that in some steps, a node needs to send or receive one message, whereas in others it may be required to send or receive multiple messages, introducing bus conflicts. For example, a *1D-send* step in which a node sends a message to its east or south neighbor (but not simultaneously), takes about $2.5 \cdot 10^{-4} + 2.2 \cdot 10^{-8} \cdot m$ seconds, where m is the message length in bytes. On the other hand, a *2D-concentrate* step in which a node receives data from all its neighbors simultaneously, requires approximately $3.0 \cdot 10^{-4} + 4.8 \cdot 10^{-8} \cdot m$ seconds.

Figure 2 plots the measured and predicted performance of the 1D implementations. The curves on the left show the predicted execution times. The curves on the right plot the measured times. For large messages, we find that the implementations run within 20% of the expected time. The error is probably due to the fact that processors operate asynchronously. However, the relative performance 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, TORGOS$(3,3,1)$ 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.

Figure 2 also compares the performance of our algorithms with the performance of an implementation that uses the global communication routine `gcolx` supplied by Intel. Clearly, our implementations are significantly 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.

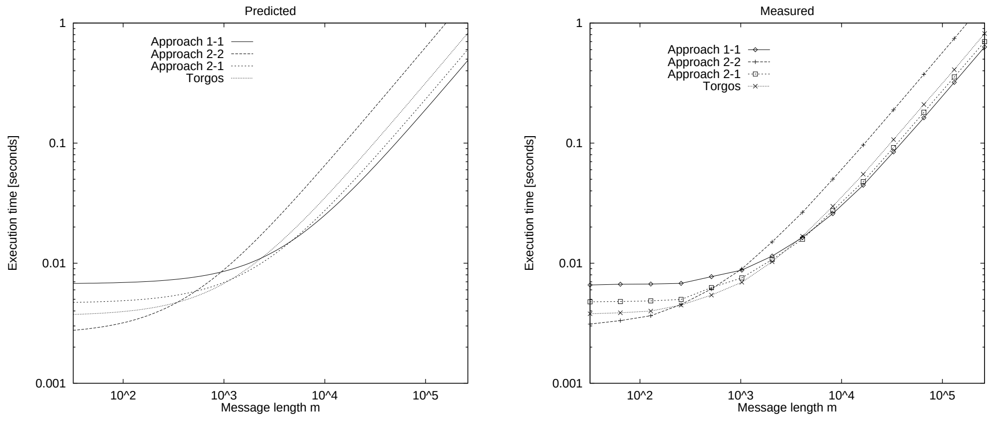

Figure 3: Predicted and measured execution times of the two-dimensional gossiping algorithms on a 9 × 9 mesh.

The predicted and measured execution times of the 2D variants are shown in Figure 3. As expected, these implementations do not outperform the 1D algorithms due to bus conflicts. Under ideal circumstances, the time by each 1D basic step and its corresponding 2D basic step would be about the same, but this is not observed in practice.

7 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.

Acknowledgment. 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 IPPS*, pp. 13–16, IEEE, 1993.

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

[3] Huang, Y., Ph. K. McKinley, 'An Adaptive Global Reduction Algorithm for Wormhole-Routed 2D Mesh Networks,' *Proc. 7th SPDP*, IEEE, 1995.

[4] Juurlink, B., P.S. Rao, J.F. Sibeyn, 'Gossiping on Meshes and Tori,' *Techn. Rep. MPI-I-96-1018*, Max-Planck Institut für Informatik, Saarbrücken, Germany, 1996.

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

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

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

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

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