Time-Independent Gossiping on Full-Port Tori

Ulrich Meyer and Jop F. Sibeyn
Max-Planck-Institut für Informatik
Im Stadtwald, 66123 Saarbrücken, Germany.
E-mail: umeyer, jopsi@mpi-sb.mpg.de.
WWW: http://www.mpi-sb.mpg.de/∼umeyer, ∼jopsi/

June 25, 1998

Abstract

Near-optimal gossiping algorithms are given for two- and higher dimensional tori. It is assumed that the amount of data each PU contributes is so large that start-up time may be neglected. For two-dimensional tori, a previous algorithm achieved optimality in an intricate way, with a time-dependent routing pattern. In all steps of our algorithms, the PUs forward the received packets in the same way.

1 Introduction

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

Tori are the variant of meshes in which the PUs on the outside are connected with "wrap-around links" to the corresponding PUs at the other end of the mesh. Tori are node symmetric. Furthermore, the bisection width for tori is twice as large as for meshes, and the diameter is halved. Numerous parallel machines, such as the Intel Paragon, Cray T3D and Cray T3E, have been built with two- and three-dimensional mesh and torus topologies Tori can be embedded in meshes with load 1, dilation 2 and congestion 2. That is, for all those algorithms that run twice as fast on tori as on meshes, one can assume without loss of performance that the network is a torus.

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

Gossiping appears as a subroutine in many important problems. For example, if M numbers are to be sorted on P PUs, then a good approach is to select a set of m splitters [12, 8] which must be made available in every PU. This means that we have to perform a gossip in which each of the P PUs contributes $s = m/P$ numbers. A second application of gossiping appears in algorithms for solving ordinary differential equations using parallel block predictor-corrector methods [11]. In each application of the block method, computations corresponding to the prediction are carried out by different PUs and these values are required by all other PUs.

Earlier Work. A substantial amount of research has been carried out on variants of the gossiping problem [10, 3, 6, 9, 4, 7]. Recently Šoch and Tvrdík [14] have analyzed the following variant of the gossiping problem:

- Packets of size s can be transferred in one *step* between adjacent PUs (store-and-forward model).

- In each step a PU can exchange packets with all its neighbors (full-port model).

The assumption that it takes one step to transfer a packet is debatable. Most modern parallel computers, such as the Intel Paragon, the Cray T3D and T3E, are characterized by a high start-up latency. This means that if the number of bytes s contributed by each PU is small (on the Paragon this means $s \cdot P < 10{,}000$) the gossiping time will be determined mainly by the number of times a PU sends a packet. In that case, the primary goal should be to minimize the number of sending operations [9, 4]. In an intermediate range of s-values (on the Paragon up to about $s = 20{,}000$), one should establish an optimal trade-off between the number of sending operations and the *routing volume*, the number of bytes sent by each PU [7]. For even larger s, the start-up latency becomes negligible and one can focus entirely on minimizing the routing volume. For such large values of s, it is correct to assume that it takes one step for every transferred packet of size s.

In [14], it was shown that on a two-dimensional $n_1 \times n_2$ torus, the given problem can be solved in $\lceil (n_1 \cdot n_2 - 1)/4 \rceil$ steps if $n_1, n_2 \geq 3$. This is optimal. The algorithm is based on time-arc-disjoint broadcast trees. This implies that the action to be performed by a PU is time-dependent, and thus, that for every routing decision, a PU has to perform some non-trivial computation. Furthermore, there are many cases to distinguish, and the description is mainly in the form of pictures. It is stated that a similar construction works for higher-dimensional meshes, but the description would be much more complicated and is omitted. We think these weaknesses are inherent in the time-dependent approach.

New Results. In this paper, we analyze the same problem as Šoch and Tvrdík. Clearly, we cannot improve their optimality. Instead, we try to determine the minimal concessions that must be made to obtain a time-independent algorithm. That is, we are aiming for algorithms in which a PU performs the same actions in each step: In our gossiping algorithms, after some $\mathcal{O}(d)$ precomputation on a d-dimensional torus, a PU knows once and for all that packets coming from direction x_i have to be forwarded in direction x_j, $1 \leq i, j \leq d$. Time-independence ensures that the routing can be performed with minimal delay, for a fixed size network the pattern might even be built into the hardware. This is also advantageous on a system in which the connections must be somehow switched.

However, in Section 2 we will show that with one packet per PU one cannot achieve the optimal number of steps with a time-independent algorithm. Thus, to obtain time-independent algorithms, we must make some concessions in comparison with [14]. Our first approach is to try to achieve the optimal number of steps while assuming that the PUs hold $k > 1$ packets. The optimal number of steps, means a routing time of $k \cdot P/(2 \cdot d)$ steps on a torus with P PUs. For d-dimensional tori this is achieved with $k = d$ by routing the packets along edge-disjoint Hamiltonian cycles.

Our second approach, which is the more interesting one, is to accept that the gossiping takes an additional $o(P)$ routing steps. For $d = 2$, the algorithm is particularly simple and needs $n_1 \cdot n_2/4 + n_1/2 + 1$ steps, which is only $n_1/2$ steps more than optimal. Therefore, this algorithm might be preferable over the one from [14]. Generally, we show that, with one packet per PU, gossiping can be performed in $P/(2 \cdot d) + o(P)$ steps on a d-dimensional torus. In these algorithms, we construct partial Hamiltonian cycles: on a d-dimensional torus, we construct d cycles, which each cover $P/d + o(P)$ PUs. These are so that for every cycle, every PU is adjacent to a PU through which this cycle runs. In particular, for the practically relevant case $d = 3$, this gives the first simple and explicit construction achieving close to optimally: on an $n_1 \times n_2 \times n_3$ torus, it requires $n_1 \cdot n_2 \cdot n_3/6 + n_1 \cdot n_2/2 + 1$ steps.

Practical Aspects Modern parallel computers mostly apply *worm-hole routing*. This means that packets are transferred between their origin and destination as a stream of "flits". As long as there is no congestion, the time for such a transfer is more or less independent of the distance. For a packet of size k, it can be written as $t_s + k \cdot t_f$.

For sufficiently large packets, the "start-up time" t_s becomes negligible, and the time depends almost exclusively on s. This suggests that it does not matter much how origins and destinations are chosen: on a network with P PUs, one could gossip by letting PU_i transfer its packet to $PU_{(i+t) \bmod P}$, in Round t, $1 \leq t < P$.

On small networks, this may be a good approach, but in general it will lead to substantial losses due to congestion. For personalized communication there is not much to do about this (see [13]), but if the packet to be sent is the same for all PUs, it is much better to apply a strategy in which packets are routed only between neighbors.

There is nothing magic about packets, and as long as they are so large that the start-up time remains negligible, we may split them up or recombine them at will. That is why we may assume that initially each PU holds some number k of packets: this only means that they have k size s/k instead of s, and that thus the start-up costs play a slightly larger role. As long as k is small, this will be no problem.

Contents. In Section 3, we describe the optimal-time gossiping algorithm for two-dimensional tori and outline its generalizations for higher dimensional meshes. Then, in Section 4, we describe near-optimal-time algorithms that require only one packet per PU. Finally, in Section 5, we numerically compare the performance of the gossiping algorithms of this paper and determine their range of optimality.

2 Lower-Bound

We show that it is not possible to construct two edge-disjoint Hamiltonian cycles so that a time-independent algorithm with one packet per PU can perform the gossiping in $\lceil (P-1)/4 \rceil$ steps by concurrently routing the packets along those cycles.

So, in the remainder of this paper we assume that there are two edge-disjoint Hamiltonian cycles. We may assume that the PUs are indexed so that the first cycle visits the PUs in consecutive order: $0, 1, \ldots, P-2, P-1$. Thus, PU i will receive the packets from PU $(i-j) \bmod P$ and PU $(i+j) \bmod P$ in Step j, $0 < i, j < P$. In the same step, PU i will also receive the packets from PU i_{-j} and PU i_{+j} where $\ldots, i_{-2}, i_{-1}, i_0, i_{+1}, i_{+2}, \ldots$ denotes the order of the second cycle relative to PU i. Let $k_i = |\mathcal{A}_i \cap \mathcal{B}_i \setminus \{i\}|$, where $\mathcal{A}_i = \{(i - P/4) \bmod P, \ldots, (i + P/4) \bmod P\}$ and $\mathcal{B}_i = \{i_{-P/4}, \ldots, i_{+P/4}\}$. k_i gives the number of multi-receives by PU i during the first $P/4$ steps, thus at least $k_i/4$ additional steps are necessary to complete the gossiping.

We prove that an i with $k_i \geq P/10$ exists. Choose i arbitrarily. If $k_i \geq P/10$, we are done. Otherwise at least $P/2 - P/10 = 2/5 \cdot P$ elements from $\mathcal{B}_i$ have indices in $\overline{\mathcal{A}_i} = \{0, \ldots, P-1\} \setminus \mathcal{A}_i$. Let $\mathcal{C}_i = \overline{\mathcal{A}_i} \cap \mathcal{B}_i$. The elements of $\mathcal{C}_i$ are sorted, and then cyclically shifted so that the smallest element larger than i comes first. Out of this arrangement the median is selected and assigned to i'. For this i', we will show that $k_{i'} \geq P/10$, but first we illustrate the definitions given with an example.

Example 1 *Consider the case* $P = 40$ *and* $i = 22$. $\mathcal{A}_{22} = \{12, \ldots, 22, \ldots, 32\}$ *and* $\overline{\mathcal{A}_{22}} = \{33, \ldots, 39, 0, \ldots, 11\}$. *Suppose that*

$$\mathcal{B}_{22} = \{35, 11, 7, 4, 34, 8, 25, 38, 9, 6, 22, 20, 1, 17, 33, 3, 5, 10, 36, 0, 39\}.$$

Hence, $\mathcal{A}_{22} \cap \mathcal{B}_{22} \setminus \{22\} = \{17, 20, 25\}$, *and thus* $k_{22} = 3$. *For* $\mathcal{C}_{22} = \overline{\mathcal{A}_{22}} \cap \mathcal{B}_{22}$ *we have*

$$\mathcal{C}_{22} = \{33, 34, 35, 36, 38, 39, 0, 1, 3, 4, 5, 6, 7, 8, 9, 10, 11\},$$

and $|\mathcal{C}_{22}| = 17$. *For* i' *we now take* $i' = 3$, *which is the median of* $\mathcal{C}_{22}$ *in the given arrangement, in which 33, the smallest element larger then 22, comes first.* $\mathcal{A}_3 = \{33, \ldots, 39, 0, \ldots, 13\}$. $\mathcal{B}_3$ *is not exactly known, but in any case*

$$\mathcal{B}_3 \supset \{8, 25, 38, 9, 6, 22, 20, 1, 17, 33, 3, 5, 10, 36, 0, 39\},$$

Hence, $\mathcal{A}_3 \cap \mathcal{B}_3 \supset \{8, 38, 9, 6, 1, 33, 3, 5, 10, 36, 39\}$, *and thus* $k_3 \geq 11$.

Theorem 1 *A time-independent gossiping on two edge-disjoint Hamiltonian cycles with one packet per PU requires at least* $\lceil (11/40) \cdot P - 1 \rceil$ *steps.*

Proof: We only have to deal with the case $k_i < P/10$, so that i' must be selected. The particular choice of i' and the position of i' on the second cycle determines how many elements of $\mathcal{C}_i$ belong to $\mathcal{A}_{i'}$. In the best case, PU i' is followed on one side of the second cycle by at least $P/4$ PUs storing elements not belonging to $\mathcal{A}_{i'}$. Let us assume this is the case on the left side, for example, $\{i'_{-P/4}, \ldots, i'_{-1}\} \cap \mathcal{A}_{i'} = \emptyset$. This situation might occur for example when i' was just located at the left end of $\mathcal{B}_i$, $i' = i_{-P/4}$. Then superfluous receives for PU i' may only arrive from the right side.

We have $|\mathcal{C}_i| \geq 2/5 \cdot P$, thus in the best case the first $P/2 - 2/5 \cdot P = P/10$ elements received from the right side of the second cycle may not belong to $\mathcal{A}_{i'}$ as well. After that, the number of multi-receives is minimized if the largest elements of $\mathcal{C}_i$ (arranged so that the smallest element larger then i comes first) are transferred first. Again, in the best case $\mathcal{C}_i$ consists of two chunks of elements: the large elements $\{(i' - |\mathcal{C}_i| + P/2) \bmod P, \ldots, (i' - |\mathcal{C}_i|/2 + P/2) \bmod P\}$ on the one hand, and the small elements $\{(i' - |\mathcal{C}_i|/2) \bmod P, \ldots, i'\}$ on the other hand. Note that the median i' is just the largest element among the chunk of small elements in $\mathcal{C}_i$ (when arranged as before). But any of those elements in $\mathcal{C}_i$ which are smaller than $(i' + P/4) \bmod P$ will fit into $\mathcal{A}_{i'}$. So, even in the best case, after at most $i' - |\mathcal{C}_i|/2 + P/2 - (i' + P/4) = P/20$ more steps, the elements received from the right are small enough to belong to $\mathcal{A}_{i'}$. Altogether, during the first $P/4$ steps PU i' receives on the second cycle at least $P/4 - P/10 - P/20 = P/10$ elements fitting into $\mathcal{A}_{i'}$. □

3 Optimal-Time Algorithms

In this section we first present optimal gossiping algorithms for two-dimensional $n_1 \times n_2$ tori. We assume that each PU initially holds two packets. The transfer of a packet between two adjacent PUs takes one step. We show that the gossiping can be performed in $n_1 \cdot n_2/2$ steps. We distinguish several cases, depending on the parity of n_1 and n_2. Finally we indicate how this idea can be generalized for higher dimensional tori.

3.1 Two-Dimensional Tori, n_1 and n_2 Even

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

$$\begin{aligned} j < n_2 - 1, j \text{ even} &\quad : \quad T \leftrightarrow R; B \leftrightarrow L. \\ j < n_2 - 1, j \text{ odd} &\quad : \quad T \leftrightarrow L; B \leftrightarrow R. \end{aligned}$$

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

$$j = n_2 - 1 \quad : \quad T \leftrightarrow R; B \leftrightarrow L.$$

The resulting routing scheme is illustrated in Figure 1.

Theorem 2 *If every PU of an* $n_1 \times n_2$ *torus, with* n_1 *and* n_2 *even, holds 2 packets, then gossiping can be performed in* $n_1 \cdot n_2/2$ *steps.*

Proof: Initially every PU knows two packets. In each step, it receives four new packets. Only in the last step, a PU receives twice the same two packets. □

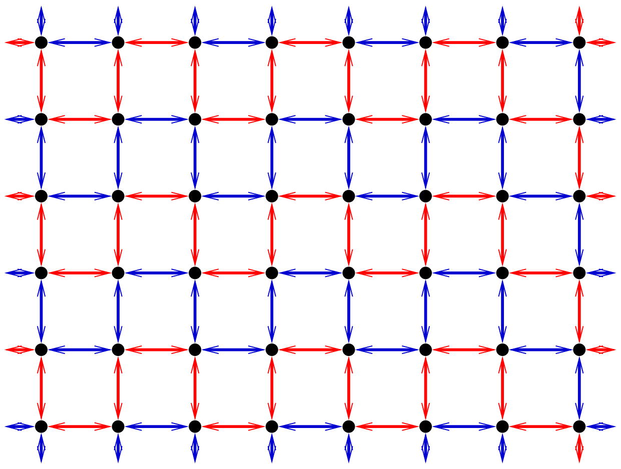

Figure 1: A Hamiltonian cycle on a 6×8 torus (blue), whose complement (drawn with red lines) also gives a Hamiltonian cycle.

3.2 Two-Dimensional Tori, n_1 Odd and n_2 Even

If n_1 or n_2 are odd, we must use a slightly modified schedule. In this section we consider only the case n_1 odd and n_2 even. The case n_1 even and n_2 odd can be solved in a similar way. Here we do not construct complete Hamiltonian cycles, but cycles that visit most PUs, and pass within distance one from the remaining PUs. Except for Column $n_1 - 2$, the rules governing how to pass on the packets are the same as in the basic case. In Column $n_1 - 2$ we perform

$$i = n_1 - 2 \quad : \quad L \leftrightarrow R.$$

PUs which do not lie on a given cycle, out-of-cycle-PUs, abbreviated *OOC-PUs*, are provided with the

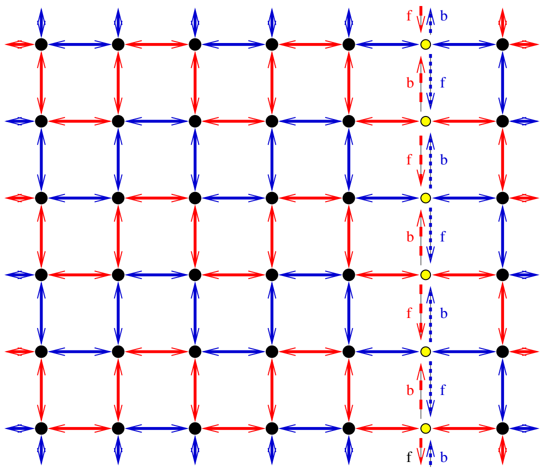

Figure 2: Partial Hamiltonian cycles on a 6×7 torus. The PUs in column 5 lie on only one cycle. Such a PU passes the packets that are running forward on this cycle to the PU below it, and those that are running backward to the PU above it.

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

The optimal performance in the basic case (n_1, n_2 even) was achieved by using both directions on every cycle, thus halving the circulation time. The inclusion of OOC-PUs complicates this approach: We need two different OC-PUs, A and B, in order to provide an OOC-PU C with packets from both directions of that cycle. Let A transfer the packets that are walking forward whereas B is responsible for the packets going backward. If m denotes the cycle length and A and B are not adjacent on the cycle, then $m/2$ circulation steps are not enough: some packets are received from both directions, while others pass by without notice. Figure 3 gives an example.

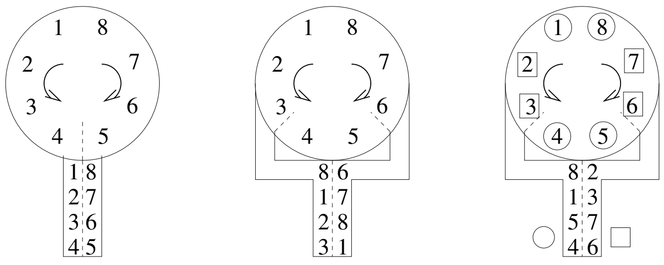

Figure 3: Packets received by an OOC-PUs. **Left:** If the relevant OC-PUs are adjacent, then 4 steps are sufficient to receive all packets. **Middle:** Otherwise multiple receives occur. **Right:** a solution to this problem by adequate packet classification.

This problem does not arise if there are four packets in each PU, two for each cycle, which are routed for $N - 1$ steps in one direction along one cycle each. In this way we obtain an entirely time-independent algorithm. If we are willing to make a small concession in this respect, in applying the following *switching-strategy*, it is sufficient to have two packets per PU. Let l be the number of OC-PUs between A and B, then during the first l steps, A transfers to C the packets that are running forward, and during the last $m/2 - l$ steps those that run backward. B operates in the opposite dircetion. In our case the number l equals $2 \cdot n_1 - 1$ for all PUs in Column $n_1 - 2$. Thus, each PU can still compute its routing decisions in constant time.

Theorem 3 *If every PU of an $n_1 \times n_2$ torus, with n_1 odd and n_2 even, holds 2 packets, then gossiping can be performed in $n_1 \cdot n_2/2$ steps.*

3.3 Two-Dimensional Tori, n_1 and n_2 Odd

If both n_1 and n_2 are odd, then we need another modification similar to that in Section 3.2. It applies to Row $n_2 - 2$ and Row $n_2 - 1$. The only difference is that we now allow OOC-PUs to be located at distance two from their supporting OC-PUs. This implies that OOC-PUs will sometimes forward a packet that they received in their capacity as an OOC-PU to an adjacent OOC-PU, the latter being unfortunate enough to have only one adjacent OC-PU. Special attention has to be given to PU $(n_1 - 2, n_2 - 2)$ which does not belong to any cycle and simply transmits every packet it receives in the opposite direction, so $L \leftrightarrow R$ and $T \leftrightarrow B$. After the preceding discussion the modified routing scheme as described in Figure 4 is self-explanatory.

Except for row $n_2 - 2$, the rules are the same as before. In row $n_2 - 2$ we perform

$$\begin{aligned} i \neq n_1 - 1 \text{ and } j = n_2 - 2 \quad &: \quad t \leftrightarrow b; l \leftrightarrow r. \\ i = n_1 - 1 \text{ and } j = n_2 - 2 \quad &: \quad t \leftrightarrow r; b \leftrightarrow l. \end{aligned}$$

Theorem 4 *If every PU of an $n_1 \times n_2$ torus, with n_1 odd and n_2 odd, holds 2 packets, then gossiping can be performed in $\lceil n_1 \cdot n_2/2 \rceil$ steps.*

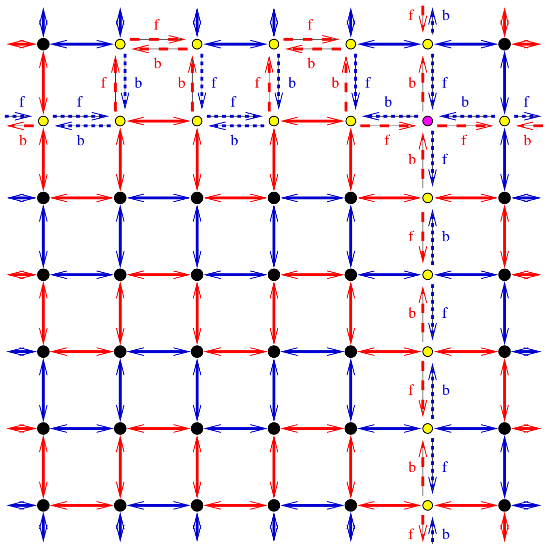

Figure 4: Routing scheme for two partial Hamiltonian cycles on a 7×7 torus.

3.4 Higher Dimensional Tori

For d-dimensional tori, one should first construct d edge-disjoint Hamiltonian cycles. Such constructions are described in [5, 2, 1]. Then, if every PU holds d packets of size s, each packet is forwarded for $P/2$ steps in both directions along one of the Hamiltonian paths. In this way, after $P/2$ steps, each PU has received all packets.

The routing is trivial, but the precomputation may be quite involved. The number of routing steps performed is also rather high due to the fact that each PU holds d packets. In the following section we give a generic construction that works for arbitrary dimensions, and which gives a routing that can be performed in $P/(2 \cdot d) + o(P)$ steps.

4 One-Packet Algorithms

In this section we give another practical alternative to the approach in [14]: we present algorithms which require only one packet per PU and are optimal to within $o(P)$ steps. The idea is simple: we construct d edge-disjoint cycles of length $P/d + o(P)$. Each of the cycles must have this special property: if a PU does not lie on it, this PU must be adjacent to two PUs that do lie on the cycle. Each of these two PUs will transmit the packets from one direction of the cycle to the out-of-cycle-PU.

4.1 Two-Dimensional Tori

The construction of two partial Hamiltonian cycles with the desired properties is easy for two-dimensional tori. At the same time, this clearly illustrates our intentions. There are two axes: the x_1-axis, running horizontally and the x_2-axis running vertically. PU $(0, 0)$ is assumed to lie in the upper-left corner. n_1 and n_2 denote the size of the torus in the x_1 and x_2 direction, respectively. We assume that n_1 is even and that $n_2 \geq 2$.

The construction is somewhat similar to the approach presented in Section 3.2. Here we have one zigzag in the highest rows of the torus. This zigzag makes positive moves along axes x_1, x_2 and x_1, respectively. Below this $n_2 - 1$ moves along the x_2-axis follow, the final move bringing us over the wrap-around connection to the next zigzag. In this way we obtain two edge-disjoint cycles of total length $n_1 \cdot n_2/2 + n_1$. The constructed cycles are illustrated in Figure 5. Binding the out-of-cycle-PUs is done exactly as in the algorithm in Section 3.2; the case of odd n_1 can be treated by inserting one special column.

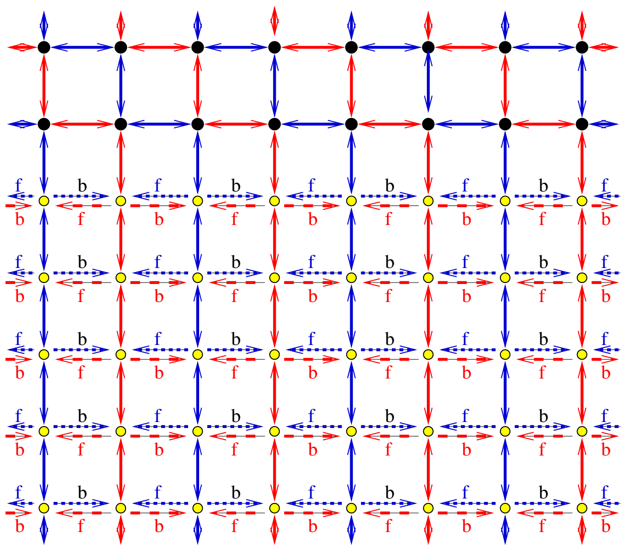

Figure 5: Two edge-disjoint cycles, each of which comes within distance at most one from all PUs.

Theorem 5 *If every PU of an $n_1 \times n_2$ torus holds 1 packet, then gossiping can be performed in $n_1 \cdot n_2/4 + n_1/2 + 1$ steps.*

Proof: Each of the cycles consists of $n_1 \cdot n_2/2 + n_1$ PUs, so using both directions in parallel one needs $n_1 \cdot n_2/4 + n_1/2 + 1$ steps to spread all the packets within the cycle. For every OOC-PU, the two supplying OC-PUs are separated by $n_2 + 1$ other PUs on their cycle, so extra steps required for multiple receives in the OOC-PUs can be easily eliminated using the switching strategy presented in Section 3.2. Omitting the simple forward/backward switching at the cost of $n_2 + 1$ extra steps leads to a fully time-independent $n_1 \cdot n_2/4 + \mathcal{O}(n_1 + n_2)$ step algorithm. □

4.2 Three-Dimensional Tori

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

For two-dimensional tori, there are two cycles, each a concatenation of $n_1/2$ *laps*. Each lap consists of a zigzag followed by a long move along the x_2-axis. The zigzags are needed to bring us two positions further, connecting the laps of a cycle. For three-dimensional tori, there are three cycles. These are identical, except that they start in different positions: Cycle j, $0 \leq j \leq 2$, starts in PU $(j, 0, 0)$. Here, the zigzags bring us three positions further. The first type of zigzag consists of positive moves along the following sequence of axes: $(1, 3, 1, 3, 1)$. A zigzag is followed by $n_3 - 2$ moves along the x_3-axis, the last move traversing a wrap-around connection. In this way we can fill up a plane, but in order to get to the next plane, there must be a second type of zigzag consisting of moves along the following sequence of axes: $(1, 3, 1, 3, 2)$. Thus, a complete cycle is the concatenation of $n_1/3 \cdot n_2$ laps. After each $n_1/3 - 1$ laps using a zigzag of the first type, one lap with a zigzag of the second type follows. The cycles are illustrated in Figure 6 and Figure 7.

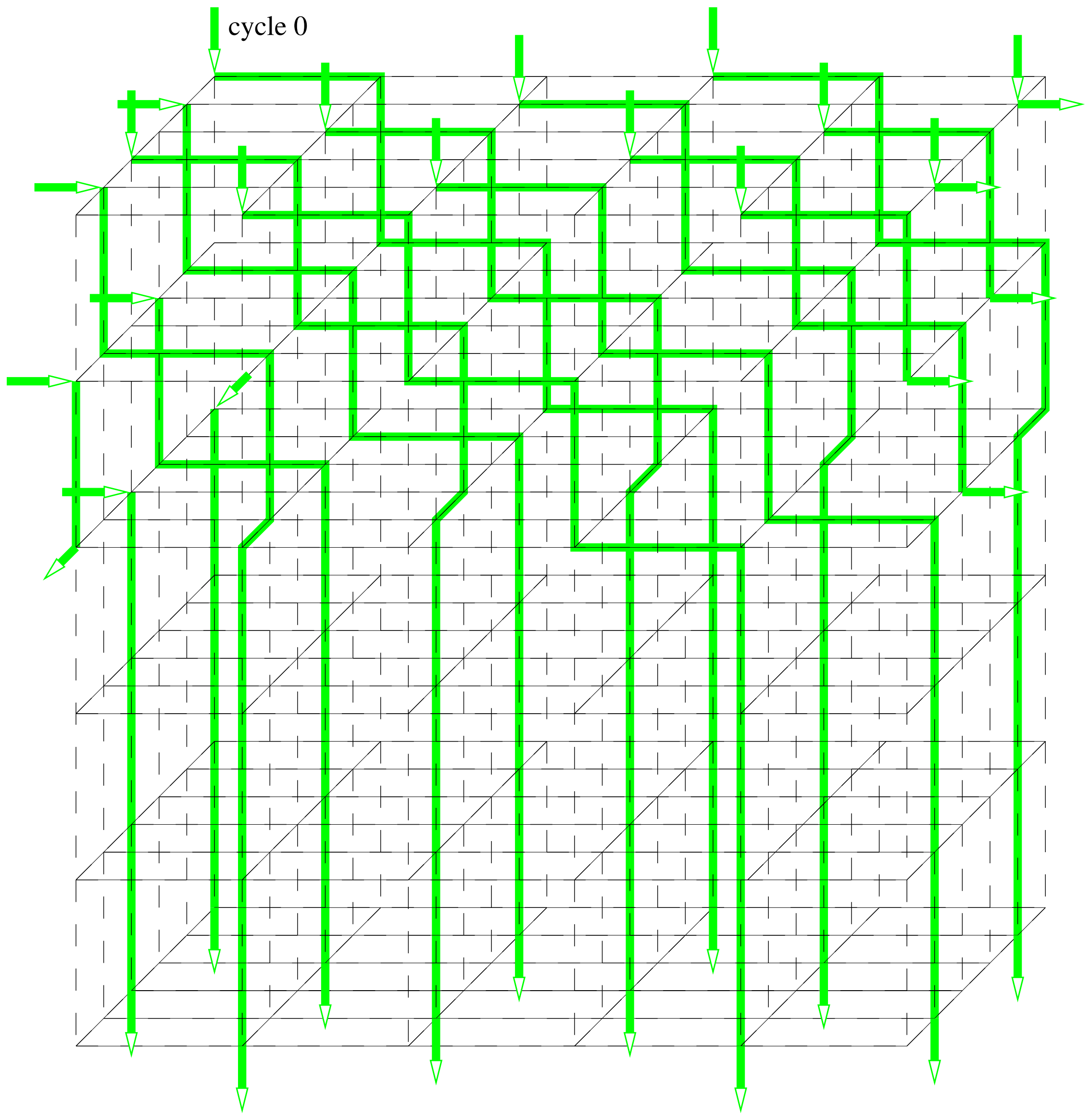

Figure 6: One edge-disjoint cycle used in the one-packet algorithm on a three-dimensional torus.

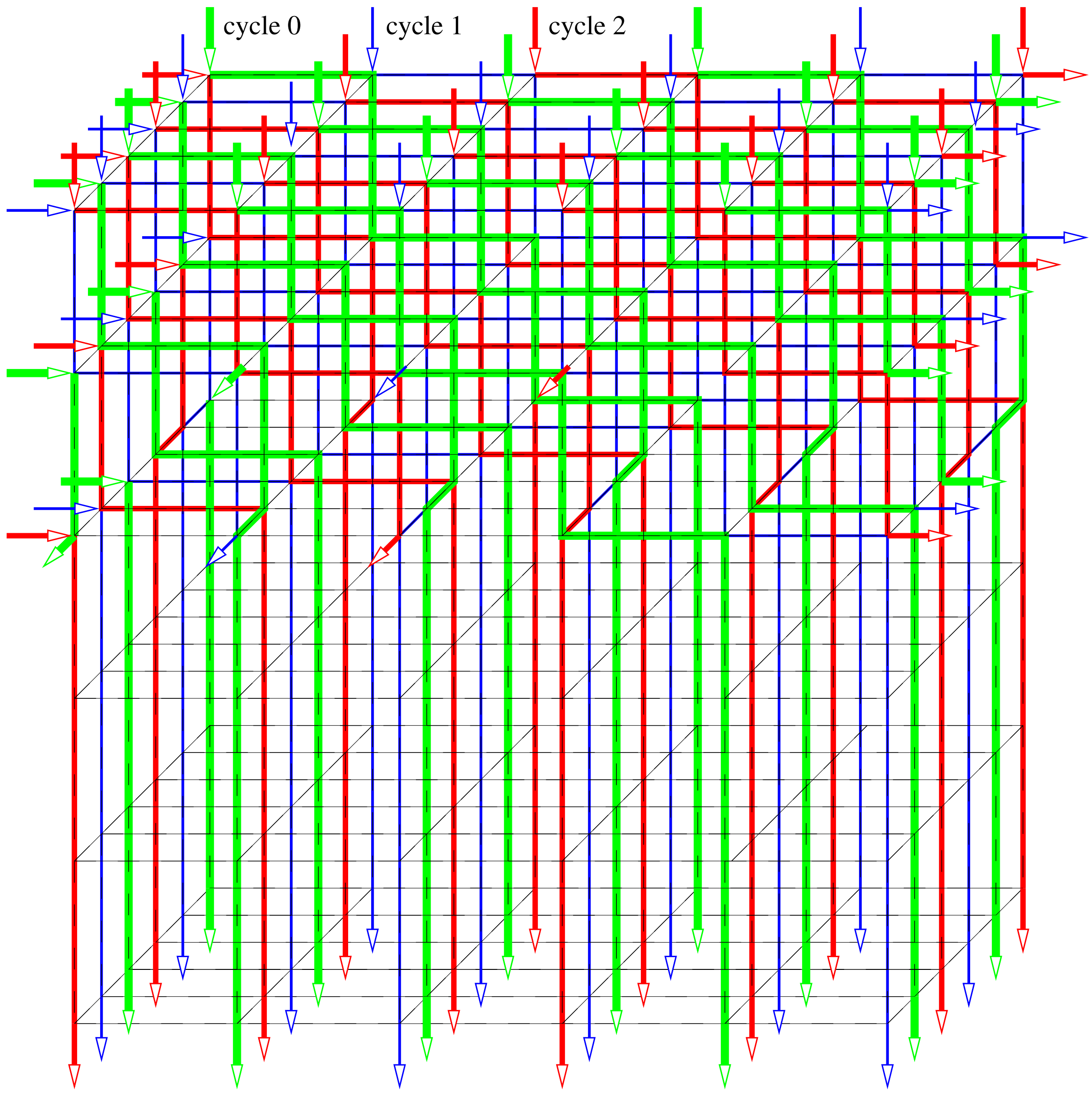

Figure 7: Three edge-disjoint cycles used in the one-packet algorithm on a three-dimensional torus.

That the constructed cycles have all the desired properties is proven generally in the next section, but in the case of low-dimensional tori, it can also be tested more explicitly with the help of the following reduction.

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

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

For any given value of $x_3 > 3$, the structure of the rods is the same. Thus,

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

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

$$\begin{array}{ccc} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{array}$$

In this way, PU (x_1, x_2, x_3), $x_3 > 3$, on Cycle j can be supplied with packets which do not lie on its own cycle: it receives packets running forward on Cycle $(j+1) \bmod 3$ from the PU $((x_1-1) \bmod n_1, x_2, x_3)$ and packets running backward from PU $((x_1, (x_2-1) \bmod n_2, x_3)$. Similarly, it is supplied with packets running forward on Cycle $(j-1) \bmod 3$ from PU $((x_1+1) \bmod n_1, x_2, x_3)$ and with backward-running packets from PU $((x_1, (x_2 + 1) \bmod n_2, x_3)$. Each pair of supporting on-cycle-PUs is separated by $(n_1/3 + 1) \cdot n_3 - 1$ other PUs.

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

Theorem 6 *If every PU of an $n_1 \times n_2 \times n_3$ torus, with n_1 a multiple of three, n_2 a multiple of n_1 and $n_3 \geq 3$, holds 1 packet, then gossiping can be performed in $n_1 \cdot n_2 \cdot n_3/6 + n_1 \cdot n_2/2 + 1$ steps.*

Proof: Each lap has length n_3+3. There are $n_1/3 \cdot n_2$ laps, so the cycles have length $n_1 \cdot n_2 \cdot n_3/3 + n_1 \cdot n_2$. Packets are routed in both directions along them, so after $n_1 \cdot n_2 \cdot n_3/6 + n_1 \cdot n_2/2$ steps, a PU has received all the packets that have started in a PU that lies on the cycle(s) in which the first-mentioned PU lies. The remaining packets running on the other cycles are received from the PU's neighbors one step after they received those packets. Multiple receives in the OOC-PUs can again be eliminated by using the switching-strategy from Section 3.2. Giving up the switching, in order to obtain a time-independent algorithm, requires $\mathcal{O}(n_1 \cdot n_3)$ extra steps. □

4.3 Higher Dimensional Tori

The idea from the previous section can be generalized with no problem for d-dimensional $n_1 \times n_2 \times \cdots \times n_d$ tori. Now we construct d edge-disjoint cycles, each of them covering $P/d + o(P)$ PUs.

4.3.1 Construction

The cycles are numbered 0 through $d-1$. Cycle j starts in PU $(j, 0, \ldots, 0)$. As before, a cycle is composed of laps, starting with a zigzag and ending with moves along the x_d-axis. There are $d-1$ types of zigzags, which are used in increasingly exceptional cases. They can be concisely represented by specifying the sequence of (positive) axes along which a move is made. We give all types of zigzags for $2 \leq d \leq 5$:

$$\begin{aligned}
\mathit{zigzag}(2,1) &= (1,2,1). \\[4pt]
\mathit{zigzag}(3,1) &= (1,3,1,3,1), \\
\mathit{zigzag}(3,2) &= (1,3,1,3,2). \\[4pt]
\mathit{zigzag}(4,1) &= (1,4,1,4,1,4,1), \\
\mathit{zigzag}(4,2) &= (1,4,1,4,2,4,1), \\
\mathit{zigzag}(4,3) &= (1,4,1,4,2,4,3). \\[4pt]
\mathit{zigzag}(5,1) &= (1,5,1,5,1,5,1,5,1), \\
\mathit{zigzag}(5,2) &= (1,5,1,5,2,5,1,5,1), \\
\mathit{zigzag}(5,3) &= (1,5,1,5,2,5,3,5,1), \\
\mathit{zigzag}(5,4) &= (1,5,1,5,2,5,3,5,4).
\end{aligned}$$

Generally, $\mathit{zigzag}(d,j)$ indicates the j-th zigzag pattern for d-dimensional tori. It makes $d-j+1$ moves along the x_1-axis, one move along the x_2 up to the x_j-axis and $d-1$ moves along the x_d-axis.

During lap i, $1 \leq i \leq n_1/d \cdot n_2 \cdot \cdots \cdot n_{d-1}$, $\mathit{zigzag}(d,j)$ is applied, where j is the largest index for which the following condition is true:

$$i \bmod (n_1/d \cdot n_2 \cdot \cdots \cdot n_{j-1}) = 0.$$

That is, after precisely this many laps, the cycle has filled up a subspace spanned by $x_1, \ldots, x_{j-1}$ and x_d and it should make a move along the x_j-axis to get to the next subspace.

For the numbers $z_{d,j}$ of $\mathit{zigzag}(d,j)$ patterns that are traversed by a cycle, we have

$$z_{d,1} = (n_1/d - 1) \cdot \prod_{l=2}^{d-1} n_l, \qquad z_{d,j} = (n_j - 1) \cdot \prod_{l=j+1}^{d-1} n_l, \quad \text{for } 1 < j < d.$$

Thus, for the total moves $m_{d,j}$ along the x_j-axis, we have

$$m_{d,1} = \prod_{l=1}^{d-1} n_l - \sum_{j=2}^{d-1} \prod_{l=j}^{d-1} n_l, \qquad m_{d,j} = \prod_{l=j}^{d-1} n_l, \quad \text{for } 1 < j < d.$$

In order to guarantee that a cycle returns to its starting point after traversing $n_1/d \cdot n_2 \cdot \cdots \cdot n_{d-1}$ laps, it must be the case that each $m_{d,j}$ is a multiple of n_j. For $1 < j < d$, this condition is void, because the expression for $m_{d,j}$ contains n_j as a factor. On the other hand, for $j = 2$, we find as one of the most important conditions on the n_l, that

$$(\sum_{j=2}^{d-1} \prod_{l=j}^{d-1} n_j) \bmod n_1 = 0. \tag{1}$$

The other conditions are that

$$\begin{aligned}
n_1 \bmod d &= 0 \qquad (2) \\
n_d &\geq d. \qquad (3)
\end{aligned}$$

(2) is required to guarantee that the x_1-x_d planes are nicely filled-up by n_1/d laps of all cycles. Maybe it is not essential, but it makes the construction much more regular. (3) must be statisfied because the zigzags make $d-1$ moves along the x_d-axis. Surprisingly, (1) together with (2) does not imply that all the n_l, $1 \leq l < d$, must be multiples of d. For example, for $d = 4$, the conditions are satisfied for $n_1 = 4$, $n_2 = 5$, $n_3 = 2$, because $n_2 \cdot n_3 + n_3 = 12$, a multiple of n_1. This case is illustrated in Figure 8.

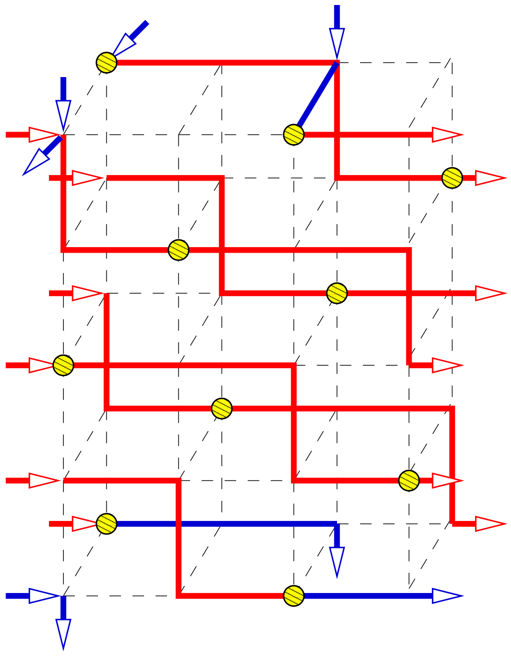

Figure 8: Cycle 0 in a four-dimensional $4 \times 5 \times 2 \times n_4$ torus. The torus is projected along the x_4-axis, so the moves along this axis are not shown. PU $(0, 0, 0, 0)$ lies in the upper-back-left corner. The x_1-, x_2- and x_3-axis run left-, down- and frontward, respectively. The dots give the beginpoints of the zigzags. Because $n_1 = d = 4$, *zigzag*$(4, 1)$ does not occur. Moves with *zigzag*$(4, 2)$ are drawn in red, those with *zigzag*$(4, 3)$ in blue.

4.3.2 Analysis for a Simplified Case

For further analysis we first consider a simplified construction with only zigzags of type *zigzag*$(d, 1)$ at the beginning of each lap. The cycles are now decomposed in many non-connected subcycles. If for PU $(i_1, \ldots, i_{d-1}, 0)$, the indices satisfy

$$(i_1 + \sum_{l=2}^{d-1} (l-1) \cdot i_l) \bmod d = j, \tag{4}$$

then this PU belongs to a subcycle of Cycle j. First we test that all desired properties hold for this simplified construction. Then, in Section 4.3.3, we will add the higher zigzags in order to connect more and more subcycles, testing that these properties are preserved.

In any two-dimensional x_1-x_d-plane, the situation is as depicted in Figure 9. So, as there are no other connections, the whole scheme is clearly edge-disjoint. Each PU is traversed by one cycle, except for the PUs in the highest d rows, which are each traversed by two cycles. By (4), if a PU P at position $(i_1, \ldots, i_l, \ldots, i_{d-1}, i_d)$ lies on Cycle j, then a PU in position $(i_1 + x, i_2, \ldots, i_{d-1}, i_d)$ lies on Cycle $(j + x) \bmod d$ and a PU in some position $(i_1, i_2, \ldots, i_l + x, \ldots, i_{d-1}, i_d)$, $l \geq 2$, lies on Cycle $(j + x \cdot (l - 1)) \bmod d$. In particular, considering all neighbors of P, that is $x = 1$ or $x = -1$, this implies that P

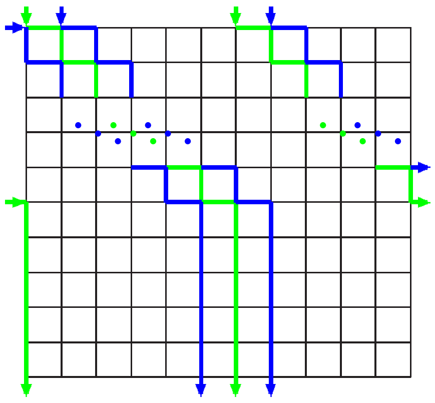

Figure 9: The situation in an x_1-x_d-plane.

is adjacent to two nodes of all cycles different from Cycle j. In the highest rows, the connections along the x_1-axis are already used, but this is not serious, a PU in these rows receives packets directly from two cycles, and the total situation is so that it can again receive all packets without delay. The whole schedule is illustrated in more detail in Figure 10. For higher dimensions this schedule is generalized in a straightforward way. A PU must determine whether its last coordinate i_d satisfies $i_d < d$ or $i_d \geq d$. Then it can determine the appropriate actions for each of the following four types of connections:

1. The connection along the negative x_1-axis: do nothing, send the backward moving packets, respectively.

2. The connections along a negative x_j-axis, $1 < j < d$: send the forward moving packets along the x_d-axis, send the forward moving packets, respectively.

3. The connection along the positive x_1-axis: do nothing, send the forward moving packets, respectively.

4. The connections along a positive x_j-axis, $1 < j < d$: send the backward moving packets along the x_1-axis, send the backward moving packets, respectively.

4.3.3 Reinserting the Higher Zigzags

In the following, we reintroduce the *zigzag*(d, j) for j running from 2 to $d - 1$, testing that the edge-disjointness is preserved and that the structure of the neighborhood is still so that each PU can receive packets running in both directions along all cycles.

First we insert one *zigzag*$(d, 2)$ pattern after each $n_1/d - 1$ laps starting with *zigzag*$(d, 1)$. In this way fewer connections along the x_1-axis are used, so the x_1-axis cannot cause problems. Between any pair of consecutive x_1-x_d planes, only one connection along the x_2-axis is used. That is, for any given set of indices $(i_2, i_3, \ldots, i_{d-1})$, one connection between some PU $(i_1, i_2, i_3, \ldots, i_{d-1}, i_d)$ and PU $(i_1, (i_2 + 1) \bmod d, i_3, \ldots, i_{d-1}, i_d)$ is used. So, these are disjoint as well. We must still check that the *zigzag*$(d, 2)$ indeed connect subcycles of the same cycle. But this is precisely what is guaranteed by (4) and the moves of *zigzag*$(d, 2)$: Comparing the positions of the cycles in an x_1-x_d plane with those in the subsequent plane, we see that they are all shifted one position along the negative x_1 axis. This is consistent with the fact that in *zigzag*$(d, 2)$ precisely one move along the positive x_1-axis is replaced by a move along the positive x_2-axis. That there are no overlaps along the x_d-axis, follows because the *zigzag*$(d, 2)$ are inserted so that they just connect each other, continuing the cycle exactly there where it was disrupted. This replacement of *zigzag*$(d, 1)$ patterns by *zigzag*$(d, 2)$ is illustrated in Figure 11.

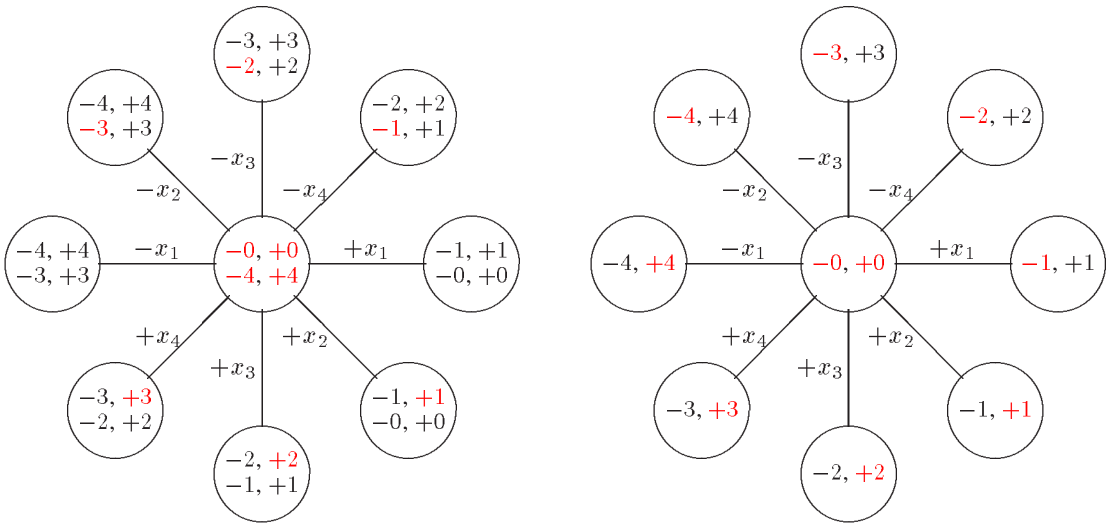

Figure 10: The structure of the neighborhoods of a PU on Cycle 0 in a five-dimensional torus. **Left:** The situation of a PU with $i_d < d$. **Right:** The situations for a PU with $i_d \geq d$. The circles indicate PUs, the lines connections. The numbers next to the axes give the connections (from the perspective of the PU in the middle). Here $+x_l$ and $-x_l$ indicate a positive and a negative move along the x_l-axis, respectively. The numbers in the circles give the packets that pass through these PUs. $+j$ and $-j$, indicate forward and backward moving packets on Cycle j, respectively. This pattern is derived from the construction in (4). *zigzag*$(d, 1)$ is so that a PU also holds in addition to the packets from some Cycle j, the packets from Cycle $(j - 1) \bmod d$. This zigzag uses the connections along the x_1-axis. If a number in the circles is printed in red, this indicates that the corresponding packets are transferred from this PU towards the PU in the middle.

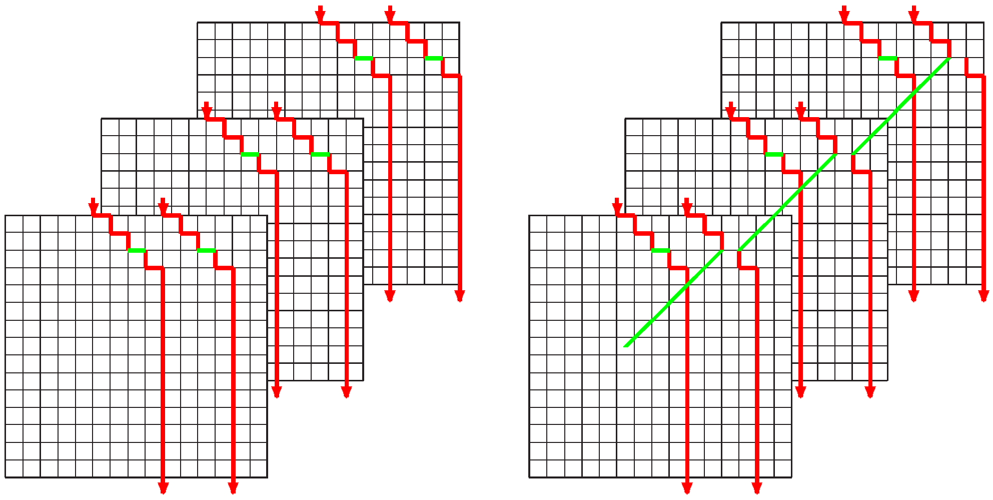

Figure 11: **Left:** Sections of subcycles of the same cycle in three consecutive x_1-x_d-planes. *zigzag*$(d, 1)$ is used at the beginning of each lap. **Right:** The rightmost *zigzag*$(d, 1)$ patterns have been replaced by *zigzag*$(d, 2)$ patterns, connecting these subcycles.

The replacement of some of the *zigzag*(d, j), $2 \leq j \leq d-2$ patterns by *zigzag*$(d, j+1)$ patterns works in the same way: one move along the x_1-axis is replaced by a move along the $(j+1)$-axis. Because of the off-sets specified in (4), these new moves fit in another subcycle of the same cycle just after a disruption caused by the removal of the move along the x_1-axis. This guarantees edge-disjointness and also ensures that all subcycles of a cycle are finally connected together.

4.3.4 Structure of the Neighborhood

It remains to show that the structure of the neighborhood still allows each PU to receive packets running in both directions along all cycles. For the PUs with $i_d \geq d$, nothing has changed by replacing some zigzags. Now consider a PU P with $i_d < d$. For $i_d = 0, 1$, the situation is unchanged. Generally, P is traversed by two zigzags, but they cannot use arbitrary connections. For $i_d = j$, we can find only four possible situations in P:

$$\begin{array}{llcl}
\textbf{Situation 1:} & (-x_d, +x_1) & + & (-x_1, +x_d), \\
\textbf{Situation 2:} & (-x_d, +x_j) & + & (-x_1, +x_d), \\
\textbf{Situation 3:} & (-x_d, +x_1) & + & (-x_j, +x_d), \\
\textbf{Situation 4:} & (-x_d, +x_j) & + & (-x_j, +x_d).
\end{array}$$

Here $(-x_d, +x_l) + (-x_{l'}, +x_d)$ denotes that one of the zigzags through P enters over the negative x_d-axis and leaves over the positive x_l-axis, and the other over the negative $x_{l'}$-axis and the positive x_d-axis. This follows from the definitions of the zigzags. These situations are illustrated in Figure 12.

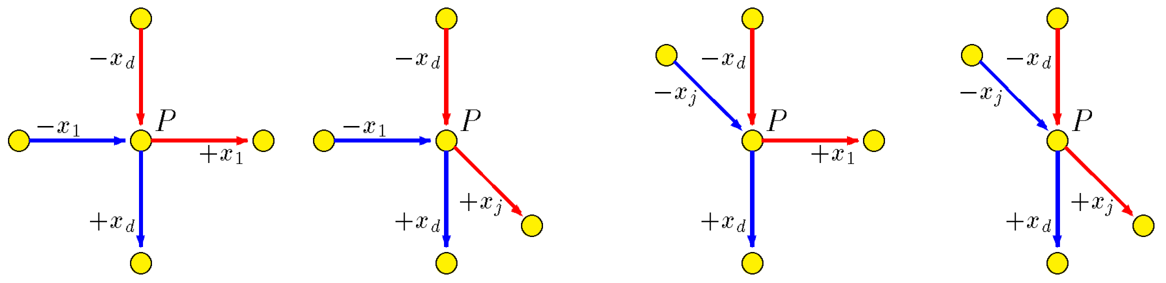

Figure 12: From left to right, Situation 1 to Situation 4 for a PU P with $i_d = j$, $2 \leq j < d$.

We use the zigzag that enters over $-x_d$ to define the Type of P: if this zigzag is a part of Cycle i, then we say that P is of Type i. By $-c_l$ and $+c_l$, $0 \leq l < d$, we denote the set of packets moving backward and forward on Cycle l, respectively. In the sequel we will explain how the following properties can be established for a PU of Type i:

1. P receives all packets from $-c_i$ and $+c_i$ without delay.
2. P receives all packets on $-c_{(i-1) \bmod d}$ and all packets on $+c_l$, $0 \leq l < d$, with delay at most 1.
3. P receives all packets on $-c_l$, $0 \leq l < d$, $l \neq (i-1) \bmod d$, with delay at most 2.

We will apply an induction-like proof. Point 1 immediately holds for all PUs, because the packets on Cycle i run through P in all situations. We may utilize this while establishing Point 2, and both Point 1 and 2 may be utilized while establishing Point 3.

Above we have seen that the three points hold when there are only *zigzag*$(d, 1)$ patterns. In the more general setting, the properties can be established easily for a PU in Situation 1. Its neighbors send P the same packets as before. In this way, P receives the packets from $\pm c_i$ and $\pm c_{(i-1) \bmod d}$ without delay; the packets on $+c_l$, $0 \leq l < d$, $l \neq i, (i-1) \bmod d$, have been received by the neighbors of P without delay (according to Point 1), and reach P with delay at most 1; similarly, the packets on $-c_l$, $0 \leq l < d$

in sit.	packets from cycle	are received via connection	with delay
1	$\pm c_0,\ \pm c_{d-1}$	on cycle	0
	$+c_l,\ 1 \le l \le d-2$	$+x_{l+1}$	1
	$-c_l,\ 1 \le l \le d-2$	$+x_{d-l}$	2
2	$\pm c_0,\ \pm c_{d-1}$	on cycle	0
	$+c_l,\ 1 \le l \le d-2,\ l \ne j-1$	$+x_{l+1}$	1
	$+c_{j-1}$	$-x_{j+1}$	1
	$-c_1$	$+x_1$	1
	$-c_l,\ 2 \le l \le j-2$	$-x_{d-l+1}$	1
	$-c_l,\ j-1 \le l \le d-2$	$-x_{d-l}$	2
3	$\pm c_0,\ \pm c_{d-j+1}$	on cycle	0
	$+c_l,\ 1 \le l \le d-2,\ l \ne d-j+1$	$+x_{l+1}$	1
	$+c_{d-1}$	$-x_1$	1
	$-c_l,\ 1 \le l \le d-j-1$	$-x_{l+1}$	2
	$+c_{d-j}$	$+x_{d-j+2}$	2
	$-c_l,\ d-j+2 \le l \le d-1$	$-x_{d-l+1}$	1
4	$\pm c_0,\ \pm c_{d-j+1}$	on cycle	0
	$+c_l,\ 1 \le l \le d-2,\ l \ne d-j,\ d-j+1$	$+x_{l+1}$	1
	$+c_{d-j}$	$-x_{j+1}$	1
	$+c_{d-1}$	$-x_1$	1
	$-c_l,\ 1 \le l \le d-1,\ l \ne d-j,\ d-j+1$	$-x_{d-l+1}$	1
	$-c_{d-j}$	$-x_{d-j+2}$	2

Table 1: Packet supply for a Type 0 PU in Situations 1 to 4.

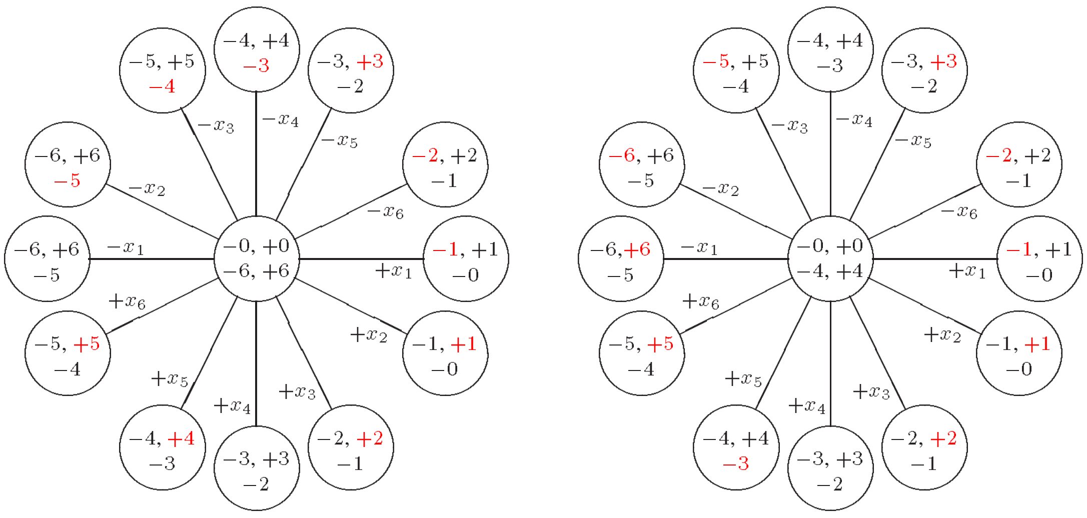

Figure 13: Situations 2 (**left**) and 4 (**right**) for a PU of Type 0 in a seven dimensional torus for the case $j = 4$. The circles represent the PUs, the lines the specified connections. In the central circle, we have indicated the streams that this PU receives directly through the cycles on which it lies. In the other circles, the numbers in the higher row indicate the streams of packets that reach these PUs without delay. The numbers in the lower row indicate the streams that reach them with delay at most one. In red, the streams that are forwarded to the central PU are indicated.

$l \neq i, (i-1) \bmod d$, have reached the neighbors with delay at most 1 (according to Point 2), and reach P with delay at most 2.

Proceeding with our induction-like proof, we will establish Point 2 and Point 3 for the other three situations. Point 2 is a very strong guideline: for a PU of Type i, the packets of $-c_{(i-1) \bmod d}$ and of $+c_l$, $0 \leq l < d$, $l \neq i$, must be drawn from the packets of its neighbors that were received without delay according to Point 1. Hereafter, for establishing Point 3 there is almost no freedom left. Table 1 gives a complete specification that meets our requirements for a Type 0 PU under all four situations. Adaptation to Type i PUs, $1 \leq i \leq d-1$, can easily be done by adding i and calculating modulo d.

4.3.5 Running Time

Having checked the correctness of the algorithm, we now turn to its performance. Each lap has length $n_d + d$. There are $n_1/d \cdot \prod_{l=2}^{d-1} n_l$ laps, so the cycles have length $(1 + d/n_d) \cdot P/d$. Packets are routed in both directions along them, so after $(1 + d/n_d) \cdot P/(2 \cdot d)$ steps, a PU has received all the packets that have started in a PU that lies on the cycle(s) on which the first-mentioned PU lies. The packets running on the other cycles are received from the PU's neighbors one step after they received those packets, the maximal delay is two. Multiple receives in the out-of-cycle-PUs can again be eliminated by the switching-strategy of Section 3.2: the supporting OC-PUs are separated by at most three hyperplanes. Giving up the switching in order to obtain a time-independent algorithm, requires at most $(n_d + d) \cdot n_1/d \cdot n_2 \cdot \cdots \cdot n_{d-2} \cdot 3$ extra steps. This would introduce an extra factor $1 + 3 \cdot d/n_{d-1}$ in the time consumption. To resume, we obtain the main result of this paper:

Theorem 7 *If every PU of a d-dimensional $n_1 \times \cdots \times n_d$ torus with P PUs holds 1 packet, then gossiping can be performed in $(1 + d/n_d) \cdot P/(2 \cdot d) + 2$ steps. A fully time-independent algorithm runs in $(1 + d/n_d) \cdot (1 + 3 \cdot d/n_{d-1}) \cdot P/(2 \cdot d)$ steps. Both results require that $(\sum_{j=2}^{d-1} \prod_{l=j}^{d-1} n_j) \bmod n_1 = 0$, $n_1 \bmod d = 0$ and $n_d \geq d$.*

In order to minimize the time consumption, depending on the applied variant, one should try to index the axes so that either $n_d \geq n_l$, for all $1 \leq l \leq d-1$, or that $n_{d-1} \geq n_d \geq n_l$, for all $1 \leq l \leq d-2$. Of course, this should only be done within the limits imposed by conditions (1), (2) and (3).

5 Comparison of Performances

We assume that every PU initially holds s bytes of data, and that sending a packet of size s' between two adjacent PUs takes

$$T_{\text{transfer}}(s') = s'/s + r \tag{5}$$

time units. Here $r = \textit{start-up-time}/(s \cdot \textit{time-to-transfer-byte})$ is the normalized start-up time. Depending on the parallel computer under consideration and the value of s, r can be large (up to 10,000) or small (less than 1). This cost model gives an accurate description of the transfer time on real systems.

On a d-dimensional $n \times \cdots \times n$ torus, the algorithms from Section 3 require $n^d/2$ steps, in each of which a PU sends packets of size $s' = s/d$ to all its neighbors. Thus, for the number of time units T_{opt} for these algorithms, we find

$$T_{\text{opt}}(n, d) = n^d/2 \cdot (1/d + r).$$

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

$$T_{\text{one}}(n, d) \simeq n^{d-1}/2 \cdot (n/d + 1) \cdot (1 + r).$$

The corresponding number of time steps T_{stv} for the algorithm from [14] is given by

$$T_{\text{stv}}(n, d) \simeq n^d/(2 \cdot d) \cdot (1 + r).$$

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

Table 2: Comparison between the number of time steps taken by the algorithms from [14] (top), from Section 3 (middle) and from Section 4 (bottom). In each cell we give the result for $d = 2$ (left) and $d = 3$ (right).

Numerical results for some characteristic values of n and r are given in Table 2. For $r > 1$, none of these approaches makes sense, because then one should better apply a strategy that requires substantially fewer start-ups (see [7]). We see that mostly the one-packet algorithm from Section 4 is only a small percent slower than the algorithm from [14], but it is much simpler and far easier to generalize for higher dimensions. Therefore, we think that in most cases it may constitute a practical alternative. The algorithm from Section 3 may be attractive for $d = 2$ and small r: for $d = 2$ it is very simple, while achieving almost optimal performance.

6 Conclusion

We have completed the analysis of the gossiping problem on full-port store-and-forward tori. In [14] only one interesting aspect of this problem was considered. We have shown that an almost equally good performance can be achieved by simpler time-independent algorithms, and have given explicit schemes for higher-dimensional tori as well.

References

[1] Alspach, B., J-C. Bermond, D. Sotteau, ‘Decomposition into Cycles I: Hamilton Decompositions,’ *Proc. Workshop Cycles and Rays*, Montreal, 1990.

[2] Aubert, J., B. Schneider, ‘Decomposition de la Somme Cartesienne d’un Cycle et de l’Union de Deux Cycles Hamiltoniens en Cycles Hamiltonien,’ *Discrete Mathematics*, 38, pp. 7–16, 1982.

[3] 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.

[4] Delmas, O., S. Perennes, ‘Circuit-Switched Gossiping in 3-Dimensional Torus Networks,’ *Proc. 2nd International Euro-Par Conference*, LNCS 1123, pp. 370–373, Springer-Verlag, 1996.

[5] Foregger, M.F., ‘Hamiltonian Decomposition of Products of Cycles,’ *Discrete Mathematics*, 24, pp. 251–260, 1978.

[6] Fraigniaud, P., J.G. Peters, ‘Structured Communication in Torus Networks,’ *Proc. 28th Hawai Conference on System Science*, pp. 584–593, 1995.

[7] Juurlink, B., P.S. Rao, J.F. Sibeyn, ‘Worm-Hole Gossiping on Meshes and Tori,’ *Techn. Rep. MPI-I-96-1018*, Max-Planck Institut für Informatik, Saarbrücken, Germany, 1996. Appearing soon in *IEEE Transactions on Parallel and Distributed Systems*.

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

[9] Peters, J.G., M. Syska, ‘Circuit-Switched Broadcasting in Torus Networks,’ *IEEE Transactions on Parallel and Distributed Systems*, 7, pp. 246–255, 1996.

[10] Plateau, B., D. Trystam, ‘Optimal Total Exchange for a 3-D Torus of Processors,’ *Information Processing Letters*, 42, pp. 95–102, 1992.

[11] Rao, P.S., G. Mouney, ‘Data Communications in Parallel Block Predictor-Corrector Methods for solving ODEs,’ *Techn. Rep. No. 95399*, LAAS-CNRS, France, 1995.

[12] Reif, J., L.G. Valiant, ‘A logarithmic time sort for linear size networks,’ *Journal of the ACM*, 34(1), pp. 68–76, 1987.

[13] Sibeyn, J.F., ‘Routing with Finite Speeds of Memory and Network,’ *Proc. 22nd Symposium on the Mathematical Foundations of Computer Science*, LNCS 1295, pp. 488–497, Springer-Verlag, 1997.

[14] Šoch, M., P. Tvrdík, ‘Optimal Gossip in Store-and-Forward Noncombining 2-D Tori,’ *Proc. 3rd International Euro-Par Conference*, LNCS 1300, pp. 234–241, Springer-Verlag, 1997.