Revision of May 1992 *LIDS-P-2087*

Partial Multinode Broadcast and Partial Exchange Algorithms for d-dimensional Meshes[1]

by

Emmanouel A. Varvarigos and Dimitri P. Bertsekas[2]

Abstract

In this paper we consider the partial multinode broadcast and the partial exchange communication tasks in *d*-dimensional meshes. The partial multinode broadcast in an N-processor network is the task in which each of $M \leq N$ arbitrary nodes broadcast a packet to all the remaining $N-1$ nodes. Correspondingly, in the partial exchange there are $M \leq N$ nodes that wish to send a separate, personalized packet to each of the other nodes. We propose algorithms for the *d*-dimensional mesh network that execute the partial multinode broadcast and the partial exchange communication tasks in near-optimal time. No assumption concerning the location of the M source nodes is made. The communication algorithms proposed are "on line" and distributed. We further look at a dynamic version of the problem, where broadcast requests are generated at random times. In particular, we assume that the broadcast requests are generated at each node of the mesh according to a Poisson distribution with rate λ. Based on the partial multinode broadcast algorithm, we propose a dynamic decentralized scheme to execute the broadcasts in this dynamic environment. We find an upper bound on the average delay required to serve each broadcast. We prove that the algorithm is stable for network utilization ρ close to 1, and the average delay is of the order of the diameter for any load in the stability region.

[1] Research supported by NSF under Grant NSF-DDM-8903385 and by the ARO under Grant DAAL03-86-K-0171.

[2] Laboratory for Information and Decision Systems, M.I.T, Cambridge, Mass. 02139.

1. INTRODUCTION

The processors of a multiprocessor system, when doing computations, often have to communicate intermediate results. The interprocessor communication time may be substantial relative to the time needed exlusively for computations, so it is important to carry out the information exchange as efficiently as possible.

Two of the most frequent communication tasks are the *multinode broadcast* (MNB) and the *total exchange* (TE). The first task involves broadcasting a packet (the same packet) from every node to all the other nodes. It arises, for example, in iterations of the form

$$x = f(x), \tag{1}$$

where each processor computes an entry (or some entries) of the vector x. At the end of each iteration it is necessary that each processor broadcasts the updated value of the component that it computes to all other processors in order to be used at the next iteration. A MNB arises also in Bellman-Ford iterations for the single destination shortest path problem;

$$D_i = \min_j (a_{ij} + D_j), \quad i = 1, 2, \ldots, N, \tag{2}$$

where D_i is the estimated distance from node i to a given node and a_{ij} is the cost of the arc (i, j). Of course, the network for which the shortest paths are computed does not have any relation with the multiprocessor network which we use (e.g., the shortest path algorithm may be used as a subroutine in some other algorithm). We assume that the ith processor of an N-processor computer stores the arc costs a_{ij}, $j = 1, 2 \ldots, N$ and updates the distance estimate D_i. At the end of an iteration it is necessary for every node i to send the updated value of D_i to every other node, which for dense networks is again a multinode broadcast.

The total exchange is the communication task where each node has to send a *personalized* (different) packet to each one of the other nodes. An example where the total exchange arises is the transposition of a matrix, when each processor stores, say, a column of the matrix. Then every processor i has to send the (i, k)th entry of the matrix to processor k, for all k, which is a total exchange.

Algorithms to perform a MNB or TE have been studied by several authors under a variety of assumptions on the communication network connecting the processors. Saad and Shultz [SaS85], [SaS86] were the first to consider these problems and to propose corresponding routing algorithms. Johnsson and Ho [JoH89] have developed minimum and nearly minimum completion time algorithms for similar routing problems as those of Saad and Schultz but using a different communication model and a hypercube network. Bertsekas et al [BOS91], and Bertsekas and Tsitsiklis [BeT89] have used the communication model of Saad and Shultz to derive minimum completion time algorithms for a MNB or TE in a hypercube. Varvarigos and Bertsekas [VaB90a] considered a class of communication tasks, called isotropic tasks, in hypercubes and d-dimensional wraparound meshes, and devised algorithms which are optimal jointly with respect to completion time, average packet delay, and storage requirements. This class of tasks includes the TE as a special case. The same authors in [VaB90b] proved that the multinode broadcast task when packets have random lengths can be executed in near-optimal time with high probability. Several other works deal with various communication problems and network architectures related to those discussed in the present paper; see [Ede91], [HHL88], [Ho90], [KVC88], [LEN90], [McV87], [SaS88], and [Top85].

In iterations of the kind given in (1) or in (2) it is very probable that only some of the components of the vector x or D change appreciably during an iteration. As these iterations approach their convergence point, fewer and fewer of the processors need to broadcast the updated values of the components of x or D that they compute. This gives rise to a task, where a strict (but unpredictable) subset of the processors have to broadcast a packet. We call this task a *partial multinode broadcast* (or PMNB for brevity). The PMNB task aside from being important on its own merit, is also a critical subroutine of the dynamic broadcast schemes that we propose in Section 6. The PMNB task arises also in clustering algorithms (see [RaS90], Chapter 5, where the M nodes that store the coordinates of the centers of the clusters broadcast them after each iteration) and other problems. Because of its many applications we believe that the PMNB deserves a position among the prototype tasks of a communication library.

Similarly, during the transposition of a matrix that has both sparse and dense columns, it is more efficient if the nodes storing sparse columns do not participate in the TE, but send instead their packets as ordinary traffic through the 1-1 routing algorithm used by the machine. Since most large problems involve sparse matrices one can see that this situation arises frequently, giving rise to the *partial exchange task* (PE), where only M nodes send a (separate) packet to every other node. A task which is related to the PE is the *partial gather* (PG) task. In this task, M arbitrary nodes have to receive a (different) packet from every other node of the network (combining packets originated at different nodes is not allowed). Note that the PG task is *dual* to the PE task; if we find an algorithm to execute the PE we immediately get an algorithm of the same time complexity that executes the PG. In the transposition of a matrix stored by columns in a multiprocessor network, a PG arises when the matrix has only M dense rows. By combining a PE and a PG algorithm we get an algorithm that transposes a matrix which has M_1 dense columns and M_2 dense rows. The dense rows and columns can be arbitrary. This sparcity pattern arises very frequently in applications. The smaller M is, the less efficient a full MNB or TE algorithm would be and the more necessary it becomes to employ algorithms that are specially designed for partial tasks.

The main focus of the paper is to propose optimal and near-optimal communication algorithms for the partial multinode broadcast and the partial exchange tasks in d-dimensional meshes, with or without wraparound. These problems are considered for the first time here. PMNB algorithms have previously been studied for hypercubes in [Sta91] and [VaB92]. The partial exchange problem was also considered for hypercube networks in [Var92]. In the course of solving the mesh PMNB problem, we formulated and solved two problems, called the mesh packing and monotone routing problems, which are of broader interest. In what follows, to avoid confusion, we call a (d-dimensional) mesh *with* wraparound a *torus* and a mesh *without* wraparound an *array.*

We will say that an algorithm is *near-optimal* if the potential loss of optimality with respect to completion time is of strictly smaller order of magnitude than the optimal completion time itself. We generally prove that an algorithm is near-optimal by showing that the leading term of its worst case time complexity (including the corresponding constant factor) is the same with the leading term of an expression which is a lower bound to the time required by any algorithm. We generally derive the optimal completion time by deriving a lower bound to the completion time of any algorithm and by constructing an algorithm that attains the lower bound; this latter algorithm is said to be *optimal.* We will say that an algorithm is of *optimal order* if its worst case time complexity is asymptotically within a constant factor from the optimal value.

One of the main contributions of the paper is the development of near-optimal algorithms for a partial multinode broadcast in a d-dimensional torus and in a d-dimensional array. We propose algorithms for two different communication models. In the first model, packets can be split and recombined at the destination without any overhead. In the second model the splitting is not allowed, and messages are always transmitted as one packet. We also present the first partial exchange algorithm of optimal order for the 2-dimensional array.

The PMNB communication task is a *static* broadcasting task, that is, it assumes that at time $t = 0$ some nodes have to broadcast a packet. In this paper, we also consider the dynamic version of this problem. We assume that broadcast requests are generated at each node according to a Poisson process with rate λ, independently of the other nodes. We propose routing schemes for d-dimensional tori and arrays that work under such a dynamic environment, and we evaluate their performance. The performance criterion used is the average packet delay, that is, the time that elapses on the average between the arrival of a packet to be broadcast at a node and the completion of the broadcast of the packet. Dynamic broadcasting schemes are also studied in the companion paper [VaB92], and in [Sta91] for hypercube networks. The scheme to be proposed for d-dimensional meshes is similar to the one given in [VaB92] for hypercubes. It is stable for load asymptotically equal to the maximum possible, and its average delay is of the order of the diameter of the mesh for any load in the stability region.

The organization of the paper is the following. Section 2 shows how a mesh without wraparound can simulate a mesh with wraparound, and presents the first (strictly) optimal multinode broadcast algorithm for 2-dimensional meshes without wraparound. We also present a theorem concerning arbitrary broadcasts in rings and linear arrays. In Section 3 we define and solve the packing and the monotone routing problems, in a d-dimensional mesh. In Section 4 we present near-optimal algorithms to execute a partial multinode broadcast in d-dimensional meshes. In particular, in Subsection 4.1 we give an algorithm where packets can be split, while in Subsection 4.2 we give an algorithm that does not require packet splitting. In Section 5 we present an algorithm for a partial exchange in a 2-dimensional array. In Section 6 we give the dynamic broadcasting scheme and evaluate its performance.

2. SOME PRELIMINARY RESULTS

The d-dimensional mesh, denoted by M_d, consists of $N = p^d$ processors arranged along the points of a d-dimensional space that have integer coordinates numbered from 0 to $p-1$. Along the ith dimension, obtained by fixing coordinates $(x_{d-1}, \dots, x_{i+1}, x_{i-1}, \dots, x_0)$ there are p processors with identities $(x_{d-1}, \dots, x_i, \dots, x_0)$, $x_i = 0, 1, \dots, p-1$. Two processors $(x_{d-1}, \dots, x_i, \dots, x_0)$ and $(y_{d-1}, \dots, y_i, \dots, y_0)$ are connected by a (two-directional) link if and only if for some i we have $|x_i - y_i| = 1$ and $x_j = y_j$ for all $j \neq i$. In addition to these links in the d-dimensional mesh *with wraparound* (also called a *torus*), all links of the type

$$((x_{d-1}, \dots, x_{i+1}, 0, x_{i-1}, \dots, x_0),\ (x_{d-1}, \dots, x_{i+1}, p-1, x_{i-1}, \dots, x_0))$$

are present. The latter links do not exist in the d-dimensional mesh *without* wraparound (also called an *array*). The set of nodes of an array (or torus) whose identities differ from the identity of node $x = (x_{d-1}, \dots, x_{i+1}, x_{i-1}, \dots, x_0)$ only in the ith digit is called the *i-level linear array* (or *ring*, respectively) of node x, and is denoted by $(x_{d-1}, \dots, x_{i+1}, *, x_{i-1}, \dots, x_0)$. The node with identity $(x_{d-1}, x_{d-2}, \dots, x_0)$ is also represented by the base p number of the form $x = x_{d-1}x_{d-2} \cdots x_0$. The 0th digit is considered the least significant digit of the above representation. A link connecting two nodes which differ only in the ith digit is called a *link of dimension i*.

Packets can be simultaneously transmitted along a link in both directions. Only one packet can travel along a link in each direction at any one time; thus, if more than one packet are available at a node and are scheduled to be transmitted on the same incident link of the node, then only one of these packets can be transmitted at the next time period, while the remaining packets must be stored at the node while waiting in queue. Each node is assumed to have infinite storage space. All incident links of a node can be used simultaneously for packet transmission and reception. Each packet requires one unit of time for transmission over a link. We consider both a model where packets can be split at the origin, and be recombined at the destination, and a model where packets cannot be split; in the first model if a packet is split in d parts, each of them requires $1/d$ units of time to be transmitted over a link.

We start by describing how a torus can be simulated by an array. A linear (i.e. one-dimensional) array can simulate a ring of the same size with a slowdown factor of two. This can be done as indicated in Fig. 1. By using this fact, a torus of any dimension can be simulated by an array of the same size and dimension with a slowdown factor of two as shown again in Fig. 1.

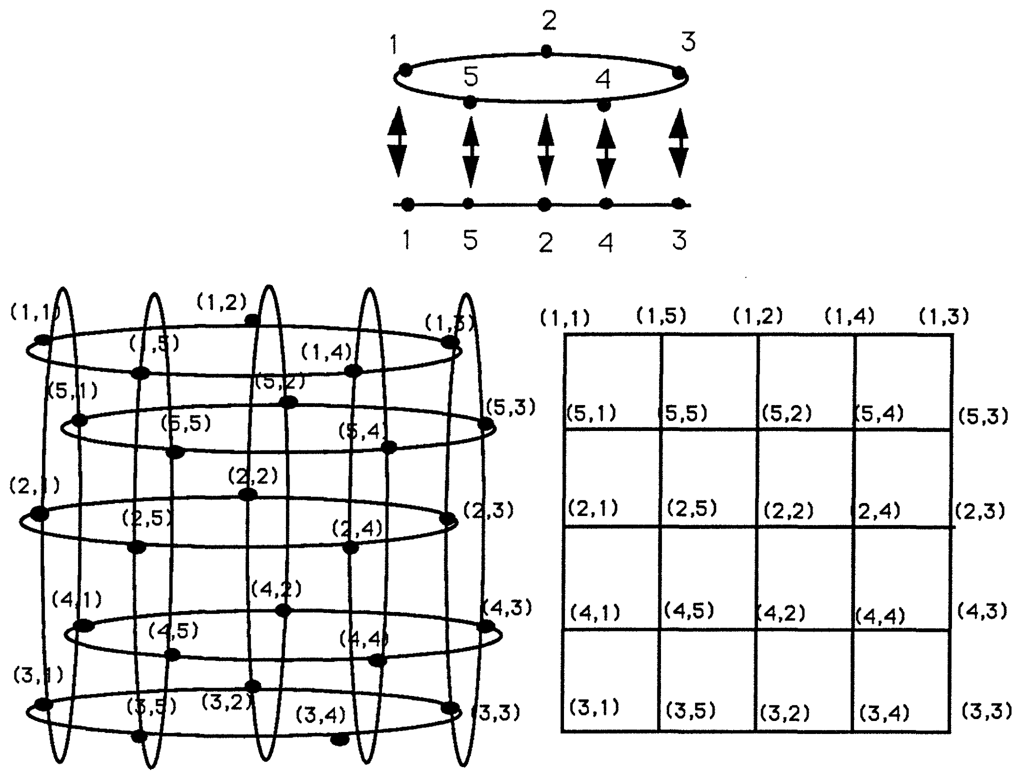

Figure 1: The upper part of the figure shows how a ring can be simulated by a linear array with a factor of two slowdown. This idea is easily extended to the simulation of a d-dimensional torus by a d-dimensional array, as can be seen from the lower part of the figure.

The optimal time T^t_{MNB} to execute a (full) multinode broadcast in a $p \times p$ torus was found in [BeT89] to be equal to

$$T^t_{MNB} = \frac{p^2}{4} = \frac{N}{4}$$

if p is even and

$$T^t_{MNB} = \frac{p^2-1}{4} = \frac{N-1}{4}$$

if p is odd.

The following theorem gives a corresponding result for the MNB task in a 2-dimensional array.

Theorem 1: The minimum time T^a_{MNB} required to execute a (full) multinode broadcast in a 2-dimensional array is exactly twice the minimum time T^t_{MNB} required to execute a multinode broadcast in a 2-dimensional torus of the same size, that is

$$T^a_{MNB} = 2T^t_{MNB} = \left\lfloor \frac{N}{2} \right\rfloor .$$

Proof: As we indicated earlier, a mesh without wraparound can simulate with a slowdown factor of two a mesh with wraparound of the same size. Each step of a torus can be simulated in two steps by an array even if all the links of the torus are simultaneously used. This gives the inequality $T^a_{MNB} \leq 2T^t_{MNB}$. Since node $(0,0)$ has only two neighbors and receives $N-1$ packets we have $T^a_{MNB} \geq (N-1)/2$. This together with the fact that T^a_{MNB} has to be integer proves that

$$T^a_{MNB} = 2T^t_{MNB} = \left\lfloor \frac{N}{2} \right\rfloor .$$

Q.E.D.

The next theorem deals with arbitrary broadcasts in rings and arrays.

Theorem 2: Consider a linear (one-dimensional) array of p nodes, where each node has a certain (not necessarily the same) number of packets to broadcast to all other nodes. Let K be the total number of packets in the array. Then the broadcasts can be completed in time less than or equal to

$$K + p - 1.$$

In a ring of the same size, the task requires half this time, provided that packets can be split into two parts without additional overhead.

Proof: Consider the following algorithm. Each node immediately transmits over its left (right) neighbor every packet that it receives from its right (left) neighbor. Whenever, a node does not receive anything from its left (right) neighbor it sends one of its own packets to the right (left). In other words, each node passes in the same direction the packets that come to it, and inserts a packet of its own whenever it sees an empty slot. Note that a packet is never delayed after it starts getting transmitted. In order to evaluate the time complexity we can focus on one direction, say the one going from left to right. Since there are K packets in the linear array, the packet can be delayed at most K times before starting transmission in this direction, and after at most $p-1$ slots it will have arrived to all the nodes in that direction. To prove the result about the ring, we can split each packet in two parts, each requiring 0.5 units of time. The ring can be viewed as two edge-disjoint unidirectional linear arrays, and by similar arguments, applied to each direction, the result follows. **Q.E.D.**

Remark: In the case where each node of the linear array has at most one packet, all the nodes can broadcast their packet in time $p-1$ (see [BeT89]). In the case of a ring the same task requires time $\lfloor p/2 \rfloor$, if the packets cannot be split, and $(p-1)/2$ if the packets can be split into two parts without overhead.

3. PACKING AND MONOTONE ROUTING FOR MESHES

In this section we present some new results on routing in meshes. These results will be useful in the PMNB algorithms to be given later, but they are also interesting on their own right. The problems to be addressed will be referred to as the *packing* and the *monotone routing* problems. We expect these results to be useful in a variety of algorithms, given the wide use that corresponding results for butterfly networks have had (see, e.g., [Lei92], pp. 524-538).

Theorem 3 (Mesh Packing Routing Theorem): Let $s^{(i)}$, $i = 0, 1, \ldots, K-1$, be nodes of a d-dimensional array such that $s^{(0)} < s^{(1)} < \cdots < s^{(K-1)}$. Consider the communication task, where each node $s^{(i)}$ sends a packet to processor i. This can be done without conflicts through a greedy scheme in time $d(p-1)$. This greedy scheme uses only links of dimension j during steps $jp, jp+1, \ldots, jp+p-1$, $j = 0, \ldots, d-1$.

Proof: The greedy routing consists of d phases, each of which has duration *exactly* $p-1$ steps. During phase l, $l = 0, 1, \ldots, d-1$, the packet generated at node $s^{(i)}$ corrects its lth digit to be equal to the lth digit of i by crossing in the natural way the links of dimension l. Thus, at the beginning of phase l the packet is at node $s^{(i)}_{d-1} s^{(i)}_{d-2} \cdots s^{(i)}_{l} i_{l-1} i_{l-2} \cdots i_0$, and at the end of phase l the packet is at node $s^{(i)}_{d-1} s^{(i)}_{d-2} \cdots s^{(i)}_{l+1} i_l i_{l-1} \cdots i_0$, where $s^{(i)}_{d-1} s^{(i)}_{d-2} \cdots s^{(i)}_{0}$ and $i_{d-1} i_{d-2} \cdots i_0$ are the identities of $s^{(i)}$

and i, respectively.

We will prove that with this routing scheme no two packets are at any time at the same node. We will use induction on d. For $d = 1$ (linear array) this is obvious. Assume that it is also true for $d-1$-dimensional arrays. Observe that $s^{(i)} - s^{(j)} \geq i - j$. At the end of phase 0 two packets $s^{(i)}$ and $s^{(j)}$, with $s^{(i)} > s^{(j)}$, can be at the same node only if their base p representations differed only at the 0th digit. In this case we have $s^{(i)} - s^{(j)} \leq p - 1$, which gives $i - j \leq p - 1$. Therefore i and j also differ in the 0th digit. Since at the end of phase 0 the two packets will be at nodes $s^{(i)}_{d-1}s^{(i)}_{d-2}\cdots s^{(i)}_{1}i_0$ and $s^{(j)}_{d-1}s^{(j)}_{d-2}\cdots s^{(i)}_{1}j_0$ with $i_0 \neq j_0$, they cannot be at the same node. Consider now the p subarrays $S_0, S_1, \ldots, S_{p-1}$ of dimension $d-1$ defined as follows;

$$S_k = \{s \mid s_0 = k\}.$$

During phases $1, 2, \ldots, d-1$ the packets will remain at the same one of the above submeshes at which they were at the end of phase 0, because no links of dimension 0 are crossed. Focusing on one of these subarrays and forgetting about the 0th digit, which is of no significance any more, we see that the routing problem within each of these arrays is a packing problem of dimension $d-1$. Using the induction hypothesis, we see that packets are not at any time at the same node during phases $1, 2, \ldots, d-1$ either. **Q.E.D.**

The next theorem treats a more general routing problem, which we call the *mesh monotone routing* problem.

Theorem 4 (Mesh Monotone Routing): Let $s^{(i)}$ and $v^{(i)}$, $i = 0, 1, \ldots, K-1$, be nodes of a d-dimensional array such that $s^{(0)} < s^{(1)} < \cdots < s^{(K-1)}$ and $v^{(0)} < v^{(1)} < \cdots < v^{(K-1)}$. Consider the communication task, where each node $s^{(i)}$ has to send a packet to processor $v^{(i)}$. This can be performed through a greedy scheme, without conflicts, in time $2d(p-1)$. The greedy scheme uses only links of dimension j during steps $jp, jp+1, \ldots, jp+p-1$, with $0 \leq j \leq d-1$, and only links of dimension $2d - j$ during steps $jp, jp+1, \ldots, jp+p-1$, with $d \leq j \leq 2d-1$.

Proof: For each i, $i = 0, 1, \ldots, K-1$, we initially send the packet of node $s^{(i)}$ to the intermediate node i. This is a packing problem and takes time $d(p-1)$ (Theorem 3). In a second phase, called *unpacking phase*, the packet of node i is sent to node $v^{(i)}$. This is the reverse of a packing problem and can be done by crossing the dimensions in the opposite order (from higher to lower dimensions) in time $d(p-1)$ again. **Q.E.D.**

Theorems 3 and 4 assume that packets $s^{(i)}$ know their rank i. The rank can be computed in time $2(p-1)dt_p$, where t_p is the time required for a single parallel prefix step, through a parallel prefix operation as explained in various references (see e.g. [Lei92], pp. 37-44), and described briefly in Phase 1 of the PMNB algorithm given in the next section.

4. PARTIAL MULTINODE BROADCAST IN D-DIMENSIONAL TORI AND ARRAYS

In this section we consider the problem where M arbitrary nodes of a d-dimensional mesh with $N = p^d$ nodes want to broadcast a packet to all the other nodes. We call these M nodes *active nodes.* Let T^t_{PMNB} be the optimal time required for the partial multinode broadcast in a d-dimensional torus, and T^a_{PMNB} be the corresponding time for a d-dimensional array. T^t_{PMNB} and T^a_{PMNB} may actually depend on the identities of the M nodes that want to broadcast. A lower bound, however, is always

$$T^t_{PMNB} \geq \frac{M-1}{2d}, \tag{3}$$

and

$$T^a_{PMNB} \geq \frac{M-1}{d}, \tag{4}$$

where d is the dimension of the mesh. To see that, note that in a d-dimensional array (or torus) node $00\cdots0$ has only d input ports (or $2d$ input ports, respectively), and has to receive at least $M-1$ packets.

One way to execute the partial multinode broadcast is to perform a full multinode broadcast (with dummy packets for the nodes that have nothing to broadcast). The optimal completion time of the MNB in a d-dimensional torus with $N = p^d$ nodes, when each packet requires one time unit (or slot) to be transmitted over a link is $\lceil \frac{N-1}{2d} \rceil$ time slots. Thus an upper bound for T^t_{PMNB} is

$$T^t_{PMNB} \leq \left\lceil \frac{N-1}{2d} \right\rceil .$$

Since a d-dimensional array can simulate a d-dimensional torus with a slowdown factor of two, an upper bound on T^a_{PMNB} is

$$T^a_{PMNB} \leq 2 \left\lceil \frac{N-1}{2d} \right\rceil .$$

When $M << N$ the previous algorithms are inefficient as the gaps between the upper and the lower bounds suggest. In this section we present communication algorithms that execute the PMNB task in d-dimensional meshes with or without wraparound in near-optimal time. In Subsection 4.1 we present an algorithm which assumes that packets can be split at the origin, and be recombined at the destination without any overhead. This algorithm executes the PMNB task in time

$$\frac{M}{2d}\frac{N-1}{N} + 2d(p-1)t_p + 1.5(p-1), \tag{5}$$

for a d-dimensional torus with $N = p^d$ nodes, and in time

$$\frac{M}{d}\frac{N-1}{N} + 2d(p-1)t_p + 2(p-1), \tag{6}$$

for a d-dimensional array of the same size, where t_p is the time required for a single parallel prefix step. For the case where the splitting of packets is undesirable (because of the overhead introduced, and the cost of packet reassembling), we will present in Subsection 4.2 an algorithm that avoids the splitting of packets, and executes the PMNB task in time less than

$$\left\lceil \frac{M}{d} \right\rceil \frac{1}{p-1} \left\lceil \frac{p-1}{2} \right\rceil \frac{N-1}{N} + (p-1)d + d\left\lceil \frac{p-1}{2} \right\rceil + 4(p-1)dt_p, \tag{5$'$}$$

for a d-dimensional torus and in time

$$\left\lceil \frac{M}{d} \right\rceil + 2(p-1)d - 1 + 4(p-1)dt_p, \tag{6$'$}$$

for a d-dimensional array. Comparing Eqs. (5)-(5$'$) and Eqs. (6)-(6$'$) with the lower bounds (3) and (4), respectively, we see that the leading terms of the corresponding right hand sides have the same coefficient. So, the algorithms to be proposed are near-optimal.

4.1. A Near-optimal PMNB Algorithm with Splitting of Packets

The algorithm in this section assumes that packets can be split at the origin, and recombined at the destination without any overhead. Each packet requires one time slot for transmission over a link. If a packet is split in d parts, each of these parts requires $1/d$ time units to be transmitted over a link.

Let $s_1, s_2, \dots, s_M$, $M \leq N$, be the active nodes. The *rank* of a packet located at node s is defined as

$$r_s = \sum_{t<s} x_t \; - 1,$$

where x_t is equal to one if processor t has a packet to broadcast and zero otherwise.

We will first present a suboptimal partial multinode broadcast algorithm for the d-dimensional mesh, with or without wraparound. This algorithm will not make full use of the links of a mesh. We will then modify the algorithm to achieve efficient link utilization and near-optimal completion time. The suboptimal algorithm consists of three phases:

Phase 1 (Rank Computation Phase):

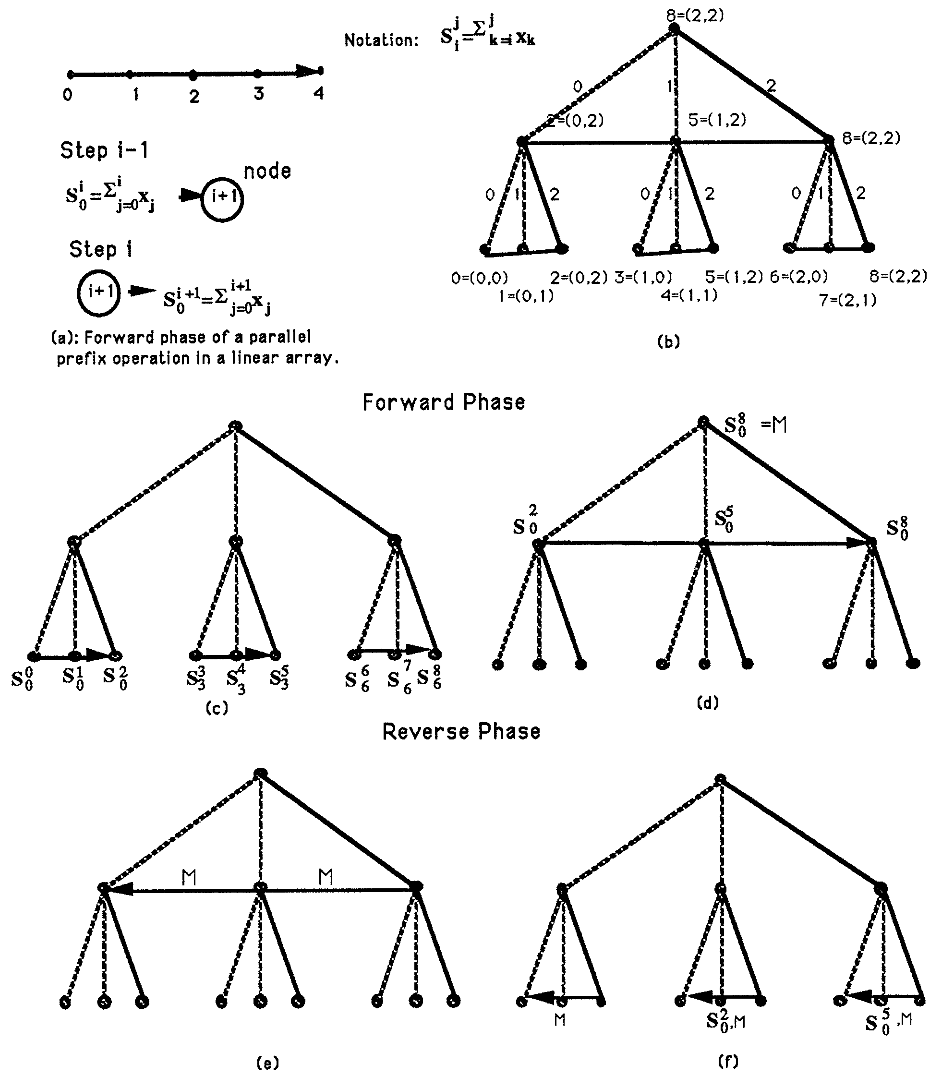
Notation: $S_i^j=\Sigma^j_{k=i}x_k$
0 1 2 3 4
Step i-1
$S_0^i=\Sigma^i_{j=0}x_j$
node
i+1
Step i
$S_0^{i+1}=\Sigma^{i+1}_{j=0}x_j$
(a): Forward phase of a parallel prefix operation in a linear array.
8=(2,2)
2=(0,2)
5=(1,2)
8=(2,2)
0=(0,0) 1=(0,1) 2=(0,2) 3=(1,0) 4=(1,1) 5=(1,2) 6=(2,0) 7=(2,1) 8=(2,2)
(b)
Forward Phase
S_0^0 S_0^1 S_0^2 S_3^3 S_3^4 S_3^5 S_6^6 S_6^7 S_6^8
(c)
S_0^8 =M
S_0^2 S_0^5 S_0^8
(d)
Reverse Phase
M M
(e)
M S_0^2,M S_0^5,M
(f)

Figure 2: Fig. 2a illustrates the operation of each node during a (forward) parallel prefix operation in a linear array. The partial sums $\sum_{k=0}^{i} x_k$ are obtained at each node i in time $p-1$. Figs. 2b-2f illustrate the parallel prefix operation in a mesh with $d=2$ and $p=3$. It consists of two phases (forward and reverse), each of which consists of d subphases. Each subphase is a parallel prefix operation in a linear array and requires $p-1$ steps. The total duration of the operation is $2d(p-1)$ steps. More precisely, Fig. 2b illustrates what we call tree representation of a mesh. An intermediate node is a root of a subtree whose leaves form a submesh of the original mesh. At the end of subphase l of the forward phase a node of level l from the bottom forms the partial sum of the values of the leaves under it. During the forward phase information moves from the bottom to the top, and from the left to the right. The notation S_i^j stands for $S_i^j=\sum_{k=i}^{j} x_k$. In the reverse phase, information moves from the top to the bottom and from the right to the left.

The rank r_s $(0 \leq r_s \leq M-1)$ of each active node s is computed. This can be done in $2(p-1)d$ steps for a d-dimensional array or a torus by performing a *parallel prefix operation* (see [Lei92], pp. 37-44) on a tree P, called *parallel prefix tree*, embedded in the mesh. The ith leaf of the tree from the left is the ith node of the mesh. The operation is described in Fig. 2 for a linear array and a mesh with $p = 3$ and $d = 2$. Note that during each step only links of a particular dimension are used. The packets involved in a parallel prefix operation are small (one byte of information), and require only t_p time units to be transmitted over a link. Thus it is reasonable to assume that $t_p \leq 1$, where one time unit is the time required to transmit a whole packet over a link; in fact it is reasonable to expect that in many parallel machines we have $t_p << 1$. Thus Phase 1 takes $2(p-1)dt_p$ time units to be completed.

Phase 2 (Packing Phase):

The packet of node s and rank r_s is sent to processor r_s, where r_s is interpreted as a p-ary number. This is a mesh packing problem, and can be performed in $(p-1)d$ time units according to Theorem 4.

Phase 3 (Broadcast Phase):

The broadcast phase consists of d subphases $l = 1, 2 \ldots, d$. During each subphase l, every node $r = r_{d-1}r_{d-2}\cdots r_0$ broadcasts (in any order) to all the nodes in the ring or linear array (depending on whether we are considering a mesh with or without wraparound) $(r_{d-1}\cdots r_{d-l+1} * r_{d-l-1}\cdots r_0)$, the packets that were located at the node at the beginning of Phase 3 plus the packets that the node has received during all the previous subphases. The broadcast algorithms used are those described in Theorem 2.

During subphase 0 the nodes have (at most) one packet and this is the only one they broadcast. Phase 3 is easy to implement since the current subphase l is easily known.

To prove that the algorithm delivers the packets to all the nodes, it is useful to introduce some new notation. Let $\beta = \beta_{d-1}\beta_{d-2}\cdots\beta_0$ be a p-ary number of length d. We denote by $S_l(\beta) = (*^l\beta_{d-l-1}\beta_{d-l-2}\cdots\beta_0)$ the submesh of the nodes whose $d-l$ less significant digits are equal to the $d-l$ less significant digits of β.

The next theorem proves that the previous algorithm actually executes the PMNB task.

Theorem 5: For each $\beta \in \{0, 1, \ldots, p-1\}^d$, at the end of subphase l of Phase 3, $l = 1, 2, \ldots, d$, each node in submesh $S_l(\beta)$ has received a copy of every packet located at the beginning of Phase 3 at some node in $S_l(\beta)$, completing a PMNB within each of these submeshes.

Proof: The proof will be done by induction on l. For $l = 0$ (i.e., at the beginning of Phase 3 of the algorithm) it holds trivially since every node has its own (if any) packet. Assume it is true for some l. Every submesh $S_l(\beta)$ is composed of the p submeshes $S_{l-1}(\beta_{d-1}\cdots\beta_{d-l+1}0\beta_{d-l-1}\cdots\beta_0)$,

$S_{l-1}(\beta_{d-1}\cdots\beta_{d-l+1}1\beta_{d-l-1}\cdots\beta_0)$, ..., $S_{l-1}(\beta_{d-1}\cdots\beta_{d-l+1}(p-1)\beta_{d-l-1}\cdots\beta_0)$. During subphase l every node in one of these submeshes broadcasts to all nodes in its $(d-l)$-level linear array (or ring) all the packets it has received during the previous subphases, together with its own packet. This together with the induction hypothesis proves the theorem. **Q.E.D.**

Letting $l = d$ we find that at the end of subphase d each packet has been broadcast to all the nodes, and therefore, the PMNB has been completed.

The next lemma calculates the time complexity of Phase 3.

Lemma 1: Phase 3 of the algorithm requires at most

$$\frac{N-1}{N}\frac{M}{\gamma}+\frac{(p-1)d}{\gamma}$$

time units, where $\gamma = 1$ for the d-dimensional array, and $\gamma = 2$ for the d-dimensional torus.

Proof: We denote by T_l the duration of subphase l, and we let $m = \lceil \log_p M \rceil$. At the beginning of Phase 3 only nodes $0, 1, \ldots, M-1$ have a packet. From Theorem 5 we know that just before the beginning of phase l, node $s = s_{d-1}s_{d-2}\cdots s_0$ has received all the packets originally located at nodes in the submesh $(*^{l-1}s_{d-l}s_{d-l-1}\cdots s_0)$. The number of these packets is equal to the cardinality of the set

$$\mathcal{W}_l(s) = \{w = w_{d-1}w_{d-2}\cdots w_0 \mid 0 \le w \le M-1,\ w_{d-l} = s_{d-l},\ w_{d-l-1} = s_{d-l-1}, \ldots, w_0 = s_0\}.$$

During subphase l, node s will broadcast these packets to the nodes in its $(d-l)$-level linear array or ring. Since a multinode broadcast in a linear array requires $p-1$ steps, while in a ring it requires $(p-1)/2$ steps (see the remark following Theorem 5), we have

$$T_l \le \frac{p-1}{\gamma}\max_s |\mathcal{W}_l(s)|,$$

where $\gamma = 1$ for d-dimensional arrays, $\gamma = 2$ for d-dimensional tori, and $|\cdot|$ denotes the cardinality of a set. Let $s' = s_{d-l}p^{d-l} + s_{d-l-1}p^{d-l-1} + \cdots + s_0$. The cardinality of $\mathcal{W}_l(s)$ is equal to the number of integers between 0 and $M-1-s'$, which are divisible by p^{d-l+1}. Thus

$$\max_s |\mathcal{W}_l(s)| \le \max_s \left\lceil \frac{M-1-s'}{p^{d-l+1}} \right\rceil \le \left\lceil \frac{M}{p^{d-l+1}} \right\rceil.$$

The total duration of Phase 3 satisfies

$$\begin{aligned}\text{Duration of Phase 3 } &= \sum_{l=1}^{d} T_l \le \frac{p-1}{\gamma}\sum_{l=1}^{d}\left\lceil \frac{M}{p^{d-l+1}} \right\rceil \\ &\le \frac{p-1}{\gamma}\left(d + M\sum_{l=1}^{d}\frac{1}{p^l}\right) \\ &= \frac{(p-1)d}{\gamma} + \frac{M}{\gamma}\left(1-\frac{1}{p^d}\right).\end{aligned}$$

Q.E.D.

Adding up the duration of Phases 1, 2 and 3 we obtain the following lemma:

Lemma 2: The partial multinode broadcast task can be executed in a d-dimensional torus with $N = p^d$ processors in

$$T^t_{PMNB} \leq \frac{M}{2}\frac{N-1}{N} + 2d(p-1)t_p + 1.5(p-1)d$$

time units, where M is the number of active nodes. Similarly, the PMNB task can be executed in a d-dimensional array with $N = p^d$ processors in

$$T^a_{PMNB} \leq M\frac{N-1}{N} + 2d(p-1)t_p + 2(p-1)d$$

time units.

The PMNB algorithm that we described so far is not of optimal order as the gap between the lower bounds of Eqs. (3) and (4), and the results of Lemma 2 indicate. In fact, they are suboptimal by a factor of roughly d. This is due to the fact that at each step only links of a particular dimension are used. In the next theorem we modify the algorithms so that all dimensions are used at the same time, and near-optimal completion time is achieved.

Theorem 6: The partial multinode broadcast task can be executed in a d-dimensional torus with $N = p^d$ processors in

$$T^t_{PMNB} \leq \frac{M}{2d}\frac{N-1}{N} + V_t \tag{7}$$

time units, where M is the number of active nodes, and

$$V_t = 2d(p-1)t_p + 1.5(p-1).$$

Similarly, the PMNB task can be executed in a d-dimensional array with $N = p^d$ processors in

$$T^a_{PMNB} \leq \frac{M}{d}\frac{N-1}{N} + V_a \tag{8}$$

time units, where

$$V_a = 2d(p-1)t_p + 2(p-1).$$

Proof: We call the PMNB algorithm analyzed in Lemmas 1 and 2 algorithm $\mathcal{A}_0$. At each step of Phases 1, 2, and 3 of $\mathcal{A}_0$, only links of a particular dimension are used. Indeed, it can be seen from Fig. 2 that during each step of the parallel prefix phase only links of a particular dimension are used. Similarly, in the packing phase, only links of a particular dimension are used at each step, as indicated in Theorem 4. Finally, during subphase l of the broadcast phase only links of dimension $d-l$ are used.

For any c, consider now another PMNB algorithm, referred to as algorithm $\mathcal{A}_c$. According to $\mathcal{A}_c$ a packet is transmitted over the link of dimension $(l+c)$ mod d of its current location, whenever the same packet would be transmitted under the $\mathcal{A}_0$ algorithm over the lth-dimensional link of its current location. Since $\mathcal{A}_c$ is identical to $\mathcal{A}_0$ after appropriately renaming the mesh dimensions (and the nodes), and since $\mathcal{A}_0$ performs the PMNB independently of the location of the M active nodes, we conclude that $\mathcal{A}_c$ also executes the PMNB task, and requires the same amount of time as $\mathcal{A}_0$. [3]

Using simultaneously all the algorithms $\mathcal{A}_0$, $\mathcal{A}_1$, ..., $\mathcal{A}_{d-1}$ we can find a new algorithm which requires the amount of time claimed in the theorem. In particular, each packet is split into d parts, called *mini packets.* Each mini packet is assigned a distinct integer c between 0 and $d-1$, called class. The mini packets of class c are routed according to algorithm $\mathcal{A}_c$. Packets of different classes use different mesh dimensions at any time. According to our communication model, a mini packet requires $1/d$ time units for transmission over a link. Therefore, the theorem follows from Lemma 2. **Q.E.D.**

The terms V_t and V_a in Eqs. (7) and (8), respectively, are growing linearly with the dimension d. In practice, however, $2d(p-1)t_p$ is small, since t_p is very small. Indeed, at each step of a parallel prefix operation only one byte has to be transmitted between neighbors. Some parallel computers, such as the Connection Machine model CM-2 of Thinking Machines Corporation, the IBM/RP-3, and the NYU Supercomputer, have very efficient implementations of the parallel prefix, otherwise called "scan" operation ([Tur88], [Ble86]). Theoretically, however, the parallel prefix operation takes time proportional to the diameter.

No upper ceilings are needed in Eqs. (7) and (8), since we allow fragmented slots. Note also

[3] In the algorithm $\mathcal{A}_c$ the rank of an active node is defined in the following way. On the p-ary numbers of length d, we first define the *order with respect to class c*, $c \in \{0, 1, \ldots, d-1\}$ (denoted by $<_c$) as follows:

$s <_c t$ iff the right shift of s by c positions is less (with the usual order) than the right shift of t by c positions.

The *rank with respect to class c* of a packet located at node s is then defined as

$$r_s^c = \sum_{\{t: t <_c s\}} x_t \; - 1,$$

where x_t is equal to one if processor t has a packet to broadcast and zero otherwise. The parallel prefix tree P^c used in the calculation of r_s^c is the same with P, but with the digits of the nodes shifted by c positions.

that under the communication model used in this section (which allows the splitting of packets in d parts), a broadcast from a single node requires $\Theta(p)$ time units, instead of $\Theta(dp)$ which is the diameter. A near-optimal PMNB algorithm which does not use the splitting of packets is presented in the next subsection.

4.2. A Near-optimal PMNB Algorithm without Splitting of Packets

In this subsection we modify the previous algorithms in order to avoid the potential drawbacks of packet splitting. This is done at the expense of a slight increase in the complexity. Messages in this section require one time slot in order to be transmitted over a link, and are always transmitted as one packet.

The algorithm makes use of the algorithm $\mathcal{A}_c$ described in the proof of Theorem 6. Recall that algorithm $\mathcal{A}_c$ consists of three phases: the rank computation, the packing, and the broadcast phases.

Class Computation Part:

The rank r_s, $0 \leq r_s \leq M-1$, $s \in \{s_1, s_2, \ldots, s_M\}$, of each packet is computed through a parallel prefix operation. This requires $2d(p-1)t_p$ time units. The packet of node s is assigned a *class number* $c = r_s$ mod d.

Main Part:

The packets of class c are routed according to algorithm $\mathcal{A}_c$. Only packets of class c take part in the rank computation phase, or in any other phase of $\mathcal{A}_c$.

To estimate the complexity of the algorithm, we first note that each class has at most $\lceil M/d \rceil$ packets. Lemma 1 has been proved under the assumption that a packet can be split into two parts. When packets cannot be split, it can be shown (by a proof similar to that in Lemma 1) that algorithm $\mathcal{A}_c$ requires time less than or equal to

$$M' \frac{1}{p-1} \left\lceil \frac{p-1}{\gamma} \right\rceil \frac{N-1}{N} + d \left\lceil \frac{p-1}{\gamma} \right\rceil,$$

where M' is the number of packets that participate in $\mathcal{A}_c$, $\gamma = 1$ for d-dimensional arrays, and $\gamma = 2$ for d-dimensional tori. Substituting $\lceil M/d \rceil$ instead of M', and adding the time required for the parallel prefix operations and the packing routing phase, we get

$$T_{PMNB} \leq \left\lceil \frac{M}{d} \right\rceil \frac{1}{p-1} \left\lceil \frac{p-1}{\gamma} \right\rceil \frac{N-1}{N} + (p-1)d + d \left\lceil \frac{p-1}{\gamma} \right\rceil + 4(p-1)dt_p.$$

The algorithm just presented for the PMNB task gives rise to an efficient algorithm for the MNB task. Indeed, a multinode broadcast can be treated as a partial multinode broadcast with $M = N$.

The class computation part and the rank computation phase are not necessary any more, since the class number and the rank of each packet are known in advance. The packing and the broadcast phases alone can execute the MNB in time less than or equal to

$$\left\lceil \frac{N}{d} \right\rceil \frac{1}{p-1} \left\lceil \frac{p-1}{\gamma} \right\rceil \frac{N-1}{N} + (p-1)d + d \left\lceil \frac{p-1}{\gamma} \right\rceil .$$

which is near-optimal for tori with p odd or arrays, and of optimal order for tori with p even. This MNB algorithm is apparently new.

5. PARTIAL EXCHANGE IN 2-DIMENSIONAL ARRAYS

In this section we present an algorithm to execute the partial exchange task in a $p \times p$ array (and, therefore also in a $p \times p$ torus). In particular, we initially assume that there are M nodes, called *active*, that want to send a personalized packet to each of the other nodes. The algorithm to be presented has time complexity which is of optimal order.

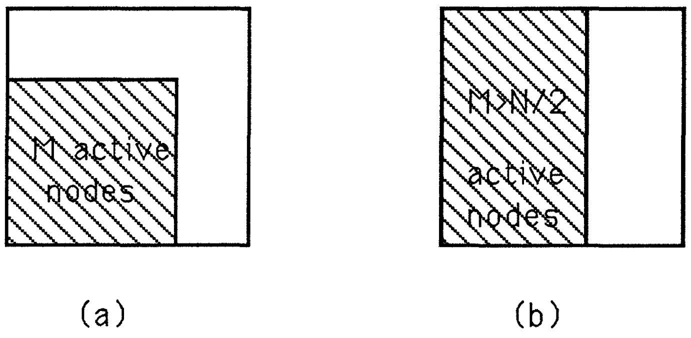

Figure 3: Position (a) of the active nodes corresponds to the first lower bound, while position (b) corresponds to the second lower bound.

We first present lower bounds on the minimum time required to execute the partial exchange in 2-dimensional array. Let us denote

$$m = \lfloor M^{1/2} \rfloor .$$

First, consider the case where m^2 of the M active nodes are in the $m \times m$ subarray $M_{0,0}$, where

$$M_{0,0} = \{(i,j) \mid 0 \le i \le m-1,\ 0 \le j \le m-1\}.$$

Then $m^2(N-m^2)$ packets have to cross the $2m$ links connecting $M_{0,0}$ to the rest of the array (since there are no wraparound links). This gives

$$T_{PE} \geq \frac{m(N-m^2)}{2}.$$

Thus,

$$T_{PE} \geq \frac{mN}{4}, \quad \text{for } M \leq N/2.$$

We next consider the case where $M \geq N/2$, and all the active nodes are at the left side of the mesh (see Fig. 3). Consider the packets that pass from left to right through the cut that bisects the mesh. At least $N^2/4$ packets cross the p links of this cut. Thus,

$$T_{PE} \geq \frac{Np}{4} \geq \frac{mN}{4}, \quad \text{for } M \geq N/2.$$

The previous bounds show that

$$T_{PE} \geq \frac{mN}{4} = \Omega(M^{1/2}N).$$

Before describing the algorithm, we introduce some notation. The nodes of the mesh are represented as pairs (i,j) with $0 \leq i,j \leq p-1$. The *row sum* (or *column sum*) of node (i,j) is defined as the number of active nodes of row i (or column j) and is denoted by r_i (or c_j, respectively).

We now describe the algorithm. It consists of two phases:

Phase 1 (Parallel Prefix Phase):

The row sums r_i and column sums c_j are computed. All r_i's and c_j's can be found in p steps by concurrently performing a parallel prefix operation within each row and column. A row or column is a linear array, and can viewed as a tree rooted at a median node of that linear array of depth $\lfloor p/2 \rfloor$. The parallel prefix operation is performed with value equal to one for active nodes and zero for the other nodes. Phase 1 requires pt_p time units, where $t_p \leq 1$ is the time required for a single parallel prefix step.

Phase 2 (Exchange Phase):

The set of active nodes is partitioned into the two sets R and C where

$$R = \{(i,j) \text{ active} \mid r_i \leq c_j\}, \quad C = \{(i,j) \text{ active} \mid r_i > c_j\}.$$

The nodes in R or in C send each of their packets along the unique shortest path that first crosses horizontal (respectively, vertical) links exclusively, and then crosses vertical (respectively, horizontal) links exclusively. The order of transmission of the packets at each link is arbitrary subject to two restrictions:

a) Packets originating at nodes of R (or of C) have priority on the horizontal (respectively, vertical) links over packets originating at nodes in C (respectively, R).

b) Transmission is non-wasting in the sense that no link remains idle if there is a packet waiting at the queue of the link.

We have the following lemma:

Lemma 3: The number of nodes of R that belong to the same row are at most m.

Proof: Our proof is by contradiction. Suppose that for some i, the nodes $(i, j_1), (i, j_2), \ldots, (i, j_x)$ belong to R, and $x > m$. Then by the definition of the set R, $c_{j_k} \geq r_i \geq x > m$ for all $k = 1, 2, \ldots, x$. This implies that there are $x \geq m + 1$ columns, each of which has at least x active nodes. This is a contradiction since there are only $M < (m+1)^2$ active nodes. **Q.E.D.**

We claim that Phase 2 requires at most $2m(N - p) + 2(p - 1)$ time units. To see this, we first note that the number of packet originating at nodes of R that must cross at least one horizontal link of any given row i is at most $m(N - p)$. The reason is that by Lemma 3 there are at most m active nodes from R in row i and each of these nodes has a total of $N - p$ packets to send to nodes that belong to a different column. Since each packet increases the delay of another packet by at most one unit along the horizontal path, we see that the time required for all the packets originating at nodes in R to traverse completely the horizontal portion of their path is at most $m(N - p) + (p - 1)$. Similarly, the time required for all the packets originating at nodes in C to traverse completely the vertical portion of their path is also at most $m(N - p) + (p - 1)$.

Let us make the worst-case assumption that packets originating at nodes of R (or C) are delayed after completing their horizontal (respectively, vertical) transmissions so that their transmission starts after exactly $m(N - p) + (p - 1)$ time units. We will show that at most $m(N - p) + (p - 1)$ additional time units are needed to complete Phase 2. Indeed, at the end of the first $m(N - p) + (p - 1)$ time units, each node has at most $m(p - 1)$ packets originating at nodes in R to send over the vertical links. Therefore, at most $m(p - 1)p$ such packets remain to traverse the links of its column. This requires at most an additional $m(N - p) + (p - 1)$ time units.

Adding up the times required for each phase, and taking into account that $N = p^2$ and $m = \lfloor M^{1/2} \rfloor$, we find that the time T_{PE} required for the partial exchange in a 2-dimensional array satisfies

$$T_{PE} \leq 2\lfloor M^{1/2} \rfloor (N - p) + 2(p - 1) + pt_p.$$

Comparing this inequality with the lower bound found earlier, we see that our algorithm is of optimal order (within a factor of roughly 8 of being optimal).

6. DYNAMIC BROADCASTING SCHEMES

The PMNB task considered in Section 4 is *static* in the sense that it is executed only once starting at time $t = 0$. In this section we consider the *dynamic* version of this task. We assume that broadcast requests arrive at each node of a mesh according to a Poisson process with rate λ. We propose an algorithm that works well in such a dynamic environment, and evaluate its performance. The performance criterion used is the average packet delay, that is, the time between the arrival of a packet to be broadcast at a node and the completion of the broadcast of the packet. The same problem has been studied in [Sta91], and in a companion paper [VaB92] where dynamic schemes are proposed for the hypercube case, and their performance is analyzed. The dynamic scheme that we propose for meshes is simple and stable for network utilization very close to the maximum possible. Furthermore, for each fixed utilization in the stability region, the average delay is of the order of the diameter of the mesh, which is the best we could hope for.

Our scheme is essentially a repetition of partial multinode broadcasts, each starting when the previous one has finished. The PMNB algorithm that we will assume throughout this section is the one of Subsection 4.1 where packets are split; if the algorithm of Subsection 4.2, where packets are not split, is used we get corresponding results (slightly worse, however, especially in the case where p is even). The time axis is divided into PMNB intervals (see Fig. 4). Within each PMNB interval, a PMNB is executed, involving exactly one packet from each of the nodes that have a packet to broadcast at the start of the interval. Each PMNB interval is divided into two parts. The first part is called *reservation interval*, and consists of the parallel prefix and the packing phases of the PMNB algorithm of Section 4. Its duration can be upper bounded by a known constant that depends only on the size of the network, and is independent of the number of active nodes M. The second part of a PMNB interval is called *broadcast interval.* Its duration is known once M is known. Thus, even though the duration of each partial multinode broadcast is random (because packet arrivals are random), it is known to all the nodes of the network, because each node learns during the broadcast interval the number M of active nodes and, from there, the duration of the following broadcast interval. Therefore, if the nodes initiate the dynamic broadcast scheme at the same time, no further synchronization is needed.

It is important for the performance of the dynamic scheme that the duration of the PMNB algorithms given in Section 4 is *linear* in the number of active nodes M, with the constant of proportionality being the smallest possible. In particular the duration of the PMNB algorithm for

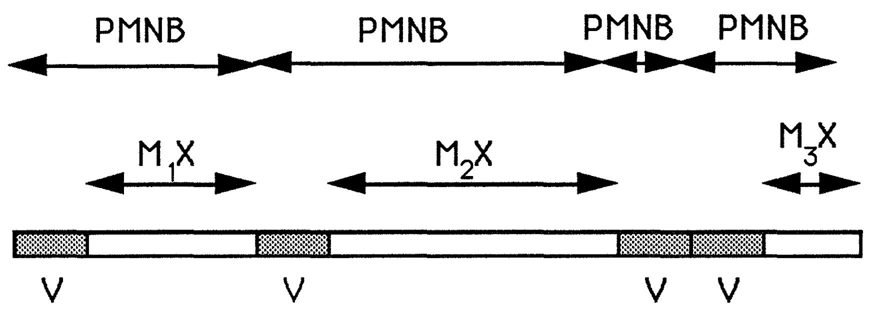

Figure 4: The dynamic broadcasting scheme. Each PMNB interval consists of two intervals: a reservation interval (marked by gray) of duration V, and a broadcast interval of duration MX, where M is the number of active nodes at the start of the PMNB interval.

a d-dimensional meshes with $N = p^d$ processors was found in Section 4 to be equal to

$$T_{PMNB} = XM + V,$$

where

$$X = \frac{1}{\gamma d}\frac{N-1}{N}, \tag{9}$$

$$V = 2d(p-1)t_p + \left(1 + \frac{1}{\gamma}\right)(p-1), \tag{10}$$

and $\gamma = 1$ for the d-dimensional array and $\gamma = 2$ for the d-dimensional torus.

In [VaB92] the following theorem was proved (in fact a slightly stronger theorem was proved there, but the following simplified version is adequate for our purposes):

Theorem 7 ([VaB92]): Let the arrivals of broadcast requests at a node of an N-processor network be Poisson with rate λ. Assume also that there exists a PMNB algorithm for that network, which executes the PMNB task in time that is at most

$$XM + V,$$

where M is the number of nodes that have a packet to broadcast and X, V are independent of M (they may depend on the size of the network). Let $\rho = \lambda N X$, and suppose that

$$1 - \rho - \lambda V > 0.$$

Then if this PMNB algorithm is used in a dynamic broadcasting scheme, in the way described earlier, the system is stable in the sense that the average packet delay is finite and satisfies

$$T \leq (1+\rho)\left(\frac{\rho X}{2(1-\rho-\lambda V)} + \frac{(1-\rho)V}{2(1-\rho-\lambda V)} + \frac{(1-\lambda V)V}{1-\rho-\lambda V}\right) + X. \tag{11}$$

For the d-dimensional mesh we have found algorithms that satisfy the conditions of Theorem 7. Thus, the average packet delay is bounded as in Eq. (11), with X and V given by Eqs. (9) and (10), respectively.

The scalar

$$\rho = \lambda N X = \frac{\lambda(N-1)}{\gamma d} \tag{12}$$

is called *mesh utilization factor*, for reasons that will become evident soon. To find necessary conditions for stability for any broadcasting scheme, consider a d-dimensional array or torus, the outgoing links of node $(00\cdots 0)$, and the traffic that passes through them. There are $2d$ such links for the torus, and d for the array. Thus, for stability we must have

$$\lambda(N-1) \leq \gamma d,$$

or

$$\rho \leq 1, \tag{13}$$

no matter what broadcasting scheme we use. For the torus ($\gamma = 2$) and a given load, ρ is equal to the ratio of the average number of transmissions per unit of time necessary to execute the broadcasts (each broadcast requires $N-1$ transmissions), over the total number of links of the network. For the array ($\gamma = 1$), which is not a symmetric network, ρ equals the average fraction of time during which the links of node $(00\cdots 0)$ have to be used under any broadcasting scheme.

Our algorithm is guaranteed to be stable for $\rho < 1 - \lambda V$. Using Eqs. (12) and (10) we find that

$$\rho < 1 - \rho\frac{\gamma d V}{N-1},$$

and

$$\rho < \frac{1}{1 + \frac{d\gamma\left(2d(p-1)t_p + (1+1/\gamma)(p-1)\right)}{p^d - 1}}. \tag{14}$$

The right hand side of the preceding inequality is very close to the maximum possible load, given by Eq. (13), that a d-dimensional array or torus could sustain. As the number of nodes p^d tends to infinity, d^2p/p^d tends to zero. Thus, the right hand side of Eq. (14) tends to one, which, in view of Eq. (13), is the maximum utilization that can be accomodated by the network.

For any fixed ρ in the stability region, Eq. (11) gives

$$T = O(V) = O(pdt_p + p),$$

where we have used Eq. (10). Since the diameter of a d-dimensional mesh is $\Theta(pd)$, the preceding relation gives $T = O(pdt_p + p) = O(t_p \cdot \mathit{diameter} + p)$ for any fixed ρ in the stability region. In particular, for light load ($\lambda \approx 0,\ \rho \approx 0$) we get from Eq. (11) that

$$T \leq 1.5V + X, \quad (\rho \approx 0).$$

REFERENCES

[BeG87] Bertsekas, D. P., and Gallager R. G., *Data Networks*, Prentice-Hall, 1987.

[BeT89] Bertsekas, D. P., and Tsitsiklis, J. N., *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, N.J., 1989.

[BOS91] Bertsekas, D. P., Ozveren C., Stamoulis, G. D., Tseng, P., and Tsitsiklis, J. N., "Optimal Communication Algorithms for Hypercubes," J. Parallel Distrib. Comput., Vol. 11, pp. 263-275.

[Ble86] Blelloch, G. E., "Scans as Primitive Parallel Operations," Proc. Int'l Conf. Parallel Processing, pp. 355-362, August 1986.

[Ede91] Edelman, A., "Optimal Matrix Transposition and Bit Reversal on Hypercubes: All-to-All Personalised Communication," J. Parallel Distrib. Comput., Vol. 11, pp. 328-331.

[HHL88] Hedetniemi, S. M., Hedetniemi, S. T., and Liestman, A. L., "A Survey of Gossiping and Broadcasting in Communication Networks," Networks, Vol. 18, pp. 319-349, 1988.

[Ho90] Ho, C. T., "Full Bandwidth Communications on Folded Hypercubes," Research Report RJ 7434 (69605), IBM Almaden Research Center, April 1990.

[JoH89] Johnsson, S. L., and Ho, C. T., "Optimum Broadcasting and Personalized Communication in Hypercubes," IEEE Trans. on Computers, Vol. C-38, 1989,. pp. 1249-1268.

[KeK79] Kermani, P., and Kleinrock, L., "Virtual Cut-Through: A New Computer Communicating Switching Technique," Comput. Networks, Vol. 3, pp. 267-286, 1979.

[KVC88] Krumme, D. W, Venkataraman, K. N., and Cybenko, G., "The Token Exchange Problem," Tufts University, Technical Report 88-2, 1988.

[Lei92] Leighton, F. T., *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes*, Morgan Kaufmann, San Mateo, CA, 1992.

[LEN90] Lan, Y., Esfahanian, A.-H., and Ni, L., "Multicast in Hypercube Multiprocessors," J. Parallel Distrib. Comput., pp. 30-41.

[McV87] McBryan, O. A., and Van de Velde, E. F., "Hypercube Algorithms and their Implementations," SIAM J. Sci. Stat. Comput., Vol. 8, pp. 227-287, 1987.

[RaS90] Ranka S., and Sahni S., *Hypercube Algorithms with Applications to Image Processinbg and Pattern Recognition*, Springer-Verlag, New York, 1990.

[SaS85] Saad, Y., and Schultz, M. H., "Data Communication in Hypercubes," Yale University Research Report YALEU/DCS/RR-428, October 1985 (revision of August 1987).

[SaS86] Saad, Y., and Schultz, M. H., "Data Communication in Parallel Architectures," Yale University Report, March 1986.

[SaS88] Saad, Y., and Schultz, M. H., "Topological Properties of Hypercubes," IEEE Trans. on Computers, Vol. 37, July 1988, pp. 867-872.

[Sta91] Stamoulis, G. D., "Routing and Performance Evaluation in Interconnection Networks," Ph.D. Thesis, MIT, Report LIDS-TH-2035, May 1991.

[Top85] Topkis, D. M., "Concurrent Broadcast for Information Dissemination," IEEE Trans. Software Engineering, Vol. 13, pp. 207-231, 1983.

[TuR88] Tucker, L. W., and Robertson, G. G., "Architectures and Applications of the Connection Machine," IEEE Computer, pp. 26-38, August 1988.

[VaB90a] Varvarigos, E. A., and Bertsekas, D. P., "Communication Algorithms for Isotropic Tasks in Hypercubes and Wraparound Meshes," Laboratory for Information and Decision Systems, Report LIDS-P-1972, M.I.T., Cambridge, MA, March 1990, to appear in Parallel Computing.

[VaB90b] Varvarigos, E. A., and Bertsekas, D. P., "Multinode Broadcast in Hypercubes and Rings with Randomly Distributed Length of Packets," Laboratory for Information and Decision Systems Report LIDS-P-2006, M.I.T., Cambridge, MA, November 1990, to appear in IEEE Trans. on Paral. and Distrib. Comput..

[VaB92] Varvarigos, E. A., and Bertsekas, D. P., "Dynamic Broadcasting in Parallel Computing," Laboratory for Information and Decision Systems, Report LIDS-P-2111, M.I.T., Cambridge, MA, May 1992.

[Var92] Varvarigos, E. A.,"Static and Dynamic Communication in High Speed Parallel Computing," Ph.D. Thesis, MIT, in preparation.