Dynamic Broadcasting in Parallel Computing

Emmanouel A. Varvarigos, *Member, IEEE,* and Dimitri P. Bertsekas, *Fellow, IEEE*

***Abstract*—We consider the problem where broadcast requests are dynamically generated at random time instants at each node of a multiprocessor network. In particular, in our model packets arrive at each node of a network according to a Poisson process, and each packet has to be broadcast to all the other nodes. We propose an on-line, distributed routing scheme to execute the broadcasts in this dynamic environment. Our scheme consists of repeated execution of a partial multinode broadcast task, which is a static communication task where any $M \leq N$ arbitrary nodes of an N-processor network broadcast a packet to all the other nodes. The dynamic broadcasting scheme that we propose can be used in any topology, regular or not, for which partial multinode broadcast algorithms with certain properties can be found. We derive such an algorithm and we analyze the corresponding dynamic broadcasting scheme for the hypercube network. We show that its stability region tends to the maximum possible as the number of nodes of the hypercube tends to infinity. Furthermore, for any fixed load in the stability region, the average delay is of the order of the diameter of the hypercube. Our analysis does not use any approximating assumptions.**

***Index Terms*—Dynamic broadcasting, queuing systems, average delay, stability region, hypercubes.**

I. Introduction

BROADCASTING is the operation where a packet is copied from a node to all the other nodes of a network. Because of the variety of applications that involve broadcasts, such operations are often implemented as communication primitives in parallel computers. One of the most frequent broadcasting tasks is the *multinode broadcast* (abbreviated MNB), where every node of a network broadcasts a packet to all the other nodes. The MNB arises, for example, in iterations of the form

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

where each processor i computes a component (or a block of components) x_i of the vector x. If iteration (1) takes place synchronously, and all the components of x change during each iteration, it is necessary that at the end of an iteration every processor i broadcasts the updated value of x_i to all the other processors for use at the next iteration; this is a MNB.

In iterations of the form given above it is very probable that only few of the components of the vector x will change appreciably during an iteration. If the new value of a component is close to its previous value, there is no reason to waste communication bandwidth in order to broadcast it. This motivates the study of the task, where only few, but arbitrary, processors broadcast a packet (see Fig. 1(a)). We call this generalization of the MNB task a *partial multinode broadcast* (or PMNB). Since the PMNB arises often in applications, we believe that it deserves a position among the prototype tasks of a communication library.

The MNB and the PMNB are *static* broadcasting tasks, that is, they assume that at time $t = 0$ each node broadcasts at most one packet, and all broadcasts start simultaneously. Static broadcasting tasks in multiprocessor networks have been studied extensively in the literature ([3], [4], [6], [7], [10], [14]). In this paper we consider the *dynamic* version of the broadcasting problem. We assume that packets are generated at each node according to a Poisson process with rate λ independently of the other nodes, and each packet has to be broadcast to all the other nodes (see Fig. 1(b)). We propose a dynamic scheme to execute the broadcasts in this dynamic environment, and we evaluate its performance without using approximations. The assumption of Poisson arrivals is made only because the mathematics of the analysis require it and is inessential for the implementation of the schemes that we propose. We are interested in two performance criteria. The first criterion is the average delay, that is, the average time between the arrival of a packet at a node, and the completion of its broadcast. The second criterion is the stability region of the scheme, that is, the maximum load that it can sustain with the average delay being finite. We set two objectives for a dynamic broadcasting scheme: stability for as big a load as possible, and average delay which is of the order of the diameter for any fixed load in the stability region.

The dynamic broadcasting problem is important for a variety of reasons. Consider, for example, the case where iteration (1) takes place asynchronously. Each processor i computes at its own speed without waiting for the others, and broadcasts the updated value of x_i whenever it is available. Asynchronous parallel computation is used to circumvent the synchronization penalty (see, e.g., [4]), and it naturally results in a dynamic communication environment like the one we are considering. In this environment static algorithms, such as the MNB, become inefficient, since a fast processor has to wait for all the other processors before starting a MNB. Algorithms for static communication tasks are also difficult to use, because they must be detected by the compiler, or called explicitly by the programmer. It is plausible that the programmer and the compiler may fail to identify a communication task. Even more importantly, broadcasts may be generated in *real time*, during the execution of a program. In such a situation, the communication pattern is not known in advance,

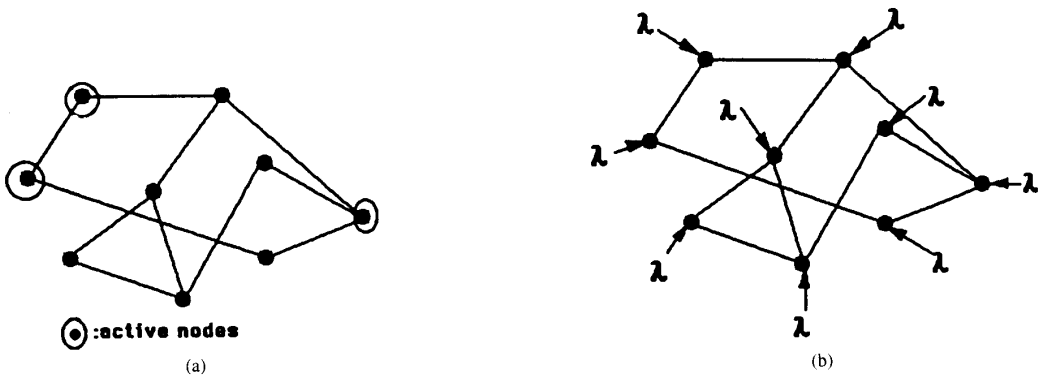
Manuscript received July 3, 1992; revised August 23, 1993. Research supported by NSF under Grant NSF-DDM-8903385, and by the ARO under Grants DAAL03-86-K-0171 and DAAL03-92-G-0309.

E. A. Varvarigos is with the University of California, Department of Electrical and Computer Engineering, Santa Barbara, CA 93106 USA.

D. P. Bertsekas is with the Massachusetts Institute of Technology, Laboratory for Information and Decision Systems, Cambridge, MA 02139 USA.

IEEE Log Number 9408132.

1045-9219/95$04.00 © 1995 IEEE



(a) (b)

Fig. 1. The PMNB and the dynamic broadcasting problems for a general network. In (a), M arbitrary nodes of the network have a packet to broadcast to all other nodes. This is the PMNB task, and it has to take place once and for all. In (b), packets arrive at each node of the network continuously according to some probabilistic rule, and each of them has to be broadcast to all other nodes. This is the dynamic broadcasting problem.

and precomputed static communication algorithms cannot be used. Multitasking and time-sharing make the communications even more unpredictable, and the use of static communication algorithms more difficult. For the preceding reasons, we believe that the dynamic broadcasting problem deserves a position among the generic problems in parallel computation. Dynamic broadcasting schemes that run continuously, and execute on-line the broadcast requests should be a part of the communication primitives of a parallel computer. The throughput and delay of a network for broadcast (one-to-many) communication are important performance criteria, in the same way that the throughput and delay for one-to-one communication are.

The only previous work on dynamic broadcasting we know of is that of Stamoulis and Tsitsiklis [12] for the hypercube network. There are two algorithms of Stamoulis and Tsitsiklis that are most interesting from a theoretical point of view: the direct algorithm, and the indirect algorithm. Both of them are conceptually different than ours in that they define and use certain spanning trees for broadcasting. In the direct algorithm, d spanning trees, where d is the dimension of the hypercube, are defined for each node. A packet that is generated at a node selects at random one of the d trees of the node and is broadcast on it. The direct algorithm meets the stability objective described above, but its average delay analysis is approximate. In the indirect algorithm, d spanning trees are defined in the hypercube. A packet that arrives at some node selects at random one of these trees. It is then sent to the root of that tree, and from there it is broadcast to all the other nodes using links of the tree. The indirect algorithm meets the delay objective, but its stability region is not the maximum possible. Therefore, the two hypercube dynamic broadcasting schemes of [12] do not provably satisfy both performance objectives.

Our dynamic broadcasting scheme has a fundamentally different philosophy: it relies heavily on finding efficient PMNB algorithms that are used as a subroutine of the dynamic scheme. Furthermore, our dynamic scheme is very general: it applies to any network for which efficient PMNB algorithms can be found, without any assumptions on the communication model used for the network. For a hypercube network, our scheme has a stability region that tends to the maximum possible as the number of nodes tends to infinity. Furthermore, its average delay for any fixed load in the stability region is of the order of the diameter. Thus, our scheme compares favorably with the hypercube algorithms in [12] none of which meets optimally the stability and the delay objective.

The dynamic broadcasting scheme consists of executing successive PMNB algorithms, each starting when the previous one has finished. Our stability and average delay results apply to any network for which we can find algorithms that execute the PMNB communication task in linear time, that is, in time

$$XM + V$$

where M is the number of nodes that have a packet to broadcast, called *active nodes*, and X, V are scalars that are independent of M (they may depend on the size of the network). For analytical purposes, our scheme is modelled after reservation and polling schemes for multiaccess communication ([1]). The network is conceptually viewed as a channel, and the nodes as users of the channel. The first V time units of the PMNB algorithm are considered as a reservation interval, and the following MX time units as a data interval, where users with reservations transmit a packet. The analogy to reservation systems is made only to analyze the performance of the scheme; no reservations actually take place, and the dynamic scheme is easy to implement in a distributed system.

For the hypercube network, we will present three different PMNB algorithms that are suitable for our purposes. The first PMNB algorithm is simple but is suboptimal. When this PMNB algorithm is incorporated in our dynamic broadcasting scheme, the latter does not provably meet the stability and average packet delay objectives that we have set. The second and third algorithms execute in linear time, are near-optimal, and have a running time bound that is roughly one half of the best existing bound for a PMNB due to Stamoulis [11]. The second algorithm assumes that packets can be split into d parts that can be routed independently. In the third algorithm the splitting of packets is not allowed, and each message is transmitted as one packet. A dynamic broadcasting scheme based on any one of the last two algorithms meets our performance objectives: its stability region tends to the maximum possible as the number of nodes tends to infinity,

and the average packet delay for any fixed load in the stability region is of the order of the diameter of the hypercube. In the companion paper [13] we present near-optimal PMNB algorithms for d-dimensional meshes that also give rise to efficient dynamic broadcasting schemes.

The structure of the paper is the following. In Section II we describe the dynamic broadcasting scheme in a given network, assuming that a PMNB algorithm with certain properties is available for that network. We also state the dynamic broadcasting theorem, which is the main result of the paper. In Section III we evaluate the performance of our dynamic broadcasting scheme. In particular, in Subsection III-A we describe an auxiliary queueing system, which we will use to prove the dynamic broadcasting theorem. In Subsection III-B we prove the dynamic broadcasting theorem, which gives an estimate on the average packet delay of the dynamic broadcasting scheme. Section IV describes three different algorithms to execute a partial multinode broadcast in a hypercube. Section V applies the dynamic broadcasting theorem to the case of the hypercube, using the results obtained in Section IV. Finally, Section VI concludes the paper.

II. Dynamic Broadcasting Schemes

In this section we will describe the dynamic broadcasting scheme for a general network. We will assume that an algorithm that executes the PMNB task in that network is given, and that it requires $XM + V$ time units, where M is the number of nodes that have a packet to broadcast, and X, V are scalars independent of M. We also assume that during the PMNB algorithm each node learns the number of active nodes M, and that the network is synchronized so that the nodes can start various phases of the algorithm simultaneously.

Our scheme is merely a repetition of successive partial multinode broadcast algorithms, each starting when the previous one has finished (see Fig. 2). The time axis is, therefore, divided into PMNB intervals. Within each PMNB interval, a PMNB is executed, involving exactly one packet from each of the M nodes that are active at the start of the interval. Each PMNB interval is divided into two parts. The first part is called *reservation interval*. Its duration can be upper bounded by a known constant V that depends only on the size of the network, and is independent of the number of active nodes M. During the reservation interval each active node s can be conceptually viewed as making a reservation for the broadcast interval (as we will see in Subsections IV-B and IV-C for the hypercube this is done automatically through the mere participation of s in the parallel prefix operation). Also, in the reservation interval some global information is gathered at the nodes (e.g., the total number of active nodes M, and other information), and some additional organizational work is performed. For example, in the PMNB algorithms described in Subsections IV-B and IV-C for the hypercube, and in the algorithms described in the companion paper [13] for the d-dimensional mesh, the packets move during the reservation interval to more favorable intermediate locations. The details of what happens in the reservation interval are in fact irrelevant, and all that matters for our purposes is that its duration is less than or equal to V. The second part of a PMNB interval is called *broadcast interval*. Its duration is equal to XM, and is therefore known once M is known (X is a scalar that may depend on the network). The broadcast interval is empty if there are no packets to broadcast ($M = 0$). Even though the duration of each PMNB 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, and the dynamic scheme is fully distributed. Note that the details of what happens in the broadcast interval are irrelevant, and all that matters for our purposes is that the duration of the broadcast interval is less than or equal to MX. Indeed our general dynamic scheme and the following theorem are valid regardless of the communication model adopted (store-and-forward, wormhole, etc).

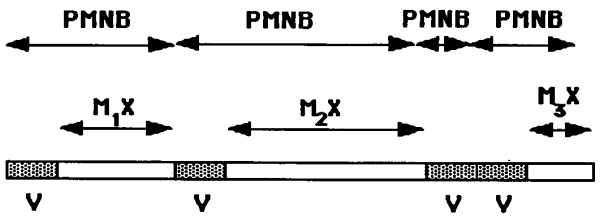


Fig. 2. The dynamic broadcasting scheme. Each PMNB period consists of two intervals: a reservation interval (marked by gray) of duration V, and a broadcast interval of duration MX, where M is the number of active nodes at the start of the PMNB period.

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

The main theorem that we prove in the paper is the following.

Dynamic Broadcasting Theorem: Assume that for a given N-processor network there exists an algorithm that executes the PMNB communication task in time

$$XM + V$$

where M is the number of nodes that have a packet to broadcast and X, V are scalars that are independent of M (they may depend on the size of the network). Assume that during the PMNB algorithm each node learns the value of M. Then the dynamic broadcasting scheme that uses this PMNB algorithm as described above has the following performance characteristics. If the packets to be broadcast arrive at each node of the network according to a Poisson process with rate λ, independently of the other nodes, the average packet delay T satisfies

$$T = W + X + aNX \leq W + X + \min\left(\frac{N-1}{2}X, \rho W\right)$$

where

$$W = \frac{\rho X}{2(1-\rho-\lambda V)} + \frac{(1-\rho)V}{2(1-\rho-\lambda V)} + \frac{(1-\rho a-\lambda V)V}{1-\rho-\lambda V}, \qquad \rho = \lambda N X$$

and a is a scalar satisfying

$$\frac{\overline{M}+(\hat{M}-1)(2\overline{M}-\hat{M})}{2N\overline{M}} - \frac{1}{2N} \le a \le \frac{1}{2} - \frac{1}{2N}$$

where

$$\overline{M} = \frac{\lambda N V}{1-\rho},$$

and $\hat{M}$ is the smallest integer which is strictly larger than $\overline{M}$.

Note that the dynamic broadcasting scheme is stable for $\rho \le 1-\lambda V$, or by using the relation $\lambda = \rho/(NX)$,

$$\rho \le \frac{1}{1+V/(NX)}.$$

The dynamic broadcasting theorem is proved in the following section.

III. Analysis of the Dynamic Broadcasting Scheme

In this section we will prove the dynamic broadcasting theorem. We first describe an auxiliary queueing system that will be used in the main proof given in Subsection III-B.

A. Limited Service Gated Reservation System with Shared Reservation and Data Intervals

In this subsection we describe an auxiliary queueing system, called *limited service gated reservation system with shared reservation and data intervals*, which is a reservation system for multiaccess communication. We are interested in the average delay required to serve a packet in this system. This average delay will be used in the next subsection to evaluate the average packet delay of the dynamic broadcasting scheme. The analysis of the auxiliary queueing system is based on unpublished research by D. P. Bertsekas and R. G. Gallager [2]. We will give this analysis in detail, since it is important for our purposes and it does not appear anywhere else.

The auxiliary queueing system is defined as follows. Consider N traffic streams, each corresponding to a different user, which share a common channel. The channel is used both for packet transmissions and reservations. In particular, the time axis is divided into data intervals, where actual data is transmitted, and reservation intervals used for scheduling future data. Each user has a separate queue, and the queues are served in cyclical order. Users make reservations during the same reservation interval, and transmit at most one packet each in the subsequent data interval. A packet can be transmitted in a data interval only if it arrived before the beginning of the previous reservation interval. For this system the following theorem holds.

Theorem 1: Let the arrival processes of packets at the users of the system be independent Poisson processes, each with rate λ. Let also $\overline{X}, \overline{X^2}$ be the first and second moments of the packet transmission times and $\overline{V}, \overline{V^2}$ be the first and second moments of the duration of a reservation interval. Then the mean waiting time in queue for this system is

$$W = \frac{\lambda N \overline{X^2}}{2(1-\rho-\lambda\overline{V})} + \frac{(1-\rho)\overline{V^2}}{2(1-\rho-\lambda\overline{V})\overline{V}} + \frac{(1-\rho a-\lambda\overline{V})\overline{V}}{1-\rho-\lambda\overline{V}} \tag{2}$$

where $\rho = \lambda N \overline{X}$ is the utilization factor, and a satisfies

$$\frac{\overline{K}+(\hat{K}-1)(2\overline{K}-\hat{K})}{2N\overline{K}} - \frac{1}{2N} \le a \le \frac{1}{2} - \frac{1}{2N}$$

where

$$\overline{K} = \frac{\lambda N \overline{V}}{1-\rho}$$

is the average number of packets per data interval, and $\hat{K}$ is the smallest integer which is strictly larger than $\overline{K}$.

Proof: Consider the ith packet arrival into the system and suppose that the user associated with packet i is user j. This packet must wait in queue for the residual time R_i until the end of the current packet transmission or reservation interval. It must also wait for the transmission of the N_i packets that must be transmitted before packet i. Finally the packet must wait for the duration of reservation intervals. Thus, the expected waiting time of packet i is

$$E(W_i) = E(R_i) + E(N_i)\overline{X} + E(Y_i)$$

where Y_i is the duration of all the whole reservation intervals during which packet i must wait before being transmitted. The expected waiting time is therefore

$$W = \lim_{i\to\infty} E(R_i) + E(N_i)\overline{X} + E(Y_i). \tag{3}$$

From Little's law we get

$$E(N_i) = \lambda N W. \tag{4}$$

Let Q_i be the number of packets in the queue of user j found by packet i upon arrival, and m_i be the number (0 or 1) of packets of user j that will start transmission between the time of arrival of packet i and the end of the frame at which packet i arrives. Then

$$E(Y_i) = (1 + E(Q_i) - E(m_i))\overline{V}. \tag{5}$$

From Little' law we have

$$Q = \lim_{i\to\infty} E(Q_i) = \lambda W. \tag{6}$$

The mean residual time $R = \lim_{i\to\infty} R_i$ can be calculated as in [1] to be

$$R = \frac{\lambda N \overline{X^2}}{2} + \frac{(1-\rho)\overline{V^2}}{2\overline{V}}. \tag{7}$$

Combining (3)–(7) we get

$$W = \lambda N W \overline{X} + \Big(1 + \lambda W - \lim_{i\to\infty} E(m_i)\Big)\overline{V} + \frac{\lambda N \overline{X^2}}{2} + \frac{(1-\rho)\overline{V^2}}{2\overline{V}}. \tag{8}$$

To find W, it remains to calculate $\lim_{i\to\infty} E(m_i)$.

There are two possibilities regarding the time of arrival of packet i.

a) packet i arrives during a reservation interval. This event, call it A, has steady state probability $1-\rho$:

$$P(A) = 1 - \rho.$$

Since the ratio of the average data interval length to the average reservation interval length is $\rho/(1-\rho)$, the average steady state length of data interval is $\rho\overline{V}/(1-\rho)$. Therefore, the average steady state number of packets per user in a data interval is

$$\frac{\rho\overline{V}}{(1-\rho)N\overline{X}} = \frac{\lambda\overline{V}}{1-\rho}.$$

This also equals the steady state value of $E(m_i|A)$ in view of the symmetry with respect to the users:

$$\lim_{i\to\infty} E(m_i|A) = \frac{\lambda\overline{V}}{1-\rho}.$$

b) Packet i arrives during a data interval. This event, call it B, has steady state probability ρ:

$$P(B) = \rho.$$

Denote

$$a = \lim_{i\to\infty} E(m_i|B),$$

$$a_k = \lim_{i\to\infty} E(m_i|B, \text{ the data interval of arrival of packet } i \text{ contains } k \text{ packets}).$$

Assuming $k > 0$ packets are contained in the data interval of arrival, there is equal probability $1/k$ of arrival during the transmission of any of these packets. Therefore

$$a_k = \sum_{n=1}^{k} \frac{1}{k}\frac{k-n}{N} = \frac{k-1}{2N}.$$

Let $P(k)$ be the unconditional steady-state probability that a data interval contains k packets, and $E(k)$ and $E(k^2)$ be the corresponding first two moments. Then we have by Bayes' rule

$$\lim_{i\to\infty} P(\text{The data interval of arrival of packet } i \text{ contains } k \text{ packets}) = \frac{kP(k)}{E(k)}.$$

Combining the preceding equations we have

$$a = \sum_{k=1}^{N} \frac{kP(k)}{E(k)} a_k = \sum_{k=1}^{N} \frac{P(k)k(k-1)}{2E(k)N} = \frac{E(k^2)}{2NE(k)} - \frac{1}{2N}.$$

We have already shown as part of the analysis of case a) above that

$$E(k) = \frac{\lambda N \overline{V}}{1-\rho} \tag{9}$$

so there remains to estimate $E(k^2)$. We have

$$E(k^2) = \sum_{k=1}^{N} k^2 P(k).$$

If we maximize the quantity above over the distribution $P(k), k = 0, 1, \cdots, N$, subject to the constraints $\Sigma_{k=0}^{N} P(k) = 1$, $\Sigma_{k=0}^{N} kP(k) = E(k)$, and $P(k) \geq 0$ (a linear programming problem) we find that the maximum is obtained for $P(N) = E(k)/N, P(0) = 1 - E(k)/N$, and $P(k) = 0, k = 1, 2, \cdots, N-1$. Therefore

$$E(k^2) \leq NE(k).$$

Similarly if we minimize $E(k^2)$ subject to the same constraints we find that the minimum is obtained for $P(\hat{k}-1) = \hat{k} - E(k), P(\hat{k}) = 1 - (\hat{k} - E(k))$ and $P(k) = 0$ for $k \neq \hat{k}-1, \hat{k}$, where $\hat{k}$ is the integer for which $\hat{k} - 1 \leq E(k) < \hat{k}$. Therefore

$$E(k^2) \geq (\hat{k}-1)^2(\hat{k} - E(k)) + (\hat{k})^2(1 - (\hat{k} - E(k))).$$

After some calculations this relation can also be written

$$E(k^2) \geq E(k) + (\hat{k}-1)(2E(k) - \hat{k}) \quad \text{for } E(k) \in [\hat{k}-1, \hat{k}), \quad \hat{k} = 1, 2, \cdots, N.$$

Note that the lower bound above is a piecewise linear function of $E(k)$, and equals $(E(k))^2$ at the breakpoints $\hat{k} = 1, 2, \cdots, N$. Summarizing the bounds we have

$$\frac{E(k) + (\hat{k}-1)(2E(k) - \hat{k})}{2NE(k)} - \frac{1}{2N} \leq a \leq \frac{1}{2} - \frac{1}{2N} \tag{10}$$

where $\hat{k}$ is the positive integer for which

$$\hat{k} - 1 \leq E(k) < \hat{k}.$$

Note that as $E(k)$ approaches its maximum value N (i.e., the system is heavily loaded), the upper and lower bounds coincide. By combining the results for cases a) and b) above we have

$$\lim_{i\to\infty} E(m_i) = P(A) \lim_{i\to\infty} E(m_i|A) + P(B) \lim_{i\to\infty} E(m_i|B) = (1-\rho)\frac{\lambda\overline{V}}{1-\rho} + a\rho$$

or finally

$$\lim_{i\to\infty} E(m_i) = \lambda\overline{V} + a\rho$$

where a satisfies (10). Using (8) and the expressions derived we obtain (2), where $E(K)$ is given by (9), and $\hat{k}$ satisfies $\hat{k} - 1 \leq E(k) < \hat{k}$. **Q.E.D.**

Note that the formula for the mean waiting time becomes exact in the limit both as $\rho \to 0$ (light load), and as $\rho \to 1 - \lambda\overline{V}$ (heavy load), in which case $E(k) \to N$ and $a \to 1/2 - 1/(2N)$. The formula is also exact if $N = 1$ ([1]).

B. Main Proof

We now complete the proof of the dynamic broadcasting theorem.

Proof of the Dynamic Broadcasting Theorem: In a PMNB period each node that has a packet to broadcast at the start of the period participates with exactly one packet. Let M be the number of such nodes at the start of a period. Each of these nodes can be viewed as making a reservation during the reservation interval. The duration of the subsequent broadcast interval is at most MX time units.

The dynamic broadcasting scheme will be called system "b" (for "broadcast"). We also consider the limited service gated reservation system with shared reservation and data intervals presented in the previous subsection. This system will be called system "a" (for "auxiliary"). Let the reservation interval of system "a" be constant and equal to V, and the service time of a packet be constant again and equal to X.

Consider the following analogy between systems "a" and "b." Let a data interval of system "a" correspond to a broadcast interval of system "b," a user of system "a" correspond to a node of system "b," and a packet arrival of system "a" correspond to a broadcast request arrival of system "b." Note the similarities between the two systems. During a data interval of system "a" (or broadcast interval of system "b") at most one packet (or broadcast request, respectively) from each user (or node, respectively) can be served. It is easy to see that the probability distributions of the length of the reservation intervals, the data (or broadcast) intervals, and the number of users (or nodes) served in a data interval are identical for both systems. In particular, the length of a reservation interval of both systems is equal to V by construction. The length of a broadcast interval of system "b" is equal to MX, where M is the number of active nodes. Similarly, the length of a data interval of system "a" is equal to MX, where M is the number of non-empty queues. The only difference between the two systems is that in system "b" a broadcast request completes service at the end of a PMNB period, while in system "a" packets complete service at times jX, $j = 1, 2, \cdots, M$, from the beginning of the data interval.

The waiting time W_a in queue for a packet of the auxiliary system is given from (2) of Subsection III-A, with $\overline{X}, \overline{X^2}, \overline{V}, \overline{V^2}$ replaced by X, X^2, V, V^2, respectively. The average delay (queueing plus service time) for the auxiliary system "a" is

$$T_a = W_a + X.$$

Let U_1 be the average time between the beginning of a data interval of system "a," and the time that a packet served in this data interval starts transmission (see Fig. 3). Similarly, let U_2 be the average time between the end of the transmission of a packet of system "a" and the end of the data interval in which it is served. Then it can be proved by using arguments similar to those used in the proof of Theorem 1 that

$$U_1 = U_2 = aNX \leq \frac{N-1}{2}X. \tag{11}$$

It can also be seen that

$$U_1 = U_2 \leq E(N_i)X = \lambda N W_a X = \rho W_a$$

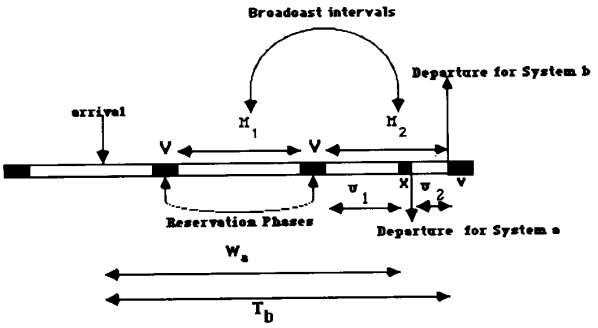


Fig. 3. Reservation and broadcast (or data) intervals for the network broadcasting scheme, and the auxiliary queueing system.

where (4) was used. The average packet delay T_b of the broacasting scheme is

$$T_b = T_a + U_2 = W_a + X + aNX \leq W_a + X + \min\left(\frac{N-1}{2}X, \rho W_a\right). \tag{12}$$

This completes the proof. **Q.E.D.**

Note that for light load the dynamic broadcasting theorem gives

$$T \leq 1.5V + X, \qquad \rho \approx 0.$$

Until now we did not have to assume any particular topology for the multiprocessor network. The dynamic broadcasting scheme and the dynamic broadcasting theorem apply to any network for which a PMNB algorithm with certain properties exists. The next section will present such algorithms for a hypercube network of processors. In [13] we present corresponding PMNB algorithms for d-dimensional meshes. We believe that such algorithms exist for many other regular topologies (in later publications we will present such PMNB algorithms for folded-cubes and Manhattan Street networks).

IV. Partial Multinode Broadcast in a Hypercube

Beginning with this section we focus on a hypercube network of processors. In particular, we consider the partial multinode broadcast problem, where M arbitrary nodes of an N-processor hypercube have to broadcast a packet to all the other nodes. We call these nodes *active nodes*. The PMNB is, as we saw, an important component of the dynamic broadcasting scheme, but it is also an important problem on its own right.

We will say that a communication 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.

Let T_{PMNB} be the optimal time required for a partial multinode broadcast in a hypercube. T_{PMNB} may depend on the identity of the M nodes that want to broadcast. A lower bound, however, is always

$$T_{\text{PMNB}} \geq \frac{M-1}{d} \tag{13}$$

where d is the dimension of the hypercube. This can be seen by arguing that each node has to receive $M-1$ or M packets, and has only d input ports. If the splitting of packets is not allowed, then a slightly stronger lower bound holds:

$$T_{\text{PMNB}} \geq \max\left(d, \left\lceil \frac{M-1}{d} \right\rceil\right)$$

where by $\lceil x \rceil$ we denote the smallest integer which is greater than or equal to x. This is because when packets are not split, the diameter of the network is a lower bound on the broadcast delay.

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 an d-dimensional hypercube with $N = 2^d$ nodes, when each packet requires one time unit (or slot) to be transmitted over a link, was found in [3] (see also [4]) to be $\lceil (N-1)/d \rceil$ time slots. Thus an upper bound for T_{PMNB} is

$$T_{\text{PMNB}} \leq \left\lceil \frac{N-1}{d} \right\rceil.$$

When $M \ll N$ the MNB algorithm is inefficient as the gap between the preceding inequality and the lower bound of (13) suggests. In the three subsections of this section we will provide three communication algorithms to execute the PMNB task in a hypercube. The first algorithm, presented in Subsection IV-A, has time complexity

$$T^{\text{I}}_{\text{PMNB}} \leq \left\lceil \frac{M-1}{m} \right\rceil + 2d + 2dt_p - m$$

in the case $M = 2^m$ for some integer m, where t_p is the time required for a single parallel prefix step ($t_p \leq 1$). If M is not a power of 2 then m should be replaced in the above expression by $\lceil \log M \rceil$ and M by $2^{\lceil \log M \rceil}$. This algorithm is *not* of optimal order [except if $M = \Theta(N^c)$ for some positive constant c, or if $M = \Theta(\log N)$], but it is a simple and practical algorithm, as numerical examples indicate.

The second and the third algorithms, to be described in Subsections IV-B and IV-C respectively execute the PMNB in near-optimal time. In particular the second algorithm, which we call *near-optimal PMNB algorithm*, executes the task in time

$$T^{\text{II}}_{\text{PMNB}} \leq \frac{N-1}{N}\frac{M}{d} + 2dt_p + 2 \tag{14}$$

independently of the value of M and the location of the active nodes. This is the best existing algorithm for the PMNB task, having roughly half the complexity of the PMNB algorithm given in [11]. Comparing (14) with the lower bound (13) we see that the leading terms of the right hand sides have the same coefficient. The algorithm assumes that packets can be split at the origin and recombined at the destinations. For the case where this is undesirable we modify the algorithm to achieve near-optimal completion time without the need of splitting and recombining the packets. This gives rise to a third PMNB algorithm that will be presented in Subsection IV-C. We call it *near-optimal PMNB without splitting of packets* and its time complexity can be bounded above by

$$T^{\text{III}}_{\text{PMNB}} \leq \left\lceil \frac{M}{d} \right\rceil + 2d + 4dt_p - 1$$

for any M. Note that the MNB is a special case of the PMNB with $M = N$. The latter PMNB algorithm gives rise to a very efficient MNB algorithm with complexity $\lceil N/d \rceil + 2d - 1$, which does not use the splitting of packets. This MNB algorithm has not appeared in the literature before.

The benefits of using a partial multinode broadcast instead of a full multinode broadcast algorithm can be best illustrated by a numerical example.

Example: We consider both the cases where $t_p = 1$ and $t_p = 0$. The case $t_p = 0$ corresponds to the situation where $t_p \ll 1$ (a realistic assumption for computers which have an efficient implementation of the parallel prefix operation), or to the situation where the position of the active nodes is known in advance. The case $t_p = 1$ corresponds to the situation where a single parallel prefix step takes time equal to the transmission time of a packet, which may be the case when the computer does not support the parallel prefix operation and packet transmission times are independent of packet lengths. Consider a hypercube of $N = 2^{16}$ nodes and a PMNB task involving $M = 2^{10}$ active nodes. A full MNB would take 4096 steps. Algorithm I requires 157 time units (or only 125 if $t_p = 0$). The near-optimal algorithm which allows splitting of packets requires 98 time units (or only 66 time units if $t_p = 0$). The near-optimal PMNB algorithm without splitting of packets requires 160 time units (96 if $t_p = 0$). The lower bound for the PMNB in this case is 64.

A. A Suboptimal PMNB Algorithm for the Hypercube

This algorithm consists of four phases. (There can be some pipelining between the phases, but we do not try to exploit this because the gain in completion time is small).

Let $s_1, s_2, \cdots, s_M, M \leq N$, 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.

Phase 1 (Rank Computation Phase): The rank $r_s (0 \leq r_s \leq M-1)$ of each active node s is computed. This can be done in $2d$ steps by performing a *parallel prefix operation* (see [9]) on a tree P, called *parallel prefix tree*, embedded in the hypercube. The ith leaf of the tree from the left is the ith

node of the hypercube. The operation is described in [9], pp. 37–44. 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 Phase 1 takes $2dt_p$ time units to be completed. 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 $t_p \ll 1$. Note that if the active nodes are known in advance then their ranks are also known and Phase 1 can be omitted.

Phase 2 (Packing Phase): Processor s sends its packet to processor r_s. This is known in the literature as the *packing problem* (see [9] for the case of a butterfly network), and can be done in d steps by using the following greedy algorithm; a packet during the ith step of the packing phase is transmitted over an i-dimensional link if the ith bit of its routing tag is a one, or stays at the current node otherwise. Packets in the packing phase use disjoint paths. This can be seen by noting that when the dimensions of the hypercube are travelled in an ascending order, the hypercube resembles a butterfly, and using the well-known results for the packing problem in butterflies (see, for example, [9], pp. 524–538).

At the end of the second phase the 2^m nodes with the smallest identities have a packet. In the last two phases, each of these packets will be broadcast to all the other processors. Note that the $M = 2^m$ nodes with the smallest identities form a subcube of dimension m, namely, the one obtained by fixing the $d - m$ most significant bits to 0.

Phase 3 (Subcube Single Node Broadcast Phase): Processor $r_s, 0 \leq r_s \leq 2^m - 1$, broadcasts its packet to the $d - m$ dimensional hypercube ($*^{d-m}r_s$) obtained by fixing the m less significant bits to equal the binary representation of r_s. This is a single node broadcast in a $(d - m)$-dimensional hypercube, and requires $d - m$ steps.

Phase 4 (Subcube Multinode Broadcast Phase): At the beginning of this phase all processors $wr_s, w = 0, 1, \cdots, 2^{d-m}$, have received the packet originating at node s. During Phase 4 processor wr_s broadcasts the packet to all the nodes in the subcube ($w*^m$). This is a full MNB in each one of these m-dimensional disjoint subcubes, and requires time $\lceil (M-1)/m \rceil$ (see [4], Section I-C).

Adding up the durations of Phases 1 through 4 we get

$$T^{\mathrm{I}}_{\mathrm{PMNB}} \leq \left\lceil \frac{M-1}{m} \right\rceil + 2d + 2dt_p - m.$$

Fig. 4 shows how the preceding algorithm works for a 4-dimensional hypercube and $M = 4$ active nodes.

B. A Near-Optimal PMNB Algorithm with Splitting of Packets

In this subsection we present a near-optimal algorithm to execute the partial multinode broadcast task in a hypercube. We will show that the time required by the algorithm satisfies

$$T^{\mathrm{II}}_{\mathrm{PMNB}} \leq \frac{N-1}{N}\frac{M}{d} + 2dt_p + 2$$

where t_p is the time required for a single parallel prefix step.

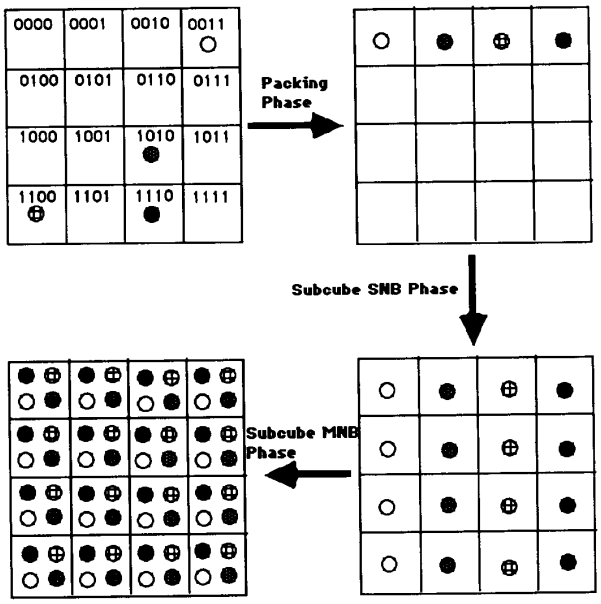


Fig. 4. The first (suboptimal) PMNB algorithm for hypercubes ($N = 16$, $M = 4$). Each column or row corresponds to a subcube.

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 in order to be transmitted 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. In the next subsection, we will present another near-optimal partial multinode broadcast algorithm, which does not require the splitting of packets.

We will start by presenting a suboptimal partial multinode broadcast algorithm. This algorithm will not make full use of the links of a hypercube. 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): The rank r_s of each active node is computed. This computation is done through a parallel prefix operation as in Phase 1 of the algorithm of the previous subsection. It requires time $2dt_p$, where t_p is the time required for a single parallel prefix step.

Phase 2 (Packing Phase): The packet of node s and rank r_s is sent to processor r_s. This is done in d steps as described in Phase 2 of the algorithm of the previous subsection.

Phase 3 (Broadcast Phase): The broadcast phase consists of d subphases $l = 1, 2 \cdots, d$. During subphase l, every node $r = r_{d-1}r_{d-2}\cdots r_0$ transmits to its neighbor across the $(d-l)$th dimension in any order 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.

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 binary number of length d. We denote by $S_l(\beta) = (*^l\beta_{d-l-1}\beta_{d-l-2}\cdots\beta_0)$ the subcube of

the nodes whose $d-l$ less significant bits are equal to the $d-l$ less significant bits of β.

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

Theorem 2: For each $\beta \in \{0,1\}^d$, at the end of subphase l of Phase 3, $l = 1, 2, \cdots, d$, each node in subcube $S_l(\beta)$ 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 subcubes.

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 subcube $S_l(\beta)$ is composed of the two subcubes $S_{l-1}(\beta_{d-1}\cdots\beta_{d-l+1}0\beta_{d-l-1}\cdots\beta_0)$ and $S_{l-1}(\beta_{d-1}\cdots\beta_{d-l+1}1\beta_{d-l-1}\cdots\beta_0)$. During subphase l every node in one of these subcubes sends to its $(d-l)$-neighbor all the packets it has received during the previous subphases, together with its own packet. This combined 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 received by 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

$$\frac{N-1}{N}M + d$$

steps.

Proof: We denote by T_l the duration of subphase l, and we let $m = \lceil \log M \rceil$. At the beginning of Phase 3 only nodes $0, 1, \cdots, M-1$ have a packet. From Theorem 2 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 subcube $(*^{l-1}s_{d-l}s_{d-l-1}\cdots s_0)$. The number of these packets is equal to the cardinality of the set

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

During subphase l, node s will send these packets to its $d-l$-neighbor. Therefore, we have

$$T_l \le \max_s |\mathcal{W}_l(s)|$$

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

$$\max_s |\mathcal{W}_l(s)| \le \max_s \left\lceil \frac{M-s'}{2^{d-l+1}} \right\rceil \le \left\lceil \frac{M}{2^{d-l+1}} \right\rceil.$$

The total duration of Phase 3 satisfies

$$\text{Duration of Phase 3} = \sum_{l=1}^{d} T_l \le \sum_{l=1}^{d} \left\lceil \frac{M}{2^{d-l+1}} \right\rceil \le d + M\sum_{l=1}^{d}\frac{1}{2^l} = d + M\left(1 - \frac{1}{N}\right).$$

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 hypercube in

$$T_{\text{PMNB}} \le M\frac{N-1}{N} + 2dt_p + 2d$$

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

The PMNB algorithm that we described so far is not of optimal order as the gap between the lower bound (13), and the result of Lemma 2 indicates. In fact, it is 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 algorithm so that all dimensions are used at the same time, and near-optimal completion time is achieved.

Theorem 3: The partial multinode broadcast task in a d-dimensional hypercube can be executed in

$$T_{\text{PMNB}} = \frac{M}{d}\frac{N-1}{N} + V_h \tag{15}$$

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

$$V_h = 2dt_p + 2$$

s the time required for the first two phases.

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, in the parallel prefix or the packing phase only links of a particular dimension are used at each step. Similarly, during subphase l of the broadcast phase only links of dimension $d-l$ are used.

For any node s and integer $c, 0 \le c \le d-1$, we denote by $R^c(s)$ the binary number obtained by shifting s by c positions to the right. We also define the order $<_c$ between binary numbers as

$$s <_c t \text{ if and only if } R^c(s) < R^c(t),$$

and the *rank of class* c of an active node s as

$$r_s^c = \sum_{\{t: t <_c s\}} x_t - 1 \tag{16}$$

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

For any c, consider now another PMNB algorithm referred to as algorithm $\mathcal{A}_c$. Algorithm $\mathcal{A}_c$ is similar to algorithm $\mathcal{A}_0$, and consists of a parallel prefix, a packing, and a broadcast phase. In the parallel prefix phase the rank r_s^c is calculated for each node by using an appropriate parallel prefix tree; this tree is obtained from the parallel prefix tree used in algorithm $\mathcal{A}_0$, by replacing each node w by node $R^c(w)$, and each link of dimension l by a link of dimension $(l+c) \bmod d$. In the packing phase, the packet of node s is sent to the intermediate node r_s^c in the following way. During the ith step of the packing phase the packet is transmitted over a link of dimension $i + c \bmod d$ if the corresponding bit of its routing tag is a one, or stays at the current node otherwise. In the broadcast phase of algorithm $\mathcal{A}_c$ a packet is transmitted over a link of dimension $(l+c) \bmod d$ of its current location,

whenever the same packet would be transmitted under the $\mathcal{A}_0$ algorithm over a link of dimension l of its current location. Since $\mathcal{A}_c$ is identical to $\mathcal{A}_0$ after appropriately renaming the hypercube 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 $\mathcal{A}_0$.

Using simultaneously all the algorithms $\mathcal{A}_0, \mathcal{A}_1, \cdots, \mathcal{A}_{d-1}$ we can find a new algorithm that 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 hypercube dimensions at any time. According to our communication model, a mini packet requires $1/d$ time units for transmission over a link. The duration of the packing and the broadcast phase is thus reduced by a factor of d, while the duration of the parallel prefix phase remains the same. Therefore, the theorem follows from Lemma 2. Q.E.D.

The scalar V_h in (15) is growing linearly with the dimension d. In practice, however, $2dt_p$ 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 ([5]). Theoretically, however, the parallel prefix operation takes time proportional to the diameter.

No upper ceilings are needed in (15), since we allow fragmented slots. Note also that under the communication model used in this section a single multinode broadcast requires 2 time units, the same time that would be required if cut-through routing ([8]) was used. A near-optimal PMNB algorithm that does not require the splitting of packets is presented in the next subsection.

C. A Near-Optimal Hypercube PMNB Algorithm without Splitting of Packets

In this subsection we modify the algorithm of the preceding section 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 consists of two parts.

1) Class Computation Part: The rank $r_s, 0 \leq r_s \leq M-1, s \in \{s_1, s_2, \cdots, s_M\}$, of each packet is computed through a parallel prefix operation. This requires $2dt_p$ time units. The packet of node s is assigned a *class number* $c = r_s \bmod d$.

2) Main Part: The packets of class c are routed according to algorithm $\mathcal{A}_c$ described in the proof of Theorem 3. Recall that algorithm $\mathcal{A}_c$ consists of three phases: the rank computation phase, the packing phase, and the broadcast phase. Only packets of class c take part in the rank computation phase, or in any other phase of $\mathcal{A}_c$. For example, when computing the rank r_s^c from (16), only the active nodes of class c set $x_t = 1$.

Each class has at most $\lceil M/d \rceil$ packets. Using Lemma 2 with $\lceil M/d \rceil$ instead of M, we get that the main part of the previous algorithm requires time less than $\lceil M/d \rceil + 2d + 2dt_p - 1$ (we have taken into account that the duration of the main part, excluding the parallel prefix operation, is an integer). Thus the total duration of the algorithm satisfies

$$T_{\text{PMNB}}^{\text{III}} \leq \left\lceil \frac{M}{d} \right\rceil + 2d + 4dt_p - 1.$$

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

$$\left\lceil \frac{N}{d} \right\rceil + 2d - 1$$

which is optimal within $2d-1$ time units. This MNB algorithm is apparently new.

V. Performance of the Dynamic Broadcasting Scheme for Hypercubes

The PMNB algorithms of Subsections IV-B and IV-C satisfy the conditions of the dynamic broadcasting theorem, and, therefore, any of them can be used as a component of the dynamic broadcasting scheme. In this section we will evaluate the average delay and the stability region of the corresponding hypercube dynamic schemes.

If we use the PMNB algorithm of Subsection IV-B as the basis for the dynamic scheme, we can view the rank computation and the packing phases as a reservation interval of duration $V = 2dt_p + 2$. If the PMNB algorithm of Subsection 4.3 that does not allow the splitting of packets is used, then the reservation interval consists of the class computation, the rank computation, and the packing phases, and it has duration $V = 2d + 4dt_p$. The analysis can be carried out for both cases. In what follows we will assume that the PMNB algorithm of Subsection IV-B is used; if the algorithm of Subsection IV-C is used, V should be replaced by $2d + 4dt_p$ throughout this section.

During the reservation interval every node s that has a packet to broadcast sets $x_s = 1$ in the parallel prefix phase of Subsection IV-B (or in the class computation phase if the algorithm of Subsection IV-C is used). In addition to that, during the packing phase, the packets move to more favorable intermediate destinations. Following each reservation interval, there is a broadcast interval, which is the last phase of the PMNB algorithm, and has duration

$$\frac{M}{d}\frac{N-1}{N}$$

time units. Each node can participate with at most one packet in a broadcast interval, and this packet must have arrived prior to the beginning of the PMNB period.

We define the *hypercube utilization factor* as

$$\rho = \frac{\lambda(N-1)}{d}. \tag{17}$$

For a given load, ρ is equal to the ratio of the average total 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 hypercube. To find a necessary condition for stability for any broadcasting scheme, note that λN broadcast requests are generated on the average per unit of time in the hypercube, each requiring at least $N-1$ transmissions to be completed. Since there are dN links, a necessary condition for stability is

$$\lambda N(N-1) \le dN$$

or equivalently

$$\rho < 1.$$

As we will see, our scheme does not guarantee stability for any $\rho < 1$, but it does for ρ very close to 1.

The PMNB algorithm of Subsection IV-B satisfies the conditions required by the dynamic broadcasting theorem to apply. Using the dynamic broadcasting theorem, and substituting X by $(N-1)/(dN)$, we see that the average delay T_h of the corresponding hypercube dynamic broadcasting scheme satisfies

$$T_h = W + \frac{a(N-1)}{d} + \frac{N-1}{N}\frac{1}{d} \le W + \frac{1}{d} + \min\left(\rho W, \frac{N-1}{2d}\right) \tag{18}$$

with

$$W = \frac{\rho}{2d(1-\rho-\lambda V)}\frac{N-1}{N} + \frac{(1-\rho)V}{2(1-\rho-\lambda V)} + \frac{(1-\rho a-\lambda V)V}{1-\rho-\lambda V} = \frac{N-1}{N}\frac{\rho}{2d(1-\rho-\lambda V)} + \frac{V(1.5-0.5\rho-\rho a-\lambda V)}{1-\rho-\lambda V} \tag{19}$$

where

$$V = 2dt_p + 2$$

ρ is given by (17), and

$$\frac{\overline{M} + (\hat{M}-1)(2\overline{M}-\hat{M})}{2N\overline{M}} - \frac{1}{2N} \le a \le \frac{1}{2} - \frac{1}{2N},$$

$$\overline{M} = \frac{\rho V d}{1-\rho}$$

where $\hat{M}$ is the smallest integer which is larger than $\overline{M}$. ($\overline{M}$ is the average number of broadcast requests served in a PMNB period).

For any fixed load which satisfies $1-\rho-\lambda V > 0$, we obtain from (18) and (19) that the average packet delay is $\Theta(V) = \Theta(d \cdot t_p)$. In particular, for an almost empty hypercube, $\rho \approx 0$, we have

$$T_h \le 1.5V + \frac{1}{d} = 3dt_p + 3 + \frac{1}{d}, \qquad \text{for } \rho \approx 0.$$

The following theorem gives the order of magnitude of the average delay of the scheme at heavy load.

Theorem 4: In the limit, as $\rho \to 1 - \lambda V$, we have

$$T_h = O\left(\frac{V}{1-\rho-\lambda V}\right).$$

Proof: Let $\rho = 1 - \lambda V - \epsilon$ with $\epsilon \to 0$. Then, as shown in Subsection III-A, $a \to 0.5 - 0.5/N$. Let $a = 0.5 - 0.5/N - \delta$ with $\delta \to 0$. Then, using (18) and (19),

$$T_h \le \frac{\rho}{2d\epsilon} + \frac{V(0.5+\epsilon+0.5\rho/N+\rho\delta)}{\epsilon} + \frac{1}{d} + \min\left(\frac{V(0.5+\epsilon+0.5\rho/N+\rho\delta)}{\epsilon}, \frac{N-1}{2d}\right) = O\left(\frac{V}{\epsilon} + \frac{\rho}{d\epsilon}\right). \qquad \textbf{Q.E.D}$$

As proved in [11], the average delay of any dynamic broadcasting scheme grows at least as $\Omega((1-\rho)^{-1})$. Since the term λV, which by (17) is equal to $\rho dV/(N-1)$, goes to 0 (very fast) as $N \to \infty$, our dynamic broadcasting scheme has good behavior for heavy load (ρ close to 1) when the number of nodes of the hypercube is large.

The hypercube broadcasting scheme is stable for $\rho < 1 - \lambda V$, or using the equations $\lambda = \rho d/(N-1)$ and $V = 2dt_p + 2$,

$$\rho \le \frac{1}{1 + (2dt_p+2)d/(N-1)},$$

which is very close to one (the maximum ρ that can be accomodated by *any* scheme), and tends to one as $N \to \infty$. The reason we get this remarkable result is the efficiency of the PMNB algorithm of Subsection IV-B, and the small overhead introduced by the reservation intervals (parallel prefix and packing phases).

Note that similar results can be obtained for the case where the PMNB algorithm of Subsection IV-C (where packets are not split) is used as the basic component of the dynamic broadcasting scheme. Then, it can be shown that the stability region is given by

$$\rho \le \frac{1}{1 + (2d + 4dt_p)d/N}, \quad \text{(packets are not split)},$$

while the average delay for ligth load satisfies

$$T_h \le 6dt_p + 3d + \frac{1}{d}, \qquad \text{for } \rho \approx 0, \quad \text{(packets are not split)}.$$

VI. Conclusions

We have provided a general method for dynamic broadcasting, which consists of successive (static) PMNB algorithm executions. The method is valid for any network and can use any PMNB algorithm for that network. If, however, the PMNB algorithm has execution time that is linear in the number of nodes participating in the PMNB, our dynamic broadcasting theorem gives sharp stability and average delay results.

These results have been specialized to the hypercube, showing that as the size of the hypercube increases, the region of stability of our dynamic scheme approaches the maximum possible, while the average delay grows at the optimal rate. Similar results have been shown elsewhere for mesh topologies [13]. It is expected that appropriate PMNB algorithms can be developed for other topologies of interest, and that a similar stability and average delay analysis of our dynamic broadcasting scheme can be obtained by using our general theorem.

References

[1] D. P. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[2] ——, unpublished research.

[3] D. P. Bertsekas, C. Ozveren, G. D. Stamoulis, P. Tseng, and J. N. Tsitsiklis, "Optimal communication algorithms for hypercubes," *J. Parallel Distrib. Comput.*, vol. 11, pp. 263–275, 1991.

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

[5] G. E. Blelloch, "Scans as primitive parallel operations," *Proc. Int. Conf. Parallel Processing*, pp. 355–362, Aug. 1986.

[6] C. T. Ho, "Full bandwidth communications on folded hypercubes," IBM Almaden Research Center, Res. Rep. RJ 7434 (69605), Apr. 1990.

[7] S. L. Johnsson and C. T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Comput.*, vol. C-38, pp. 1249–1268, 1989.

[8] P. Kermaniand and L. Kleinrock, "Virtual cut-through: A new computer communicating switching technique," *Comput. Networks*, vol. 3, pp. 267–286, 1979.

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

[10] Y. Lan, A.-H. Esfahanian, and L. Ni, "Multicast in hypercube multiprocessors," *J. Parallel Distrib. Comput.*, pp. 30–41, 1990.

[11] G. D. Stamoulis, "Routing and performance evaluation in interconnection networks," Ph.D. thesis, Rep. LIDS-TH-2035, Massachusetts Inst. of Tech., May 1991.

[12] G. D. Stamoulis and J. N. Tsitsiklis, "Efficient fouting schemes for multiple broadcasts in hypercubes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, pp. 725–739, 1993.

[13] E. A. Varvarigos and D. P. Bertsekas, "Partial multinode broadcast and partial exchange in d-dimensional wraparound meshes," to appear in *J. Parallel Distrib. Comput.*

[14] ——, "Multinode broadcast in hypercubes and rings with randomly distributed length of packets," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, pp. 144–154, 1993.

Emmanouel (Manos) Varvarigos (M'93) was born in Athens, Greece, in 1965. He received a Diploma (1988) in electrical engineering from the National Technical University of Athens, Greece, and the M.S. (1990), Engineer (1991), and Ph.D. (1992) degrees in electrical engineering and computer science from the Massachusetts Institute of Technology.

In 1990 he worked on optical fiber communications at Bell Communications Research, Morristown. He is currently an Assistant Professor of Electrical and Computer Engineering at the University of California, Santa Barbara. His research interests are in the areas of parallel and distributed computation, optical fiber data networks, and mobile communications.

Dr. Varvarigos received the First Panhellenic Prize in the Greek Mathematic Olympiad in 1982, and four times (1984–1988) the Technical Chamber of Greece award. He is a member of the Technical Chamber of Greece.

Dimitri P. Bertsekas (F'83) received a combined B.S.E.E. and B.S.M.E. degree from the National Technical University of Athens, Greece in 1965, the M.S.E.E. degree from George Washington University in 1969, and the Ph.D. degree in system science from the Massachusetts Institute of Technology in 1971.

He has held faculty positions with the Engineering-Economic Systems Dept., Stanford University (1971–1974) and the Electrical Engineering Dept. of the University of Illinois, Urbana (1974–1979). He is currently Professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology. He consults regularly with private industry and has held editorial positions in several journals. He has done research in the areas of estimation and control of stochastic systems, linear, nonlinear and dynamic programming, data communication networks, and parallel and distributed computation, and has written numerous papers in each of these areas. He is the author of *Dynamic Programming and Stochastic Control*, Academic Press, 1976, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, 1982, *Dynamic Programming: Deterministic and Stochastic Models*, Prentice-Hall, 1987, *Linear Network Optimization: Alogorithms and Codes*, M.I.T. Press, 1991; and co-author of *Stochastic Optimal Control: The Discrete-Time Case*, Academic Press, 1978, *Data Networks*, 1987, and *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, 1989.