The Priority Broadcast Scheme for Dynamic Broadcast in Hypercubes and Related Networks

Chi-Hsiang Yeh, Emmanouel A. Varvarigos, and Hua Lee
Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106-9560, USA

Abstract

Dynamic broadcast is a communication problem where each node in a parallel computer generates packets to be broadcast to all the other nodes according to a certain random process. The lower bound on the average time required by any oblivious dynamic broadcast algorithm in an n-dimensional hypercube is $\Omega(n+\frac{1}{1-\rho})$ *when packets are generated according to a Poisson process, where* ρ *is the load factor. The best previous algorithms, however, only achieve* $\Omega(\frac{n}{1-\rho})$ *time, which is suboptimal by a factor of* $\Theta(n)$*. In this paper, we propose the* priority broadcast scheme *for designing dynamic broadcast algorithms that require optimal* $O(n+\frac{1}{1-\rho})$ *time in an n-dimensional hypercube. We apply the routing scheme to other network topologies, including k-ary n-cubes, meshes, tori, star graphs, generalized hypercubes, as well as any symmetric network, for efficient dynamic broadcast. In particular, the algorithms for star graphs, generalized hypercubes, and k-ary n-cubes with* $k = O(1)$ *are also asymptotically optimal. We also propose a method for assigning priority classes to packets, called* optimal priority assignment*, which achieves the best possible performance for dynamic multiple broadcast in any network topology.*

1. Introduction

Hypercubes and k-ary n-cubes are among the most popular network topologies for parallel processing. Many commercial and experimental parallel computers are built based on these networks. Star graphs [2, 3] and generalized hypercubes [6] are also important networks that are receiving increasing attention recently. Numerous algorithms have been proposed for these networks [5, 8, 10, 13, 15, 18, 19].

Among the properties and algorithms investigated for these networks, dynamic broadcast is a communication problem where each node in a parallel computer generates packets to be broadcast to all the other nodes according to a certain random process. A necessary condition for the stability of dynamic broadcast in an n-dimensional hypercube is that the *load factor* (or called *throughput factor*)

$$\rho \stackrel{def}{=} \lambda \frac{2^n - 1}{n} < 1,$$

when the packets to be broadcast are generated according to a Poisson process with rate λ [14]. The lower bounds on the average broadcast delay and average reception delay required by any oblivious dynamic broadcast algorithm are $\Omega(n+\frac{1}{1-\rho})$ when the packets to be broadcast are generated according to a Poisson process [14]. Stamoulis and Tsitsiklis [14] proposed a *direct scheme* based on n completely unbalanced spanning trees and an *indirect scheme* based on n edge-disjoint spanning trees for dynamic broadcast in hypercubes. Their direct scheme is stable when $\rho < 1$ and requires $O(\frac{n}{1-\rho})$ average broadcast delay and reception delay; their indirect scheme is stable only when $\rho < \frac{2}{3}$ and requires $O(\frac{n}{2/3-\rho})$ average broadcast delay and reception delay. Varvarigos and Bertsekas [20] proposed a *dynamic broadcasting scheme* based on partial multinode broadcast (PMNB) for dynamic broadcast in hypercubes. Varvarigos and Banerjee [21] also proposed a *direct broadcasting scheme* and an *indirect broadcasting scheme* for dynamic broadcast in arbitrary network topologies. None of these algorithms can achieve optimal performance when the load factor is large.

In this paper, we propose the *priority broadcast scheme* for dynamic broadcast in an n-dimensional hypercube that requires optimal $O(n+\frac{1}{1-\rho})$ average reception delay. Our dynamic broadcast algorithm for hypercubes is optimal within a factor approximately equal to 1 when the load factor is close to 0 and is optimal within a small constant factor for any other load factor. Our dynamic broadcast algorithms are very easy to implement in parallel computers and are considerably faster than the best previous algorithms for networks investigated in this paper. In particular, our algorithms for n-dimensional hypercubes improve on the broadcast time required by the direct scheme proposed in [14] by a factor of

$\Theta(n)$ when the load factor is large, and have similar performance when the load factor is close to 0.

We apply the priority broadcast scheme to other network topologies, including k-ary n-cubes, meshes, tori, star graphs, generalized hypercubes, as well as any symmetric network, for efficient dynamic broadcast. In particular, the algorithms for $k_1 \times k_2 \times \cdots \times k_n$ meshes or tori with $k_i = O(1)$, k-ary n-cubes with $k = O(1)$, star graphs, and generalized hypercubes are asymptotically optimal. We also generalize the priority broadcast scheme to product networks [7, 27]. The proposed broadcast scheme can also be applied to a variety of other network topologies for dynamic broadcast with high performance. We propose an efficient method for assigning priority classes to packets, called the *optimal priority assignment* method, which achieves the best possible performance for dynamic multiple broadcast in any network topology.

In Section 2, we introduce a simple dynamic broadcast algorithm for hypercubes, illustrate the central idea of the priority broadcast scheme, and provide an analysis for the broadcast time. In Section 3, we present a simple dynamic broadcast algorithm for k-ary n-cubes, meshes, and tori. In Section 4, we propose the optimal priority assignment method for arbitrary network topologies. In Section 5, we generalize the dynamic broadcast algorithms to any vertex and edge symmetric network as well as any homogeneous product network.

2. Optimal dynamic broadcast in hypercubes

In this section, we present a simple oblivious algorithm for dynamic broadcast in hypercubes, illustrate the central idea of our *priority broadcast scheme*, and analyze its performance.

2.1. A simple oblivious broadcast algorithm for hypercubes

In what follows, we describe a simple routing example of the priority broadcast scheme we propose. When a node generates a packet to be broadcast, it randomly selects a dimension d and then partition the hypercube into two $(n-1)$-dimensional subcubes across dimension-d links; that is, nodes in a subcube have the same value for the d^{th} bit of their addresses. We then broadcast the packet with high priority within the subcube to which the source node belongs and forward the packets from this subcube to the other subcube along dimension-d links with low priority (see Fig. 1). Note that a packet is forwarded to the other subcube as soon as the associated dimension-d link is available. Then each of the two classes of traffic is responsible for about 50% of the total traffic. Therefore, the high-priority traffic becomes a traffic load with throughput factor ρ' smaller than 0.5 when the system is stable and, as a result, delivering these high-priority packets with small delay becomes a trivial job.

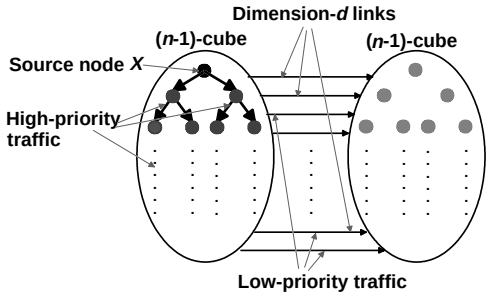

Figure 1. Dynamic broadcast in a hypercube based on the priority broadcast scheme.

A simple method for broadcasting in an n-cube under the single-port communication model can be presented as follows:

- At time 1, the source node sends the packet to be broadcast to its dimension-$(d+1 \bmod n)$ neighbor.
- At time t, $t = 2, 3, \ldots n$, each node that has a packet forwards the packet to its dimension-$(d+t \bmod n)$ neighbor.

We can easily modify this algorithm to obtain a *nonidling queueing version* for dynamic broadcast under the all-port communication model. More precisely, in this modified algorithm all the packets are sent along exactly the same path as the preceding simple algorithm, but a node forwards all its packets as soon as the associated links are available. For example, the source node will send the packet to all its n neighbors at time 1 if all its outgoing links are available. Note that there may be other broadcast or routing tasks in the network, so some links may be busy. Therefore, when an associated link is not available, the packet has to be stored in the associated output queue and waits for service. The *nonidling direct scheme* proposed in [14] uses this nonidling queueing version for dynamic broadcast.

To combine this nonidling queueing version with our priority broadcast scheme, all we have to do is assign low priority to the packets that will be forwarded over dimension-d links (i.e., the last step along a routing path), and assign high priority to the remaining packets. In the following subsections we will show that the resultant algorithm is asymptotically optimal and improves on the best previous algorithms significantly.

2.2. The central idea of the priority broadcast scheme

To intuitively illustrate the central idea of our routing scheme, we first analyze the broadcast time using a simple approximation, which assumes that the arrival processes of high-priority packets and all the packets at a node can be approximated by Poisson processes.

We assume unit service time and let ρ' be the approximate arrival rates of low-priority and high-priority packets (they are approximately the same). Since $2\rho' = \rho$ and $\rho < 1$ when the system is stable, we have $\rho' < 0.5$. Therefore, the queues for high-priority packets become slotted M/D/1 queues with arrival rate $\rho' < 0.5$ and the average waiting time for a high-priority packet can be approximated by $\frac{\rho'}{2(1-\rho')} + \frac{1}{2} < 1 = O(1)$.

According to the conservation law [11], the average waiting time in a queue will not be affected by assigning different priority classes to packets when the arrival process remains the same and the assignment of priority classes is independent of the service time of the packets. Therefore, the average waiting time for packets (including both low-priority and high-priority packets) in our broadcast scheme is equal to that of a slotted M/D/1 queue with arrival rate ρ and is equal to $\frac{1}{2(1-\rho)}$. Thus, the average waiting time for low-priority packets is smaller than $\frac{1}{(1-\rho)}$.

From the broadcast algorithm given in Subsection 2.1, we can see that a packet is forwarded as a high-priority packet for $(n-1)/2$ steps and is forwarded as a low-priority packet for $1/2$ step only on the average before it is received by a node. Therefore, an upper bound on the approximation of the average reception delay $T_p(n,\rho)$ is given by

$$T_p(n,\rho) < \frac{3n}{4} + \frac{(n-1)\rho}{4(1-\rho/2)} + \frac{\rho}{2(1-\rho)} = O\left(n + \frac{1}{1-\rho}\right),$$

assuming the arrival processes at each node can be approximated by Poisson processes. In particular,

$$T_p(n,\rho \to 0) \approx \frac{3n}{4}; \quad T_p(n,\rho \to 1) \approx \frac{5n}{4} + \frac{1}{2(1-\rho)}.$$

Note that when all network nodes are synchronized, a packet can be forwarded right after it is received and the average reception delay can be reduced.

From the above analysis, we can see that before a packet is received by a node, it was transmitted as a low-priority packet with large delay $O(\frac{1}{1-\rho})$ at most once, and was transmitted as a high-priority packet with small delay $O(1)$ at most $O(n)$ times. Therefore, the overall delay is only $O(n + \frac{1}{1-\rho})$. As a comparison, a packet in the nonidling direct scheme proposed in [14] goes through $O(n)$ queues on the average, each with large delay $O(\frac{1}{1-\rho})$ (when the load factor ρ is large), for a total of $O(\frac{n}{1-\rho})$. As a result, when the average queue lengths in our priority broadcast scheme and in the direct scheme proposed in [14] are of the same order, our priority broadcast scheme can have performance that is better by a factor of $\Theta(n)$ when traffic is heavy (by separating the factors $O(n)$ (i.e., the average distance) and $O(\frac{1}{1-\rho})$ (i.e., the average queue length) from multiplicative factors in the time required by the algorithm in [14] to the addictive factors for the average reception delay required by our algorithm). Our proposed algorithm is the only known algorithm in the literature that matches the asymptotic lower bound for any oblivious algorithm.

2.3. Analysis of the simple dynamic broadcast algorithm

The approximations presented in the previous subsection provide an intuitive explanation of the central idea of our broadcast scheme. are not Poisson processes. In this subsection, we present a more complicated and accurate estimation of the broadcast time by more accurately describing the arrival process.

In the following analysis we assume that the random processes are in steady-state. We follow the notation and the method for analysis in [14]. We let $Y_k^{(i)}$ and $Z_k^{(i)}$ be the numbers of low-priority packets and high-priority packets waiting for transmissions over the dimension-i link of a node X at the beginning of the k^{th} slot, including the packet to be transmitted. We let B_k be the number of packets generated by node X during the k^{th} slot and let $C_k^{(i)}$ be the number of packets generated by node X during the k^{th} slot that choose a dimension other than i to partition the hypercube for broadcasting (see Subsection 2.1). We also let $P_k^{(m,i)}$ and $Q_k^{(m,i)}$ be the numbers of low-priority packets and high-priority packets received by node X from its dimension-m neighbor during the k^{th} slot that have to be forwarded through its dimension-i link. Note that random variables $P_k^{(m,i)}$ and $Q_k^{(m,i)}$ are of Bernoulli type; that is, $P_k^{(m,i)}, Q_k^{(m,i)} \in \{0,1\}$. Then we have

$$Y_{k+1}^{(i)} = [Y_k^{(i)} - 1]^+ + B_k + \sum_{m=1}^{n} P_k^{(m,i)}, \text{ for } k = 1,2,\ldots,$$

$$Z_{k+1}^{(i)} = [Z_k^{(i)} - 1]^+ + C_k + \sum_{m=1}^{n} Q_k^{(m,i)}, \text{ for } k = 1,2,\ldots, \quad (1)$$

where $[\alpha]^+ = \max(\alpha, 0)$. By vertex symmetry, these random variables for all the network nodes are the same; by edge symmetry, these random variables for network links of dimension i are the same for all i.

To analyze the broadcast time required by our algorithm, we need to introduce two *approximating assumptions*.

- *Assumption A:* For any (m,i), the random variables $(P_k^{(m,i)})_{k=1,2,\ldots}$ are taken to be *independent* and identi-

cally distributed; the random variables $(Q_k^{(m,i)})_{k=1,2,\ldots}$ are also taken to be independent and identically distributed.

- *Assumption B:* The processes $(P_k^{(1,i)})_{k=1,2,\ldots}$, $(P_k^{(2,i)})_{k=1,2,\ldots}$, …, $(P_k^{(n,i)})_{k=1,2,\ldots}$, and $(Q_k^{(1,i)})_{k=1,2,\ldots}$, $(Q_k^{(2,i)})_{k=1,2,\ldots}$, …, $(Q_k^{(n,i)})_{k=1,2,\ldots}$, are taken *mutually independent*.

These assumptions are also used in [14] for the analysis of their nonidling direct scheme for dynamic broadcast. They are also of similar spirit as those used in [1, 9] for different routing problems.

Similar to the analysis given in Subsection 2.2, we need to compute the average queue length and the queue length of high-priority packets. Both of them can be obtained using the method in [14] for the analysis of their direct scheme. We let ρ be the load factor; that is, a link is busy with probability ρ. Then the average waiting time of all the packets is given by

$$W = \frac{\rho}{3(1-\rho)} + O\left(\frac{n}{2^n(1-\rho)}\right)$$

for links of any dimension as shown in [14].

In what follows, we derive the average waiting time for high-priority packets. We let ρ' be the load factor of high-priority packets; that is, a link is transmitting a high-priority packet at a given time with probability ρ'. Then the expected value of $Q_k^{(m,i)}$ is equal to

$$E[Q_k^{(m,i)}] = \rho' g_{m,i} = \frac{\rho g_{m,i}(2^{n-1}-1)}{2^n-1},$$

where $g_{m,i}$ is the probability that a packet received from a dimension-m link will be forwarded over a dimension-i link,

$$g_{m,i} = \frac{2^{m-i-1 \bmod n}-1}{2^n-1} \quad \text{for } m \neq i,$$

and $g^{i,i} = 0$. We define the random process

$$A_k^{(i)} = C_k + \sum_{m=1}^{n} Q_k^{(m,i)}. \tag{2}$$

Since $(C_k)_{k=1,2,\ldots}$ is a renewal process and is independent of $(Q_k^{(1,i)})_{k=1,2,\ldots}$, $(Q_k^{(2,i)})_{k=1,2,\ldots}$, …, $(Q_k^{(n,i)})_{k=1,2,\ldots}$, the random process $(A_k^{(i)})_{k=1,2,\ldots}$ is taken as a renewal process that assumes the distribution of a random variable $A^{(i)}$ under the approximating assumptions. Then we have

$$E[A^{(i)}] = \frac{\lambda(n-1)}{n} + \sum_{m=1}^{n} \rho' g_{m,i} = \frac{\rho}{2} \pm O\left(\frac{\rho n}{2^n}\right).$$

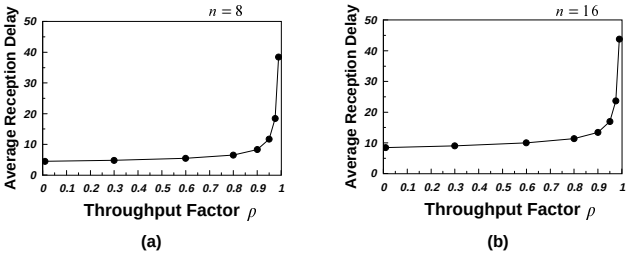

Figure 2. Performance of our priority broadcast scheme for dynamic broadcast in (a) an 8-dimensional hypercube and (b) a 16-dimensional hypercube with various load factors.

After some calculation, we can also obtain

$$\mathrm{var}[A^{(i)}] = \frac{\rho}{2} - \frac{\rho^2}{6} \pm O\left(\frac{\rho^2 n}{2^n}\right),$$

where λ is the arrival rate of new packets to be broadcast from a node and $\lambda = \frac{\rho n}{2^n-1}$.

From Eqs. (2) and (1), we have

$$Z_{k+1}^{(i)} = [Z_k^{(i)} - 1]^+ + A_k^{(i)}. \tag{3}$$

Since the arrival process $(A_k^{(i)})_{k=1,2,\ldots}$ is taken as a renewal process, it follows from Eq. (3) that $(Z_{k+1}^{(i)})_{k=1,2,\ldots}$ can be approximated by the process of the queue length of a discrete-time $G/D/1$ queue with unit service time. Since the average waiting time of high-priority packets is equal to

$$W' = \frac{\mathrm{var}[A^{(i)}]}{2E[A^{(i)}](1-E[A^{(i)}])} - \frac{1}{2},$$

we have

$$W' = \frac{\rho}{3(2-\rho)} \pm O\left(\frac{n}{2^n(1-\rho)}\right) < \frac{1}{3} + O\left(\frac{n}{2^n(1-\rho)}\right)$$

for links of any dimension. Therefore, from the conservation law [11], the average waiting time of low-priority packets is given by

$$W_L = 2W - W' = \frac{\rho(3-\rho)}{3(1-\rho)(2-\rho)} \pm O\left(\frac{n}{2^n(1-\rho)}\right);$$

$$W_L \approx \frac{2}{3(1-\rho)} \quad \text{when } \rho \to 1.$$

Since a packet is forwarded as a high-priority packet for $(n-1)/2$ steps and is forwarded as a low-priority packet for $1/2$ step on the average before it is received by a node, the average reception delay is given by

$$T(n,\rho) = \frac{(n-1)(W'+1)}{2} + \frac{W_L+1}{2} + \frac{1}{2}$$

$$= \frac{n(3-\rho)}{3(2-\rho)} + \frac{\rho}{3(1-\rho)(2-\rho)} + \frac{1}{2} \pm O\left(\frac{n^2}{2^n(1-\rho)}\right)$$
$$= O\left(n + \frac{1}{1-\rho}\right).$$

In particular, we have

$$T(n,\rho \to 0) \approx \frac{n}{2} + \frac{1}{2}; \quad T(n,\rho \to 1) \approx \frac{2n}{3} + \frac{1}{3(1-\rho)} + \frac{1}{2}.$$

Figure 2 shows the performance of the simple oblivious broadcast algorithm presented in Subsection 2.1 for dynamic broadcast in an 8-dimensional hypercube and a 16-dimensional hypercube with various throughput factors. We can see that the time required for dynamic broadcast using our priority broadcast scheme increases slowly when the load factor ρ increases, until ρ is very close to 1.

When no other communication tasks are taking place, the system is stable as long as the throughput factor $\rho < 1$. This can be shown by arguing that the queues of all the network links will not build up with time. Although the approximating assumptions may lead to error in analysis when the traffic is very heavy, we can use other methods to show that the average reception delay is $O(n + \frac{1}{1-\rho})$. The details will be reported in the near future. Therefore, our algorithm improves on the broadcast time required by the direct schemes proposed in [14] by a factor of $\Theta(n)$ when the throughput factor is close to 1, and has similar performance when the throughput factor is close to 0.

3. A simple dynamic broadcast algorithm for k-ary n-cubes

In this section, we generalize the algorithm given in Section 2 for dynamic broadcast in k-ary n-cubes.

We randomly choose a dimension d, then a broadcast algorithm for a k-ary n-cube under the single-port communication model can be presented as follows:

- At time 1, the source node sends the packet to be broadcast along dimension $d+1 \bmod n$.
- At time t, $t = 2,3,\ldots n$, each node that has a packet forwards the packet along dimension $d+t \bmod n$.

This algorithm can also be easily modified to obtain a non-idling queueing version for dynamic broadcast under the all-port communication model as we did in Subsection 2.1. More precisely, in this modified algorithm all the packets are sent along exactly the same path as the preceding broadcast algorithm, but a packet is stored in an output queue when the associated link is not available and is forwarded as soon as the associated link is available. In our priority broadcast scheme, we simply assign low priority to the packets that will be forwarded over dimension-d links (i.e., the last $\lfloor k/2 \rfloor$ steps along a routing path), and assign high priority to the remaining packets. Figure 3 illustrates an example for dynamic broadcast in a k-ary n-cube based on the priority broadcast scheme.

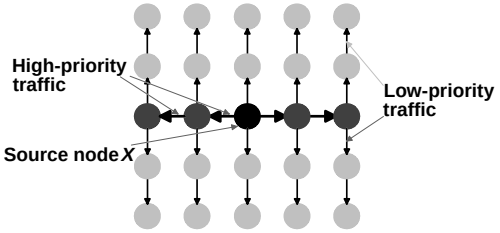

Figure 3. Dynamic broadcast in a 5-ary 2-cube based on the priority broadcast scheme.

From the preceding dynamic broadcast algorithm, we can see that a packet is forwarded as a high-priority packet for at most $\lfloor k/2 \rfloor (n-1)$ steps and is forwarded as a low-priority packet for at most $\lfloor k/2 \rfloor$ steps before it is received by a node. Moreover, since only (slightly less than) $1/n$ of the total traffic is high-priority traffic, the waiting time for a high-priority packet is a very small constant. Similar to the analysis given in Subsection 2.3, we find that the waiting time for a low-priority packet is $O(\frac{1}{1-\rho})$. Therefore, the average reception delay is given by

$$O\left(kn + \frac{k}{1-\rho}\right).$$

When k is a constant, the average reception delay is equal to $O(n + \frac{1}{1-\rho})$ as an n-dimensional hypercube and is asymptotically optimal. As a comparison, by generalizing the broadcast scheme proposed in [14] for dynamic broadcast in k-ary n-cubes, the average reception delay is $O(\frac{kn}{1-\rho})$ and is suboptimal. Our priority broadcast scheme, again, improves on this time complexity and that required by the dynamic broadcast algorithm proposed in [21] for arbitrary network topology by a factor of $\Theta(n)$ when the load factor is large.

The algorithm proposed in this section can be easily generalized to meshes and tori for dynamic broadcast. The only difference is that the probability for a dimension to be selected is determined by the number of nodes along that dimension in order to balance the traffic. The details will be reported in the near future.

4. Optimal priority assignment for dynamic broadcast

In this section, we present several methods for assigning priority classes to packets, including the *optimal priority as-*

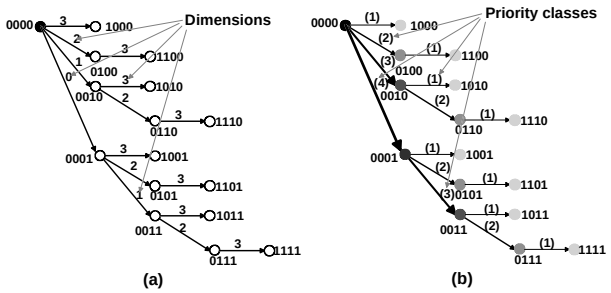

Figure 4. (a) The completely unbalanced spanning tree T_1 rooted at node 0 of a 4-cube. (b) Optimal priority assignment in the completely unbalanced spanning tree T_1.

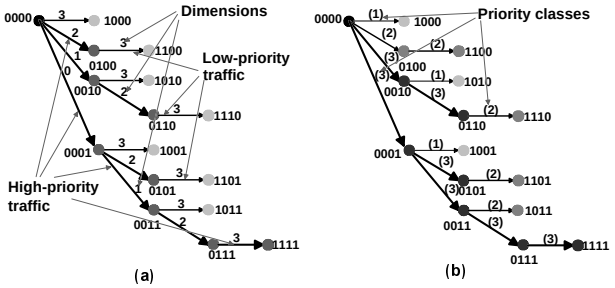

Figure 5. Assignments of priority classes to reduce the delay of dynamic broadcast in a 4-cube. (a) An assignment with 2 priority classes. (b) An assignment with 3 priority classes.

signment method, which achieves the best performance in terms of the average reception delay for dynamic broadcast.

There exist many other alternatives to the methods used in previous sections for the assignment of priority classes to packets. For example, we can randomly select 2 (or more) dimensions, and partition the hypercube into four $(n-2)$-dimensional subcubes by fixing these two dimensions. We then broadcast the packet with high priority within the $(n-2)$-dimensional subcube to which the source node belongs and forward the packets from this subcube to the other subcubes with low priority. About 25% of the total packets are high-priority packets in this case and the high-priority traffic becomes a traffic load with throughput factor smaller than 0.25 when the system is stable. We can also recursively use the priority broadcast scheme for broadcasting within a subcube. In the latter case, we will assign more than 2 priority classes to packets in the network.

These examples and the algorithm given in Subsection 2.1 are actually equivalent to selecting several spanning trees and assigning different priority classes to the nodes in the spanning trees. For example, the algorithm given in Subsection 2.1 uses *n completely unbalanced spanning trees* rooted at the source node X, and assign low priority to the packets that will be forwarded to the leaves of the trees. Figure 4a presents a completely unbalanced spanning tree rooted at node 0 in a 4-cube. The other 3 spanning trees can be obtained by rotating the dimensions of the links (see Section 5). The first example in this section uses $n(n-1)$ completely unbalanced spanning trees and assigns low priority to the packets that will be transmitted over the two selected dimensions.

An efficient but more complicated method, called the *optimal priority assignment*, assigns a packet with higher priority if it has more descendants to be broadcast to. More precisely, let d_u and d_v be the numbers of nodes in the subspanning trees rooted at a node for broadcasting packets u and v, respectively. Then packet u has higher priority at that node if $d_u > d_v$, and vice versa. Such assignment can be easily represented when we use n completely unbalanced spanning trees for broadcasting in hypercubes using the algorithm given in Subsection 2.1. More precisely, a packet that has to be transmitted over a dimension-i link in the j^{th} completely unbalanced spanning tree (i.e., the spanning tree obtained by rotating the dimensions for $j-1$ times, see Section 5 or [10, 14] for details) is assigned priority class $(j-i-1 \bmod n)+1$. Figure 4b illustrates the optimal priority assignment for the first completely unbalanced spanning tree of node 0 in a 4-cube. The average reception delay required using optimal priority assignment is smaller than that required by the simple dynamic broadcast algorithm presented in Section 2. Actually, when the assignment of priority classes does not affect the arrival processes of packets at any node, this assignment achieves the best possible performance in terms of the average reception delay.

When two packets has the same number of descendants, we can also assign higher priority to the packet that is older. That is, when $d_u = d_v$, packet u has higher priority if its source is generated before that of packet v. To reduce the broadcast delay of dynamic broadcast (that is, the average time required for the last packet to be received in a broadcast task), we can assign different priority to packets according to their locations in the spanning tree. For example, we can assign nodes in critical paths with higher priority. Figure 5 presents two possible assignments of priority classes for dynamic broadcast in a 4-cube with small broadcast delay.

5. Dynamic broadcast in any vertex and edge symmetric networks

In this section, we propose a dynamic broadcast algorithm that is generally applicable to any vertex and edge symmetric network.

To obtain a broadcast algorithm for a symmetric network, we first derive a *shortest-path spanning tree* T_1 rooted at node X for the network. This can be easily done by *flooding* the network with packets from node X, and *killing* redundant packets when applicable [4]. Then we "rotate" the dimensions of links in this shortest-path spanning tree to derive the other $l-1$ shortest-path spanning trees, where l is the degree of the network. More precisely, the shortest-path spanning tree T_i, $i = 2,3,\ldots l$, is obtained by replacing each dimension-j link at level 0 with the dimension-$(j+i-1 \bmod n)$ link of node X, replacing each dimension-j link at level 1 (that was connected to the Y^{th} node at level 1 of T_1), with the dimension-$(j+i-1 \bmod n)$ link of the newly obtained Y^{th} node of T_i at level 1, and repeating this process until all links of T_1 are replaced. When node X generates a packet to be broadcast, it randomly selected a shortest-path spanning tree T_d and the network broadcasts the packet along the spanning tree. By vertex and edge symmetry, we can see that the traffic is balanced over all network nodes and links, when the sources are uniformly distributed among all network nodes. Note that for some networks, some of the shortest-path spanning trees generated using this method may be identical and can be removed. Since the packets are broadcast along shortest-path spanning trees, the average reception delay is minimized.

To combine the preceding broadcast algorithm with our priority broadcast scheme, we will assign low priority to packets with fewer descendants, and high priority to the remaining packets. A simple method is to assign low priority to packets for the leaves, then to packets with 2 descendants, and so on, until a constant fraction of the traffic is assigned with low priority. We can use the optimal priority assignment introduced in the previous section to optimize the performance. We can also use a method that requires fewer priority classes and achieves performance between those of the simple and optimal priority assignment methods.

The dynamic broadcast algorithm proposed in this section is simple and powerful and can be applied to a variety of important networks. In fact, the algorithms presented in Sections 2 and 3 are special cases of this broadcast algorithm, where half of the shortest-path spanning trees generated for a k-ary n-cube by the preceding method are redundant and have been removed. We can apply this algorithm to generalized hypercubes, and easily show that the reception delay for dynamic broadcast in an n-dimensional radix-r hypercube is $O(n+\frac{1}{1-\rho}) = O(\log_r N + \frac{1}{1-\rho})$, where N is the size of the network. We can also apply this algorithm to star graphs [2, 3] and show that the reception delay for dynamic broadcast in an n-dimensional star graph is $O(n+\frac{1}{1-\rho}) = O(\frac{\log N}{\log\log N} + \frac{1}{1-\rho})$. Both algorithms for the star graphs and generalized hypercubes are asymptotically optimal. The lower bounds can be derived as the proof given in [14] for the lower bound on the time required by any oblivious algorithm for dynamic broadcast in hypercubes.

We can also use other spanning trees (e.g., [8, 22]) to execute the broadcast task. The most important criterion for selecting the spanning trees is that the traffic should be balanced among network nodes and links in order to maximize the maximum possible throughput. It is also important that the routing paths are as short as possible in order to reduce the reception delay and broadcast delay. Moreover, it is desirable that these spanning trees has $O(N)$ leaves or has $O(N)$ nodes in the lowest few levels.

Homogeneous product networks form a subclass of product networks with identical factor graphs [7]. More precisely, an l-level homogeneous product network is the iterated Cartesian product $\underbrace{G \times G \times \cdots \times G}_{l}$ of the same graph G. Hypercubes, k-ary n-cubes, and radix-r generalized hypercubes are all examples of homogeneous product networks whose factor graphs are the 2-node ring, k-node ring, and r-node complete graph, respectively. Similar to the way we generalize the dynamic broadcast algorithms for hypercubes and k-ary n-cubes to general symmetric networks, we can also generalize these dynamic broadcast algorithms to general homogeneous product networks. More precisely, we randomly select a level d, and broadcast the packet within the $(d+1 \bmod l)^{th}$ factor graph to which the source node belongs, and then broadcast the packet using links of the $(d+2 \bmod l)^{th}$, $(d+3 \bmod l)^{th}$, ..., d^{th} factor graphs. All (or part) of the packets transmitted over links of the d^{th} factor graph are assigned low priority and the remaining packets are assigned high priority. We can also use any other method introduced in this paper, such as the optimal priority assignment method, for the assignment of priority classes to packets. We should use the criteria introduced in this section for selecting spanning trees for dynamic broadcast within a factor graph of the homogeneous product networks. The priority broadcast scheme proposed in this paper can also be applied to a variety of other network topologies, such as the macro-satr networks [26], cyclic networks [24], and hierarchical swapped networks [23], for dynamic broadcast with high performance. The details will be reported in the future.

6. Conclusion

In this paper, we proposed an efficient routing scheme, called the priority broadcast scheme, for dynamic broadcast in hypercubes, k-ary n-cubes, meshes, tori, star graphs, and generalized hypercubes, as well as any symmetric net-

work or homogeneous product network. In particular, the dynamic broadcast algorithms we proposed for hypercubes improve the best previous algorithms significantly and are the only known algorithms that achieve optimal $O(n+\frac{1}{1-\rho})$ reception delay. In particular, our algorithms are optimal within a factor approximately equal to 1 when the load factor is close to 0 and within a small constant factor for any other load factor. The proposed algorithms for $k_1 \times k_2 \times \cdots \times k_n$ meshes or tori with $k_i = O(1)$, k-ary n-cubes with $k = O(1)$, star graphs, and generalized hypercubes are also asymptotically optimal. The proposed broadcast scheme can be applied to a variety of other network topologies for efficient dynamic broadcast. We also introduced the *optimal priority assignment* method for efficient assignment of priority classes to packets, which achieves the best possible performance for dynamic broadcast in any network topology.

References

[1] Abraham, S. and K. Padmanabhan, "Performance of the direct binary n-cube network for multiprocessors, *IEEE Trans. Computers*, vol. 38, no. 7, Jul. 1989, pp. 1000-1011.

[2] Akers, S.B., D. Harel, and B. Krishnamurthy, "The star graph: an attractive alternative to the n-cube," *Proc. Int'l Conf. Parallel Processing*, 1987, pp. 393-400.

[3] Akers, S.B. and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE Trans. Computers*, Vol. 38, Apr. 1989, pp. 555-565.

[4] Bertsekas, D.P. and R. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, N.J., 1992.

[5] Bertsekas, D.P. and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, 1997.

[6] Bhuyan L.N. and D.P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. Comput.*, vol. 33, no. 4, Apr. 1984, pp. 323-333.

[7] Efe, K. and A. Fernandez, "Products of networks with logarithmic diameter and fixed degree," *IEEE Trans. Parallel Distrib. Sys.*, vol. 6, no. 9, Sep. 1995, pp. 963-975.

[8] Fragopoulou, P. and S.G. Akl, "Edge-disjoint spanning trees on the star network with applications to fault tolerance," *IEEE Trans. Computers*, vol. 45, no. 2, Feb. 1996, pp. 174-185.

[9] Greenberg, A.G. and B. Hajek, "Deflection routing in hypercube networks," *IEEE Trans. Communications*, vol. 40, no. 6, Jun. 1992. pp. 1070-1081.

[10] Johnson, S.L. and C.-T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Computers*, vol. 38, no. 9, Sep. 1989, pp. 1249-1268.

[11] Kleinrock, L., *Queueing Systems, Vol. II: Computer Applications*, John Wiley & Sons, New York, 1976.

[12] Lakshmivarahan, S. and S.K. Dhall, "A new hierarchy of hypercube interconnection schemes for parallel computers," *J. Supercomputing*, vol. 2, 1988, pp. 81-108.

[13] Sharma, V. and E.A. Varvarigos, "Circuit switching with input queuing: an analysis for the d-dimensional wraparound mesh and the hypercube," *IEEE Trans. Parallel Distrib. Sys.*, vol. 8, no. 4, Apr. 1997, pp. 349-366.

[14] Stamoulis, G.D. and J.N. Tsitsiklis, "Efficient routing schemes for multiple broadcasts in hypercubes," *IEEE Trans. Parallel Distrib. Sys.*, vol. 4, no. 7, Jul. 1993, pp. 725-739.

[15] Stamoulis, G.D. and J.N. Tsitsiklis, "The efficiency of greedy routing in hypercubes and butterflies," *IEEE Trans. Communications*, vol. 42, no. 11, Nov. 1994, pp. 3051-3061.

[16] Valiant, L.G., "A scheme for fast parallel communication," *SIAM J. Computing*, vol. 11, no. 2, May 1982, pp. 350-361.

[17] Varvarigos, E.A., "Static and dynamic communication in parallel computing," Ph.D. dissertation, Dept. Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1992.

[18] Varvarigos, E.A. and D.P. Bertsekas, "Multinode broadcast in hypercubes and rings with randomly distributed length of packets," *IEEE Trans. Parallel Distrib. Sys.*, vol. 4, no. 2, Feb. 1993, pp. 144-154.

[19] Varvarigos, E.A. and D.P. Bertsekas, "Partial multinode broadcast and partial exchange algorithms for d-dimensional meshes," *J. Parallel Distributed Computing*, vol. 23, no. 2, Nov. 1994, pp. 177-189.

[20] Varvarigos, E.A. and D.P. Bertsekas, "Dynamic broadcasting in parallel computing," *IEEE Trans. Parallel Distrib. Sys.*, vol. 6, no. 2, Feb. 1995, pp. 120-131.

[21] Varvarigos, E.A. and A. Banerjee, "Routing schemes for multiple random broadcasts in arbitrary network topologies," *IEEE Trans. Parallel Distrib. Sys.*, vol. 7, no. 8, Aug. 1996, pp. 886-895.

[22] Wang, F.-H. and F.-C. Lin "On constructing multiple spanning trees in a hypercube," *Information Processing Letters*, vol. 45, no. 4, Mar. 1993, pp. 177-183.

[23] Yeh, C.-H. and B. Parhami, "Recursive hierarchical swapped networks: versatile interconnection architectures for highly parallel systems," *Proc. IEEE Symp. Parallel and Distributed Processing*, Oct. 1996, pp. 453-460.

[24] Yeh, C.-H. and B. Parhami, "Cyclic networks – a family of versatile fixed-degree interconnection architectures," *Proc. Int'l Parallel Processing Symp.*, Apr. 1997, pp. 739-743.

[25] Yeh, C.-H., "Efficient low-degree interconnection networks for parallel processing: topologies, algorithms, VLSI layouts, and fault tolerance," Ph.D. dissertation, Dept. Electrical & Computer Engineering, Univ. of California, Santa Barbara, Mar. 1998.

[26] Yeh, C.-H. and E. A. Varvarigos, "Macro-star networks: efficient low-degree alternatives to star graphs," *IEEE Trans. Parallel Distrib. Sys.*, Vol. 9, no. 10, Oct. 1998, pp. 987-1003.

[27] Youssef, A., "Design and analysis of product networks," *Proc. Symp. Frontiers of Massively Parallel Computation*, 1995, pp. 521-528.