An Optimal Routing Scheme for Multiple Broadcast

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

The dynamic broadcast problem is the communication problem where source packets to be broadcast to all the other nodes are generated at each node of a parallel computer according to a certain random process, such as a Poisson process. The lower bounds on the average reception delay required by any oblivious dynamic broadcast algorithm in a d-dimensional hypercube are $\Omega(d + \frac{1}{1-\rho})$ *when packets are generated according to a Poisson process, where* ρ *is the load factor. The best previous algorithms for hypercubes only achieve* $\Omega(\frac{d}{1-\rho})$ *average reception delay. In this paper, we propose dynamic broadcast algorithms that require optimal* $O(d + \frac{1}{1-\rho})$ *average reception delay in d-dimensional hypercubes and* $n_1 \times n_2 \times \cdots n_d$ *tori with* $n_i = O(1)$*. We apply the proposed broadcast scheme to a variety of other network topologies for efficient dynamic broadcast and present several methods for assigning priority classes to packets.*

1. Introduction

Meshes, tori, hypercubes, and n-ary d-cubes [12] are among the most popular network topologies for parallel processing, and many commercial and experimental parallel computers are built based on these networks. Star graphs [1, 2] and generalized hypercubes [6, 11] are also important networks that are recently receiving increasing attention. Numerous algorithms have been proposed for these networks [3, 5, 9, 21, 24].

Among the properties and algorithms investigated for these networks, dynamic broadcast is an important problem where each node in a parallel computer generates packets to be broadcast to all the other nodes according to a certain random process. When dynamic broadcast is the only type of communication tasks taking place, we define the *load factor* (or called *throughput factor*) of an N-node network as

$$\rho \stackrel{def}{=} \lambda_B \frac{N-1}{d_{ave}},$$

where λ_B is the rate at which the source packets to be broadcast are generated and d_{ave} is the average number of links per node. For example, the load factor of a d-dimensional hypercube is given by

$$\rho \stackrel{def}{=} \lambda_B \frac{2^d - 1}{d};$$

the load factor of an $n \times n$ mesh is given by

$$\rho \stackrel{def}{=} \lambda_B \frac{n^2 - 1}{4 - 4/n}.$$

Note that an N-node network will generate $\lambda_B N$ broadcast tasks per unit of time, which require at least $\lambda_B N(N-1)$ packet transmissions on the average. Since there are Nd_{ave} directed links in the network, the utilization of the most congested network links is at least equal to the load factor ρ. Note that the utilization is equal to ρ if and only if copies of the same source packet of a broadcast task are received exactly once by each node, and the packet transmissions are uniformly distributed over all network links. Therefore, a necessary condition for the stability of dynamic routing and dynamic broadcast in any network is that the load factor $\rho < 1$, when the source packets to be routed or broadcast are generated according to a random process.

The average broadcast delay is the average time that elapses between the generation of a source packet at a node and the time its broadcast to all the other nodes is completed; the average reception delay is the average time that elapses between the generation of a source packet at a node and the time a particular node receives a copy of the packet, averaged over all nodes. The lower bounds on the average broadcast delay and average reception delay required by any oblivious dynamic broadcast algorithm for a d-dimensional hypercube are $\Omega(d + \frac{1}{1-\rho})$ when the packets to be broadcast are generated according to a Poisson process [13]. Stamoulis and Tsitsiklis [13] proposed a *direct scheme* based on d completely unbalanced spanning trees and an *indirect scheme* based on d edge-disjoint spanning trees for dynamic broadcast in hypercubes. Their direct scheme requires $O(\frac{d}{1-\rho})$ average broadcast delay and reception delay; their indirect scheme is stable only when $\rho < \frac{2}{3}$ and re-

quires $O(\frac{d}{2-3\rho})$ average broadcast delay and reception delay. Varvarigos and Bertsekas [14] proposed the *dynamic broadcasting scheme* for the d-dimensional $n \times n \times \cdots n$ mesh, which is stable when $\rho < 1/2 - \Omega(\lambda_B dn)$ and requires $O(\frac{dn}{1/2-\lambda_B V-\rho})$ average broadcast delay and reception delay, where $V = (d+1)(n-1)$ when the splitting of packets is allowed and $V = 3d(n-1)+1$ otherwise. Varvarigos and Bertsekas [15] also formulated and proved the *dynamic broadcasting theorem* for dynamic broadcast based on partial multinode broadcast (PMNB). The dynamic broadcasting algorithm proposed in [15] is stable when $\rho < 1 - O(\lambda_B d)$ and requires $O(\frac{d}{1-\rho})$ average broadcast delay and reception delay. Varvarigos and Banerjee [16] also proposed a *direct broadcasting scheme* and an *indirect broadcasting scheme* for dynamic broadcast in arbitrary network topologies. The analyses given in [14, 15, 16] do not use any approximation. All these previously proposed algorithms can only achieve suboptimal performance (by a factor of $\Theta(d)$) when the load factor is large; some of them cannot achieve maximum load factor $\rho \approx 1$.

In this paper, we propose the priority broadcast scheme for dynamic broadcast in meshes, tori, hypercubes, n-ary d-cubes, as well as any vertex- and edge-symmetric networks. Our goal is to achieve the maximum possible load factor $\rho \approx 1$ and optimal average reception delay at the same time for dynamic routing and dynamic broadcast in these networks. We show that dynamic broadcast can be executed in d-dimensional hypercubes and $n_1 \times n_2 \times \cdots n_d$ tori with optimal $O(d + \frac{1}{1-\rho})$ average reception delay when $n_i = O(1)$ for all i. 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. Moreover, our dynamic broadcast algorithms are easy to implement in parallel computers and are considerably faster than the best previous algorithms for networks investigated in this paper.

We present algorithms for performing dynamic broadcast in tori with exactly balanced traffic over all network links, which achieves maximum throughput factor $\rho \approx 1$. We show that, based on our *algorithm-reconfigured communication (ARC) scheme*, dynamic broadcast can be executed in the interior $(n_1-2) \times (n_2-2) \times \cdots (n_d-2)$ submesh of an $n_1 \times n_2 \times \cdots n_d$ mesh with maximum throughput factor $1 - n_{max}d/N'$ and average reception delay $O(\sum_{i=1}^{d} n_i + \frac{n_{max}}{1-n_{max}d/N'-\rho})$, where $n_{max} = \max_{i=1,2,\ldots,d} n_i$ and $N' = \prod_{i=1}^{d}(n_i-2)$. The maximum throughput factor of our proposed algorithm is very close to optimal and is better than the best previous result [14] by a factor of approximately 2. The average reception delay of our proposed algorithm is asymptotically optimal when $n_{max} = O(1)$ and $\rho < 1 - cdn_{max}/N'$, for any constant $c > 1$. Moreover, we show that based on the ARC scheme, multinode broadcast (MNB) [5] can be executed in $\frac{N'}{2d} + n_{max} - 3$ time in the interior submesh, which is optimal within a factor of $1 + O(dn_{max}/N')$. We propose an efficient method for assigning priority classes to packets, called the *optimal priority assignment* method, which can achieves the best possible performance for dynamic broadcast in any network topology. We also extend these techniques and algorithms to star graphs [1, 2], generalized hypercubes [6, 11], homogeneous product networks [7], and any vertex- and edge-symmetric network.

In Section 2, we present dynamic broadcast algorithms in tori and illustrate the central idea of the priority broadcast scheme. In Section 3, we introduce the ARC scheme for dynamic broadcast and MNB in meshes. In Section 4, we propose the optimal priority assignment method for assigning priority classes to packets in arbitrary network topologies. In Section 5, we generalize the dynamic broadcast algorithms to any vertex and edge symmetric network.

2. Dynamic broadcast in tori

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

2.1. STAR and REDO broadcast for tori

For a given dimension l, a simple broadcast algorithm for a d-D $n_1 \times n_2 \times \cdots \times n_d$ torus under the *single-dimension communication model* [23], where the nodes are allowed to use only links of the same dimension at any given time, can be presented as follows:

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

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 broadcast 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 $2d$ 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. When an associated link is not available, the packet is stored in the associated output queue and waits for service.

The central idea of our proposed broadcast scheme is to balance the traffic over all network nodes and links, and then assign a proper priority class to each packet. Observe

that a broadcast task generates $a_{l+1,l} = n_{l+1} - 1$ packets over dimension-$(l+1)$ links, $a_{l+2,l} = (n_{l+2} - 1)n_{l+1}$ packets over dimension-$(l+2)$ links, $a_{i,l} = (n_i - 1)\prod_{j=l+1}^{i-1} n_j$ packets over dimension-i links if $i > l$, and $a_{i,l} = (n_i - 1)\prod_{j=l+1}^{n} n_j \prod_{j=1}^{j-1} n_j$ packets over dimension-i links if $i \leq l$. To balance the traffic, a node needs to select dimension $l = i$ with certain probability x_i for all $i = 1, 2, \ldots, d$. When there is no traffic other than dynamic broadcast tasks, the probability vector $(x_1, x_2, \ldots, x_d)$ can be obtained by solving the following system of d linear equations in d unknowns $\sum_{j=1}^{d} a_{i,j} x_j = \frac{N-1}{d}$ for $i = 1, 2, \ldots, d$, where $N = \prod_{i=1}^{d} n_i$ is the size of the torus. Note that it is guaranteed that the solution satisfies $\sum_{i=1}^{d} x_i = 1$ since $\sum_{i=1}^{d} a_{i,j} = N - 1$ for all $j = 1, 2, \ldots, d$. Clearly, if $n_i = n$ for all $i = 1, 2, \ldots, d$ (that is, the torus is an n-ary d-cube), we have $x_j = 1/d$ for all $j = 1, 2, \ldots, d$. When a node has a source packet to be broadcast to all the other nodes in the network, it randomly selects $l = i$ with probability x_i and then use the nonidling queueing version of the simple broadcast algorithm. Then the expected number of packets to be transmitted on each network link is the same for all links.

The preceding algorithm for the all-port communication model essentially find a broadcast algorithm under the single-dimension communication model and then rotate the dimensions of the single-dimension algorithm by l dimensions with probability x_i in order to find a broadcast algorithm that utilizes all dimensions uniformly. This strategy is useful for many problems and is called the *Single-To-All dimensions Rotation (STAR)* technique in this paper. The resultant broadcast algorithm is called a *STAR broadcast* algorithm.

We can generalize the STAR broadcast algorithm as follows. We first randomly select an order for all the dimensions $i = 1, 2, \ldots, d$ from $d!$ possible orders. Then at Phase t, $t = 1, 2, \ldots d$, each node that has a packet sends/forwards the packet along the t^{th} dimension in the selected order. When $n_i = n$ for all $i = 1, 2, \ldots, d$, we can select equal probability $x_j = 1/d!$ for each of the $d!$ possible orders to balance the traffic on network links; otherwise, we may need to solve a system of d linear equations in $d!$ unknowns. There are usually many solutions to such a system of linear equations. We can choose a solution where some x_i, $1 \leq x_i \leq d!$, are equal to 0 so that only a subset of the $d!$ possible orders of dimensions are used. This strategy is also useful and is called the *random even-dimensions order (REDO)* technique in this paper. The resultant broadcast algorithm is called a *REDO broadcast* algorithm.

2.2. The priority broadcast scheme for tori

A simple way to combine the nonidling queueing discipline with the *priority broadcast scheme* we propose is to assign low priority to the packets that will be forwarded over dimension-l links and assign high priority to the remaining packets. Figure 1 illustrates an example for dynamic broadcast in a 5-ary 2-cube based on the priority broadcast scheme.

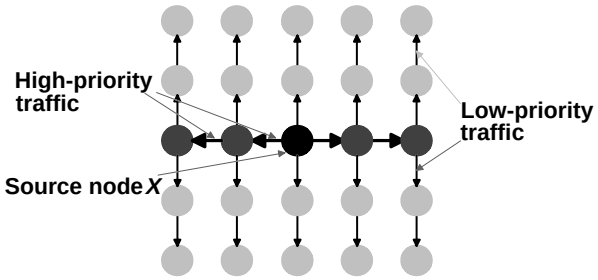

Figure 1. Dynamic broadcast in a 5-ary 2-cube based on the priority broadcast scheme.

/capt

To intuitively illustrate the central idea of our priority broadcast scheme, we analyze the average reception delay in a torus with $n_i = n$ for all i (i.e., an n-ary d cube) using a simple approximation, which assumes that the arrival processes of high-priority packets and of the aggregate packet traffic at a node can be approximated by Poisson processes. We assume that all packets have equal length and require one unit of time for transmission over links and let ρ_L be the arrival rate of low-priority packets and ρ_H be the approximate arrival rate of high-priority packets. Since there are $N/n - 1$ high-priority packets and $(1 - 1/n)N$ low-priority packets generated by a broadcast task, we have $\rho_H < 1/n$ and $\rho = \rho_H + \rho_L < 1$ when the system is stable. Therefore, the queues for high-priority packets can be approximated by M/D/1 queues [4, 10] with very small arrival rate so the average waiting time for a high-priority packet is approximately equal to $\frac{\rho_H}{2(1-\rho_H)} = O(1/n) = o(1)$.

According to the conservation law [10], 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. Assuming that the arrival process of the aggregate packet traffic at a node can be approximated by a Poisson process, then the average waiting time for packets (including both low-priority and high-priority packets) in our priority broadcast scheme can be approximated by using an M/D/1 queue with arrival rate ρ and is approximately equal to $\frac{\rho}{2(1-\rho)}$. Thus, the average waiting time for low-priority packets is approximately equal to $\frac{\rho n}{2(1-\rho)(n-1)} \approx \frac{\rho}{2(1-\rho)}$ under such assumption. Note, however, that about $(n-1)/n$ of the packets for transmission over a dimension i link of a node comes from the dimension-

i neighbor of that node so that the actual waiting time is considerably smaller.

From the preceding dynamic broadcast algorithm, we can see that a packet is forwarded as a high-priority packet for at most $\lfloor n/2 \rfloor (d-1)$ steps and is forwarded as a low-priority packet for at most $\lfloor n/2 \rfloor$ 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 average waiting time for a high-priority packet is a very close to 0. Since the average waiting time for a low-priority packet is $O(\frac{1}{1-\rho})$, the average reception delay is given by

$$O\left(nd + \frac{n}{1-\rho}\right).$$

When n is a constant, the average reception delay is $O(d + \frac{1}{1-\rho})$ and is asymptotically optimal from the lower bound shown in [13] for any oblivious algorithm. As a comparison, by generalizing the broadcast scheme proposed in [13] for dynamic broadcast in n-ary d-cubes or torus, the average reception delay is $O(\frac{dn}{1-\rho})$ and is suboptimal by a factor of $\Theta(d)$ even when $n = O(1)$. Our priority broadcast scheme also improves on this time complexity and that required by the dynamic broadcast algorithm proposed in [16] for arbitrary network topology by a factor of $\Theta(d)$ when the load factor is large. The approximate analysis given in this section can be easily generalized to general tori.

3. Dynamic broadcast in meshes

The best previous algorithms for dynamic broadcast in meshes can only achieve maximum throughput factor $\rho \approx 1/2$, and it is derived for $n \times n \times \cdots n$ mesh only [14]. In this section, we introduce the *algorithm-reconfigured communication (ARC) scheme* for efficient multinode broadcast (MNB) and dynamic broadcast in general meshes (i.e., n_i may not be equal to n_j). The goal of our routing scheme is to derive dynamic broadcast algorithms whose maximum possible throughput factor ρ is as close to 1 as possible.

3.1. The algorithm-reconfigured communication (ARC) scheme for dynamic broadcast in meshes

The load factor ρ of an N-node d-D mesh for dynamic broadcast with arrival rate λ is slightly larger than $\lambda N/2d$. However, if we use the dynamic broadcast algorithm proposed in Section 2, the maximum load factor that can be achieved is no larger than 1/2, even if the traffic is balanced for the most congested links of different dimensions (which are the directed links connecting to the surface of the mesh or to the top/bottom rows in a 2-D mesh). Since all the packets are routed along shortest paths in that algorithm, it is impossible to reduce the number of transmissions. In fact, we can show that no algorithm can achieve load factor larger than 1/2 for dynamic broadcast in the entire mesh.

In the algorithm-reconfigured communication (ARC) scheme, we "reconfigure" the mesh by selecting a subset of nodes in the mesh for computation, in order to improve the efficiency of communication by a factor of 2. This scheme is especially useful for communication-intensive algorithms where the network performance and throughput are determined by the communication time required.

To perform dynamic broadcast, we select the interior $(n_1-2) \times (n_2-2) \times \cdots (n_d-2)$ submesh for the purpose of computation. Then, when a source packet arrives, we first route it outside the submesh along a certain dimension i, and then broadcast it to the surface hyperplane that is outside the submesh and is orthogonal to dimension i. In other words, if the source node is $(y_1, y_2, \ldots, y_d)$, then we route the packet to node $(y_1, y_2, \ldots, y_{i-1}, 1, y_{i+1}, \ldots, y_d)$ (or node $(y_1, y_2, \cdots, y_{i-1}, n_i, y_{i+1}, \cdots, y_d)$ if $y_i > n_i/2$), and then broadcast it to nodes $(z_1, z_2, \ldots, z_{i-1}, 1, z_{i+1}, \ldots, z_d)$ (or $(z_1, z_2, \ldots, z_{i-1}, n_i, z_{i+1}, \ldots, z_d)$, respectively) for all $z_i = 2, 3, \ldots, n_i - 1$. Finally each node in the hyperplane that has a copy of the source packet broadcasts it back into the submesh along dimension i. Note that we should use the nonidling queueing version of this broadcast algorithm for dynamic broadcast. Figure 2 illustrates the ARC scheme for dynamic broadcast in a 2-D 6×6 mesh.

When dynamic broadcast is the only type of communication tasks taking place, the load factor of the N'-node interior submesh is

$$\rho \stackrel{def}{=} \lambda_B \frac{N'-1}{2d},$$

where $N' = \prod_{i=1}^{d} (n_i - 2)$. To achieve maximum throughput $\rho \approx 1$, we partition the submesh into $\frac{N'}{2d}$ equal parts so that each hyperplane receives the same number of source packets per unit of time on the average to be broadcast. Then the traffic on all network links is approximately balanced since about $\frac{N'}{2d}$ broadcast packets (excluding the source packet that is routed out of the submesh) will be sent across each link of the submesh every N' broadcast tasks. Then, when $\rho < 1 - dn_{max}/N'$, the utilization of all the network links is smaller than 1 so that the system is stable. Note that we can further improve the maximum possible throughput slightly by routing the source packet to the outside row or hyperplane along paths that generate more balanced traffic, at the expense of somewhat larger average reception delay. Note also that this interior submesh is the largest submesh that can achieve maximum throughput factor close to 1. Any submesh larger than this can only achieve maximum throughput factor $\rho < 3/4$.

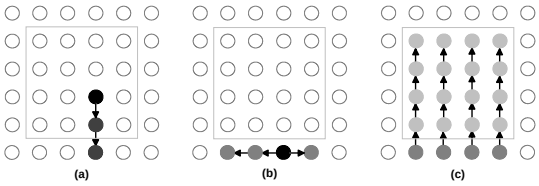

Figure 2. Dynamic broadcast in a 6×6 mesh based on the algorithm-reconfigured communication (ARC) scheme. (a) The source packet is routed out of the interior 4×4 submesh as a high-priority packet. (b) The packet is broadcast along the bottown row of the 6×6 mesh. (c) The packets are broadcast back to the 4×4 submesh along dimension 1.

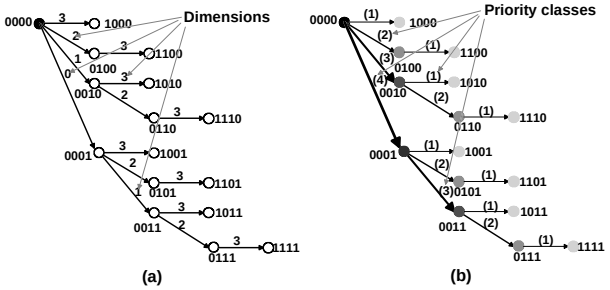

Figure 3. (a) The completely unbalanced spanning tree T_1 rooted at node 0 of a 4-cube. (b) Optimal priority assignment in the completely unbalanced spanning tree T_1.

3.2. The algorithm-reconfigured communication (ARC) scheme for MNB in meshes

The dynamic broadcast algorithm for meshes can be modified to obtain a multinode broadcast (MNB) algorithm for the interior $(n_1-2) \times (n_2-2) \times \cdots (n_d-2)$ submesh that requires $\frac{N'}{2d} + n_{max} - 3$ time. We simply use the ARC scheme to broadcast all packets with proper scheduling. We can show that since time $n_{max}-1$, all nodes in the submesh begin to receive packets from all their $2d$ links. If $n_i = n_{max}$, than some nodes will receive their last packets along dimension i at time $\frac{N'}{2d} + n_{max} - 3$, which is the execution time of the MNB task. The algorithm is asymptotically optimal within a factor of $1 + O(dn_{max}/N')$ from a trivial lower bound $\frac{N'-1}{2d}$ and is, to the best of our knowledge, smaller than the best previous algorithms reported in the literature [14] by a factor of approximately 2 for executing MNB tasks in meshes.

3.3. The priority broadcast scheme for dynamic broadcast in meshes

It is very easy to combine the priority broadcast scheme with the ARC scheme for dynamic broadcast in meshes. A simple method is to assign low priority to the last $n_i - 3$ transmissions (that is, to the packets that are broadcast back to the interior submesh from the surface hyperplane), and assign high priority to the remaining packets. Similar to the analysis given in Section 2 for hypercubes and n-ary d-cubes, we can show that the average reception delay for dynamic broadcast in the interior $(n_1-2) \times (n_2-2) \times \cdots (n_d-2)$ submesh is

$$O(\sum_{i=1}^{d} n_i + \frac{n_{max}}{1 - dn_{max}/N' - \rho}),$$

which is asymptotically optimal when $n_{max} = O(1)$ and $\rho < 1 - cdn_{max}/N'$ for any constant $c > 1$. The resultant algorithm achieves the best throughput and delay reported in the literature for dynamic broadcast in an N'-node submesh.

Our proposed dynamic broadcast algorithm for meshes achieve maximum throughput factor that is larger than that of the best previous algorithm for dynamic broadcast in the entire mesh [14] by a factor of approximately 2.

4. Optimal priority assignment for dynamic broadcast

In this section, we present several methods for assigning priority classes to packets, including the *optimal priority assignment* method, which achieves the best performance in terms of the average reception delay for dynamic broadcast.

For the algorithms presented in this paper, there are many ways in which priority classes can be assigned to packets. We can select several spanning trees of a mesh, torus, or hypercube and assign different priority classes to the nodes in the spanning trees. For example, the STAR broadcast algorithm for hypercube is equivalent to using *d 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 3a 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).

An efficient but more complicated method, which we call *optimal priority assignment*, assigns higher priority to packets that have more descendants to be broadcast to. More precisely, if N_u and N_v are the numbers of nodes in the subspanning trees rooted at a node for broadcasting packets u and v, respectively, and $N_u > N_v$, then packet u has higher

priority than v at that node. Such assignment can be easily represented when we use d completely unbalanced spanning trees for broadcasting in hypercubes. More precisely, a packet that has to be transmitted over a link of dimension i 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 [9, 13] for details) is assigned priority class $(j-i-1 \bmod d)+1$. Figure 3b 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 2. Actually, when the assignment of priority classes does not affect the packet arrival processes 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 $N_u = N_v$, packet u has higher priority if its source is generated before that of packet v. To reduce the average 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.

5. Dynamic broadcast in vertex and edge symmetric networks

In this section, we propose a dynamic broadcast algorithm for an arbitrary vertex and edge symmetric network.

To obtain a STAR 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 $p-1$ shortest-path spanning trees, where p is the degree of the network. More precisely, the shortest-path spanning tree T_i, $i = 2,3,\ldots p$, is obtained by replacing each dimension-j link at level 0 with the dimension-$(j+i-1 \bmod p)$ 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-$(j+i-1 \bmod p)$ 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_l 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. Similar to the method used in Section 2, we can also derive a more general REDO broadcast algorithm for vertex and edge symmetric networks.

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 fewer priority classes and achieve 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 STAR broadcast algorithms for n-ary d-cubes and d-dimensional hypercubes presented in Sections 2 are special cases of this broadcast algorithm, where half of the shortest-path spanning trees generated for an n-ary d-cube by the preceding method are redundant and have been removed. We can apply this algorithm to generalized hypercubes [6, 11], and easily show that the reception delay for dynamic broadcast in a d-dimensional radix-r hypercube is $O(d+\frac{1}{1-\rho}) = O(\log_r N + \frac{1}{1-\rho})$, where N is the size of the network. We can also apply this algorithm to star graphs [1, 2] and show that the reception delay for dynamic broadcast in a d-dimensional star graph is $O(d+\frac{1}{1-\rho}) = O(\frac{\log N}{\log\log N} + \frac{1}{1-\rho})$. Both algorithms for the star graphs and generalized hypercubes are asymptotically optimal. The lower bounds can be derived in a way similar to that given in [13] 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, 17]) 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 average reception delay and broadcast delay. Moreover, it is desirable that the spanning trees used have $O(N)$ leaves or $O(N)$ nodes in the lowest few levels.

Another interesting case is that of *homogeneous product networks*, which are a subclass of product networks with identical factor graphs [7]. More precisely, a d-level homogeneous product network is the iterated Cartesian product $\underbrace{G \times G \times \cdots \times G}_{d}$ of the same graph G. Hypercubes, n-ary d-cubes, and radix-r generalized hypercubes are all examples

of homogeneous product networks whose factor graphs are the 2-node ring, n-node ring, and r-node complete graph, respectively. By generalizing dynamic broadcast algorithms for tori, we can also obtain dynamic broadcast algorithms for homogeneous product networks. More precisely, we randomly select a level l, and broadcast the packet within the $(l+1 \bmod d)^{th}$ factor graph to which the source node belongs, and then broadcast the packet using links of the $(l+2 \bmod d)^{th}$, $(l+3 \bmod d)^{th}$, ..., l^{th} factor graphs. All (or part) of the packets transmitted over links of the l^{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 [23], cyclic networks [20, 22], and hierarchical swapped networks [18, 19], for dynamic broadcast with high performance. The details will be reported in the future.

6. Modified dynamic broadcast algorithms

In this section, we propose several algorithms that can further improve the performance for dynamic broadcast in networks investigated in this paper.

6.1. Heuristic dynamic broadcast algorithms

In an M/D/p queue [4, 10], a server is idle only when other servers are idle or have at most one customer; while in p M/D/1 queues whose arival processes are independant, it is possible that some of the servers are idle while the others have long queues. Therefore, the capacity of the p M/D/1 queues may be wasted and an M/D/p queue performs much better than p M/D/1 queues when the load factor is large and their aggregate arrival rates are the same.

The dynamic broadcast algorithms presented in the previous sections are oblivious and a node with p links performs like p independant M/D/1 queues. In algorithms proposed in this subsection, we manage to maintain similar queue lengths for links of all the dimensions in order to improve the performance. The central idea of the proposed algorithms is that if links of dimension i have longer queue on the average and/or have to transmit more packets in the near future, we assign smaller probability x_i for selecting ending dimension i (or for the choice of spanning trees that require more transmissions over dimension-i links of the network), and vice versa.

To implement the idea, we first define the *average unfinished load* of a network. Let $Tul(i)$ be the *total unfinished load* for network links of dimension i, the number of packets that have to traverse dimension i links in the network in order to complete all the currently unfinished broadcast tasks. Then the *average unfinished load* for a dimension-i link, $Aul(i)$, is equal to $Tul(i)/N$, where N is the number of dimension-i links in the network.

Each node X calculates the *expected unfinished load*, $Eul_X(i)$ for each dimension i, $i = 1, 2, ..., n$, for packets currently resides in a queue of node X, where $Eul_X(i)$ is the total number of transmissions over network links of dimension i required to broadcast the packets currently in its queues. We can show that the average value of $Eul_X(i)$ over all network nodes X is equal to $Aul(i)$.

For the oblivious broadcast algorithms introduced in previous sections, we randomly choose an ending dimension for STAR broadcast or an order of dimensions for REDO broadcast (with equal probability when the network is symmetric). In the algorithm proposed in this section, node X assigns probabilities for these orders according to the values of $Eul_X(i)$ in order to make $Aul(i) \approx Aul(j)$ for any $i \neq j$ at any time. We can show that for sufficiently large integer s, there exist a set of probabilities for selecting the order of dimensions such that the expected values of $Aul(i)$ are the same for all $i = 1, 2, 3, ..., p$ after s slots.

In addition to improved performance when the load factor is large, the requirement for buffers is also reduced by using this algorithm. Alternatively, a node can estimate the network traffic by averaging the queue lengths among neighboring or nearby nodes. More details will be reported in the near future.

6.2. Dynamic broadcast with mini-packets

When a packet can be split into several mini-packets, the performance of dynamic broadcast can be significantly improved for any load factor.

When the traffic is light, we split the source packet into t mini-packets and broadcast them along edge-disjoint spanning trees [8, 9], where t is the number of edge-disjoint spanning trees of the network. For example, there are d edge-disjoint spanning trees in a d-dimensional hypercube. When using this algorithm, it can be easily shown that the average broadcast delay for dynamic broadcast in hypercubes is $O(1)$ when $\rho \to 0$ and the transmission of a mini-packet over a network link require $1/d$ time.

When the traffic is heavy, we can split the source packet into t mini-packets and broadcast them along edge-disjoint spanning trees, or into d (or p) mini-packets for d-D tori (or degree-p symmetric networks) and broadcast them using the STAR technique, each with a different ending dimension. The average reception delay for dynamic broadcast is im-

proved by a factor of $\Theta(d)$ when $\rho \to 1$ and the transmission of a mini-packet over a network link require $1/d$ time.

7. Conclusion

In this paper, we proposed the priority broadcast scheme for dynamic routing and dynamic broadcast in meshes, tori, hypercubes, n-ary d-cubes, star graphs, and generalized hypercubes, as well as any symmetric network 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 $O(d + \frac{1}{1-\rho})$ average reception delay. Our dynamic broadcast algorithms for hypercubes 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 $n_1 \times n_2 \times \cdots \times n_d$ tori with $n_i = O(1)$, n-ary d-cubes with $n = O(1)$, star graphs, and generalized hypercubes achieve maximum load factor $\rho \approx 1$ and asymptotically optimal average reception delay. We showed that dynamic broadcast can be executed in the interior $(n_1 - 2) \times (n_2 - 2) \times \cdots (n_d - 2)$ submesh with maximum load factor ρ close to 1 and optimal average reception delay. We also showed that multinode broadcast (MNB) can be executed in the interior submesh in about $\frac{N}{2d}$ steps. 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] 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.

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

[3] Bertsekas, D.P., C. Ozveren, G.D. Stamoulis, P. Tseng, and J.N. Tsitsiklis, "Optimal communication algorithms for hypercubes," *J. Parallel Distrib. Computing*, vol. 11, no. 4, Apr. 1991, pp. 263-275.

[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] 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.

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

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Yeh, C.-H. and B. Parhami, "Swapped networks: unifying the architectures and algorithms of a wide class of hierarchical parallel processors," *Proc. Int'l Conf. Parallel and Distributed Systems*, 1996, pp. 230-237.

[19] 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.

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

[21] Yeh, C.-H. and B. Parhami, "Optimal sorting algorithms on incomplete meshes with arbitrary fault patterns," *Proc. Int'l Conf. Parallel Processing*, Aug. 1997, pp. 4-11.

[22] 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.

[23] Yeh, C.-H. and E.A. Varvarigos, "Macro-star networks: efficient low-degree alternatives to star graphs," *IEEE Trans. Parallel Distrib. Sys.*, Oct. 1998, to appear.

[24] Yeh, C.-H. and B. Parhami, "Efficient sorting algorithms on incomplete meshes," *J. Parallel Distrib. Comput.*, to appear.