A Fast Tree-Based Barrier Synchronization on Switch-Based Irregular Networks

Sangman Moh[†], *Chansu Yu*[†], *Dongsoo Han*[†], *Ben Lee*[†‡], *and Dongman Lee*[†]

[†]School of Engineering
Information and Communications University
58-4 Hwa-am, Yu-sung, Taejon, 305-348 KOREA
{smmoh,cyu,dshan,dlee}@icu.ac.kr

[‡]Department of Electrical and Computer Engineering
Oregon State University
Corvallis, OR 97331
benl@ece.orst.edu

Abstract

Cooperating processes in different nodes of cluster systems exchange messages through the switch-based network of irregular topology, and the corresponding communication performance is the most critical factor in assessing the overall system performance. In particular, barrier synchronization among multiple processes in a process group usually constitutes the sequential or bottleneck part of a parallel program. In this paper, we propose a *Barrier Tree for Irregular Networks* (*BTIN*) and a barrier synchronization scheme using BTIN. The synchronization latency of the proposed BTIN scheme is asymptotically $O(\log n)$ while that of the fastest scheme reported in the literature is bounded by $O(n)$, where n is the number of member nodes. Extensive simulation study shows that for the group size of 256 the BTIN scheme improves the synchronization latency by a factor of 3.3 $\sim$ 3.8. It is also shown to be more scalable than conventional schemes with less network traffic.

Index terms: Cluster systems, switch-based network, irregular topology, barrier synchronization, hardware-supported barriers, communication latency, wormhole routing, MPI.

1 Introduction

Switch-based cluster systems have been widely accepted as cost-effective alternatives for high performance computers. Since computational nodes[1] or switches may be added to or detached from the network dynamically, it is generally assumed that the switches form an irregular topology [1, 2]. Such irregularity provides higher interconnection flexibility, greater system scalability, and incremental system expandability which are not attainable in traditional regular interconnection networks. Those advantages can be overshadowed by the irregularity itself, though. First, a system must identify the network topology before computation begins. Distributed reconfiguration algorithms in this regard have been studied [1, 2], and commercial switches such as Myrinet [3] employ such algorithms in practice. The second issue is on *routing*. Here the irregular topology makes it difficult to avoid *deadlock* among multiple packets traveling simultaneously [1, 2, 4, 5, 6]. *Up/down routing* algorithm prevents deadlock by restricting the sequences of *turns* in the routing paths [1], the idea of which was originally suggested for a regular mesh topology [7].

Routing complexity directly affects the communication performance which is the most critical factor in assessing the overall system performance in the switch-based cluster systems. Especially, collective operations need more attention than the operations with point-to-point communication since they often determine the execution time of the sequential part of a parallel program, which usually constitutes the bottleneck. As a result, *collective operations* have been studied over a decade for message-passing multicomputers which employ *point-to-point* direct networks of regular topology such as 2-D meshes and hypercubes [8, 9, 10]. Such collective operations, however, become more complicated for switch-based cluster systems [11, 12] due to the irregularity mentioned above.

This paper intends to draw attention on an important collective operation, *barrier synchronization*, on the switch-based networks of irregular topology. A *barrier* is a synchronization point in a parallel program at which all processes participating in the synchronization must arrive before any of them can proceed beyond the synchronization point. In general, barrier synchronization is split into two phases – *reduction* and *distribution*. During the reduction phase, each participating process notifies the root process of its arrival at the barrier point. Upon the notification from all member processes, the distribution phase begins and the root process notifies them that they can proceed further.

A straightforward implementation of barrier operation is to have multiple point-to-point messages from the root to the member nodes, but the performance can be significantly improved by reducing the number of messages. For regular interconnection networks, many research efforts have been devoted to develop efficient implementations of barrier synchronization with either software [13, 14, 15, 16] or hardware support [17, 18, 19, 20, 21, 22]. For instance, Xu, *et al.* [15] proposed a software tree approach for barrier synchronization in

[1]Nodes, in this paper, actually mean PCs or workstations in a cluster system.

wormhole-routed hypercube multicomputers. Tree-based schemes perform better than path-based schemes mainly due to the fact that the time complexity of tree-based approaches is $O(\log n)$ whereas that of path-based ones is $O(n)$, where n is the number of member nodes. *Software barriers*, however, inherently suffer from large communication latency. *Hardware-supported barriers*[2] are usually an order of magnitude faster than software barriers [22]. We have seen actual implementations of hardware support for barriers in some commercial massively parallel computers such as T3D and CM-5 [20, 22].

In this paper, we propose a *Barrier Tree for Irregular Networks* (*BTIN*) for the switch-based cluster systems of irregular topology, which significantly reduces the barrier synchronization latency and network traffic with no deadlock. It is a tree-based combining scheme which constructs a barrier tree and embeds it into the corresponding switches by putting special registers into the switches. It is also possible to build a barrier tree based on the up/down routing algorithm, but it is not optimal because a process group to be synchronized is just a subset of all the nodes in a system. We focus on establishing a new barrier tree or a collective routing tree to reduce the tree height effectively. The synchronization latency of the proposed BTIN scheme is $O(\log n)$ while that of the fastest scheme reported in the literature is bounded by $O(n)$, where n is the number of member nodes. Extensive simulation study shows that for the group size of 256 the BTIN scheme improves the synchronization latency by a factor of $3.3 \sim 3.8$. It is also shown to be more scalable than conventional schemes with less network traffic.

The rest of the paper is organized as follows. Switch-based cluster systems with irregular topology and parallel programming environment of barrier synchronization are described in the following section. In Section 3, we present the proposed tree-based barrier synchronization scheme, and the construction of a BTIN and the corresponding switch operations are discussed. Section 4 is devoted to analyzing the inherent characteristics of the BTIN scheme including tree height and deadlock issue. The performance of the proposed scheme is evaluated and discussed in Section 5, and conclusions and future works are covered in Section 6.

2 Barrier Synchronization on Irregular Networks

In this section, we introduce general switch-based cluster systems of irregular topology and the programming environment of barrier synchronization. The proposed tree-based barrier synchronization scheme is discussed in the next section. We first describe the system environment such as system model and point-to-point routing.

[2]Recently, hardware-supported multicast, another important type of collective operations, has been extensively studied in the context of switch-based irregular networks [25, 26, 27, 28, 29, 30, 31]. However, it is not directly applicable to implement barriers since multicasting does not support the many-to-one reduction phase of barrier operation.

2.1 Switch-Based Cluster Systems

System Model

A switch-based cluster system is assumed, which comprises k-port switches interconnected in an irregular topology and computational nodes attached to the switches. The switch-based network of irregular topology is represented with a graph $G(V, E)$ in which each vertex in $V(G)$ corresponds to a switch and each edge in $E(G)$ corresponds to a communication channel between vertices. A computational node is represented as a pair of (v_i, c_j), where v_i is a vertex (switch) in $V(G)$ and c_j is a pair of input and output channel attached to the switch v_i. Note here that $i = 1, 2, \cdots, |V(G)|$ and $j = 1, 2, \cdots, k$ for k-port switches. The distance between two nodes is defined as the number of routing steps or *hops* in the transmission of a message.

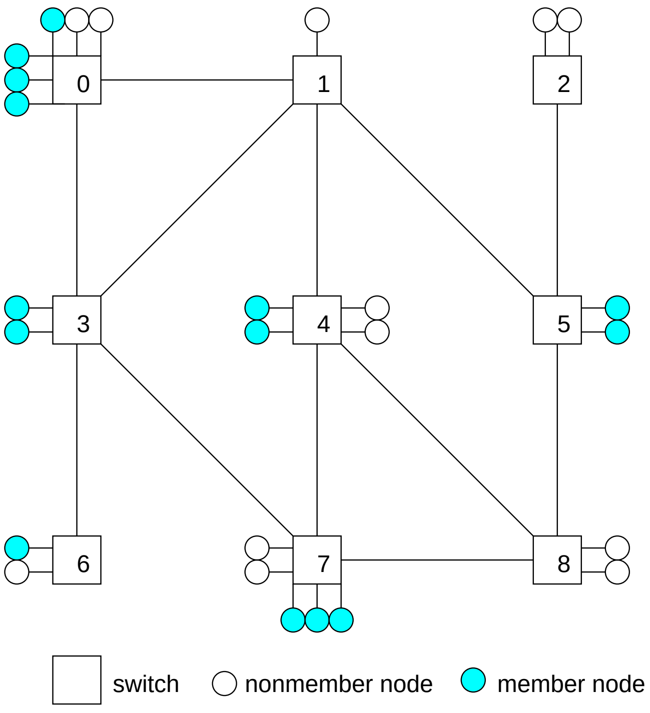

Figure 1: A switch-based cluster system with irregular topology.

An example of switch-based cluster systems with irregular topology is drawn in Figure 1. Network topology is irregular in a switch-based cluster system, where each switch has a set of ports and each port is connected to a computational node or other switch. Some ports may be left open and they can be used for further system expansion. The system in Figure 1 consists of nine 8-port switches, 26 computational nodes, and 12 inter-switch links. In this example, the problem is to synchronize the 14 member nodes in a process group (the dark circles in Figure 1) at a barrier point, which are selected for running a parallel application.

Cut-through switching and *wormhole routing* [34, 35, 36, 37] have been employed in most new generation networks to minimize the communication latency and buffer requirement. As explained earlier, routing in interconnection networks is restricted to prevent deadlock. Most recently, as it is feasible to implement a single-chip switch that can accommodate several whole packets with current VLSI technology, *virtual cut-through routing*[3] is considered in a cluster environment, while still achieving the advantages of virtual channels and adaptive routing [38, 39]. Without loss of generality, in this paper, we consider cut-through switches implementing wormhole routing [34, 35, 36, 37] with input and output buffers of one flit wide each.

A k-port switch typically provides a $k \times k$ crossbar connectivity in order to make a message from any input port to be transferred to any output port. As in most system configurations, we assume a simple node-switch connection with *one-port model*, in which a switch (router) is connected to the local node via a pair of input and output channel [40]. The switch is responsible for entering, leaving, passing, and replicating messages. The crossbar connectivity within the switch allows simultaneous transmission of messages between different input and output channel.

Hardware support for barrier synchronization is the barrier registers within switches. There will be a register assigned for each barrier, and similar concept has been assumed both in most hardware-supported synchronization schemes for regular networks [19, 20, 21] and in the switch-based multicast approach [32], where a processor can access an internal register within a switch. We assume that the barrier registers can hold an entire synchronization message. This can be justified by the fact that a synchronization message is very short and fixed in length since it needs not carry multiple destination addresses, and the size of synchronization data is quite small. Detailed usage and format of the barrier registers will be presented in the following sections.

Point-to-Point Routing

Roughly, the routing methods for irregular networks are grouped as two categories [38]. The *up/down routing* was proposed and commercially used in the DEC Autonet [1], in which a spanning tree constructed by a distributed algorithm prohibits channel dependency cycle while achieving deadlock freedom. Some modifications of the up/down routing has been proposed [6, 38, 41, 42]. *Eulerian trail routing* is another scheme proposed for irregular networks [4], which is also deadlock-free. Two unidirectional adaptive trails are constructed from two opposite unidirectional Eulerian trails. The Eulerian trails are determined based on some heuristics during initialization. However, not every network topology has such Eulerian trails [38].

Here we assume the routing scheme based on the *up/down routing* which is a variant of the *turn model* [7]. Upon startup of a cluster system, a node in each switch starts the configuration

[3]Virtual cut-through routing pipelines message transmission across multiple routers. In the absence of contention, it is the same as the wormhole routing. However, when a packet is blocked, it is removed from the network and stored in a single buffer [33, 36], requiring buffers with capacity for one or more packets to store blocked packets.

algorithm to figure out the overall topology. A *breadth-first spanning* (*BFS*) *tree* is computed and all switches eventually agree on a unique spanning tree. Deadlock-free routing is based on loop-free assignment of direction to the operational links and the following up/down rule: *a legal route must traverse zero or more links in the up direction followed by zero or more links in the down direction.* Routing algorithms for Myrinet switch from Myricom and Berkeley are similar to the up/down routing[4] [2]. Detailed explanation of the BFS and the up/down routing can be found in [1, 28].

2.2 Programming Interface of Barrier Synchronization

Collective operation primitives including barrier synchronization are included in most message-passing libraries. Among them, we target our discussion to *MPI* (*Message Passing Interface*) standard [23, 24]. However, the algorithms presented here can be applied to other message-passing systems with very little modifications.

In MPI, a group of communicating processes is defined within a context called *process group.* A unique group identifier is associated with each distinct process group. It is assumed that every group member has full information on both all member nodes in the group and the network topology of the cluster system. In reality, this can be easily implemented by a simple broadcast at the group creation time. After receiving the information, every member node can set up the barrier registers in the switch attaching it, in a distributed manner at the group creation time[5]. Furthermore, since a switch contains several barrier registers, multiple concurrent barriers from different groups can be supported. The maximum number of concurrent barriers is limited by the number of barrier registers available. If no available barrier register exists, additional barriers can be mapped onto the local memory of one member node attached to the corresponding switch, at the cost of increased synchronization latency of the barrier. When a process group is terminated, the corresponding barrier registers are released. Typically, the `MPI_Finalize()` routine terminates a process group.

3 Tree-Based Barrier Synchronization

In this section, we describe the proposed BTIN and the corresponding synchronization operations first, followed by the detailed procedure to construct BTIN including the selection of root switch.

[4]One disadvantage with the up/down routing is the increased congestion at the root of the BFS tree. Also, it is not always able to provide a minimal path between every pair of nodes due to the restriction, which will be more important as the network size increases [6]. Basic up/down routing algorithms have been extended to allow adaptivity [4, 5, 6] and multicast has also been studied in switch-based arbitrary networks [25, 26, 27, 28].

[5]The process group can be created not only statically (initially) using `MPI_Initialize()` but also dynamically using `MPI_Spawn()` or `MPI_Spawn_multiple()`. The dynamic process management is a new feature introduced in MPI-2. A major impetus comes from the intention of users migrating from PVM (Parallel Virtual Machine) [43] which defines a wealth of dynamic process management primitives [24].

3.1 Barrier Tree for Irregular Networks (BTIN)

In this paper, we define a *member switch* as a switch with at least one member node and a *nonmember switch* as one without any member node. Also note that a *representative member node* is defined as the member node attached to a member switch via the lowest numbered port. A BFS tree is constructed by a distributed algorithm around the root switch, the selection of which will be described in detail later in this subsection. At the group creation time, one of the member nodes attached to each member switch starts the algorithm to figure out the BFS topology, and then the nodes running the algorithm eventually agree on a unique BFS tree. Such a BFS tree prohibits channel dependency cycle while achieving deadlock freedom. The deadlock freedom will be described in the following section.

Once a BFS tree is found, the algorithm checks whether there is any nonmember leaf switch in the tree. If such a leaf switch exists, it is removed from the BFS tree. Each representative member node sets up a barrier register in the corresponding switch properly to embed the resulting BTIN into the network.

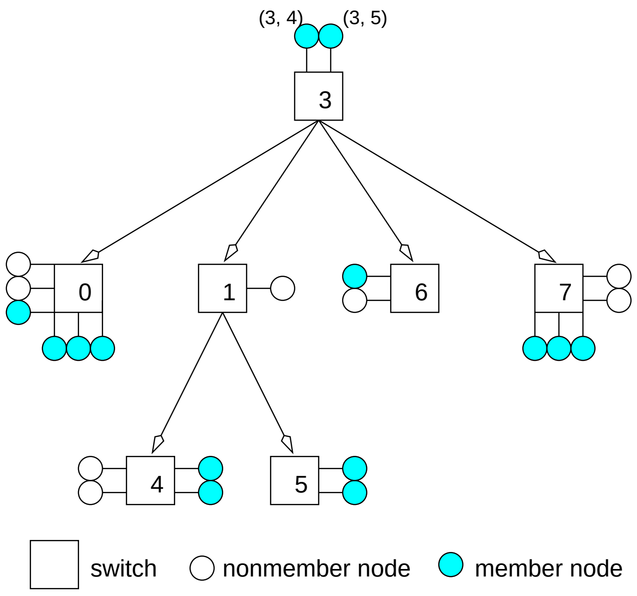

Figure 2: A BTIN constructed from the cluster system in Figure 1

Figure 2 shows a BTIN at the distribution phase, which is constructed from the cluster system in Figure 1 and contains the same 14 member nodes. The root switch is the switch labeled 3, and the root node is the node $(3, 4)$. Note here that the node notation $(3, 4)$ represents the node attached to port 4 of the switch 3. The children switches of the root switch are the switches 0, 1, 6, and 7. The child switch 1 is spanned down again to two children switches 4 and 5 as shown in Figure 2. The switches 4 and 5 have no children, and thus there is no further spanning. In Figure 2, the distribution message follows the arrows in accordance with the BTIN routing algorithm.

Note that the nonmember switches are not included in the tree unless they are intermediate switches in the tree. For example, the switches 2 and 8 in Figure 1 are not included in the constructed BTIN while the switch 1 is included as an intermediate switch in the tree. We next describe in detail how to systematically embed a BTIN into the network including the selection of root switch and root node.

Root Switch and Root Node

The root switch of a BTIN must be a member switch and is chosen so that the resulting BTIN has a minimum tree height among all possibilities. Our simple approach is as follows: Given u member switches in the system with q switches, u BFS trees rooted at each member switch are first generated and one with the minimum tree height is selected, where $q = |V(G)|$ of Section 2.1. If two or more BFS trees have the same minimum height, one with the minimum number of edges is chosen. However, if two or more BFS trees have the same number of edges as shown in Figure 3, one with the minimum number of leaves is selected. In Figure 3, for example, the second BFS tree of Figure 3(b) with two leaves is chosen among three BFS trees, which corresponds to BTIN shown in Figure 2. Finding the root switch is performed by the representative member node of each member switch in a distributed manner.

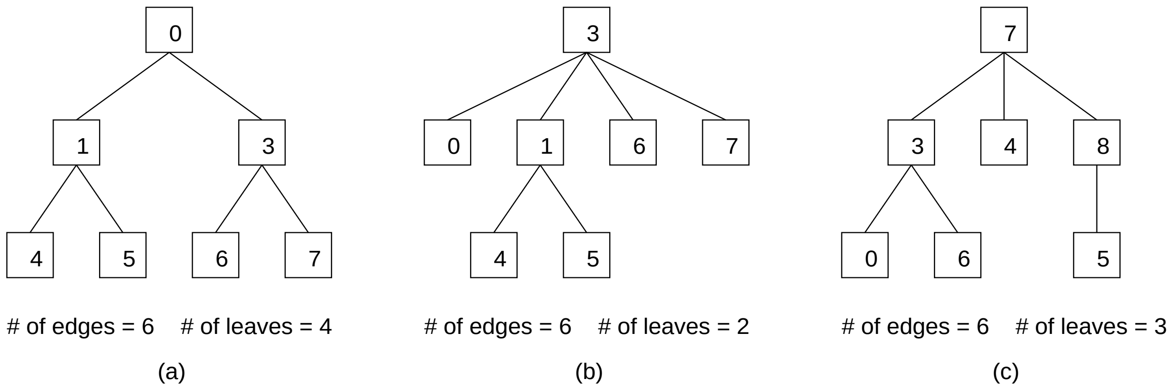
0
1
3
4
5
6
7
# of edges = 6 # of leaves = 4
(a)
3
0
1
6
7
4
5
# of edges = 6 # of leaves = 2
(b)
7
3
4
8
0
6
5
# of edges = 6 # of leaves = 3
(c)

Figure 3: Three BFS trees with the same minimum height of 2.

In the example of Figure 2, the root switch is the switch labeled 3, and the node $(3, 4)$ is selected as the root node since it is connected via the lowest numbered port (port identifier of 4) between the two member nodes. Note here that, unlike the up/down routing tree, edges between siblings are not permitted in BTIN. This makes the BTIN routing be simple. Upon the `MPI_Initialize()` call, every member process determines the root switch and root node simultaneously and independently. This is possible because every member in a process group has full information on both all member nodes in the group and the network topology of the cluster system.

Switch Setup

Next step is to setup the barrier registers within the switches properly, embedding BTIN itself into the barrier registers. Given k-port switches in an arbitrary network, each barrier register contains a group identifier (GID), a parent port number (P), parent and children bits, arrival bits for children, and synchronization data as shown in Figure 4. Essentially, barrier registers are embedded only in the switches and computational nodes need not have them.

Unlike other fields, arrival bits and message fields are used when the synchronization message is processed rather than at the initial setup time. For example, A_0 identifies that a reduction message has arrived from the child switch connected to C_0 during the reduction phase. As explained in Section 2.2, if there are more barriers than available registers in the switch, one can back up some registers into the local memory of the representative member node to make room for the new barriers, increasing the synchronization latency of the barrier.

Group Id (GID)	P	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Message

P: Port number for parent C_i : Parent or children for port i A_i : Arrived from C_i

Figure 4: Structure of a barrier register.

Below we describe the algorithm to setup the barrier register during the construction of a BTIN. At the tree creation time, every member node runs the distributed algorithm, `Setup_Register()`. A special operation is required to setup a barrier register in intermediate nonmember switches involved in BTIN. That is, if a member switch has any nonmember descendant switch which can be reached without passing through intermediate member switches, the representative member node of the member switch requests a node of the nonmember descendant switch to setup a barrier register in the descendant switch by transmitting a point-to-point message. Then, the destinated node will write the GID, the parent port number (P), and the parent and children bits into a barrier register in the corresponding switch.

Setup_Register(S, M, s_r, m_r, s_l, m_l, `GID`)

1. Let $S = \{s_0, s_1, \cdots, s_{q-1}\}$ be all the switches, $M = \{m_0, m_1, \cdots, m_{n-1}\}$ be the addresses of member nodes, s_r be the root switch, m_r be the root node, s_l be the local switch, m_l be the local node, and GID be the group identifier.

2. If the local node m_l is not the representative member node of the local switch s_l, return.

3. Around the root switch s_r, establish the corresponding BFS tree, scanning switches in order of switch identifier (address).

4. Remove nonmember leaf switches from the found BFS tree until there is no such a nonmember leaf switch, making the tree become a BTIN.

5. Write the GID, the parent port number (P), and the parent and children bits into a barrier register in the local switch s_l.

6. If there exists any nonmember descendant switch which can be reached without passing through intermediate member switches, request a node of the nonmember descendant switch to setup a barrier register in the descendant switch by transmitting a point-to-point message. Then, the destinated node will write the GID, the parent port number (P), and the parent and children bits into a barrier register in the corresponding switch. Repeat this step 6 until there is no such a nonmember descendant switch.

In the algorithm `Setup_Register()`, the switch set S is the same as $V(G)$ and the address of a member node is represented as a pair of switch and port (channel) as defined in Section 2.1. Note that, in the proposed barrier synchronization tree, nonmember switches but intermediate ones of BTIN are not included in BTIN and thus they need not allocate any resource for the barrier synchronization. Only the switches forming a logical BTIN are involved in the switch setup operation.

3.2 Barrier Synchronization Using BTIN

Synchronization Message

As described in Subsection 2.1, it is assumed that a barrier register can hold an entire synchronization message because the synchronization message is quite short in length. Figure 5 shows the format of synchronization message which contains message type, group identifier, and small synchronization data. For the example shown in Figure 1, a synchronization message may comprise at most two bytes, *i.e.*, 2-bit message type, 8-bit group identifier for at most 256 different groups, and at most 6-bit synchronization data including control information.

Message type	Group id.	Synchronization data

Figure 5: Format of the barrier synchronization message.

Collective Routing and Switch Operations

The BTIN routing or *collective routing* performs message merging and replication at the reduction and distribution phase, respectively. Collective merging and replication are carried out at the switches. That is, reduction messages are collectively merged at each switch and forwarded to its parent during the reduction phase while a distribution message is replicated at each switch and forwarded to its children during the distribution phase. The primary difference between BTIN and the up/down routing tree is that BTIN has no edges between siblings, and thus there is no adaptive path in BTIN and the *up/down rule* of the up/down routing is simply applied to the BTIN routing as a basic rule of inter-switch routing. The deadlock issue in the collective routing will be presented in the following section in detail.

Table 1: Switch operations for barrier synchronization.

Switch position	Reduction phase
Leaf switch	- Receive reduction messages from the local member nodes specified in the barrier register. - Forward the lastly arriving reduction message to the parent switch specified in the barrier register.
Intermediate switch	- Receive reduction messages from the local member nodes and children switches specified in the barrier register. - Forward the lastly arriving reduction message to the parent switch specified in the barrier register.
Root switch	- Receive reduction messages from the local nonroot member nodes and children switches specified in the barrier register. - Notify the root node of the reduction message arrival.
Switch position	**Distribution phase**
Root switch	- Receive a distribution message from the root node. - Forward it to the children switches specified in the barrier register, and at the same time, notify the local nonroot member nodes of the distribution message arrival.
Intermediate switch	- Receive a distribution message from the parent switch specified in the barrier register. - Forward it to the children switches specified in the barrier register, and at the same time, notify the local member nodes of the distribution message arrival.
Leaf switch	- Receive a distribution message from the parent switch specified in the barrier register. - Notify the local member nodes of the distribution message arrival.

In the proposed collective routing, three types of switches, the root switch, the intermediate switch, and the leaf switch, perform different operations in accordance with their position in BTIN. *Switch operations* at the three types of switches are described in Table 1 which explains the switch operations for the subsequent reduction and distribution phase during a barrier synchronization.

During the reduction phase, the reduction messages traverse in the up direction upward the root switch, being combined collectively at each branch switch. At each switch of BTIN, reduction messages are received along at most $(k-1)$ incoming links from the children switches and the local member nodes, and one of the reduction messages (e.g. the lastly arriving message) is forwarded to the parent switch. Since the leaf switches do not have children, they perform no the message reduction operation. The root node is the final destination of reduction messages.

During the distribution phase, the distribution messages traverse in the down direction downward all the leaf switches, being replicated at each branch switch. The root node is the original source of distribution messages. At each switch of BTIN, a distribution message is replicated and forwarded along at most $(k-1)$ outgoing links toward the children switches. At the same time, the switch notifies the local member nodes of the fact that the distribution message is received if they are destinated in the corresponding barrier register. Then, the destinated local member nodes can proceed beyond the barrier point. However, since the leaf switches do not have children, they perform no the message replication operation.

4 Characteristics of BTIN

Characteristics of BTIN are analyzed in this section. We will discuss the tree height and synchronization latency of BTIN first, and then move on to the deadlock problem, and lastly compare them with those of conventional approaches.

4.1 Tree Height of a BTIN

If we have an irregular network with k-port switches, each port of a switch is connected to a computational node or other switch. Some ports may be left open and they can be used for further system expansion. We define *connectivity*, or *connection ratio*, f of k-port switches as the ratio of the average number of connected ports over k [25]. Hence, $f \cdot k$ is the average number of ports in a switch, which are connected to either other switches or computational nodes. Practically, the range of f can be represented by $\frac{2}{k} \leq f \leq 1$ since at least two ports, one for other switch and the other for either other switch or a node, must be connected.

Synchronization latency is linearly proportional to the height of BTIN. In this subsection, thus, we analyze the average tree height of a BTIN which is established on a randomly built irregular network. Without loss of generality, we assume that all the possibilities of network configuration with k-port switches of connectivity f are equally likely. The following Theorem 1 formally analyzes the average height of a BTIN under the assumption.

Theorem 1: *For an irregular network with k-port switches of connectivity f, the average height, h, of a BTIN is asymptotically given by $h = \log_{(fk-p/q-1)} n$, where n is the number of member nodes, p is the number of nodes, and q is the number of switches.*

Proof: For valid network configuration, the average number of connected ports in a switch, fk, is given by $2 \le fk \le k$. In a switch, the average number of ports connected to other switches can be represented by

$$\frac{fkq - p}{q} = fk - \frac{p}{q},$$

where fkq is the total number of connected ports in a system. If all inter-switch links are utilized in a BTIN, the average number of children in a switch, q_c, can be obtained from the above equation by subtracting one link connected to its parent, which is given by

$$q_c = fk - \frac{p}{q} - 1.$$

However, after the construction of BTIN, some links in the network may not actually utilized. Thus, q_c is given by an inequality

$$q_c \le fk - \frac{p}{q} - 1.$$

On the other hand, as assumed earlier, all the possibilities of network configuration with k-port switches of connectivity f are equally likely. Let the number of switches in a subtree spanned along with port i be q_i. Then, due to probability theory, the number of distinct cases (combinations) which satisfy $q_1 + q_2 + \cdots + q_{k-1} = q - 1$, where $q_i \ge 0$ for all $i = 1, 2, \cdots, k-1$ and $q - 1 \ge 0$, is given by

$$\begin{pmatrix} (q-1) + (k-1) - 1 \\ (k-1) - 1 \end{pmatrix} = \begin{pmatrix} q+k-3 \\ k-2 \end{pmatrix}.$$

Since a BTIN has at most $(k-1)$ subtrees spanned down from the root switch, the tree height of a BTIN with q switches, $h(q)$, can be represented by

$$h(q) = 1 + \frac{1}{\begin{pmatrix} q+k-3 \\ k-2 \end{pmatrix}} \sum_{i=1}^{\begin{pmatrix} q+k-3 \\ k-2 \end{pmatrix}} \max(h(q_1), h(q_2), \cdots, h(q_{k-1})),$$

where q_i is the number of switches with i ports which is represented as $0 \le q_i \le q-1$ and $\sum_{i=1}^{k-1} q_i = q - 1$ for each network configuration. Note here that, if a port i is either connected to a node or left open, q_i is necessarily zero (0). If q is probabilistically large, $h(q)$ can be approximated by

$$h(q) = 1 + \log_{(fk-p/q-1)} \frac{q-1}{fk}.$$

Here fk is a small constant since $0 < f \leq 1$ and typically $4 \leq k \leq 16$. Thus, $h(q)$ can be rewritten asymptotically by

$$h(q) = \log_{(fk-p/q-1)} q.$$

Therefore, for an irregular network with k-port switches of connectivity f, asymptotically $h = \log_{(fk-p/q-1)} n$. $\Box$

According to Theorem 1, for an irregular network with k-port switches of connectivity f, the average height of a BTIN is $O(\log_{(fk-p/q-1)} n)$, where n is the number of member nodes, p is the number of nodes, and q is the number of switches. It is simply rewritten by $O(\log n)$ because $fk - \frac{p}{q} - 1$ becomes a constant. Hence, the associated routing latency has a time complexity of $O(\log n)$.

4.2 Deadlock Freedom

As explained earlier, the primary difference between BTIN and the up/down routing tree is that BTIN has no edges between siblings. Consequently, a BTIN is simply a $(k-1)$-ary tree embedded into k-port switches in an irregular network, and the corresponding collective routing of a BTIN, which is essentially based on wormhole routing, is performed along with the tree edges.

Wormhole routing is assumed to be used in the proposed scheme since it reduces the communication latency by pipelining the message transfer over a number of channels along its path. In irregular networks as well as regular ones, a major issue with the wormhole routing is deadlock [44]. When the path of a message is blocked, the message head as well as the rest of the message are stopped where they are, holding the buffers and channels along the path[6]. Deadlock could occur if these stoppages create a cyclic dependency. However, if the message size is small enough, deadlock could be easily avoided by holding the entire message in the switch. In our barrier synchronization scheme, the synchronization messages need not carry all the destination addresses, and thus the lengths are identical and very small as in switch-based barrier synchronization on regular networks [21]. Recently, a deadlock-free input-buffer-based replication mechanism was proposed [28] for implementing switch. This technique was shown to be effective in breaking the interdependency between tree branches. It also reduces the probability of network blocking significantly. For the barrier operations of the proposed scheme, since the size of the synchronization message is small and can be fixed, the size of the input buffer is also small, *i.e.*, the input buffer can be limited to a few flits, the number of flits in a synchronization message.

[6]With the store-and-forward or the virtual cut-through strategy, the rest of the message is moved to the switch where the head is stopped. Thus, deadlock can be easily avoided at the cost of large buffer in each switch for holding the entire message [45].

The basic technique for proving that a network is deadlock-free is to articulate the dependences that can arise between channels as a result of message movement, and to demonstrate that there exists no cycle in the resulting channel dependence graph [33]. This implies that no traffic patterns can lead to deadlock, where the traffic patterns include those incurred by three cases; a barrier synchronization, multiple concurrent synchronizations, and a mixture of synchronization messages and normal messages. For the BTIN-based barrier synchronization, the proof of deadlock freedom is similar to that in the CS scheme because, in both schemes, an entire synchronization message can be stored in a storage (barrier register) in the switch.

Now, we informally prove that BTIN is deadlock-free by explaining that the messages incurred by the three cases above do not create deadlock situation. The first case is whether multiple messages induced by a barrier synchronization create a cyclic dependency or not. There exist multiple barrier registers in a switch, which are allocated disjointly to different barriers. They allow a barrier register to be occupied only by one barrier. During a barrier synchronization, the reduction phase is carried out first, and then the distribution phase is performed subsequently. That is, the reduction message and the distribution message of the same barrier synchronization never compete for the same barrier register. In addition, since reduction messages travel upward BTIN and distribution messages travel downward BTIN for a barrier synchronization, there exists no cyclic dependency among the barrier registers belonging to the same barrier.

Secondly, when a reduction message enters a switch via the input buffer of an incoming link, its flits can be moved into its associated barrier register. A distribution message behaves similarly. Since each barrier register can hold an entire synchronization message, the message does not occupy the input buffer and it blocks neither synchronization messages nor normal messages. Also, since the synchronization messages of different barriers use different barrier registers, they never interfere with each other.

Finally, even though a blocked normal message holds a chain of channels and thus may also block a synchronization message, the synchronization message is stored in a barrier register in the switch, resulting in no channel dependency. When the blocking is removed later, the synchronization message can travel toward the next node with no possibility of a deadlock. It can thus be concluded that the BTIN routing algorithm and the associated switch operations for barrier synchronization is deadlock-free.

4.3 Comparison of Characteristics

In this subsection, we compare the characteristics of BTIN with those of two conventional approaches, the method using point-to-point messages and the method using switch-based multicast at the distribution phase. For simplicity, in this paper, we call the two approaches the unicast scheme and the multicast scheme, respectively. The comparisons are summarized in Table 2. Here various important factors of barrier synchronization are considered for comparative evaluation. As can be seen from the table, the proposed BTIN possesses preferable characteristics for all the factors studied, which results in a significantly better performance as identified in the following section.

Table 2: Characteristics of barrier synchronization schemes on irregular networks.

	Unicast scheme	Multicast scheme	BTIN scheme
Initialization at group creation time			
Routing path/tree construction	Centralized at the root	Centralized at the root	Distributed at all members
Router setup	None	At member nodes (for distribution)	At member nodes
During a barrier operation			
Hardware support	None	Distribution phase	Reduction and distribution phase
Synchronization message size	Short (single destination address)	Long (multiple destination addresses)	Short (no destination address)
Number of start-ups	$2n$ (n for reduction and n for distribution)	$n + 1$ (n for reduction and one for distribution)	2 (one for reduction and one for distribution)
Complexity of routing latency	$O(n)$	$O(n)$	$O(\log n)$
Primary weakness			
Primary weakness	Repetitive $2n$ unicast transfers (very slow)	Repetitive n unicast transfers for reduction and hardware for multicast	Hardware complexity at the router

The unicast scheme requires repetitive n point-to-point message transfers for each of reduction and distribution phase, and thus it has the complexity of $O(2n)$ for routing latency, which is simply rewritten by $O(n)$. Since the multicast scheme is the most recent and efficient barrier synchronization scheme, the BTIN scheme is compared with that in more detail. We assume that, in the multicast scheme, switch-based multicasting tree is used for the distribution phase in hardware level. In the multicast scheme, it is simple to see that the complexity of routing latency of the distribution phase is $O(\log n)$. However, for the reduction phase, repetitive n unicast message transfers are required, resulting in the complexity of $O(n)$. Hence, the complexity of the multicast scheme is $O(n + \log n)$, which is simply rewritten by $O(n)$.

The BTIN scheme alleviates the functional complexity at the switch both by embedding multiple destination addresses in the switches and by fully supporting all the operations during a barrier synchronization in hardware level. The most imminent advantage of employing BTIN is the reduced synchronization latency due to the tree-based hardware support which is embedded in the switches.

5 Performance Evaluation

In this section, the performance of the proposed BTIN scheme is evaluated using computer simulation. For different system configurations and parameters, it is also compared with that of the method using switch-based multicast. The experiment environment is presented first.

5.1 Experiment Environment

We evaluate the performance of the proposed tree-based barrier synchronization scheme on two different system configurations; (i) 256 nodes and 75 switches and (ii) 1024 nodes and 300 switches. We assume that the network is interconnected with 8-port switches having 75% connectivity. Here the 75% connectivity ($f = 0.75$) means that 6 out of 8 ports in a switch are connected to either other switches or computational nodes on the average. The member nodes are picked randomly and all the members are assumed to arrive at the barrier at the same time. Channel contention is not considered.

The *synchronization latency* is the most important performance metric of barrier synchronization, which is the interval from the time when the barrier synchronization is invoked until the time when all the member nodes finish the distribution phase. As another performance measure in our simulation, the *network traffic* incurred by the barrier synchronization is also investigated. This is measured by the number of links (hops) traversed by the synchronization messages during a barrier operation.

The default performance parameters have been assumed on the basis of overhead-minimized communication on advanced switches which can be implemented using contemporary silicon technology. Based on the basic performance values of such environment, we assume the following default performance parameters: communication startup time (t_s) of $2 \sim 10$ μsec, link propagation delay (t_p) of $20 \sim 40$ $nsec$, and switch (router) delay (t_r) of $300 \sim 500$ $nsec$. The startup time includes the software overheads for allocating buffers, copying messages, and initializing the router and DMA [46]. The router delay includes several steps of complicated operations and varies for various routing algorithms as Chien [45] analyzed. We also assume that the network interface delay is almost the same as the switch delay for our evaluation. Since synchronization messages do not need any data flits, the communication latency of a message transfer can be approximated to $t_s + d \cdot t_p + (d+1) \cdot t_r$, where d is the *distance* between the source and destination node in a communication. In a BTIN, d is simply obtained by adding two links to h, *i.e.*, $d = h + 2$, which is due to the two nodes, the root and a leaf node.

5.2 Simulation Results and Discussion

In this subsection, we present the simulation results for two different system configurations explained earlier. Two important performance metrics, the synchronization latency and the network traffic, are presented first. Next, the effect of several system parameters on the synchronization latency is analyzed.

Synchronization Latency and Network Traffic

Figure 6 shows the synchronization latency, where t_s, t_p, and t_r are assumed to be 2.0 μsec, 20.0 $nsec$, and 300.0 $nsec$, respectively. Here for each parameter set, 100 simulation runs are executed, and the results are averaged. In most of the cases, very small variance is observed. Both the number of nodes and the number of switches are shown in the parenthesis of labels in Figure 6, where the connectivity f is 75%.

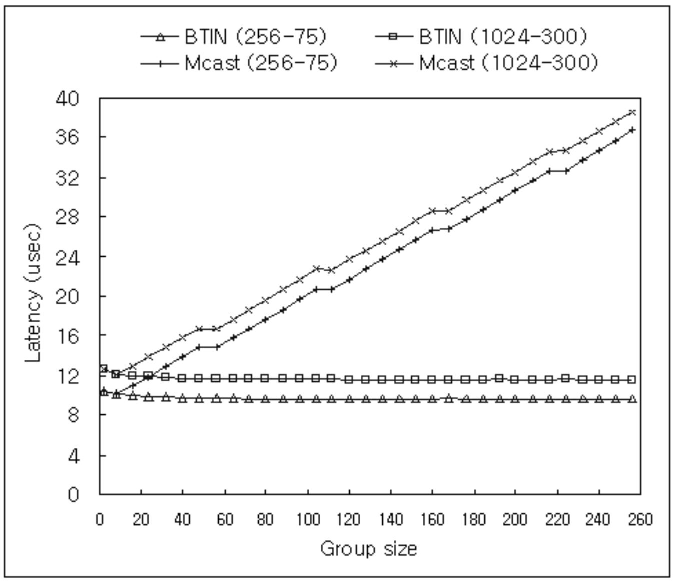

Figure 6: Synchronization latency.

The synchronization latency of the BTIN scheme is significantly lower than that of the multicast scheme. Observe from the figure that the synchronization latency of the BTIN scheme is almost independent on the group size except for very small groups. This is mainly due to the fact that the tree height of BTIN is bounded by $\log_{(fk-p/q-1)} q$ on an irregular network with k-port switches of connectivity f, where p is the number of nodes and q is the number of switches. In a system of 1024 nodes and 300 switches, the group size can be more than 256. Even though it is not shown here, we increased the group size up to 1024. Then the synchronization latency of the system of 1024 nodes and 300 switches converges to 11.5 μsec and 120.5 μsec for BTIN and multicast scheme, respectively. The performance improvement is more substantial as the size of network increases. For instance, for the group size of 256, the BTIN scheme is faster than the multicast scheme by factors of 3.8 and 3.3 for the system of 256 nodes and 75 switches and the system of 1024 nodes and 300 switches, respectively. It is obvious that the BTIN scheme is more scalable than the multicast scheme as the synchronization latency of the BTIN scheme is increased by 1.8 when both system size and group size are quadruplicated while that of the multicast scheme is increased by more than three times (a factor of 3.3).

The network traffic shown in Figure 7 is measured as the number of round-trip hops (links) which are traversed during both reduction and distribution phase. As shown in the figure, the network traffic of the BTIN scheme is significantly lighter than that of the multicast scheme. The performance of network traffic is more improved as the network size increases. For instance, for the group size of 256, the network traffic of the BTIN scheme is significantly lighter than the multicast scheme by factors of 46.8 and 88.6 for the system of 256 nodes and 75 switches and the system of 1024 nodes and 300 switches, respectively. As the group size is increased, the network traffic is also increased for both of the schemes because more nodes and switches of the network are involved in a barrier synchronization. In addition, the proposed BTIN scheme is clearly more scalable than the multicast scheme even in terms of network traffic.

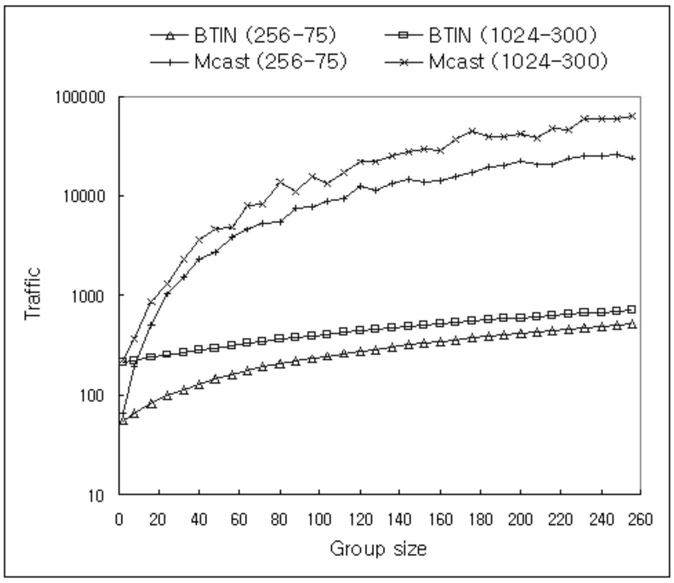

Figure 7: Network traffic.

Effect of Switch Connectivity

We also study the impact of variations in the switch connectivity f on the performance of the barrier synchronization schemes. On a system configuration of 256 nodes and 75 switches, in which 8-port switches are interconnected, we consider three different cases of f; 50%, 70%, and 90%. This parameter is shown in the parenthesis of labels in Figure 8, where the same performance parameters as in the earlier are applied. Even in this experiment, the BTIN scheme outperforms the multicast scheme for all the three cases of widely varying connectivity values. As the connectivity increases, more performance gain is achieved. For instance, for the group size of 256 on a system of 256 nodes and 75 switches, the BTIN scheme is faster than the multicast scheme by factors of 8.7 and 4.1 and 3.2 for the three cases of f, respectively. Again, the proposed BTIN scheme is still more scalable than the multicast scheme for various connectivity values. Note here that, compared to the multicast scheme, the BTIN scheme is less sensitive to connectivity f. The proposed scheme thus can be said to be significantly better than the multicast scheme for widely range of system size and network conditions.

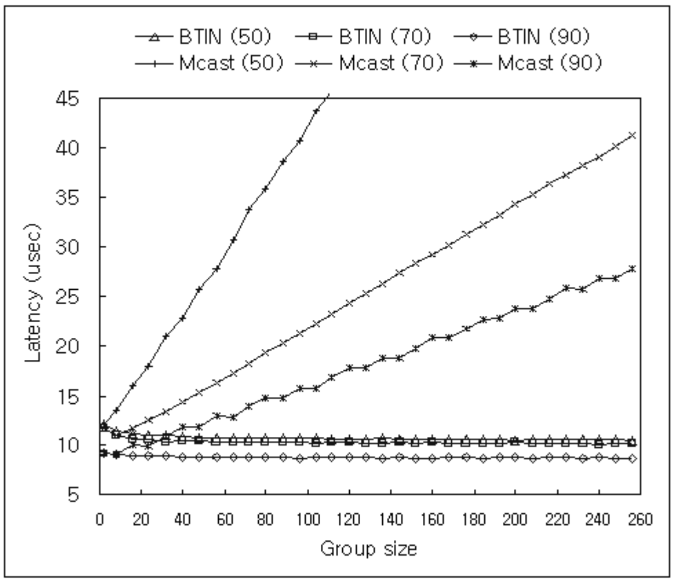

Figure 8: Effect of the switch connectivity.

6 Conclusions

In this paper, we have proposed a tree-based fast barrier synchronization scheme for switch-based irregular networks, which is, to the authors' knowledge, the first approach to hardware support for barrier synchronization on irregular networks. The tree constructed by the BTIN setup algorithm `Setup_Register()` (see Section 3) is at most $(k-1)$-ary, and the complexity of routing latency is $O(\log n)$ while that of the fastest scheme, which is the method using switch-based multicast at the distribution phase, is bounded by $O(n)$, where n is the number of member nodes. We have simulated and evaluated the performance of the proposed scheme, in which synchronization latency, effect of switch connectivity, and the network traffic incurred during the barrier synchronization have been compared with the multicast scheme. Extensive simulation study shows that for the group size of 256 the BTIN scheme improves the synchronization latency by a factor of $3.3 \sim 3.8$. From the quantitative evaluation, it is also obvious that the proposed BTIN scheme is more scalable than conventional schemes with less network traffic.

We currently investigate the application of the BTIN scheme to other collective communications such as multicast or total exchange. It is also an interesting subject to consider the BTIN scheme for dynamic environment caused by load balancing and node/link failures.

References

[1] M. D. Schroeder, *et.al.*, "Autonet: a High-speed, Self-configuring Local Area Network Using Point-to-point Links," *SRC Research Report*, No.59, Digital Equipment Corporation, April 1990.

[2] A. M. Mainwaring, B. N. Chun, S. Schleimer, and D. S. Wilkerson, "System Area Network Mapping," *Proceedings of the Annual Symposium on Parallel Algorithms and Architectures*, 1997.

[3] N. Boden, *et.al.*, "Myrinet: A Gigabit-per-Second Local Area Network," *IEEE Micro*, Vol. 15 No. 1, pp. 29-36, February 1995.

[4] W. Qiao and L. M. Ni, "Adaptive Routing in Irregular Networks Using Cut-Through Switches," *Proceedings of the International Conference on Parallel Processing*, Vol. 1, pp. 52-60, Aug. 12-16, 1996.

[5] F. Silla and J. Duato, "On the Use of Virtual Channels in Networks of Workstations with Irregular Topology," *Proceedings of the Workshop on Parallel Computer Routing and Communication*, Also in *Lecture Notes in Computer Science*, Vol. 1417, pp. 202-216, 1997.

[6] F. Silla, J. Duato, A. Sivasubramaniam, and C. R. Das, "Virtual Channel Multiplexing in Networks of Workstations with Irregular Topology," *Proceedings of the 5th International Conference on High Performance Computing*, pp. 147-154, Dec. 17-20, 1998.

[7] C. Glass and L. Ni, "The Turn Model for Adaptive Routing," *Journal of the ACM*, Vol. 41 No. 4, September 1994.

[8] K. Hwang, C. Wang, and C.-L. Wang, "Evaluating MPI Collective Communication on the SP2, T3D, and Paragon Multicomputers," *Proceedings of the Third International Symposium on High-Performance Computer Architecture*, pp. 106-115, Feb. 1997.

[9] K. Verstoep, K. Langendoen, and H. Bal, "Efficient Reliable Multicast on Myrinet," *Proceedings of the International Conference on Parallel Processing*, Vol. III, pp. 156-165, 1996.

[10] T. Kielmann, R. F. H. Hofman, H. E. Bal, A. Plaat, and R. A. F. Bhoedjang, "MAGPIE: MPI's Collective Communication Operations for Clustered Wide Area Systems," *Proceedings of the ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming*, pp. 131-140, 1999.

[11] R. Buyya, *High Performance Cluster Computing: Architectures and Systems,* Prentice-Hall Inc., NJ, 1999.

[12] G. F. Pfister, *In Search of Clusters*, 2nd Edition, Chapter 5, Prentice-Hall, Inc., NJ, 1998.

[13] R. Gupta, "The Fuzzy Barrier: A Mechanism for the High Speed Synchronization of Processors," *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 54-63, 1989.

[14] R. Gupta and C. R. Hill, "A Scalable Implementation of Barrier Synchronization Using an Adaptive Combining Tree," *International Journal of Parallel Programming*, Vol. 18, No. 3, pp. 161-180, June 1989.

[15] H. Xu, P. K. McKinley, and L. M. Ni, "Efficient Implementation of Barrier Synchronization in Wormhole-Routed Hypercube Multicomputers," *Journal of Parallel and Distributed Computing*, Vol. 16, pp. 172-184, 1992.

[16] P. K. McKinley, H. Xu, A. -H. Esfahanian, and L. M. Ni, "Unicast-Based Multicast Communication in Wormhole-Routed Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 12, pp. 1252-1265, Dec. 1994.

[17] C. J. Beckmann and C. D. Polychronopoulos, "Fast Barrier Synchronization Hardware," *Proceedings of the Supercomputing '90*, pp. 180-189, Nov. 1990.

[18] M. T. O'Keefe and H. D. Dietz, "Hardware Barrier Synchronization: Dynamic Barrier MIMD (DBM)," *Proceedings of the International Conference on Parallel Processing*, Vol. I, pp. 43-46, Aug. 1990.

[19] X. Lin, P. K. McKinley, and L. M. Ni, "Deadlock-Free Multicast Wormhole Routing in 2-D Mesh Multicomputers," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 8, pp. 793-804, Aug. 1994.

[20] D. K. Panda, "Fast Barrier Synchronization in Wormhole k-ary n-cube Networks," *Proceedings of the First IEEE Symposium on High-Performance Computer Architecture*, pp. 200-209, Jan. 22-25, 1995.

[21] J. -S. Yang and C. -T. King, "Designing Tree-Based Barrier Synchronization on 2D Mesh Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 6, pp. 526-534, June 1998.

[22] V. Ramakrishnan, I. D. Scherson, and R. Subramanian, "Efficient Techniques for Nested and Disjoint Barrier Synchronization," *Journal of Parallel and Distributed Computing*, Vol. 58, pp. 333-356, Aug, 1999.

[23] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard*, Version 1.1, June 12, 1995.

[24] Message Passing Interface Forum, *MPI-2: Extensions to the Message-Passing Interface*, July 18, 1997.

[25] R. Kesavan, K. Bondalapati, and D. K. Panda, "Multicast on Irregular Switch-Based Networks with Wormhole Routing," *Proceedings of the 3rd International Symposium on High-Performance Computer Architecture*, pp. 48-57, Feb. 1-5, 1997.

[26] R. Kesavan and D. K. Panda, "Multicasting on Switch-Based Irregular Networks using Multi-drop Path-Based Multidestination Worms," *Proceedings of the Workshop on Parallel Computer Routing and Communication*, Also in *Lecture Notes in Computer Science*, Vol. 1417, pp. 217-230, 1997.

[27] R. Sivaram, D. K. Panda, and C. B. Stunkel, "Multicasting in Irregular Networks with Cut-Through Switches Using Tree-Based Multidestination Worms," *Proceedings of the Workshop on Parallel Computer Routing and Communication*, Also in *Lecture Notes in Computer Science*, Vol. 1417, pp. 39-52, 1997.

[28] C. B. Stunkel, R. Sivaram, and D. K. Panda, "Implementing Multidestination Worms in Switch-Based Parallel Systems: Architectural Alternatives and their Impact," *Proceedings of the 24th ACM Annual International Symposium on Computer Architecture*, pp. 50-61, June 1997.

[29] R. Libeskind-Hadas, D. Mazzoni, and R. Rajagopalan, "Tree-Based Multicasting in Wormhole-Routed Irregular Topologies," *Proceedings of the International Parallel Processing Symposium*, pp. 244-249, Mar. 30 - Apr. 3, 1998.

[30] M. Gerla, P. Palnati, and S. Walton, "Multicasting Protocols for High-Speed, Wormhole-Routing Local Area Networks," *Proceedings of the International Conference on Applications, Techniques, Architectures, and Protocols for Computer Communication*, pp. 184-193, Aug. 28-30, 1996.

[31] D. Buntinas, D. K. Panda, J. Duato, and P. Sadayappan, "Broadcast/Multicast over Myrinet Using NIC-Assisted Multidestination Messages," *Proceedings of the Fourth International Workshop on Communication, Architecture, and Applications for Network-Based Parallel Computing*, Jan. 2000.

[32] R. Sivaram, R. Kesavan, D. K. Panda, and C. B. Stunkel, "Where to Provide Support for Efficient Multicasting in Irregular Networks: Network Interface or Switch?," *Proceedings of the International Conference on Parallel Processing*, pp. 452-459, Aug. 10-14, 1998.

[33] D. E. Culler, J. P. Singh, and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1999.

[34] W. J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers*, Vol. 20, No. 5, pp. 547-553, May 1987.

[35] L. Ni and P. K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *IEEE Computer*, Vol. 23, No. 2, pp. 62-76, Feb. 1993.

[36] J. Duato, S. Yalamanchile, and L. Ni, *Interconnection Networks: An Engineering Approach*, pp. 175-226, IEEE Computer Society, Los Alamitos, CA, 1997.

[37] P. Mohapatra, "Wormhole Routing Techniques for Directly Connected Multicomputer Systems," *ACM Computing Surveys*, Vol. 30, No. 3, pp. 374-410, Sep. 1998.

[38] H.-C. Chi and C.-T. Tang, "A Deadlock-Free Routing Scheme for Interconnection Networks with Irregular Topologies," *Proceedings of the International Conference on Parallel and Distributed Systems*, pp. 88-95, Dec. 10-13, 1997.

[39] J. Duato, A. Robles, F. Silla, and R. Beivide, "A Comparison of Router Architectures for Virtual Cut-Through and Wormhole Switching in a NOW Environment," *Proceedings of the 10th Symposium on Parallel and Distributed Processing*, pp. 240-247, Apr. 12-16, 1999.

[40] P. K. McKinley, Y.-J. Tsai, and D. F. Robinson, "Collective Communication in Wormhole-Routed Massively Parallel Computers," *IEEE Computer*, Vol. 28, No. 12, pp. 39-50, Dec. 1995.

[41] F. Silla, M. P. Malumbres, A. Robles, P. Lopez, and J. Duato, "Efficient Adaptive Routing in Networks of Workstations with Irregular Topology," *Proceedings of the Workshop on Communications and Architectural Support for Network-Based Computing*, Feb. 1-2, 1997.

[42] F. Silla and J. Duato, "Improving the Efficiency of Adaptive Routing in Networks with Irregular Topology," *Proceedings of the 4th International Conference on High-Performance Computing*, pp. 330-335, Dec. 18-21, 1997.

[43] Oak Ridge National Laboratory, *PVM: Parallel Virtual Machine*, `http://www.epm.ornl.gov/pvm/`, Nov. 1999.

[44] S. Warnakulasuriya and T. M. Pinkston, "Characterization of Deadlocks in Irregular Networks," *Proceedings of the International Conference on Parallel Processing*, pp. 75-84, Sep. 21-24, 1999.

[45] A. A. Chien, "A Cost and Speed Model for k-ary n-Cube Wormhole Routers," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 2, pp. 150-162, Feb. 1998.

[46] P. Pacheco, *Parallel Programming with MPI*, Morgan Kaufmann, San Francisco, CA, 1997.