A Fast Tree-Based Barrier Synchronization without Nonmember Interference on 2-D Meshes

Sangman Moh[†], *Chansu Yu*[†], *Hee Yong Youn*[†], *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, youn, dshan, dlee}@icu.ac.kr

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

Abstract

Message-passing multicomputers usually employ point-to-point direct networks for better scalability. Cooperating processes in different nodes of multicomputers exchange messages through the network, and thus 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 Meshes* (*BTM*) to minimize the barrier synchronization latency for two-dimensional (2-D) meshes. The proposed BTM scheme has two distinguishing features. First, the synchronization tree is 4-ary. The synchronization latency of the BTM scheme is asymptotically $\Theta(\log_4 n)$ while that of the fastest scheme reported in the literature is bounded between $\Omega(\log_3 n)$ and $O(n^{1/2})$, where n is the number of member nodes. Second, the construction of a BTM and synchronization operations do not interfere with nonmember nodes, which further reduces the synchronization latency. Extensive simulation study shows that, for up to 64×64 meshes, the BTM scheme results in about $40 \sim 70\%$ shorter synchronization latency, and it is more scalable than conventional schemes.

Index terms: Parallel/distributed algorithms, routing algorithms, barrier synchronization, communication latency.

1 Introduction

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. For example, `MPI_Barrier` routine defined in Message Passing Interface standard [1] blocks the calling process until all members in the same process group call the routine. Barrier synchronization is not only a fundamental and frequently used operation in parallel computing systems but also one of the basic synchronization primitives. For example, other collective operations such as `gather` and `reduce` can be regarded as special cases of barrier operation. 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. After 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 a barrier operation is to have multiple point-to-point messages sent from the root to the member nodes, but the performance can be significantly improved by reducing either the number of messages or the latency of each synchronization message. The number of messages can be reduced by combining them[1] as in Hamiltonian Path (HP)-based [2], Base Routing-Confirmed Path (BRCP)-based [3], and Collective Synchronization (CS) tree-based [4] schemes. Since a combined message needs to be delivered to multiple destinations, it has to carry multiple addresses as in HP and BRCP schemes. In the CS scheme, messages do not carry the destination addresses. Instead, at the creation time of a process group, the routers at both the member and the nonmember nodes along the path are properly set up. The resultant short synchronization message fits into a small buffer inside a router, which in turn allows simple deadlock solution.

The CS scheme minimizes the routing steps for a barrier operation by constructing a combining tree for barrier synchronization messages. Intuitively, it requires shorter routing steps to reach the destinations compared to the path-based HP or BRCP schemes. This is mainly due to the fact that the time complexity of tree-based schemes is $O(\log n)$ whereas that of path-based schemes is $O(n)$, where n is the number of member nodes. One drawback, however, is the increased overhead for constructing the CS tree. Furthermore, intermediate nonmember nodes are involved in the barrier synchronization in the sense that nonmember as well as member nodes have to participate in building the CS tree and execute the corresponding barrier operation. The irregular structure and nonmember intervention make the CS tree construction algorithm complex, and consequently result in long synchronization latency.

In this paper, we propose a *Barrier Tree for Meshes* (*BTM*) to further improve the performance of a barrier synchronization on 2-D mesh networks. It is a tree-based combining scheme with the destination addresses embedded in the routers. The proposed approach reduces the number of messages and routing steps, and provides deadlock avoidance. It differs from the conventional schemes in several aspects. First, the synchronization tree is 4-ary, which further reduces the tree height and thus reduces the routing latency by decreasing the hop counts. BTM is systematically constructed by first dividing the 2-D mesh network into four *quadrants* around the root node. This procedure is then recursively applied to construct the rest of the tree. The synchronization latency of the BTM scheme is $\Theta(\log_4 n)$ while that of the fastest scheme (CS tree) reported in the literature is bounded between $\Omega(\log_3 n)$ and $O(n^{1/2})$. Second, the construction of a BTM and the synchronization operation do not involve nonmember nodes. Direct consequences of this are simpler router design, smaller initial setup time, and more importantly no side effect due to inter-group interference. In BTM, a synchronization message also traverses nonmember nodes. However, unlike the CS tree, messages are simply forwarded to the next member node without

[1] Synchronization messages may be combined in either hardware or software. Our discussion in this paper is restricted to hardware-supported barriers because they are usually an order of magnitude faster than software barriers [5].

requiring any lookup at the barrier registers of the router. Thus, the routing delay across a nonmember node in BTM is only a small fraction of that in the CS tree. Extensive computer simulation for up to 64×64 meshes shows that the BTM scheme results in about $40 \sim 70\%$ shorter synchronization latency, and it is more scalable than the conventional schemes.

The rest of the paper is organized as follows. The proposed tree-based barrier synchronization mechanism and the corresponding router operation are presented in the following section. Section 3 analyzes the characteristics of the BTM scheme and compares it with conventional schemes. Using simulation, the performance of the BTM scheme is presented in Section 4. Conclusions and future works are discussed in Section 5.

2 Barrier Tree for Meshes (BTM)

The proposed BTM scheme assumes a wormhole-capable [7, 8, 9] 2-D mesh network, where a router is connected to the local node via a pair of input and output channels [6]. Hardware support for barrier synchronization is provided using barrier registers within the routers. Similar concept has been assumed in most hardware-supported synchronization schemes [2, 3, 4]. We assume each barrier register can hold an entire synchronization message. This can be justified by the fact that a synchronization message is very short and can be fixed in length since it does not need to carry multiple destination addresses. Detailed usage and format of the barrier registers will be presented in the following subsections.

2.1 Construction of a BTM

A BTM is constructed in a recursive manner. The algorithm starts by partitioning the 2-D mesh into four disjoint submeshes (or quadrants), denoted by Q_{+X}, Q_{-X}, Q_{+Y}, and Q_{-Y}, around the chosen root node. Then, for each quadrant, a local root node is chosen and the quadrant is partitioned again into four submeshes around the local root node. The recursive partitioning continues until there remains only one node (*i.e.*, leaf node) in each submesh.

Figure 1 shows a BTM at the distribution phase involving 14 member nodes. The root node is located at (4, 4), and the four children of the root are chosen as (6, 7), (1, 6), (2, 4), and (6, 0). The upper-left quadrant (Q_{+Y}) is partitioned again into four submeshes around the local root (1, 6). Since each of the four submeshes contains only one node, no further partitioning is required and thus the four nodes (2, 7), (1, 5), (0, 5), and (0, 7) become the children of (1, 6). The distribution message follows the arrows.

Root Node and Four Quadrants

The root node of the BTM is chosen as the one at the center of the member nodes. This is done by obtaining the average of respective x and y addresses of all the member nodes. In the example shown in Figure 1, it is (3.36, 4.42). The nearest member node to this is (4, 4), and thus it is selected as the root node. In case of a tie, a priority is given to the one from $+X$, $+Y$, $-X$ and then $-Y$.

Four quadrants of a 2-D mesh network with a vertex set V and a root node r are disjoint submesh networks with vertex sets $Q_{+X}(V,r)$, $Q_{-X}(V,r)$, $Q_{+Y}(V,r)$, and $Q_{-Y}(V,r)$. For instance, $Q_{+X}(V,r)$ can be defined as $Q_{+X}(V,r) = \{(x,y) \mid (x,y) \in V,\ x > x_r,\text{ and } y \geq y_r\}$. As mentioned earlier, once a root node and four quadrants are decided, the same procedure is recursively applied to each quadrant.

Setting up Barrier Registers

BTM is essentially embedded in the barrier registers of the routers of the participating member nodes. When a BTM is constructed, addresses of four children (C_{+X}, C_{-X}, C_{+Y}, and C_{-Y}) as well as the parent (P) of a member node are stored in a barrier register as shown in Figure 2.

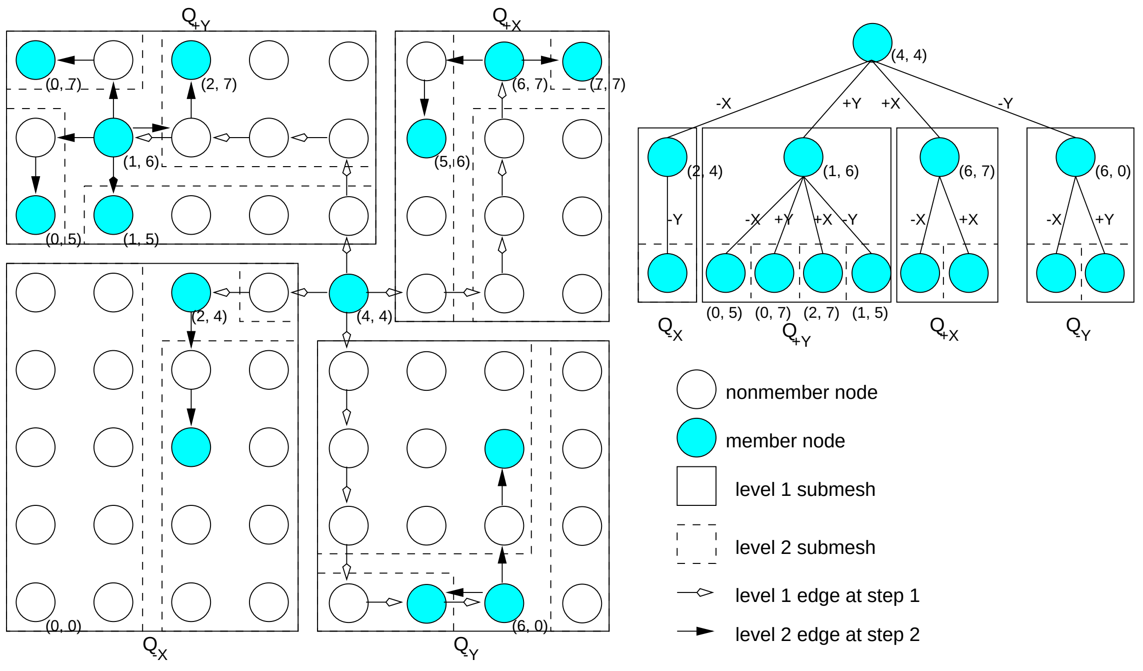

Figure 1: A BTM and its four quadrants (14 member nodes with the root at (4, 4)).

Here `GID` field identifies the particular process group, and the R bit represents the routing scheme used (X-Y or Y-X routing) when a member node sends a synchronization message to its parent in the reduction phase. This will be explained in detail in Subsection 2.2 when the router operation is discussed.

A_{+X}, A_{-X}, A_{+Y}, A_{-Y}, and message fields are used when the synchronization message is processed. For example, A_{+X} indicates whether a reduction message has arrived from a child node C_{+X} during the reduction phase. If there are more barriers than available registers in the router, some registers can be mapped onto the node memory to make room for new barriers.

Group Id (GID)	Parent (P)	R	C_{+X}	C_{-X}	C_{+Y}	C_{-Y}	A_{+X}	A_{-X}	A_{+Y}	A_{-Y}	Message

R: Routing scheme to parent	C_{+X} : Child on +X edge	A_{+X} : Arrived from C_{+X}
0: X-Y routing	C_{-X} : Child on -X edge	A_{-X} : Arrived from C_{-X}
1: Y-X routing	C_{+Y} : Child on +Y edge	A_{+Y} : Arrived from C_{+Y}
	C_{-Y} : Child on -Y edge	A_{-Y} : Arrived from C_{-Y}

Figure 2: Structure of a barrier register.

Algorithm 1 describes the procedure for setting up the barrier register within the routers. Every member node, denoted by m, runs the distributed algorithm at the process group creation time. It traverses from the root downward the BTM until a member node finds itself as the root of a quadrant. The algorithm then finds at most four children nodes of the member node and stores the GID, the parent, the routing scheme to the parent, and the addresses of four children into a barrier register in the router.

Algorithm 1: Setup_Register(r, m, `GID`)

1. Around the root node r, partition the mesh into four quadrants, Q_{+X}, Q_{-X}, Q_{+Y}, and Q_{-Y}.
2. If the member node m is equal to the root node r, jump to step 4. Otherwise, determine which quadrant (Q) the member node m belongs to.
3. Determine the local root of the quadrant Q, and set the new root as r and the previous root as parent (P). Go to step 1.
4. Determine the four root nodes of the four quadrants and define them as the four children nodes of the member node m, and write GID, parent (P), and four children into a barrier register. Set the R bit to X-Y routing if the member node belongs to Q_{+X} or Q_{-X} of the parent node; otherwise, set the R bit to Y-X routing.

Note that intermediate nonmember nodes need not allocate any resource for the barrier synchronization. Only the member nodes are involved in the router setup operation. Moreover, no inter-node communication is required during BTM construction, and thus the overhead of constructing BTM is minimized compared to the conventional schemes. Once a member node is determined to be a root of a subtree, it is never examined again during the tree construction. The tree is also constructed only downward from the root node to leaf nodes. Therefore, no cycles exist in the constructed tree, and the algorithm establishes a 4-ary barrier synchronization tree containing all the member nodes.

2.2 BTM Operations

Synchronization Message
As described earlier in this section, it is assumed that a barrier register can hold an entire synchronization message. Figure 3 shows the format of a synchronization message, which contains message type, group identifier, single destination address, and small synchronization data. For a 32×32 mesh, for example, a synchronization message consists of at most three bytes, *i.e.*, 2-bit message type for up to four message types, 8-bit group identifier for at most 256 different groups, 10-bit destination address, and at most 4-bit synchronization data for some additional information if any.

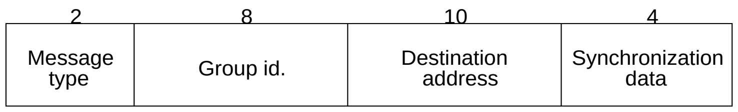

Figure 3: Format of a barrier synchronization message.

Routing and Router Operations
The BTM routing is based on a modified simple *dimension order routing*, *i.e.*, a modified version of X-Y routing [8, 10]. A routing path of BTM messages is either X-Y path where the message travels along X-axis first and then along Y-axis, or Y-X path where it travels along Y-axis first and then along X-axis. For the reduction phase, if a local member node is C_{+X} or C_{-X} of its parent, then X-Y routing is used; otherwise, Y-X routing is used. For the distribution phase, X-Y routing is used when a message is sent to the child in C_{+X} or C_{-X}; otherwise, Y-X routing is used.

As described in Section 1, a barrier synchronization is split into two phases – reduction and distribution. During the reduction phase, reduction messages are received from, at most, four incoming links from the children nodes, and one of the reduction messages (*e.g.*, the last arriving message) is forwarded to the parent node. During the distribution phase, a distribution message is replicated at intermediate member nodes and forwarded to, at most, four outgoing links to the children nodes. Since the leaf nodes do not have children, they perform neither the message reduction nor message replication operation. The root node is the final destination for reduction messages and also the original source of distribution messages.

3 Characteristics of BTM

The characteristics of BTM are analyzed in this section. We first discuss the issue of tree height and the deadlock problem, and then finally provide a comparison with other schemes.

3.1 Tree Height of a BTM

The height of a tree is the most significant parameter in terms of communication performance of any tree-based communications. The theorem below analyzes the height of a BTM for a complete barrier that occurs when all the nodes in a mesh are members in a group. (See [11] for a complete proof.) According to the theorem, the height of a BTM with n member nodes is $O(\log_4 n)$, and thus the associated routing latency has an upper bound of $O(\log_4 n)$. Even in the best case, the height cannot be lower than $\lceil \log_4 n \rceil$ because the outgoing degree of every member node is at most 4. Hence, the lower bound of the height of BTM can be represented as $\Omega(\log_4 n)$, and thus the associated routing latency has a time complexity of $\Theta(\log_4 n)$.

Theorem 1: *For a $k \times k$ mesh network, the height, h_k, of a BTM is given by $h_k \leq \log_4 k^2 + 1$, where $k = 2^i$ for some positive integer i.*

3.2 Deadlock Freedom

Wormhole routing is assumed to be used in BTM since it reduces the communication latency by pipelining the message transfer over a number of channels along its path. A major issue with the wormhole routing is deadlock. When the path of a message is blocked, the message head as well as the rest of the message are blocked, holding the buffers and channels along the path. Deadlock can occur if these blockages create a cyclic dependency. However, if the message size is small enough, deadlock could be easily avoided by holding the entire message in the router. In BTM, the synchronization messages only carry the immediate node address, and thus the message lengths are identical and very small. In BTM, synchronization messages are only 3 bytes for a 32×32 mesh as in Figure 3, and therefore input buffer can be limited to only a few flits.

The basic technique for proving that a network is deadlock-free is to identify the dependences that can arise between channels as a result of message movement, and to demonstrate that no cycles exist in the resulting channel dependence graph [10]. This implies that no traffic patterns can lead to deadlock. These traffic patterns include a barrier synchronization, multiple concurrent synchronizations, and a mixture of synchronization messages and normal messages. For the BTM scheme, the proof for deadlock freedom is quite simple because an entire synchronization message can be stored in a storage (barrier register) in the router. See [11] for a complete discussion on how messages incurred by the above three cases do not create a deadlock situation.

3.3 Comparison of Characteristics

In this subsection, BTM is compared with other barrier synchronization schemes, namely HP, BRCP, and CS schemes. The comparisons are summarized in Table 1, where a number of important factors are considered for comparative evaluation. As can be seen from the table, the proposed BTM possesses favorable characteristics for all the factors studied, which results in a significantly improved performance as will be shown in the following section.

For complete barriers on an 8×8 mesh where all 64 nodes in the mesh are member nodes, the height of BTM is 4 whereas that of the CS tree is 8 (because of a chained critical path). That is, the upper bound height of BTM is 4 ($= \log_4 64 + 1$), but that of the CS tree is 8 ($= \sqrt{64}$). It is simple to see that the upper bound on routing latency of the CS tree is $O(n^{1/2})$. Also note that BTM is fairly balanced while the CS tree is not. In the best case, the routing latency of the CS scheme can be represented as $\Omega(\log_3 n)$ because the outgoing degree of every branching node is at most 3 except the root node which has degree of 4.

Table 1: Characteristics of hardware-supported barrier synchronization schemes.

	HP scheme	BRCP scheme	CS scheme	BTM scheme
Initialization at group creation time				
Path/tree construction	Centralized at the root	Centralized at the root	Distributed at all member nodes	Distributed at all member nodes
Tree skewness	–	–	Skewed	Balanced
Router setup	At member nodes	At member nodes	At member and nonmember nodes	At member nodes
Initialization overhead	Small	Small	Large (unicast messages to nonmember nodes)	Small
During a barrier operation				
Multiple destination addresses	Embedded in message	Embedded in message	Embedded in routers	Embedded in message and routers
Synchronization message	Long (multiple destination addresses)	Long (multiple destination addresses)	Short (no destination address)	Short (single destination address)
Number of start-ups	2 (one for reduction and one for distribution)	6 (3 for reduction and 3 for distribution)	2 (one for reduction and one for distribution)	2 (one for reduction and one for distribution)
Routing algorithm	Hamiltonian path	Basic routing schemes	X-Y routing	X-Y (Y-X) routing
Upper bound of routing latency	$O(n)$	$O(n)$	$O(n^{1/2})$	$O(\log_4 n)$
Lower bound of routing latency	$\Omega(n)$	$\Omega(n)$	$\Omega(\log_3 n)$	$\Omega(\log_4 n)$
Nonmember intervention	Message-passing	Message-passing	Tree buildup, register lookup and update, and message replication	Message-passing
Primary weakness				
Primary weakness	Long path	More number of startups	Nonmember node intervention and initialization overhead	Hardware complexity at the router

As briefly discussed earlier, one of the main differences between BTM and the CS tree lies in the embedding of destination addresses. The CS scheme embeds them at the routers, and the routers need to differentiate a synchronization message from an ordinary one due to the different routing mechanisms used. An ordinary message is routed according to the destination address in the message, but routing of the synchronization message follows the direction embedded in the router itself. The consequences are increased overhead at the router for processing a barrier, and no support for adaptive routing of synchronization messages due to the predetermined routes at setup time. BTM alleviates the functional complexity at the router by including only one immediate destination address in the message and embedding rest of them in the routers. Synchronization messages are treated as ordinary messages by routing them with the destination address in the message. Adaptive routing is also possible with BTM. The most obvious advantage of BTM is the reduced synchronization latency due to the reduced tree height.

4 Performance Evaluation

In this section, the performance of the proposed BTM scheme is evaluated and compared to the CS scheme using simulation.

4.1 Simulation Environment

In our simulation study, the member nodes were picked randomly and all the members are assumed to arrive at a barrier at the same time. 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, network traffic incurred by the barrier synchronization was also investigated. This is determined by the number of links (hops) traversed by the synchronization messages during a barrier operation.

Since synchronization messages do not need any data flits, the communication time of a message in wormhole-routed systems can be approximated to $t_s + d \cdot t_p + (d+1) \cdot t_r$, where t_s is the startup time, t_p is the propagation delay per hop, t_r is the average delay at each router, and d is the distance between the source and destination nodes in a communication. The routing delay, t_r, at a member node can be quite different from that at a nonmember node. On a barrier synchronization message, the router of a member node has to look up the content-addressable barrier registers using `GID`, mark (updates) its arrival at the corresponding barrier register, find the next destination address, merge or replicate the message, and inform the local processor, if necessary. A router recognizes itself as a nonmember node by matching the destination address of the synchronization message. It then simply forwards the message to the proper output channel like an ordinary message. Thus, to distinguish its cost, we denote the routing delay at a member node as t_{rm} and that at a nonmember node as t_{rn}. The synchronization latency, t_b, of the proposed BTM is then represented by

$$t_b = 2 \cdot \{t_s + d \cdot t_p + (d - h) \cdot t_{rn} + (h + 1) \cdot t_{rm}\},$$

where h is the height of BTM.

Since nonmember as well as member nodes participate in barrier operations with the CS tree, the synchronization latency, t_c, of the CS scheme can be represented by

$$t_c = 2 \cdot \{t_s + d \cdot t_p + (d + 1) \cdot t_{rm}\}.$$

The startup time, t_s, is assumed to be $1 \sim 10$ *µsec*, and the link propagation delay, t_p, is assumed to be $5 \sim 15$ *nsec* as others have done [3, 4]. For example, the Cray T3D with PVM is quoted as having a startup time of 3 *µsec*, whereas an IBM SP-2 with MPI has a startup time of 35 *µsec* [13]. The startup time, t_s, includes the software overheads for allocating buffers, copying messages, and initializing the router and DMA. Chien [12] analyzed the router delay for various routing algorithms using a 0.8 micron gate array technology. Based on that study and current VLSI technology, the router delay at a nonmember node t_{rn} is assumed to be $5 \sim 15$ *nsec*. The router delay at a member node t_{rm}, which includes several steps of operations described above, is assumed to be $30 \sim 60$ *nsec*.

4.2 Simulation Results and Discussion

We present the simulation results for two different system configurations of 32×32 and 64×64 meshes. Two important performance metrics, average tree height and synchronization latency, are presented first. Next, the network traffic and the effect of the routing delay are analyzed. Here for each parameter set, 100 simulation runs were executed, and the results were then averaged. In most cases, a very small variance was observed.

Figure 4 shows the average tree height for BTM and the CS tree. Nonmember nodes are not counted in both cases. For the CS tree, average tree height increases linearly with the group size. However, since nonmember nodes have almost the same overhead as member nodes, the corresponding synchronization latency will be saturated with smaller group size as evident in Figure 5. For BTM, since the intermediate nonmember nodes are not included in the parent-child relationship, they do not influence the tree height. As can be seen from Figure 4, the tree height of BTM is almost independent of the group size except for very small group sizes. In a 64×64 mesh, the group size can be more than 1024 nodes. Although not shown in the graph, when the group size is increased up to 4096 nodes, the tree height of a 64×64 mesh converges to 7 and 64 for BTM and the CS tree, respectively. It is obvious that BTM is more scalable than the CS tree.

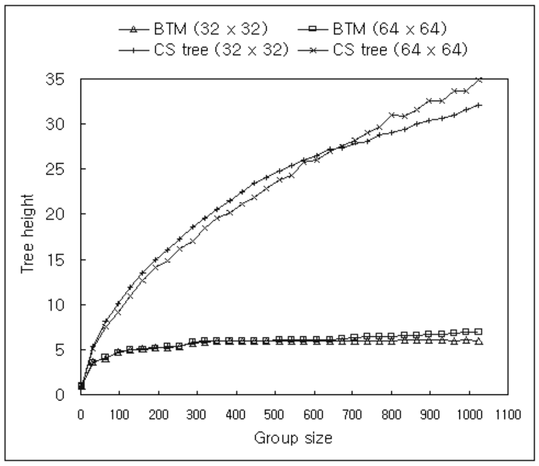

Figure 4: Average tree height.

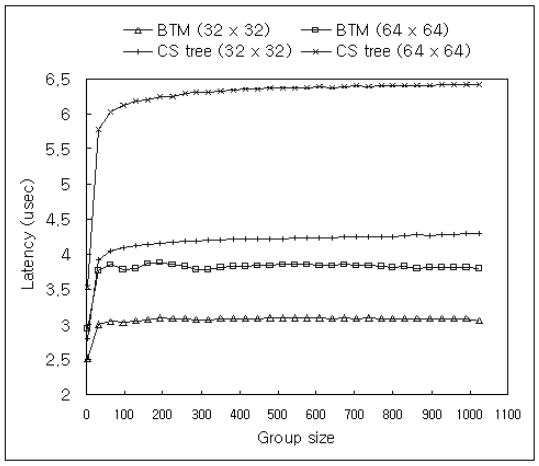

Figure 5: Synchronization latency.

Figure 5 shows the synchronization latency, where t_s, t_p, t_{rn}, and t_{rm} are assumed to be 1 μsec, 5 $nsec$, 5 $nsec$, and 30 $nsec$, respectively. The synchronization latency of the BTM scheme is significantly lower than that of the CS scheme, and again it is almost independent on the group size. This is mainly due to the fact that the tree height of BTM is bounded by $\log_4 k^2 + 1$ on a $k \times k$ mesh. The performance improvement is more substantial as the size of network increases. For instance, for the group size of 1024, the BTM scheme is faster than the CS scheme by factors of 1.4 and 1.7 for 32×32 and 64×64 meshes, respectively. The proposed BTM scheme is clearly more scalable than the CS scheme.

In general, the BTM scheme is expected to have more network traffic than the CS scheme. This is because more nonmember nodes are traversed in a barrier operation. Even though they do not participate in building and processing of the barrier, simple forwarding of the synchronization messages incurs more network traffic. Here the network traffic is measured as the number of round-trip hops (links) traversed during both reduction and distribution phases. As shown in Figure 6, the network traffic of the BTM scheme is larger than that of the CS scheme. As the group size increases, the network traffic is also increased for both schemes because the trees have more nodes and edges. However, the impact of the increased network traffic on the overall performance is minimal. During a barrier synchronization, all the participating nodes wait for the completion of the synchronization and do not issue any communication request, which results in low-loaded network. Furthermore, since the barrier synchronization message is very short in length, it does not cause any hot spots.

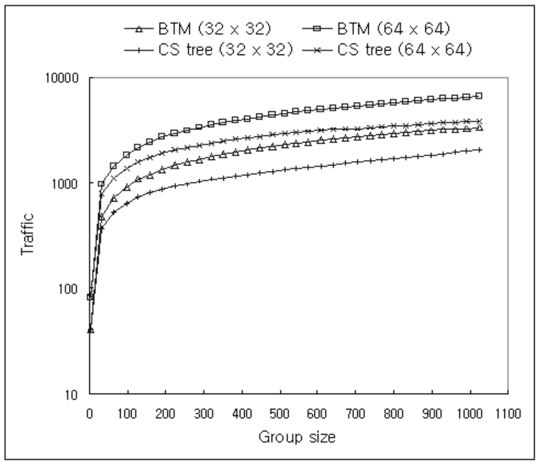

Figure 6: Network traffic.

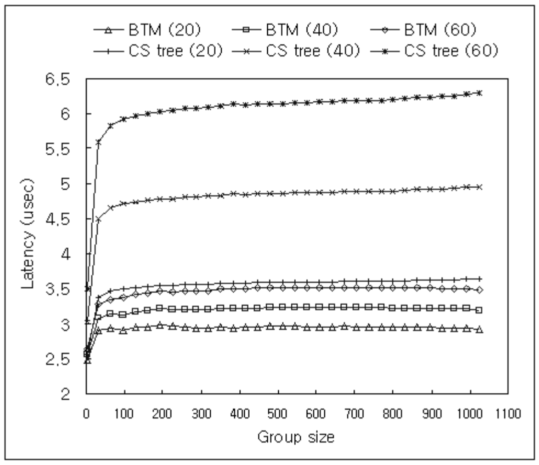

Figure 7: Effect of the routing delay at a member node, t_{rm}.

We also studied the impact of variations in the routing delay at a member node (t_{rm}) on the performance. On a 32×32 mesh, we consider three different cases of t_{rm}; 20 *nsec*, 40 *nsec*, and 60 *nsec*. This parameter is shown in the parenthesis of labels in Figure 7, where t_s, t_p, and t_{rn} are 1 μsec, 5 *nsec*, and 5 *nsec*, respectively. Even in this experiment, the BTM scheme outperforms the CS scheme for all the three cases of widely varying delay parameter values. As t_{rm} increases, more performance gain is achieved. For instance, for the group size of 1024 on a 32×32 mesh, the BTM scheme is faster than the CS scheme by factors of 1.3, 1.6, and 1.8 for the three cases of t_{rm}, respectively. This property is very important, especially as more communication functions are embedded in a router and the corresponding router operations become more complicated. The proposed scheme consistently outperforms the CS scheme for a wide range of mesh sizes and operating conditions.

5 Conclusions

In this paper, we proposed a fast tree-based barrier synchronization scheme for 2-D meshes. The proposed BTM scheme has two distinguishing characteristics compared to other conventional schemes. First, the synchronization messages pass through the nonmember nodes without interference resulting in negligible delay. Thus, the routing latency is bounded by the tree height. Since the tree constructed by the BTM setup algorithm `Setup_Register()` (see Section 2) is 4-ary, the complexity of routing latency is $\Theta(\log_4 n)$ while that of the fastest scheme (CS tree) reported in the literature is bounded between $\Omega(\log_3 n)$ and $O(n^{1/2})$, where n is the number of member nodes. Second, initial setup overhead of constructing BTM is very low. At the tree creation time, all the member processes run the setup algorithm and construct the tree independently without any message exchanges between them or between nonmember nodes.

We have simulated and evaluated the performance of the proposed scheme. Performance effect of various parameters such as tree height, synchronization latency, network traffic and the effect of routing delay was studied. According to the simulation results, compared to the CS scheme, the BTM scheme reduces synchronization latency by 40% and 70% for 32×32 and 64×64 meshes, respectively.

We are currently working on extending the BTM approach to other interconnection topologies such as k-ary n-cubes and more importantly, irregular and arbitrary networks used in switch-based cluster systems. Our preliminary simulation study shows that the barrier tree method is a viable solution for irregular networks as well. Future works include the application of BTM to other collective communications, such as multicast or total exchange. It would also be an interesting subject to consider how BTM can be applied to a dynamic environment caused by load balancing and node/link failures.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

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

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

[11] S. Moh, C. Yu, H. Y. Youn, D. Han, B. Lee, and D. Lee, "A Fast Tree-Based Barrier Synchronization without Nonmember Interference on 2-D Meshes," *Technical Report*, School of Engineering, Information and Communications University, July 2000.

[12] 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.

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