Efficient Collective Communication on Heterogeneous Networks of Workstations [1]

Mohammad Banikazemi, Vijay Moorthy and Dhabaleswar K. Panda

Dept. of Computer and Information Science
The Ohio State University, Columbus, OH 43210-1277
Tel: (614)-292-5199, Fax: (614)-292-2911

Contact Author: Dhabaleswar K. Panda

Abstract:

Networks of Workstations (NOW) have become an attractive alternative platform for high performance computing. Due to the commodity nature of workstations and interconnects, the NOW environments are being gradually redefined as *Heterogeneous Networks of Workstations* (HNOW) environments. This paper presents a new framework to implement collective communication operations (as defined by the Message Passing Interface (MPI) standard) efficiently for the emerging HNOW environments. We first classify different types of heterogeneity in HNOW and then focus on one important characteristic: *communication capabilities of workstations*. By taking this characteristic into account, we propose a new *Speed-Partitioned Ordered Chain* (SPOC) approach to order the participating nodes of a given collective communication operation in an optimal fashion. Such an ordering allows flexibility to implement collective communication operations with lower latency. Using the new approach, a set of tree-based algorithms are proposed for frequently used collective communication operations such as broadcast, multicast, and barrier synchronization. All these algorithms are compared with the standard MPI implementations on experimental as well as simulated testbeds. On a 16-node existing HNOW environment with SGI workstations and ATM interconnection, our approach reduces the latency of broadcast, multicast, and barrier synchronization operations by a factor of up to 1.4 compared to the existing MPICH implementation. On a 64-node simulated testbed, our approach promises to reduce the latency of broadcast operation by a factor of up to 4.8 compared to the MPI implementations. Improvements up to a factor of 3.3 and 2.3 are observed for multicast and barrier synchronization, respectively. Thus, these results demonstrate significant potential to be applied towards designing scalable collective communication libraries for current and future generation HNOW environments.

Keywords: Network of Workstations (NOW), Heterogeneous Network of Workstations (HNOW), Collective Communication, Broadcast, Multicast, Barrier Synchronization.

[1] This research is supported in part by NSF Career Award MIP-9502294 and NSF Grant CCR-9704512.

Contents

1 Introduction **3**

2 Characterizing Heterogeneous Networks of Workstations **4**

2.1 Major Characteristics 4

2.2 Overhead of Point-to-Point Communication under Heterogeneity 5

3 A New Framework for Collective Communication **6**

3.1 Basic Idea 6

3.2 The Speed-Partitioned Ordered Chain (SPOC) Approach 7

4 Algorithms for Collective Communication **8**

4.1 Broadcast 8

4.2 Multicast (Single and Multiple) 9

4.3 Barrier Synchronization 9

5 Performance Evaluation **10**

5.1 Experimental Results 10

5.1.1 Experimental Setup 10

5.1.2 Broadcast 10

5.1.3 Multicast 11

5.1.4 Barrier Synchronization 12

5.2 Simulation Results 13

5.2.1 Simulation Setup 13

5.2.2 Broadcast 13

5.2.3 Multicast 15

5.2.4 Barrier Synchronization 15

6 Conclusions and Future Research **15**

7 Acknowledgment **18**

1 Introduction

Networks of Workstations (NOW) are becoming increasingly popular for providing cost-effective and affordable parallel computing for day-to-day computational needs [1, 31]. Such environments consist of clusters of workstations connected by *Local Area Networks* (LANs). Hardware and software LAN technology was not initially developed for parallel processing, and thus the communication overhead between workstations can be quite high. In order to achieve performance comparable to MPP systems, many research projects are currently being undertaken in academia and industry to provide fast communication and synchronization in NOW systems. However, most of these research projects focus on *homogeneous* NOW systems where similar kinds of workstations (nodes) are connected over a single network architecture. Some popular network architectures used in the current NOW environments are Ethernet, ATM [2], Myrinet [4], FDDI, Cut-through switches like SP-2 [30], and Fiber-Channel.

Due to the commodity nature of workstations and networking equipment, LAN environments are gradually becoming *heterogeneous*. The capability of a LAN environment to incrementally expand by incorporating new generations of workstations and network architectures over a period of time is also forcing this trend. Such heterogeneity may get reflected in terms of varying speed and communication capability of workstations, coexistence of multiple network architectures, availability of alternative communication protocols, and availability of specialized support for communication and synchronization over a set of workstations. Thus, in today's networked high performance computing environment, heterogeneity is common and its extent will continue to grow over the years. This is forcing the NOW environments to be gradually redefined as *Heterogeneous Networks of Workstations* (HNOW) environments.

A portable parallel programming environment is key to the success of the NOW/HNOW paradigm. Over the last few years, researchers have developed softwares like PVM [31] and *Message Passing Interface* standards like MPI [21, 6] to provide such portability. Even though these softwares and standards do not force an application developer to understand the intricate details of the hardware, software, and network characteristics, the performance of an application in a NOW/HNOW environment heavily depends on the latency and bandwidth required for interprocessor communication and synchronization across the workstations (nodes).

The need for *collective communication* operations such as *broadcast, multicast, global reduction, scatter, gather, complete exchange*, and *barrier synchronization* arises frequently in parallel applications [27, 5, 3, 10, 19, 22, 15]. Thus, it is critical that the collective communication operations be implemented in the best possible manner (scalable as well as high performance) in a HNOW system. Recently, some projects have emphasized issues related to collective communication in NOW systems. These projects have been centered around the following interconnects: ATM [11], Ethernet [8], and Myrinet [32]. Performance of collective communication operations in NOW environments have been evaluated in [17, 24]. However, all these studies focus on only one type of interconnect in a NOW system. They also do not consider heterogeneity in workstation speeds, communication protocols, etc. Thus, the solutions derived in these research can not be directly applied to HNOW systems to obtain maximum performance.

To the best of our knowledge, the ECO [18] package has been the only effort made to consider the heterogeneity of workstations in NOW environments. ECO is built on top of PVM. It proposes heuristics to partition the participating workstations of a collective communication operation into subnetworks based on pair-wise round-trip latencies. Next, it divides the required

communication steps into two major phases: inter-subnetwork and intra-subnetwork. Different trees are used for performing collective communication operations in each of these phases. However, the proposed partitioning approach based on pair-wise round-trip latencies may result in incorrect partitioning in the presence of many factors such as background traffic and workstations with different communication capabilities. This may cause inefficient implementation of collective communication operations. This leads to the following challenges: 1) how to characterize the heterogeneity of a HNOW environment and 2) how to implement efficient collective communication on HNOW environments by exploiting one or more of the heterogeneous characteristics.

In this paper we take on these challenges. We first classify different types of heterogeneity that can exist in HNOW environments and characterize them. Then, we focus on one major characteristic, *communication capabilities of workstations*. We study the impact of this characteristic on the communication overhead of MPI point-to-point communication in a typical LAN environment consisting of heterogeneous workstations. The experimental results indicate that the communication overhead among workstations in a HNOW environment may vary as much as 5:1. Using this observation, we propose a generalized framework to implement efficient collective communications on HNOW systems. We introduce a new *Speed-Partitioned Ordered Chain (SPOC)* approach to order the participating nodes of a collective communication based on their communication capabilities. Using this framework, a set of algorithms are developed to implement frequently used collective communication operations (such as broadcast, multicast, multiple multicast, and barrier synchronization) with reduced latency. These algorithms are evaluated on an experimental testbed consisting of a cluster of (16) SGI workstations and compared with the existing MPICH implementations. Furthermore, the algorithms are evaluated on a 64-node simulated HNOW environment with different architectural characteristics. It is shown that latency of these collective communication operations can be reduced by a factor of up to 5 using the proposed framework.

The paper is organized as follows. Different types of heterogeneity are characterized in section 2. Experimental results on message initiation cost on a set of workstations are also presented. The new SPOC framework for collective communication is proposed in section 3. In section 4, algorithms for broadcast, multicast, multiple multicast, and barrier synchronization operations are developed using the SPOC framework. Performance evaluation results are presented in section 5. Finally, we conclude the paper with conclusions and future research directions.

2 Characterizing Heterogeneous Networks of Workstations

In this section we characterize factors leading to heterogeneity in HNOW systems. We show how overhead for MPI point-to-point communication can vary significantly in a HNOW system by considering only one factor - communication capabilities of the nodes.

2.1 Major Characteristics

A typical HNOW system can be characterized by the following four factors. These factors are orthogonal to each other. A typical HNOW environment can have one or more of these characteristics.

Communication capabilities of workstations (nodes): The commodity nature of a LAN environment forces workstations with different processor speed, memory speed, and network

interface support to be interconnected. These parameters reflect the hardware capabilities to send or receive a message. A node with faster processor and memory can run a communication routine faster and prepare a message to be injected in less time. Similarly, a faster network interface support can also inject a message into the network at a faster speed. Similar arguments also hold good for receiving a message. These capabilities have direct impact on communication overheads for point-to-point communication as well as collective communication.

Network architectures: A typical LAN system can have multiple networks (intra-networks) connected to each other through an inter-network. Different network architectures such as Ethernet, Fast Ethernet, ATM, Myrinet, and Fiber Channel can coexist in a LAN. Different parameters which can affect the overall performance of a LAN are: network characteristics, overall topology, connectivity (inter-network and intra-network), and communication bandwidth (intra-network and inter-network). Thus, algorithms for collective communication need to take care of these characteristics to achieve reduced latency.

Communication protocols: Current generation LANs typically support different communication protocols (such as TCP/IP, ATM [2], and FM [25]) which workstations (nodes) use to communicate. These protocols have their own characteristics and affect the latency of communication operations. Furthermore, some nodes may have access to more than one set of protocols. In this case it is crucial to choose an appropriate protocol to implement a collective communication operation with reduced latency.

Dedicated support for communication and synchronization: Since a typical HNOW environment consists of multiple subnets, there might be dedicated support for collective communication within some subnets. Examples include hardware multicasting within ATM subnet, broadcasting capability within Ethernet and Fast Ethernet, fast synchronization within SMP systems, etc. Availability of such dedicated support for communication and synchronization needs to be taken into account to implement collective communication operations with reduced latency.

All of the above factors have significant impact on the implementation of collective communication operations on HNOW systems. To illustrate this significance, in this paper, we limit our scope to the first characteristic only. Similar approach can also be used for other characteristics and we are currently working along these directions.

2.2 Overhead of Point-to-Point Communication under Heterogeneity

We present experimental results to show the effect of communication capabilities of workstations on the latency of MPI point-to-point communication. We measured round trip latency between four different pairs of workstations in a heterogeneous environment. The workstations were connected via Ethernet and used MPICH communication library[9] to communicate. Table 1 shows these results. Since the results are symmetric, values are shown for only the upper triangle entries. These values indicate how processor speed affects the time taken to transmit a message from one workstation to another. The fastest workstation we used was an HP 735 and the slowest one was a Sun 4. It can be observed that the communication startup time for a Sun 4 is around 5 times that of an HP 735.

These experimental results demonstrate that workstation speeds can have direct impact on the communication latency. Since collective communication operations involve more than one workstations the question arises whether we can use the heterogeneity to our advantage to implement the operations faster. We propose such a framework in the following section.

Table 1: Roundtrip times in microseconds between different types of workstations in a heterogeneous network. (Entries with * indicate that the sender and receiver were in different clusters.)

	HP 735	HP 715/100	HP 715/64	Sun4
HP 735	871	973	2491 *	5806 *
HP 715/100		1020	2538 *	5871 *
HP 715/64			1869	6050 *
Sun4				4196

3 A New Framework for Collective Communication

In this section we propose a framework to take advantage of heterogeneity in communication capabilities to implement a collective communication operation faster. First, we provide the basic idea behind such a framework. Next, we propose a generalized approach.

3.1 Basic Idea

As we observed earlier, various factors such as processor speed, memory speed, and network interface support affect the communication capability of a node. The above parameters can be combined together to a single parameter known as message initiation cost, t_{ini}. Let us consider a broadcast operation on an example HNOW environment consisting of eight workstations (1,2,...8). Let node 1 be the source node. Let four of these workstations (2, 3, 5, and 6) be slow ones having a message initiation cost, t^s_{ini}. Similarly, let the other four (1, 4, 7, and 8) be fast ones and have lower message initiation cost, t^f_{ini}. Based on our experimental data (shown in Table 1) , it can be observed that the t^s_{ini}/t^f_{ini} can be as large as 5. For an example quantitative evaluation, let us consider $t^f_{ini} = 100.0$ microsec and $t^s_{ini} = 200.0$ microsec, leading to $t^s_{ini}/t^f_{ini} = 2$. Because of the high value of t_{ini} (which is typical of NOW systems) let us ignore the time required for transmitting the messages in our example.

Consider a naive and simple binomial-tree-based scheme [20], based on node numbering, leads to the broadcast tree as shown in Fig. 1(a). Using this scheme the broadcast can be completed in 500.0 microsec in this HNOW environment. However, a more efficient scheme, as shown in Fig. 1(b), can be designed by considering the differences in the communication capabilities of the workstations. Here the fast workstations are used as intermediate nodes to broadcast the message faster. This scheme takes only 300.0 microsec to implement the broadcast and demonstrates a 40% improvement in broadcast latency.

It can be observed from this example that several alternatives exist to implement a broadcast communication faster by considering the communication capabilities of the participating nodes. In this example we assumed that there are only two types of workstations. The problem becomes more complex if we have multiple levels of communication capabilities. In general, the communication capabilities of the nodes in an N-node HNOW environment can be modeled as two sets of parameters: number of nodes having similar communication capabilities (N_i) and their respective message initiation costs, t^i_{ini}. Now the problem reduces to designing efficient broadcast trees using these two sets of parameters.

Similar approach can also be used for other collective communication operations. In the following subsection, we present a generalized approach to design efficient broadcast trees by

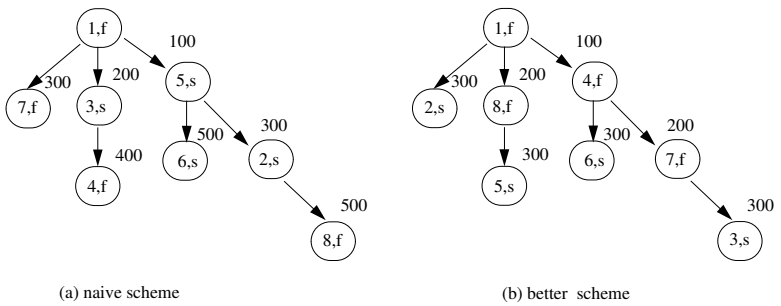

Figure 1: Different ways to implement broadcast in an eight workstation HNOW environment: (a) simple binomial tree-based scheme using node numbering and (b) an optimal scheme by considering communication capabilities of the workstations.

considering multiple levels of communication capabilities. In the next section, we use this approach to develop algorithms for frequently used collective communication operations.

3.2 The Speed-Partitioned Ordered Chain (SPOC) Approach

This approach uses information about the message initiation cost (t_{ini}) of nodes which are participating in a collective communication operation. Using this information, an efficient ordering of participating nodes is derived to implement the operation faster.

Consider a collective communication operation, say broadcast, which can be implemented by using a binomial-tree [20]. The Binomial tree algorithm is well suited for homogeneous networks. However, nodes that are higher up in the binomial tree send more messages and therefore incur more message startup overhead. Thus, if faster nodes appear higher up in the binomial tree, then the overall startup overhead incurred during the collective communication operation can be reduced. For example, in Figure 1(a) node 5 is a slow node and has 3 children while node 4 is a fast node and has no children. In Figure 1(b) node 4 has 3 children and node 5 has none. Other faster nodes have also been moved to tree nodes having more children. As a result the latency in Figure 1(b) is reduced by 200 microsec.

The problem of finding the fastest way to implement a binomial tree-based algorithm for a collective communication operation , therefore , is just finding a way to assign faster workstations to nodes in the binomial tree which have to send more messages. Let us assume that the number of participating workstations in this operation is N. We can always construct a binomial-tree consisting of N tree nodes and having a depth of $\lceil \log_2 N \rceil$. The source node of the broadcast operation remains fixed as the root of the tree, but we have the flexibility of assigning the rest of the participating nodes to any of the other tree nodes as we please.

In order to find an efficient assignment we construct the binomial tree for the given number of participating nodes and find the total number of children at each tree node. The number of children of a tree node simply specifies the number of nodes that should receive the message from that node, directly or indirectly. We sort the tree nodes from the node with the highest number of children to the node with the lowest number of children. Obviously, the root of the tree will have the highest number of children ($N-1$), but the source node is already assigned to it. We can also sort participating nodes in ascending order of message initiation cost and then assign faster participating nodes to tree nodes with greater number of children from these two sorted

lists.

It can be observed that the dimension-ordered chain [20] resulting from such an assignment will use faster nodes in the initial steps of the communication and will lead to reduced broadcast latency. We call this ordered chain of participating nodes a speed-partitioned ordered chain (SPOC). In the next section, we show how this generalized SPOC approach can be used for developing algorithms for different collective communication operations.

4 Algorithms for Collective Communication

In this section we use the SPOC approach to develop new algorithms for implementing frequently used collective communication operations such as broadcast, multicast, multiple multicast, and barrier synchronization.

4.1 Broadcast

First we consider the situation where only two classes of workstations are present in a HNOW environment. Then we extend it to include multiple classes.

Consider a HNOW system in which N workstations are participating in a broadcast. Let these workstations be divided into two classes: C_1 and C_2, with N_1 and N_2 workstations, respectively. Workstations in each class have similar communication capabilities. Let t^1_{ini} and t^2_{ini} be the message initiation cost of workstations in classes C_1 and C_2. Let us assume that $t^1_{ini} < t^2_{ini}$. As outlined in Section 3.2, a binomial-tree with N nodes can be created and an ordering based on the number of their children, in descending order, can be constructed. The source node of the broadcast is assigned to be the root of the tree. Let us consider the scenario where the source belongs to class C_1. In this case, the rest of the $N_1 - 1$ workstations are assigned to the first $N_1 - 1$ tree nodes appearing in the ordered list of the nodes right after the root. Then N_2 workstations of the class C_2 are assigned to the rest of the tree nodes. If the source node belongs to class C_2 then all N_1 workstations of class C_1 are first assigned to the tree nodes coming immediately after the root in the ordered list. Next $N_2 - 1$ nodes of C_2 are assigned to the following tree nodes in the ordered list. To implement the broadcast operation it is enough to make each workstation (except the source) in the binomial tree receive the message from its parent and forward it to all of its children.

Let us consider the case where more than two classes of workstations exist. Let the workstations be partitioned into a collection of workstation classes (C). The number of workstations belonging to each class ($N_i, 1 \leq i \leq C$) can also vary. Let t^i_{ini} be the message initiation cost for the workstations in class C_i. As outlined in Section 3.2, a binomial tree with N nodes can be created and all tree nodes be sorted based on the number of their children. Let's call this ordered list as $SL_t : \{T_0, T_1, \cdots, T_{N-1}\}$. We also sort all participating nodes (excluding the source of the broadcast) based on their t_{ini}. Let's call this list as $SL_n : \{W_1, W_2, \cdots, W_{N-1}\}$ where, $t^{W_i}_{ini} \leq t^{W_{i+1}}_{ini}$. Then we can assign workstation W_i to node T_i in the binomial tree. Workstations participating in the broadcast receive the message from their parent and forward it to their children (if they have any). The most general form of SPOC-based tree construction for broadcast, where each node has a different t_{ini}, is presented in Fig. 2.

SPOC-based tree construction for Broadcast

Input:

root: the source node of broadcast;
L_n: The list of participating nodes with their respective initiation costs (t_{ini}).

Tree construction steps:

1. $N \leftarrow number_of_nodes(L_n)$
2. $T_b \leftarrow construct_binomial_tree(N)$
3. $SL_t \leftarrow sort(list(T_b), Descending, Num_Children)$
   $// \ SL_t = \{T_i \mid 0 \leq i < N, \ num_child(T_i) \leq num_child(T_j) \ for \ i < j)$
4. $SL_n \leftarrow sort(L_n, Ascending, T_ini)$
   $// \ SL_n = \{W_i \mid 0 \leq i < N, \ t_ini(W_i) \leq t_ini(W_j) \ for \ i < j)$
5. $SL_n \leftarrow remove(SL_n, root)$
6. $assign_node_tree(item(SL_t, 0), root)$
7. for $i = 1$ to N do
   $assign_node_tree(item(SL_t, i), item(SL_n, i-1))$

Figure 2: Outline of the SPOC-based tree construction for broadcast.

4.2 Multicast (Single and Multiple)

A multicast [7, 23, 12, 13, 29, 32] operation involves only a subset of workstations in the HNOW environment. The above approach for broadcast can also be used for efficient implementation of single multicast. A binomial tree is first created with the participating nodes in the multicast. Workstations belonging to the multicast group are assigned to the nodes of the binomial tree based on their message initiation cost. Faster nodes are assigned to higher levels of the tree so that they can pass the message to their children with reduced latency.

In a multiple multicast operation, more than one group of nodes participate in the multicast [14]. Each group has its own source and destination nodes. Each node can be the source of at most one multicast while, a node can appear in more than one group as one of the destinations. An algorithm similar to the algorithm used for multicast can be used to implement the multiple multicast with the only difference being the fact that different trees are used for each multicast.

4.3 Barrier Synchronization

Barrier synchronization operation [33, 26] is implemented in two phases: gather and broadcast. The SPOC approach can be used to create a binomial tree in which each node represent a workstation participating in the barrier. Faster nodes are at the higher levels of the tree. The gather phase uses reverse-binomial tree. All nodes receive the barrier message from all of their children and then forward it to their parent. The workstation assigned to the root of the binomial tree starts the broadcast phase after receiving the barrier message from all of its children. The broadcast phase is implemented as described earlier.

5 Performance Evaluation

In this section, the performance of the algorithms developed in the previous section are evaluated and the results are presented. First, we present experimental results obtained from a cluster of SGI workstations connected by an ATM network. Due to limitations of our experimental testbed (limited number of nodes and only two levels of speed), we also present simulation results for larger number of nodes with greater variation in speed.

5.1 Experimental Results

We used an ATM network of 24 SGI workstations at the Ohio Supercomputer Center (OSC) to implement and evaluate the proposed algorithms. We compared the performance of our algorithms with those of MPICH v1.1. In the following subsections, the setup used in the experiments is explained and then the results are presented.

5.1.1 Experimental Setup

The testbed used in our experiments consisted of 24 SGI workstations with two different speeds. There were 16 slow and 8 fast nodes. This allowed us flexibility to take measurements on three different 16-node configurations with 12.5% (2), 25% (4), and 50% (8) fast nodes, respectively. Using MPICH point-to-point communication, the roundtrip latency for a short message (4-byte long) was measured to be 1380 microsec between two fast nodes and 2960 microsec between two slow nodes. The ratio of the communication capabilities of slow and fast nodes is therefore 2.15:1.0.

5.1.2 Broadcast

For measuring the broadcast latency we followed a method similar to the one used in [16]. A broadcast operation starts when the source node initiates it. It is said to be complete when all the other nodes have received the broadcast message. The broadcast latency is defined as the time elapsed between the source node initiating it and the last recipient receiving it. Measurement of broadcast latency was done in the following way. For an N node system, $N-1$ broadcasts were performed. Each time , after the broadcast, one of the $N-1$ recipients sent back an acknowledgment (instead of issuing another broadcast as in [16]). At the source node the time between initiation of the broadcast and receipt of the acknowledgment was measured. The maximum of these $N-1$ time readings corresponds to the last broadcast recipient sending back the acknowledgment. Therefore, this maximum is the sum of the broadcast latency and half of the round trip latency between the source node and the last recipient. Since the roundtrip time can be easily measured, the broadcast latency is obtained by subtracting half of the roundtrip time from the maximum reading. Furthermore, each experiment was repeated 100 times for a given source node and the minimum broadcast latency was taken into account. Each experiment was repeated for different nodes (picked randomly) acting as source node. The reported broadcast latency is the average of recorded latencies obtained from different sources.

Figure 3 shows the results. In the graphs, latencies of the MPICH implementation are referred to as MPI latencies and the latencies of our proposed algorithms are referred to as SPOC latencies. It can be observed that the proposed algorithms always perform better than

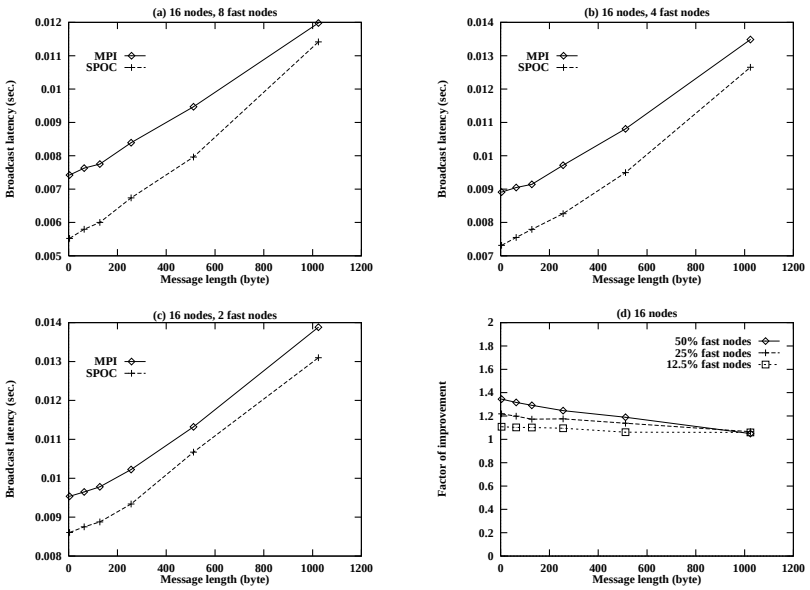

Figure 3: Broadcast latency and factor of improvement on a HNOW system with 16 nodes and different number of fast nodes.

the naive ordering scheme used in MPICH. As the fraction of fast nodes in the system increases, so does the factor of improvement. A factor of improvement of 1.4 is achieved in a system with 50% faster nodes. It can also be observed that for different fractions of faster nodes, the factor of improvement curves almost coincide beyond a certain message length due to the fact that with increasing message length, the message transmission time begins to dominate the latency. For shorter messages, where the startup overhead dominates, we get higher factors of improvement.

5.1.3 Multicast

In a multicast, the set of recipients is a subset of the set of nodes. The same procedure was therefore used to measure multicast latencies, that is, each experiment for multicast was repeated $R - 1$ times, where R is the number of recipients.

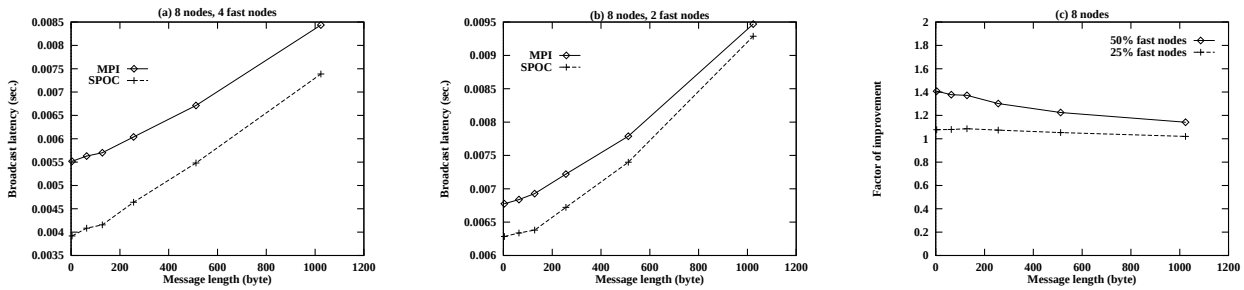

Figure 4: Multicast latency and factor of improvement on a HNOW system with 8 nodes and different number of fast nodes.

Figures 4 and 5 shows results for multicast. Characteristics similar to broadcast results are observed. The factor of improvement increases with increasing system size. Again, as message length increases, transmission time dominates and the factor of improvement decreases. For

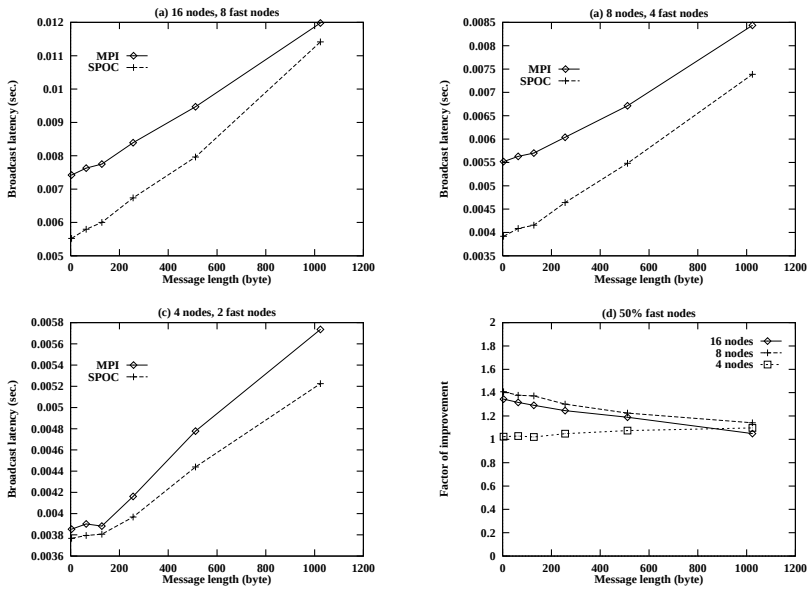

Figure 5: Multicast latency and factor of improvement on configurations with half of the nodes being fast nodes.

an 8 node multicast with 50% fast participating nodes factors of improvement of up to 1.4 are observed.

5.1.4 Barrier Synchronization

A method similar to the one used for broadcast was used for measuring the barrier synchronization latency. The barrier synchronization latency was measured as the time from the initiation of the barrier to the time when all participating nodes are done with the operation. Similar to the method used for measuring broadcast and multicast latencies, all participating nodes send an acknowledgment to a certain node one at a time. The highest measured value was recorded as the barrier synchronization latency.

Figure 6 shows the results. Barrier synchronization latency reduces as the number of fast participating nodes increases. A factor of improvement of 1.4 is achieved on a system with 50% fast nodes. Obviously, the factor of improvement increases with increasing fraction of faster nodes in the system.

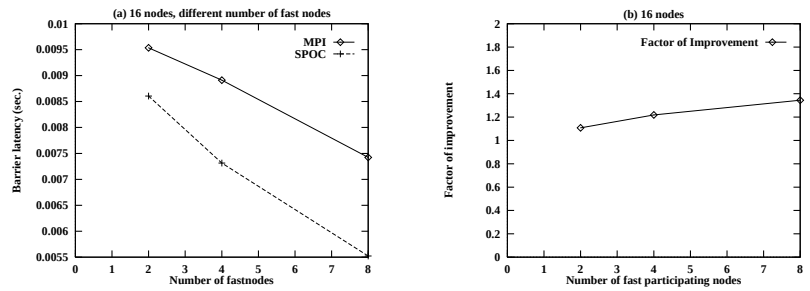

Figure 6: (a) Barrier synchronization latency for a system with 16 nodes and different number of fast nodes; and (b) factor of improvement.

5.2 Simulation Results

In this section we present our simulation results. Several experiments were performed to measure the impact of different workstation speeds, number of workstations and fraction of faster nodes on the performance of our algorithms. In the following subsections, first the simulation setup is detailed and then results for broadcast. multicast and barrier synchronization are presented.

5.2.1 Simulation Setup

We modeled a representative HNOW system where workstations are interconnected with Myrinet switches. A detailed flit level simulator (built using CSIM [28]) was used to model irregular topologies and the wormhole switching technique. A 64-node HNOW system was considered. Based on the experimental results, presented in Section 2.2, we considered the following communication startup times. Two classes of workstations were considered. The communication startup time for faster class was kept constant at t^f_{ini} = 400.0 microsec. The time for slower class (t^s_{ini}) was varied (800, 1600 and 2400 microsec). These values lead to *speed factors* (ratio of communication cost between the slow class to the fast class) of 2, 4, and 6, respectively. Such a variation helps to study a wide range of HNOW systems.

With respect to interconnection, the following parameters, representing current generation systems, were used: t_{phy} (link propagation time per byte) = 12.5 nanoseconds, t_{route} (routing delay at switch) = 500 nanoseconds, t_{sw} (switching time across the router crossbar for a flit) = 12.5 nanoseconds, t_{inj} (time to inject a flit into network) = 12.5 nanoseconds and t_{cons} (time to consume a flit from network) = 12.5 nanoseconds. For all experiments we assumed the following default system configuration: a 64 workstation system interconnected by 16 eight-port switches and a network having 50% connectivity [2].

We performed several experiments to study the impact of message length, speed factor and fraction of faster nodes. We evaluated our new algorithms (indicated as SPOC in the graphs) with naive random ordering algorithms (indicated as RO), used by the current MPI implementations. Participating nodes for a given collective communication and the network configurations were generated randomly. Latency value for each data point in the graphs was averaged over 100 experiments (10 different sets of participating nodes for each of 10 different network configurations).

5.2.2 Broadcast

For broadcast operation, we simulated our SPOC-based binomial tree algorithm and naive random ordering scheme (RO) for system configurations with 10%, 25% and 50% faster nodes, and for message lengths of 64 bytes, 1 KByte and 4 KBytes. Figures 7 and 8 show these results. It can be observed that the new algorithms always perform better than the naive schemes being currently used in the MPI implementations. The factor of improvement increases with increase in percentage of faster nodes and with increase in differences in the startup time. For a broadcast with 4 KByte message in a system having 50% faster nodes and speed factor of 6, the SPOC approach gives a factor of improvement of 4.8 over the RO scheme.

[2]Connectivity is defined as the fraction of ports in a switch which are used for interconnection with other switches [13].

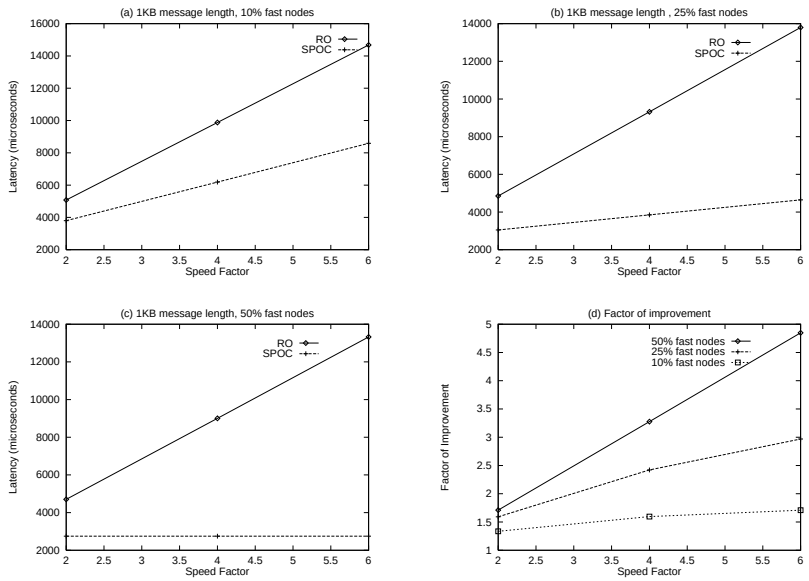

Figure 7: Impact of percentage of faster nodes on broadcast latency: (a) 10% fast nodes, (b) 25% fast nodes, (c) 50% fast nodes, and (d) factor of improvement for (a) - (c).

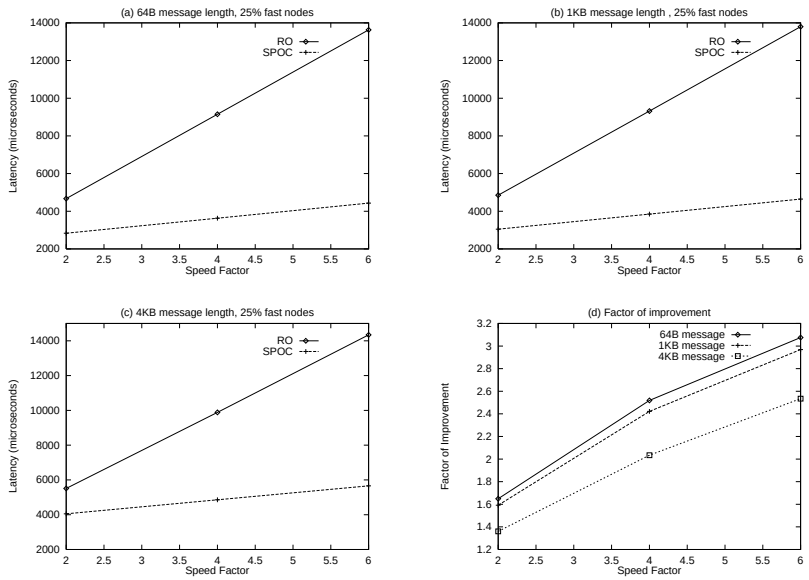

Figure 8: Impact of message length on broadcast latency: (a) 64 Bytes, (b) 1 KByte, (c) 4 KBytes, and (d) factor of improvement for (a) - (c).

5.2.3 Multicast

Figures 9, 10, and 11 show the impact of speed factor, percentage of faster nodes, and message length on the latency of single multicast with varying sizes of destination sets. The factor of improvement of the SPOC approach over the RO increases with increasing number of destinations, increasing speed factor, decrease in message length. For a multicast involving 1 KByte message on a system with 50% faster nodes and a speed factor of 4, our algorithm gives a factor of improvement of 3.3.

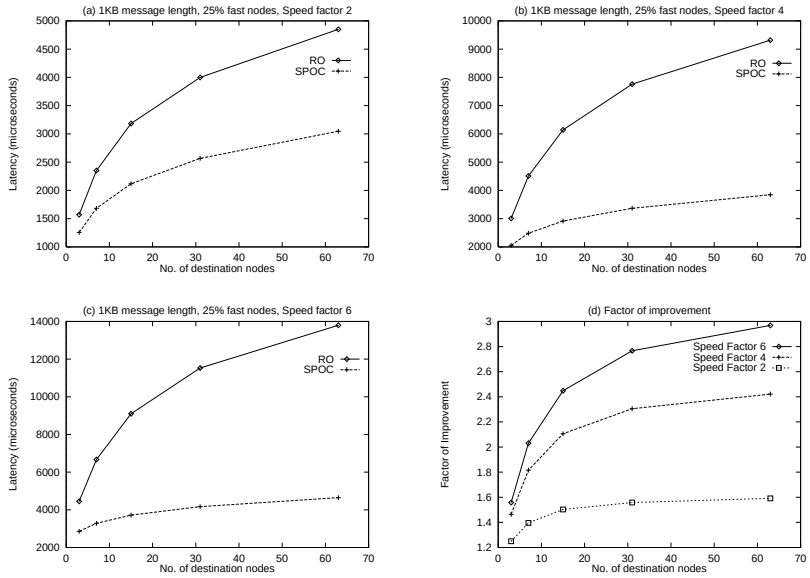

Figure 9: Impact of speed factor on single multicast latency with 25% fast nodes: (a) speed factor 2, (b) speed factor 4, (c) speed factor 6, and (d) factor of improvement for (a) - (c).

Results for multiple multicasts with varying number of sources and destinations are presented in Figure 12. It can be observed that the factor of improvement increases as more and more sources are involved in the multiple multicast. It can be observed that for a system with 50% faster nodes and a speed factor of 4, multiple multicast latency can be reduced by a factor of 2.4 using our new algorithm.

5.2.4 Barrier Synchronization

Figures 13 and 14 show results for barrier synchronization latency for varying number of participating nodes under different speed factors and percentage of fast nodes. The factor of improvement in using the new algorithm increases with increase in speed factor and percentage of fast nodes. A reduction in barrier latency by a factor of 2.3 can be observed for a system with 25% faster nodes and a speed factor of 6.

6 Conclusions and Future Research

In this paper, we have presented a new approach to implement fast collective communication in the emerging HNOW systems. Major factors in HNOW systems have been characterized. A

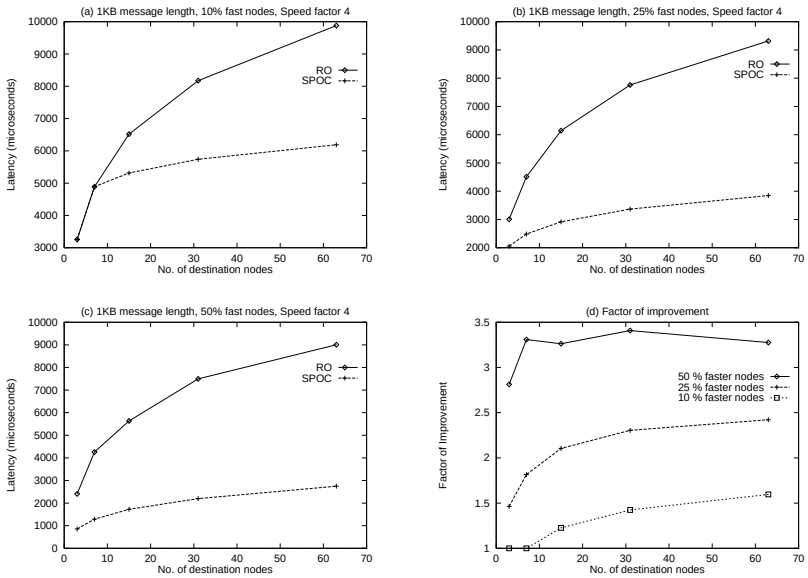

Figure 10: Impact of percentage of faster nodes on single multicast latency with speed factor 4: (a) 10% fast nodes, (b) 25% fast nodes, (c) 50% fast nodes, and (d) factor of improvement for (a) - (c).

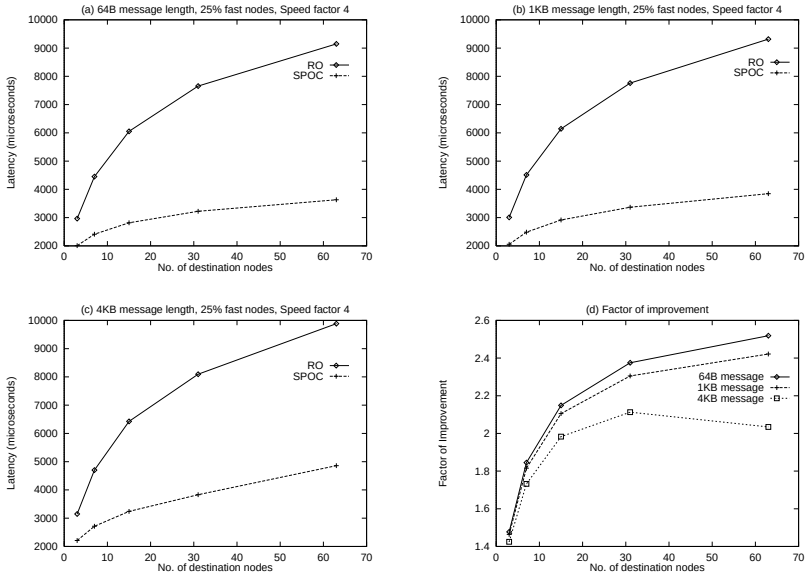

Figure 11: Impact of message length on single multicast latency with 25% fast nodes and speed factor 4: (a) 64 Bytes, (b) 1 KByte, (c) 4 KByte, and d) factor of improvement for (a) - (c).

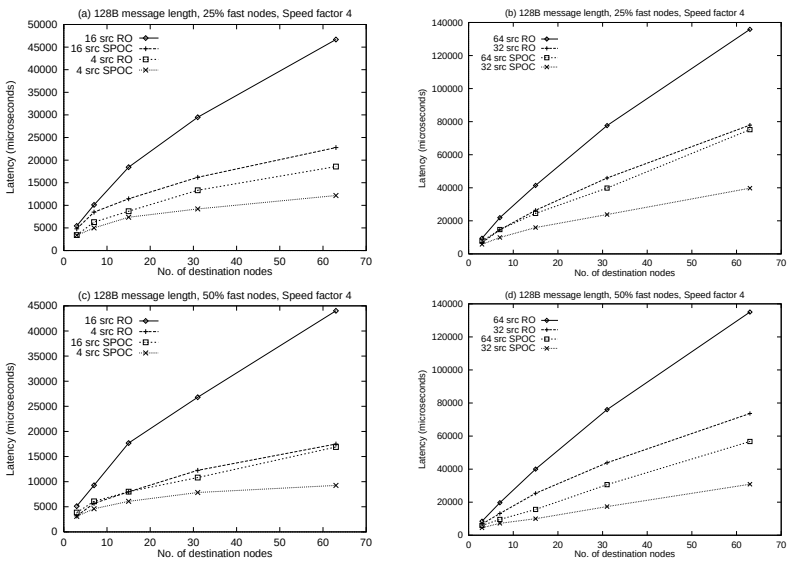

Figure 12: Impact of percentage of faster nodes on latency for multiple multicast: (a) 25% fast nodes with 4 and 16 sources, (b) 25% fast nodes with 32 and 64 sources, (c) 50% fast nodes with 4 and 16 sources, and (d) 50% fast nodes with 32 and 64 sources.

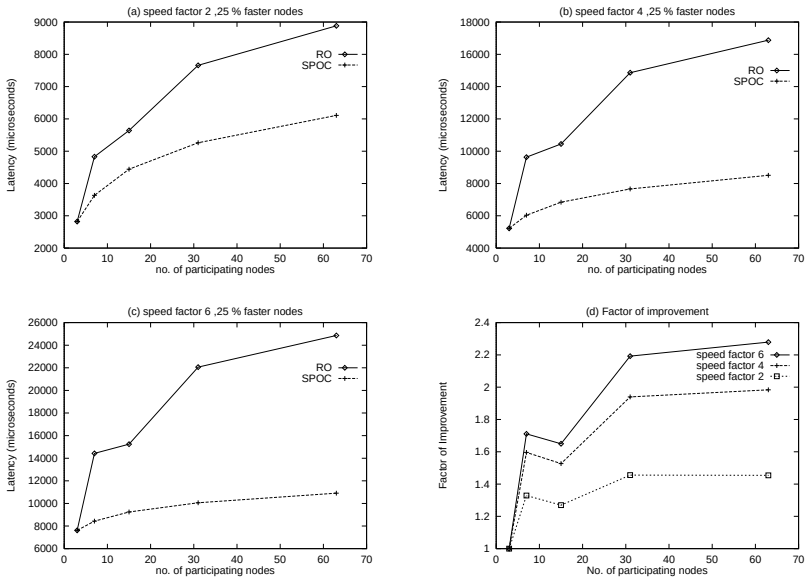

Figure 13: Impact of speed factor on barrier synchronization latency: (a) speed factor 2, (b) speed factor 4, (c) speed factor 6, and (d) factor of improvement for (a) - (c).

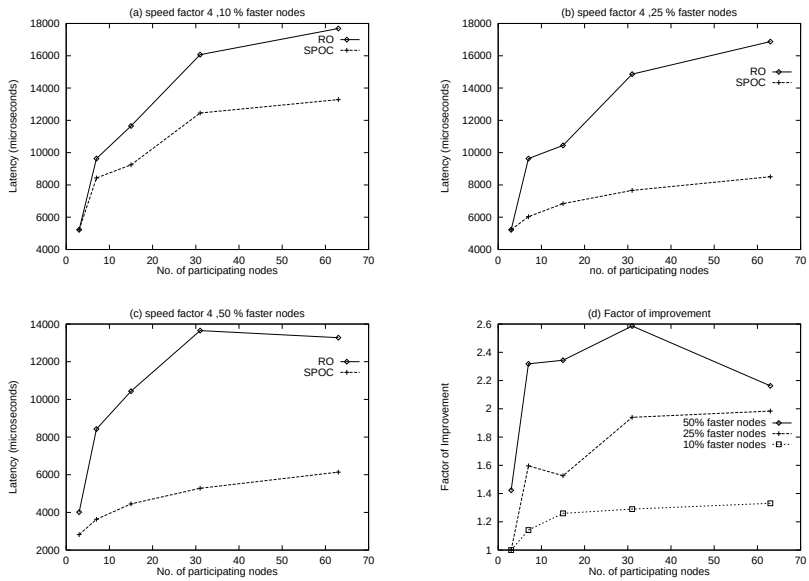

Figure 14: Impact of percentage of faster nodes on barrier synchronization latency: (a) 10% fast nodes, (b) 25% fast nodes, (c) 50% fast nodes, and (d) factor of improvement for (a) - (c).

new SPOC-based framework has been introduced to order the participating nodes of a collective communication based on their communication capabilities. Using this framework, algorithms for frequently used collective communication operations (broadcast, single multicast, multiple multicast, and barrier synchronization) have been developed. Performance evaluation of the new algorithms on a 16-node experimental testbed and a 64-node simulated testbed indicates that latency of collective communication operations can be reduced by a factor up to 5 compared to the naive algorithms used in current MPI implementations. As the commodity LAN systems go towards heterogeneity, this new approach provides a mean to harness maximum performance benefits from these systems.

In this paper, we have used only the heterogeneous communication capabilities of nodes to implement collective communication efficiently. Since binomial tree-based algorithms are not optimal, we are working on employing our approach for designing optimal tree-based algorithms for HNOW environments. We are also developing schemes to take advantage of heterogeneity in network architectures, communication protocols, and dedicated communication/synchronization units to obtain further improvements. Finally, we plan to propose a combined framework which can take advantage of all these factors and build a scalable collective communication library for HNOW systems.

7 Acknowledgment

The authors are grateful to The Ohio Supercomputing Centre for making their SGI cluster available for obtaining experimental results for this paper.

Additional Information: A number of related papers and technical reports are available electronically through the home page of *Parallel Architecture and Communication* (PAC) research group. The URL is `http://www.cis.ohio-state.edu/~panda/pac.html`. Alternatively, the

papers can be obtained by anonymous ftp from: `ftp.cis.ohio-state.edu`, `cd pub/communication`.

References

[1] T. Anderson, D. Culler, and Dave Patterson. A Case for Networks of Workstations (NOW). *IEEE Micro*, pages 54–64, Feb 1995.

[2] ATM Forum. *ATM User-Network Interface Specification, Version 3.1*, September 1994.

[3] M. Barnett, S. Gupta, D. G. Payne, L. Shuler, R. van de Geijn, and J. Watts. Interprocessor Collective Communication Library (Intercom). In *Scalable High Performance Computing Conference*, pages 357–364, 1994.

[4] N. J. Boden, D. Cohen, and et al. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, pages 29–35, Feb 1995.

[5] J. Bruck, R. Cypher, P. Elustando, A. Ho, C.T. Ho, V. Bala, S. Kipnis, and M. Snir. CCL: A Portable and Tunable Collective Communication Library for Scalable Parallel Computers. In *Proceedings of the International Parallel Processing Symposium*, 1994.

[6] J. Bruck et al. Efficient Message Passing Interface (MPI) for Parallel Computing on Clusters of Workstations. *Journal of Parallel and Distributed Computing*, pages 19–34, Jan 1997.

[7] G. T. Byrd, N. P. Saraiya, and B. A. Delagi. Multicast Communication in Multiprocessor Systems. In *International Symposium on Computer Architecture*, pages I:196–200, 1989.

[8] T. V. Eicken, A. Basu, V. Buch, and W. Vogels. U-Net: A User-level Network Interface for Parallel and Distributed Computing. In *ACM Symposium on Operating Systems Principles*, 1995.

[9] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A High-Performance, Portable Implementation of the MPI, Message Passing Interface Standard. Technical report, Argonne National Laboratory and Mississippi State University.

[10] C. Huang, Y. Huang, and P. K. McKinley. A Thread-based Interface for Collective Communication on ATM Networks. In *International Conference on Distributed Computing Systems*, 1995.

[11] C. Huang and P. K. McKinley. Communication Issues in Parallel Computing Across ATM Networks. *IEEE Parallel and Distributed Technology*, Winter:73–86, 1994.

[12] Y. Huang, C. C. Huang, and P. K. McKinley. Multicast Virtual Topologies for Collective Communication in MPCs and ATM Clusters. In *Proceedings of the Supercomputing*, 1995.

[13] R. Kesavan, K. Bondalapati, and D. K. Panda. Multicast on Irregular Switch-based Networks with Wormhole Routing. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA-3)*, pages 48–57, 1997.

[14] R. Kesavan and D. K. Panda. Minimizing Node Contention in Multiple Multicast on Wormhole k-ary n-cube Networks. In *Proceedings of the International Conference on Parallel Processing*, pages I:188–195, Chicago, IL, Aug 1996.

[15] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms.* Benjamin/Cummings, 1994.

[16] M. Lauria. High Performance MPI Implementation on a Network of Workstations. Master's thesis, Department of Computer Science, University of Illinois at Urbana-Champaign , Oct 1996.

[17] M. Lin, J. Hsieh, D. H. C. Du, J. P. Thomas, and J. A. MacDonald. Distributed Network Computing over Local ATM Networks. *IEEE Journal on Selected Areas in Communications*, 13(4), May 1995.

[18] B. Lowekamp and A. Beguelin. ECO: Efficient Collective Operations for Communication on Heterogeneous Networks. In *Int'l Parallel Processing Symposium*, pages 399–405, 1996.

[19] P. K. McKinley and D. F. Robinson. Collective Communication in Wormhole-Routed Massively Parallel Computers. *IEEE Computer*, pages 39–50, Dec 1995.

[20] 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*, 5(12):1252–1265, Dec 1994.

[21] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*, Mar 1994.

[22] P. Mitra, D. G. Payne, and et al. Fast Collective Communication Libraries, Please.

[23] Lionel Ni. Should Scalable Parallel Computers Support Efficient Hardware Multicasting? In *ICPP Workshop on Challenges for Parallel Processing*, pages 2–7, 1995.

[24] N. Nupairoj and L. M. Ni. Performance Evaluation of Some MPI Implementattions on Workstation Clusters. In *Proceedings of the SPLC Conference*, 1994.

[25] S. Pakin, M. Lauria, and A. Chien. High Performance Messaging on Workstations: Illinois Fast Messages (FM). In *Proceedings of the Supercomputing*, 1995.

[26] D. K. Panda. Fast Barrier Synchronization in Wormhole k-ary n-cube Networks with Multidestination Worms. In *International Symposium on High Performance Computer Architecture*, pages 200–209, 1995.

[27] D. K. Panda. Issues in Designing Efficient and Practical Algorithms for Collective Communication in Wormhole-Routed Systems. In *ICPP Workshop on Challenges for Parallel Processing*, pages 8–15, 1995.

[28] D. K. Panda, D. Basak, D. Dai, R. Kesavan, R. Sivaram, M. Banikazemi, and V. Moorthy. Simulation of Modern Parallel Systems: A CSIM-based approach. In *Proceedings of the 1997 Winter Simulation Conference (WSC'97)*, December 1997.

[29] J. Y. L. Park, H. A. Choi, N. Nupairoj, and L. M. Ni. Construction of Optimal Multicast Trees Based on the Parameterized Communication Model. In *Proceedings of the International Conference on Parallel Processing*, Chicago, IL, Aug 1996.

[30] Craig B. Stunkel, D. G. Shea, B. Abali, et al. The SP2 High-Performance Switch. *IBM System Journal*, 34(2):185–204, 1995.

[31] V. S. Sunderam. PVM: A Framework for Parallel and Distributed Computing. *Concurrency: Practice and Experience*, 2(4):315–339, December 1990.

[32] K. Verstoep, K. Langendoen, and H. Bal. Efficient Reliable Multicast on Myrinet. In *Proceedings of the International Conference on Parallel Processing*, pages III:156–165, Aug 1996.

[33] H. Xu, P. K. McKinley, and L. Ni. Efficient Implementation of Barrier Synchronization in Wormhole-routed Hypercube Multicomputers. *Journal of Parallel and Distributed Computing*, 16:172–184, 1992.