Efficient Collective Communication on Heterogeneous Networks of Workstations [*]

Mohammad Banikazemi   Vijay Moorthy   Dhabaleswar K. Panda

Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210
Email: {banikaze,moorthy,panda}@cis.ohio-state.edu

Abstract

Networks of Workstations (NOW) have become an attractive alternative platform for high performance computing. Due to the commodity nature of workstations and interconnects and due to the multiplicity of vendors and platforms, the NOW environments are being gradually redefined as Heterogeneous Networks of Workstations *(HNOW) environments. This paper presents a new framework for implementing 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. *Taking this characteristic into account, we propose two new approaches (*Speed-Partitioned Ordered Chain *(SPOC) and* Fastest-Node First *(FNF)) to implement collective communication operations with reduced latency. We also investigate methods for deriving* optimal *trees for broadcast and multicast operations. Generating such trees is shown to be computationally intensive. It is shown that the FNF approach, in spite of its simplicity, can deliver performance within 1% of the performance of the optimal trees. Finally, these new approaches are compared with the approach used in the MPICH implementation on experimental as well as on simulated testbeds. On a* 24*-node existing HNOW environment with SGI workstations and ATM interconnection, our approaches reduce the latency of broadcast and multicast operations by a factor of up to* 3.5 *compared to the approach used in the existing MPICH implementation. On a* 64*-node simulated testbed, our approaches can reduce the latency of broadcast and multicast operations by a factor of up to* 4.5*. Thus, these results demonstrate that there is significant potential for our approaches to be applied towards designing scalable collective communication libraries for current and future generation HNOW environments.*

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]. 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 Massively Parallel Processor (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, FDDI, Fiber-Channel, and cut-through routed networks such as SP-2 and Myrinet.

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 software packages like PVM [14] and *Message Passing Interface* standards like MPI [3, 10] 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 these characteristics.

The need for *collective communication* operations such as *broadcast, multicast, global reduction, scatter, gather, complete exchange*, and *barrier synchronization* arises frequently in parallel applications [9]. 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, Ethernet, and Myrinet. Performance of collective communication operations in NOW environments have been evaluated in [7, 11]. 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 projects cannot be directly applied to HNOW systems to obtain maximum performance.

[*]This research is supported in part by NSF Career Award MIP-9502294, NSF Grants CCR-9704512 and CDA-9514898, and an OCARNet grant from the Ohio Board of Regents.

To the best of our knowledge, the ECO [8] 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 framework also does not consider other types of heterogeneity. 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 first 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, broadcast and multicast collective communication operations are implemented with reduced latency using binomial trees. Then, we argue that using binomial trees might not be the best approach for implementing these collective operations. We propose a method for finding optimal trees for broadcast and multicast operations. We show that deriving an optimal tree for a set of nodes with arbitrary communication capabilities is however a computationally intensive operation. Next, we introduce a more efficient approach called *Fastest-Node First* (FNF) for implementing these operations. A performance comparison of the FNF scheme with that of optimal trees suggests that the FNF scheme (with its low complexity) can deliver performance within 1% of the performance of optimal trees.

Finally, the SPOC and FNF approaches are evaluated on an experimental testbed consisting of a cluster of 24 SGI workstations and compared with the existing approach used in the MPICH implementation of the MPI standard. Furthermore, these approaches 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 4.9 using the proposed algorithms. These results show that considerable benefits can be obtained by using the proposed approaches for implementing collective communication operations in HNOW environments.

The remaining part of 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 basic idea behind the development of the proposed algorithms is presented in section 3. The new SPOC framework for collective communication and its corresponding algorithms are proposed in section 4, and the method for finding optimal trees is introduced in section 5. The FNF approach, its application for implementing efficient broadcasts and multicasts, and its comparison with the SPOC and optimal tree approaches are presented in section 6. Experimental and simulation results are presented in section 7. 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: 1) Communication Capabilities of Workstations (Nodes), 2) Network Architectures, 3) Communication Protocols, and 4) Dedicated Support for Communication and Synchronization [2]. These factors are orthogonal to each other. A typical HNOW environment can have one or more of these characteristics.

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 [4] 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.

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.)

	HP735	HP715/100	HP715/64	Sun4
HP735	871	973	2491 *	5806 *
HP715/100		1020	2538 *	5871 *
HP715/64			1869	6050 *
Sun4				4196

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.

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 and then formalize the problem. In the following sections, we provide alternative approaches to solve the problem.

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 $(1, 2, \cdots, 8)$. Let node 1 be the source node. Let six of these workstations $(2, 3, 4, 5, 7,$ and 8) be slow ones having a message initiation cost, t^s_{ini}. Similarly, let the other two (1 and 6) 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} = 300.0 microsec, leading to $t^s_{ini}/t^f_{ini} = 3$. 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, based on node numbering. This leads to the broadcast tree as shown in Fig. 1(a). Using this scheme the broadcast can be completed in 700.0 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 although we are still using a binomial tree, the fast workstations are used as intermediate nodes to broadcast the message faster. This scheme takes only 500.0 microsec to implement the broadcast and demonstrates a 29% improvement in broadcast latency. However, this is not the best way of implementing broadcast in this system. Fig. 1(c) illustrates the optimal implementation with a latency of 400.0 microsec (43% improvement over the naive implementation). This example shows that binomial trees which are optimal trees for implementing broadcast on homogeneous systems might not be optimal in heterogeneous environments.

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 and the tree used for implementing the collective operations. 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 totally different. This leads to the following problem:

Problem Statement: *How to design efficient algorithms for collective communication operations on an N-node HNOW environment with each node having arbitrary communication capability:* $t^i_{ini},\ 0 \leq i \leq N-1$.

In the following sections, we provide alternative solutions

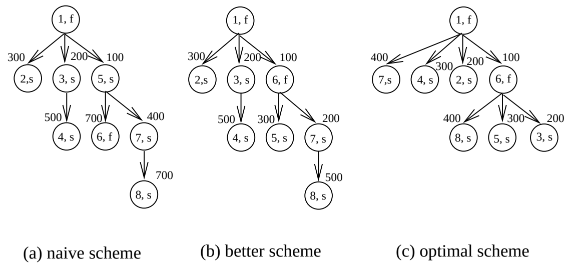

Figure 1. Different ways to implement broadcast in an eight workstation HNOW environment: (a) simple binomial tree-based scheme using node numbering, (b) a better implementation by considering communication capabilities of the workstations, and (b) the optimal implementation.

to solve this problem. We first present a generalized approach to design efficient binomial broadcast trees by considering multiple levels of communication capabilities across nodes, and develop algorithms for broadcast and multicast collective communication operations. Then, we propose a method to derive optimal trees for performing broadcast and multicast operations. Since deriving such trees is computationally expensive, we finally propose a simple approach which can deliver performance close to those of the optimal trees.

4 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. 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 Fig. 1(a) node 5 is a slow node and has 3 descendents while node 6 is a fast node and has no descendent. In Fig. 1(b) node 6 has 3 descendent and node 5 has none. Other faster nodes have also been moved to tree nodes having more children. As a result, the latency of the broadcast in Fig. 1(b) is 200 microsec lower than the latency of the tree in Fig. 1(a).

The above observation leads to the following: The problem of finding the fastest way to implement a binomial tree-based algorithm for broadcast 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 descendents at each tree node. The number of descendents 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 descendents to the node with the lowest number of descendents. Obviously, the root of the tree will have the highest number of descendents ($N-1$), but the source node is already assigned to it. Thus, we have flexibility of assigning other participating nodes to tree nodes. We can sort participating nodes in ascending order of message initiation cost and then assign faster participating nodes to tree nodes with greater number of descendents from these two sorted lists. We call this ordered chain of participating nodes a *Speed-Partitioned Ordered Chain* (SPOC). The outline of this algorithm is illustrated in Fig. 2.

The completion time of broadcast, where W_0 is the source and $\{W_1, W_2, \cdots, W_{N-1}\}$ is the list of other participating nodes in a nondecreasing order with respect to their message initiation cost, can be expressed as:

$$T^{SPOC}_{broadcast} = \sum_{i=0}^{\lceil \log N \rceil - 1} \max(t^0_{ini}, t^{2^i-1}_{ini}) \qquad (1)$$

The same algorithm can be easily used for implementing the multicast [12] operation. An algorithm similar to the algorithm used for multicast can also be used to implement the multiple multicast with the only difference being the fact that different trees are used for each multicast.

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. $SL_t \leftarrow sort(list(T_b), Desc, Num\_Desc)$
   // $SL_t = \{T_i \mid 0 \leq i < N,\ num\_decs(T_i) \leq num\_decs(T_j)\ for\ i < j\}$
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.

5 Optimal Trees

In Section 3 we showed that binomial trees might not be the best tree for implementing broadcast (or multicast) operations on a HNOW. Consider the construction of an optimal tree for performing broadcast (or multicast) among N nodes, $\{W_0, W_1, \cdots, W_{N-1}\}$, where W_0 is the source node. In the first step W_0 sends the message to W_i. Then, W_0 and W_i will be responsible for sending the message to the rest of the participating nodes through two subtrees. First, the Node W_i must be chosen such that the overall tree is *optimal*. Second, the two subtrees must themselves be *optimal*. The same procedure can be applied recursively to each of the multicast subtrees. Therefore, this optimization problem which exhibits optimal substructure and overlapping subproblems properties can be solved by using the dynamic programming technique. This technique can be used for the current problem as follows.

Let $L_{i,A}$ be the minimum latency required to multicast the message from node W_i to all nodes in the set A. If A is an empty set, the latency will be equal to zero. Otherwise, in the first step, message is sent to a node in A (say W_j), and the latency would be the maximum of latencies associated with the two obtained subtrees (where W_i and W_j are the roots of these subtrees and descendents of these two nodes will be all nodes in $A - \{j\}$). Therefore, the overall latency can be obtained through the following recurrence:

$$L_{i,A} = \begin{cases} 0 & \text{if } |A| = 0 \\ min\{t^i_{ini} + max(L_{i,B}, L_{j,C})\}, & \text{otherwise} \\ \text{where } j \epsilon A, \text{ and } B + C = A - \{j\} & \end{cases}$$

It is to be noted that the total running time for finding the optimal tree for broadcasting a message among N nodes will be $O(N^2 2^{2N})$. However, this method for finding the optimal tree is too computationally intensive to be useful in any practical system. In the following section, we propose a near-optimal algorithm which runs in polynomial time. Before describing this near-optimal algorithm, let us look at two important properties of optimal trees presented as the following two lemmas (the proofs can be found in [2]).

Lemma 1 *Let W_0 be the source node of a broadcast (or multicast) operation and $\{W_1, W_2, \cdots, W_{N-1}\}$ be the set of other participating nodes in the order of the time they have received the message. There exists an optimal tree for performing the broadcast (or multicast) operation such that W_k ($1 \leq K \leq N-1$) receives the message from one of the nodes in the set $\{W_0, W_1, \cdots, W_{k-1}\}$ and the time at which it receives the message is the earliest possible time.*

Lemma 2 *Let W_0 be the source node of a broadcast (or multicast) operation and $\{W_1, W_2, \cdots, W_{N-1}\}$ be the set of other participating nodes. Let t^i_{ini} be the message initiation cost of node W_i. There exists an optimal tree for performing the broadcast (or multicast) operation in which the message initiation cost of any node other than the source node is less than or equal to that of its children.*

In the next section we use these two properties to propose a near-optimal algorithm which runs in polynomial time with respect to the number of participating nodes.

6 Fastest-Node First (FNF) Approach

Using the above properties of an optimal tree, we propose a greedy algorithm called Fastest-Node First (FNF). In each iteration of this algorithm, one node which has not received the message is added to the tree. Obviously at each instance, we need to make two decisions. First, we need to decide which node is going to send the message to the new node. From Lemma 1 we can easily find the node which should deliver the message to the new node. The second decision to be made is selecting the new node among the nodes which have not been added to the tree yet. To make sure that the property presented in Lemma 2 is preserved, we select the fastest node among the nodes not in the tree. This way, we can generate trees through which multicast and broadcast operations can be implemented. Now, we describe how FNF can be specifically applied for implementing broadcast and multicast operations.

```
FNF-based tree construction for broadcast
Input:
    $Tini$: the initiation costs of all nodes($t_{ini}$);
    root: the source node of broadcast;
    $L_n$: The list of other participating nodes.

Tree construction steps:
  1. $N \leftarrow number\_of\_nodes(L_n) + 1$
  2. $R_0 \leftarrow 0$
  3. for $i = 1$ to $N - 1$ do
        $R_i \leftarrow \infty$
  4. $Senders \leftarrow \{root\}$
  5. $Receivers \leftarrow L_n$
  6. $Solution \leftarrow \{\}$
  7. for $i = 1$ to $N - 1$ do
     $sender \leftarrow k$ ,where $k \in Senders$ and
        $\forall\, l \in Senders\ (R_k + Tini_k) \leq (R_l + Tini_l)$
     $receiver \leftarrow k$ ,where $k \in Receivers$ and
        $\forall\, l \in Receivers\ Tini_k \leq Tini_l$
     $Solution \leftarrow concat(Solution, (sender, receiver))$
     $R_{sender} \leftarrow R_{sender} + Tini_{sender}$
     $R_{receiver} \leftarrow R_{sender}$
     $Receivers \leftarrow remove(Receivers, receiver)$
     $Senders \leftarrow concat(Senders, receiver)$
     endfor
```

Figure 3. Outline of the FNF algorithm.

6.1 Broadcast

Consider a HNOW system in which N workstations are participating in a broadcast. Let the workstations be partitioned into C workstation classes ($C_0, C_1, \cdots, C_{C-1}$), where $1 \leq C \leq N$ (note that when $C = 1$ we actually have a homogeneous NOW with respect to the message initiation cost). The number of workstations in each class can also vary. Let t^i_{ini} be the message initiation cost for the workstations in class C_i. The FNF algorithm whose outline is presented in Figure 3, starts by creating a list of the nodes which have a copy of the message ($Senders$), and a list of the nodes which are participating in broadcast but have not received the message yet ($Receivers$). Obviously, at the beginning the first list contains only the source node and the second list contains all other nodes. A variable (R_i) is associated with each node indicating the earliest time when the node can send out a message to another node. Since at the beginning of the operations none of the nodes except the source node has the message, infinity is assigned to this variable of all nodes except that of the source which is set to zero. The FNF tree which is presented as a set of (parent, child) two-tuples ($Solution$) is also initialized. Then, in $N - 1$ successive iterations, the node from which the message is supposed to be received at one of the participating nodes (except the root) is found. In each iteration, the best candidate for sending the message to one of the nodes which have not received the message is found by minimizing the time by which this message can be delivered. On the other hand, the fastest node among those which have not received the message yet is chosen as the receiver. After both sender and receiver are selected, the times at which these nodes can send out a new message are each adjusted. During this step, the receiver is taken out of the $Receivers$ list and is added to the $Senders$ list.

It should be noted that the order of the tuples in $Solution$ is important. For any two-tuples whose first items (or sender nodes) are the same, the sender node will send the message to the receiver mentioned in the first two-tuples before sending the message to the other receiver. In other words, the obtained tree is an ordered tree. It should also be noted that for an efficient implementation of the algorithm, nodes can be sorted based on their message initiation costs. This algorithm can be easily implemented with $O(N^2)$ complexity where N is the number of participating nodes.

6.2 Multicast (Single and Multiple)

Similar to the broadcast operation, multicast can be implemented in a more efficient manner under this approach. The FNF algorithm can be directly applied for implementing multicast operation by limiting the list of the participating nodes to the nodes participating in the multicast operation. Similarly, FNF can also be used for implementing multiple multicast by constructing one tree for each multicast operation.

6.3 Comparison With Optimal Trees

To compare the performance of the FNF-based algorithms with that of the optimal algorithms in a qualitative fashion, we evaluated the latency of broadcast and multicast operations obtained from these algorithms. We considered a system with 9 nodes. (We could not go further because of the complexity of the optimal algorithm which is exponential with respect to the number of participating nodes.) The message initiation cost of each node was randomly chosen from the $\{100, 200, 300, \cdots, 800\}$ set. For each operation, the source node and other participating nodes were chosen randomly. We recorded the latency of the FNF-based algorithm (L_{FNF}) and that of the optimal algorithm (L_{opt}) for 10000 cases for each particular number of participating nodes. We then calculated the average latency of the FNF-based and the optimal algorithms. Table 2 shows these latencies. It can be observed that the latency of FNF-based algorithms is equal to that of the optimal algorithms up to 5 participating nodes. Beyond that, the FNF algorithm produces latencies which are within 1% of the latency produced by the optimal algorithm. Furthermore, trees generated by the FNF-based algorithm were found to be identical to those generated by the optimal algorithm for 90-100% of the cases. Considering the very minor difference between the latency of the FNF-based and the optimal algorithms, and the very low complexity of the FNF-based algorithms, we conclude that FNF-based algorithms will be more practical to be incorporated in future HNOW environments. Thus, for the remaining part of the paper, we does not consider the optimal algorithm any further. We only consider the FNF-based algorithms.

Table 2. Comparison between FNF-based and optimal algorithms for implementing broadcast and multicast on a system with 9 nodes.

# Participating Nodes	L_{FNF}	L_{opt}	$\frac{(L_{FNF} - L_{opt})}{L_{opt}}$
2	453.37	453.37	0.00%
3	705.60	705.60	0.00%
4	805.70	805.70	0.00%
5	871.01	871.01	0.00%
6	914.90	913.15	0.19%
7	947.56	942.26	0.56%
8	976.90	967.27	0.99%
9	984.29	977.12	0.73%

6.4 Comparisons between FNF-based and SPOC-based Trees

Let us compare the trees produced by the FNF-based and SPOC-based trees. In general, they will be different because SPOC-based trees are always binomial in nature. It is to be noted that when all nodes participating in broadcast or multicast have the same message initiation startup (i.e. the participating nodes are homogeneous), the tree obtained from the FNF algorithm will be the same as that obtained from the SPOC algorithm and the tree produced by the naive binomial tree algorithm (such as the one used in the existing MPI implementations). It is also to be noted that where 50% of the participating nodes (including the source node) belong to the fastest group of participating nodes, the trees obtained from the SPOC and FNF algorithms will be identical.

7 Performance Evaluation

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

7.1 Experimental Results

We used an ATM network of 24 SGI workstations 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.

7.1.1 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 to take measurements on three different 16-node configurations with 12.5% (2), 25% (4), and 50% (8) fast nodes, respectively. Using the 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.

7.1.2 Broadcast

For measuring the broadcast latency we followed a method similar to the one used in [6]. 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 [6]). 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.

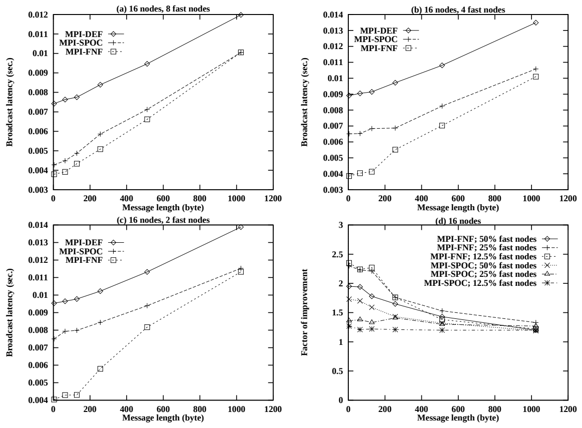

Figure 4. Broadcast latency and factor of improvement on a HNOW system with 16 nodes and different number of fast nodes.

Figure 4 shows the results. In the graphs, latencies of the MPICH implementation are referred to as MPI-DEF (MPI with default implementation) latencies. The modified version of MPICH routines in which SPOC-based algorithms are used, is called MPI-SPOC. We refer to the modified version of MPICH in which FNF-based algorithms are employed to perform MPI-Bcast as MPI-FNF. It can be observed that the proposed algorithms always perform better than the naive ordering scheme used in MPICH. Factors of improvement over MPI-DEF are shown in Figure 4(d). As the fraction of fast nodes in the system increases, so does the factor of improvement. A factor of improvement of 1.7 is achieved in a system with 50% faster nodes by using the SPOC-based algorithm. Factors of improvement up to 2.3 are achievable when FNF-based algorithms are used. As predicted in section 6.4, when 50% of the nodes (including the source node) are fast nodes, SPOC-based algorithms perform as good as FNF-based algorithms. 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.

7.1.3 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 $100 \times (r-1)$ times, where r is the number of recipients.

Figures 5 and 6 show results for multicast. Characteristics similar to broadcast results are observed. The factor of improvement increases with increasing system size. Again, as the message length increases, transmission time dominates and the factor of improvement decreases. For an 8 node multicast with 50% fast participating nodes, factors of improvement of up to 2.5 are observed.

7.2 Simulation Results

Several experiments were performed to measure the impact of different workstation speeds, number of workstations

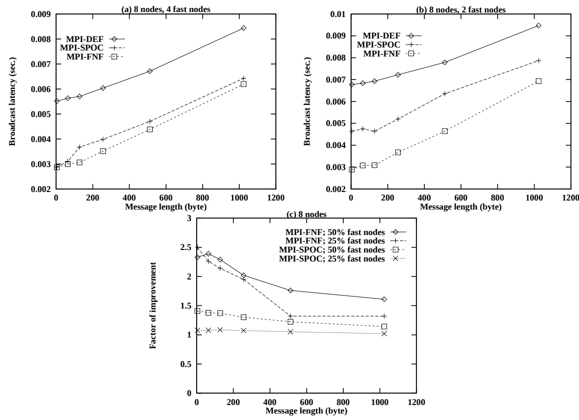

Figure 5. Multicast latency and factor of improvement on a HNOW system with 8 nodes and different number of fast nodes.

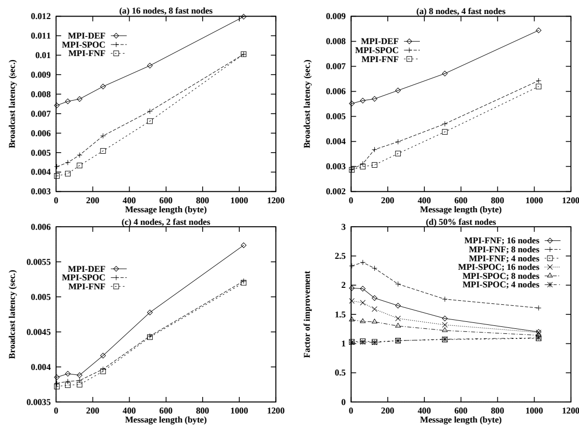

Figure 6. Multicast latency and factor of improvement on configurations with half of the nodes being fast nodes.

and fraction of faster nodes on the performance of our algorithms. In the following subsections, first the simulation setup is described in detail. Next, the results for broadcast, single multicast, and multiple multicast are presented.

7.2.1 Simulation Setup

We modeled a representative HNOW system where workstations are interconnected with Myrinet switches. A detailed flit-level simulator (built using CSIM [13]) 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 the interconnection network, 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[1].

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 and FNF in the graphs) with the default ordering algorithm (indicated as DEF) which is 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). Due to the space limitations, and since broadcast is a special case of multicast, we only present the multicast results. The simulation results for broadcast can be found in [2].

7.2.2 Multicast

Figures 7, 8, and 9 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 and FNF approaches over the DEF increases with increasing number of destinations, increasing speed factor, and decrease in message length. For a multicast involving 1 KBytes message on a system with 25% faster nodes and a speed factor of 4, SPOC and FNF algorithms give factors of improvement of 2.3 and 3.3, respectively.

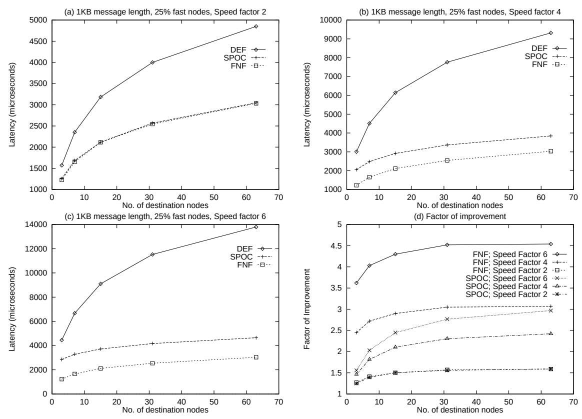

Figure 7. Impact of speed factor on single multicast latency with 25% fast nodes: (a) speed factor 2, (b) speed factor 4, (c) speed factor 6, and (d) factor of improvement for (a) - (c).

Results for multiple multicasts with varying number of sources and destinations are presented in Figure 10. 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 algorithms.

8 Conclusions and Future Research

In this paper, we have presented three new approaches to implement fast collective communication in the emerging HNOW systems. Major factors in HNOW systems have been

[1]Connectivity is defined as the fraction of ports in a switch which are used for interconnection with other switches [5].

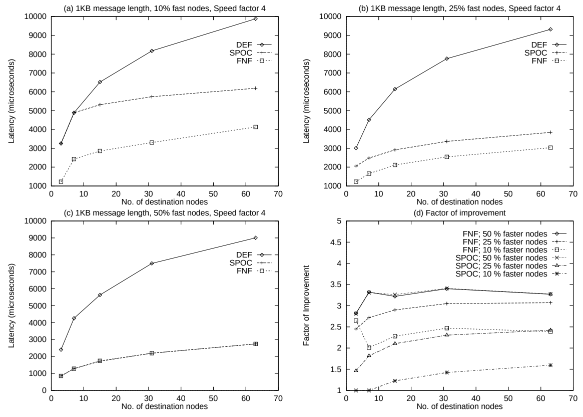

Figure 8. Impact of percentage of faster nodes on single multicast latency with speed factor 4: (a) 10% fast nodes, (b) 25% fast nodes, (c) 50% fast nodes, and (d) factor of improvement for (a) - (c).

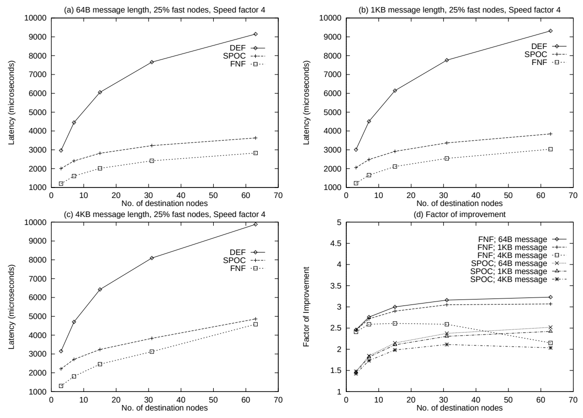

Figure 9. Impact of message length on single multicast latency with 25% fast nodes and speed factor 4: (a) 64 Bytes, (b) 1 KBytes, (c) 4 KBytes, and d) factor of improvement for (a) - (c).

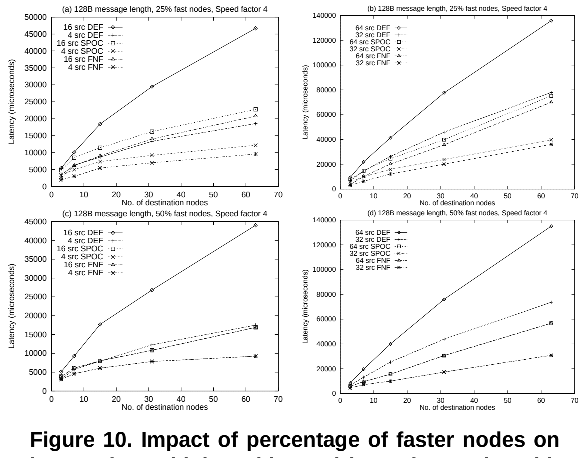

Figure 10. Impact of percentage of faster nodes on latency for multiple multicast: (a) 25% fast nodes with 4 and 16 sources, (b) 25% fast nodes with 32 and 64 sources, (c) 50% fast nodes with 4 and 16 sources, and (d) 50% fast nodes with 32 and 64 sources.

characterized. A 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, and multiple multicast) have been developed. An algorithm for generating optimal trees for these problems have been proposed. Furthermore, a new approach (FNF) with a low complexity has been introduced in which near-optimal trees are used for implementing collective communications efficiently. Performance evaluation of these new algorithms on a 24-node experimental testbed and a 64-node simulated testbed indicates that latency of collective communication operations can be reduced by a factor up to 4.5 compared to the naive algorithms used in current MPI implementations.

In this paper, we have used only the heterogeneous communication capabilities of nodes to implement collective communication efficiently. 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.

Acknowledgment and Additional Information: We would like to thank Ohio Supercomputing Center for access to their workstation cluster. A number of related papers and technical reports can be obtained from http://www.cis.ohio-state.edu/~panda/pac.html.

References

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

[2] M. Banikazemi, V. Moorthy, and D. K. Panda. Efficient Collective Communication on Heterogeneous Networks of Workstations. Technical report OSU-CISRC-03/98-TR07, Dept. of Computer and Information Science, The Ohio State University, March 1998.

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

[4] 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.

[5] R. Kesavan, K. Bondalapati, and D. K. Panda. Multicast on Irregular Switch-based Networks with Wormhole Routing. In *HPCA-3*, pages 48–57, February 1997.

[6] 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.

[7] M. Lin, J. Hsieh, D. H. C. Du, J. P. Thomas, and J. A. MacDonald. Distributed Network Computing over Local ATM Networks. *IEEE JSAC*, 13(4), May 1995.

[8] B. Lowekamp and A. Beguelin. ECO: Efficient Collective Operations for Communication on Heterogeneous Networks. In *IPPS*, pages 399–405, 1996.

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

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

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

[12] 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.

[13] 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)*, pages 1013–1020, December 1997.

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