Efficient All-to-All Broadcast in Heterogeneous Network of Workstations

Jan-Jan Wu
Shih-Hsien Yeh
Institute of Information Science, Academia Sinica
Taipei 115, Taiwan R.O.C.
wuj@iis.sinica.edu.tw

Abstract

Due to the commodity nature of workstations and networking equipment, LAN environments are gradually becoming heterogeneous. In this paper, we design efficient all-to-all broadcast algorithms for heterogeneous network of workstations. We propose a framework called Cluster-Agent Framework as the basis for efficient implementations of all-to-all communication for scientific applications in heterogeneous network of workstations. The framework partitions the participating workstations into clusters, with each cluster having a fast node as the agent to perform all-to-all communication for the slower nodes. Based on this framework, we develop two algorithms, Gather-Broadcast and Two-Step Broadcast, for all-to-all broadcast. Our preliminary experimental results demonstrate performance advantage of our algorithms compared with the MPICH implementation and a number of existing algorithms.

1 Introduction

In recent years, networks of workstations/PCs (so called NOW) are becoming appealing vehicles for cost-effective parallel computing. Due to the commodity nature of workstations and networking equipment, LAN environments are gradually becoming heterogeneous. The diverse sources of heterogeneity in NOW systems pose a challenge on the design of efficient communication algorithms for this class of systems.

Many research projects are currently in progress to provide efficient communication for NOW systems. However, most of these research projects focus on homogeneous NOWs, systems comprising of the same type of PCs/workstations connected over a single network architecture. Due to the commodity nature of workstations and networking equipment, LAN environments are gradually becoming heterogeneous. The heterogeneity could be due to the difference in processing speed and communication capability of the workstations, or coexistence of multiple network architectures or communication protocols. This trend of heterogeneity is forcing networks of workstations to be redefined as Heterogeneous Networks of Workstations (HNOW).

Collective communication provides important functionality for many applications, and thus efficient implementations of collective communication is crucial for achieving maximum performance of applications on message-passing systems. Typical examples of collective operations include *broadcast*, *barrier synchronization, reduction, gather, scatter*, and all-to-all communication.

Over the past few years, a large number of efficient algorithms for collective communication have been devised to facilitate direct programming of massively parallel, distributed memory computers by taking advantage of special network topology in this class of machines (e.g. [7, 11, 10, 3]). These algorithms may not be efficient for networks of workstations, however, due to lack of special network topology in NOW environment. Both PVM[5] and MPI[4] support a set of collective communication routines. The performance of some available MPI implementations for networks of workstations was measured in [9]. However, to our knowledge, none of the existing implementations of PVM or MPI addresses the issue of communication optimization in HNOW environment.

In this paper, we study all-to-all broadcast in HNOW systems. We propose a framework called *Cluster-Agent Framework* as the basis for efficient implementations of all-to-all communication. The framework partitions the participating worksta-

tions into clusters, with each cluster having a fast node as the agent. Slower nodes in a cluster can take advantage of the communication capability of the agent (the fast node) by sending their local data for collective communication to the agent and let the agent does the work for them.

Based on this framework, we implemented two algorithms, *Gather-Broadcast* and *Two-Step Broadcast*, for all-to-all broadcast. In the *Gather-Broadcast* algorithm, the agent of each cluster gathers data from the slower nodes in its cluster, then performs inter-cluster all-to-all broadcast with the gathered data, and then the agents distribute the final result back to the slower nodes. In the *Two-Step Broadcast* algorithm, we further decompose the inter-cluster all-to-all broadcast operation into two, and reduce communication latency by overlapping the intra-cluster gather operation with inter-cluster all-to-all broadcast.

Our preliminary experimental results in a heterogeneous network of workstations show that both algorithms outperform the MPICH implementation and two existing algorithms for all message lengths (less than 2k bytes), and the Two-Step Broadcast algorithm is superior to the Gather-Broadcast algorithm for longer messages.

The second part of the paper addresses the issue of clustering and agent assignment in our framework. We define a characterizing model for point-to-point communication in HNOW systems. The model defines four parameters: sending overhead, receiving overhead, end-to-end latency, and bandwidth. We then formalize intra-cluster communication and inter-cluster communication using these parameters, and define the cost functions for all-to-all broadcast. We then propose heuristic algorithms to assign clusters and agents with the goal to minimize the value of the cost function.

The rest of the paper is organized as follows. Section 2 defines the characterized model for communication systems. Section 3 gives overview of our Client-Agent framework for all-to-all communication. Section 4 presents more details of the communication analysis under our characterizing model. Section 5 presents the algorithms for constructing clusters and assigning communication agents. Section 6 reports our experimental results on a heterogeneous cluster of workstations. Section 7 describes some related works, and Section 8 gives some concluding remarks.

2 Characterizing Communication Model

This section characterizes the heterogeneity of HNOW systems by the communication capabilities of the participating workstations, and describes the communication model in which we design efficient all-to-all broadcast.

Our model abstracts the communication capability of a workstation by four parameters: the *end-to-end latency* with others, *sending overhead*, *receiving overhead*, and the *bandwidth*. We characterize a heterogeneous workstation cluster by the point-to-point communication latency of its processors, which may be very different from one processor to another. The latency of a point-to-point communication is the time to initiate, generate, send, and receive a message from the source to the destination. The software overhead, which constitutes a large portion of the latency, includes message copying from the local memory to the network interface buffer at the sender, and from the interface buffer to the local memory at the receiver.

The communication capability of a workstation is characterized by the following parameters.

- T_{send} : start-up latency for the message passing initiation at the sender.
- T_{recv} : software overhead at the receiver.
- T_{end} : end-to-end latency between the sender and the receiver.
- T_{hold} : the holding time, which is the minimum time interval between two consecutive send operations.

The reciprocal of T_{hold} corresponds to the available communication bandwidth of the node. For simplicity, in this paper we assume that T_{hold} is equal to T_{send}.

3 The Cluster-Agent Framework

Most of the existing all-to-all communication algorithms assume a homogeneous environment and distribute the communication evenly among processors. In a HNOW with different communication

capabilities among processors, such scheduling algorithms can easily cause imbalance in communication load in all-to-all communication. An efficient algorithm for HNOW should assign the proper amount of network loads to processors based on their communication capability to take advantage of the communication power of faster nodes as much as possible.

Based on this idea, we present a framework called Cluster-Agent Framework for all-to-all broadcast. In an HNOW environment consisting of n workstations as shown in Figure 1, we select the m fastest nodes as the communication agents. The remaining $n-m$ slower nodes are partitioned into m groups of clients. Each agent is responsible for one group of clients, in that it gathers data from the clients and performs all-to-all broadcast for them. An agent and the clients it serves form a cluster.

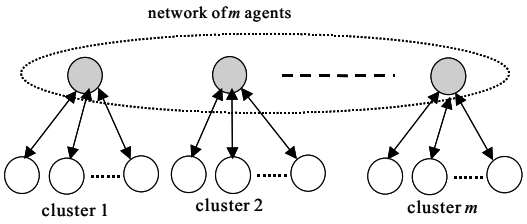

Figure 1: The Cluster-Agent framework for an n-node HNOW environment

We have designed two algorithms to implement all-to-all broadcast based on this framework. One is *Gather-Broadcast*, the other is *Two-Step Broadcast*. The following section describes these two algorithms.

3.1 Gather-Broadcast

Gather-Broadcast has three stages.

1. *Gather* Stage. The agents gather messages from their own clients concurrently.
2. *Broadcast* Stage. After gathering messages from the clients, each agent combines its own local data with the gathered data, and then performs inter-cluster all-to-all broadcast with all the other agents. When the broadcast is completed, each agent has a copy of the final result of the all-to-all operation.
3. Agents send the final result to their clients and finish the all-to-all communication.

3.2 Two-Step Broadcast

The second algorithm, Two-Step Broadcast, is an enhancement to the first algorithm. The idea is to decompose the broadcast into two in order to allow overlapping of communication between different stages. Two-Step Broadcast also has three stages.

1. Each agent broadcasts its message to all the other agents, and at the same time the clients send their messages to their agents.
2. After broadcasting their own messages to other agents, the agents receive the messages of their clients. Then the agents perform an all-to-all broadcast among themselves to distribute the messages of their clients to other agents.
3. Agents send the final result to their clients and finish the all-to-all communication.

We have implemented these two algorithms. The performance comparison of these two algorithms will be presented in later section. While the idea behind the cluster-agent framework is simple, its efficiency depends on two important factors. First, how to decide which nodes and how many nodes should be assigned to each agent? Secondly, how to decide the appropriate number of agents and how to choose them? To solve these problems satisfactorily requires formal analysis of both intra-cluster and inter-cluster communication, which is presented in the following section.

4 Cluster Communications

Cluster communications in our model include intra-cluster and inter-cluster communications. Intra-cluster communication refers to the message passing between an agent and its clients, while inter-cluster communications are communications among the agents. They are both important to the overall performance of all-to-all broadcast.

We make the following assumptions to simplify the analysis. (1) The collective communication is implemented by performing multiple point-to-point message passing, (2) For a sending node, a sending operation is complete when the sender can reuse the sending buffer. In other words, the sender just needs to wait a T_{hold} time before it can issue

another send operation, (3) For a receiving node, the receiving operation is in blocking mode. That is, the receiving node can continue to execute its next operation only when the received message is removed from its local network buffer to its own local memory, (4) We assume each node sends its messages independently. This assumption is reasonable for NOW connected by modern high-speed network or switch-based interconnection network, and (5) A client can only communicate with its agent, but cannot communicate with its peers in the same cluster. This assumption avoids the inevitable synchronization points when a client acts as an intermediate node between other peers and the agent. In addition, this constraint reduces the network load of the overall system.

4.1 Intra-Cluster Communications

The first stage of the Gather-Broadcast algorithm and the second stage of the Two-Step Broadcast algorithm require a gather operation to collect messages from the clients to the agents. The efficiency of the gather operation will affect the overall performance of the all-to-all broadcast.

We formalize the intra-cluster communication problem as follows. Assume all the clients of a cluster start sending message to the agent at time 0 and these messages arrive at the agent according to their end-to-end latency. The agent must collect these messages one at a time, with an extra overhead of T_{recv}. The question is in what order the agent should collect these messages so that the total time is minimized. We adopt an intra-cluster scheduling in which the agent of a cluster receives the messages of its clients in first-come-first-served order. Figure 2 shows an example of the resulting schedule. In the figure, L_i is the arriving time of the message sent by client i and T_{recv} is the receiving overhead of the agent

Definition 1 Assume there are K clients in a cluster. Let $T_{end}(j)$ be the end-to-end latency of the point-to-point communication between the agent and the jth client it receives, and $T_{recv}(j)$ be the local receiving overhead of the agent to receive the message of the jth client. Then the time for the agent to complete receiving the message from the jth client, denoted by $T_r(j)$, can be described recursively as follows, and the latency of the gathering operation as $T_g = T_r(K)$.

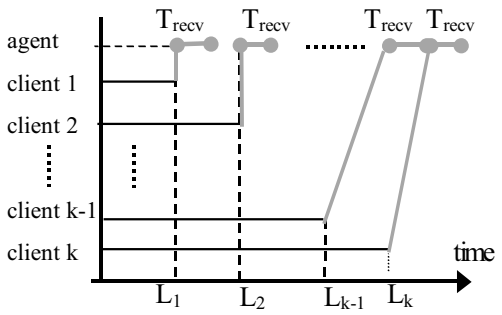

Figure 2: An example of intra-cluster scheduling

$$T_r(j) = \begin{cases} T_{end}(1) & j = 1 \\ \max\{T_r(j-1) + T_{recv}(j), T_{end}(j)\} & j > 1 \end{cases}$$

4.2 Inter-Cluster Communications

In our Cluster-Agent framework the agents must perform all-to-all broadcast among themselves so that every agent will have messages from all processors before sending them to their clients. The efficiency of this inter-cluster communication will have great impact on the overall performance since a large amount of data will be transmitted via the interconnection network.

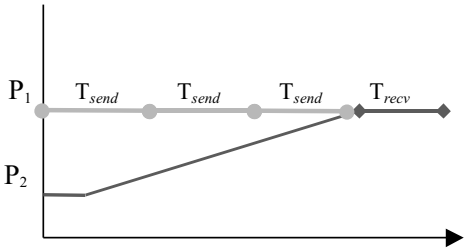

Figure 3: P_1 executes multiple sends before receiving the message sent by P_2.

We use the simultaneous broadcast algorithm for all-to-all broadcast among agents. Each agent sends its message to all the other agents, and receives message from them as well. If there are N agents, then each agent sends and receives $(N-1)$ messages to complete an all-to-all broadcast.

To improve communication efficiency we overlap the send and receive operations. In our model a

receive operation is blocking, i.e., the receiver cannot execute the next operation until the receiving operation has completed. To reduce communication latency, we overlap multiple sending operations with a single receive, as illustrated in Figure 3. Since a send operation is non-blocking, we may issue multiple sends before a receive. During the time to send these outgoing messages, an incoming message the processor is waiting for may have already traveled through the network and arrived at the buffer of the receiving node.

In order to overlap k sending operations with a receive operation, we need to decide the proper value of k so that the completion time of the k sends at the receiver is close to the time for the message to travel through the network and arrive at the network buffer. We estimate the value of k by $(k = T_{end}/T_{send})$ where T_{end} is the end-to-end latency of the point-to-point communication and T_{send} is the sending overhead of the receiving node. However, in a heterogeneous environment an agent may receive messages from other agents with different communication capability. For simplicity we estimate the value of k in a heterogeneous environment by

$$k = \texttt{min}\left\{m-1, \lceil\frac{T_{max}}{T_{send}}\rceil\right\}$$

where T_{max} is the maximum among all the end-to-end latencies from the other agents and m is the number of agents.

In the above definition, we consider the case in which the total number of sending operations performed, $(m-1)$, is less than the value estimated as the ratio of T_{max} to T_{send}. To evaluate the overall performance of the two all-to-all broadcast algorithms we proposed, we estimate the latency of an inter-cluster broadcast stage as follows. Let m be the number of agents, and $T_{send}(i)$ and $T_{recv}(i)$ be the sending and receiving overhead of agent i respectively. The latency $T_B(i)$ of the broadcast operation is estimated as follows, where m is the number of agents and n is the number of agents having a message to broadcast.

$$T_B(i) = \begin{cases} \texttt{max}\{(m-1)\cdot T_{send}(i) + (n-1)\cdot T_{recv}(i), T_R(i,n-1)\} \\ \quad \text{if agent i has a message to broadcast} \\ T_R(i,n-1) \end{cases}$$

$T_R(i,j)$, similar to **Definition** 1, is the estimated receiving completion time of the message from the jth agent received by agent i and it is estimated as follows, where $T_{end}(i,j)$ is the end-to-end latency of point-to-point communication between agent i and the jth agent it receives.

$$T_R(i,j) = \begin{cases} T_{end}(i,1) & j = 1 \\ \texttt{max}\{T_R(i,j-1) + T_{recv}(i), T_{end}(i,j)\} & j > 1 \end{cases}$$

5 Cluster Assignment

Next, we consider the issues of agent and clients assignment. To fully utilize the communication bandwidth, we need to select those agents carefully so that inter-cluster communication can be optimized. Similarly, clients must be assigned to agents so that intra-cluster communication can be completed in minimum time. Before describing our assignment algorithm, we define a set of cost functions that evaluate the performance of a cluster assignment. First, we consider the latency at each stage of the all-to-all broadcast. For the two proposed all-to-all algorithms we estimate their costs as follows.

Definition 2 Let $T_g(i)$ be the gathering latency from **Definition** 1 for cluster i and m be the number of clusters. The cost of stage 1 for Gather-Broadcast Approach is estimated as

$$T_1 = \max_{1\le i\le m} T_g(i)$$

And let $T_B(i)$ be the broadcasting latency described in the previous section for cluster i, K_j be the number of clients in cluster i, then the cost of stage 1 for Two-Step Broadcast Approach is estimated as

$$T_1 = \max_{1\le j\le m}\{\texttt{max}\{T_B(j) + K_j\cdot T_{rev}(j), T_g(j)\}\}$$

In **Definition** 2, since the stage 1 of Two-Step Broadcast approach performs the broadcasting operation for the messages owned by the agents themselves and at the same time the gathering operation of messages from clients for each cluster, an agent cannot serve the messages of the clients that have arrived at its network buffers until its broadcasting operation completes. Thus, when an agent completes the broadcasting, the messages from some clients have been queued at its network buffers, and the agent can start to move the queued messages to its local memory.

Definition 3 Let $T_B(i)$ be the broadcasting latency for cluster i defined in the previous section and m be the number of clusters. The cost of stage 2 for both Gather-Broadcast Approach and Two-Step Broadcast approach is estimated as follows.

$$T_2 = \max_{1 \le i \le m} T_B(i)$$

Definition 4 Let m be the number of clusters, K_j be the number of the clients in cluster i, $T_{send}(i)$ be the sending overhead of the agent of cluster i, and $T_{end}(i,j)$ be the end-to-end latency from the agent of cluster i to the jth clients it served. The cost of stage 3 is estimated as follows.

$$T_3 = \max_{1 \le i \le m} \{ \max_{1 \le j \le K_j} \{ j \cdot T_{send}(i) + T_{end}(i,j) \} \}$$

Finally, the overall latency of the all-to-all broadcast is the sum of the latencies of the three stages.

Definition 5 The overall latency of the all-to-all communication is $T_{all} = T_1 + T_2 + T_3$.

5.1 Client Assignment

Suppose we have selected m processors as agents, how do we assign the remaining processors to these agents so that the intra-cluster gathering time can be minimized?

We propose a greedy algorithm that assigns clients to agents one at a time, and tries to schedule the client as early as possible. That is, among the m agents a client can be assigned to, we pick the one that can receive it earliest (after receiving all the previously assigned clients). The clients are assigned in decreasing order of their communication speed, i.e., the fastest client will be assigned first, so that the decision of which agent can receive the current client, can be made quickly.

5.2 Agent Selection

Finally the only question in our cluster formation problem is how to select agents, since from the discussion above we know how to assign clients to agents.

For simplicity we assume all the agents are the fastest processors among the cluster. The rational is that the agents are responsible for the bulk of the communication, namely to perform an all-to-all communication among themselves. In order to improve the overall performance the agents must be fast enough. As a result we only need to determine the number of agents, namely the number of clusters. We determine the number of clusters by testing all possible values. For a HNOW system of P nodes, we evaluate the total cost for a particular cluster number, based on the cost function, T_{all}, defined earlier. Then we select the one that leads to the minimum overall cost and assign clients to agents accordingly.

We would like to point out that once the problem size is given, the decision of agents can be done once for all before the execution of all-to-all broadcast operations. Thus, after paying the one-time cost of analyzing the networks, the execution of all-to-all broadcast is not delayed.

6 Experimental Result

We use a 100-Mbps Fast Ethernet network of 8 Ultrasparc workstations as our experimental environment to evaluate the proposed algorithms. There are 4 fast nodes (UltraSparc2) and 4 slow nodes (UltraSparc1).

In Figure 4 we show the comparison between the estimated time (time computed using our communication model) and the measured time (actual execution time) of all-to-all broadcast of 32-byte messages. The parameters T_{send}, T_{recv} and T_{end} are measured using a ping-pong scheme described in [2]: T_{send}(fast node) = 90 microsec, T_{send}(slow node) = 160 microsec, T_{recv}(fast node) = 70 microsec, T_{send}(slow node) = 130 microsec, T_{end}(fast to fast) = 250 microsec, T_{end}(fast to slow) = T_{end}(slow to fast) = 350 microsec, and T_{end}(slow to slow) = 450 microsec. The best choices of the number of agents (four agents in Gather-Broadcast and three in Two-Step-Broadcast) by our algorithms (estimated cost) are quite consistent with those of the actual runs (meassured cost).

In Figure 5, we show the performances of various All-to-all broadcast algorithms with different message lengths. The purpose is to demonstrate the importance of taking heterogeneity into consideration while designing communication algorithms. We can observe that for small message length the

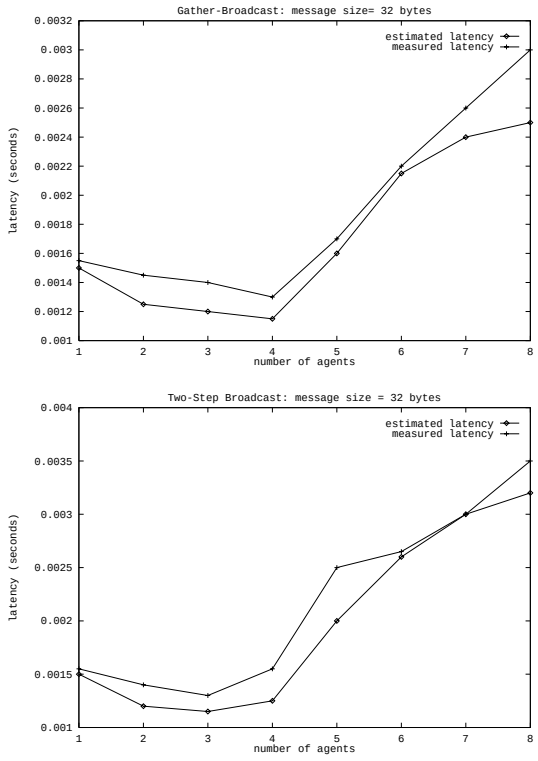

Figure 4: Estimated cost and measured completion time of all-to-all broadcast on 8 nodes (4 fast and 4 slow) with different numbers of agents for the two approaches.

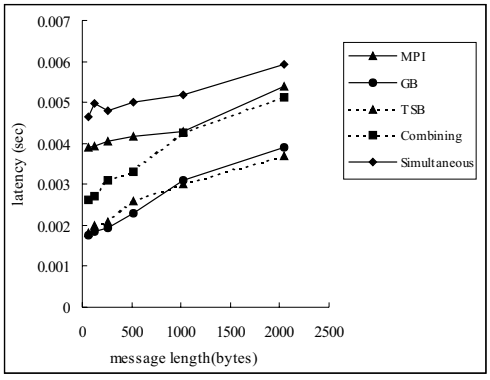

Figure 5: Performance comparison of all-to-all broadcast implemented by different algorithms

two proposed algorithms, Gather-Broadcast and Two- Step Broadcast, can achieve speedup of a factor of 2, compared to the MPICH implementation. The proposed algorithms also outperform the two well-known algorithms for All-to-all broadcast on homogeneous systems, the Combing Broadcast and the Simultaneous Broadcast [6]. Moreover, the Two-Step Broadcast algorithm is superior to the Gather-Broadcast algorithm for larger message length. This is because at the stage of broadcasting among agents, Gather-Broadcast algorithm will incur higher communication overhead than the Two-Step algorithm due to the transmission of longer messages.

7 Related Work

The study on collective communication for heterogeneous networks of workstations was initiated by the ECO project [8]. ECO proposed heuristic algorithms to partition the workstations participating in a collective communication into subnetworks based on pair-wise round-trip latencies between workstations. it then decomposes the collective communication into two phases: inter-subnetwork and intra-subnetwork. ECO automatically chooses a suitable tree algorithm for each of these phases. The network partitioning approach based on pair-wise round-trip latencies was shown to be effective in implementing collective communication on heterogenous networks (i.e. systems where multiple network architectures coexist). However, it does not consider other types of heterogeneity, such as the communication capabilities of individual workstations.

Banikazemi et al. [1] proposed two new algorithms to optimize multicast communication for NOW with heterogeneity in communication capability of workstations. The Sped-Partitioned Ordered Chain (SPOC) algorithm ordered the participating nodes of a collective communication based on their communication capabilities (measured by round-trip latency between different types of workstations), and then assigns the nodes to the binomial trees for broadcast/multicast based on the ordering. The authors show that SPOC may not be efficient for general cases and then proposed a greedy algorithm called Fastest Node First (FNF). In each iteration of the algorithm, the fastest node which has not received the message is added to the tree. Their simulation results show that the FNF algorithm approaches near optimal solution

for multicast communication.

Although FNF can also be used for implementing multiple multicast by constructing one tree for each multicast operation, it may not be the most efficient way to do it. In this paper we take on this challenge and design efficient Cluster-Agent based algorithms for all-to-all communication (all-to-all broadcast, complete exchange, and multiple multicast) for NOW/HNOW.

Jacunski et al. studied all-to-all broadcast on clusters of workstations based on commodity switch-based networks [6]. A new algorithm called *link scheduling*, which is an enhancement to the *simultaneous broadcast* algorithm, was proposed to avoid link contention problem between switches. The basic idea is that use of the interconnecting link is scheduled among nodes in a way that permits every node to remain busy with useful work. Since the link scheduling algorithm employs a homogeneous scheduling algorithm for all-to-all broadcast between the nodes connected to the same switch, it may not be efficient for heterogeneous NOW.

8 Conclusion

In this paper we have presented the Cluster-Agent framework for all-to-all communication and two algorithms for efficient all-to-all broadcast based on this framework. Our preliminary experimental results on a heterogeneous network of eight workstations show performance advantage of our algorithms compared with the MPICH implementation and two other existing algorithms. It is, however, desirable to do more extensive simulation to evaluate our algorithms. As part of this research effort, we are currently developing a CSIM-based network simulator for HNOW systems. We will include our simulation data for all-to-all broadcast in the final version of this paper.

The analysis of cluster communications in this paper ignore the impact of message concatenation during all-to-all broadcast between agents. This is acceptable only when the messages to be broadcasted are short. Fortunately, in most scientific applications, broadcast messages are usually short (less than 1k bytes). In the future, we will relax this restriction and extend our algorithms for arbitrary message lengths.

References

[1] M. Banikazemi, V. Moorthy, and D. K. Panda. Efficient collective communication on heterogeneous networks of workstations. In *Proceedings of International Conference on Parallel Processing*, pages 460–467, 1998.

[2] M Banikazemi, J. Sampathkumar, S. Prabhu, D. K. Panda, and P. Sadayappan. Communication modeling of heterogeneous networks of workstations for performance characterization of collective operations. In *Proceedings of the Heterogeneous Computing Workshop*, 1999.

[3] V. V. Dimakopoulos and N. J. Dimopoulos. A theory for total exchange in multidimensional interconnection networks. *IEEE Transactions on Parallel and Distributed Systems*, 9(7):639–649, July 1998.

[4] Message Passing Interface Forum. MPI: A message-passing interface standard, 1994.

[5] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine*. MIT Press, 1994.

[6] M. Jacunski, P. Sadayappan, and D. K. Panda. All-to-all broadcast on switch-based clusters of workstations. In *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS)*, 1999.

[7] S. Latifi. Fast broadcasting and gathering in q-ary cubes using error correction codes. *Journal of Parallel and Distributed Computing*, 48(1):52–63, Jan 1998.

[8] B. Lowekamp and A. Beguelin. ECO: Efficient collective operations for communication on heterogeneous networks. In *Proceedings of International Parallel Processing Symposium*, pages 399–405, 1996.

[9] N. Nupairoj and L. M. Ni. Performance evaluation of some MPI implementations on workstation clusters. In *Proceedings of the SPLC Conference*, 1994.

[10] R. Sivaram, D. K. Panda, and C. B. Stunkel. Efficient broadcast and multicast on multistage interconnection networks using multiport encoding. *IEEE Transactions on Parallel and Distributed Systems*, 10(1):1004–1028, Jan 1999.

[11] Y.-C. Tseng, S.-Y. Wang, and C.-W. Ho. Efficient broadcast in wormhole-routed multicomputers: a network-partitioning approach. *IEEE Transactions on Parallel and Distributed Systems*, 10(1):44–61, Jan 1999.