Communication Modeling of Heterogeneous Networks of Workstations for Performance Characterization of Collective Operations [*]

Mohammad Banikazemi Jayanthi Sampathkumar Sandeep Prabhu
Dhabaleswar K. Panda P. Sadayappan

Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210
Email: {banikaze,sampath,sprabhu,panda,saday}@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). Having an accurate model for the communication in HNOW systems is crucial for design and evaluation of efficient communication layers for such systems. In this paper we present a model for point-to-point communication in HNOW systems and show how it can be used for characterizing the performance of different collective communication operations. In particular, we show how the performance of broadcast, scatter, and gather operations can be modeled and analyzed. We also verify the accuracy of our proposed model by using an experimental HNOW testbed. Furthermore, it is shown how this model can be used for comparing the performance of different collective communication algorithms. We also show how the effect of heterogeneity on the performance of collective communication operations can be predicted.*

1 Introduction

The availability of modern networking technologies [1, 4] and cost-effective commodity computing boxes is shifting the focus of high performance computing systems towards Networks of Workstations (NOW). The NOW systems comprise of clusters of PCs/workstations connected over the Local Area Networks (LAN) and provide an attractive price to performance ratio. Many research projects are currently in progress to provide efficient communication and synchronization for NOW systems. However, most of these projects focus on *homogeneous* NOWs, systems comprising of similar kinds of PCs/workstations connected over a single network architecture. The inherent scalability of the NOW environment combined with the commodity nature of the PCs/workstations and networking equipment is forcing the NOW systems to become *heterogeneous* in nature. The heterogeneity could be due to: 1) the difference in processing and communication speeds of workstations, 2) coexistence of diverse network interconnects, or 3) availability of alternative communication protocols. This adds new challenges to providing fast communication and synchronization on HNOWs while exploiting the heterogeneity.

The need for a portable parallel programming environment has resulted in development of software tools like PVM [22] and standards such as the Message Passing Interface (MPI) [14, 20]. MPI has become a commonly accepted standard to write portable parallel programs using the message-passing paradigm. The MPI standard defines a set of primitives for point-to-point communication. In addition, a rich set of collective operations (such as broadcast, multicast, global reduction, scatter, gather, complete exchange and barrier synchronization) has been defined in the MPI standard. Collective communication and synchronization operations are frequently used in parallel applications [3, 5, 7, 13, 15, 18, 21]. Therefore, it is important that these operations are implemented on a given platform in the most efficient manner. Many research projects are currently focusing on improving the performance of point-to-point [17, 23] and collective [11, 16] communication operations on NOWs. However, none of these studies ad-

[*]This research is supported in part by NSF Career Award MIP-9502294, NSF Grant CCR-9704512, an Ameritech Faculty Fellowship, and an Ohio Board of Regents Collaborative Research Grant.

dress heterogeneity. The ECO package [12], built on top of PVM, uses pair-wise round-trip latencies to characterize a heterogeneous environment. However, such latency measurements do not show the impact of heterogeneity on communication send/receive overhead (the *fixed* component as well as the *variable* component depending on the message length). A detailed model is necessary to develop optimal algorithms for collective communication operations for a given HNOW system. The existence of such a model also allows to predict/evaluate the impact of an algorithm for a given collective operation on a HNOW system by simple analytical modeling instead of a detailed simulation.

In our preliminary work along this direction [2], we demonstrated that heterogeneity in the speed of workstations can have significant impact on the *fixed* component of communication send/receive overhead. Using this simple model, we demonstrated how near-optimal algorithms for broadcast operations can be developed for HNOW systems. However, this model does not take care of the *variable* component of communication overhead and the transmission component.

In this paper, we take on the above challenges and develop a detailed communication model for collective operations. First we develop a model for point-to-point communication. We present a methodology to determine different components of this model through simple experiments. Next, we show how this model can be used for evaluating the performance of various collective communication operations based on the configuration of the system and the algorithm used for that collective operation. The correctness of this model is validated by comparing the results predicted by this model with the results gathered from our experimental testbed for different system configurations and message sizes. Finally, we illustrate how this model can be used by algorithm designers to select strategies for developing optimal collective communication algorithms and by programmers to study the impact of a HNOW system configuration on the performance of a collective operation.

This paper is organized as follows. The communication model for point-to-point operations is presented in Section 2. The communication model for a set of collective operations and the evaluation of these models are discussed in Section 3. In Section 4, it is shown how the proposed model can be used for comparing different schemes for implementing collective communication operations. It is also shown how we can predict the performance of collective operations with changes in system configuration. We conclude the paper with future research directions.

2 Modeling Point-to-point Communication

For the characterization of collective communication operations on heterogeneous networks, the knowledge of send and receive costs on the various nodes is imperative. In most implementations of MPI, such as MPICH [8], collective communication is implemented using a series of point-to-point messages. Hence, from the characterization of point-to-point communication on the various type of nodes in a heterogeneous network, the cost of any collective communication operation can be estimated. This section explains the method adopted to determine the send and receive costs as a function of the message size for point-to-point communication in a heterogeneous network.

2.1 Characterization of Communication Components

The cost of a point-to-point message transfer consists of three components, namely, the send overhead, transmission cost and the receive overhead. These components comprise of a message size dependent factor and a constant factor. Thus, the one-way time for a single point-to-point message between two nodes can be expressed as:

$$\begin{aligned} T_{ptp} &= O_{send} + O_{trans} + O_{receive} && (1)\\ O_{send} &= S_c^{sender} + S_m^{sender} \cdot m && (2)\\ O_{trans} &= X_c + X_m \cdot m && (3)\\ O_{receive} &= R_c^{receiver} + R_m^{receiver} \cdot m && (4) \end{aligned}$$

where m is the message size (in bytes). The components S_c, X_c, and R_c are the constant parts of the send, transmission and receive costs respectively. The components $S_m \cdot m$, $X_m \cdot m$, and $R_m \cdot m$ are the message dependent parts.

To obtain an empirical characterization of point-to-point communication, each term in these equations needs to be measured. In order to measure these terms we need two sets of experiments: *ping-pong experiment* and *consecutive sends experiment*. The time for completing a point-to-point message between a pair of nodes can be measured using a ping-pong experiment. In this experiment, one of the nodes performs a send followed by a receive while the other does a receive followed by a send. The round-trip time is determined by averaging over multiple such iterations. If the nodes involved are identical, the time for a point-to-point message transfer is equal to half the round-trip time. For small messages, the message size dependent cost can be ignored compared to the constant costs. Therefore,

$$\begin{aligned} T_{ptp,small} &= \frac{1}{2} T_{ping-pong,small} \\ &\approx S_c^{sender} + R_c^{receiver} && (5) \end{aligned}$$

The send component of point-to-point messages (S_c) in the above equation can be obtained by measuring the time for a small number of consecutive sends from one of the nodes and averaging the time over the number of sends.

Using the measured value of S_c and the measured value of $T_{ptp,small}$, R_c can be obtained from Equation 5. The consecutive sends experiment can be used repeatedly for varying message sizes to calculate the send overhead for those message sizes. The linear fit of the send overheads can be plotted as a function of message size. The component S_m is equal to the slope of the straight line fitted on this plot. The transmission cost (O_{trans}) can be determined from the configuration of the underlying network. Then, R_m can be measured by taking the difference of the T_{ptp} obtained from the ping-pong experiment and the sum of all other components in Equation 1.

2.2 Measurement of Communication Components

In this section we describe the testbed used to verify our proposed model and present the measured values of different components of point-to-point communication.

We used two types of personal computers in our testbed. The first group of nodes were Pentium Pro 200MHz PCs with 128MB memory and 16KB/256KB L1/L2 cache. These machines are referred to as *slow* nodes in the rest of the paper. The second group of nodes were Pentium II 300MHz PCs with 128MB memory and 32KB/256KB L1/L2 cache. We refer to these machines as *fast* nodes. All the nodes were connected to a single Fast Ethernet switch with a peak bandwidth of $100Mbits/sec$ and a $8\mu sec/port$ latency. Thus, the transmission cost ($X_m \cdot m + X_c$) for our testbed can be expressed as $(m/12.5 + 2 * 8)\mu sec$. The factor of two in this expression is used to reflect the effect of latencies for both input and output switch ports in the one-way communication.

We performed the experiments described in Section 2.1 for both slow and fast nodes. The components corresponding to the slow and fast nodes are shown in Table 1.

Table 1. Send and Receive parameters.

Type	S_c (μsec)	S_m	R_c (μsec)	R_m
slow	90.0	0.18	140.0	0.08
fast	60.0	0.05	110.0	0.03

Observation 1 *The send overhead and receive overhead can be different in a heterogeneous environment. Using the same value for both overheads or just considering the send overhead may result in inaccurate models.*

We also verified these values by measuring the one way latency of messages between different types of nodes. Table 2 illustrates the latency (of zero-byte messages) between fast and slow nodes. The experimental results corresponding to one-way latencies agree with those calculated from Equations 1 through 4. It should be noted that the same set of experiments can be used for systems with multiple types of computing nodes. In general, for a system with n different types of computers, the experiments should be performed n times (once for each type).

Table 2. One-way latency (microsec) between different types of nodes in our testbed.

Sender Type	Receiver Type	Time (μsec)
Fast	Fast	170
Fast	Slow	200
Slow	Fast	200
Slow	Slow	230

In heterogeneous environments, there is a possibility of having different types of machines at the sending and receiving sides of a point-to-point communication. Thus, using a simple model for communication cost such as $T_{ptp} = O_{constant} + O_{per-byte} \cdot m$ with the same $O_{constant}$ and $O_{per-byte}$ for all pairs of nodes will not be sufficient.

3 Modeling Collective Communication Operations

The MPI standard provides a rich set of collective communication operations. Different implementations of MPI use different algorithms for implementing these operations. However, most of the implementations, specially those used for NOW environments, implement the different collective operations on top of point-to-point operations. Different types of trees (e.g. binomial trees, sequential trees, and k-trees) can be used for implementing these operations [6, 9, 10, 19]. We can classify the MPI collective communication operations into three major categories: *one-to-many* (such as MPI_Bcast and MPI_Scatter), *many-to-one* (such as MPI_Gather), and *many-to-many* (such as MPI_Allgather and MPI_Alltoall). We present the analysis for some of the representative operations.

In the following sections, we provide analytical models for the broadcast, scatter, and gather operations as they are implemented in MPICH. We also verify the accuracy of our analytical models by comparing the estimated times with measured times on the experimental testbed. It should be noted that our model does not consider the effect of contention and is applicable only to fully-connected systems.

3.1 Broadcast and Multicast

Binomial trees are used by MPICH for implementing broadcast and multicast. Figure 1a illustrates how broadcast is performed on a four-node system. The completion

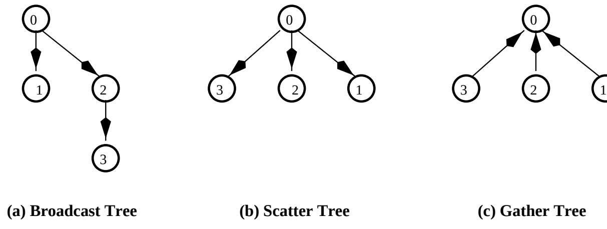

Figure 1. Trees used in MPICH for the broadcast, scatter, and gather operations. The numbers inside the tree nodes indicate the rank of the computing nodes assigned to those tree nodes.

time of broadcast on a n-node system can be modeled by using the following expression:

$$T_{broadcast} = max\{T^0_{recv}, T^1_{recv}, \cdots, T^{n-1}_{recv}\} \quad (6)$$

where T^i_{recv} is the time when node i receives the entire message. The value of T_{recv} for the root of broadcast is obviously zero, and T_{recv} for all other nodes can be calculated from the following recurrence:

$$\begin{aligned} T^i_{recv} &= T^{parent(i)}_{recv} + \\ & childrank(parent(i), i) \cdot \\ & \quad (S^{parent(i)}_c + S^{parent(i)}_m \cdot msg\_size) + \\ & X_m \cdot msg\_size + X_c + \\ & R^i_m \cdot msg\_size + R^i_c \quad (7) \end{aligned}$$

where $parent(i)$ indicates the parent of node i in the broadcast tree and $childrank(parent(i), i)$ is the order, among its siblings, in which node i receives the message from its parent.

This model can be used to estimate the completion time of a broadcast operation on a given heterogeneous system. In order to verify the accuracy of this model, we also measured the completion time of MPI_Bcast on our experimental testbed. The procedure in Figure 2 was used for measuring the completion time of MPI_Bcast (or other MPI collective operations). In order to minimize the effect of external factors, this procedure was executed several times and the minimum of the measured times is reported.

The comparison between the measured and estimated completion times of broadcast on a four-node system with four different configurations is shown in Figure 3. It can be observed that the estimated times using our analytical models are very close to the measured times. It is also interesting to note that the completion time of broadcast for a system with two fast nodes and two slow nodes varies with the order of the nodes in the binomial tree (configurations B and D).

```
MPI_Barrier
get start_time
for (i=0; i < ITER; i++) {
   MPI_Barrier
}
get end_time
barrier_time= (end_time - start_time) / ITER
MPI_Barrier
get start_time
for (i=0; i < ITER; i++) {
   MPI_Bcast /* or any other collective operation */
   MPI_Barrier
}
get end_time
local_time= (end_time - start_time) / ITER
global_time= reduce(local_time, MAXIMUM)
broadcast_time= global_time-barrier_time
```

Figure 2. Outline of the procedure used for measuring the completion time of broadcast and other collective operations.

Observation 2 *The completion time for a given broadcast tree depends not only on the type of participating nodes, but also on how these nodes are assigned to the broadcast tree nodes.*

3.2 Scatter

Scatter is another one-to-many operation defined in the MPI standard. Sequential trees are used to implement scatter in MPICH[1]. Figure 1b illustrates how scatter is implemented on a four-node system. The completion time of scatter on an n-node system can be modeled as follows:

$$\begin{aligned} T_{scatter} &= max\{T^0_{recv}, T^1_{recv}, \cdots, T^{n-1}_{recv}\} \quad (8) \\ T^i_{recv} &= childrank(root, i) \cdot \\ & \quad (S^{parent(i)}_c + S^{parent(i)}_m \cdot msg\_size) + \\ & X_m \cdot msg\_size + X_c + \\ & R^i_m \cdot msg\_size + R^i_c \quad (9) \end{aligned}$$

Figure 4 compares the estimated completion times (based on Equation 8) of the scatter operation with those measured on the experimental testbed with four nodes and different configurations. It can be observed that the estimated times (using our analytical models) are very close to the measured times.

[1]It should be noted that in the scatter operation, as implemented in MPICH, a message is sent from the root to itself through point-to-point communication. Since the time for this operation is small in comparison with the total completion time of scatter we ignore it in our analysis. We use the same approach when we model the gather operation.

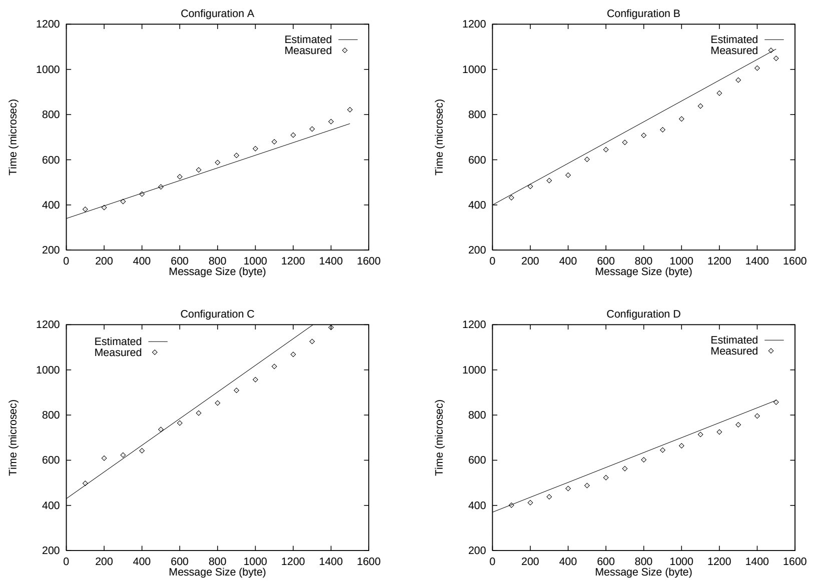

Figure 3. Estimated and measured completion times for the broadcast operation on four nodes with four different configurations. The types of the participating nodes from node 0 to node 3 (in MPI ranking) are: A: fast, fast, fast, fast; B: fast, fast, slow, slow; C: slow, fast, slow, fast; D: fast, slow, fast, slow.

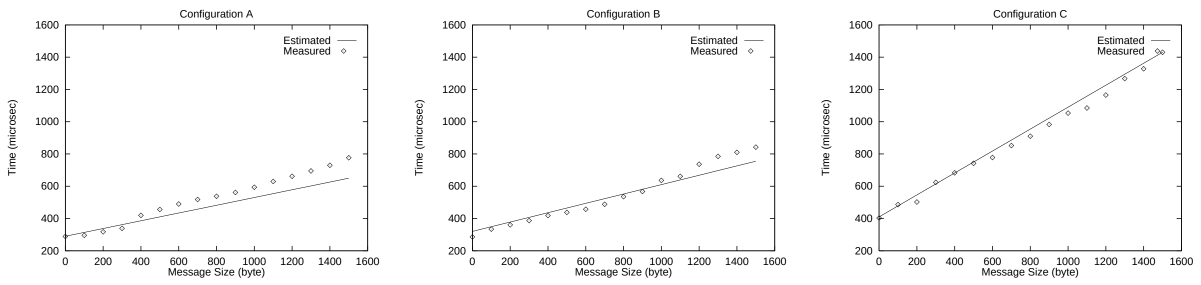

Figure 4. Estimated and measured completion times for the scatter operation on four nodes with three different configurations. The types of the participating nodes from node 0 to node 3 (in MPI ranking) are: A: fast, fast, fast, fast; B: fast, fast, slow, slow; C: slow, fast, slow, fast.

3.3 Gather

Gather is a many-to-one collective operation implemented in MPICH by using reverse sequential trees (Fig. 1c). The completion time of this operation can be modeled by using the following expression:

$$\begin{aligned} T_{gather} &= T_{receive}^{n-1} && (10)\\ T_{receive}^{i} &= max\{T_{receive}^{i-1}, T_{arrive}^{i}\} + \\ & \quad R_m^{root} \cdot msg\_size + R_c^{root} && (11)\\ T_{arrive}^{i} &= (\ The\ ith\ smallest\ element\ of \\ & \quad \{S_c^j + S_m^j \cdot msg\_size\} \\ & \quad for\ 1 < j < n-1\ ) + \\ & \quad i \cdot (X_m \cdot msg\_size + X_c) && (12) \end{aligned}$$

where $T_{receive}^{i}$ is the time by which the messages from i nodes (out of the the $n-1$ sender nodes) have been received at the root node. T_{arrive}^{i} is the time when the ith message has arrived at the root node.

The estimated completion times using our models and the measured completion times for a four-node system with different configurations are shown in Figure 5. It can be seen that the estimated times are close to the measured times.

4 Applying the Communication Model

In this section, we explain how the models developed in the previous sections can be used for evaluating different collective communication algorithms. We also discuss how these models can be used to characterize the effect of the heterogeneous configuration (the number of nodes from different types). Without loss of generality, we consider systems with two different types of computing nodes.

4.1 Choice of Algorithms

Collective communication operations can be implemented by using different algorithms (which use different types of trees). For instance, MPICH and many other communication libraries use the binomial trees for broadcast. However, it has been shown that binomial trees are not the best choice for all systems [2]. Therefore, in order to find the best scheme for a given collective operation, it is important to compare the performance of different schemes. Our proposed communication model can be used to evaluate the performance of these algorithms analytically.

Consider an 8-node system (four fast nodes and four slow nodes with characteristics described in Section 2.2) and three different trees (as shown in Fig. 6) for the broadcast operation. The estimated completion times of broadcast on this system using different tree structures are shown in Figure 7a. It can be seen that for the given configuration, the binomial tree performs worse than the other trees. For messages up to $4K$ bytes in size, the hierarchical tree performs better than others. For messages larger than $4K$ bytes, the sequential tree has the best performance. Figure 7b shows the results for a 16-node system. It can be seen that the hierarchical algorithm outperforms the other two algorithms for the 16-node system.

The models presented in this paper can be used to evaluate the performance of different collective communication operations in heterogeneous environments. These models can be used for comparing the performance of different algorithms for different collective operations and identifying the most efficient algorithms.

4.2 Effect of Configuration on Performance

The proposed communication models can also be used for predicting the effect of configurations in HNOW systems. For instance, consider a system with a set of slow and fast nodes. We are interested in knowing how the performance of collective communication operations varies based on the number of fast/slow nodes in the system. Figure 8 illustrates the estimated completion times of the gather operation for varying number of fast nodes in systems with 8 and 16 nodes, respectively. (The root is always considered to be a fast node.) It can be observed that having more than four fast nodes in these systems does not improve the performance of this operation significantly. The models presented in this paper can be used to evaluate the performance of other collective communication operations and other system configurations.

5 Conclusions and Future Work

In this paper, we have proposed a new model for estimating the cost of point-to-point and different collective operations in the emerging HNOW systems. We have verified the validity of our models by using an experimental heterogeneous testbed. In addition, we have shown how this model can be used to compare different algorithms for different collective operations. We have also shown that this model can be used to predict the effect of having different types of computing nodes on the performance of collective operations.

We plan to evaluate our model by using other promising networking technologies (such as Myrinet and ATM) and underlying point-to-point communication layers (such as FM and U-Net). We also plan to extend our model for systems with heterogeneous networking technologies. We are exploring how this model can be used to predict the execution time of parallel applications on heterogeneous systems. By doing so, we will be able to consider the effect

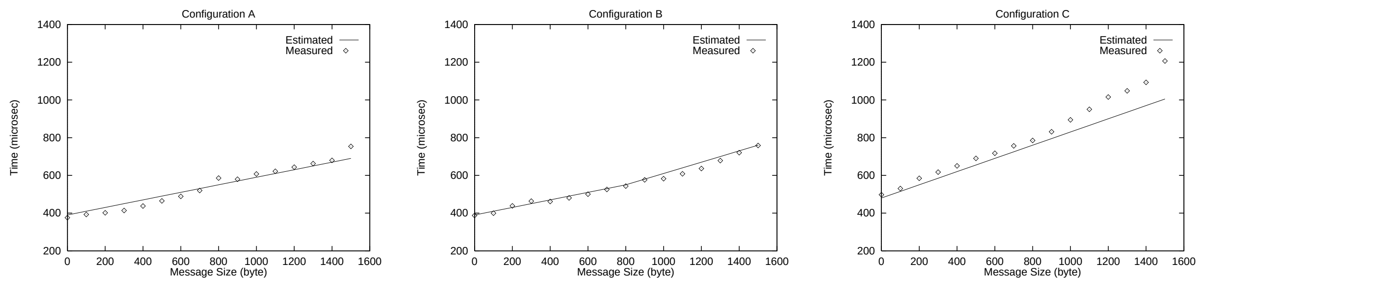

Figure 5. Estimated and measured completion times for the gather operation on four nodes with three different configurations. The types of the participating nodes from node 0 to node 3 (in MPI ranking) are: A: fast, fast, fast, fast; B: fast, fast, slow, slow; C: slow, fast, slow, fast.

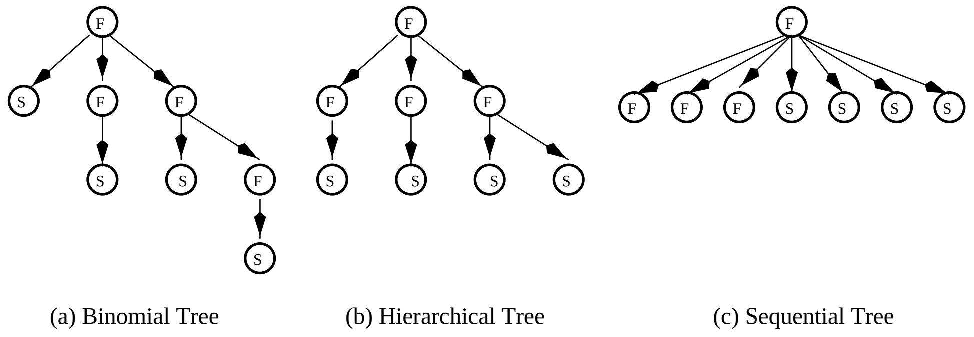

Figure 6. Three different possible trees for implementing broadcast in a HNOW system with eight nodes (F = fast node, S = slow node).

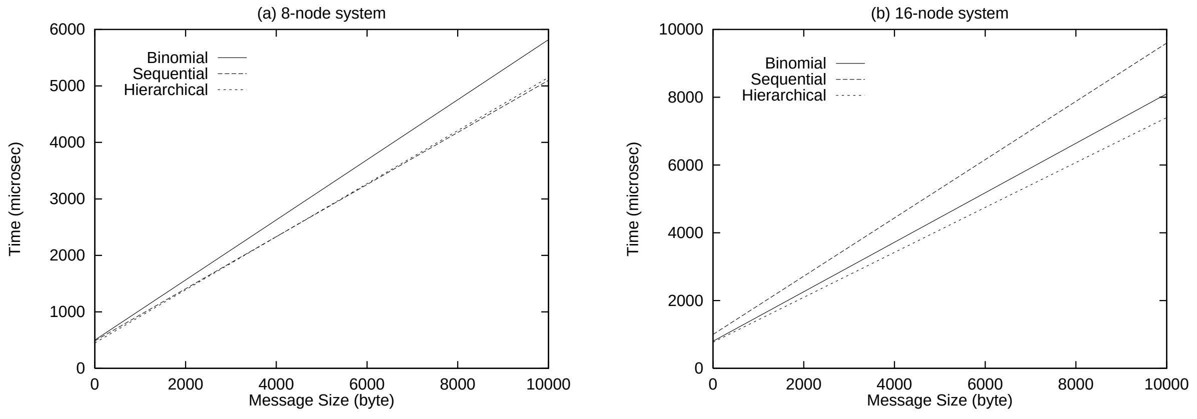

Figure 7. Comparing the performance of three different broadcast algorithms (based on the respective tree structures of Figure 6) on 8-node and 16-node heterogeneous systems.

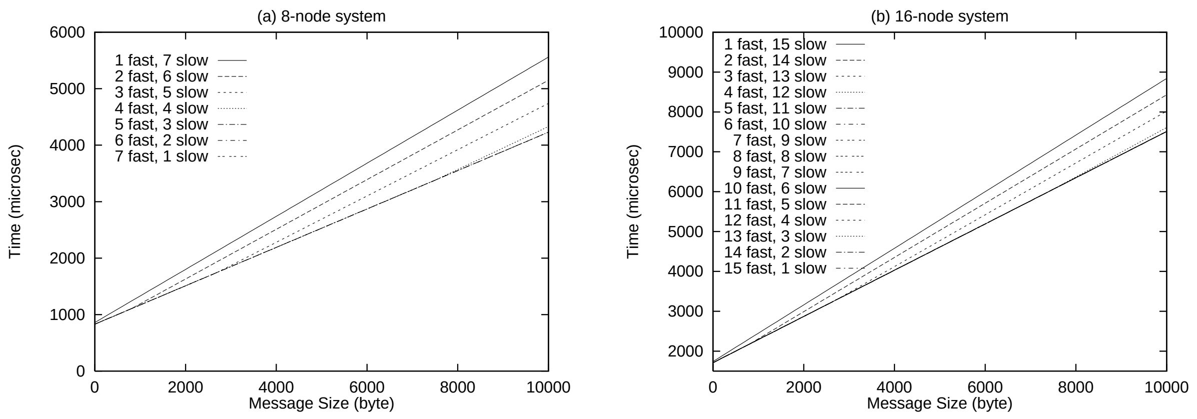

Figure 8. Comparison between the completion times of the gather operation on 8-node and 16-node systems with different number of fast and slow nodes.

of communication time in overall execution time of an application more accurately and based on that come up with better load balancing schemes.

References

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

[2] M. Banikazemi, V. Moorthy, and D. K. Panda. Efficient Collective Communication on Heterogeneous Networks of Workstations. In *International Conference on Parallel Processing*, pages 460–467, 1998.

[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] Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovik, and Wen-King Su. Myrinet — A Gigabit-per-Second Local-Area Network. *IEEE MICRO*, 15(1):29–36, February 1995.

[5] R. V. Boppana, S. Chalasani, and C. S. Raghavendra. On Multicast Wormhole Routing in Multicomputer Networks. In *Symposium on Parallel and Distributed Processing*, pages 722–729, 1994.

[6] 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.

[7] D. Dai and D. K. Panda. Reducing Cache Invalidation Overheads in Wormhole DSMs Using Multidestination Message Passing. In *International Conference on Parallel Processing*, pages I:138–145, Chicago, IL, Aug 1996.

[8] 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.

[9] 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, February 1997.

[10] R. Kesavan and D. K. Panda. Multiple Multicast with Minimized Node Contention on Wormhole k-ary n-cube Networks. *IEEE Transactions on Parallel and Distributed Systems*. in press.

[11] 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.

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

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

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

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

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

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

[18] 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.

[19] 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*, 9(10), October 1998.

[20] Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker, and Jack Dongarra. *MPI: The Complete Reference*. The MIT Press, 1996. Chapter 4 of this book deals with collective communications.

[21] C. B. Stunkel, R. Sivaram, and D. K. Panda. Implementing Multidestination Worms in Switch-Based Parallel Systems: Architectural Alternatives and their Impact. In *Proceedings of the 24th IEEE/ACM Annual International Symposium on Computer Architecture (ISCA-24)*, pages 50–61, June 1997.

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

[23] T. von 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.

Biographies

Mohammad Banikazemi is a Ph.D. student in the Department of Computer and Information Science at The Ohio State University. His research interests include network-based computing, heterogeneous computing, and interprocessor communication. He received an M.S. degree in Electrical Engineering from The Ohio State University in 1996, and a B.S. degree in Electrical Engineering from Isfahan University of Technology, Isfahan, Iran.

Jayanthi Sampathkumar is a graduate student at The Ohio State University. Her research is currently focused on parallel computing. She received a B.Tech. degree form Institute of Technology, Banara Hindu University, India.

Sandeep Prabhu is currently working at Microsoft Corporation. He received a MS degree in computer and information science from The Ohio State University and a B.Tech. in Computer Engineering from Indian Institute of Technology, Bombay, India.

Dhabaleswar K. Panda is an Associate Professor in the Department of Computer and Information Science, The Ohio State University, Columbus, USA. His research interests include parallel computer architecture, wormhole-routing, interprocessor communication, collective communication, network-based computing, and high-performance computing. He has published over 60 papers in major journals and international conferences related to these research areas.

Dr. Panda has served on Program Committees and Organizing Committees of several parallel processing conferences. He was a Program Co-Chair of the 1997 and 1998 Workshops on Communication and Architectural Support for Network-Based Parallel Computing and a co-guest editor for special issue volumes of the *Journal of Parallel and Distributed Computing* on “Workstation Clusters and Network-based Computing”. Currently, he is serving as a Program Co-Chair of the 1999 International Conference on Parallel Processing, an Associate Editor of the *IEEE Transactions on Parallel and Distributed Computing*, an IEEE Distinguished Visitor Speaker, and an IEEE Chapters Tutorials Program Speaker. Dr. Panda is a recipient of the NSF Faculty Early CAREER Development Award in 1995, the Lumley Research Award at Ohio State University in 1997, and an Ameritech Faculty Fellow Award in 1998. He is a member of the IEEE, the IEEE Computer Society, and the ACM.

P. Sadayappan is a Professor of Computer and Information Science at The Ohio State University. He obtained a B. Tech. in Electrical Engineering from I.I.T. Madras, India, and an M.S. and Ph. D. in Electrical Engineering from S.U.N.Y at Stony Brook. His research interests include network-based computing and high-performance scientific computing.