High Performance and Reliable NIC-Based Multicast over Myrinet/GM-2[*]

Weikuan Yu Darius Buntinas Dhabaleswar K. Panda

Network-Based Computing Laboratory
Dept. of Computer and Information Science
The Ohio State University
{yuw, buntinas, panda}@cis.ohio-state.edu

Abstract

Multicast is an important collective operation for parallel programs. Some Network Interface Cards (NICs), such as Myrinet, have programmable processors that can be programmed to support multicast. This paper proposes a high performance and reliable NIC-based multicast scheme, in which a NIC-based multi-send mechanism is used to to send multiple replicas of a message to different destinations, and a NIC-based forwarding mechanism to forward the received packets without intermediate host involvement. We have explored different design alternatives and implemented the proposed scheme with the set of best alternatives over Myrinet/GM-2. MPICH-GM has also been modified to take advantage of this scheme. At the GM-level, the NIC-based multicast improves the multicast latency by a factor up to 1.48 for messages $\leq$ 512 bytes, and a factor up to 1.86 for 16KB messages over 16 nodes compared to the traditional host-based multicast. Similar improvements are also achieved at the MPI level. In addition, it is demonstrated that NIC-based multicast is tolerant to process skew and has significant benefits for large systems.

1. Introduction

Multicast is an important collective operation in parallel and distributed programs. Message passing standards, such as MPI [10], often have the multicast operation, also called broadcast, included as a part of their specifications. Some interconnects such as QsNet [12], Infiniband [8] and BlueGene/L [9], provide hardware primitives to support multicast communication. Other interconnects, such as Myrinet, do not have hardware multicast and provide unicast communication along point-to-point links. Thus a multicast operation is usually implemented at the user level with unicast operations. Such an approach can lead to higher multicast latency. Thus it would be beneficial to reduce the latency of this operation as much as possible. Some network interface cards (NICs) have programmable processors which can be customized to support collective communication. This type of NIC support has been studied in [14, 2, 5, 6, 4]. Among them, NIC-supported multicast has been studied with several different schemes, namely, FM/MC by Verstoep [14], LFC by Bhoedjang [2] and a NIC-assisted scheme by Buntinas [5].

In this paper, we start with the characterization of features that are important to NIC-based multicast. These features include reliability, forwarding, scalability, protection, and tree construction etc. In this context, we analyze the existing multicast schemes [14, 2, 5] and determine that each of them lacks one or more features. Accordingly, we propose a new NIC-based multicast scheme that provides a complete set of features. We then explore different design alternatives and implement the proposed scheme with the set of best alternatives. The NIC-based scheme is used together with an optimal spanning tree [1] to support multicast to an arbitrary set of nodes in a system. Our evaluation indicates that the NIC-based multicast scheme achieves significant improvement over the traditional host-based multicast. We have also modified MPICH-GM to use this scheme and observed that it significantly reduces broadcast latency and benefits large size systems with reduced effects of process skew.

The rest of the paper is structured as follows. In the next section, we describe and characterize general features of NIC-supported multicast schemes, followed by Section 3 where we describe our scheme. In Section 4, we give an overview of Myrinet and GM. In Section 5, we describe design issues and implementation details for the NIC-based multicast scheme, as well as the modification of MPICH-GM. We then present the performance results and evaluation of our implementation in Section 6. Finally, we conclude the paper in Section 7.

2. NIC-based Multicast

In this section we describe and characterize the features of NIC-supported multicast schemes. We identify the following features that are important to the multicast in modern parallel systems:

Tree Construction, Tree information and Forwarding – Broadcast/multicast on point-to-point net-

[*]This research is supported in part by a DOE grant #DE-FC02-01ER25506 and NSF Grants #EIA-9986052 and #CCR-0204429.

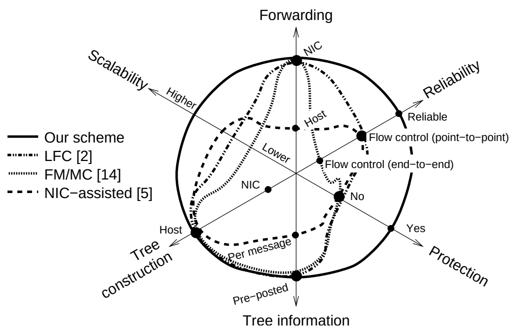


Figure 1. An axes diagram comparing the features of different multicast schemes.

works is typically done by having the message forwarded to the destinations along a spanning tree [7]. Three major features of a multicast scheme are the spanning tree construction, how the tree information is specified and how messages are forwarded. Note, a node in a network consists of the host and the NIC. The spanning tree can either be constructed at the host or at the NIC. Since the NIC processor is typically much slower than the host processor, it is more efficient to construct the tree at the host. The tree information can be either specified with each message [13, 5] or preposted to the NIC [14, 2]. With preposted tree information, at intermediate nodes, a message does not have to wait for the host process to provide the tree information before it can be forwarded, which can lead to reduced host involvement. Message forwarding can either be done by the NIC or by the host. Using host-based forwarding, a node must pass the received message to the host first and then back to the NIC for forwarding. This leads to a large overhead to the multicast latency.

Reliability – Reliability is also important to a multicast scheme, which can be done either directly or indirectly. A direct scheme uses acknowledgments to confirm the delivery, and timeout/retransmission to deal with the loss of messages, while an indirect scheme typically assumes the network is reliable and uses a credit-based scheme to manage the receive buffer. In general, however, a network cannot be considered reliable. Though bit error-rates are low in modern networks, they are not zero. There are also drawbacks with credit-based schemes. A centralized credit scheme has a bottleneck at the centralized component. A distributed credit scheme, in which the credits are managed from hop to hop, can lead to deadlock since a multicast message may be initiated by the root, while an intermediate node is running out of credits to forward it.

Protection and Scalability – Depending on the implementation, a multicast scheme may or may not provide protection of concurrent NIC access by several processes. Without protection, a user process may modify the NIC-memory used by another process, which can lead to unpleasant scenarios. In addition, high scalability has since been a desirable feature of parallel systems, and it becomes indispensable, as the number of nodes reaches thousands in a cluster.

Figure 1 shows a diagram, which uses six axes to represent these features, and compares the features of available multicast schemes, as well as the scheme we are proposing in this paper. In this diagram, a line is used to connect the points on the axes to describe the features of a particular scheme. To be efficient in tree construction, all these schemes have the host construct the spanning tree. The NIC-assisted scheme [5] specifies the tree information along with the message, but it requires the intermediate host involvement to perform the message forwarding. FM/MC [14] provides an end-to-end flow control with host-level credits. A centralized credit manager is used to recycle multicast credits, which does not scale. LFC [2] provides link-level point-to-point flow control with NIC-level credits. But it is deadlock prone since a multicast packet may be injected into the network by the root, while an intermediate NIC is running out of credits to forward the message.

In this paper, we propose a high performance and reliable NIC-based multicast scheme with features such as NIC-based forwarding, protection of concurrent NIC-memory access between processes, tree construction at the host and preposting of the tree information to the NIC. We propose to implement this scheme over Myrinet/GM. GM [11] is a user-level protocol that provides a reliable ordered delivery of packets with low latency and high bandwidth. It can support clusters of over 10,000 nodes and concurrent memory-protected OS-bypass access to the NIC by several user-level applications. By modifying GM to support the NIC-based multicast while maintaining the original features of GM, it is possible to achieve our proposed scheme. Recent alpha releases of GM-2.0 [11] provide a myrinet packet descriptor for every network packet and also a callback handler to each descriptor. A packet descriptor and its callback handler provide a way to take necessary actions on this packet when appropriate. We have implemented the proposed scheme using these features, which results in a high-performance and reliable multicast scheme, supporting concurrent memory-protected access to the NIC.

3. Our Scheme

In this section, we describe our proposed scheme. For a tree-based broadcast/multicast operation over point-to-point networks multiple copies of the data will be transmitted by root nodes of the tree or subtrees to their children. In our scheme, we use a NIC-based mechanism, by which a message is transfered only once from the host to the NIC, and from the NIC multiple replicas are transmitted to a set of destinations, referred to as the NIC-based *multisend* operation. For a multicast operation, the NIC at the intermediate nodes, when having received the message, forwards it without the host involvement. This leads to reduced latency. We describe our scheme and its benefits over a host-based multicast as we look into the process of the multicast communication.

NIC-Based Multisend and its Benefits – Figure 2a shows abstract timing diagrams for sending a message to four destinations with a host-based mechanism. The host-based sending can be broken into three steps. In the

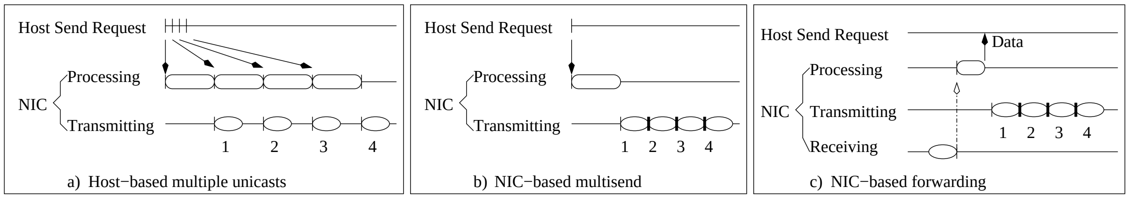


Figure 2. Abstract Timing Diagrams Comparing Host-Based Multicast and NIC-Based Multicast

figure, three lines in parallel represent the timing of the three steps. First, the host posts four send requests. At the second step, the NIC processes the requests sequentially, during which the messages are downloaded from the host and queued for transmission. At the third step, the NIC transmit DMA engine completes the transmission of a queued message. As shown in the figure, when the host posts four requests, the NIC has to repeat the second processing step four times. If the processing of a request can not be overlapped with the transmission time of a queued message, it results in a higher latency. With a NIC-based mechanism, we can avoid repeated processing. Figure 2b shows a corresponding timing diagram for the NIC-based mechanism. The host posts only one multisend request. The NIC then finds a corresponding list of destinations and queues the message for transmission to the first destination. When that transmission completes, the NIC modifies the packet header and queues it for transmission to another destination, and so on. The same data is transmitted again with a small overhead, represented in the figure with the wide bars. However, the repeated transmission of the message does not have to wait for the processing of a host request. A NIC-based multisend operation can potentially transmit the messages at a faster speed, resulting in a lower latency.

NIC-Based Forwarding and Its Benefits – With a host-based mechanism, the intermediate host initiates another set of unicasts after receiving the message. A message just received by the NIC must be copied into the host memory and then back to the NIC for forwarding. This leads to a large overhead to the latency of the multicast. Figure 2c shows the timing diagram for forwarding a message with a NIC-based approach. When having received a multicast packet, the intermediate NIC looks into its table to find a list of destinations for that packet. This packet can then be queued for forwarding with a changed header. Thus the overhead at the intermediate host to receive the message and initiate the forwarding is eliminated. For multiple packet messages, using NIC-based forwarding an intermediate NIC can forward the packets of a message without waiting for the arrival of the complete message. This can further reduce the multicast latency.

Tolerance to Process Skew – The NIC-based multicast scheme also has the potential of reducing the effects of process skew. When processes in a parallel program are skewed, or unsynchronized, they may not reach the same point of computation at the same time. For a broadcast operation, some processes call the broadcast earlier, while others do after some delay. As the traditional host-based broadcast is implemented, an intermediate process will not forward the message until it calls the broadcast and receives the message. A process could be forced to wait longer if one or more of its ancestors are lagging behind due to some skew. With the NIC-based approach, the message can be forwarded by an intermediate NIC to its children even if the host process has not called the broadcast. So a delayed process will not keep the processes lower in the tree from receiving the messages. Therefore the effects of process skew to the overall multicast performance can be reduced.

4. Overview of Myrinet and GM

In this section, we give some background information on Myrinet and GM. Myrinet is a high-speed interconnect technology using wormhole-routed crossbar switches to connect all the NICs. GM is a user-level communication protocol that runs over the Myrinet [3] and provides a reliable ordered delivery of packets with low latency and high bandwidth.

Sending a Message – To send a message, A user application calls the appropriate function from the library. This function constructs a send descriptor, referred to as a *send event* in GM, which describes what data to be sent and to which process to send the data to. The NIC translates the event to a *send token* (a form of send descriptor that NIC uses), and appends it to the send queue for the desired destination. For each send token, the NIC will DMA the data from the host buffer into a *send buffer* and transmit the message on a per packet basis. The NIC keeps a *send record* of the sequence number and the time for each packet it has sent. The send record will be removed when a corresponding acknowledgment is received. If the acknowledgment is not received within the timeout time, the sender will retransmit the packet. When all the send records are acknowledged, the NIC will pass the send token back to the host.

Receiving a Message – To receive a message, the host provides some registered memory as the receive buffer by preposting a receive descriptor. A posted receive descriptor is translated into a *receive token* by the NIC. When the NIC receives a packet, it checks the sequence number. If it is expected, the NIC locates a receive token, DMAs the packet data into the host memory, and also acknowledges the sender. When all the packets for a message have been received, the NIC will also generate a *receive event* to the host process for it to detect that a message has been received.

New Features of GM-2 – In recent alpha releases of GM-2.0, a data structure called the myrinet packet descriptor, is introduced to describe every network packet. Inside this structure, there is also a callback handler, which allows the possibility of taking actions on the packet. Either the packets that have data to be transmitted at the send side or the packets that have data to be copied to the final location at the receive side, are queued using the descriptors. At the completion of transmission or copying, the packets are freed by freeing the descriptors. Using the descriptor and its callback handler, one can easily have a packet queued again for transmission before it is freed. For example, to send a replica to another destination, a callback handler can change the packet header and queue it for transmission again. This can be done to both a send packet and a receive packet.

5. Design Issues and Our Implementation

In this section, we describe the design issues and the implementation details of the NIC-based multicast. We implemented the proposed scheme by modifying GM version 2.0_alpha1. There are several design issues for this implementation: the sending of message replicas to multiple destinations, message forwarding at the intermediate NIC, reliability and in order delivery, deadlock, and construction of the spanning tree. For each of these issues, we describe design alternatives below and show how we choose the best alternative. At the end of this section, we describe the modification to MPICH-GM for taking advantage of the NIC-based multicast.

Sending of Multiple Message Replicas – To send replicas of a message to multiple destinations, one can readily generate multiple send tokens and queue them to multiple destinations. Another way to do this is to use a callback handler as just described in the previous section. A third way to do this is to change the header right after the transmit DMA engine is done transmitting the header and queue the packet again for transmission. The first approach performs the processing for each of the tokens, and it saves nothing more than the posting of multiple send events. The benefits of this is no more than 1μs, if any, since the host overhead over GM is less than 1μs. Both the second and third approach can save the repeated processing, but the third approach takes special care and demands good timing strategy in order to avoid clobbering the packet header before it is transmitted out. We implemented the second approach in our multicast scheme. The benefits of the third approach could be more, but we decided to leave it for later research.

Messages Forwarding – For a received message to be forwarded, we need to consider: 1) how to set up timeout and retransmission mechanisms, and 2) which replica of the message should be made available for the retransmission. As to the first issue, we create send records to record the time the packets are forwarded. When the records are not acknowledged within the timeout period, retransmission of the packets is triggered. Since the intermediate NIC does not have a send token for this multicast, one has to generate a token for the purpose of transmission. This can be done by grabbing a send token from the free send token pool, or by transforming the receive token into a send token. Using the former approach can lead to the possibility of deadlock when the intermediate nodes are running out of send tokens. We take the second approach since it does not require additional resources at the NIC. The receive token is presumed to be available to receive any message. In this approach, the receive token is used for transferring the data to the host at the intermediate NIC, and is also used to retransmit the message when timeout. As to the second issue, a naive solution would be keeping the received packet available until all the children acknowledge the transmission. The problem with this approach is that the NIC receive buffer is a limited resource, and holding on to one or more receive buffer will slow down the receiver or even block the network. An alternative is to release the packet as the forwarding is done, and use the message replica in the host memory for retransmission. Since GM can only send and receive data from registered memory, this requires the host memory to be kept registered until all the children acknowledged that the packets are correctly received. We take the second alternative in our implementation.

Reliability and In Order Delivery – To ensure ordered sending, GM employs a form of Go-back-N protocol to ensure ordered delivery between peer-to-peer communication end points, called ports. When a packet is not acknowledged within a timeout period, the sender NIC will retransmit the packet, as well as all the later packets from the same port.

A reliable ordered multicast requires modification to the existing ordering scheme. Since each sender is involved with multiple receivers, the sending side must keep track of the ordering of packets in a one-to-many manner to all its children. A modified ordering scheme works as described below. Multicast *send token*s are queued by group. Each multicast group has a unique group identifier. For each group, the NIC keeps tracks of: 1) a receive sequence number to record the sequence number for the packets received from its parent, 2) a send sequence number to record the packets that have been sent out, and 3) an array of sequence numbers to record the acknowledged sequence number from each child. A multicast packet sent from one NIC to its children has the same sequence number and send record, ensuring ordered sending for the same group's multicast packets. When an acknowledgment from one destination is received, the acknowledged sequence number for that destination is updated. If the record for a packet is timed out, the retransmission of the packet and the following ones will be performed only for the destinations which have not acknowledged. A receiver only acknowledges the packets with expected sequence numbers for the desired group sequentially.

Deadlock – Deadlock is an important aspect of concern for any collective communication, which may occur if there is a cyclic dependence on using some shared resources among multiple concurrent operations. We take the following approaches to avoid the possibilities of the deadlock. First, we do not use any credit-based flow control, avoiding one source of deadlock. In addition, we provide a unique group identifier and a separate queue for each multicast group with a sender, so that one group does not block the progress of another. The other possibility for a deadlock is when some nodes in multiple broadcast operations form a cyclic parent-child relationship, in which all of them are using its last receive token while requesting another to receive its message with a new receive token. Since the root node in a broadcast operation only uses its send token, it will not be in such a cycle. To break a possible cycle among the rest of the nodes, we sort the list of destinations linearly by their network IDs before tree construction, and a child must have a network ID greater than its parent unless its parent is the root. Thus a deadlock on the use of receive token can not form under either situation (See [15] for more information on the deadlock). As long as receive tokens are available at the destinations, multicast packets can be received by all the destinations. The responsibility of making receive tokens available to receive multicast messages is left to client programs, the same way as is required to receive regular point-to-point messages.

The Spanning Tree – The tree topology is also important for multicast performance. One tree topology may give better performance over another depending on the communication characteristics and also the desired performance metrics, latency or throughput. The performance of logical tree topology can be affected by the underlying hardware topology. In this paper, our intent is not to study the effects of hardware topology. In this study, Myrinet network uses its default hardware topology, Clos network. One relevant design issue in this paper is *where* to generate the tree, since the NIC processor is typically rather slow to perform intensive computation. To better expose the potential of the NIC-based multicast protocol, we use an algorithm similar to [5] for constructing an optimal tree in terms of latency. The optimality of such trees has been shown by Bar-Noy and Kipnis [1]. The basic idea of constructing an optimal tree is to have maximum number of nodes involved in sending at any time. In other words, we construct the tree such that a node will send to as many destinations as possible before the first destination it sent to becomes ready to send out data to its own children. We compute the number of destinations a sender can send to before its first receiver can start sending as the ratio of: (a) the total amount of time for a node to send a message until the receiver receives it, and (b) the average time for the sender to send a message to one additional destination. The message delivery time is calculated as end-to-end latency. Different message lengths leads to different optimal tree topologies. Since the LANai processor is much slower compared to the host processor, we carried out the following division of labor in order to be efficient on tree construction: the host generates a spanning tree and inserts it into a group table stored in the NIC and the NIC is responsible for the protocol processing related to communication.

Incorporating into MPICH-GM – MPICH is a widely used MPI implementation. MPICH-GM is a port of MPICH on top of GM. We investigated how MPICH-GM can benefit from the NIC-based multicast. Since MPICH-GM uses remote DMA operations in its rendezvous protocol for transferring messages larger than 16K, our modified channel interface still performs broadcast operation in its original manner for these messages. For messages less than 16K, the first broadcast operation from a particular root in a communicator will cause a new group context to be created and the group membership to be updated into the NIC. Thus the first broadcast operation for any group will pay the cost of creating group membership, we consider this demand-driven approach to be appropriate for dealing with the vast number of possible combinations of communicators and root nodes. Once the unique group context is created or identified, the root node initiates a NIC-based multicast operation, while the destinations invoke blocking receive operations as MPI_Recv does.

6. Performance Evaluation

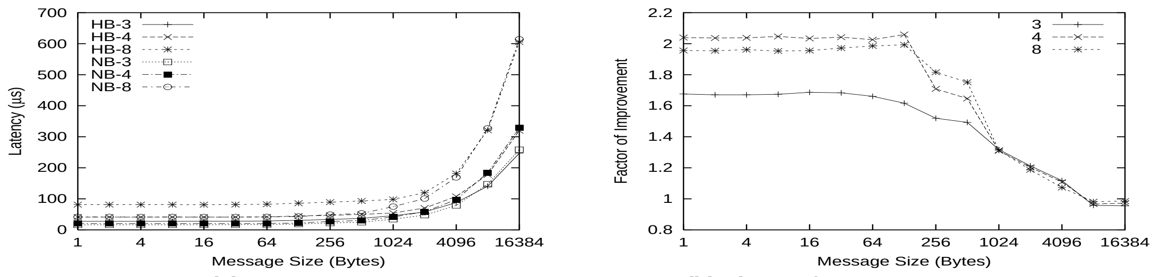
In this section, we describe the performance evaluation of our implementation. The experiments were conducted on a 16 node cluster consisting of 16 quad-SMP 700 MHz Pentium-III nodes with 66MHz/64bit PCI bus. The nodes have Myrinet NICs with 133MHz LANai 9.1 processors and are connected to a Myrinet 2000 network. Each of these nodes run the 2.4.18 Linux kernel. We compared our NIC-based implementation, which is based on GM-2.0_alpha1, to the host-based implementation using the same version of GM. MPICH-GM version 1.2.4..8a was modified to use the NIC-based multicast. The same version of MPICH-GM was used as the comparison at the MPI-level.

6.1. GM Level

Our modification to GM was done by leaving the code for other types of communications mostly unchanged. The evaluation indicated that it has no noticeable impact on the performance of non-multicast communications.

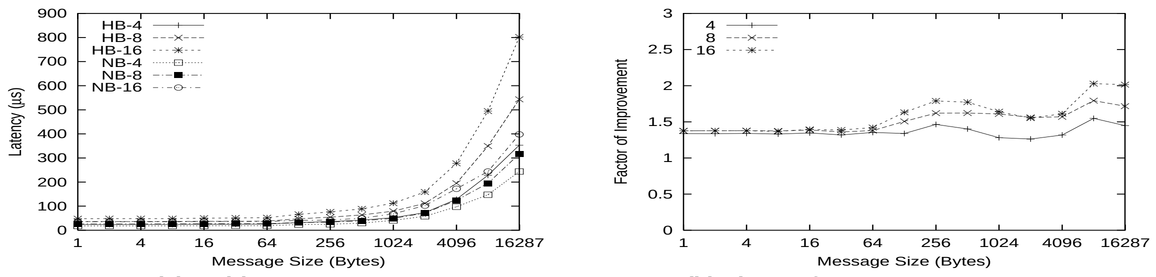
NIC-based multisend – We first evaluated the performance of the NIC-based multisend operation. Our tests were conducted by having the source node transmit a message to multiple destinations, and wait for an acknowledgment from the last destination. All destinations received the message from the source node, and none of them forwarded the message. The first 20 iterations were used to synchronize the nodes. Then the average for the next 10,000 iterations was taken as the latency. Figures 3(a) and 3(b) show the performance and the improvement of using the NIC-based multisend operation to transmit messages to 3, 4 and 8 destinations, compared to the same tests conducted using host-based multiple unicasts. For sending messages $\leq$ 128 bytes to 4 destinations, an improvement factor up to 2.05 is achieved. This is due to the fact that the NIC-based multicast was able to save repeated processing. As the message size gets larger, the improvement factor decreases and eventually levels off at a little below 1. This is to be expected because large message sizes leads to longer transmission time. With host-based multiple unicasts, the request processing is completely overlapped with the transmission of a previous queued packet, but there is still an overhead each time the packet header is changed with the NIC-based multisend.

NIC-based multicast – We evaluated the performance of the multicast with NIC-based forwarding us-



(a) Latency **(b) The Performance Improvement**

Figure 3. The performance of the NIC-based (NB) multisend operation, compared to Host-based (HB) multiple unicasts



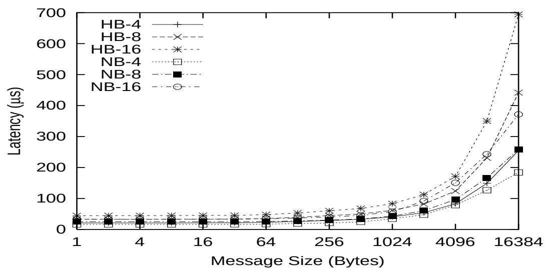
(a) Multicast Latency **(b) The Performance Improvement**

Figure 4. The MPI-level performance of the NIC-based (NB) multicast, compared to the host-based multicast (HB), for 4, 8 and 16 node systems

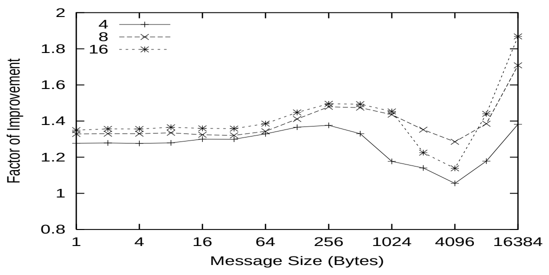
ing an optimal tree. Our tests were conducted by having the root initiate the NIC-based multicast operation, and wait for an acknowledgment from one of the leaf nodes in the spanning tree. The first 20 iterations were used to synchronize the nodes. Then 10,000 iterations were timed to take the average latency. The same test was repeated with different leaf nodes returning the acknowledgment. The maximum from all the tests was taken as the multicast latency. The traditional host-based multicast was also evaluated in the same manner using the same version of GM as a comparison. Figures 5(a) and 5(b) show the performance of the NIC-based multicast compared to the performance of host-based multicast. For broadcasting messages $\leq$ 512 bytes on a 16 node system, the NIC-based multicast achieves an improvement factor up to 1.48. Because multiple replicas of small messages can be sent out faster with the NIC-based multicast, the optimal tree constructed for small messages has a larger average fan-out degree and so a shallower depth, compared to the same size binomial tree used in the traditional host-based multicast. The average fan-out degree is the ratio as described in Section 5, the spanning tree, and it imposes little impact on the latency. So the shallower depth reduces the multicast latency significantly. As also shown in the figures, when broadcasting a 16KB message on a 16 node system, the NIC-based multicast achieves an improvement factor up to 1.86. This is due to the fact that, in the NIC-based multicast, intermediate nodes do not have to wait for the arrival of the complete message to forward it. Thus the NIC-based multicast achieves its performance benefits for the reduced intermediate host involvement and the capability of pipelining messages. Moreover, Figure 5(b) shows dips in the improvement factor curves when multicasting 2KB and 4KB messages. The drop of improvement for these message is because these messages do not have the benefit for large multiple packet messages and also they do not have the benefit for small messages. The maximum packet size in GM is 4096 bytes, therefore $\leq$ 4096 byte messages do not benefit from message pipelining. On the other hand, since the NIC-based multisend does not have much improvement for these $\geq$ 1KB messages (See Figure 3(b)), the fan-out degree chosen in the optimal tree is about 1 and the shape of the resulted optimal tree is not significantly different from the binomial tree used in the host-based approach. Therefore for these messages, the multicast latency does not benefit much from the change of the spanning tree shape either. Taken together, the performance improvement is low for multicasting these messages.

6.2. MPI Level

Since our modification to MPICH-GM only uses the NIC-based multicast support for the eager mode message passing, the largest message that uses the NIC-based multicast is the largest eager mode message, which is 16,287 bytes. We measured the broadcast latency at the MPI level in the same manner as that at the GM level. The maximum latency obtained was taken as the broadcast latency. Figures 4(a) and 4(b) show the latency performance and the improvement factor of the NIC-based multicast at the MPI level, respectively. We observed an improvement factor of up to 2.02 for broadcasting 8KB messages over 16 node system. Also the

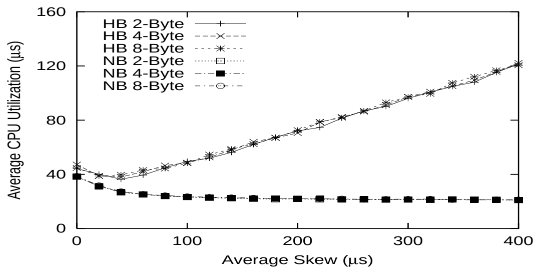


(a) Multicast Latency

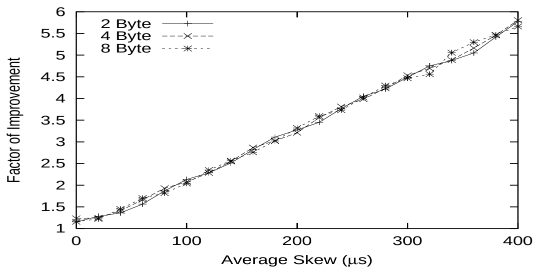


(b) The Performance Improvement

Figure 5. The GM-level performance of the NIC-based (NB) multicast, compared to the host-based multicast (HB), for 4, 8 and 16 node systems



(a) Average Host CPU Time



(b) Improvement Factors

Figure 6. Average host CPU time on performing the MPI_Bcast under different amount of average skew with both the host-based approach (HB) and the NIC-based (NB) approach

trend of the performance improvements are similar to the trend at the GM-level (Figure 5(b)). However, when broadcasting 16,287 byte messages, there is a dip in the improvement factor curve. That is due to the larger cost of copying the data to their final locations. So the broadcast latency for a 16,287 byte message with the NIC-based multicast is comparatively high, which leads to a lower improvement factor.

6.3. Tolerance to Process Skew

Another major benefit of the NIC-based multicast is the tolerance to process skew. Typically, with the blocking implementation of MPI_Bcast, the host CPU time, the time spent on performing the MPI_Bcast, becomes larger if a process is delayed at an intermediate node. In reality, all processes skew at random. Some processes call MPI_Bcast before the root node does, and others do after the root node. The effects of the former can not be reduced by a multicast operation, but those from the latter can be reduced if possible, because all the processes that have called MPI_Bcast inevitably have to wait for the root process. We evaluate the effects of the delayed processes, relative to the root processes, to the average host CPU time. We measure the average host CPU time to perform the MPI_Bcast with varying amount of process skew. All the processes are first synchronized with a MPI_Barrier. Then each process, except the root, chooses a random number between the negative half and the positive half of a maximum value as the amount of skew they have. The processes with a positive skew time perform computation for this amount of skew time before calling the MPI_Bcast operation. The average host CPU time from 5,000 iterations was plotted against the average process skew (See [15] for more information on the process skew).

Figure 6(a) shows the average host CPU time for broadcasting small messages (2, 4 and 8 bytes) over 16 nodes with varying amount of average skew. The NIC-based broadcast has much smaller host CPU time compared to the host-based broadcast. With a skew under 40μs, the host CPU time decreases using either approach. This is to be expected because a small amount of skew time can overlap with some of the message broadcasting time. When the skew goes beyond 40μs, the host CPU time increases with the host-based approach, while it decreases with the NIC-based approach. This is to be expected. As the skew increases, more intermediate processes get delayed. With the host-based approach, more processes wait longer for their ancestors to call MPI_Bcast and forward the messages, which results in longer average host CPU time. In contrast, with the NIC-based approach, the delayed intermediate processes does not prevent their children from receiving the message and, on the other hand, their delay have more overlap with the message transmission time. which leads to less average host CPU time. Figure 6(b) shows that the improvement factor of the NIC-based approach over the host-based approach for small messages. With an average skew of 400μs, the NIC-based multicast achieves an improvement factor up to 5.82. We also observed that the improvement factor becomes greater as the skew increases. When broadcasting large messages (2KB to 8KB), a similar trend of benefits on average host CPU time is also observed when comparing the NIC-based multicast to the host-based broadcast (See [15] for corresponding graphs and explanations).

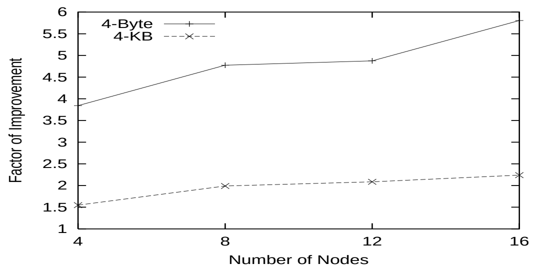


Figure 7. The effect of process skew for systems of different sizes

We also evaluated the effect of process skew on the average host CPU time for different size systems. Figure 7 shows the factors of improvement on the host CPU time for broadcasting 4 byte and 4KB messages using the NIC-based multicast compared to the host-based multicast, over systems of different sizes. For both sizes of messages, the improvement factor becomes greater as the system size increases for a fixed amount of process skew of 400μs. This suggests that a larger size system can benefit more from the NIC-based multicast for the reduced effects of process skew.

7. Conclusions and Future Work

We have characterized features of multicast schemes that uses Myrinet programmable NICs, and proposed a NIC-based multicast scheme with a complete set of features. We have implemented this scheme, which mainly consists of a NIC-based multisend mechanism, using the NICs to send multiple replicas of a message to different destinations, and a NIC-based forwarding mechanism, using intermediate NICs to forward the received packets without intermediate host involvement. This results in a high performance and reliable NIC-based multicast. We have also modified MPICH-GM to take advantage of this NIC-based multicast. The performance benefits of the NIC-based multicast has been evaluated at both the GM-level and the MPI-level.

At the GM-level, the NIC-based multicast scheme provides an improvement factor up to 1.86 for 16KB messages and an improvement factor up to 1.48 for $\leq$ 512 byte messages over 16 nodes compared to the traditional host-based multicast. At the MPI-level, the NIC-based multicast achieves an improvement factor up to 2.02 for 8KB messages, and an improvement factor up to 1.78 for small messages $\leq$ 512 bytes over 16 nodes. In addition, at the MPI-level, the NIC-based multicast was shown to have better tolerance to process skew. In the presence of an average skew of 400μs on a 16 node system, using the NIC-based approach to perform MPI_Bcast provides an improvement factor up to 5.82 for small (2 to 8 bytes) messages and an improvement factor up to 2.9 for large (2KB) messages.

The NIC-based scheme achieves its reliability and efficiency without using a centralized manager and requires minimum memory and processor resources at the NIC, which promises good scalability. In the future, we intend to study its scalability in large scale systems. MPICH-GM employs remote DMA for over 16K messages in its rendezvous protocol. So we also intend to study the NIC-based multicast using remote DMA operations and its performance impacts at different levels. Moreover, in view of the benefits of NIC-based multicast, we intend to expand the NIC-based support to other collective operations, for example, Allreduce and Alltoall broadcast.

References

[1] A. Bar-Noy and S. Kipnis. Designing broadcasting algorithms in the postal model for message-passing systems. In *SPAA*, 1992.

[2] R. Bhoedjang, T. Ruhl, and H. Bal. LFC: A Communication Substrate for Myrinet. In *Proceedings of the Fourth Annual Conference of the Advanced School for Computing and Imaging*, pages 31–37, June 1998.

[3] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1):29–36, 1995.

[4] D. Buntinas and D. K. Panda. NIC-Based Reduction in Myrinet Clusters: Is It Beneficial? In *SAN-02 Workshop (in conjunction with HPCA)*, Feb 2003.

[5] D. Buntinas, D. K. Panda, J. Duato, and P. Sadayappan. Broadcast/Multicast over Myrinet Using NIC-Assisted Multidestination Messages. In *CANPC*, 2000.

[6] D. Buntinas, D. K. Panda, and P. Sadayappan. Fast NIC-Level Barrier over Myrinet/GM. In *IPDPS*, 2001.

[7] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An Engineering Approach*. The IEEE Computer Society Press, 1997.

[8] C. Eddington. InfiniBridge: An InfiniBand Channel Adapter With Integrated Switch. *IEEE Micro*, (2):48–56, April 2002.

[9] M. Gupta. Challenges in Developing Scalable Software for BlueGene/L. In *Scaling to New Heights Workshop*, Pittsburgh, PA, May 2002.

[10] Message Passing Interface Forum, MPIF. MPI-2: Extensions to the Message-Passing Interface. Technical Report, University of Tennessee, Knoxville, 1996.

[11] Myricom. Myrinet Software and Customer Support. http://www.myri.com/scs/GM/doc/, 2003.

[12] F. Petrini, W. C. Feng, A. Hoisie, S. Coll, and E. Frachtenberg. The Quadrics Network (QsNet): High-Performance Clustering Technology. In *the Proceedings of Hot Interconnects '01*, August 2001.

[13] R. Sivaram, R. Kesavan, D. K. Panda, and C. B. Stunkel. Architectural Support for Efficient Multicasting in Irregular Networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(5):489–513, 2001.

[14] K. Verstoep, K. Langendoen, and H. E. Bal. Efficient Reliable Multicast on Myrinet. In *ICPP*, 1996.

[15] W. Yu, D. Buntinas, and D. K. Panda. High Performance and Reliable NIC-Based Multicast over Myrinet/GM-2. Technical Report OSU-CISRC-5/03-TR33, Ohio State University, Dept. of Computer Science, 2003.