To appear in the Proceedings of the 27th International Conference on Parallel Processing (ICPP '98), August 1998, Minneapolis, MN

Where to Provide Support for Efficient Multicasting in Irregular Networks: Network Interface or Switch?*

Rajeev Sivaram† Ram Kesavan† Dhabaleswar K. Panda† Craig B. Stunkel‡

†Dept. of Computer and Information Science
The Ohio State University
Columbus, OH 43210
Email: {sivaram,kesavan,panda}@cis.ohio-state.edu

‡IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598
Email: stunkel@watson.ibm.com

Abstract

Recent research has proposed methods for enhancing the performance of multicast in networks with irregular topologies. These methods fall into two broad categories: (a) network interface (NI) based schemes that make use of enhanced functionality of the software/firmware running at the NI processor, and (b) switch-based methods that use enhancements to the switch architecture to support hardware multicast. However, it is not clear how these methods compare to each other and when it makes sense to use one over the other. In order to answer such questions, we perform a number of simulation experiments to compare the performance of three efficient multicasting schemes: an NI-based multicasting scheme that uses a k-binomial tree [5], a switch-based multicasting scheme that uses path-based multidestination worms [4], and a switch-based multicasting scheme that uses a single tree-based multidestination worm [14]. We first study the performance of the three schemes for single multicast traffic while changing a number of system parameters one at a time to isolate their impact. We then study the performance of these schemes under increasing multicast load. Our results show that the switch-based multicasting scheme using a single tree-based multidestination worm performs the best among the three schemes. However, the NI-based multicasting scheme is capable of delivering high performance compared to the switch-based multicast using path-based worms especially when the software overhead at the network interface is less than half of the overhead at the host. We therefore conclude that support for multicast at the NI is an important first step to improving multicast performance. However, there is still considerable gain that can be achieved by supporting hardware multicast in switches. Finally, while supporting such hardware multicast, it is better to support schemes that can achieve multicast in one phase.

1 Introduction

In a world where computing needs are increasing by the day, there is a constant search for cost-effective high-performance computing solutions. Using a network of commodity workstations as a parallel computing environment has the potential to provide such a cost-effective high-performance computing environment. Much recent research has therefore focussed on extending parallel processing solutions to such networks of commodity workstations (NOWs).

However, networks of workstations environments have typically evolved from computing environments where communication requirements are less stringent (both in terms of latency and bandwidth). In such an environment, the workstations that form the processing nodes are typically interconnected in arbitrary, irregular topologies. Using such topologies allows easy addition and deletion of nodes to the computing environment making the overall environment more amenable to network reconfigurations and resistant to faults. However, these topologies do not possess many of the attractive mathematical properties of the regular topologies. A lot of the problems relating to inter-processor/inter-node communication that have been solved for regular topologies are therefore being revisited for such irregular topologies. One such problem is that of collective communication.

Collective communication plays an important role in parallel systems, and involves communication among groups of (2 or more) processes [9]. Examples of collective operations include multicast [8], barrier synchronization, reduction, etc. The importance of such operations is underlined by the inclusion of several primitives for collective communication in the Message Passing Interface (MPI) standard [7]. Such collective operations are also used for system level operations in distributed shared memory systems, such as for cache invalidations, acknowledgment collection, and synchronization [2].

Of these collective operations, multicast is most fundamental and important and is used for implementing several of the other collective operations. Traditionally, multicast has been implemented using the underlying support for point-to-point (unicast) communication. The best of these schemes reduces the impact of the high overhead associated with sending/receiving point-to-point messages by allowing multiple nodes to simultaneously transmit/receive messages in a given phase. Such multicasting schemes require at least $log_2(n+1)$ communication steps, and are the best achievable using unicast communication primitives.

A number of schemes have been recently proposed to further improve multicast performance. These schemes can be classified into two main categories. The first category of schemes is referred to as *switch-based multicasting schemes*. These schemes use enhancements at the routers/switches of the network to allow an incoming packet to be forwarded to multiple output ports of the router/switch [15]. Such a method allows messages (christened *multidestination messages*) to be communicated to multiple destinations incurring a single software overhead for sending the message. There are two subclasses of switch-based multicasting schemes: tree-based [14] and path-based [4]. In the best tree-based multicasting scheme, multicast can be performed in a single phase using one multidestination worm from the source node to all the destinations [14, 15]. The path-based multicasting schemes typically require multiple multidestination worms to perform multicast

*This research is supported in part by NSF Career Award MIP-9502294, NSF Grant CCR-9704512, an IBM Cooperative Fellowship, and an Ohio State University Presidential Fellowship.

to arbitrary destination sets. These worms are transmitted in multiple phases with the destinations in a phase acting as secondary sources in succeeding phases of the multicast [4].

The second category of multicasting schemes is referred to as *network interface-based* or *NI-based multicasting schemes*. Typically, communication between two nodes incurs software overhead at both the host and at the network interface for both sending and receiving a message. The NI-based schemes use enhancements at the network interface so that multicast messages are forwarded to the next destination as soon as they are received at the network interface of an intermediate destination (and the message has begun to be transferred to the memory of the local host) [16]. This hides the significant software overhead for receiving a message at an intermediate destination host, and eliminates the overhead at the host for sending the message to other destinations in the succeeding phases of the multicast.

It is obvious that a multicasting scheme with enhanced support at the network interface *and* the switches will perform better than a scheme that makes use of support at *either* the network interface or at the switches. However, it is necessary to first compare the effect of improved support for multicast at the switches to support for multicast at the network interface to decide which of these schemes results in better performance and when. While making such a comparison, we must therefore assume no network interface support for the switch-based multicasting schemes, i.e., every communication phase under the tree-based and path-based schemes should incur software overhead at the host and network interface of the source and (intermediate) destinations. Such a comparison is needed for an architect to evaluate the relative merits of these two categories of multicasting schemes and to decide which scheme to support and when. In order to make such a decision, a number of questions need to be answered: how quickly can multicast be performed with support at the network interfaces and low overhead messaging layer support? How does multicast with switch/router support (alone) compare with multicast implemented with network interface support? For what range of system parameters does it make sense to use one over the other?

The goal of this paper is to provide answers to these questions by comparing the performance of the single phase tree-based multicasting scheme to the best path-based multicasting scheme and to an optimal NI-based multicasting scheme. We perform extensive simulations to evaluate the impact of various system parameters on the performance of the three schemes by considering the performance of single multicasts and by varying each of the parameters one at a time. These parameters include message length, number of switches, and the ratio of the overhead at the host to the overhead at the network interface. Finally, we study the latency of these schemes under increasing multicast load with a variation of a few selected parameters.

Our analysis and simulations show that the single phase tree-based multicasting scheme studied in this paper is the most powerful multicasting scheme. However, the NI-based scheme can deliver extremely high performance, especially when the software overhead for absorbing and retransmitting messages at the interface is considerably lower than the corresponding overheads at the host. The path-based multicasting scheme's performance varies with variation in a number of network parameters and it can perform worse than the NI-based scheme in a number of cases. We therefore conclude that support for multicast at the NI is an important first step to improving multicast performance. However, there is still considerable gain that can be achieved by supporting hardware multicast in switches. Finally, while supporting such hardware multicast, it is better to support schemes that can achieve multicast in one phase.

The remaining part of the paper is organized as follows. In Sec. 2 we present our network model and the underlying routing algorithm assumed in this paper. We then present in Sec. 3 an overview of the multicasting schemes used for our comparison in this paper. An exhaustive simulation based comparison of the three schemes is then presented in Sec. 4. Finally, we present our conclusions in Sec. 5.

2 System Model

In this section, we present the network model assumed in this paper. The related deadlock-free routing issues for such a network are also discussed.

2.1 Network Model

Figure 1(a) shows a typical parallel system using a switch-based interconnect with irregular topology. Such a network consists of a set of switches where each switch has a set of ports. The system in the figure consists of eight switches with eight ports per switch. Some of the ports in each switch are connected to processors, some ports are connected to ports of other switches to provide connectivity between the processors, and some ports are left open for further connections. Such connectivity is typically irregular and the only thing that is guaranteed is that the network is connected. Thus, the interconnection topology of the network can be denoted by a graph G = (V,E) where V is the set of switches, and E is the set of bidirectional links between the switches [1, 12]. Figure 1(b) shows the interconnection graph for the irregular network in Fig. 1(a). It is to be noted that all links are bidirectional and multiple links between two switches are possible.

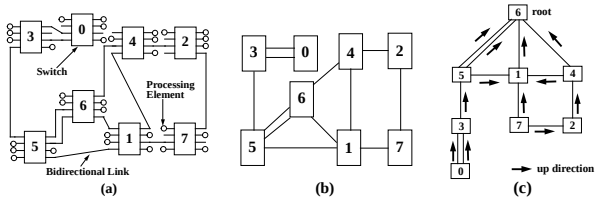

Figure 1. (a) An example system with switch-based interconnect and irregular topology; (b) corresponding interconnection graph G; (c) corresponding BFS spanning tree rooted at node 6.

2.2 Routing Issues

Several deadlock-free routing schemes have been proposed in the literature for irregular networks [1, 12]. Without loss of generality, in this paper we assume the routing scheme for our irregular network to be similar to that used in Autonet [12] due to its simplicity and its commercial implementation. Such routing allows adaptivity, and is deadlock-free.

In this routing scheme, a breadth-first spanning tree (BFS) is first computed on the graph G using a distributed algorithm. The algorithm has the property that all nodes will eventually agree on a unique spanning tree. Deadlock-free routing is based on a loop-free assignment of direction to the operational links. In particular, the "up" end of each link is defined as: 1) the end whose switch is closer to the root in the spanning tree; or 2) the end whose switch has the lower ID, if both ends are at

switches at the same tree level. The result of this assignment is that the directed links do not form loops. Figure 1(c) shows the BFS spanning tree corresponding to the interconnection graph shown in Fig. 1(b), and the assignment of the "up" direction to the links. To eliminate deadlocks while still allowing all links to be used, this routing uses the following up/down rule: a legal route must traverse zero or more links in the "up" direction followed by zero or more links in the "down" direction. Putting it in the negative, a packet may never traverse a link along the "up" direction after having traversed one in the "down" direction. Details of this routing scheme can be found in [12].

3 Multicasting Approaches

We now present an overview of the three multicasting schemes that we compare in this paper. We first describe the traditional approach to multicasting using multiple phases of unicast messages. We then present the two categories of techniques for providing enhanced support for efficient multicast and present the three schemes studied in this paper.

3.1 Multi-phase Software Approaches to Multicast

Efficient multicast algorithms are typically hierarchical in nature. This means that some destinations serve as intermediate sources, i.e., when they receive a message, they forward copies of it to other destinations. Many such hierarchical algorithms have been proposed in the literature [3] to implement multicast. Figure 2 shows an example of a multicast from a source node to seven other destinations. In the figure, the numbers in brackets indicate the step numbers.

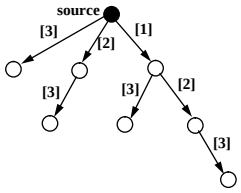

Figure 2. Example of a hierarchical multicast algorithm on a destination set size of 7.

It can be easily observed that $\lceil log_2(n+1) \rceil$ communication steps are required for such a binomial tree based hierarchical multicast to be completed [3]. A communication step is the time required for a message to be sent from one host node to another. Even with lightweight messaging layers, the latency of such a multicast operation is still dominated by the communication software overhead. Recent research to reduce the effect of this overhead on multicast performance has been in two main directions.

3.2 Enhanced Multicasting Schemes

The two major directions in which enhancements have been proposed for reducing the effect of the communication overhead are: (a) increasing the functionality of the software/firmware running at the NI processor, and (b) supporting hardware multicast at the switch. We describe these concepts in greater detail in the following subsections.

3.2.1 Multicasting using Network Interface Support

Let us consider the multicast of a message that spans multiple packets. Figure 3(a) shows the forwarding of a 2-packet multicast message at an intermediate node of a multicast tree. Each of the packets of the message is received at the network interface and copied to host memory using DMA. The host processor at the intermediate node receives the complete message and then initiates send operations to each of its children in the multicast tree. For each of the send operations, a copy of the message is sent to the network interface, from where it is sent into the network. Therefore, an intermediate node incurs the message send overhead for every copy of the message that it forwards to other destinations. This overhead includes the software start-up overhead and the overhead at the NI for each packet transmission.

Smart NI support can reduce this overhead for multicast, especially for multi-packet messages. Current lean messaging layers allow modification of part of the software running on the NI processor. This part of the software can be modified to allow it to identify a multicast packet [5, 16]. If the next outgoing packet in the send queue of the source node is a multicast packet, the NI processor forwards replicas of the packet to the nodes adjacent to the root of the multicast tree. When a multicast packet is received at the NI of an intermediate node, the NI processor starts copying (using DMA) the packet to host memory. Simultaneously, it forwards replicas of the packet to its children in the multicast tree. Thus, the overhead at the intermediate node's host to receive a packet is hidden, and the overhead at the host to send the packet to its children in the multicast tree is eliminated. Figure 3(b) shows such forwarding with smart network interface support. An implementation of such a smart network interface has been described in [16].

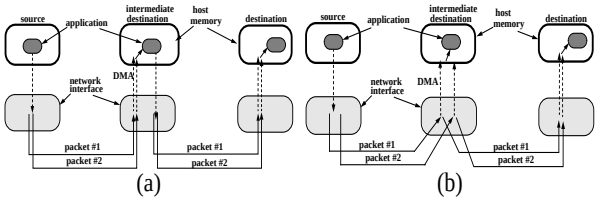

Figure 3. Forwarding of a 2-packet long multicast message by intermediate node using (a) conventional NI support, and (b) smart NI support.

In this paper, we use the FPFS (First-Packet-First-Served) implementation of the smart network interface support [5]. In this implementation, the NI forwards the message on a per-packet basis. The NI at the source node sends the first packet to all the children of the source, then sends the second packet to all the children of the source, and so on. When the first packet of the multicast message arrives at the NI of an intermediate node, it forwards the packet to each of the children of the intermediate node. Subsequently, when the second packet of the multicast message arrives at the NI, it forwards the packet to each of the children, and so on till the last packet is forwarded.

A k-binomial tree has been shown to be optimal for multi-packet multicast on systems with such smart NI support [5]. A k-binomial tree is defined as a recursively doubling tree where each vertex has at most k children. The value of k is a function of the size of the multicast set and the number of packets in the multicast message. A method for constructing k-binomial trees with minimized contention on irregular switch-based networks has been proposed in the literature [5]. In this paper we therefore use the k-binomial tree for our multicasting scheme using network interface support.

3.2.2 Multicasting using Switch Support

Another method for improving multicast performance is to provide switches with support for replicating incoming messages to multiple output ports. The basic idea behind such a

scheme is to communicate a single message (called a *multi-destination message*) from a source to multiple destinations in almost the same time it takes to send a unicast message to one node [6]. The number of destinations covered depends on the type of encoding/decoding used for the message header.

Given support for replication at the switches, there are two schemes that have been proposed for carrying out multicast in switch based irregular networks: tree-based [14] and path-based [4]. The two schemes differ in the restrictions placed on worm replication, the number of worms required to perform multicast to an arbitrary destination set, the complexity of multicast header formation, and the complexity of the header decoding logic required at the switches. However, we assume the multidestination worms under either scheme conform to the base up*/down* routing algorithm, i.e., the path followed by a multicast packet does not violate any of the rules for routing unicast packets in the system [11]. These schemes are discussed in greater detail in the following two subsections.

3.2.3 Tree-based Multicasting using Multidestination Worms

Tree-based multicasting places no restriction on the replication of a worm at a given switch. The basic idea of tree based replication is to have multiple copies of a packet propagate down a tree which has switches as internal vertices and destinations nodes as leaf vertices.

Two methods have been proposed for tree-based multicast in irregular networks [14]. In this paper, we use the method that uses a single multidestination worm with a bit-string encoded header to perform multicast [14, 15] as a representative of the tree-based multicasting method. The basic idea is to encode the multidestination worm header using an N-bit string where N is the number of nodes in the system. The destinations of the multicast are identified by setting the ith bit of the the bit string to '1' if node i is a destination of the multicast (all other bits are set to '0'). Every switch has a similar N-bit string (called the *reachability string*) associated with every one of its output ports that lead to links in the "down" direction. The reachability string specifies the set of destination nodes that are reachable through the given ("down") output port (subject to the restrictions of the base routing algorithm). After traveling adaptively to a *least common ancestor* switch using links in the "up" direction, a multicast packet is forwarded to those "down"-ward output ports whose reachability string has one or more '1' bits in the same position(s) as in the packet header [14].

Figure 4 illustrates the tree-based multidestination scheme using bit-string encoding for a sample multicast. Figure 4(a) shows the BFS graph of a sample irregular network. Let us assume that there is at least one destination node connected to each of the switches that are highlighted, and that the source node is connected to switch 10. Figure 4(b) shows the tree-based multidestination worm that covers all the destinations of the multicast. Finally, Fig. 4(c) shows an example of a bit-string encoded header and the reachability strings in a 4-port switch of an 8-node system. The worm with the header shown in Fig. 4(c) must be replicated to all the output ports of the switch.

3.2.4 Path-based Multicasting using Multidestination Worms

Under the path-based multicasting scheme [4], multidestination worms use almost exactly the same path followed by a unicast worm from a source to one of its destinations (as allowed under the base routing algorithm). However, at every

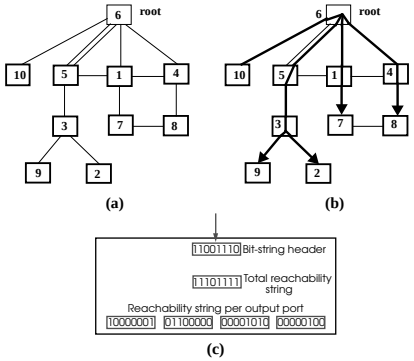

Figure 4. Illustrating the single-phase tree-based multicasting scheme: (a) sample BFS graph of an irregular network with participating switches in the multicast highlighted, (b) the tree-based multidestination worm, and (c) an example of a bit-string encoded header and the various reachability strings in a 4-port switch.

switch along its path, the multidestination worm is allowed to replicate and cover all the destinations of the multicast that are directly connected to that switch. At every switch, the worm is allowed to replicate to multiple output ports that are connected to destination nodes and (if needed) to exactly one other output port that is connected to a switch. Such multidestination worms are called *multi-drop path-based* worms. Figure 5(a) shows an example of a multi-drop path-based multidestination worm. Since there may be no single path from the source which covers all of the destinations, path-based multicasting requires multiple multidestination worms to cover an arbitrary multicast destination set. Furthermore, to arrive at a relatively small number of multidestination worms that cover a given destination set, an algorithm is needed that chooses appropriate paths so that as many destinations as possible are covered by each of the worms. Multiple algorithms have been proposed for this [4]. In addition, a multi-phase algorithm is required to cover the destination set using the multidestination worms so that contention is reduced and multicast can be performed efficiently. In this paper, we use the Multi-Drop Path-based Less Greedy (MDP-LG) algorithm that finds a small number of multidestination worms to cover the destination set and decides how to send these worms in multiple-phases so as to reduce contention and enhance performance. Details of this algorithm can be found in [4], where this algorithm was shown to perform best among the proposed algorithms.

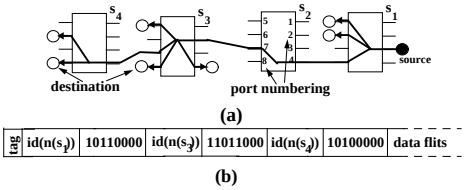

Figure 5. The Multi-drop mechanism: (a) example of a multidestination worm and (b) the header encoding.

The encoding used for the header of each of the multidestination worms under the path-based scheme is as follows. The header of a path-based multidestination worm consists of $2r$ fields, where r is the number of switches on the worm's path

in which replication occurs. The header consists of alternating fields consisting of a node ID field followed by a p-bit string field (where p is the number of input ports/output ports in each switch). Figure 5(b) shows the header encoding of the worm shown in Fig. 5(a). The expression $id(n(s_i))$ is the ID of any arbitrary destination node which is connected to the switch s_i. The node ID field is used to index into the tables for unicast message routing and to route the message towards the switch that connects to the specified node. When that switch is reached, the node ID field is stripped and the p-bit string field is examined. The p-bit string has '1' bits corresponding to the output ports to which the worm must be forwarded. As mentioned above, at most one of these ports leads to another switch. The other ports correspond to destination nodes where the message is absorbed. The p-bit string field is stripped before the worm is forwarded to the next switch (if any) where the procedure is repeated. When the message reaches the last switch on its path, the node ID field is stripped away and the message is forwarded to the output ports that correspond to the '1' bits in the last (p-bit string) field of the header.

3.3 Comparison of Architectural Requirements

We now qualitatively compare the three enhanced multicasting schemes described above. The NI-based multicasting scheme requires the computation of the entire k-binomial multicast tree. It may also require additional memory at the network interfaces to buffer multicast packets. This is because each packet is forwarded to multiple destinations, and has to be buffered till the NI-processor has injected all required copies into the network. However, the NI-based multicasting scheme requires no additional support at the switches than what is required for traditional point-to-point message communication.

The tree-based multicasting scheme only requires a single multidestination worm to perform multicast. Furthermore, encoding the multidestination worm header is fairly simple: start with an N bit string and set the bits in the positions corresponding to the destinations to '1'. However, decoding the multidestination header is more complex. First, the switches need to be equipped with reachability strings, which means space is required at the switches for this. Second, there is a one time cost of setting up the values in the strings (which can be done at the time of network startup or reconfiguration). Third, the N bit string has to be compared with the reachability strings of each of the N output ports and the copy of the worm forwarded through a given output port should carry a modified header. Depending on the size of the bit string (which in turn depends on the system size), the cost of such logic may be significant. Finally, support for deadlock-free replication is required at the switches.

The path-based multicasting scheme also requires support for replication at the switches. Furthermore, encoding the worm header is relatively harder. The source needs to know the network topology and needs to run an algorithm to decide on paths that can cover the destination set using very few multidestination worms. Once the paths have been formed, the header must be formed in the format described above. However, decoding the multidestination header may be relatively easy. First, there is no necessity for maintaining reachability strings at each of the switches. Second, the decoding operation at the switches is relatively simple: it is the same table lookup operation required for unicast messages or it is the simple processing of a k-bit string. Finally, the cost of decoding logic does not increase with increase in system size.

We have provided the various trade-offs relating to the cost and complexity of each of three multicasting schemes above. In the next section we compare the performance of the three schemes using simulation experiments.

4 Simulation-based Evaluation

The relative performance of the three multicasting schemes described in the previous sections may depend on a number of parameters. To study the factors that may affect multicast performance in the presence of network contention, we performed simulation experiments varying a number of system parameters one at a time. In the following subsection we present the experiments we performed and the parameters that we varied. The results of our experiments with single multicast are presented next, followed by the results for our experiments to measure the impact of increasing applied load on multicast latency.

4.1 Experiments and Performance Measures

We used a C++/CSIM based simulation testbed [10] for our experiments. The simulation testbed models a large number of topologies, and uses cut-through routing as the flow control technique with an input buffer size of 640 flits.

For each of our experiments, we assume the following default parameters. We assume that the I/O Bus at every host has a bandwidth of 266 MB/s. The current PCI Bus standard calls for a bandwidth of 133 MB/s: our assumptions reflect the belief that I/O bus bandwidths will increase in the future. Let the communication software overhead per message at the sending and receiving host processors be t_{hs} and t_{hr}, respectively, and let the corresponding overhead at the sending and receiving NI processors be t_{ns} and t_{nr}, respectively. For all our experiments, we assume $t_{hr} = t_{hs}$ and $t_{nr} = t_{ns}$. We use the term R to denote the ratio of t_{hs} to t_{ns}, and we assume a default value of $R = 1$. This reflects our belief that messaging layers will become thin and efficient, and that while most work relating to initiating a message may still be done on the host processor, the relatively low speed of the NI processor makes the value of t_{ns} comparable to t_{hs}. We assume a default cycle time of 5 ns, and a default value of 1000 cycles for t_{hs}. This value corresponds to the software overhead incurred at the host using many of the current-day lightweight messaging layers.

In the network, we assume that links are 1 byte wide and that this is equal to the flit size. The link propagation time for a flit across a physical link is assumed to be 1 cycle, as is the time to propagate through a switch crossbar from the input to the output buffer of the switch. We assume a uniform routing overhead of 1 cycle for all three schemes. This reflects our assumption that while the *cost* of the logic involved for decoding and routing a header under the different schemes may vary, the approximate *latency* for doing these operations is likely to be kept within a cycle in most switch implementations. We assume a default packet size of 128 flits, and a default message size of 1 packet. Finally, we assume a default system of 32 nodes that are interconnected by eight 8-port switches in an irregular topology. Our method for generating different irregular topologies is described in [13]. Using this method we generated 10 different topologies, and our results are averaged over all these topologies.

We use two types of experiments to measure the performance of the three multicasting schemes. In the first type of experiments, we measure the latency of single multicasts for each of the three schemes and study the effect of different parameters on the relative latencies of the three schemes. We assume that exactly one multicast occurs in the system at any given time and that there is no other network traffic. This

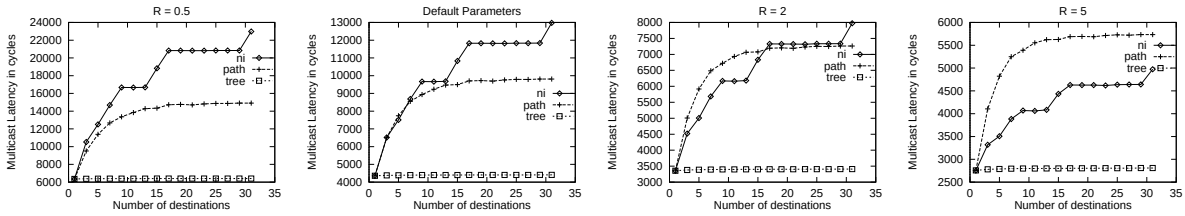

Figure 6. Effect of R on single multicast performance.

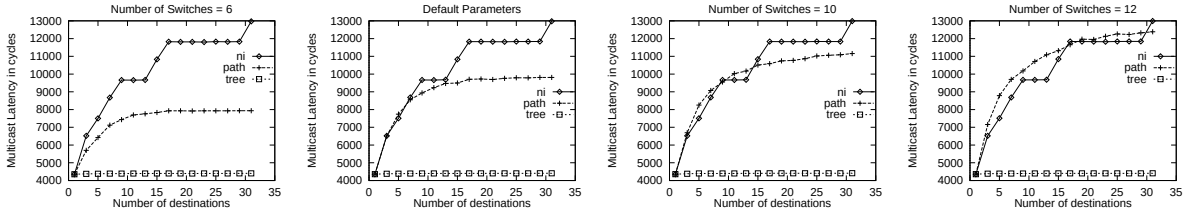

Figure 7. Effect of Number of Switches

gives us an estimate of the best possible performance of each of the three schemes in isolation. We use traffic consisting of multiple concurrent multicasts for our second type of experiments. We apply an increasing load consisting of multicast traffic alone and examine the load at which the network saturates with each of the three multicasting schemes under the influence of the various parameters. As in [15], we use effective applied load as a measure of our stimulus. For a multicast of degree m and a load of B, the effective applied load is mB.

4.2 Single Multicast Performance

We now present our results for the effect of single multicasts on the three different multicasting schemes. One by one, we examine the effect of each parameter on the performance of the schemes. As mentioned in Sec. 1, it must be kept in mind that the multi-phase path-based multicasting scheme can also make use of support at the NI to further enhance multicast performance. However, since we are evaluating the effect of support at the NI versus support at the switches, we assume that under the path-based scheme, every intermediate destination receives the incoming message completely at the host and then retransmits the message to the NI and then onto the network.

4.2.1 Effect of R

We first examine the effect of variation in the ratio R (t_{hs}/t_{ns}) on the performance of the three multicasting schemes. Given our default value of $t_{hs} = 1000$ cycles we vary the value of t_{ns} to take on the values 2000, 1000 (default), 500, and 200 cycles (to generate the following values of R: 0.5, 1, 2, and 5, respectively). Figure 6 presents the results of our experiments. We find here (as well as in the other single multicast experiments to follow) that the tree-based multicasting scheme performs extremely well, since it requires only one message and therefore only one communication phase. The interplay between the NI-based and path-based schemes is more interesting. As the ratio increases (i.e. t_{ns} shrinks relative to t_{hs}), the NI-based multicasting scheme begins to outperform the path-based scheme. This is because although the NI-based scheme involves more communication phases, every communication phase incurs a receive overhead of t_{nr} and a send overhead of t_{ns}. On the other hand, the fewer phases of the path-based scheme each incur a receive overhead of $t_{nr} + t_{hr}$ and a sending overhead of $t_{hs} + t_{ns}$.

4.2.2 Effect of Number of Switches

To see the effect of increase in number of switches on multicast performance we increased the number of switches used while keeping the system size (i.e., the number of nodes) constant. However, all switches had 8 ports.

As the number of switches for a given system size increase, the average number of nodes per switch decreases as does the average number of multicast destinations per switch. The results of Fig. 7 show that the number of multidestination worms, the number of phases for multicast, and the multicast latency for the path-based scheme increase with decrease in the average number of destinations per switch. The other two multicasting schemes remain largely unaffected by this increase in the number of switches.

It is to be noted that for the NI-based scheme, the average path length increases with larger number of switches. However, this affects only the propagation time of the worms. Furthermore, since we are using cut-through routing (which is almost "distance independent") this increase in propagation time is negligible.

4.2.3 Effect of Message Length

Figure 8 shows the effect of increasing message length on multicast performance. Here too the path-based scheme begins to perform worse than the NI-based multicast beyond a message length of 512 flits. However, the reason for this decrease in the relative performance of the path-based scheme can be attributed to the increase in the latency of each of its phases: the number of phases remains unchanged. We assume a packet size of 128 flits. Messages larger than this size are split into multiple packets. Under the path-based scheme, a phase finishes only when all the packets of the message arrive at an intermediate destination: only then can the node initiate the path-based multidestination worm of the next phase. On the other hand, under the NI-based scheme (and following the FPFS discipline outlined earlier), a packet can be forwarded to the each of the recipients of the next phase as soon as it arrives at the network interface of an intermediate destination node. The NI-based scheme therefore begins to gain in performance as the number of packets in a message increases.

We also performed a number of experiments to study the effect of startup overhead at the host, system size, and packet length. However, due to lack of space, these results are not presented. Interested readers may refer to [13] for details.

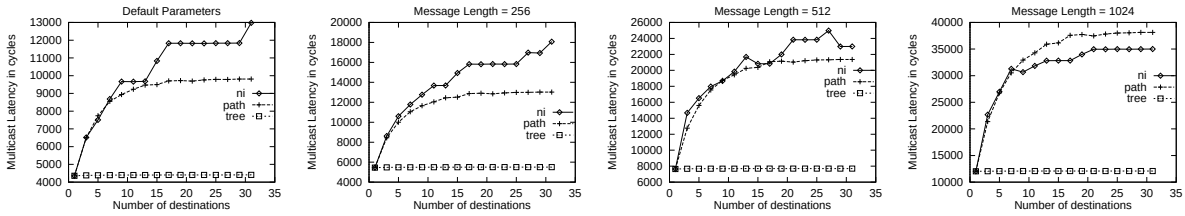

Figure 8. Effect of Message Length

4.3 Latency versus Applied Load for Multicast

We now present our results for multicast latency under an increasing multicast load for each of the three schemes. We used two different multicast degrees in our experiments: 3-way and 15-way multicasts (i.e., multicasts with 3 and 15 destinations, respectively). For each of our experiments, our simulations were run for at least one million cycles, with measurements beginning after a cold-start time of 500,000 cycles. It is worth keeping in mind that for each of the networks, the maximum unicast throughput (assuming no software overheads and no contention for the I/O bus) was observed to be less than 0.18 using up*/down* routing. Also, each of the plots in this section show multicast latency against *effective applied load* as discussed in Sec. 4.1.

4.3.1 Effect of R

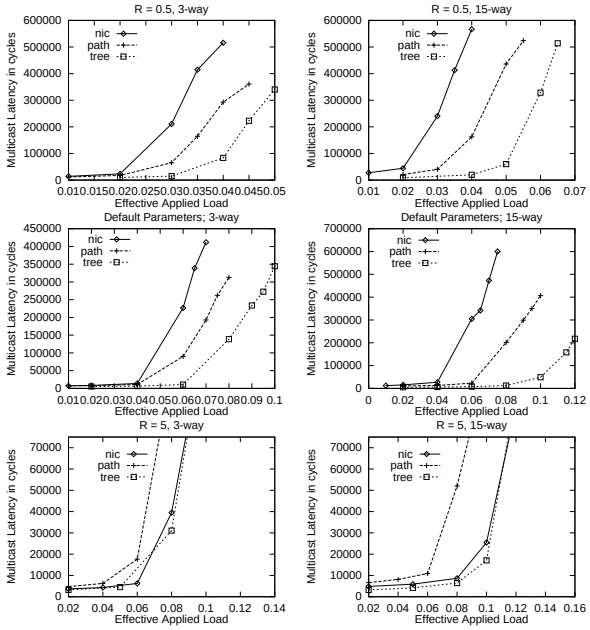

Figure 9. The effect of parameter R on the latency of multicasts under increasing multicast load for 3-way and 15-way multicasts.

Figure 9 shows the results of our experiments under variation of R. In general, for a value of R less than or equal to 1.0, the NI-based scheme performs worst followed by the path-based scheme. The tree-based scheme performs best for such values of R. However, when R becomes greater than 1.0, we note an interesting trend. Now the NI-based scheme performs comparably to the tree-based scheme and much better than the path-based scheme. A possible reason for this is that the tree-based scheme causes an almost simultaneous reception in all its recipients leading to an increase in the contention for resources at the recipient nodes. On the other hand, the NI-based scheme "spreads" the receive times among the recipients of the multicasts, causing the performance improvement.

4.3.2 Effect of Number of Switches

Our experiments with single multicasts in systems with increasing number of switches has shown that the path-based scheme begins to perform worse as the number of switches in the system increases. We observe a similar trend for our results with multiple multicast traffic (shown in Fig. 10). As the number of switches increases, the saturation load for the path-based scheme approaches that of the NI-based scheme. However, the NI-based scheme results in a greater amount of traffic and higher contention in the network. The tree-based multicast performs almost uniformly well with increase in the number of switches and saturates much later than the other two schemes.

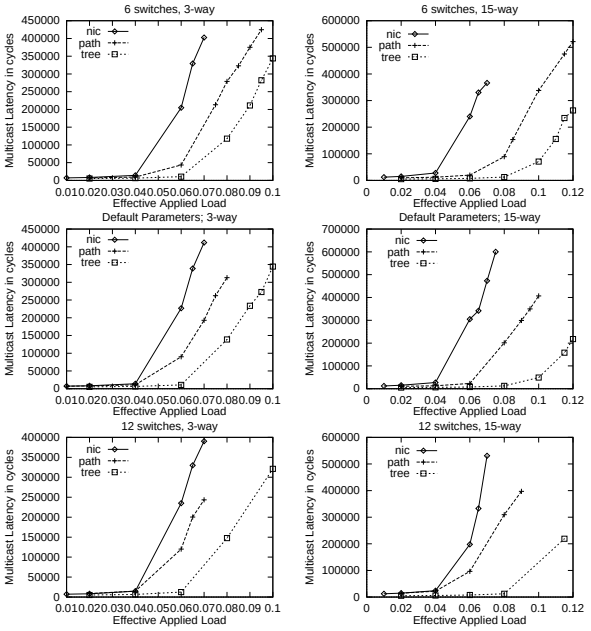

Figure 10. The effect of the number of switches on the latency of multicasts under increasing multicast load for 3-way and 15-way multicasts.

4.3.3 Effect of Message Length

Our results for multicast performance under increasing message length are shown in Fig. 11. The results show that the tree-based multicasting scheme performs best for all message lengths. Furthermore, as was noted for single multicasts, the

performance of the NI-based and path-based schemes become comparable as the message lengths increase. Recall that for a single multicast with a message length of 1024 flits (Sec. 4.2.3) we observed that the NI-based scheme performs better than the path-based scheme. Under multiple multicast traffic, the NI-based scheme performs worse (has a lower saturation point and higher latencies) than the path-based scheme for this value of message length, especially for large multicast degrees. This is because the NI-based scheme involves more communication phases and results in more traffic than the path-based scheme, thereby increasing the contention in the network.

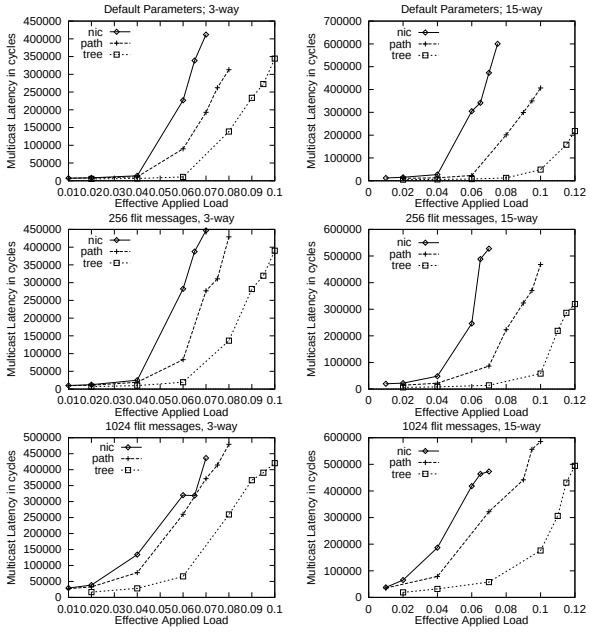

Figure 11. The effect of the message length on the latency of multicasts under increasing multicast load for 3-way and 15-way multicasts.

5 Conclusion

In this paper, we have compared three schemes for efficient multicast in switch-based irregular networks. We find that the tree-based multicasting scheme performs better than the path-based and NI-based schemes. The relative performance of the path and NI-based schemes is sensitive to a number of parameters. The most important of these parameters is the ratio R of overhead at the host to the overhead at the NI. We find that the path-based scheme performs better than the NI-based scheme for values of R less than 1, smaller system sizes, larger switch sizes, fewer switches for a given system size, and for multicasts with fewer packets. In all other cases, the NI-based scheme outperforms the path-based scheme.

Since a wealth of research has focussed on more efficient network interfaces, the value of R is likely to rise in the future. It is also important that the performance of multicast scale with increasing system size and with increase in the number of switches. We therefore conclude that support for multicast at the NI is an important first step to improving multicast performance. However, there is still considerable gain that can be achieved by supporting hardware multicast in switches. In particular, unlike with the NI-based schemes, the performance of the switch-based multicasting schemes is able to scale with the trend of increasing switch size. Finally, while supporting such hardware multicast, it is better to support schemes that can achieve multicast in one phase even at a (perhaps) additional cost.

Additional Information: A number of related papers and technical reports can be obtained from http://www.cis.ohio-state.edu/~panda/pac.html.

References

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

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

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

[4] R. Kesavan and D. K. Panda. Multicasting on Switch-based Irregular Networks using Multi-drop Path-based Multidestination Worms. In *PCRCW '97*, pages 179–192, June 1997.

[5] R. Kesavan and D. K. Panda. Optimal Multicast with Packetization and Network Interface Support. In *ICPP*, pages 370–377, Aug 1997.

[6] X. Lin and L. M. Ni. Deadlock-free Multicast Wormhole Routing in Multicomputer Networks. In *ISCA*, pages 116–124, 1991.

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

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

[9] 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.

[10] D. K. Panda, D. Basak, D. Dai, R. Kesavan, R. Sivaram, M. Banikazemi, and V. Moorthy. Simulation of Modern Parallel Systems: A CSIM-based approach. In *WSC'97*, pages 1013–1020, December 1997.

[11] D. K. Panda, S. Singal, and R. Kesavan. Multidestination Message Passing in Wormhole k-ary n-cube Networks with Base Routing Conformed Paths. *IEEE TPDS*, to appear.

[12] M. D. Schroeder et al. Autonet: A High-speed, Self-configuring Local Area Network Using Point-to-point Links. Technical Report SRC research report 59, DEC, Apr 1990.

[13] R. Sivaram, R. Kesavan, D. K. Panda, and C. B. Stunkel. Where to Provide Support for Efficient Multicasting in Irregular Networks: Network Interface or Switch? Technical Report OSU-CISRC-02/98-TR05, The Ohio State University, February 1998.

[14] R. Sivaram, D. K. Panda, and C. B. Stunkel. Multicasting in Irregular Networks with Cut-Through Switches using Tree-Based Multidestination Worms. In *PCRCW '97*, pages 35–48, June 1997.

[15] C. B. Stunkel, R. Sivaram, and D. K. Panda. Implementing Multidestination Worms in Switch-Based Parallel Systems: Architectural Alternatives and their Impact. In *ISCA-24*, pages 50–61, June 1997.

[16] K. Verstoep, K. Langendoen, and H. Bal. Efficient Reliable Multicast on Myrinet. In *ICPP*, pages III:156–165, Aug 1996.