PERFORMANCE MODELING AND ANALYSIS OF MULTICAST INFRASTRUCTURE FOR HIGH-SPEED CLUSTER AND GRID NETWORKS

By

HAKKI SARP ORAL

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2003

ACKNOWLEDGMENTS

I wish to thank the members of the High-performance Computation and Simulation Lab for their help and technical supports, especially Ian Troxel for spending countless hours with me and Dr. Alan. D. George for his patience and guidance. I also wish to thank everyone who supported me with their encouragement; without your support I would not have been able to complete this research and dissertation.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

PERFORMANCE MODELING AND ANALYSIS OF
MULTICAST INFRASTRUCTURE FOR
HIGH-SPEED CLUSTER AND GRID NETWORKS

By

Hakki Sarp Oral

December 2003

Chair: Alan D. George
Major Department: Electrical and Computer Engineering

Cluster computing provides performance similar to that of supercomputers in a cost-effective way. High-performance, specialized interconnects increase the performance of clusters. Grid computing merges clusters and other geographically distributed resources in a user-transparent way. Collective communications, such as multicasting, increase the efficiency of cluster and Grid computing. Unfortunately, commercial high-performance cluster interconnects do not inherently support multicasting. Moreover, Grid multicasting schemes only support limited high-level applications, and do not target latency-sensitive applications. This dissertation addresses these problems to achieve efficient software-based multicast schemes for clusters. These schemes are also used for defining a universal multicast infrastructure for latency-sensitive Grid applications.

The first phase of research focuses on analyzing the multicast problem on high-performance, direct, torus networks for clusters. Key contributions are experimental

evaluation, latency modeling, and analytical projections of various multicast algorithms from literature for torus networks. Results show that for systems with small messages and group sizes, simple algorithms perform best. For systems with large messages and group sizes, more complex algorithms perform better because they can efficiently partition the network into smaller segments.

High-performance clusters can also be built using indirect networks with NIC-based processors. The second phase introduces multicast performance analysis and modeling for such networks. Main contributions are experimental evaluation, latency modeling, and analytical projections for indirect networks with NIC-based processors with different levels of host-NIC work-sharing for communication events. Results show that for small message and system sizes, a fast host CPU outperforms other configurations. However, with increasing message and system sizes, the host CPU is overloaded with communication events. Under these circumstances, work offloading increases system performance. Experimental and comparative multicast performance analyses for direct torus networks and indirect networks with NIC-based processors for clusters are also introduced.

The third phase extends key results from the previous two phases to conceptualize a latency-sensitive universal multicast framework for Grid-connected clusters. The framework supports both clusters connected with high-performance specialized interconnects and ubiquitous IP-based networks. Moreover, the framework complies with the Globus infrastructure. Results show that lower WAN-backbone impact is achieved and multicast completion latencies are not affected by increased retransmission rates or the number of hops.

CHAPTER 1
INTRODUCTION

Because of advancements in VLSI technology, it has become possible to fit more transistors, gates, and circuits in the same die area, operating at much higher speeds. These enhancements have allowed the performance of microprocessors to increase steadily. Following this trend, commercial off-the-shelf (COTS) PCs or workstations built around mass-produced, inexpensive CPUs have become the basic building blocks for parallel computers instead of expensive and special-built supercomputers. Mass-produced, fast, commodity network cards and switches have allowed tightly integrated clusters of these workstations to fill the gap between desktop systems and supercomputers. Although in terms of computing power and cost, this approach is accepted as the optimal solution for small-scale organizations, it is not cost-effective for large-scale organizations. Large-scale organizations incorporate geographically distributed locations, and the cost of maintaining and operating a separate parallel-computing cluster in each location impedes the benefit of cluster-based parallel computing. Therefore, the question is still open: What is the most optimal way to orchestrate geographically distributed resources, devices, clusters, and machines in a computationally effective and cost-effective manner simultaneously? Distributed computing has emerged as a partial solution to this problem.

Distributed computing orchestrates geographically distributed resources, machines, and clusters for solving computational problems in a cost-effective way. This approach is cost-oriented and does not meet the computational needs of large-scale problems. On the

other hand, Grid computing focuses on the computational side of the quest, and is therefore problem-driven [1]. As computing technology emerges, researchers work on bigger real-life scientific problems and to solve these very large-scale problems more complex experiments and simulations than ever before are designed. Data sets obtained from these simulations and experiments also steadily grow larger in scale and size. For example, Higher Energy and Nuclear Physics (HENP) tackles the problem of analyzing collisions of high-energy particles which provides invaluable insights into these fundamental particles and their interactions. Thus, HENP is expected to provide better insight into the understanding of the unification of forces, the origin and stability of matter, and structures and symmetries that govern the nature of matter and space-time in our universe. The HENP experiments currently produce data sets in the range of PetaBytes (10^{15}), and it is estimated that by the end of this decade the data sets will reach ExaBytes (10^{18}) in size [2].

Another example is Very-Long-Baseline Interferometry (VLBI) [3]. The VLBI is a technique used by astronomers for over three decade for studying objects in the universe at ultra-high resolutions and measuring earth motions with high precision. The VLBI provides much better resolutions than the optical telescopes. Currently VLBI readings produce gigantic data sets on the order of GigaBytes (10^{9}) per second continuously.

To solve these and similar scientific problems, an unprecedented degree of scientific collaboration on a global scale is needed. To cope with these enormous problem and data-set complexities and to provide a global-level of scientific collaboration, Grid computing has been proposed. Grid computing brings together parallel and supercomputers, databases, scientific instruments, and display devices

located at geographically distributed sites in a user-transparent way. According to Foster *et al.* [1; pp. 1-2]

> Grid computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation. … The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. The sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what we call a virtual organization (VO).

The two key concepts in bringing together these resources and VOs are high-speed networks, and light-weight high-performance middleware and communication services. The need for high-performance collaboration and computing demands high-performance connectivity on the global scale between these resources and VOs. Projects like iVDgL [4], GriPhyN [5], DataTAG [6], and StarLight [7] are among numerous initiatives that involve research to build or exploit global-scale high-performance interconnects and middleware and communication services for Grids. These high-performance interconnects are designed and implemented in a hierarchical fashion, from high-speed global backbones that connect organizations to small proximity networks that connect the resources and clustered computational nodes. Conventional and relatively cheap interconnects such as Fast Ethernet or Gigabit Ethernet, or propriety and relatively expensive high-performance System Area Networks (SANs) are used to connect these clustered nodes. A SAN is a low-latency, high-throughput interconnection network that uses reliable links to connect clustered computing nodes over short physical distances for high-performance connectivity.

The success of Grids depends on the performance of the physical interconnect, and also on the efficiency and performance of middleware and communication services. Interprocess communication is the basis of most, if not all, communication services. Interprocess communication primitives can be classified as point-to-point (unicast), involving a single source and destination node, or collective, involving more than two processes. Interprocess communication is often handled by collective communication operations in parallel or distributed applications. These primitives play a major role in Grid-level applications by making them more portable among different platforms. Using collective communication simplifies and increases the functionality and efficiency of parallel tasks. As a result, efficient support of collective communication is important in the design of high-performance parallel, distributed, and Grid computing systems.

Multicast communication is an important primitive among collective communication operations. Multicast communication (the one-to-many delivery of data) is concerned with sending a single message from a source node to a set of destination nodes. Special cases of multicast include unicast, in which the source node must transmit a message to a single destination, and broadcast, in which the destination node set includes all defined network nodes. Multicast communication has been subject to extensive research in both Layer 3 (IP-based) and Layer 2 (MAC-based) levels. Multicast is widely used for simultaneous and on-demand audio and video data distribution to multiple destinations, and data and replica delivery over IP-based networks. Multicast over Active Networks is another area of research that targets increased efficiency and improved reliability [8]. There are well-defined and deployed Layer 3 multicast protocols such as MBone [9] and MBGP/PIM-SM/MSDP [10-12] for

high-performance IP-based interconnects, although most of these are not widely standardized and are still evolving. Layer 2 multicast is widely used as a basis for many collective operations, such as barrier synchronization and global reduction, and for cache invalidations in shared-memory multiprocessors [13]. Layer 2 multicast also functions as a useful tool in parallel numerical procedures such as matrix multiplication and transposition, Eigenvalue computation, and Gaussian elimination [14]. Moreover, this type of communication is used in parallel search [15] and parallel graph algorithms [16].

Grids and distributed systems often form as the integration of multiple networks with different performance characteristics and functionalities combined. These networks can be listed as Wide Area Networks (WANs) as the backbones; Metropolitan Area Networks (MANs) or Campus Area Networks (CANs) providing regional connectivity to the backbones; Local Area Networks (LANs) providing connectivity to the regional networks; and SANs connecting the computational nodes and clusters or devices to the rest of the integrated networks hierarchy. These large-scale integrated systems support and use various communication protocols (e.g., IP/SONET or IP/Ethernet for WANs, IP/Ethernet for MANs, CANs, and LANs, and proprietary protocols for SANs). Providing an efficient multicast communication service for such large-scale systems imposes multiple challenges. For example, the data distribution patterns must be optimized in order to minimize the utilization and impact on the Grid and distributed system backbones as these are the longest links that the data has to traverse. Also, an efficient multicast service must route the data over the fastest interconnect possible towards the destination to obtain low latencies, for the cases where multiple interconnects, such as LANs and SANs, both provide connectivity to the destination at

the same time. Therefore, it is beneficial for such a communication system to support multiple communication protocols. Furthermore, for unsuccessful transmissions, data must be re-transmitted from the closest upper-level parent node to the destinations, again to obtain low-latency characteristics, and to minimize the impact on the backbones.

In this dissertation, the multicast problem is evaluated for Grid-connected IP-based and SAN-based clusters. A framework for low-level multicast communication is proposed that can be used as a service for high-level Grid or distributed applications. The proposed framework targets high-performance and latency-sensitive applications. Also, the proposed framework supports multiple protocols and different levels of interconnects in a hierarchical way.

The problem is solved using a bottom-up approach. First, the Layer 2 multicast is investigated for various communication and networking scenarios using experimental and analytical modeling over various SANs. Results obtained from these Layer 2 studies are then combined with the existing Layer 3 research available in literature, to build a hierarchical multi-protocol and low-level multicast communication framework for Grids and distributed systems.

Chapter 2 analyzes the multicast communication problem for the Scalable Coherent Interface SAN [17]. Chapter 3 evaluates the multicast problem for the Myrinet SAN [18]. A comparative performance evaluation of multicast on these two SANs is presented in Chapter 4. Chapter 5 combines results obtained in Chapters 2 and 3 and then focuses on building a universal, latency-sensitive multicast communication framework for Grid-connected clusters. Conclusions are presented in Chapter 6.

CHAPTER 2
MULTICAST PERFORMANCE ANALYSIS AND MODELING FOR HIGH-SPEED UNIDIRECTIONAL TORUS NETWORKS

Direct torus networks are widely used in high-performance parallel computing systems. They are cost-effective, as the switching tasks are distributed among the hosts instead of having centralized switching elements. Torus networks also provide good scalability in terms of bandwidth. With each added host, the aggregate bandwidth of the system also increases. Moreover, torus networks allow efficient routing algorithms to be designed and implemented. The Scalable Coherent Interface (SCI) is a high-performance interconnect that supports tori topologies. The SCI is a widely used SAN because of its high unicast performance, but its multicast communication characteristics are still unclear. This chapter focuses on evaluating the performance of various unicast-based and path-based multicast protocols for high-speed torus networks. The tradeoffs in the performance of the selected algorithms are experimentally evaluated using various metrics, including multicast completion latency, tree creation latency, CPU load, link concentration, and concurrency. Analytical models of the selected algorithms for short messages are also presented. Experimental results are used to verify and calibrate the analytical models. Analytical projections of the algorithms for larger unidirectional torus networks are then produced.

Scalable Coherent Interface

The SCI initially aimed to be a very high-performance computer bus that would support a significant degree of multiprocessing. However, because of the technical

limitations of "bus-oriented" architectures, the resulting ANSI/IEEE specification [19] turned out to be a set of protocols that provide processors with a shared-memory view of buses using direct point-to-point links. Based on the IEEE SCI standard, Dolphin's SCI interconnect addresses both the high-performance computing and networking domains. Emphasizing flexibility and scalability; and multi-gigabit-per-second data transfers, SCI's main application area is as a SAN for high-performance computing clusters.

Recent SCI networks are capable of achieving low latencies (smaller than 2μs) and high throughputs (5.3Gbps peak link throughput) over point-to-point links with cut-through switching. Figure 2-1 is an architectural block diagram of Dolphin's PCI-based SCI NIC. Using the unidirectional ringlets as a basic block, it is possible to obtain a large variety of topologies, such as counter-rotating rings and unidirectional and bi-directional tori. Figure 2-2 shows a sample unidirectional SCI ringlet. Figure 2-3 shows a 2D 3-ary SCI torus.

Unlike many other competing SANs, SCI also offers support for both the shared-memory and message-passing paradigms. By exporting and importing memory chunks, SCI provides a shared-memory programming architecture. All exported memory chunks have a unique identifier, which is the collection of the exporting node's SCI node ID, and the exporting application's Chunk ID and the Module ID. Imported memory chunks are mapped into the importer application's virtual memory space. To exchange messages between the nodes, the data must be copied to this imported memory segment. The SCI NIC detects this transaction, and automatically converts the request to an SCI network transaction. The PCI-to-SCI memory address mapping is handled by the SCI protocol engine. The 32-bit PCI addresses are converted into 64-bit SCI addresses, in

which the most significant 16 bits are used to select between up to 64K distinct SCI devices.

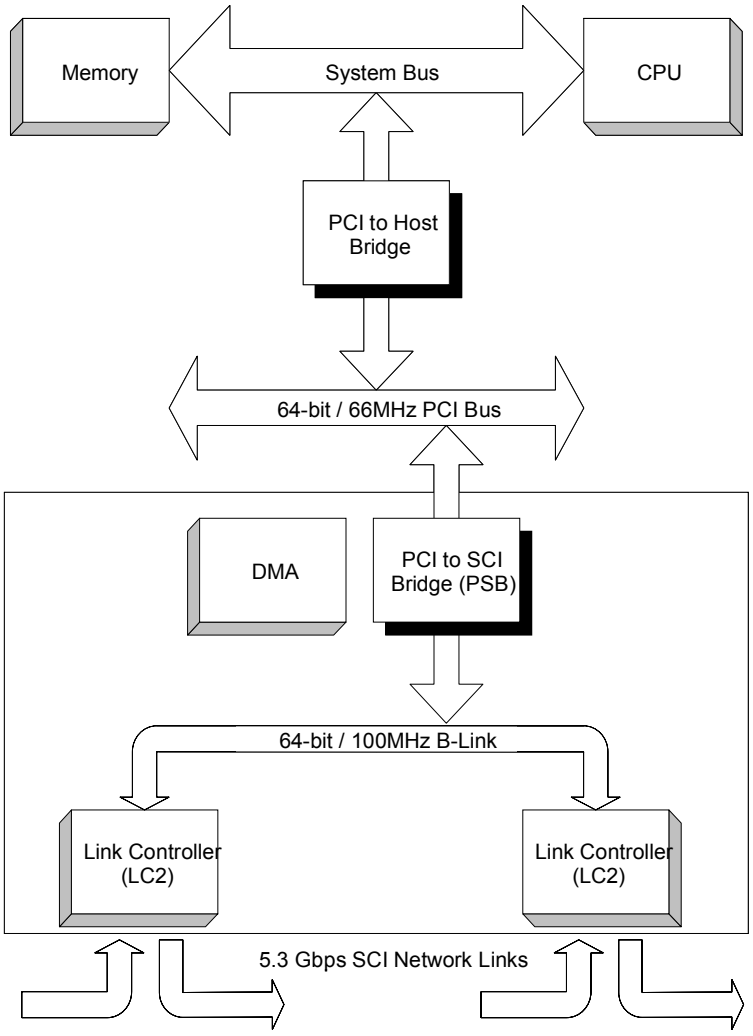


Figure 2-1. Architectural block diagram of Dolphin's PCI-based SCI NIC.

Each SCI transaction typically consists of two sub-transactions: a request and a response. For the request sub-transaction, a read or write request packet is sent by the requesting node to the destination node. The destination node sends an echo packet to the requesting node upon receiving the request packet. Concurrently, the recipient node processes the request, and sends its own response packet to the requesting node. The

requesting node will acknowledge and commit the transaction by sending an echo packet back to the recipient node upon receiving the response packet.

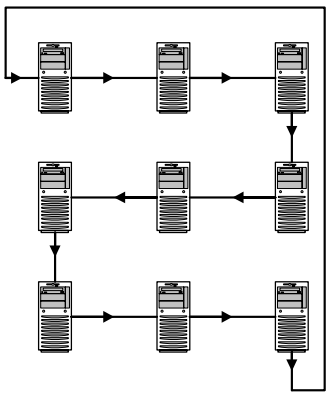


Figure 2-2. Unidirectional SCI ringlet.

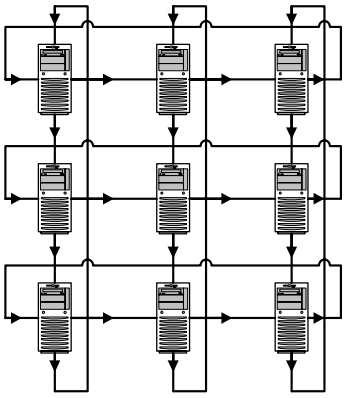


Figure 2-3. Unidirectional 2D 3-ary SCI torus.

Related Research

Research for multicast communication in the literature can be briefly categorized into two groups: unicast-based and multi-destination-based [15]. Among the unicast-based multicasting methods, separate addressing is the simplest one, in which the source node iteratively sends the message to each destination node one after another as separate unicast transmissions [20]. Another approach for unicast-based multicasting is to use a multi-phase communication configuration for delivering the message to the destination nodes. In this method, the destination nodes are organized in some sort of binomial tree, and at each communication step the number of nodes covered increases by a factor of *n*, where *n* denotes the fan-out factor of the binomial tree. The U-torus multicast algorithm proposed by Robinson *et al.* [20] is a slightly modified version of this binomial-tree approach for direct torus networks that use wormhole routing.

Lin and Ni [21] were the first to introduce and investigate the path-based multicasting approach. Subsequently, path-based multicast communication has received

attention and has been studied for direct networks [14, 20, 22]. Regarding path-based studies, this dissertation concentrates on the work of Robinson *et al.* [20, 22] in which they have defined the *U-torus*, *S-torus, M_d-torus, M_u-torus* algorithms. These algorithms were proposed as a solution to the multicast communication problem for generic, wormhole-routed, direct unidirectional and bi-directional torus networks. More details about path-based multicast algorithms for wormhole-routed networks can be found in the survey of Li and McKinley [23]. Tree-based multicasting also received attention [23, 24]; and these studies focused on solving the deadlock problem for indirect networks.

The SCI unicast performance analysis and modeling has been discussed in the literature [24-26, 28], while collective communication on SCI has received little attention and its multicast communication characteristics are still unclear. Limited studies on this avenue have used collective communication primitives for assessing the scalability of various SCI topologies from an analytical point of view [29, 30], while no known study has yet investigated the multicast performance of SCI.

Selected Multicast Algorithms

The algorithms analyzed in this study were defined in the literature [20, 22]. This section simply provides an overview of how they work and briefly points out their differences. Bound by the limits of available hardware, two unicast-based and three path-based multicast algorithms were selected, thereby keeping an acceptable degree of variety among different classes of multicast routing algorithms. In this dissertation, the aggregate collection of all destination nodes and the source node is called the *multicast group*. Therefore, for a given group with size d, there are d–1 destination nodes. Figure 2-4 shows how each algorithm operates for a group size of 10. The root node and the

destination nodes are clearly marked and the message transfers are indicated. Alphabetic labels next to each arrow indicate the individual paths, and the numerical labels represent the logical communication steps on each path.

Separate Addressing

Separate addressing is the simplest unicast-based algorithm in terms of algorithmic complexity. For small group sizes and short messages, separate addressing can be an efficient approach. However, for large messages and large group sizes, the iterative unicast transmissions may result in large host-processor overhead. Another drawback of this protocol is linearly increasing multicast completion latencies with increasing group sizes. Figure 2-4A shows separate addressing for a given multicast problem.

The U-torus Algorithm

The U-torus [20] is another unicast-based multicast algorithm that uses a binomial-tree approach to reduce the total number of required communication steps. For a given group of size d, the lower bound on the number of steps required to complete the multicast by U-torus will be $\lceil \log_2 d \rceil$. This reduction is achieved by increasing the number of covered destination nodes by a factor of 2 in each communication step. Figure 2-4B shows a typical U-torus multicast scenario.

Applying U-torus to this group starts with dimension ordering of all the nodes, including the root, based on their physical placement in the torus network given in a (column, row) format. The dimension-ordered node set is then rotated around to place the root node at the beginning of the ordered list as given below:

Φ={(1,1), (1,2), (1,3), (2,2), (2,4), (3,1), (3,3), (4,1), (4,2), (4,4)}

Φ'={(2,2), (2,4), (3,1), (3,3), (4,1), (4,2), (4,4), (1,1), (1,2), (1,3)}

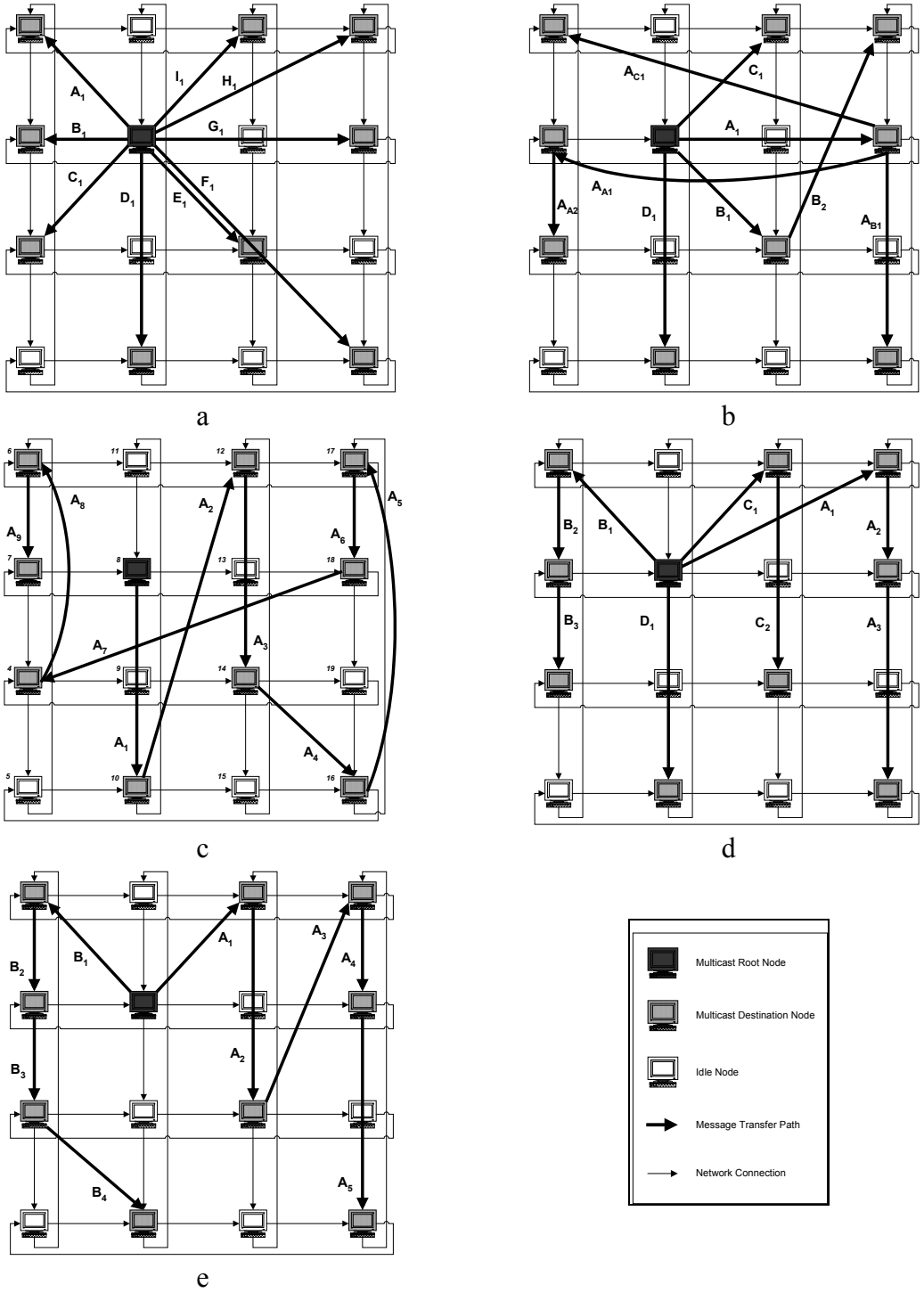


Figure 2-4. Selected multicast algorithms for torus networks. Multicast group size is 10. Individual message paths are marked alphabetically, and the numerical labels represent the logical communication steps for each message path. A) The separate addressing algorithm. B) The U-torus algorithm. C) The S-torus algorithm. D) The M_d-torus algorithm. E) The M_u-torus algorithm.

where Φ is the dimension-ordered group and Φ' denotes the rotated version of Φ. The order in Φ' also defines the final ranking of the nodes, as they are sequentially ranked starting from the leftmost node. As an example, for Φ' given above, node (2,2) has a ranking of 0, node (2,4) has a ranking of 1, and the node (1,3) has a ranking of 9.

After obtaining the Φ', the root node sends the message to the *center* node of Φ' to partition the multicast problem of size d into two subsets of size $\lceil d/2 \rceil$ and $\lfloor d/2 \rfloor$. The center node is calculated by Eq. 2-1 as described by Robinson *et al.* [20], where *left* denotes the ranking of the leftmost node, and *right* denotes the ranking of the rightmost node.

$$center = left + \left\lceil \frac{right - left + 1}{2} \right\rceil \tag{2-1}$$

For the group given above, the left is rank 0 and the right is 9, therefore the center is 5, which implies the node (4,2). The root node transmits the multicast message, and the new partition's subset information, D_{subset}, to the center node. Using the same example, at the end of the first step the root node will have the subset D_{subset_root} = {(2,2), (2,4), (3,1), (3,3), (4,1)} with the values of left and right being rank 0 and 4, respectively. The node (4,2) will have the subset $D_{subset\_(4,2)}$ = {(4,2), (4,4), (1,1), (1,2), (1,3)}with the values of left and right being again 0 and 4.

In the second step, the original root and the (4,2) node both act as root nodes, partitioning their respective subsets in two; and sending the multicast message to their subset's center node, along with the new partition's D_{subset} information. This process continues recursively, until all destination nodes have received the message.

The S-torus Algorithm

The S-torus, a path-based multicast routing algorithm, was defined by Robinson *et al.* [22] for wormhole-routed torus networks. It is a single-phase communication algorithm. The destination nodes are ranked and ordered to form a Hamiltonian cycle.

A Hamiltonian cycle is a closed circuit that starts and ends at the source node, where every other node is listed only once. For any given network, more than one Hamiltonian cycle may exist. The Hamiltonian cycle that S-torus uses is based on a ranking order of nodes, which is calculated with the formula given in Eq. 2-2 for a *k*-ary 2D torus.

$$l(u) = \left[\left(\sigma_0(u) + \sigma_1(u)\right) \bmod k\right] + k\left[\sigma_1(u)\right] \tag{2-2}$$

Here, $l(u)$ represents the Hamiltonian ranking of a node u, with the coordinates given as $(\sigma_0(u), \sigma_1(u))$. More detailed information about Hamiltonian node rankings can be found in [9]. Following this step, the ordered Hamiltonian cycle Θ is rotated around to place the root at the beginning. This new set is named as Θ'.

The root node then issues a multi-destination worm which visits each destination node one after another following the Θ' ordered set. At each destination node, the header is truncated to remove the visited destination address and the worm is re-routed to the next destination. The algorithm continues until the last destination node receives the message. Robinson *et al.* also proved that S-torus routing is deadlock-free [9].

Figure 2-4C shows the S-torus algorithm for the same example presented previously, for a torus network without wormhole routing. The Hamiltonian rankings are noted in the superscript labels of each node, where Θ and Θ' are obtained as:

$\Theta = \{^{4}(1,3), ^{6}(1,1), ^{7}(1,2), ^{8}(2,2), ^{10}(2,4), ^{12}(3,1), ^{14}(3,3), ^{16}(4,4), ^{17}(4,1), ^{18}(4,2)\}$

$\Theta' = \{^{8}(2,2), ^{10}(2,4), ^{12}(3,1), ^{14}(3,3), ^{16}(4,4), ^{17}(4,1), ^{18}(4,2), ^{4}(1,3), ^{6}(1,1), ^{7}(1,2)\}$

The M-torus Algorithm

Belying its simplicity, single-phase communication is known for large latency variations for a large set of destination nodes [31]. Therefore, to further improve the S-torus algorithm, Robinson *et al.* proposed the multi-phase multicast routing algorithm: M-torus [22]. The idea was to shorten the path lengths of the multi-destination worms to stabilize the latency variations and to achieve better performance by partitioning the multicast group. They introduced two variations of the M-torus algorithm, M_d-torus and M_u-torus. The M_d-torus algorithm uses a dimensional partitioning method, whereas M_u-torus uses a uniform partitioning mechanism. In both of these algorithms, the root node separately transmits the message to each partition and the message is then further relayed inside the subsets using multi-destination worms. The M_d-torus algorithm partitions the nodes based on their respective sub-torus dimensions, therefore eliminating costly dimension-switching overhead. For example, in a 3D torus, the algorithm will first partition the group into subsets of 2D planes of the network, and then into ringlets for each plane.

For a k-ary N-dimensional torus network, where k^N is the total number of nodes, the M_d-torus algorithm needs N steps to complete the multicast operation. By contrast, the M_u-torus algorithm tries to minimize and equalize the path length of each worm by applying a uniform partitioning. M_u-torus is parameterized by the partitioning size, denoted by r. For a group size of d, the M_u-torus algorithm with a partitioning size of r requires $\lceil \log_r(d) \rceil$ steps to complete the multicast operation. For the same example

presented previously, Figure 2-4D and Figure 2-4E show M_d-torus and M_u-torus, respectively, again assuming a network without wormhole routing where r=4.

Case Study

To comparatively evaluate the performance of the selected algorithms, an experimental case study is conducted over a high-performance unidirectional SCI torus network. The following subsections explain experiment details and the results obtained.

Description

There are 16 nodes in the case study testbed. Each node is configured with dual 1GHz Intel Pentium-III processors and 256MB of PC133 SDRAM. Each node also features a Dolphin SCI NIC (PCI-64/66/D330) with 5.3Gb/s link speed using Scali's SSP (Scali Software Platform) 3.0.1, Redhat Linux 7.2 with kernel version 2.4.7-10smp, mtrr patched, and write-combining enabled. The nodes are interconnected to form a 4×4 unidirectional torus.

For all of the selected algorithms, the polling notification method is used to lower the latencies. Although this method is known to be effective for achieving low latencies, it results in higher CPU loads, especially if the polling process runs for extended periods. To further decrease the completion latencies, the multicast-tree creation is removed from the critical path and performed at the beginning of each algorithm in every node.

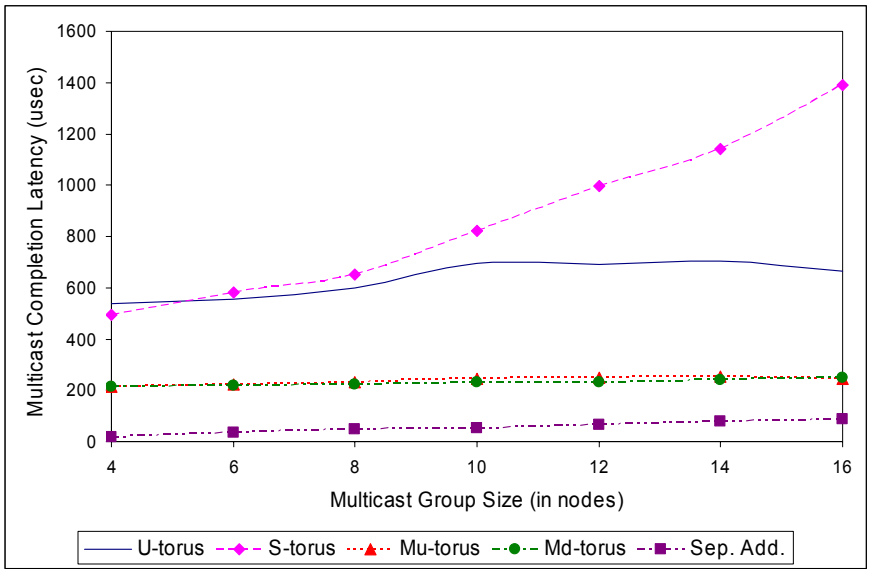
Throughout the case study, modified versions of the three path-based algorithms, S-torus, M_d-torus, and M_u-torus are used. These algorithms were originally designed to use multi-destination worms. However, as with most high-speed interconnects available on the market today, our testbed does not support multi-destination worms. Therefore, store-and-forward versions of these algorithms are developed.

On our 4-ary 2D torus testbed, M_d-torus partitions the torus network into simple 4-node rings. For a fair comparison between the M_d-torus and the M_u-torus algorithms, the partition length *r* of 4 is chosen for M_u-torus. Also, the partition information for U-torus is embedded in the relayed multicast message at each step. Although separate addressing exhibits no algorithmic concurrency, it is possible to provide some degree of concurrency by simply allowing multiple message transfers to occur in a pipelined structure. This method is used for our separate address algorithm.

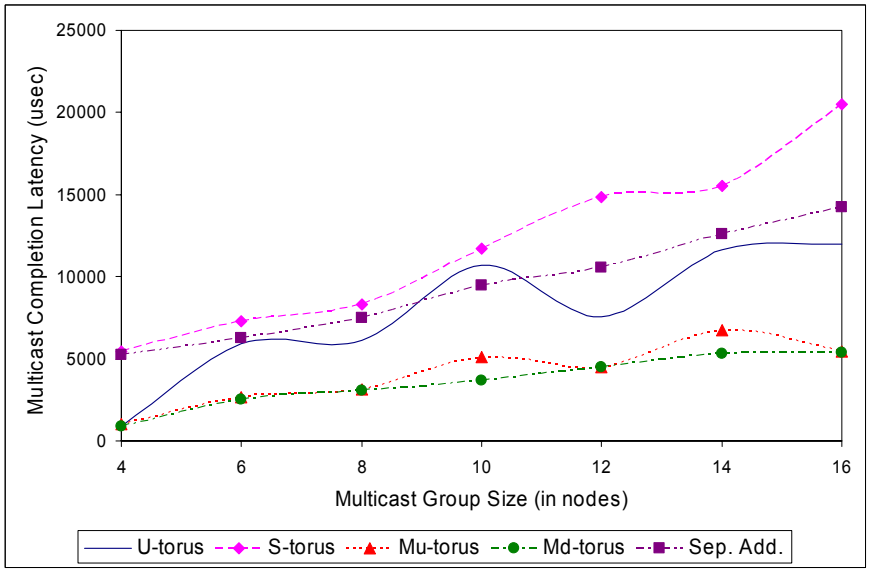
Case study experiments with the five algorithms are performed for various group sizes and for small and large message sizes. Each algorithm is evaluated for each message and group size 100 times, where each execution has 50 repetitions. The variance was found to be very small and the averages of all executions are used in this study. Four different sets of experiments are performed to analyze the various aspects of each algorithm, which are explained in detailed in the following subsections.

Multicast Completion Latency

Two different sets of experiments for multicast completion latency are performed, one for a message size of 2B and the other for a message size of 64KB. Figure 2-5 shows the multicast completion latency versus group size for small and large messages.
The S-torus algorithm has the worst performance for both small and large messages. Moreover, S-torus shows a linear increase in multicast completion latency with respect to the increasing group size, as it exhibits no parallelism in message transfers. By contrast, the separate addressing algorithm has a higher level of concurrency because of its design and performs best for small messages. However, it also presents linearly increasing completion latencies for large messages with increasing group size.



a



b

Figure 2-5. Completion latency vs. group size. A) Small messages. B) Large messages.

The M_d-torus and M_u-torus algorithms exhibit similar levels of performance for both small and large messages. The difference between these two becomes more distinctive at certain data points, such as 10 and 14 nodes for large messages. For group sizes of 10 and 14 the partition length for M_u-torus does not provide perfectly balanced partitions, resulting in higher multicast completion latencies. Finally, U-torus has nearly

flat latency for small messages. For large messages, it exhibits similar behavior to M_u-torus. Overall, separate addressing appears to be the best for small messages and groups, while for large messages and groups M_d-torus performs better compared to other algorithms.

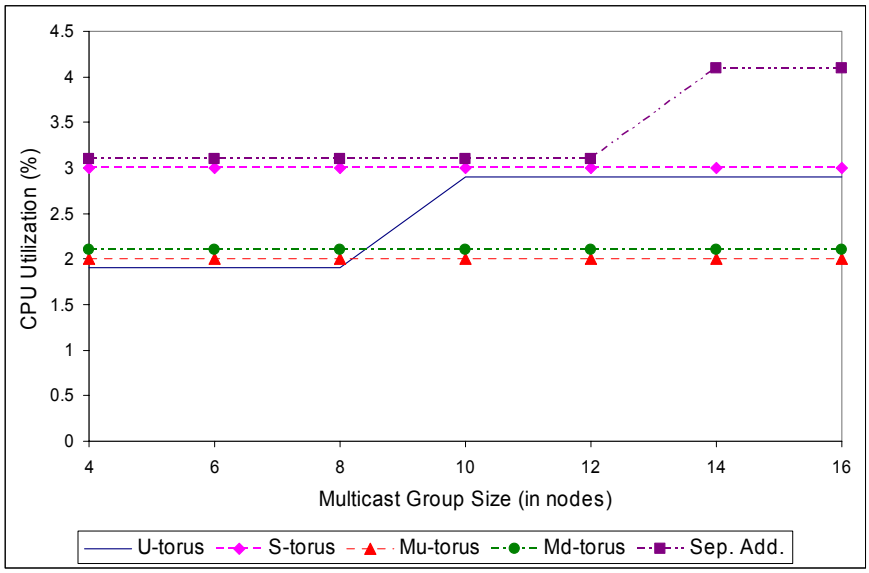
User-level CPU Utilization

User-level host processor load is measured using Linux's built-in `sar` utility. Figure 2-6 shows the maximum CPU utilization for the root node of each algorithm for small and large messages.

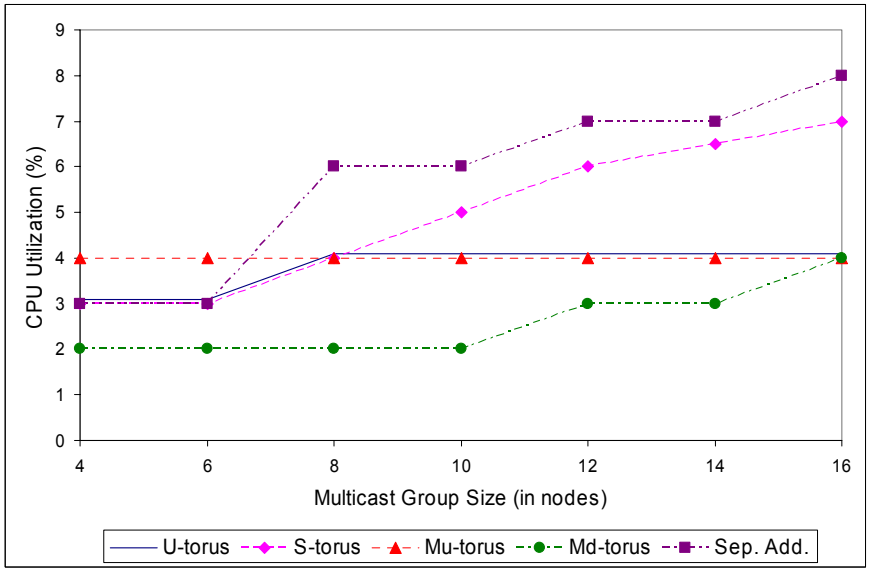
It is observed that S-torus exhibits constant CPU load for the small message size independent of the group size. However, for large messages, as the group size increases the completion latency also linearly increases as shown in Figure 2-5B, and the extra polling involved results in higher CPU utilization for the root node. This effect is clearly seen in Figure 2-6B.

In the separate addressing algorithm, the root node iteratively performs all message transfers to the destination nodes. As expected, this behavior causes a nearly linear increase in CPU load with increasing group size, which can be observed in Figure 2-6B.

By contrast, since the number of message transmissions for the root node stays constant, M_d-torus provides a nearly constant CPU overhead for small messages for every group size. For large messages and small group sizes, M_d-torus performs similarly. However, for group sizes greater than 10, the CPU utilization tends to increase because of the variations in the path lengths causing extended polling durations. Although these variations are the same for both small and the large messages, the effect is more visible for the large message size.



a



b

Figure 2-6. User-level CPU utilization vs. group size. A) Small messages. B) Large messages.

The M_u-torus algorithm exhibits behavior identical to M_d-torus for small messages. Moreover, for large messages, M_u-torus also provides higher but constant CPU utilization. For U-torus, the number of communication steps required to cover all destination nodes is given in previous sections. It is observed that at certain group sizes,

such as 4, 8, and 16, the number of these steps increases, therefore the CPU load also increases . This behavior of U-torus can be clearly seen in Figure 2-6.

Multicast Tree Creation Latency

Multicast tree creation latency of the SCI API is also an important metric since, for small message sizes, this factor might impede the overall communication performance. The multicast tree creation latency is independent of the message size. Figure 2-7 shows multicast tree creation latency versus group size.

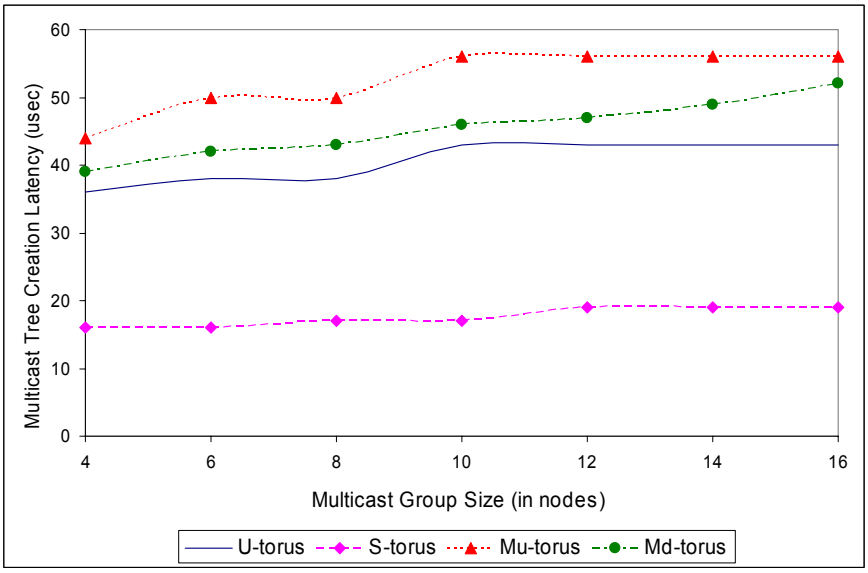


Figure 2-7. Multicast tree-creation latency vs. group size.

Figure 2-7 shows the multicast tree creation latencies for the four algorithms that use a tree-like group formation for message delivery. The M_u-torus and M_d-torus algorithms only differ in their partitioning methods as described before and both methods are quite complex compared to the other algorithms. This complexity is seen in Figure 2-7 as they exhibit the highest multicast tree-creation latencies.

The U-torus algorithm has a simple and distributed partitioning process and, compared to the two M-torus algorithms, it has lower tree-creation latency. Unlike the

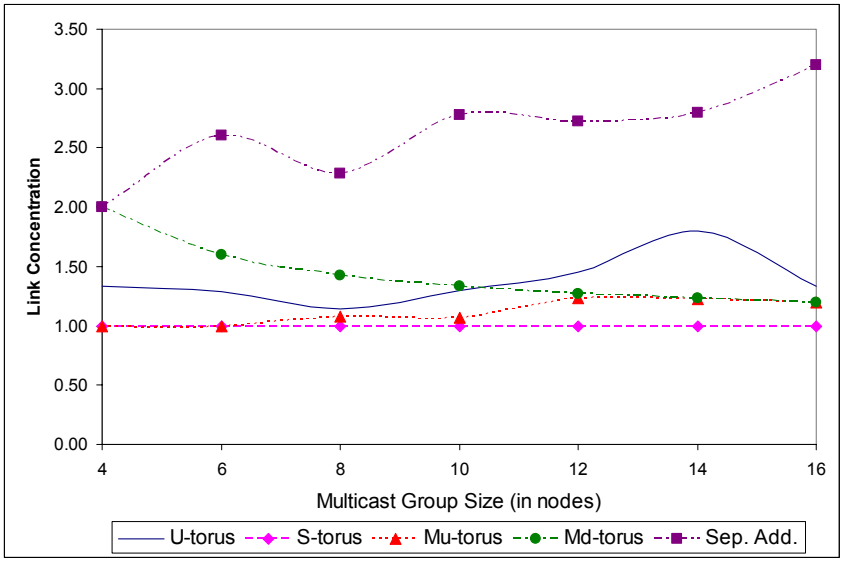
other tree-based algorithms, S-torus does not perform any partitioning and it only orders the destination nodes as described previously. Therefore, S-torus exhibits the lowest and a very-slowly and linearly increasing latency because of the simplicity of its tree formation.

Link Concentration and Concurrency

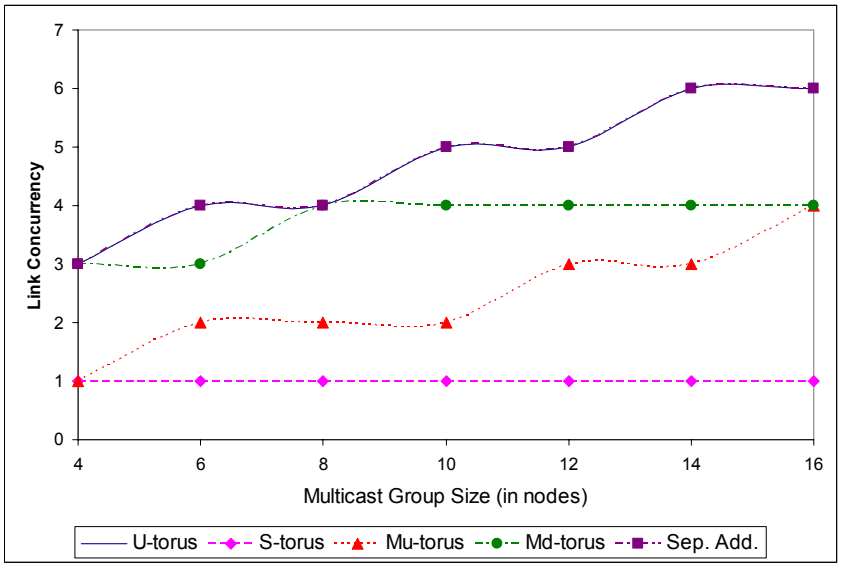
Link concentration is defined here as the ratio of two components: number of *link visits* and number of *used links*. *Link visits* is defined as the cumulative number of links used during the entire communication process, while *used links* is defined as the number of individual links used. Link concurrency is the maximum number of messages that are in transit in the network at any given time. Link concentration and link concurrency are given in Figure 2-8. Link concentration combined with the link concurrency illustrates the degree of communication balance. The concentration and concurrency values presented in Figure 2-8 are obtained by analyzing the theoretical communication structures and the experimental timings of the algorithms.

The S-torus algorithm is a simple chained communication and there is only one active message transfer in the network at any given time. Therefore, S-torus has the lowest and a constant link concentration and concurrency compared to other algorithms. By contrast, because of the high parallelism provided by the recursive doubling approach, the U-torus algorithm has the highest concurrency. Separate addressing exhibits an identical degree of concurrency to the U-torus, because of the multiple message transfers overlapping at the same time because of the network pipelining. The M_d-torus algorithm has inversely proportional link concentration versus increasing group size. In M_d-torus, the root node first sends the message to the destination header nodes, and they relay it to their child nodes. As the number of dimensional header nodes is constant (*k* in a *k*-ary

torus), with the increasing group size each new child node added to the group will increase the number of available links. Moreover, because of the communication structure of the M_d-torus, the number of used links increases much more rapidly compared to the number of link visits with the increasing group size. This trend asymptotically limits the decreasing link concentration to 1. The concurrency of



a



b

Figure 2-8. Communication balance vs. group size. A) Link concentration. B) Link concurrency.

M_d-torus is upper bounded by *k* as each dimensional header relays the message over separate ringlets with *k* nodes in each.

The M_u-torus algorithm has low link concentration for all group sizes, as it multicasts the message to the partitioned destination nodes over a limited number of individual paths as shown in Figure 2-4D, where only a single link is used per path at a time. By contrast, for a given partition length of constant size, an increase in the group size results in an increase in the number of partitions and an increase in the number of individual paths. This trait results in more messages being transferred concurrently at any given time over the entire network.

Multicast Latency Modeling

The experiments throughout SCI case study have investigated the performance of multicast algorithms over a 2D torus network having a maximum of 16 nodes. However, ultimately modeling will be a key tool in predicting the relative performance of these algorithms for system sizes that far exceed our current testbed capabilities. Also, by modifying the model, future systems with improved characteristics could be evaluated quickly and accurately. The model presented in the next subsections assumes an equal number of nodes in each dimension for a given *N*-dimensional torus network. The presented small-message latency model follows the *LogP* model [32].

The LogP is a general-purpose model for distributed-memory machines with asynchronous unicast communication [32]. Under LogP, at most at every *g* CPU cycles a new communication operation can be issued, and a one-way small-message delivery to a remote location is formulated as

$$t_{communication} = 2L + 2 \times (o_{sender} + o_{receiver}) \quad (2\text{-}3)$$

where L is the upper bound on the latency for a message delivery of a message from its source processor to its target processor, and o_{sender} *and* $o_{receiver}$ represent the sender and receiver communication overheads, respectively.

Today's high-performance NICs and interconnection systems are fast enough that any packet can be injected into the network as soon as the host processor produces it without any further delays [33]. Therefore, g is negligible for modeling high-performance interconnects. This observation yields the relaxed LogP model for one-way unicast communication, given as

$$t_{communication} = o_{sender} + t_{network} + o_{receiver} \tag{2-4}$$

where $t_{network}$ is the total time spent between the injection of the message into the network and drainage of the message.

Following this approach, a model is proposed to capture the multicast communication performance of high-performance torus networks on an otherwise unloaded system. The model is based on the concept that each multicast message can be expressed as sequences of serially forwarded unicast messages from root to destinations. The proposed model is formulated as

$$t_{multicast} = Max\left[o_{sender} + t_{network} + o_{receiver}\right] \quad \text{over} \quad \underset{paths}{\forall} \tag{2-5}$$

where the *Max*[ ] operation yields the total multicast latency for the deepest path over all paths and $t_{multicast}$ is the time interval between the initiation of multicast and the last destination's reception of the message. Figure 2-9 shows this concept on a sample binomial-tree multicast scenario. For this scenario, $t_{multicast}$ is determined over the deepest path, which is the route from node 0 to node 7.

Gonzalez *et al.* [27] modeled the unicast performance of direct SCI networks and observed that $t_{network}$ can be divided into smaller components as given by:

$$t_{network} = h_p \times L_p + h_f \times L_f + h_s \times L_s + h_i \times L_i \quad (2\text{-}6)$$

Here h_p, h_f, h_s, and h_i represent the total number of hops, forwarding nodes, switching nodes, and intermediate nodes, respectively. Similarly L_p, L_f, and L_s denote the propagation delay per hop, forwarding delay through a node, and switching delay through

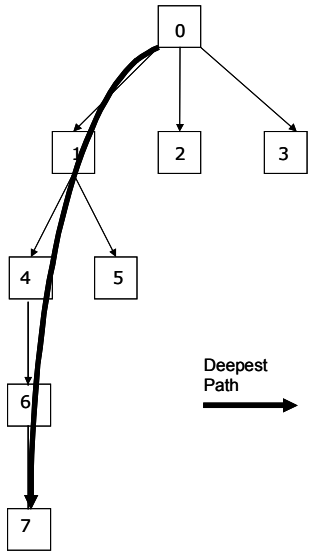


Figure 2-9. Sample multicast scenario for a given binomial tree.

a node, respectively. The L_i denotes the intermediate delay through a node, which is the sum of the receiving overhead of the message, processing time, and the sending overhead of the message. Figure 2-10A shows an arbitrary mapping of the problem given above in Figure 2-9 to a 2D 4×4 torus network. Figure 2.10B shows a visual break down of $t_{multicast}$ over the same arbitrary mapping.

Following the method outlined by Gonzales *et al.* [27] and using the obtained experimental results from the case study presented in this dissertation, the model parameters are measured and calculated for short message sizes. Assuming that the

electrical signals propagate through a copper conductor at approximately half the speed of light, and observing that our SCI torus testbed is connected with 2m long cables, resulted in 14ns of propagation latency per link. Since L_p represents the latency for the head of a message passing through a conductor, it is therefore independent of message size [27].

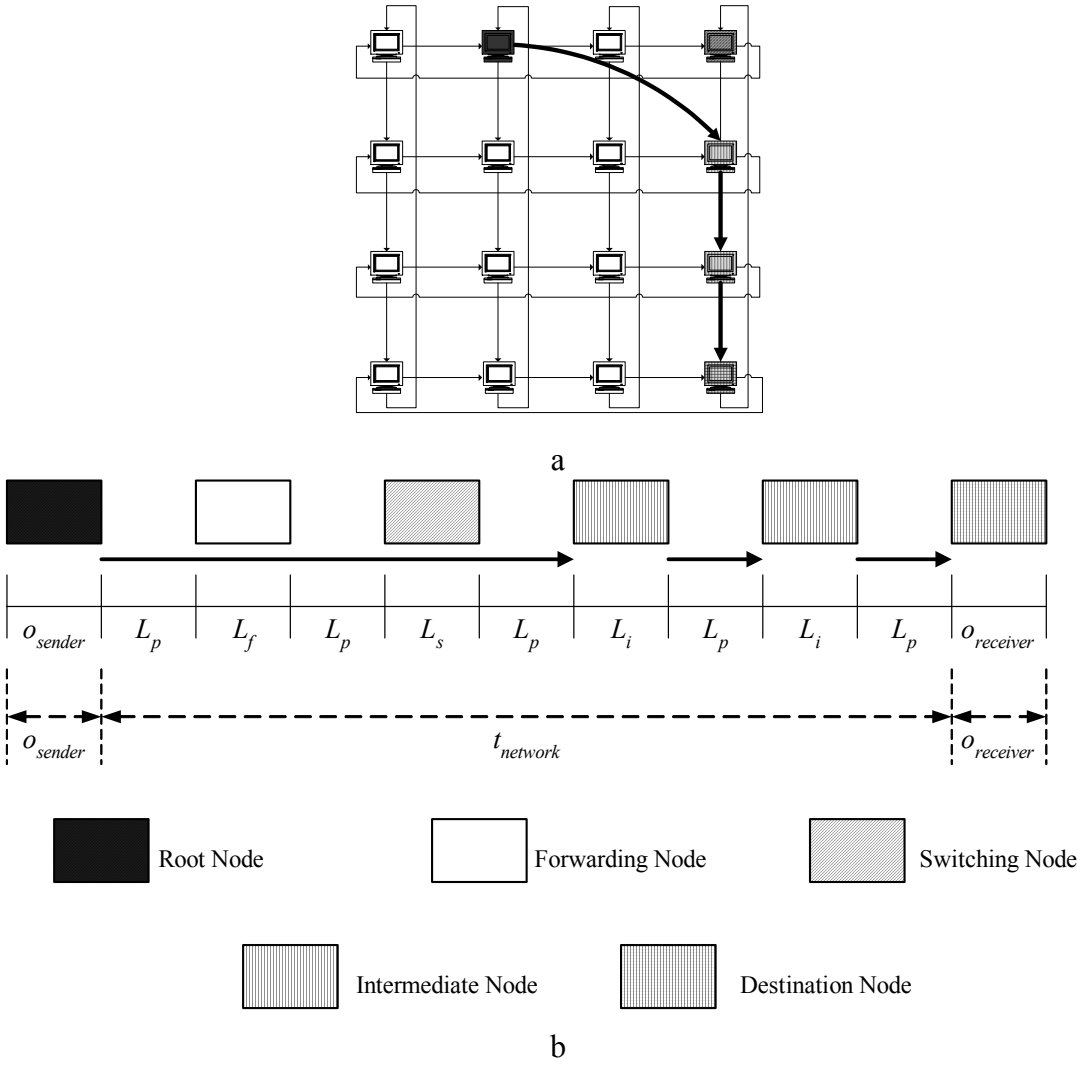


Figure 2-10. Small-message latency model parameters. A) Arbitrary mapping of the multicast problem given in Figure 2-9 to a 2D 4-ary torus. B) Break-down of the $t_{multicast}$ parameter over the deepest path for the given multicast scenario.

Model parameters o_{sender}, $o_{receiver}$, L_f and L_s were obtained through ring-based and torus-based API unicast experiments. Ring-based experiments were performed with two-, four-, and six-node ring configurations. For each setup, h_p and L_p are known. Inserting these values into Eqs. 2-4 and 2-6 and taking algebraic differences between two-, four-, and six-node experiments yields o_{sender}, $o_{receiver}$, and L_f. Switching latency, L_s, is determined through torus-based experiments by comparing latency of message transfer within a dimension versus between dimensions. Figure 2-11 shows the L_p, L_f, L_s, o_{sender}, and $o_{receiver}$ model parameters for various short message sizes.

The L_p, L_f, L_s, o_{sender}, and $o_{receiver}$ parameters are communication-bounded. These parameters are dominated by NIC and interconnect performance. The L_i is dependent upon interconnect performance and the node's computational power, and is formulated as:

$$L_i = o_{sender} + t_{process} + o_{receiver} \tag{2-7}$$

Calculations based on the experimental data obtained previously show that each multicast algorithm's processing load, $t_{process}$, is different. This load is composed mainly of the time needed to check the child node positions on the multicast tree and the calculation time of the child node(s) for the next iteration of the communication. Also, $t_{process}$ is observed to be the dominant component of L_i for short message sizes. Moreover, it can be easily seen that compared to the $t_{process}$ and L_i parameters, the L_p, L_f, and the L_s values are drastically smaller, and thus they are relatively negligible. Dropping these negligible parameters, Eq. 2-5 can be simplified and expressed as

$$t_{multicast} \approx (\textit{total number of destination nodes}) \times (o_{sender} + t_{process}) + o_{receiver} \tag{2-8}$$

for the separate addressing model. For the M_d-torus, M_u-torus, U-torus and S-torus algorithms the simplified model can be formulated as in Eq. 2-9. As can be seen, the modeling is straightforward for separate addressing and for the remaining algorithms the modeling problem is now reduced to identifying the number of intermediate nodes over the longest multicast message path. Moreover, without loss of generality, o_{sender} and $o_{receiver}$ values can be treated equal to one another [27] for simplicity and we represent them by a single parameter, o.

$$\begin{aligned} t_{multicast} &\approx Max\left[o_{sender} + h_i \times L_i + o_{receiver}\right] \quad \text{over} \quad \underset{paths}{\forall} \\ &\approx Max\left[2 \times o + h_i \times L\right] \quad \text{over} \quad \underset{paths}{\forall} \end{aligned} \tag{2-9}$$

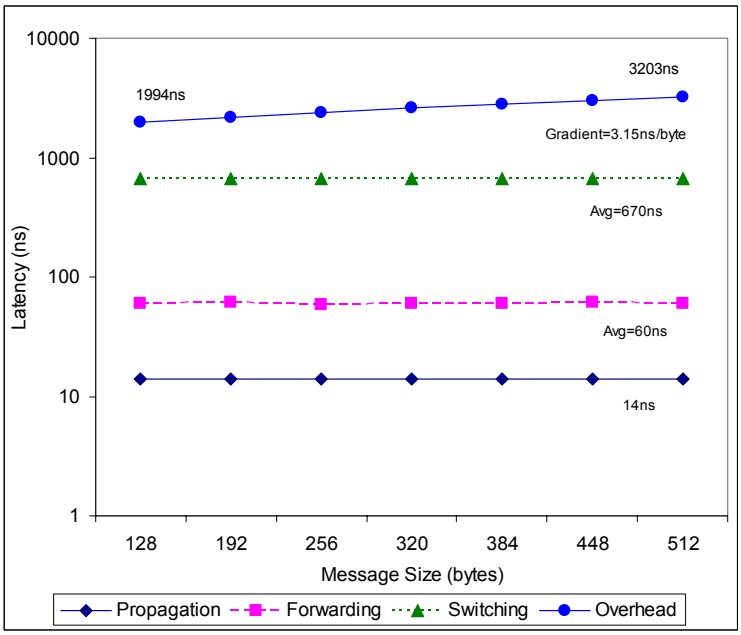


Figure 2-11. Measured and calculated model parameters for short message sizes: L_p=14ns, L_f=60ns, L_s=670ns, and o=1994+3.15×(M-128)ns.

Of course, variable L_i is not involved in the separate addressing algorithm. The reason is simply that separate addressing consists of a series of unicast message transmissions from source to destination nodes so there are no intermediate nodes that are needed to relay the multicast message to other nodes. Table 2-1 shows the $t_{process}$ and L_i values for short

message sizes. The $t_{process}$ parameter is independent of the multicast message and group size but strictly dependent on the performance of the host machine. Therefore, for different computing platforms, different $t_{process}$ values will be obtained

Table 2-1. Calculated $t_{process}$ and L_i values.

	$t_{process}$ (µs)	L_i (µs)
Separate Addressing	7	N/A
M_d-torus	206	206+2×o
M_u-torus	201	201+2×o
U-torus	629	629+2×o
S-torus	265	265+2×o

The following subsections will discuss and provide more detail about the simplified model for each multicast algorithm. For each algorithm, the modeled values, the actual testbed measurements, and the modeling error will also be presented.

The Separate Addressing Model

With the separate addressing algorithm, for a group size of G, there are G-1 destination nodes that the root node alone must serve. Therefore, a simplified model for separate addressing can be expressed as given in Eq. 2-10.

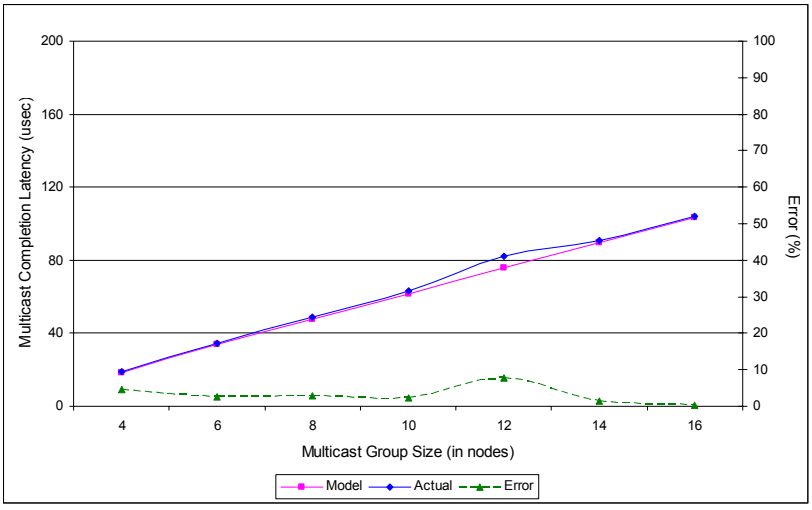


Figure 2-12. Simplified model vs. actual measurements for separate addressing algorithm.

$$t_{multicast} \approx (G-1)\times(o_{sender} + t_{process}) + o_{receiver} \quad (2\text{-}10)$$

Figure 2-12 shows the simplified model and the actual measurements with various multicast group sizes for a 128-byte multicast message using the o and $t_{process}$ values previously defined. The results show that the model is accurate with an average error of ~3%.

The M_d-torus Model

For M_d-torus, the total number of intermediate nodes for any communication path is observed to be a function of G, the multicast group size, and k, the number of nodes in a given dimension. The simplified M_d-torus model is formulated given in Eq. 2-11.

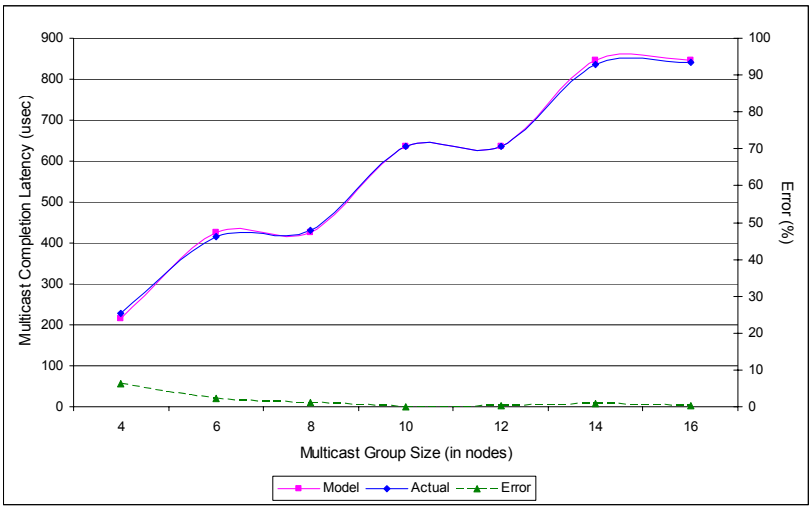


Figure 2-13. Simplified model vs. actual measurements for M_d-torus algorithm.

$$t_{multicast} \approx 2\times o + \left\lceil \frac{G}{k} \right\rceil \times L_i \quad (2\text{-}11)$$

The simplified model and actual measurements for various group sizes with 128-byte messages are plotted in Figure 2-13. As can be seen the simplified model is accurate with an average error of ~2%.

The M_u-torus Model

The number of partitions for the M_u-torus algorithm, denoted by p, is a byproduct of the multicast group size, G, and the partition length, r. For systems with r equal to G, there exists only one partition and the multicast message is propagated in a chain-type communication mechanism among the destination nodes. Under this condition, the number of intermediate nodes is simply two less than the group size. The subtracted two are the root and the last destination nodes. For systems with partitions equal to or more than 2, the number of intermediate nodes becomes a function of the group size, partition length and the number of nodes in a given dimension. The simplified model is given as:

$$t_{multicast} \approx \begin{cases} 2 \times o + \left(G - r \times \left(1 - \left\lfloor \frac{G}{r} \right\rfloor + \left| r - k \right| \right) \right) \times L_i, & p \geq 2 \\ 2 \times o + (G-2) \times L_i, & p < 2 \end{cases} \tag{2-12}$$

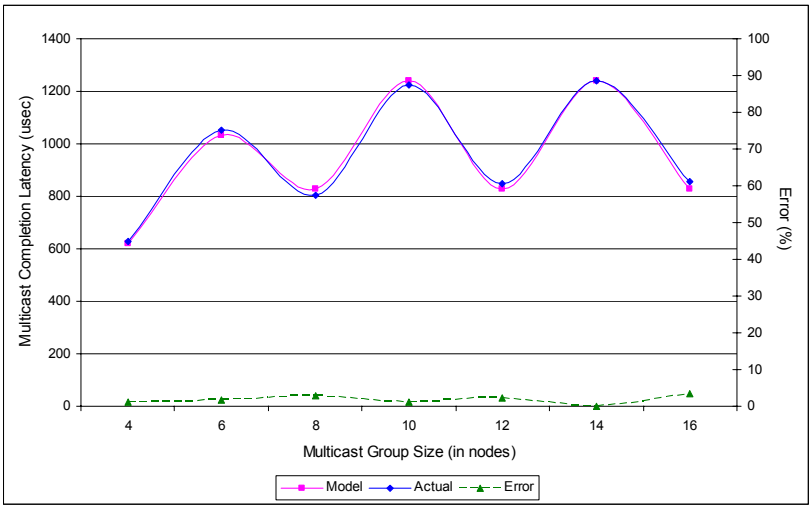


Figure 2-14. Simplified model vs. actual measurements for M_u-torus algorithm.

Figure 2.14 shows the small-message model versus actual measurements for 128-byte messages. The results show that the model is accurate with an average error of ~2%.

The U-torus Model

For U-torus the minimum number of communication steps required to cover all destination nodes can be expressed as $\lceil \log_2 G \rceil$. The number of intermediate nodes in the U-torus algorithm is a function of the minimum required communication steps, the group size and the number of nodes in a given dimension. The simplified U-torus model is given as:

$$t_{multicast} \approx 2 \times o + \lceil \log_2 G \rceil \times \frac{(G \bmod k) + k}{k} \times L_i \tag{2-13}$$

Figure 2-15 shows the short-message model and actual measurements for various group sizes. The results show the model is accurate with an average error of ~2%.

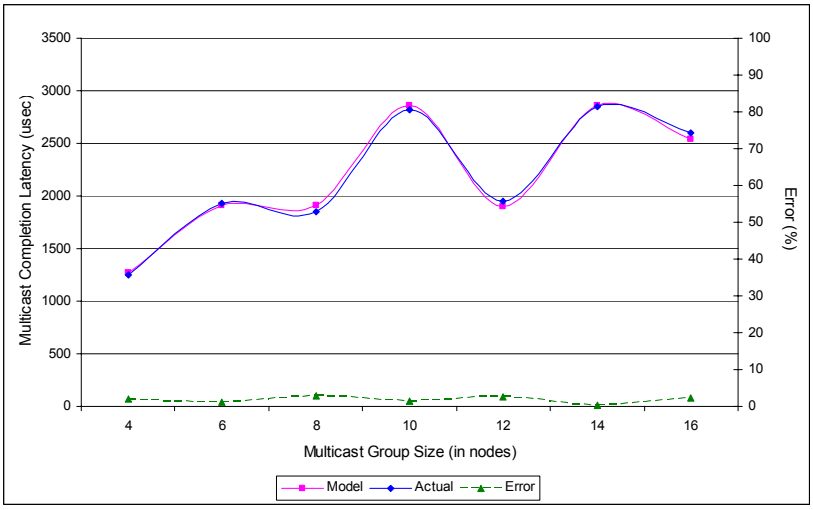


Figure 2-15. Simplified model vs. actual measurements for U-torus algorithm.

The S-torus Model

The S-torus is a chain-type communication algorithm and can be modeled identically to the single partition case of the M_u-torus algorithm. The simplified model for S-torus is formulated as:

$$t_{multicast} \approx 2 \times o + (G - 2) \times L_i \tag{2-14}$$

The results from the short-message model versus actual measurements for 128-byte messages are shown in Figure 2-16. As previously stated, S-torus routing is based on Hamiltonian circuit. This type of routing ensures that each destination node will receive only one copy of the message, but some forwarding nodes (i.e., non-destination nodes that are on the actual message path) may be visited more than once for routing purposes. Moreover, depending on the group size, single-phase routing traverses many unnecessary channels, creating more traffic and possible contention. Therefore, S-torus has unavoidably large latency variations because of the varying and sometimes extremely long message paths [31]. The small-message model presented is incapable of tracking these large variations in the completion latency that are inherent to the S-torus multicast

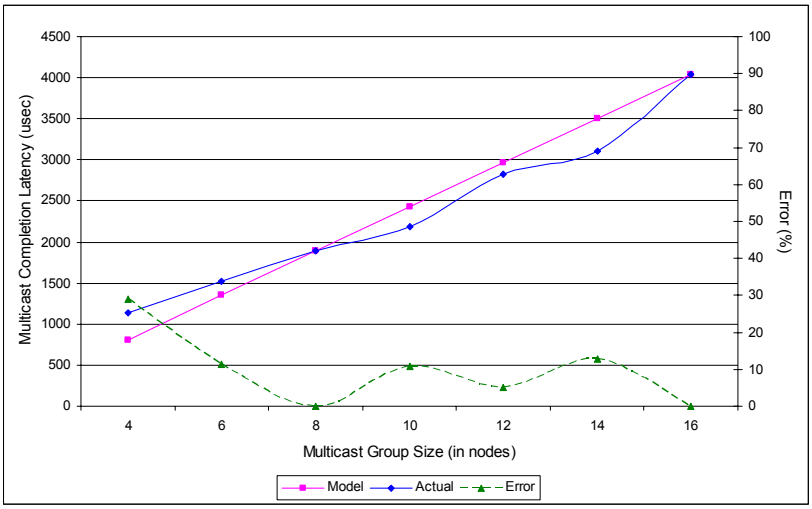


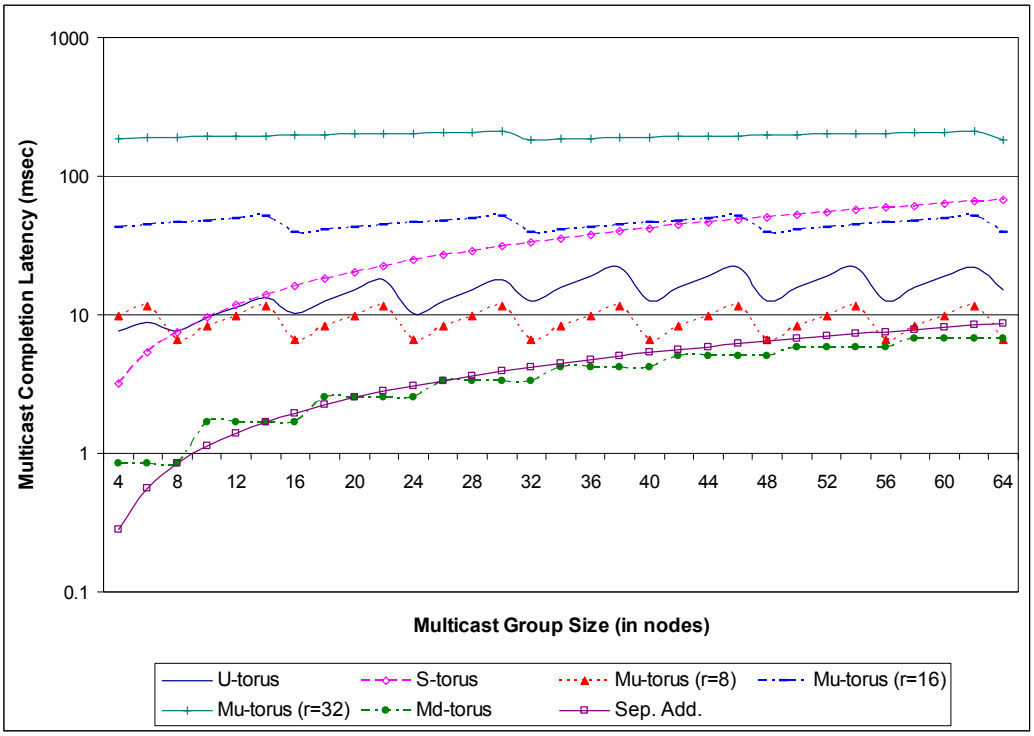
Figure 2-16. Simplified model vs. actual measurements for S-torus algorithm.

algorithm. The modeling error is relatively high, unlike the other algorithms evaluated in this study. The instability of the modeling error is not expected to lessen with the increasing group size.

Analytical Projections

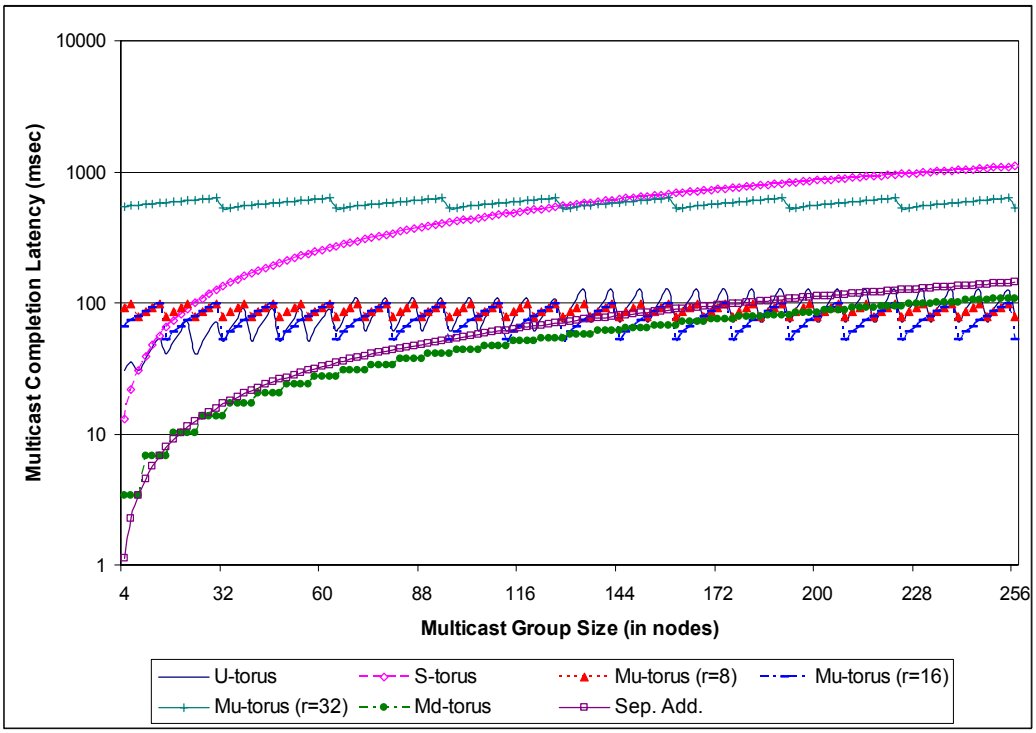
To evaluate and compare the short-message performance of the multicast algorithms for larger systems, the simplified models are used to investigate 2D torus-based parallel systems with 64 and 256 nodes. The effects of different partition lengths (i.e., r=8, r=16, r=32) for the M_u-torus algorithm over these system sizes are also investigated analytically with these projections. The results of the projections are plotted in Figure 2-17.

The M_d-torus algorithm has step-like completion latency caused by the fact that, with every new k destination nodes added to the group, a new ringlet is introduced to the multicast communication which increases the completion latency. The optimal performance for the 8×8 torus network is obtained when the M_u-torus has a partition length of 8 and for the 16×16 torus network when the partition length is 16. Therefore, it is surmised that the optimal partition length for M_u-torus is equal to k for 2D SCI tori.



a

Figure 2-17. Continued.



b

Figure 2-17. Small-message latency projections. A) Projection values for a 8×8 torus system. B) Projection values for a 16×16 torus system.

The U-torus, on average, has slowly increasing completion latency with increasing group sizes. The U-torus, M_u-torus, and M_d-torus algorithms all tend to have similar asymptotic latencies. The S-torus and separate addressing algorithms, as expected, have linearly increasing latencies with increasing group sizes. Although separate addressing is the best choice for small-scale systems, it loses its advantage with increasing group sizes for short messages. The S-torus, by contrast, again proves to be a poor choice, as it is simply the worst performing algorithm for group sizes greater than 8.

Summary

This phase of the dissertation has investigated the multicast problem on high-performance torus networks. Software-based multicast algorithms from the literature were applied to the SCI network, a commercial example of a high-performance

torus network. Experimental analysis and small-message latency models of these algorithms are introduced. Analytical projections based on the verified small-message latency models for larger size systems are also presented.

Based on the experimental results presented earlier, it is observed that the separate addressing algorithm is the best choice for small messages or small group sizes from the perspective of multicast completion latency and CPU utilization because of its simple and cost effective structure. The M_d-torus algorithm performs best from the perspective of completion latency for large messages or large group sizes, because of the balance provided by its use of dimensional partitioning. In addition, M_d-torus incurs a very low CPU overhead and achieves high concurrency for all the message and group sizes considered. The U-torus and M_u-torus algorithms perform better when the individual multicast path depths are approximately equal. Furthermore, the M_u-torus algorithm exhibits its best performance when group size is an exact multiple of the partition length. The U-torus and M_u-torus algorithms have nearly constant CPU utilizations for small and large messages alike. Moreover, the U-torus algorithm has the highest concurrency among all algorithms evaluated, because of the high parallelism provided by the recursive-doubling method. The S-torus algorithm is always the worst performer from the perspective of completion latency and CPU utilization because of its lack of concurrency and its extensive communication overhead. As expected, S-torus exhibits a nearly linear increase in completion latency and CPU utilization for large messages with increasing group size.

The small-message latency models, using only a few parameters, capture the essential mechanisms of multicast communication over the given platforms. The models

are accurate for all evaluated algorithms except the S-torus algorithm. Small-message multicast latency projections for larger torus systems are provided using these models. Projection results show that with increasing group size the U-torus, M_u-torus, and M_d-torus algorithms tend to have asymptotically bounded similar latencies. Therefore, it is possible to choose an optimal multicast algorithm among these three for larger systems, based on the multicast completion latency and other metrics such as CPU utilization or network link concentration and concurrency. It is also possible and straightforward to project the multicast performance of larger-scale 2D torus networks with our model. Projected results show that S-torus and separate addressing have unbounded and linearly increasing completion latencies with increasing group sizes, which makes them unsuitable for large-scale systems. Applying the simplified models to other torus networks and/or multicast communication schemes is possible with a minimal calibration effort.

These results make it clear that no single multicast algorithm is best in all cases for all metrics. For example, as the number of dimensions in the network increases, the M_d-torus algorithm becomes dominant. By contrast, for networks with fewer dimensions supporting a large number of nodes, the M_u-torus and the U-torus algorithms are most effective. Separate addressing is an efficient and cost-effective choice for small-scale systems. Finally, S-torus is determined to be inefficient as compared to the alternative algorithms in all the cases evaluated. This inefficiency is caused by the extensive length of the paths used to multicast, which in turn leads to long and widely varying completion latencies and a high degree of root-node CPU utilization.

CHAPTER 3
MULTICAST PERFORMANCE ANALYSIS AND MODELING FOR HIGH-SPEED INDIRECT NETWORKS WITH NIC-BASED PROCESSORS

Chapter 2 was focused on investigating the topological characteristics of high-performance torus networks for multicast communication. An experimental case study was presented for SCI torus networks. Chapter 3 determines the optimum level of work sharing between the host processor and the NIC processor for multicast communication. The goal is to achieve an optimum balance between multicast completion latency and host processor load. With its onboard NIC RISC processor, Myrinet [18] is an example of such "intelligent" high-performance interconnects. The following sections explain the details of a study to achieve an optimal work balance between the host and the NIC processors for multicast communication over a Myrinet interconnect.

Myrinet

Myrinet is an indirect high-performance interconnect constructed of switching elements and host interfaces using point-to-point links. The core of the switching element is a pipelined crossbar that supports non-blocking, wormhole routing of unicast packets over bi-directional links up to 2.0Gbps.

A crossbar switching chip is the building block of a Myrinet network. It can be used to build a non-blocking switch. It can also be interconnected to build arbitrary topologies, such as switch-based stars, n-dimensional meshes or Multistage

Interconnection Network (MIN) topologies. Myrinet provides reliable, connectionless unicast message delivery between communication end-points called ports.

Myrinet NICs are equipped with a programmable RISC processor (LANai), three DMA engines, and SRAM memory. Myrinet also supports user-level host access to the NIC bypassing the operating system for decreased host-to-NIC access latencies and increased throughput. The latest version of Myrinet also supports 64-bit 133MHz PCI-X interfaces. Figure 3-1 shows the architectural block diagram of a Myrinet NIC.

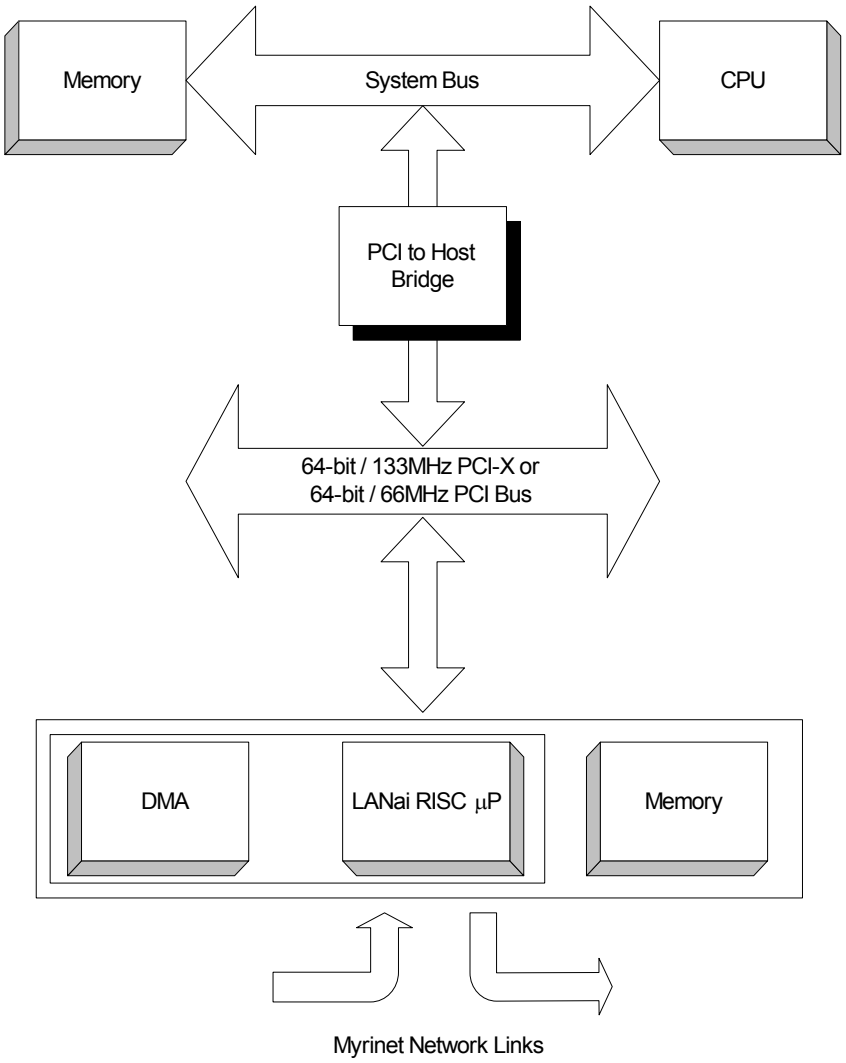


Figure 3-1. Architectural block diagram of Myricom's PCI-based Myrinet NIC.

A striking feature of a Myrinet interconnect is the on-board NIC processor. The main task of this processor is to offload work from the host processor on communication events. This programmable, 32-bit, RISC processor runs at 66 or 133 MHz, which is roughly an order of magnitude slower than today's host processors (1000-3000 MHz).

Related Research

Myrinet is the most successful and most widely deployed commercial high-performance interconnect for clusters. It has received extensive attention from academia and industry. Among the many topics of research, collective communication on Myrinet exploiting the NIC processor is of particular interest to this phase of the dissertation. As Myrinet does not support multicasting in hardware, designing efficient and optimal software-based collective communication is the goal of many researchers.

All communication-related operations are performed on the Myrinet NIC RISC processor in NIC-based collective communication. This approach is a well-studied method to avoid expensive host-NIC interaction and to reduce system overhead and network transaction latency [34-38]. It was observed that under such communication schemes, very low host CPU loads can be obtained at a cost of increased overall multicast completion latencies. This trait is caused by the fact that the NIC processor is considerably slower (66 or 133 MHz) compared to the host CPU.

On the multicasting side, Verstoep *et al.* [35] extended the Illinois Fast Messages (FM) protocol to produce a totally-ordered, reliable multicasting scheme that is fully processed by the Myrinet NIC processor. The performance of this scheme was evaluated over various spanning-tree multicast protocols. Kesavan *et al.* [36] presented a simulative evaluation of NIC-based multicast performance of an optimal binomial tree algorithm with packetization support at the NIC level. Bhoedjang *et al.* [37] simplified

and improved the scalability of Verstoep's design by developing another multicasting scheme that is completely performed by the NIC co-processor. Bunitas *et al.* [38] presented a NIC-based barrier operation over the Myrinet/GM messaging layer and reported that they achieved performance improvement by a factor of 1.83 compared to the host-based operations. An analytical model for performance estimation of the barrier operation was also presented.

The NIC-assisted multicasting was proposed as an answer to improve the multicast completion latencies of NIC-based multicast communication schemes while obtaining similar degrees of host CPU loads. Bunitas *et al.* [39] presented a NIC-assisted binomial tree multicast scheme over FM to improve the latency characteristics of NIC-based multicast algorithms.

Among other multicast-related Myrinet research, Sivaram *et al.* [40] proposed enhancements to a network switch architecture to support reliable hardware-based multicasting. They have also presented a detailed latency model of their hardware-based multicast communication approach.

This study complements and extends previous work by providing experimental evaluations and small-message latency models of host-based, NIC-based and NIC-assisted multicast schemes for obtaining optimal host CPU loads and multicast completion latencies. These multicasting schemes are analyzed for the binomial and binary trees, serial forwarding and separate addressing multicast algorithms. Accurate small-message latency models for these algorithms are also developed and, using these models, multicast completion latencies for larger systems are projected. Results of these comparisons determine the optimum balance of support between the host processor and

the NIC co-processor for multicast communication under various networking scenarios. The next section provides detailed information about the different multicasting schemes.

The Host Processor vs. NIC Processor Multicasting

Multicast communication primitives can be implemented at two different extremes for interconnects with an onboard NIC processor, namely, host-based and NIC-based. Between these two extremes there lies another level of implementation: NIC-assisted multicasting. The following subsections will provide detailed information about these three design strategies.

The Host-based Multicast Communication

Host-based multicast communication is the easiest and most conventional way of implementing a multicast communication primitive. In this scheme, the host processor handles all multicasting tasks, such as multicast tree creation and issuing of the unicast send and receive operations. This type of implementation introduces increased CPU load, resulting in lower computation/communication overlap available for parallel programs. However, as the fast host processor performs all the tasks, host-based multicasting achieves small multicast-completion latencies.

The NIC-based Multicast Communication

In the NIC-based scheme, the NIC co-processor handles all multicasting tasks instead of the host processor. Therefore this scheme provides a reduced CPU load and high computation/communication overlap for parallel programs. However, the Myrinet NIC processor is roughly an order of magnitude slower than modern host processors. Performing all the tasks on this relatively slow processor increases multicast completion latency.

The NIC-assisted Multicast Communication

Between these two extremes a compromise has been proposed to this problem. It is called NIC-assisted multicasting. In this approach, work is shared between the two processors with the host processor handling computationally intensive multicast tasks, such as multicast tree creation, and the NIC processor handling communication-only multicast tasks, such as unicast send and receive operations. This solution presents low multicast completion latency with moderate CPU loads, resulting in an acceptable computation/communication overlap available for parallel programs.

The main difference between these three multicast schemes is on how they initiate the multicast send operation and how the intermediate nodes behave for relaying the multicast message to their child nodes. Figure 3-2 shows a binomial tree multicast operation graphically. All three schemes are applicable to any multicast algorithm.

As can be seen from Figure 3-2, in the host-based scheme the host processor issues each multicast send one after another as if the NIC had no onboard processor. Also, upon the reception of the multicast message, the intermediate nodes pass the message to local host buffers immediately and the host processor processes it and determines the child nodes for the next communication step. After the child nodes have been determined, the host processor also issues the necessary multicast sends.

In the NIC-based multicast scheme the host processor issues only one send command. This command includes the multicast message data and the destination node set. The NIC processor creates the multicast tree based on this destination node set information, and then issues all the necessary multicast sends one after another. As soon as the intermediate node receives the multicast message the NIC processor analyzes the header,

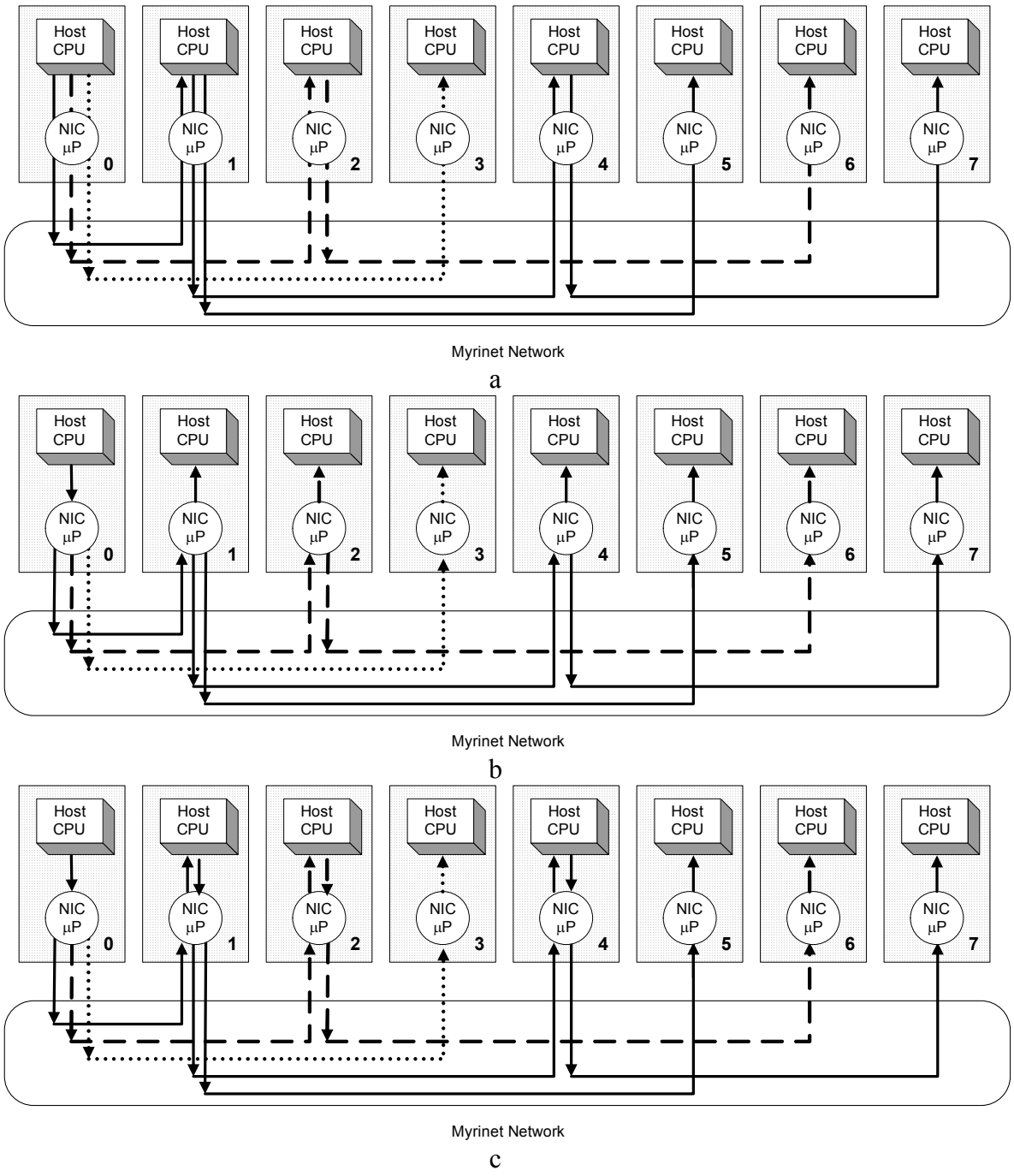


Figure 3-2. Possible binomial tree multicasting variations for Myrinet interconnects. The difference between these schemes can be observed on the host-NIC interactions at nodes 0 and 1. A) Host-based multicast communication scheme. B) NIC-based multicast communication scheme. C) NIC-assisted multicast communication scheme.

transfers the multicast data to local buffers for processing, notifies the host processor, and issues the necessary multicast sends to the child nodes. Depending on the DMA capability of the NIC, transfer of the multicast data to local buffers and issuing the multicast messages to child nodes can occur concurrently.

As previously stated, the NIC-assisted scheme regulates the two processors to share the workload between them. In this scheme, for a multicast send operation the host processor issues only one send command to the NIC. However, unlike the NIC-based scheme, the host processor in the NIC-assisted scheme encapsulates the destination node set, and the pre-computed multicast tree along with the data. The NIC processor only executes the communication commands, which results in a shorter network interface latency as compared to the NIC-based version. Upon reception, the NIC processor immediately transfers the multicast message to the local buffers and notifies the host processor. The host processor then processes the data and creates the child multicast tree, passing it back to the NIC processor. The NIC processor, therefore, issues multicast sends without performing any processing.

Case Study

To comparatively evaluate the performance of the host-based, NIC-based, and NIC-assisted communication schemes, an experimental case study is conducted over a 16-node Myrinet network. The following subsections explain experiment details and the results obtained.

Description

The case study is performed on a 16-node system, each node composed of dual 1GHz Intel PentiumIII processors, 256MB of PC133 SDRAM, ServerSet III LE (rev 6) chipset, and a 133MHz system bus. A Myrinet network is used as the high-speed

interconnect, where each node has a M2L-PCI64A-2 Myrinet NIC with 1.28 Gb/s link speed using Myricom's GM-1.6.4 and Redhat Linux 8.0 with kernel version 2.4.18-14smp. The nodes are interconnected to form a 16-node star network using 16 3m Myrinet LAN cables and a 16-port Myrinet switch.

Binomial tree, binary tree, serial forwarding and separate addressing algorithms are developed for the host-based, NIC-based, and NIC-assisted communication schemes individually. All of the multicast algorithms are evaluated for small (2B) and large (64KB) message sizes and multicast group sizes of 4, 6, 8, 10, 12, 14, and 16. Multicast completion latency, multicast tree creation latency, host CPU utilization, link concentration, and concurrency are measured for each combination of multicast algorithms, communication schemes, and message and group sizes previously described.

The Myrinet interconnect runs a propriety software, GM, on hosts. GM is a software implementation of the Virtual Interface Architecture (VIA) [41]. Like VIA, GM is targeted for providing low-latency, OS-bypassing interactions between user-level applications and communication devices. In GM, the OS is only responsible for establishing the communication channels and enforcing required protection mechanisms through the GM driver. Once the initial setup phase is completed, host to NIC communications are performed through the GM library. Following the setup phase, both host and NIC are able to initiate one-side data transfers to each other using the provided DMA engines in the Myrinet NIC hardware. Figure 3-3 shows the three-level GM-software architecture. All of the host-based multicast algorithms used in this research are designed in the user-level application layer on top of GM-1.6.4 provided by Myricom.

The Myrinet Control Program (MCP) provided by Myricom has been modified as follows to fit the new communication schemes for both the NIC-based and the NIC-assisted multicast algorithms. The MCP is a firmware that runs on the Myrinet NIC

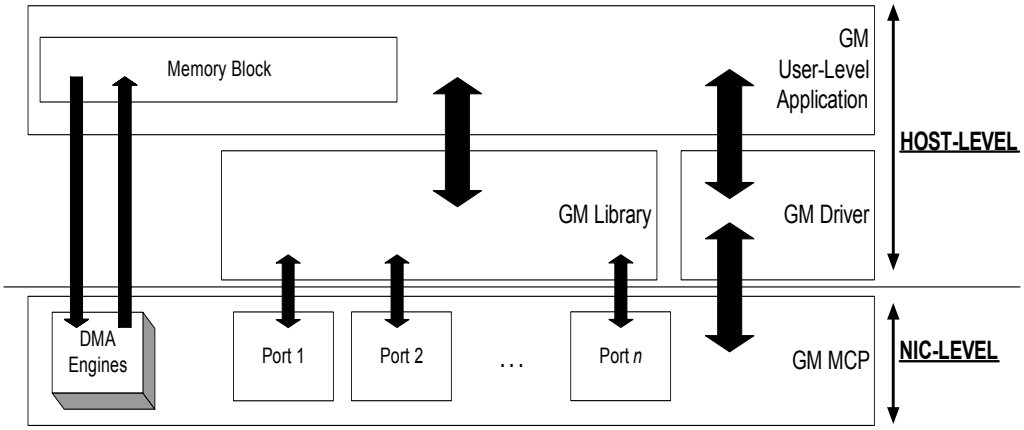


Figure 3-3. Myricom's three-layered GM software architecture.

RISC processor. The original MCP has four state machines. These state machines are *SDMA*, *SEND*, *RECV*, and *RDMA*, as shown in Figure 3-4. Each one is responsible for a particular task and these tasks will be explained in detail in the following subsection. Both the NIC-based and the NIC-assisted communication schemes use NIC-issued multi-unicast messages which required modification of the state machine code. Table 3-1 shows the pseudocode of the overall process for both the NIC-assisted and the NIC-based communication schemes for the root, intermediate and destination nodes, including the host and NIC tasks and interactions. Parts in italics are either performed or initiated by the host CPU and the rest is performed by the NIC processor. The SDMA process includes the host writing to NIC memory and signaling the NIC upon completion of the write operation. The RDMA process includes the NIC writing to the host memory and signaling it upon the completion of the write operation. The updated state machines

added an extra 8μs to the unicast sends whereas the unmodified minimum one-way latency was measured as 17μs from the host level using the unmodified MCP code.

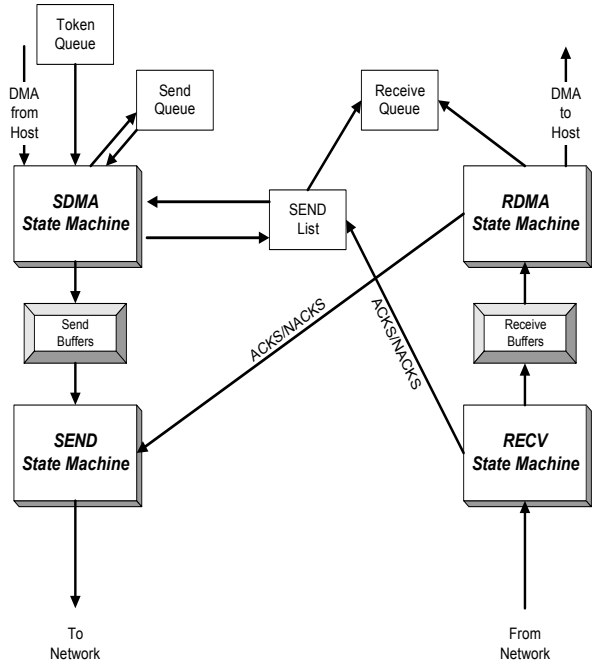


Figure 3.4. The GM MCP state machine overview.

Table 3-1. Pseudocode for NIC-assisted and NIC-based communication schemes.

NIC-Assisted Multicast (Root Node)	NIC-Based Multicast (Root Node)
Obtain multicast host names *Obtain GM MCP base address pointer* *Build multicast tree* *SDMA* *Wait for completion* Do multicast RDMA	*Obtain multicast host names* *Obtain GM MCP base address pointer* *SDMA* *Wait for completion* Build multicast tree Do multicast RDMA
NIC-Assisted Multicast (Intermediate and Destination Nodes)	**NIC-Based Multicast (Intermediate and Destination Nodes)**
Listen for incoming multicast calls Receive message RDMA *Check multicast tree* *SDMA* Do multicast	Listen for incoming multicast calls Receive message Check multicast tree Relay message to root/child hosts RDMA

Case study evaluations of the host-based, NIC-based, and NIC-assisted communication schemes are undertaken for each of the four multicast algorithms. Each

experiment is performed for each message and group size, for 100 executions, where every execution has 50 repetitions. Four different sets of metrics are probed in each experiment, including multicast completion latency, user-level CPU utilization, multicast tree-creation latency, and link concentration and concurrency. The maximum user-level host CPU utilization of the root node is measured using the Linux built-in `sar` utility. Link concentration and concurrency of each algorithm are calculated as described in the Chapter 2 of this dissertation, for each group size based on the communication pattern observed throughout the experiments.

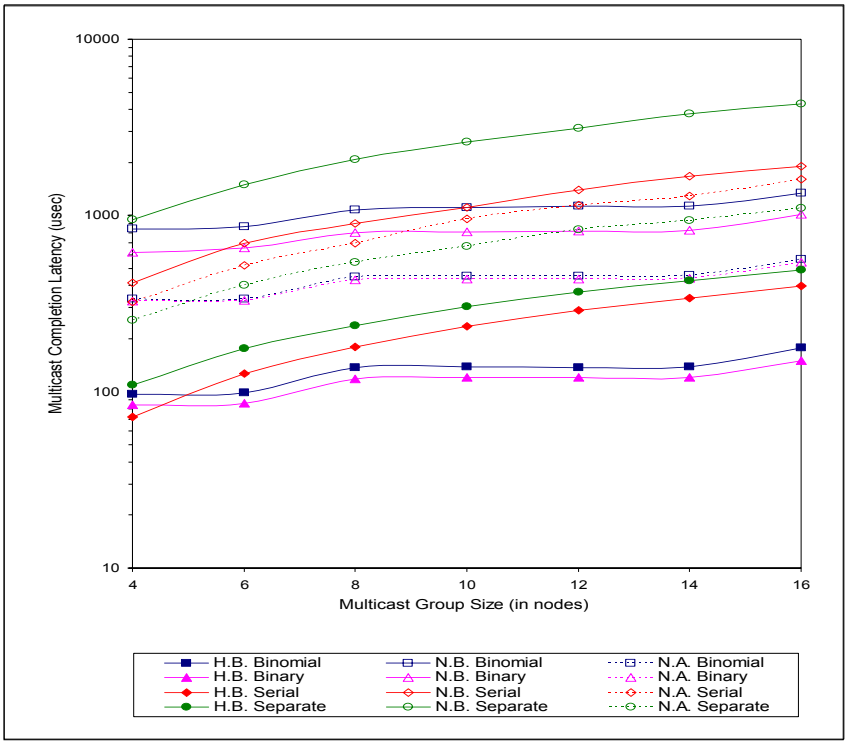
Multicast Completion Latency

As previously stated, completion latency is an important metric for assessing the quality and efficiency of multicast algorithms and communication schemes. Two different sets of experiments for multicast completion latency are performed in this case study, one for a small message size of 2B, and the other for a large message size of 64KB. Figure 3-5A shows the multicast completion latency versus group size for small messages. Figure 3-5B shows multicast completion latency versus group size for large messages. Figure 3-5A is presented with a logarithmic scale for clarity.

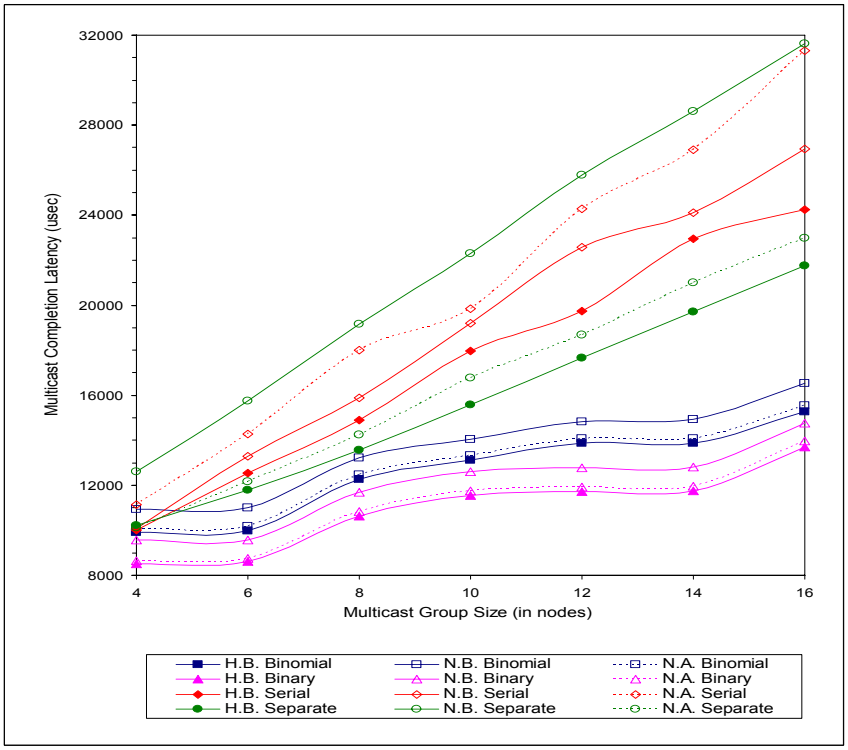
The small-message results presented in Figure 3-5A show that among the host-based algorithms, binary tree performs the best for any group size for small messages. The host-based binomial tree algorithm is second best in terms of latency. The difference in the performance level of these two algorithms is because of the higher computational load of binomial tree compared to the binary tree algorithm. The step-like shape of the binomial and binary tree algorithms is due the $\lfloor log_2 G \rfloor$-1 intermediate nodes on the deepest path traveled by the multicast message. The host-based serial forwarding

algorithm shows a linear increase in completion latency by increasing group size. The host–based separate addressing algorithm performs worst among all host-based algorithms and also shows a linear increase in latency with increasing group size. The NIC-based approach has the highest multicast completion latency for all algorithms and all group sizes as expected. The reason for this poor performance level is because the slower NIC processor (as compared to the host processor) handles all of the computation and communication tasks. In particular, this communication scheme impedes the performance of separate addressing more than the other algorithms. The NIC-assisted solution provides slightly smaller latencies compared to the NIC-based solutions for all algorithms. The increases in efficiency of communication are caused by the fact that the faster host processor performs all the communication-related computational tasks. This communication scheme improves the latency characteristics of the separate addressing algorithm more than the serial forwarding algorithm.

The large-message results presented in Figure 3-5B show that the binary tree performs the best compared to all other host-based communication algorithms as evident in the small-message host-based results. Binomial tree has slightly higher latency values for all group sizes because of its higher algorithmic computational load as compared to the binary tree algorithm. Serial forwarding, as described previously, has more latency variations compared to the small-message case. The NIC-based communication schemes still significantly impede the performance of all algorithms compared to the host-based approach as previously seen in the small-message case. However, contrary to the small-message case, the NIC-assisted approach lowers the large-message latencies. Although overheads are significantly high and they overlap with the expensive host



a



b

Figure 3-5. Multicast completion latencies. Host-based, NIC-based, and NIC-assisted communication schemes are denoted by H.B., N.B., and N.A, respectively. A) Small messages vs. group size. B) Large messages vs. group size.

CPU–NIC LANai interaction events. This overlapping hides the latencies of host CPU–NIC LANai interactions and results in the dramatic reduction of completion latencies of the NIC-assisted approach for large messages. Binomial and binary tree algorithms benefit most from the NIC-assisted communication approach for large messages.

The large-message results presented in Figure 3-5B show that the binary tree performs the best compared to all other host-based communication algorithms as evident in the small-message host-based results. Binomial tree has slightly higher latency values for all group sizes because of its higher algorithmic computational load as compared to the binary tree algorithm. Serial forwarding, as described previously, has more latency variations compared to the small-message case. The NIC-based communication schemes still significantly impede the performance of all algorithms compared to the host-based approach as previously seen in the small-message case. However, contrary to the small-message case, the NIC-assisted approach lowers the large-message latencies closer to the host-based approach. In the NIC-assisted approach the sender and receiver overheads are significantly high and they overlap with the expensive host CPU–NIC LANai interaction events. This overlapping hides the latencies of host CPU–NIC LANai interactions and results in the dramatic reduction of completion latencies of the NIC-assisted approach for large messages. Binomial and binary tree algorithms benefit most from the NIC-assisted communication approach for large messages.

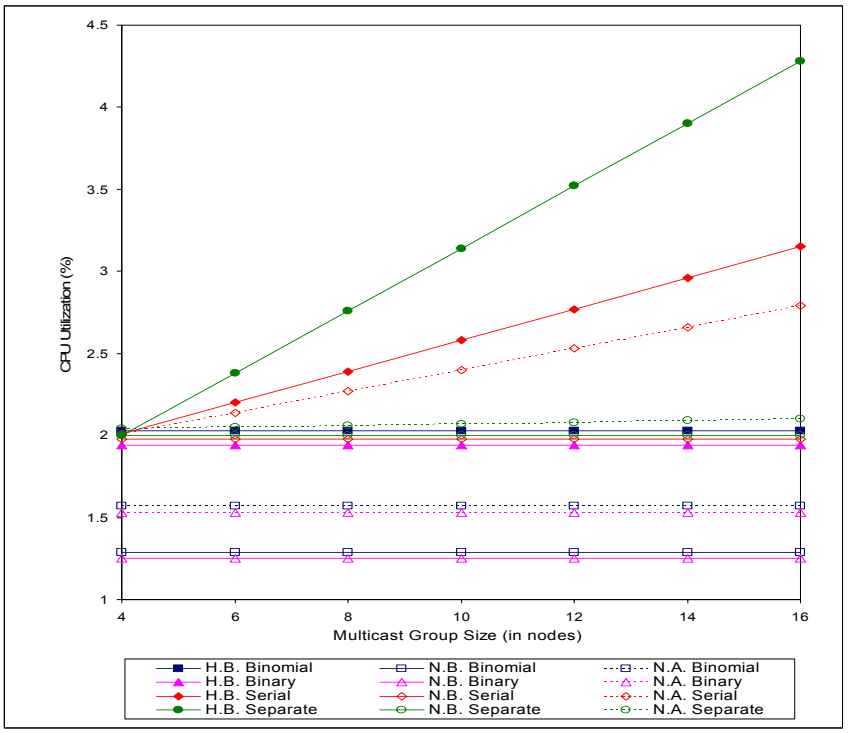
User-level CPU Utilization

The CPU utilization is measured at the host processor for all experiments. For both small messages and large messages, host-based multicasting produces the highest CPU utilization level for each group size because the host-processor handles all the multicast communication and computation tasks. By contrast, NIC-based communication provides

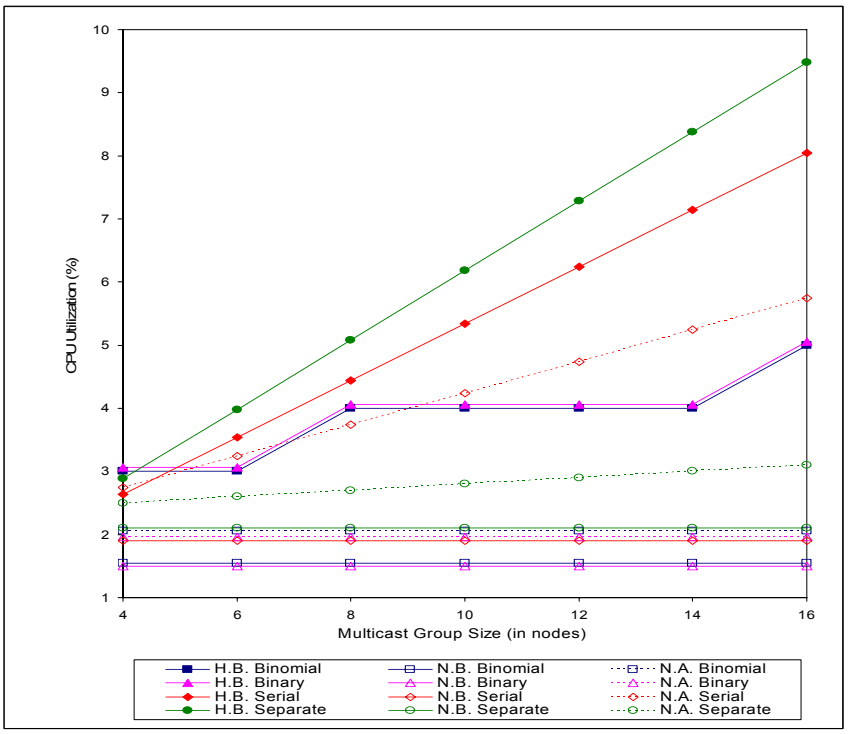
a constant level of CPU utilization that is lower than host-based and NIC-assisted schemes for all algorithms. In NIC-based approach, the host CPU is only responsible for setting up the multicast communication environment and the rest of the tasks are carried out by the NIC processor, decreasing the workload of the host processor. NIC-assisted multicast reduces host CPU utilization as compared to the host-based scheme. Figure 3-6A shows the small-message CPU utilizations for host-based, NIC-based, and NIC-assisted communication schemes. Figure 3-6B shows the large-message CPU utilizations for host-based, NIC-based, and NIC-assisted communication schemes. The results show that all of the algorithms exhibit low CPU utilizations for NIC-based and NIC-assisted schemes. This fact proves, in terms of the host CPU utilization, that the NIC-assisted scheme is the preferable one among the three for all networking scenarios. Overall, low CPU utilization and low multicast completion latency characteristics of the NIC-assisted scheme, makes it a good choice for multicast communication.

Multicast Tree Creation Latency

Multicast tree creation is an all-computational task, and tree creation latencies are independent of the multicast message size but dependent on the group size. Figure 3-7 shows the tree creation latencies for all combinations of communication schemes and multicast algorithms versus all group sizes. Host-based multicast tree creation provides the lowest latencies for all algorithms as can be seen from Figure 3-7. The NIC-based tree creation is roughly an order of magnitude slower than host-based tree creation, reflecting the performance gap between those two processors. The NIC-assisted tree creation has latencies identical to the host-based scheme because the host-processor handles tree-creation tasks in this communication scheme.



a



b

Figure 3-6. User-level CPU utilizations. Host-based, NIC-based, and NIC-assisted communication schemes are denoted by H.B., N.B., and N.A, respectively. A) Small messages vs. group size. B) Large messages vs. group size.

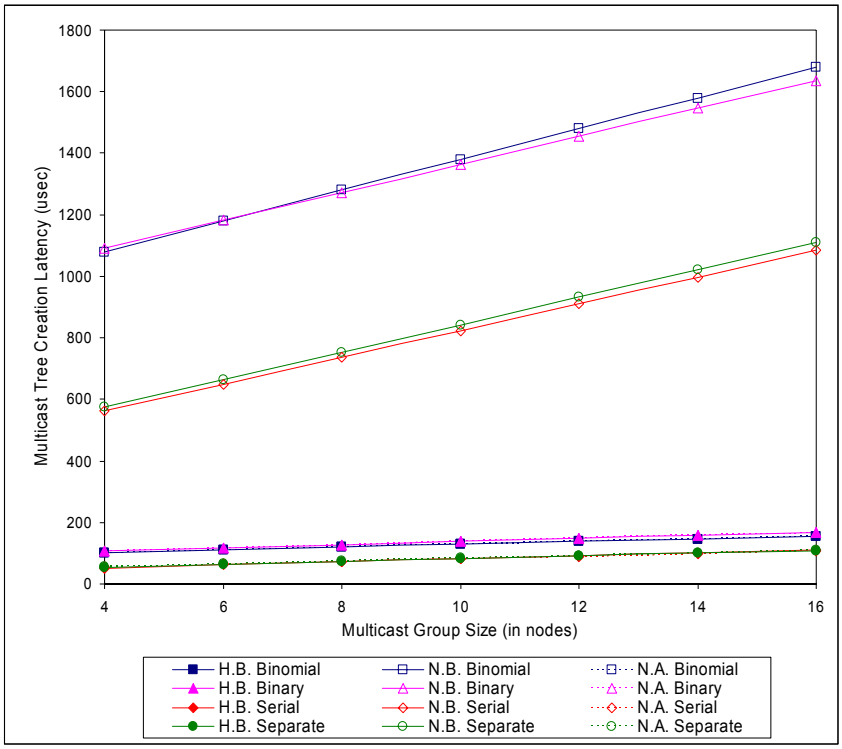


Figure 3-7. Multicast tree creation latencies. Host-based, NIC-based, and NIC-assisted communication schemes are denoted by H.B., N.B., and N.A, respectively.

Link Concentration and Link Concurrency

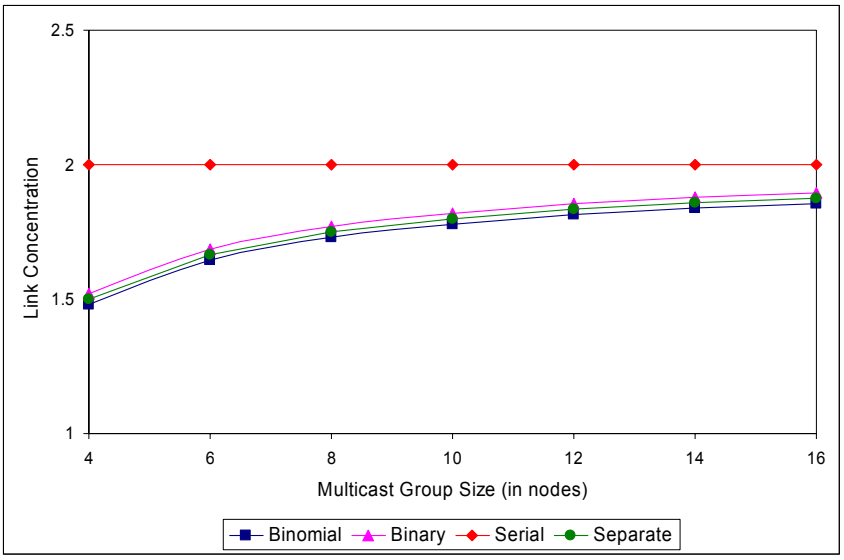
Link concentration measures the degree to which communication is concentrated on individual links. This metric, combined with link concurrency, can be used for assessing the effectiveness of the network link usage for a given communication structure. As all communication schemes access the network in the same manner, and the only difference between these schemes is their host-NIC access patterns. Thus, the network link usage is independent of the deployed communication scheme. Figure 3-8A shows the link concentration for all algorithms versus multicast group sizes. Figure 3-8B shows the link concurrency for all algorithms versus multicast group sizes. From Figure 3-8A it can be seen that the binomial and binary tree and the separate addressing algorithms have the lowest link concentrations and asymptotically bounded link

concentration of 2. Although the link access pattern of binary and binomial tree algorithms are different than the separate addressing algorithm, the number link visits and the used links are the same for both three of them. For the testbed with a single switch used in this case study, the link concentration can never exceed the asymptotic bound of 2, as it is number of maximum links that has be crossed for any point-to-point communication. Serial forwarding, which uses a different pair of hosts and links in every step of the multicast communication, has a constant and bounded link concentration of 2 for every group size.

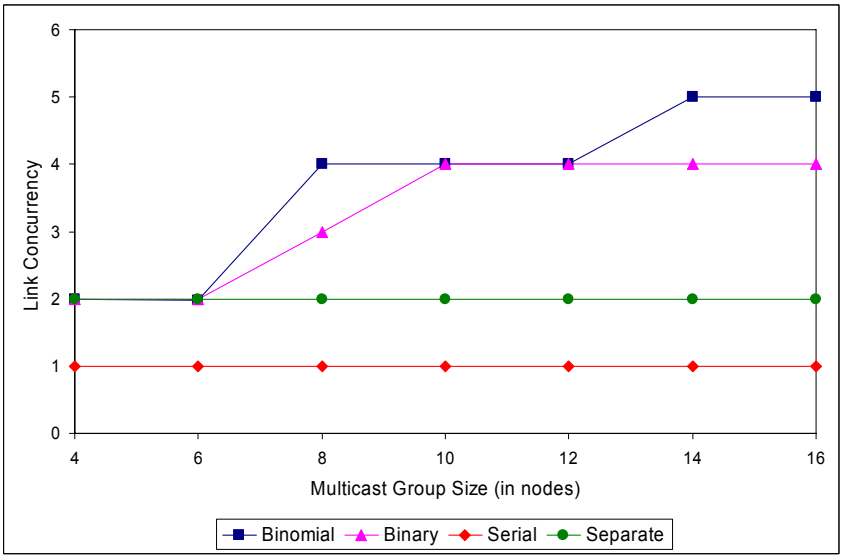
Link concurrency, given in Figure 3-8B, shows that binomial tree has the best link concurrency of all algorithms for most group sizes. Binary tree exhibits similar concurrency to binomial tree. Combined with the link concentration results presented in Figure 3-8A, the binomial tree algorithm appears to be the best. The difference between these binary and binomial tree algorithms is caused by their fan-out numbers. Serial forwarding has the lowest concurrency while the separate addressing algorithm is slightly better. Both show constant link concurrency.

Multicast Latency Modeling

Myrinet experiments for evaluating the host-based, NIC-based, and NIC-assisted communication schemes have been performed over a 16-node star network. These experiments provide useful information for understanding the basic aspects of these schemes and algorithms, but are insufficient for drawing further detailed analyses for systems of arbitrary sizes beyond the testbed capabilities. Latency modeling complements the experimental study by providing more detailed insight and establishes a better understanding of the problem.



a



b

Figure 3-8. Communication balance vs. group size. A) Link concentration. B) Link concurrency.

Latency modeling for the Myrinet network is based on Eq. 2-5 and follows the same outlines approach in Chapter 2. The basic idea in latency modeling presented in this chapter is to express the communication events with a few parameters without oversimplifying the process.

Although the outlined method is same, there are some differences between the Myrinet latency model and the SCI latency model. First of all, $t_{network}$ is defined differently because Myrinet is an indirect network unlike the direct SCI network. In an indirect network, nodes establish point-to-point communications through switches as there are no forwarding nodes present on a given communication path. Eq. 3-1 reflects the new $t_{network}$ parameter for an indirect network.

$$t_{network} = h_p \times L_p + h_s \times L_s + h_i \times L_i \quad (3\text{-}1)$$

Here, h_p, h_s, and h_i represent the total number of hops, total number of switching nodes, and total number of intermediate nodes, respectively. Similarly L_p and L_s denote the propagation delay per hop and switching delay through a switch, respectively. The L_i parameter denotes the intermediate delay through a node, which is the sum of the receiving overhead of the message, processing time, and the sending overhead of the message at the host and NIC layers for a given intermediate node.

The second difference between SCI and Myrinet latency models is that in Myrinet the host and NIC processors coordinate to perform the communication-related tasks. Therefore, the Myrinet model has to account for this interaction and work sharing between the host and NIC processors. Also, this coordination and interaction is different for each communication scheme for a given Myrinet network. Eqs. 3-2 through 3-10 present each scheme's sender and receiver overheads; and their intermediate node delays in detail.

In a software-based multicast communication it is likely that a node will issue more than one message. To formulate the sender overhead for the host-based communication

scheme, it is assumed that the host processor is issuing the M^{th} individual multicast message. The overhead for issuing M^{th} message is expressed as:

$$o_{sender} = M \times t_{Host\_send} \tag{3-2}$$

However, for NIC-based communication the send process is completely performed by the NIC processor. The sender overhead for the M^{th} consecutive multicast message is expressed as in Eq. 3-3.

$$o_{sender} = M \times t_{NIC\_send} \tag{3-3}$$

In NIC-assisted multicasting, a different level of work sharing exists between the host and NIC processors, compared to the NIC-based scheme. In NIC-assisted scheme, the host processor is only responsible for the multicast related computational tasks. In this scheme the NIC processor is only responsible for the communication-only events, such as the send operation. From the NIC point of view, issuing NIC-assisted multicast messages is the same with the NIC-based scheme. Therefore, the sender overhead for the NIC-assisted multicast scheme for a given M^{th} consecutive multicast message is same with the NIC-based case and is expressed as:

$$o_{sender} = M \times t_{NIC\_send} \tag{3-4}$$

The last node on the multicast message path incurs receiver overhead. For the host-based scheme, the GM software layer automatically drains the message from the network and places in the appropriate user-level memory space. The receiver overhead for the host-based communication is expressed as:

$$o_{receiver} = t_{Host\_recv} \tag{3-5}$$

For the NIC-based communication scheme, the receive operation involves both the host and the NIC processors. The NIC processor receives and removes the multicast

message from the network, writes it to the appropriate user-level memory address and then notifies the host processor. Upon this notification, the host processor completes the receive operation. As explained before, the RDMA operation consists of the memory transfer of the message to the user-level memory, and the NIC processor's notification of the host processor upon transfer completion. The receiver overhead for NIC-based multicasting is:

$$o_{receiver} = t_{NIC\_recv} + RDMA + t_{Host\_recv} \tag{3-6}$$

From the host CPU point of view the reception of a message is identical for NIC-based and NIC-assisted communication schemes. For both of these schemes, the NIC-processor drains the message from the network and performs an RDMA operation to the user-level memory space and notifies the host CPU. Therefore, the receiver overhead of the NIC-assisted communication is identical to the NIC-based receiver overhead, and is expressed as follows:

$$o_{receiver} = t_{NIC\_recv} + RDMA + t_{Host\_recv} \tag{3-7}$$

The intermediate nodes act as relays in multicast communication. These nodes receive messages from the network, calculate the next set of destination nodes and relay the messages to them. Therefore, the delay for each intermediate node includes sum of the receiver overhead, processing delay, and the sender overhead. For the host-based communication scheme, the intermediate node delay, L_i, for the M^{th} consecutive message sent is:

$$L_i = t_{Host\_recv} + t_{Host\_process} + M \times t_{Host\_send} \tag{3-8}$$

For the NIC-based communication scheme, all intermediate node tasks are performed by the NIC processor. For the M^{th} consecutive message sent, the NIC-based intermediate node delay is expressed as follows:

$$L_i = t_{NIC\_recv} + t_{NIC\_process} + M \times t_{NIC\_send} \quad (3\text{-}9)$$

The intermediate node delay for the NIC-assisted communication scheme starts when the NIC processor receives a message and performs an RDMA operation to the host processor. Consequently, the host processor processes the incoming multicast message and performs an SDMA operation back to the NIC processor to perform the actual multicast send operation. For the M^{th} consecutive send, the NIC-assisted intermediate node delay is expressed as:

$$L_i = t_{NIC\_recv} + RDMA + t_{Host\_process} + SDMA + M \times t_{NIC\_send} \quad (3\text{-}10)$$

As explained before, for achieving higher network concurrency, all the algorithms evaluated in this chapter are designed such that the root node always serves the deepest path first. Therefore, all equations given above can be simplified as taking M equal to 1. This simplification ensures that the root serves the deepest path first when modeling the total multicast latency.

The following paragraph explains how the components of $t_{network}$ are acquired. The latency model parameter, L_s, is obtained by measuring the difference between the two latency measurements over two nodes that are first connected through the Myrinet switch and than directly without the switch. The L_s parameter is measured as 500ns. For the experiment setup h_s is set to 1. Assuming, 7ns per meter of propagation delay, L_p is calculated as 21ns as all the Myrinet interconnect cables used were 3m long. In our testbed, 2 links are crossed for each pair of nodes that establish a point-to-point

communication. For the single-switch, 16-node system used in our experiments, crossing 2 links per each connection yields h_p as 2 for the separate addressing algorithm. All other algorithms use a tree-based communication structure which depends on relaying the message through intermediate nodes. Therefore, binary tree, binomial tree, and serial forwarding have an h_p equal to $2 \times h_i$. The number of intermediate nodes, h_i, is calculated as $(G - 2)$ for serial forwarding and $\lfloor \log_2 G \rfloor - 1$ for the binary and binomial tree algorithms.

Following the same approach defined by Bunitas *et al.* [39], the sender and receiver overhead estimates of the host and NIC processors are obtained individually. As the clock frequency of the host processor is known, the `RDTSC` assembly instruction is used to read the internal 64-bit cycle counter to get the exact time spent on the host processor. Averaging the host-based multicast communication for each previously defined experiment for each message and group size, an accurate estimate of the t_{Host_send} is obtained. For simplicity, t_{Host_recv} is approximated to equal t_{Host_send}.

The real-time clock register (RTC) on the LANai 7 chip is incremented at a regular interval automatically by default. The incremental interval of this register is set at initialization time of the NIC based on the actual PCI bus clock. All access to the NIC registers from the host side, are performed as PCI bus transactions. Therefore, to remove this bus delay, the 64-bit host processor cycle counter is read immediately after reading the RTC. The host-processor is put to sleep for 1 second. The same process is then repeated. Comparing the elapsed time in these repeated readings allowed accurately assessing and eliminating the PCI bus transaction delays from the RTC reading operations. The t_{NIC_send} and the t_{NIC_recv} values are obtained for the NIC-based and the NIC-assisted schemes for all group sizes and small and large messages. The NIC-based

and NIC-assisted *SDMA* and the *RDMA* tasks are also measured using the outlined method, from the host and NIC respectively.

Table 3-2. Measured latency model parameters.

	t_{Host_send} (μs)	t_{Host_recv} (μs)	t_{NIC_send} (μs)	t_{NIC_recv} (μs)	*SDMA* (μs)	*RDMA* (μs)
Host-Based	$9.864+0.0315\times m$	$9.864+0.0315\times m$	N/A	N/A	N/A	N/A
NIC-Based	N/A	N/A	$0.697+8.3325\times m$	$0.697+8.3325\times m$	26	28
NIC-Assisted	N/A	N/A	$0.697+8.3325\times m$	$0.697+8.3325\times m$	26	28

Table 3-3. Calculated $t_{process}$ and L_i values.

		$t_{process}$ (μs)	L_i (μs)
Host-Based	Binomial	19.2	39.05
	Binary	12.2	32.05
	Serial	7.2	27.05
	Separate	22.2	N/A
NIC-Based	Binomial	145.2	250.25
	Binary	92.2	178.25
	Serial	79.2	121.08
	Separate	233.1	N/A
NIC-Assisted	Binomial	19.2	113.9
	Binary	12.2	107.9
	Serial	7.2	101.7
	Separate	22.2	N/A

Table 3-2 summarizes the acquired values of t_{Host_send}, t_{Host_recv}, t_{NIC_send}, t_{NIC_recv}, *SDMA*, and *RDMA* for the host-based, NIC-based, and NIC-assisted communication schemes. Table 3-3 shows the $t_{process}$ (i.e., $t_{Host\_process}$ or $t_{NIC\_process}$ depending on the communication scheme) and L_i values for each possible communication scheme and algorithm combination. These values are obtained by substituting the values given in Table 3-2 into Eqs. 3-2 through 3-10. The $t_{process}$ and L_i values are specific to the system used. However, these values are independent of multicast message size. The following

subsections provide the details of the small-message latency model for each scheme and algorithm.

The Host-based Latency Model

The host-based latency models are obtained using Eqs. 3-2, 3-5, and 3-9. The binary and binomial trees have the same number of intermediate nodes for each case, therefore these two algorithms is formulated in Eq. 3-11.

$$t_{multicast} = t_{Host\_send} + \left(\lfloor \log_2 G \rfloor - 1\right) \times \left(t_{Host\_recv} + t_{Host\_process} + t_{Host\_send}\right) + t_{Host\_recv} \quad (3\text{-}11)$$

The model for the serial forwarding is given as:

$$t_{multicast} = t_{Host\_send} + \left(G - 2\right) \times \left(t_{Host\_recv} + t_{Host\_process} + t_{Host\_send}\right) + t_{Host\_recv} \quad (3\text{-}12)$$

Separate addressing model is formulated as:

$$t_{multicast} = \left(G - 1\right) \times \left(t_{Host\_process} + t_{Host\_send}\right) + t_{Host\_recv} \quad (3\text{-}13)$$

Figure 3-9 shows results from the host-based, small-message model for all algorithms versus multicast group size along with the actual measurements. The model is accurate for the binomial, binary tree and separate addressing algorithms, and the average error is ~1%, ~2%, and ~1%, respectively. The host-based serial forwarding algorithm has a relatively higher modeling error, ~3%, because of its inherent latency variations as explained in the previous chapter.

The NIC-based Latency Model

NIC-based latency models are obtained by substituting the values presented in the previous section into Eqs. 3-3, 3-6, and 3-9. The small-message model for the binary and binomial trees is formulated as given in Eq. 3-14.

$$\begin{aligned} t_{multicast} = \; & SDMA + t_{NIC\_process} + t_{NIC\_send} + \left(\lfloor \log_2 G \rfloor - 1\right) \\ & \times \left(t_{NIC\_recv} + t_{NIC\_process} + t_{NIC\_send}\right) + t_{NIC\_recv} + RDMA \end{aligned} \quad (3\text{-}14)$$

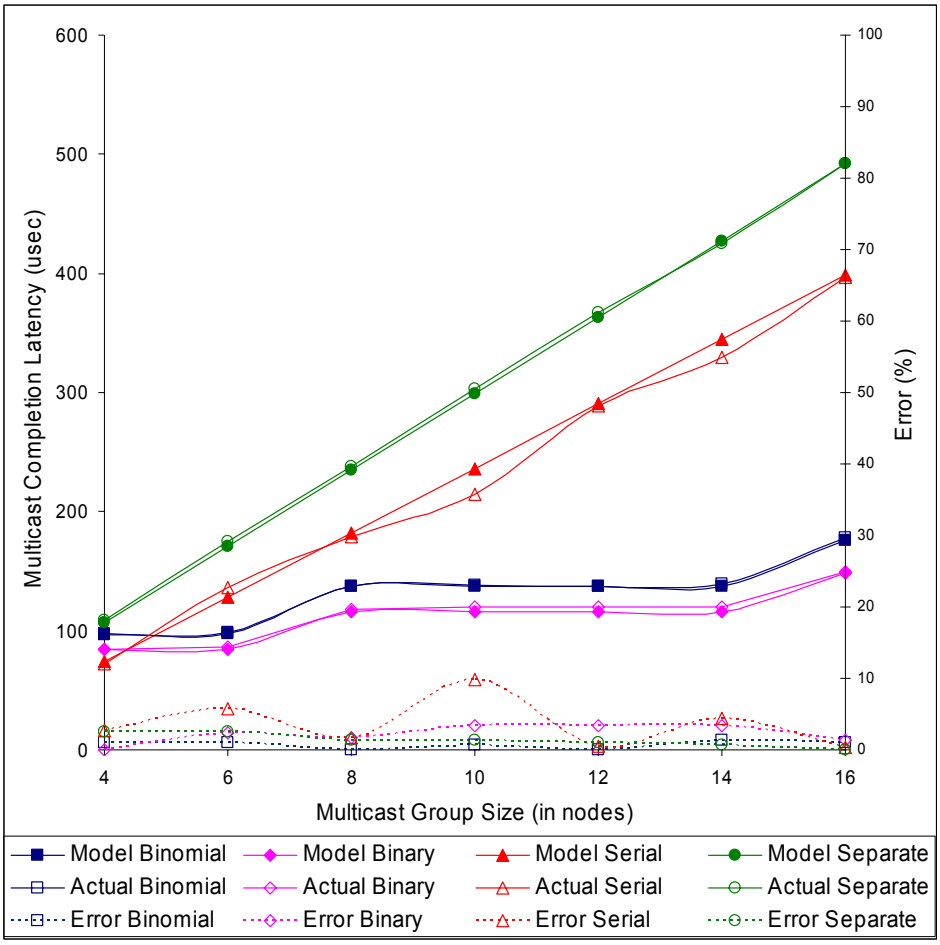


Figure 3-9. Simplified model vs. actual measurements for the host-based communication scheme.

The model for serial forwarding is as follows:

$$t_{multicast} = SDMA + t_{NIC\_process} + t_{NIC\_send} + (G-2)\times \left(t_{NIC\_recv} + t_{NIC\_process} + t_{NIC\_send}\right) + t_{NIC\_recv} + RDMA \quad (3\text{-}15)$$

The separate addressing model is formulated as given as:

$$t_{multicast} = SDMA + (G-1)\times\left(t_{NIC\_process} + t_{NIC\_send}\right) + t_{NIC\_recv} + RDMA \quad (3\text{-}16)$$

Figure 3-10 shows the NIC-based small-message model and the actual measurements versus multicast group size for all algorithms. Similar to the host-based case, the model is accurate for the binomial, binary tree and separate addressing algorithms, and the average error is ~2%, ~2%, and ~1%, respectively. Serial forwarding has a relatively

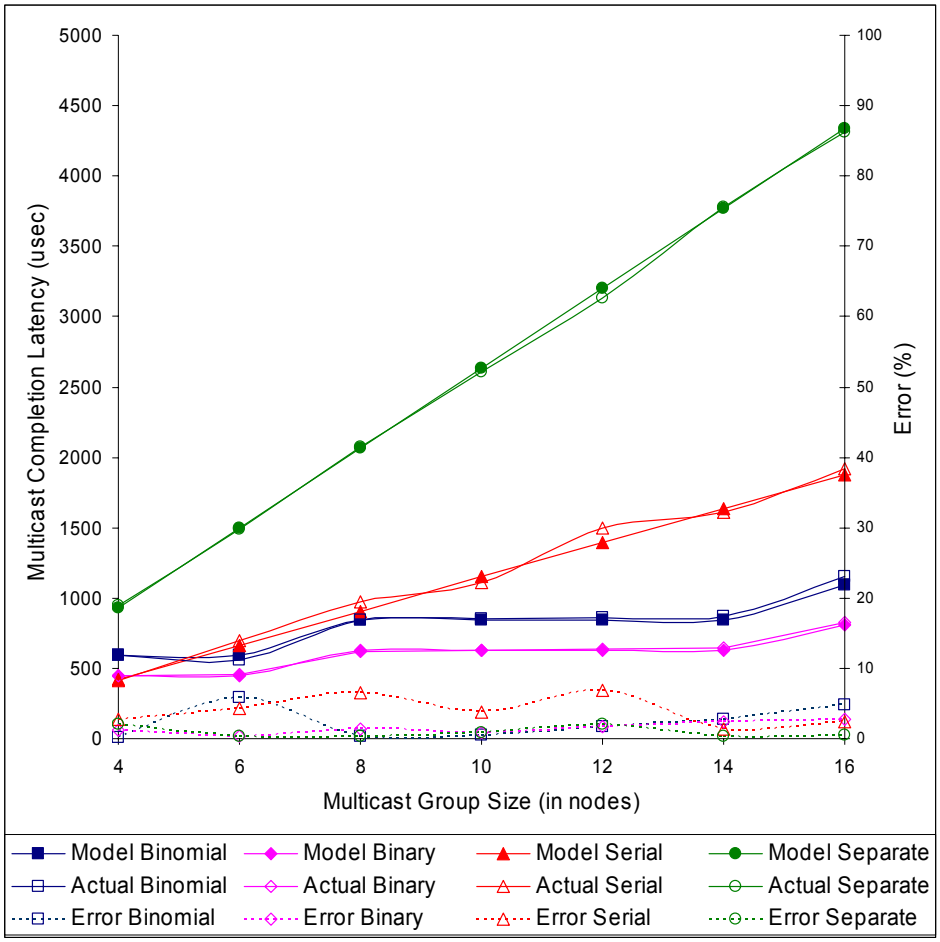


Figure 3-10. Simplified model vs. actual measurements for the NIC-based communication scheme.

higher modeling error, ~4%, because of its unavoidable latency variations.

The NIC-assisted Latency Model

As explained previously, the difference between the NIC-based and the NIC-assisted model is where communication-related computational processing is handled. Using Eqs. 3-4, 3-7, and 3-10, the small-message model for binary and binomial models is formulated as:

$$t_{multicast} = SDMA + t_{NIC\_send} + \left(\lfloor \log_2 G \rfloor - 1\right) \times \left(t_{NIC\_recv} + SDMA + t_{Host\_process} + RDMA + t_{NIC\_send}\right) + t_{NIC\_recv} + RDMA \quad (3\text{-}17)$$

The separate addressing model is:

$$t_{multicast} = SDMA + t_{NIC\_send} + (G-2) \times \left(t_{NIC\_recv} + SDMA + t_{Host\_process} + RDMA + t_{NIC\_send}\right) + t_{NIC\_recv} + RDMA \quad (3\text{-}18)$$

The serial addressing model is:

$$t_{multicast} = (G-1) \times \left(t_{Host\_process} + SDMA + t_{NIC\_send}\right) + t_{NIC\_recv} + RDMA \quad (3\text{-}19)$$

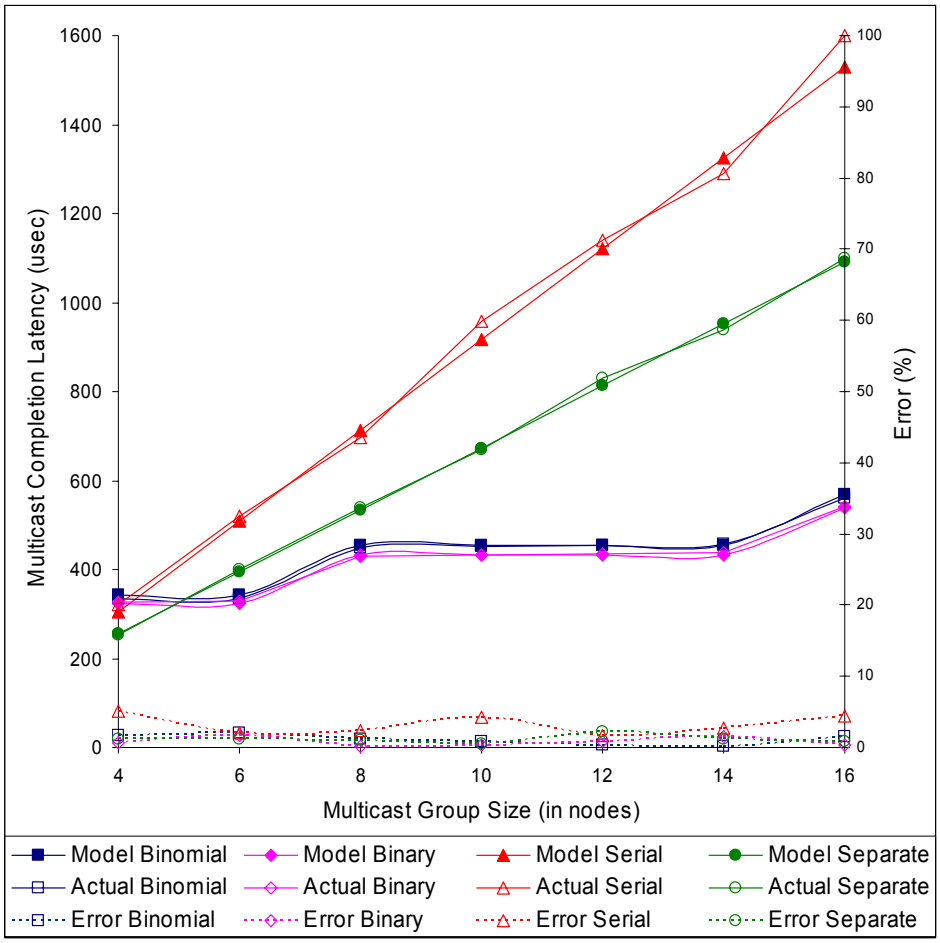


Figure 3-11. Simplified model vs. actual measurements for the NIC-assisted communication scheme.

Figure 3-11 shows the results for the NIC-assisted small-message model and the actual measurements versus multicast group size for all algorithms. The figure shows that the model is accurate for the binomial and binary tree and separate addressing algorithms, and the average error is ~1% in all cases. Like the host-based and the NIC-based cases, serial forwarding has a relatively higher modeling error, ~3%.

Analytical Projections

To evaluate and compare the short-message performance of the communication schemes and the multicast algorithms for larger systems, the simplified models are used to investigate the performance characteristics of the multicast completion latency for an indirect Myrinet star network with 64 and 256 nodes. For the first case, a 64-node network with a single 64-port Clos switch is considered. A 256-node network with a single 256-port Clos switch is considered for the second case. Currently the largest commercially available Myrinet switch is for 128 nodes, but the technology is progressing towards higher capacity switches. Higher capacity switches are preferable because they are easier to maintain and cheaper to build. These switches also provide higher bisection bandwidths with less number of cable connections than MINs. Moreover, higher capacity switches provide full-bisection bandwidth with fewer connections. Vendor provided results show that the commercially available 128-node Myrinet Clos switch achieves similar switching latencies to the 16-port version. Therefore, projections for a 256-node system with a single Clos switch are considered to be realistic.

Figures 3-12 and 3-13 show the projection results for the two cases under study. Strictly in terms of multicast completion latency, the host-based binary and binomial trees provide the lowest completion latencies. The difference between these two is caused by the processing load and fan-out numbers. A more efficient binomial tree design with a lower processing load and a higher fan-out number, compared to the one used in this dissertation, may outperform the binary tree algorithm. Also it is observed that the host-based serial forwarding and separate addressing algorithms have a linear

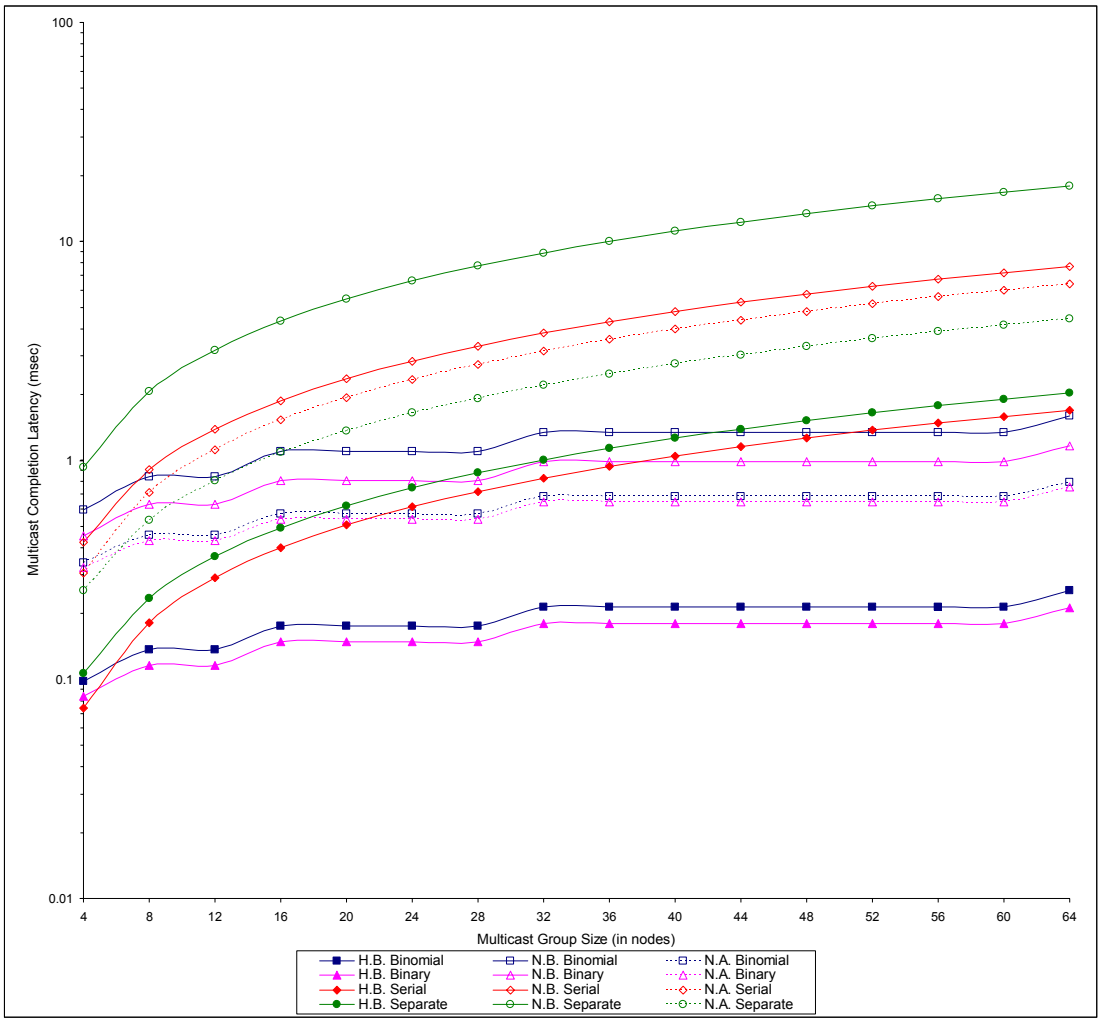


Figure 3-12. Projected small-message completion latency for 64 nodes.

increase in completion time with increasing group sizes. Unfortunately, NIC-based approaches are a poor solution for any of the algorithms in any given network scenario as compared to the host-based approach because of their extensive completion latencies. However, as can be seen from the figures, the NIC-assisted algorithms lower the completion latencies compared to the NIC-based solutions. This approach provides significant latency reduction in the separate addressing, binomial and binary tree algorithms, where the largest gain in terms of completion latency is in the separate addressing algorithm. Unfortunately, serial forwarding is less affected by the

NIC-assisted algorithm because of the expensive SDMA-RDMA operations incurred by each intermediate node.

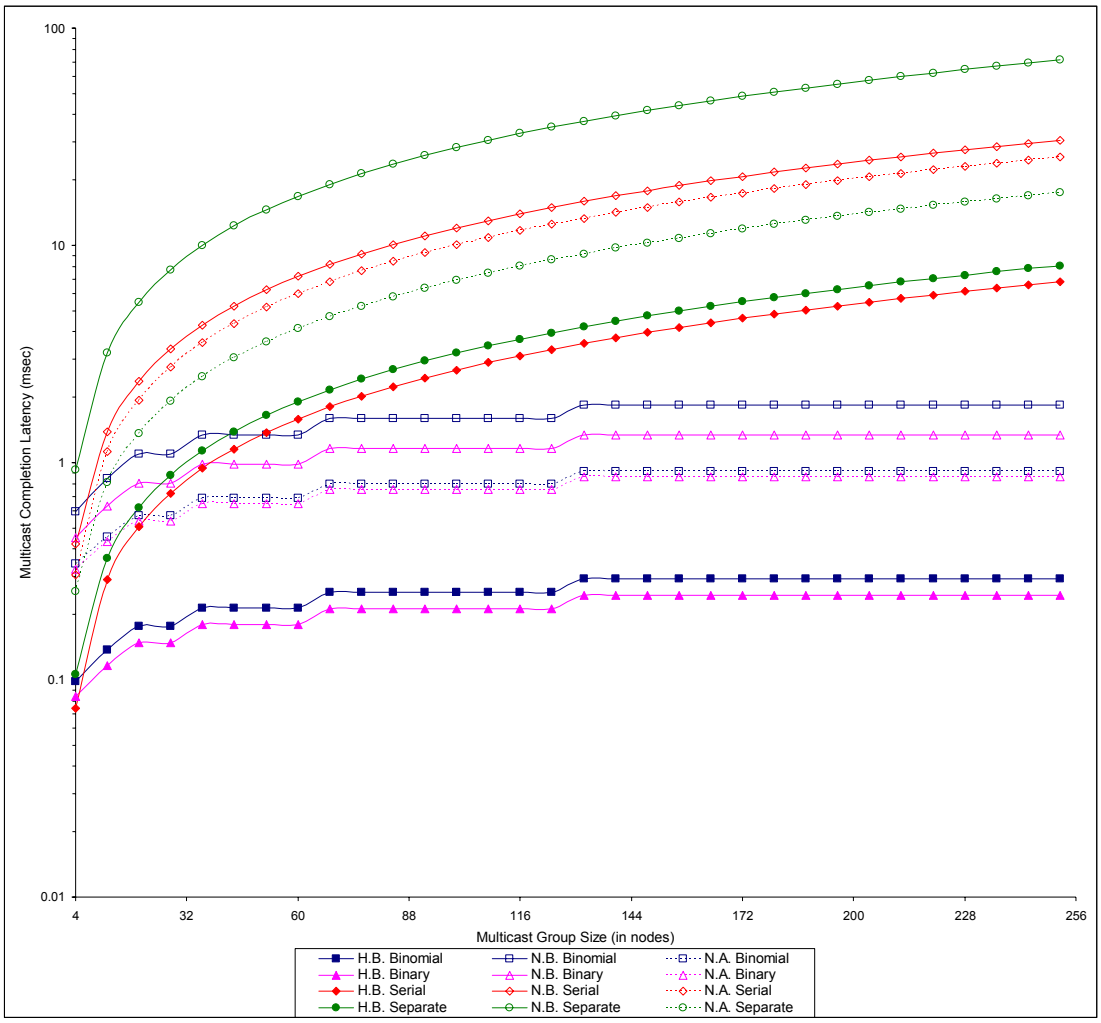


Figure 3-13. Projected small-message completion latency for 256 nodes.

Other than multicast completion latency, host CPU utilization is also an important parameter to consider when choosing a multicast communication scheme and algorithm. For larger systems a sacrifice in completion latency can be made to achieve a lower CPU utilization. For such systems, NIC-assisted schemes may be well suited, because they provide lower CPU utilization and more communication–computation overlap.

Summary

This chapter of the dissertation investigated the multicast problem for high-speed indirect networks with NIC-based processors. Chapter 3 introduced a multicast performance analysis and modeling for this type of interconnects. These interconnects are widely used in the parallel computing community because of their reprogrammable flexibility and work offloading from host CPU characteristics. This phase of the dissertation analyzed various degrees of host and NIC processor work sharing. Host-based, NIC-based, and NIC-assisted multicast communication schemes for work sharing are analyzed. Binomial and binary tree, serial forwarding and separate addressing multicast algorithms are used for these analyses. Experimental evaluations are performed using various metrics, such as multicast completion latency, root-node CPU utilization, multicast tree creation latency, and link concentration and concurrency, to evaluate their key strengths and weaknesses. To further analyze the performance of aforementioned multicast communication schemes, small-message latency models of binomial and binary tree, serial forwarding and separate addressing multicast algorithms are developed and verified based on the experimental results. The models are observed to be accurate. Projections for larger systems are also presented and evaluated.

Experimental and latency modeling evaluations showed that for latency-sensitive applications that utilize small-messages which run over networks with NIC coprocessors, the host-based multicast communication scheme performs best. The disadvantage of host-based multicasting scheme is its high host-CPU utilization. The NIC-based solutions obtain the lowest and constant host-CPU utilizations for all cases at the cost of increased completion latencies. A compromise solution, the NIC-assisted multicasting provides lower CPU utilizations than host-based ones. Moreover, the NIC-assisted

multicasting also provides comparable CPU utilizations to the NIC-based algorithms. The NIC-assisted approach also provides comparable multicast completion latencies to host-based schemes for lower host CPU utilizations. Thus, NIC-assisted multicasting appears to be better choice for applications that demand a high level of computation-communication overlapping.

CHAPTER 4
MULTICAST PERFORMANCE COMPARISON OF SCALABLE COHERENT INTERFACE AND MYRINET

Chapter 2 and 3 are focused on individually evaluating the multicast performance of high-speed torus interconnects high-speed indirect networks with onboard adapter-based processors. Experimental case studies and small-message latency models and analytical projections for both types of networks are presented. Chapter 4 summarizes and provides head-to-head multicast performance comparisons between SCI and Myrinet. The comparison is based on the case study data presented in the previous two chapters. All SCI multicast algorithms, Myrinet multicast schemes and algorithms are included in the comparison except the Myrinet NIC-based communication scheme. Myrinet NIC-based was excluded because it was the worst performing one compared to Myrinet host-based and NIC-assisted schemes in terms of completion latency, and it does not appear to be a viable solution for the software-based multicast problem for Myrinet interconnects.

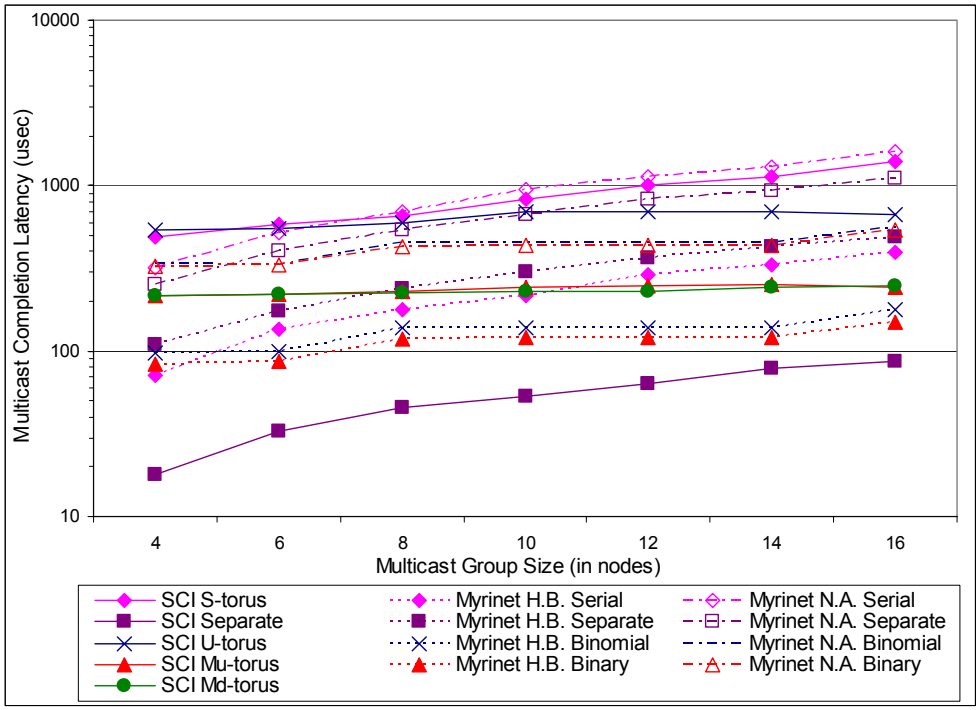
The unicast performance of SCI and Myrinet has been comparatively evaluated by Kurmann and Stricker [42] previously. They showed that both networks suffer performance degradation with non-contiguous data block transfers. Fischer *et al.* [43] also compared SCI and Myrinet. In their study, they concluded that, in terms of their performance analyses, Myrinet is a better choice compared to SCI since unlike Myrinet there is a magnitude of difference in SCI's remote read vs. write bandwidths.

Chapter 4 complements and extends previous work available in literature summarized above. Chapter 4 provides comparative experimental evaluations of torus-optimized multicast algorithms for SCI versus various degrees of multicast work-sharing between the host and the NIC processors optimized for Myrinet. Both for SCI and Myrinet networks, the multicast performance is evaluated using different metrics, such as multicast completion latency, root-node CPU utilization, link concentration and concurrency.

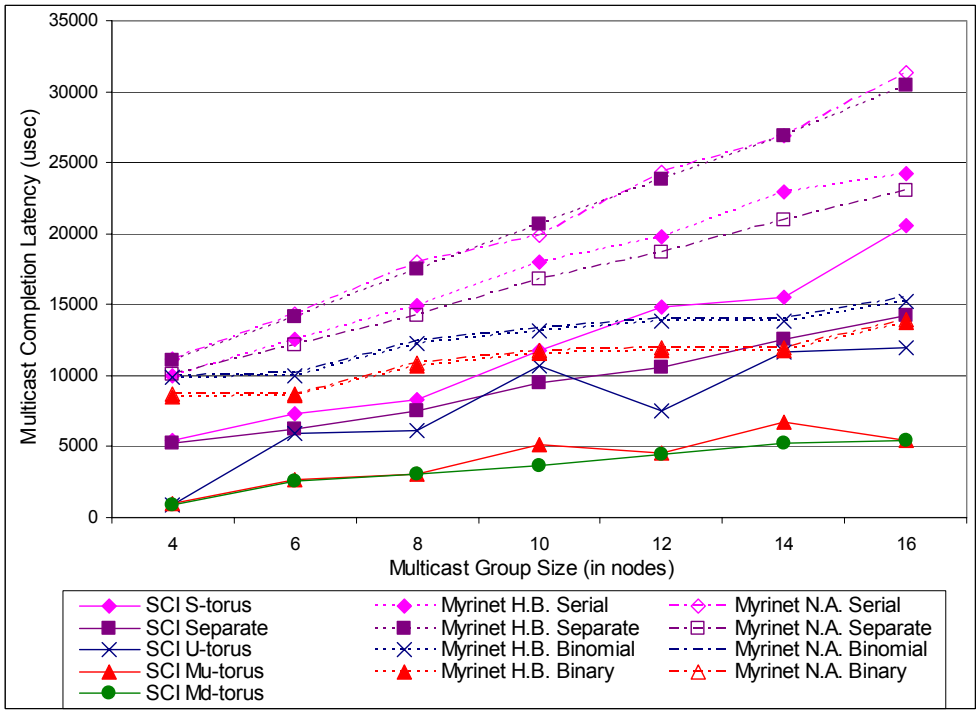
Multicast Completion Latency

Completion latency is an important metric for evaluating and comparing different multicast algorithms, as it reveals how suitable an algorithm is for a given network. Two different sets of experiments for multicast completion latency are used in this case study, one for a small message size of 2B, and another for a large message size of 64KB. Figure 4-1A shows the multicast completion latency versus group size for small messages, while Figure 4-1B shows the same for large messages for both networks combined with the various multicast algorithms. Figure 4-1A is presented with a logarithmic scale for clarity.

As explained previously, separate addressing is based on a simple iterative use of the unicast send operation. Therefore, for small messages the inherent unicast performance of the underlying network significantly dictates the overall performance of the multicast algorithm. This trait can be observed by comparing the small-message multicast completion latencies of SCI and Myrinet shown in Figure 4-1A. The SCI is inherently able to achieve almost an order of magnitude lower unicast latency compared to Myrinet. Simplicity and cost-effectiveness of the separate addressing algorithm for small messages, combined with SCI's unicast characteristics, result in the outcome where



a



b

Figure 4-1. Multicast completion latencies. Host-based, NIC-based, and NIC-assisted communication schemes are denoted by H.B., N.B., and N.A, respectively. A) Small messages vs. group size. B) Large messages vs. group size.

SCI separate addressing clearly performs the best compared to all other SCI and Myrinet multicast algorithms.

The NIC-assisted Myrinet separate addressing does not provide a comparable performance level to the host-based version because of the costly SDMA and RDMA operations. It is observed that the SDMA and RDMA operations impose a significant overhead for small-message communications. Moreover, all three multicast schemes show a linear increase with increasing group size.

The SCI S-torus is one of the worst performing algorithms next to the Myrinet NIC-assisted serial forwarding algorithm for small messages. Host-based Myrinet serial forwarding performs better compared to these two algorithms. The store-and-relay characteristics of serial forwarding algorithms result in no parallelism in message transfers and thus degradation of performance. Moreover, the expensive SDMA and RDMA operations cause the NIC-assisted serial forwarding algorithm to perform poorly compared the host-based version. As can be seen, all three multicast algorithms show a linear increase in multicast completion latency with respect to the increasing group size.

Unlike the separate addressing and serial forwarding algorithms of SCI and Myrinet, binomial and binary tree algorithms exhibit nearly constant completion latencies with increasing group sizes. Among these, Myrinet host-based binary and binomial tree algorithms perform best. Comparable algorithms, such as SCI U-torus, M_d-torus and M_u-torus algorithms show higher completion latencies. The difference is attributed to the low sender and receiver overheads of host-based Myrinet multicasting and the simplicity of the star network compared to the more complex torus structure of the SCI network. For a given star network, the average message transmission paths are shorter compared to

the same sized torus network. One other reason is increasing efficiency of the lightweight Myrinet GM message-passing library with increasing complexity of communication algorithms, compared to the shared-memory Scali API of the SCI interconnect. The effect of the expensive SDMA and RDMA operations can be clearly seen in the NIC-assisted Myrinet binary and binomial tree algorithms compared to their host-based counterparts.

For the large-message multicast latencies in Figure 4-1B, the SCI algorithms appear to perform best compared to their Myrinet counterparts. This outcome is judged to be primarily caused by the higher data rate of SCI compared to Myrinet (i.e., 5.3Gb/s vs. 1.28Gb/s). It should be noted that the Myrinet testbed available for these experiments is not representative of the latest generation of Myrinet equipment (which feature 2.0Gb/s data rates). However, we believe that our results would follow the same general trend for large messages on the newer hardware.

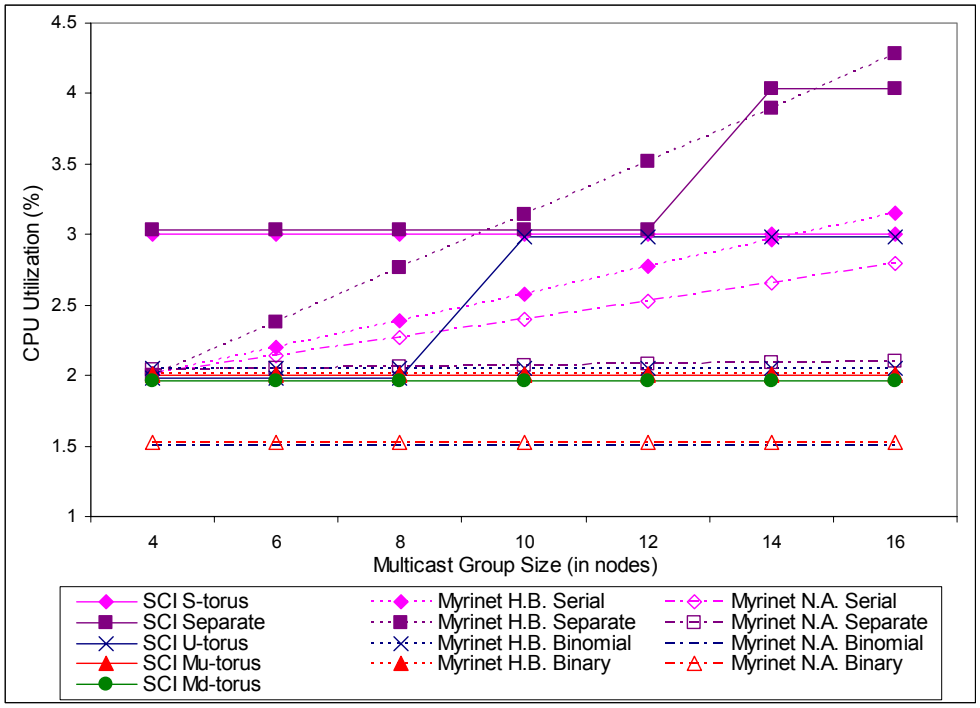
Among the SCI algorithms, M_d-torus is found to be the best performer. The M_d-torus and M_u-torus algorithms exhibit similar levels of performance. The difference between these two becomes more distinctive at certain data points, such as 10 and 14 nodes. For these group sizes, the partition length for M_u-torus does not provide perfectly balanced partitions, resulting in higher multicast completion latencies. For large messages, U-torus exhibits similar behavior to M_u-torus. The S-torus is the worst performer compared to all other SCI multicast algorithms. Moreover, S-torus, similar to other single-phase, path-based algorithms, has unavoidably large latency variations caused by the long multicast message paths [31].

Myrinet multicast algorithms seem to be no match for the SCI-based ones for large messages. Unlike the small-message case, Myrinet NIC-assisted binary and binomial tree algorithms provide nearly identical completion latencies to their host-based counterparts for large messages. The SDMA and RDMA overheads are negligible for large messages and because of this reason NIC-assisted multicast communication performance is enhanced significantly. Moreover, NIC-assisted communication improves the performance of the separate addressing algorithm most, compared to all other Myrinet multicast algorithms. This improvement is the result of the relative reduction of the overall effect of the SDMA and RDMA overheads on the multicast completion latencies. By contrast, NIC-assisted communication degrades the performance of the Myrinet serial forwarding algorithm, because the SDMA and the RDMA overheads are incurred at each relaying node.

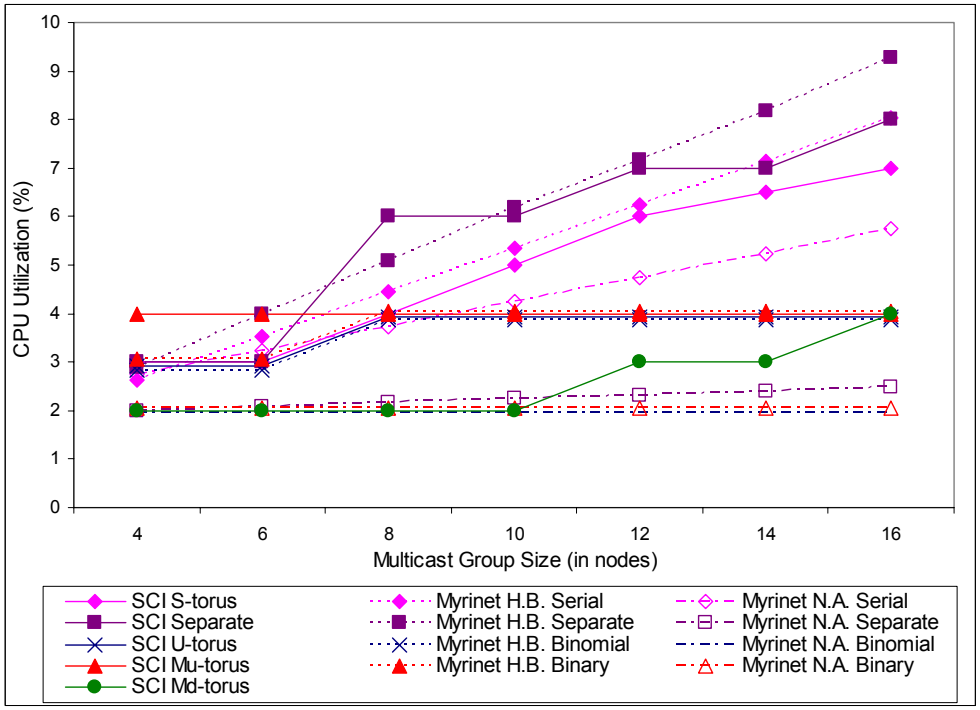
User-level CPU Utilization

Host processor load is another useful metric to assess the quality of a multicast protocol. Figures 4-2A and 4-2B present the maximum CPU utilization for the root node of each algorithm. As before, results are obtained for various group sizes and for both small and large message sizes. Root-node CPU utilization for small messages is presented with a logarithmic axis for clarity.

For small messages, SCI M_d-torus and M_u-torus exhibit constant 2% CPU utilization for all group sizes. Both algorithms use a tree-based scheme for multicast, which increases the concurrency of the message transfers and decreases the root-node workload significantly. Also, it is observed that SCI S-torus exhibits relatively higher utilization compared to these two but at the same time provides a constant CPU load independent of group size. As can be seen, SCI U-torus exhibits a step-like increase for



a



b

Figure 4-2. User-level CPU utilizations. Host-based, NIC-based, and NIC-assisted communication schemes are denoted by H.B., N.B., and N.A, respectively. A) Small messages vs. group size. B) Large messages vs. group size.

small messages caused by the increase in the number of communication steps required to cover all destination nodes.

Myrinet host-based binomial and binary tree algorithms provide an identical CPU utilization to that of SCI M_d-torus and M_u-torus. Host-based separate addressing and serial forwarding algorithms both show a perfect linear increase in terms of small-message CPU utilization, where serial forwarding performs better compared to separate addressing.

Myrinet NIC-assisted binomial and binary tree algorithms lower the root-node CPU utilizations as expected. These two algorithms provide the lowest and a constant CPU utilization for all group sizes. Similarly, NIC-assisted separate addressing lowers the CPU utilization compared to the host-based version, and provides a very slowly increasing utilization. Similar reduction is also observed for the NIC-assisted serial forwarding algorithm compared to its host-based counterpart. Unlike NIC-assisted separate addressing, serial forwarding cannot sustain this low utilization, and it increases linearly with increasing group size. The linear increase of the serial forwarding is caused by the ever extending path lengths with increasing group sizes.

Meanwhile, for large messages, it is observed that as group size increases the CPU load with the SCI S-torus algorithm also linearly increases. The SCI separate addressing algorithm has a nearly linear increase in CPU load for large messages with increasing group size. By contrast, since the number of message transmissions for the root node is constant, M_d-torus provides a nearly constant CPU overhead for large messages and small group sizes. However, for group sizes greater than 10, the CPU utilization tends to

increase caused by variations in the path lengths causing extended polling durations. For large messages, SCI M_u-torus also provides higher but constant CPU utilization.

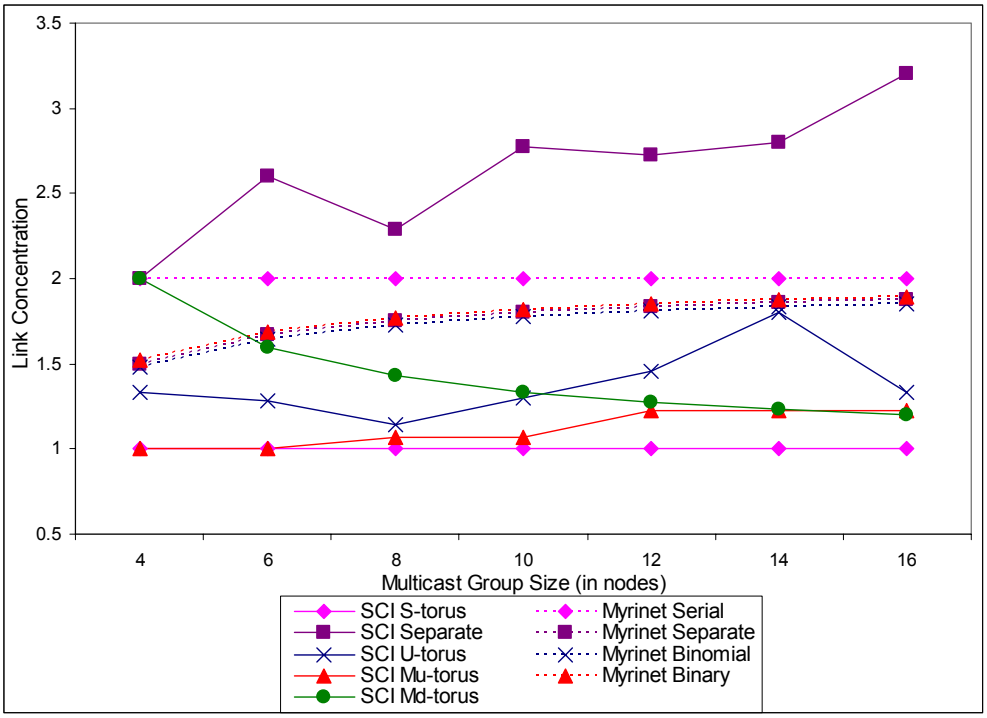
Host-based Myrinet binomial and binary tree algorithms provide similar levels of CPU utilization to SCI M_u-torus for large messages and large group sizes. Similar to the small-message case, host-based separate addressing and serial forwarding have linearly increasing utilizations with increasing group sizes, where serial forwarding performs better compared to separate addressing.

NIC-assisted binomial and binary tree algorithms again achieve the smallest and sustainable constant CPU utilizations for large messages for all group sizes. Similar to the small-message case, separate addressing benefits most from NIC-assisted communication as can be seen from Figure 4-2B. Serial forwarding also exhibits lower CPU utilizations with the NIC-assisted communication.

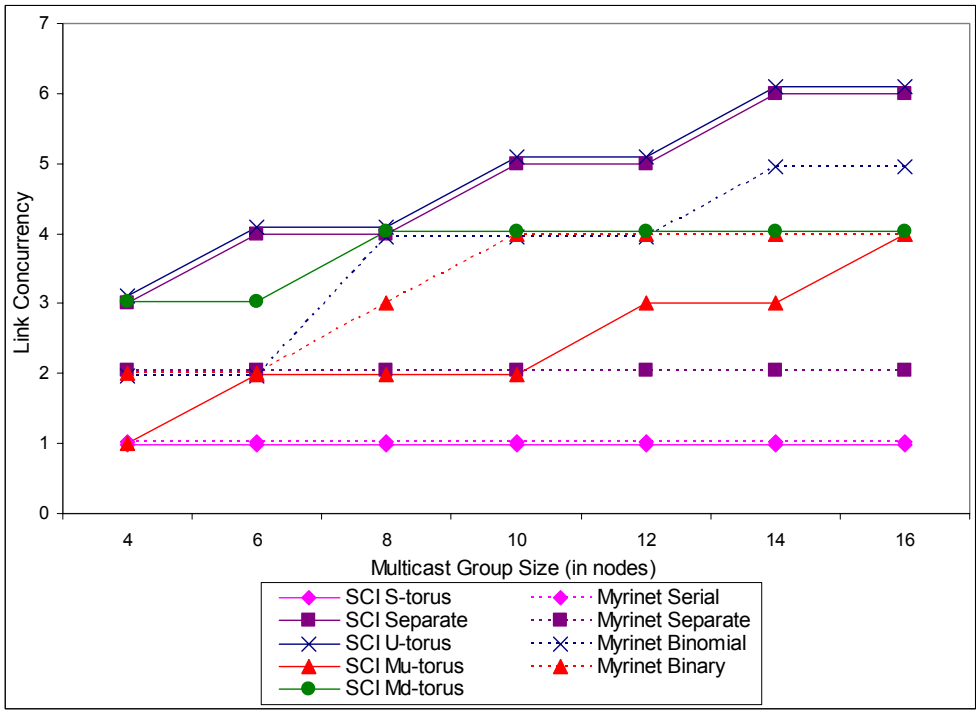
Link Concentration and Link Concurrency

Link concentration and link concurrency for SCI and Myrinet are given in Figures 4-3A and 4-3B, respectively. Myrinet host-based and NIC-assisted communication schemes have identical link concentration and concurrency, so they are not separately plotted. Link concentration combined with the link concurrency illustrates the degree of communication balance. The concentration and concurrency values presented in Figure 4-3 are obtained by analyzing the theoretical communication structures and the experimental timings of the algorithms.

SCI S-torus is a simple chained communication and there is only one active message transfer in the network at any given time. Therefore, it has the lowest and a constant link concentration and concurrency compared to other algorithms. Because of



a



b

Figure 4-3. Communication balance vs. group size. A) Link concentration. B) Link concurrency.

the high parallelism provided by the recursive doubling approach, the SCI U-torus algorithm has the highest concurrency. SCI separate addressing exhibits an identical algorithm has the highest concurrency. SCI separate addressing exhibits an identical degree of concurrency to the U-torus, because the multiple message transfers overlap at the same time caused by the network pipelining feature available over the SCI torus network. The SCI M_d-torus algorithm has inversely proportional link concentration versus increasing group size. In M_d-torus, the root node first sends the message to the destination header nodes, and they relay it to their child nodes. As the number of dimensional header nodes is constant (k in a k-ary torus), with increasing group size each new child node added to the group will increase the number of available links. Moreover, because of the communication structure of the M_d-torus, the number of used links increases much more rapidly compared to the number of link visits with the increasing group size. This trend asymptotically limits the decreasing link concentration to 1. The concurrency of M_d-torus is upper bounded by k as each dimensional header relays the message over separate ringlets with k nodes in each. The SCI M_u-torus algorithm has low link concentration for all group sizes, as it multicasts the message to the partitioned destination nodes over a limited number of individual paths, where only a single link is used per path at a time. By contrast, for a given partition length of constant size, an increase in the group size results in an increase in the number of partitions and an increase in the number of individual paths. This trait results in more messages being transferred concurrently at any given time over the entire network.

The Myrinet serial forwarding algorithm is very similar to SCI S-torus in terms of its logical communications structure. Therefore, as expected it also exhibits a constant

concentration. However, Myrinet serial forwarding has a higher link concentration compared to S-torus, and the difference is caused by the physical structure of the two interconnects. In Myrinet, the degree of connectivity of each host is fixed at 1, whereas in SCI it is *N* for an *N*-dimensional torus system. Similar to S-torus, serial forwarding has the lowest link concurrency. Myrinet binomial and binary trees and the separate addressing algorithm have asymptotically bounded link concentration of 2 with increasing group sizes as the number link visits and the used links are the same both three of them. The number of required links to establish a connection between any two nodes is 2 for a single-switch Myrinet network, which is the upper bound on the number of used links. Myrinet serial forwarding has the lowest and constant link concurrency for all group sizes caused by the reasons explained before. Myrinet separate addressing also has a constant but higher link concurrency. Myrinet binomial and binary tree algorithms have higher and variable link concurrencies with respect to the group size. Binary tree has a bounded fan-out number which decreases the link concurrency compared to the binomial tree.

Summary

In summary, the results reveal that multicast algorithms differ in their algorithmic and communication pattern complexity. The functionality of the algorithms increases with complexity, but occasionally the increased complexity degrades the performance in some circumstances. For some cases, such as small-message multicasting for small groups, using simple algorithms helps to obtain the true performance of the underlying network. For example, because of its simplicity and the inherently lower unicast latency of SCI, the SCI separate addressing algorithm is found to be the best choice for small-message multicasting for small groups.

The lightweight GM software for message passing in Myrinet performs efficiently on complex algorithms. Therefore, while simple algorithms such as separate addressing perform better on SCI, it is observed that more complex algorithms such as binomial and binary tree achieve good performance on Myrinet for small-message multicast communication.

For large messages, SCI has a clear advantage due its higher link data rate compared to Myrinet (i.e., 5.3Gb/s of SCI vs. 1.28Gb/s of Myrinet used in this study). Although the newest Myrinet hardware features higher data rates (i.e., 2.0Gb/s) than our testbed, these rates are still significantly lower than SCI. Therefore, we expect that our results for large messages would follow the same general trend even for the newest generation of Myrinet equipment.

Myrinet NIC-assisted communication provides low host-CPU utilizations for small and large message and group sizes. Complex algorithms such as binomial and binary tree, and simple ones like separate addressing benefit significantly from this approach. However, multicast performance of NIC-assisted communication is directly affected by the cost of SDMA and RDMA operations. The overhead of these operations limits the potential advantage of this approach.

CHAPTER 5
LOW-LATENCY MULTICAST FRAMEWORK FOR GRID-CONNECTED CLUSTERS

Previous chapters analyzed and evaluated the multicast problem on two different SANs. Chapter 5 introduces an analysis of a low-level topology-aware multicast infrastructure for Grid-connected SAN- or IP-based clusters. The proposed framework integrates Layer 2 and Layer 3 protocols for low-latency multicast communication over geographically dispersed resources.

Grid-connected clusters provide a global-scale computing platform for complex, real-world scientific and engineering problems. The Globus Alliance [44] is an important project among the numerous Grid-related research initiatives mentioned in Chapter 1. The Globus Alliance initiative for scientific and engineering computing, which is led by various research groups around the world, is a multi-disciplinary research and development project. Typical research areas that the Globus project tackles include resource and data management and access, security, and application and service development, all on a massive, distributed scale. A collection of services, the Globus Toolkit (GT), has emerged as a result of these collective research efforts. The Globus toolkit mainly includes services and software for Grid-level resource and data management, security, communication, and fault detection. The open source GT is a complete set of technologies for letting people share computing power, databases, and other tools securely online across corporate, institutional, and geographic boundaries without sacrificing local autonomy. The main components of GT are as follows:

- *Globus Resource Allocation Manager* (GRAM). Provides Grid-level resource allocation and process creation, monitoring, and management services among distributed domains.

- *Grid Security Infrastructure* (GSI). Provides a global security system for Grid users over distributed resources and domains. GSI establishes a single-sign-on authentication service for distributed systems by mapping from global to local user identities, while maintaining local control over access rights.

- *Monitoring and Discovery Service* (MDS). Grid-level information service that provides up-to-date information about the computing and network resources and datasets.

GT also has three more software services for establishing homogenous Grid-level access for distributed resources:

- *Global Access to Secondary Storage* (GASS). Provides automatic and programmer-managed data movement and data access strategies.

- *Nexus.* Provides communication services for heterogeneous environments.

- *Heartbeat Monitor* (HBM). Detects system failures of distributed resources.

GSI isolates the local administrative domains from the Grid structure by using gateways or gatekeepers at each domain. These gateways or gatekeepers are responsible for accepting and validating incoming global user login requests and converting them to local credentials. Currently GT implements these gateways and gatekeepers as Condor nodes [45]. Other than GSI specific tasks, Condor gateways are generally also responsible for job submission, scheduling and resource management of local domains and clusters.

Chapter 5 introduces a latency-sensitive multicast infrastructure for Grid-connected clusters that is compliant with the GSI and also with other GT services. The following sections explain the proposed infrastructure in detail.

Related Research

Layer 3 multicast research has been investigated widely for more than a decade. Multicasting over IP networks, such as a Grid backbone, provides a portable solution but is not suitable for high-performance distributed computing platforms because of the inherent high IP overhead. Well-established and implemented Layer-3 protocols exist for both intra-domain and inter-domain Layer 3 multicast.

The Distance Vector Multicast Routing Protocol (DVMRP) [46] is one of the basic intra-domain multicast protocols. DVMRP propagates multicast datagrams to minimize excess copies sent to any particular network and is based on the controlled flood method. Sources are advertised using a broadcast-and-prune method. However, controlled flooding presents a scalability problem. Also, each router on the multicast network must forward incoming source advertisements to all its output ports. Based on the responses of receivers, the routers will forward back prune messages, if required. This process requires each router on the multicast network to keep very large multicast state and routing tables. Overall, DVMRP is not scalable, and is only efficient in densely populated networks.

Protocol Independent Multicast (PIM) [11] is a step beyond DVMRP and aims to overcome the scalability problem of DVMRP. It uses readily available unicast routing tables, therefore eliminating the need to keep an extra multicast routing state table. PIM Sparse Mode (PIM-SM), an extension of the PIM protocol, uses a shortest-path tree method. In this method, “Rendezvous Points” (RPs) are created for receivers to meet new sources. Members join or depart a PIM-SM tree by sending explicit messages to RPs. Receivers need only know one RP initially, which eliminates the need for flooding all multicast network with source advertising messages. RPs must know each other to

transfer data from different senders to various multicast groups. Senders announce their existence to all RPs while receivers query RPs to find ongoing multicast sessions. Actual data transmission does not require RPs to be on the distribution path.

While DVMRP and PIM-SM connect the sources and receivers in the same domain, they do not answer the needs for inter-domain multicasting. The Multicast Border Gateway Protocol (MBGP) [10], an extension of the unicast Border Gateway Protocol (BGP) [47], connects multicast domains. In MBGP, every router only needs to know the topology of its domain and how to access the next domain. Each PIM-SM domain uses its own independent RP(s) and does not depend on RPs in other domains. However, in order to connect the source in a domain with receivers in different domains, the RPs must know each other. Multicast Source Discovery Protocol (MSDP) [12] provides a solution this problem. The MSDP describes a mechanism to connect multiple PIM-SM domains (i.e., RPs) together over the MBGP-connected domains. It enables a mechanism to inform an RP in one domain that there are sources in other domains. MSDP is not scalable for large groups and group sizes. Overall, the current Layer 3 multicast architecture deployed in the commercial Internet, vBNS, and Abilene networks is a combination of PIM-SM, MBGP, and MSDP protocols. Figure 5-1 shows a sample PIM-SM/MBGP/MSDP multicast architecture.

As mentioned earlier, both DVMRP and MSDP are efficient with small groups but are not scalable for large groups. This scalability problem makes current setups, such as PIM-SM/MBGP/MSDP, poor solutions for grid applications. Both of these methods are based on the group membership approach, which is susceptible to congestion and contention problems, as unauthorized sources can freely multicast to a group.

Furthermore, there is no indication of group size from the source's view, which is not favorable from the service provider's point of view for obvious economic reasons. In addition, these methods need unique worldwide-multicast addresses for each group: another scalability and management problem in and of itself.

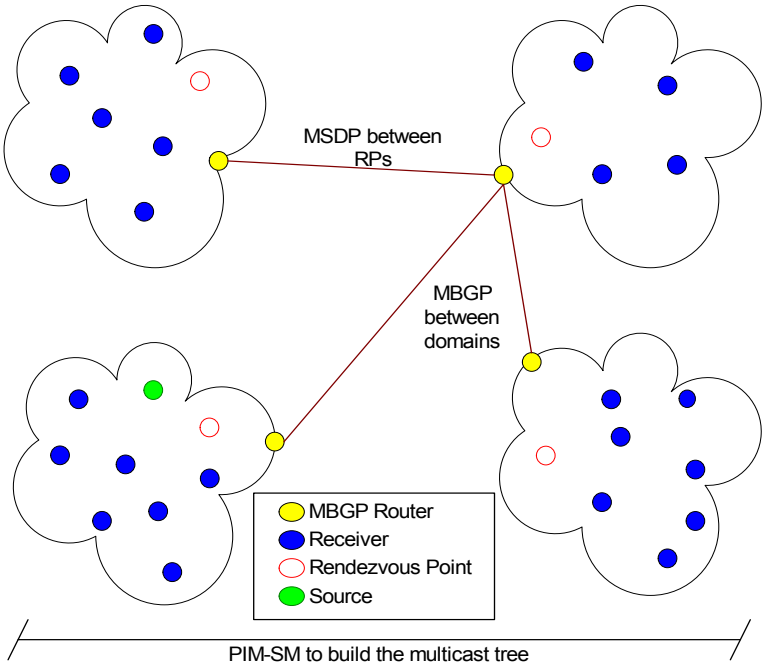


Figure 5-1. Layer 3 PIM-SM/MBGP/MSDP multicast architecture.

New ideas address these shortcomings. For example, Express Multicast (EM) [48] is based on the idea that most multicast applications are either single-source or have an easily identifiable primary source. This assumption greatly reduces the complexity of multicast algorithms. The EM is also based on the channel membership concept, in which each source (*S*) defines a channel (*S*,*E*), where *E* is the channel address, and only *S* can send to (*S*,*E*). Channels are source addressed, which eliminates the requirement for the receivers or receiver groups to have a unique multicast address in the global scale, further increasing scalability. Although different sources can use the same channel addresses, this situation will not create a conflict, as each channel is identified by the

combination of the source and channel address; and no two sources can have the same address. In EM, receivers send explicit join messages to *S*, which eliminates the scalability problem of previous algorithms caused by their broadcast-and-prune approach. EM uses bidirectional trees to reduce the state table overhead on multicast routers. Figure 5-2 shows examples of channel membership and group membership multicast.

Although the channel membership approach solves the scalability problem of the group membership approach, it does not fit the current Globus/Condor architecture. The Condor gateway nodes, which are ordinary nodes, not routers, isolate local domains and clusters from the rest of the Grid network. This current architecture prevents the EM root from building a full-scale channel tree that reaches the receiver nodes in local domains.

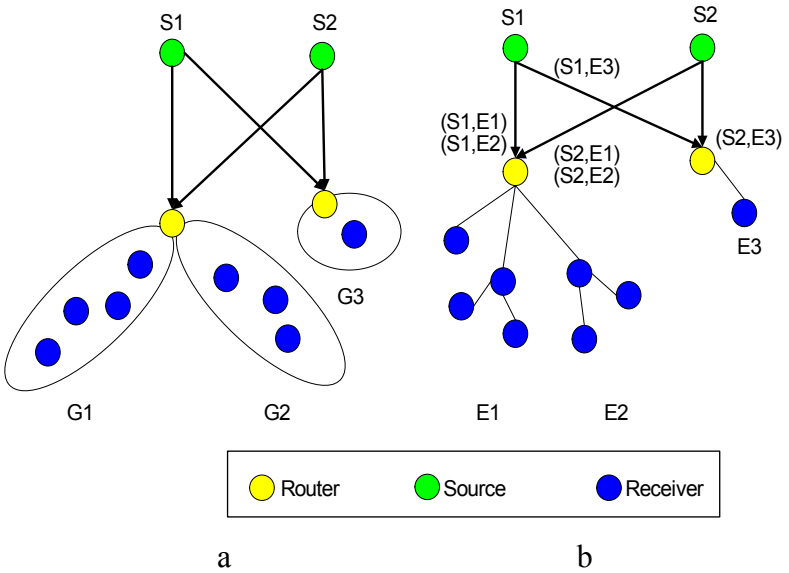


Figure 5-2. Sample group versus channel membership multicast. A) Group membership. B) Channel membership.

In summary, the existing multicast approaches using group membership or channel membership do not propose an answer to solve the Grid-level cluster-to-cluster multicast problem by themselves. The proposed solution is this dissertation combines the best qualities of these two approaches to solve the Grid-level problem. In this proposed

approach, channel membership is used between the domains and group membership is used within domains. Channel membership between the domains provides a tree architecture for gateways while lowering the WAN backbone impact and complying with the GSI. Group membership within domains provides a simple scalable solution for domains and clusters. Figure 5-3 shows a sample multicasting scenario for Grid-connected clusters. The clouds in the figure represent the domains. The next subsection explains the proposed framework in detail.

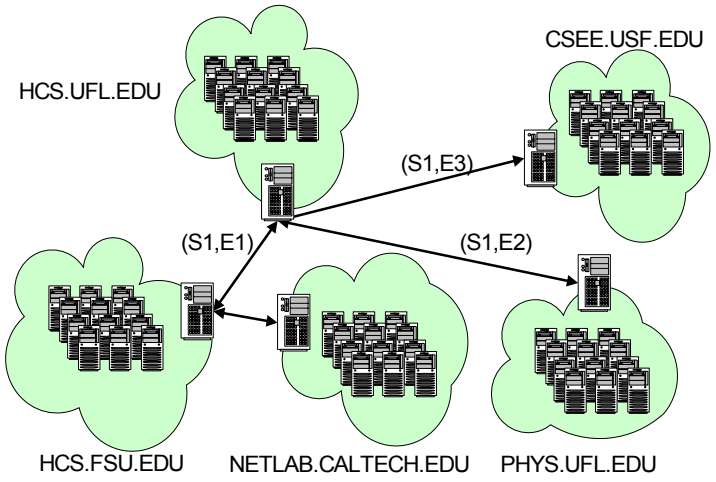


Figure 5-3. Sample multicast scenario for Grid-connected clusters.

Framework

The proposed multicast framework is based on the assumption that participating nodes (i.e., sources and receivers) can be clearly organized in terms of administrative domains and each domain has a gateway node. These gateway nodes are Condor-like nodes in the current Globus architecture, not specific routers, which complies with the current Globus architecture. The main difference between the current Globus gateway nodes and those that we propose is that the proposed framework's nodes are responsible for job distribution, as is the case in Globus, and also capable of initiating, organizing, orchestrating and managing multicast communication. The gateways are identified as

regular system nodes, and have been simply identified as the gatekeepers of their domain. Gateways in the proposed framework can be a dedicated node or part of the multicast group, and this decision is left to the system administrator of a given domain. All access to a specific domain has to go through these nodes, as they are the only interface allowed to and from the outside world. This approach also eliminates the need for users to have extra administrative rights over multiple domains.

The gateways are organized in a binary or binomial tree structure based on their distance from the root. The gateway distances are measured in terms of their latencies. At initialization the source gateway informs each gateway that participates in the multicast about the existence of other participating gateways. Each gateway, upon receiving this information from the source gateway, measures the latency between itself and all other gateways and sends the results to the source gateway. The source composes the gateway latency graph of participating gateways to build the channel-based gateway multicast tree.

The single source of the multicast operation is located at the root of the tree. This leads to a minimal number of message transfers initiated by the root, and minimized impact on the WAN backbone. Each source that participates in multicast communication builds its own version of the top-level tree.

As the top-level tree build, join, and setup operations are performed offline; they are no longer on the critical communication path. Also, once built, the same top-level tree can be used multiple times as long as the network setup and participant list remains the same.

The top-level tree is built by partitioning the all-participant latency graph into physical domains based on neighborhood distances. This idea was introduced by Lowenkamp *et al.* [49] and has been widely used in MPICH-G [50] and MPICH-G2 [51] and other works for optimizing WAN communications. The partitioning is based on the cost of the link between neighbors. Each new link introduced to the multicast tree is checked for its cost. The source assesses the new node's cost connection to every node in the system. As a result, the least expensive connection for the new node is determined, and added to the multicast top-level tree. Lowenkamp *et al.* have chosen a 20% cutoff value empirically for cost assessment. They have experimentally showed that this cutoff value is accurate [49].

The outline of the top-level tree building process is as follows:

- Start with source as the root to the binomial tree
- Add the domain with the minimal cost link at each step to the tree
- Stop when all domains are analyzed and embedded in the binomial tree

The pseudocode for this process is given below:

```
Number nodes in any order starting from 1
   Source =1
N={root}
U=U-N
For all nodes in U
   For all nodes in N
      Choose a node from U with minimum cost with respect to source
      Compare the cost of new node with the cost of ALL nodes
      If cost is NOT less than (1.2×cost of ALL existing routes)
         Add new node as a child to the lowest cost connection
      Else
         Add new node as a separate branch connected to source
      Add new node to N
      Delete node from U
      Repeat process until U={}
```

where U is the universal set of all available nodes and N is the set of selected nodes.

Figure 5-4 shows the top-level tree building process. The process starts with the source node and evaluates the cost of each new node added at every step. In Figure 5-4

HCS.UFL.EDU is chosen as the source node in Step 1. Figure 5-4A shows the initial link latencies for this example.

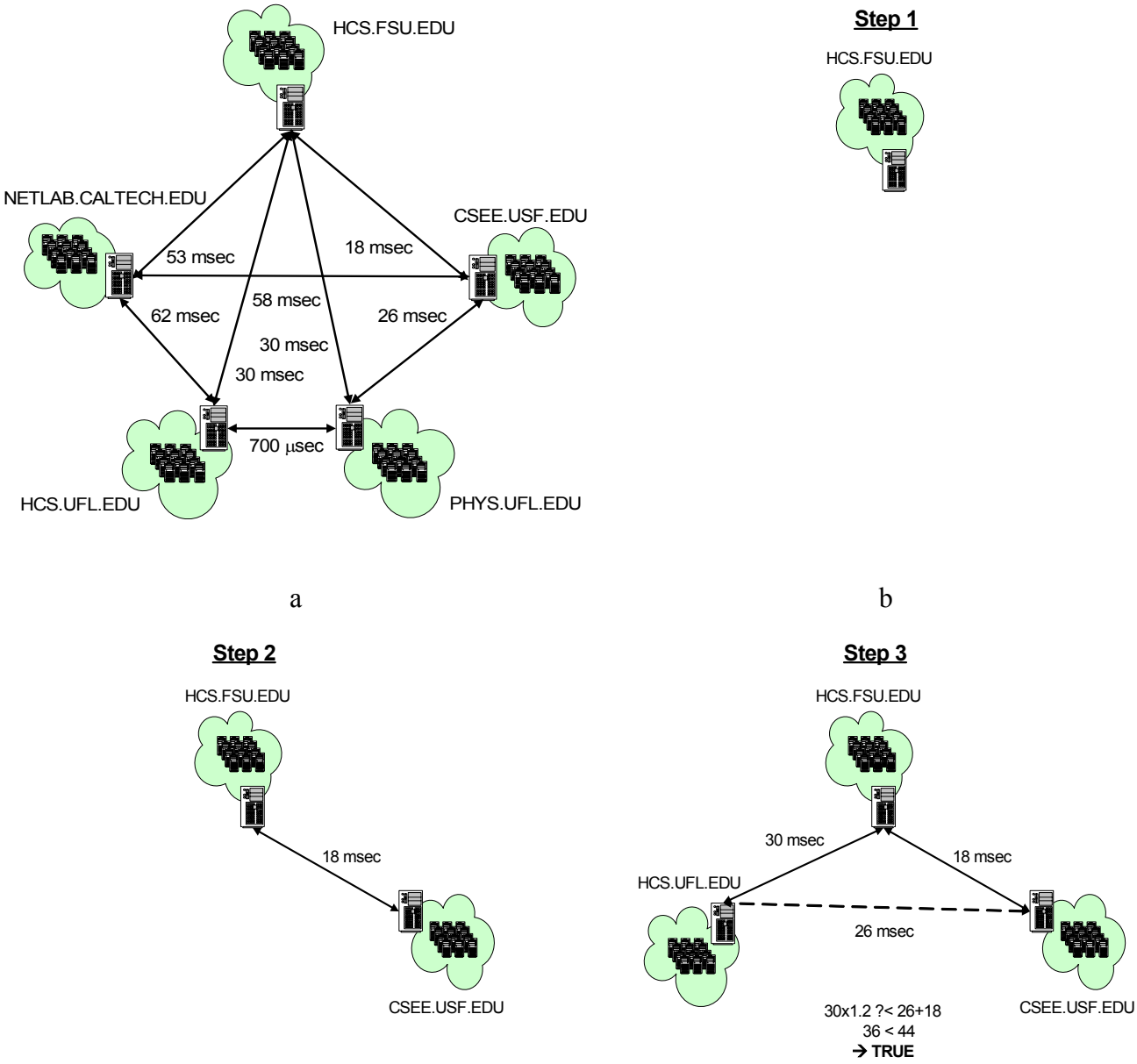
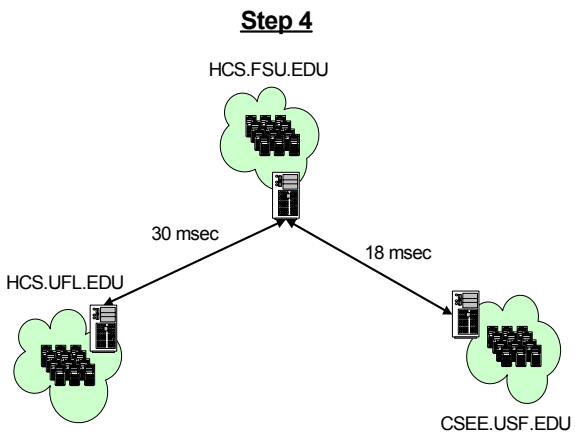
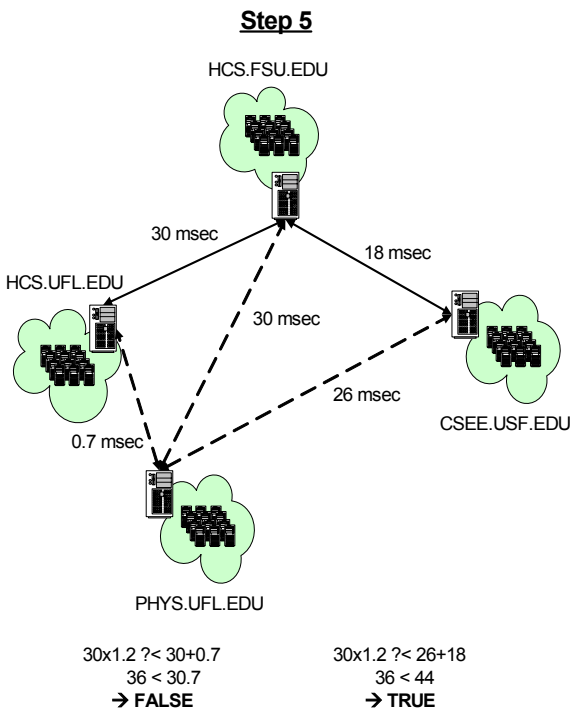


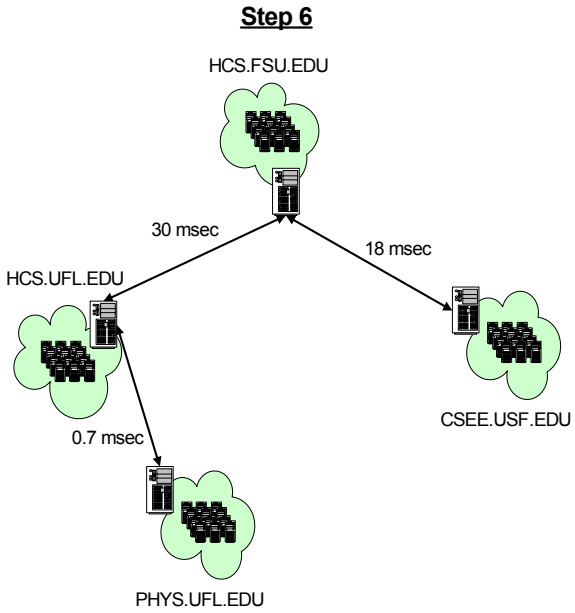
Figure 5-4. Continued.



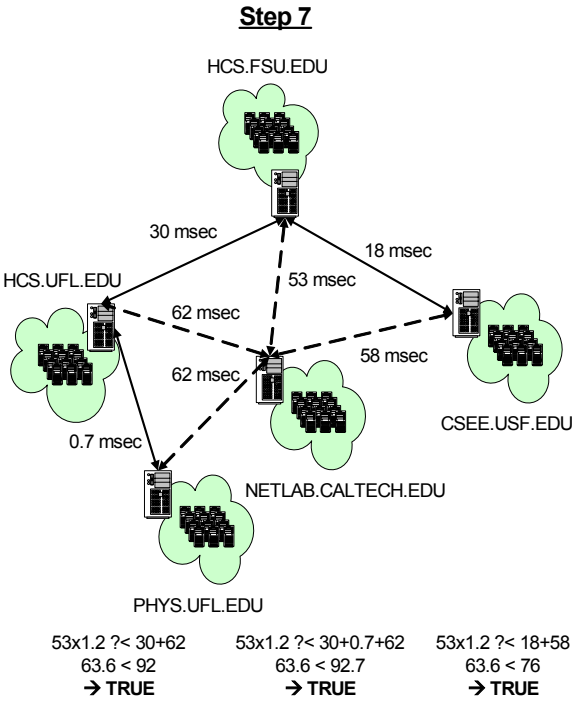
e



f

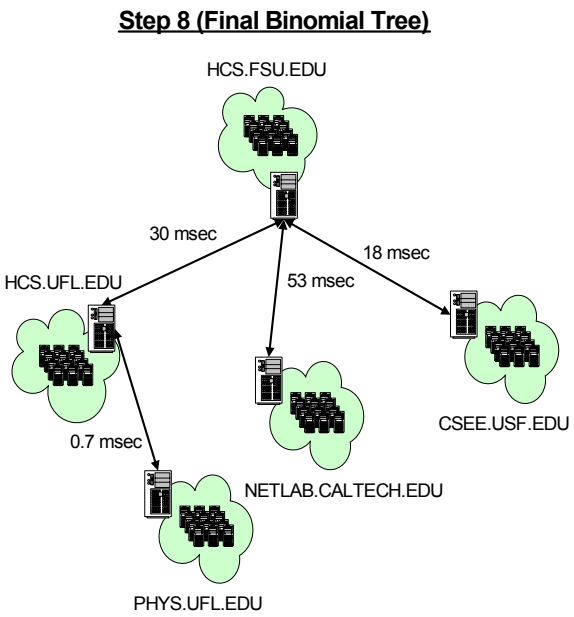


g



h

Figure 5-4. Continued.



i

Figure 5-4. Step by step illustration of the top-level tree building process.

The gateways simultaneously form their own intra-domain multicast groups while participating in the top-level multicast tree. Intra-domain multicast is based on group membership. Although it is not mandatory, the DVMRP protocol is chosen for intra-domain multicast. DVMRP is known to have a scalability problem for large groups and topologies, but limiting this protocol only to intra-domain multicast eliminates the scalability problem. If necessary, domain system administrators are free to choose any intra-domain multicast protocol that supports group membership, as long as the intra-domain multicast group and the top-level tree are strictly isolated from one another at each gateway. Isolation and intra-domain group multicast combined together enforce a closed-group concept for security, where only its domain gateway can multicast to any particular closed group. Moreover, direct Layer-3 connection with a local/cluster node from the outside world is not allowed in most of the current Grid-connected clusters for

security reasons. Information about local domains and clusters can be obtained from the gateways through a resource allocation service such as GRAM, or an information service such as MDS, which is beyond the scope of this work. Joins and departures from the multicast group in a domain are provided with broadcast-and-prune messages. For performance reasons, domains with more than one subnet can be divided into hierarchical groups. Figure 5-5 shows an example of the proposed group-membership based multicast approach.

Different types of clusters with various interconnects can participate in local multicast groups. These might be Ethernet-based or SAN-based clusters. Ethernet-based clusters support IP, while SAN-based clusters allow proprietary SAN or high-performance BSD-compatible protocols for SANs to be used for communication. If a SAN protocol that is not BSD-compatible is used for intra-domain multicast, then a packet conversion from IP to the SAN protocol or vice versa is required. Where BSD-compatible protocols for SANs exist, a seamless transformation from or to BSD sockets can be achieved.

If the selected interconnect is a SAN, then there are two possible scenarios for a gateway placement. It can either be part of the SAN subnet, in which case it can convert and transmit incoming multicast data to receivers directly over the SAN, or it can be outside of the SAN subnet, in which case it will transmit incoming multicast data to a head-node in the SAN subnet for further conversion and delivery to the multicast group. Figure 5-6 shows these two possible gateway placement scenarios.

To further increase the performance of the proposed multicast framework, an extension to the gateways is introduced. Because of link errors and router buffer/queue

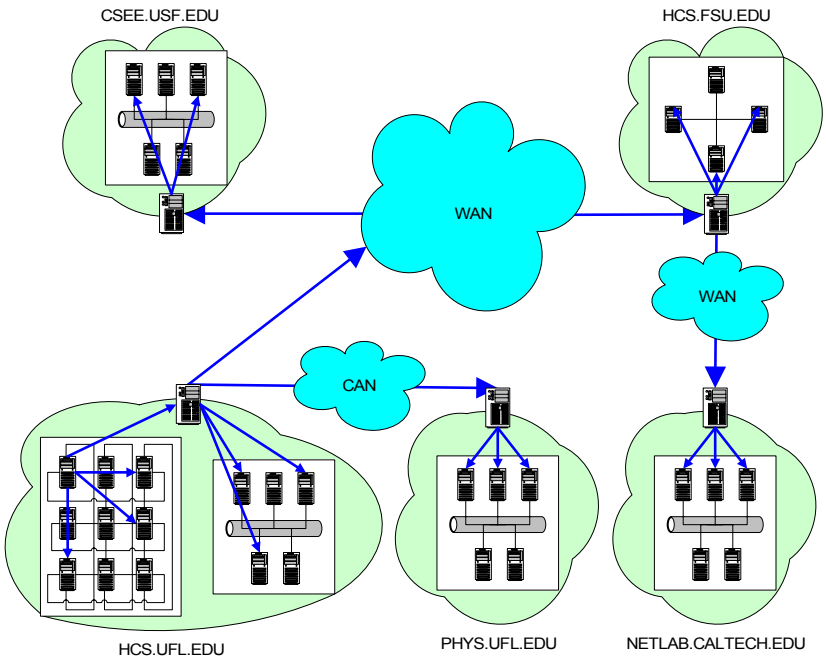


Figure 5-5. Sample illustration of group-membership multicast approach.

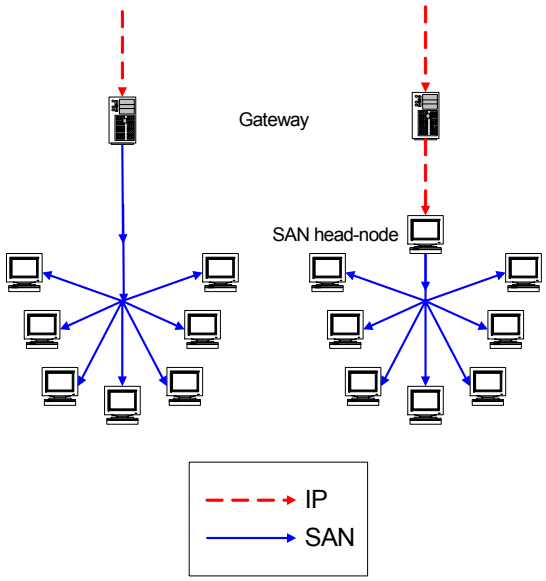


Figure 5-6. Possible gateway placement and incoming packet forwarding scenarios.

problems, transmission errors may occur. When such errors occur, it is the receivers' responsibility to request a retransmission from upper-level data providers (i.e., gateways). In the proposed scheme, gateways keep a local copy of incoming multicast data in a local cache. When the closest gateway to the re-transmission requestor intercepts a request, it

automatically suppresses the request and provides the data from its local cache. This approach requires extra storage on gateways, but when considering that 1) these gateways are regular nodes and not specific high-cost switching devices and 2) storage device prices are decreasing while their capacities are constantly increasing, it is obvious that this would not impose much of an extra burden on the system. It is also obvious that this approach will provide lower retransmission latencies when compared to a single source and single data-holder approach. Also, the proposed method will provide lower backbone impact as the requestor can at most only be one hop away from the gateway which provides the data in the system. Figure 5-7 shoes this low-latency retransmission process. As can be seen, the gateway which requests a retransmission, PHYS.UFL.EDU, is only one hop away from the gateway that provides the data, HCS.UFL.EDU. This retransmission is not visible by the rest of the top-level gateways therefore it does not bring an additional impact to the system.

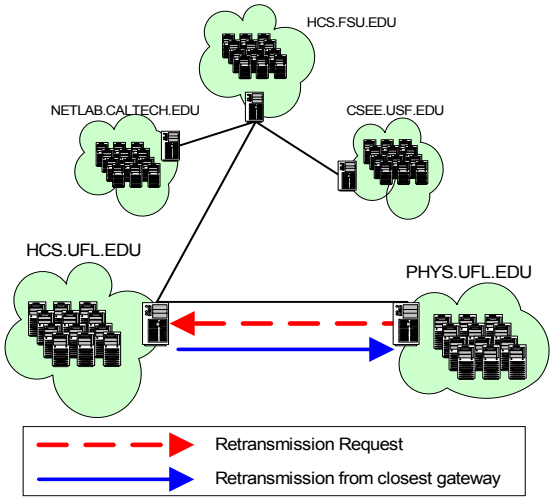


Figure 5-7. Illustration of low-latency retransmission system.

In summary, the proposed framework defines a low-latency, multi-protocol framework to connect distributed clusters over WAN backbone links for multicast

communication, integrating Layer-3 and Layer-2 multicast protocols and algorithms. It takes advantage of faster interconnects whenever possible and appropriate, such as SANs. The design is not complex, and does not require any changes in existing deployed routers. It also only requires minimal changes to existing gateway nodes and complies with the current Globus architecture. The proposed framework in this dissertation provides lower impact on WAN backbones because of the top-level tree-based architecture. The framework does not require extra access privileges to the current administrative structures, and therefore is also compliant with the GSI. The proposed framework assists different classes of users, such as Grid end users and Grid application and tool developers, with enhanced performance, functionality and scalability features. The following sections describe simulative performance analysis case studies for the proposed framework.

Simulation Environment

As experimentation time on grid-scale systems is not feasible, a simulation study based on grid-level infrastructure is used to evaluate and analyze the proposed design versus previous schemes. Mission-Level Designer (MLD) from MLDesign technologies, Inc., was used for this purpose [52]. While MLD is an integrated software package with a diverse set of modeling and simulation domains, the discrete-event domain for event-driven simulation of data transfer systems is used in this study. The tool allows for a hierarchical, dataflow representation of hardware devices and networks with the ability to import finite-state machine diagrams and user-developed C/C++ code as functional primitives. Figure 5-8 shows a screenshot of the MLD simulation tool. The computer systems and networks simulation capability of MLD is based upon its precursor, the

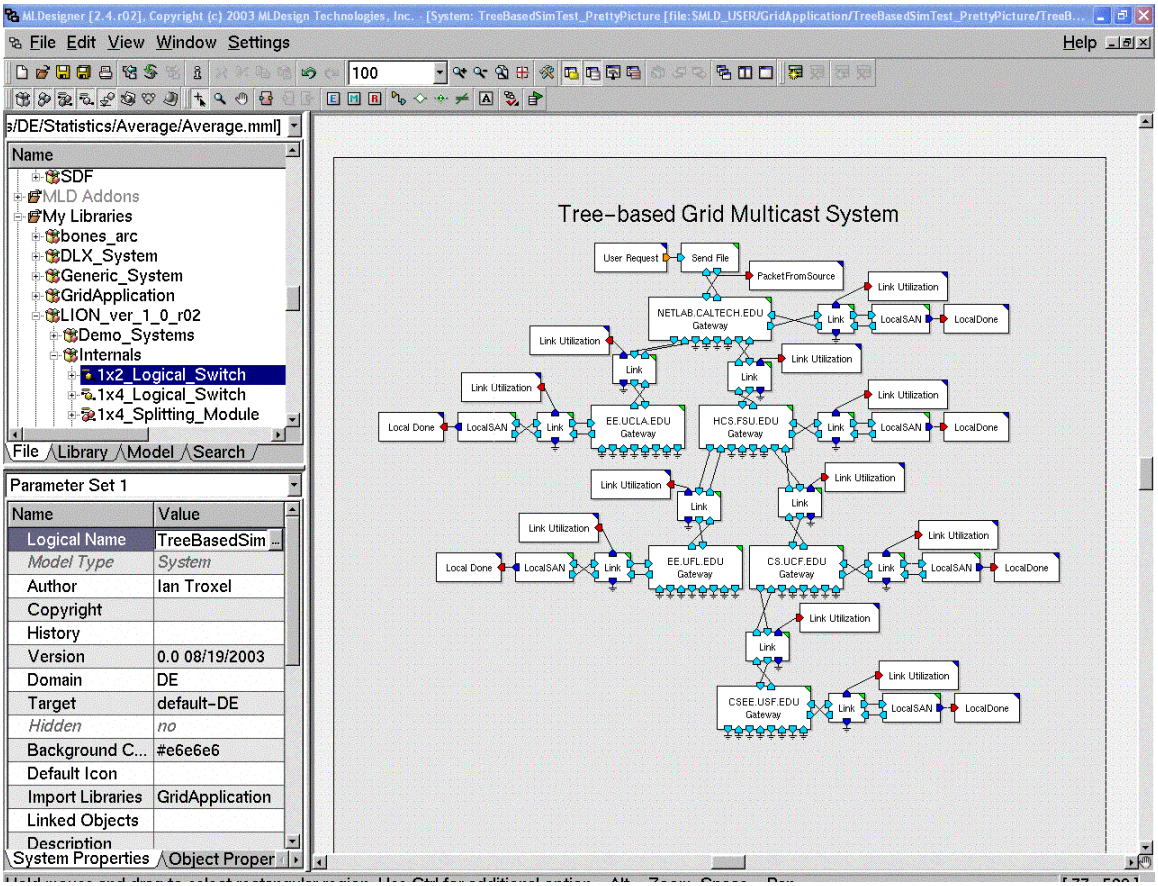


Figure 5-8. Screenshot of the MLD simulation tool.

Block-Oriented Network Simulator (BONeS) Designer tool from Cadence Design Systems [53]. The BONeS was developed to model and simulate the flow of information represented by bits, packets, messages, or any combination of these.

Case Studies

To analyze and evaluate the performance of the proposed multicast framework, two sets of simulative case studies are performed. The simulation model is calibrated and validated using real-world WAN, LAN, and SAN link latencies and experimental results. The following sections provide details about the simulative case study setups.

The first case study analyzes a multicast scenario for latency-sensitive parallel applications over SANs connected over the WAN backbone. The multicast scheme (i.e.,

all local multicast, all remote multicast, or mixture of these two) for various multicast group sizes (i.e., 4, 8, 12, and 16) for a 64KB multicast message size is evaluated for two WAN-connected SAN clusters. Myrinet and SCI are the selected SANs and various multicast algorithms analyzed in previous chapters are used for this simulative study. Gateway placement is also another independent variable for both cases. Gateways can be set up as part of the SAN clusters or as independent nodes outside the clusters. This case study analysis botch cases to better evaluate the effects of each setup. For the case where the gateway is not part of the SAN cluster, a slow link, such as Fast Ethernet, that connects the gateway to the SAN head node and a fast one, such as Gigabit Ethernet, are analyzed. Multicast completion latency is the only dependent variable chosen in this case study.

The second case study is focused on large-file (i.e., Terabyte and above) data distribution and staging over multiple domains connected through a WAN grid backbone. For this case study, retransmission rate, network topology and number of hops (i.e., 0, 1, 2, and 3), and local domain interconnect are chosen as independent variables. The system is analyzed for retransmission rates of 1%, 5%, 10%, and 20%. Different local domain interconnects, such as Ethernet variants and SCI and Myrinet SANs, for flat, tree with no local caching (Tree N.C.), and tree with local caching (Tree L.C.) network topologies are also investigated. The multicast message size and the file size are set to 64KB, and 1TB, respectively. Dependent variables for this case study are end-to-end multicast completion latency, WAN backbone impact, and local cluster interconnect utilization. The next section discusses the results obtained for each set of case studies in detail.

Case Study 1: Latency-sensitive Distributed Parallel Computing

Many programming models and libraries exist today for parallel systems, but the most well-known ones are MPI, PVM, and Active Messages [54]. Currently, only MPI supports distributed parallel computing over Grid networks (i.e., MPICH-G, MPICH-G2 and variants), but MPI is based on the message passing paradigm and might not be suitable for all problems and cases, such as shared-memory applications. This case study evaluates how a low-latency distributed parallel computing application might benefit from the proposed framework.

User starts the multicast communication by inputting the node/job list. The gateway is notified about the remote multicast IPC calls at run-time. The gateway generates the receiver domain list and the number of nodes per domain for remote multicast calls. The gateway then notifies each SAN domain gateway of the total participant node list. Each SAN domain gateway will have its own version of the top-level multicast tree as multicast communication for such an application might be initiated from any of the participating computing nodes. Each SAN gateway sends "*probe messages*" to all domain gateways on the receiver list and waits for replies. The total participant list is also included with the probe messages. Upon receiving explicit "join messages", each gateway forms its version of the top-level tree based on the request-reply latencies. Then they each send "*channel messages*" (*S*, *E*) to first-layer gateways with the appropriate child gateway information embedded in the channel messages. Simultaneously, they also form multicast groups in their local SAN clusters. They then await the "*multicast tree ready*" signal from the first-layer gateways. The tree is formed off-line and can be re-used multiple times during the life time of the application. Figure 5-9 shows the tree-formation actions for the outlined scenario.

During the computation phase IPC will be required. The frequency and characteristics of IPC are dictated by the computational code and may be coarse-grained or fine-grained, and may employ small messages or large messages. For increased efficiency, IPC multicast calls can be organized as intra-SAN multicast, which may use the SAN links without any modification, or inter-SAN multicast, which is directed to the gateway for further routing over the WAN backbones.

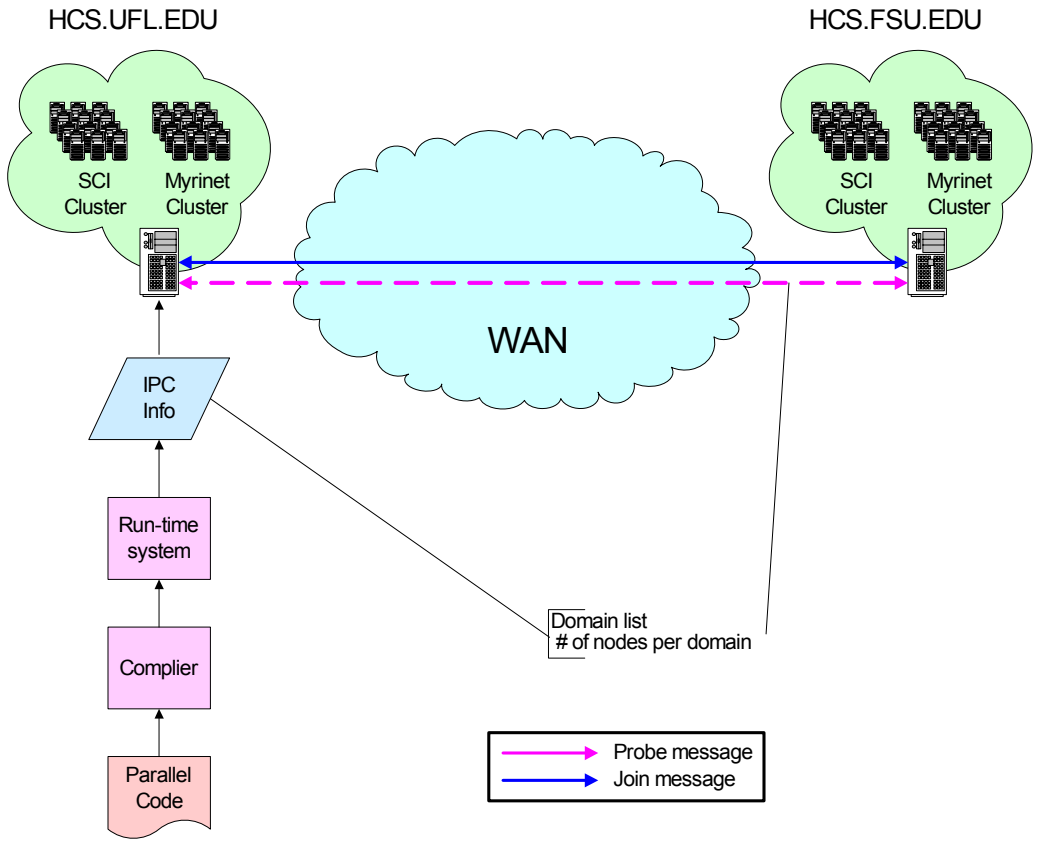


Figure 5-9. Latency-sensitive distributed parallel job initiation and top-level multicast tree formation

In the case of an inter-domain multicast IPC request, receiver node IDs are transmitted along with the message to the gateway. There are two possible setups. The domain gateway may be part of the SAN cluster or it may be a dedicated node. If the gateway is a dedicated node, then the SAN head-node will convert the SAN packet into

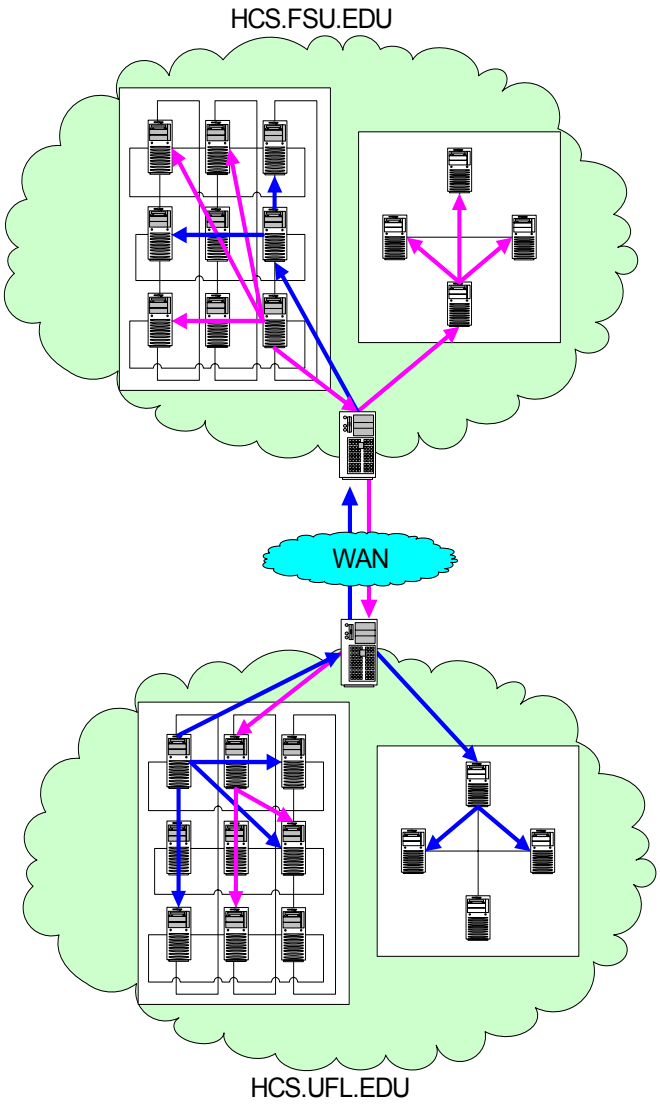


Figure 5-10. IPC multicast communication scenario for Grid-connected SAN clusters

an IP packet, and transfer the message to the domain gateway over the Layer-3 WAN backbone. Otherwise, the gateway will convert SAN packets into IP packets. The gateways also extract the domain information from the receiver node IDs, organize inter-SAN multicast IPC requests, and transfer them to the next level gateway. IPC multicast calls routed by the gateways propagate over the pre-defined multicast tree. Upon intercepting the multicast IPC call at the gateway of the receiver domain, the reverse order of the initiation process takes place. When needed, retransmission requests

will be handled by the closest gateway. Figure 5-10 shows multicast communication for a low-latency distributed parallel computing scenario.

In this simulation model two clusters are arbitrarily chosen to be located in the HCS.UFL.EDU and HCS.FSU.EDU domains. The WAN latency between these two clusters is experimentally measured as ~30msec for a 64KB message size. In our scenario we have chosen a SAN cluster at each site, but other possibilities such as Ethernet-based clusters or a mixture of SAN-based and Ethernet-based clusters may also be applied to the model. The SAN-to-IP and IP-to-SAN packet conversion latency is measured experimentally on a Pentium III machine as ~3μsec for both SCI and Myrinet SANs. Figure 5-11 shows a snapshot of the simulation model.

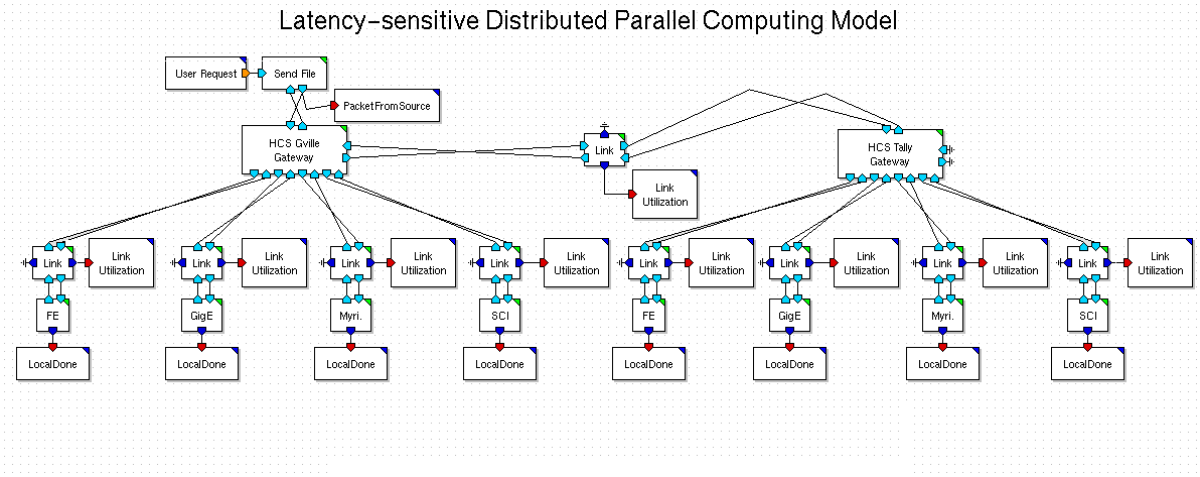
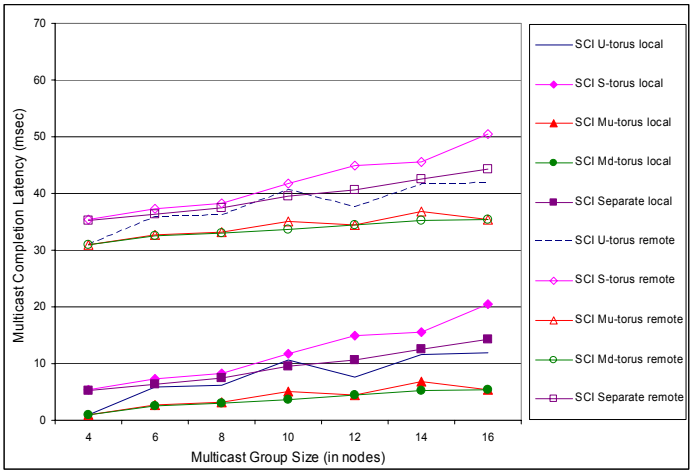
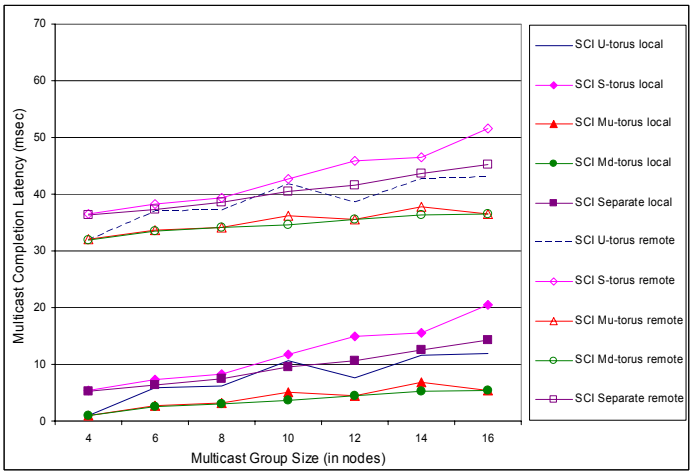


Figure 5-11. Snapshot of simulation model for latency-sensitive distributed cluster multicast

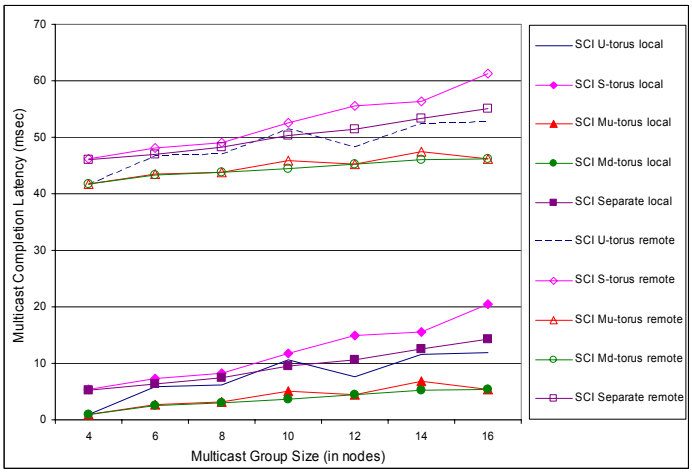
Figure 5-12 shows the multicast completion latencies for various system setups for all remote multicast IPC calls. SCI serves as the SAN cluster interconnect for these trials. Figure 5-12A shows the case where the gateway is part of the local SCI SAN cluster. Figures 5-12B and 5-12C show cases where the gateway is not part of the local SCI SAN cluster, and Gigabit Ethernet and Fast Ethernet are the gateway-to-SAN links, respectively.



a



b



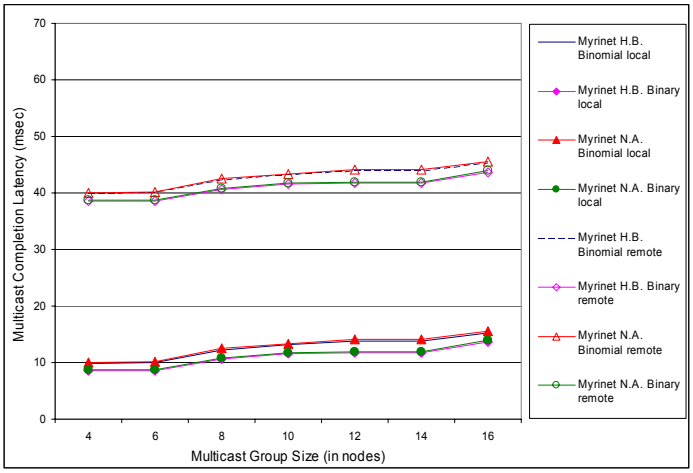
c

Figure 5-12. SAN-to-SAN multicast completion latencies. A) Gateway is part of the SCI SAN cluster. B) Gateway is a dedicated node and gateway-to-SAN interconnect is Gigabit Ethernet. C) Gateway is a dedicated node and gateway-to-SAN interconnect is Fast Ethernet.

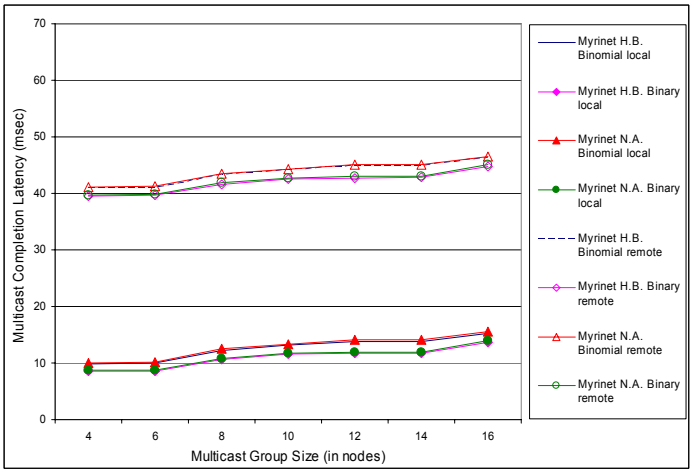
For all cases, the WAN backbone latency is the dominant component and is fixed for a given system. Better performance is obtained when the gateway is part of the SAN cluster, or a fast enough (e.g., Gigabit Ethernet, ~1.2msec one-way latency at 64 KB) gateway-to-SAN link is used. If a slower (e.g., Fast Ethernet, ~12msec one-way latency at 64 KB) interconnect technology is used for the gateway-to-SAN link, it may be the major bottleneck of the system.

Figure 5-13 shows multicast completion latencies between two SAN clusters placed in two different domains and connected over the WAN backbone. The SAN clusters are using a Myrinet interconnect, and all the multicast IPC calls originate from one SAN cluster and are targeted to nodes in the remote cluster. Figure 5-13A shows the case where the gateway is part of the local Myrinet SAN cluster. Figures 5-13B and 5-13C show cases where the gateway is not part of the local Myrinet SAN cluster, and Gigabit Ethernet or Fast Ethernet is the gateway-to-SAN link, respectively.

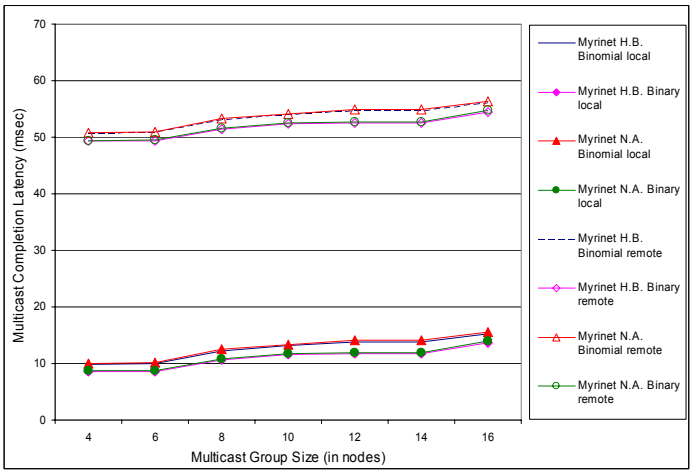
Figure 5-13 shows similar results to the SCI-SCI all-remote case given in Figure 5-12. The WAN link latency is again the dominant component and is on the critical path. Once again the results reveal that it is important to ensure that the chosen gateway-to-SAN link does not impose additional latency if the gateway is not part of the SAN cluster. Figures 5-12 and 5-13 show that Gigabit Ethernet is a viable gateway-to-SAN link for current WAN and SAN technologies, if the gateway is a dedicated node, and that SCI and Myrinet clusters with integrated gateways provide good gateway-to-SAN link for current WAN and SAN technologies, if the gateway is a dedicated node, and that SCI and Myrinet clusters with integrated gateways provide good completion latencies. Also, the results show that a slow interconnect, such as Fast



a



b



c

Figure 5-13. SAN-to-SAN multicast completion latencies. A) Gateway is part of the Myrinet SAN cluster. B) Gateway is a dedicated node and gateway-to-SAN interconnect is Gigabit Ethernet. C) Gateway is a dedicated node and gateway-to-SAN interconnect is Fast Ethernet.

Ethernet, as a gateway-to- SAN link imposes an additional bottleneck and increases the completion latency. Overall it was observed that the total all-remote multicast completion latency for Grid-connected distributed clusters, denoted as $t_{all\_remote\_multicast}$ in this dissertation, with a sufficiently large remote multicast group size can be expressed as given in Eq. 5-1.

$$t_{all\_remote\_multicast} = t_{gateway-to-SAN} + t_{conversion} + t_{WAN} + t_{conversion} + t_{gateway-to-SAN} + t_{remote\_completion} \quad (5\text{-}1)$$

where $t_{gateway\text{-}to\text{-}SAN}$ denotes the gateway-to-SAN link latency, $t_{conversion}$ denotes the packet conversion latency, t_{WAN} denotes the WAN link latency, and $t_{remote\_completion}$ is the multicast completion latency in the remote cluster.

Figure 5-14 shows the mixed mode (i.e., local and remote multicast calls combined) IPC multicast calls between two SAN clusters connected with a WAN backbone. SCI and Myrinet are used as the SAN interconnects. The system setup for SCI and Myrinet simulations are identical. The local and remote multicast group sizes are given as (*Local*, *Remote*). The total mixed multicast completion latency is the greater of the local multicast completion latency and the total all-remote multicast completion latency given above. For clarity, only the best performing SAN multicast communication schemes and algorithms are included in these trials.

As can been seen, the remote completion latency dominates for small local and remote group sizes. However, local completion latency is likely to dominate for significantly larger local group sizes. Latency hiding appears to be an important aspect to consider, as it is possible to hide the local multicast latency for small multicast group sizes, because it is relatively small compared to initiating and completing a remote

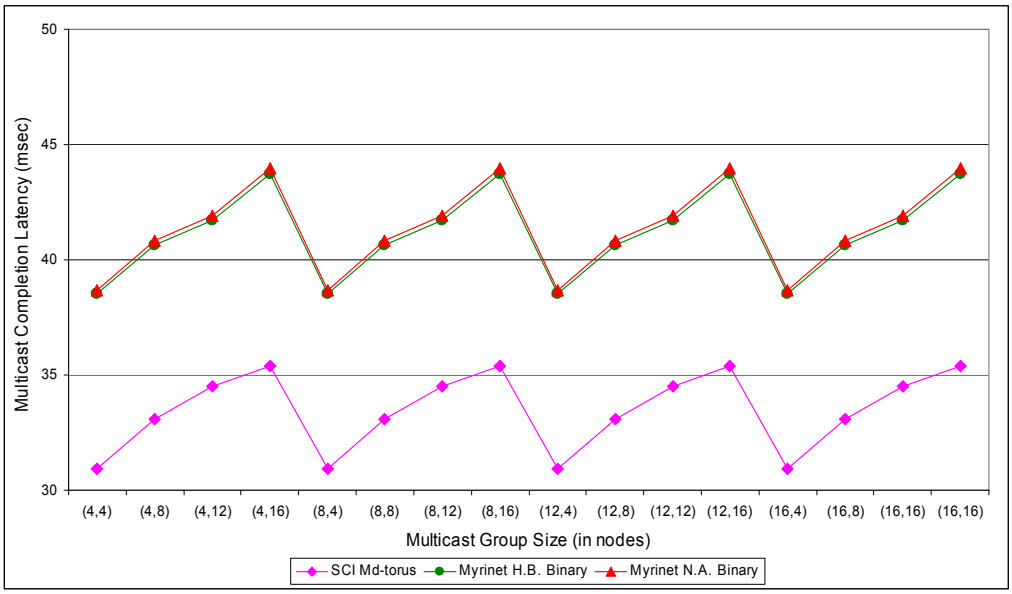


Figure 5-14. Mixed mode (local, remote) IPC multicast completion latencies for SCI and Myrinet SAN clusters connected with a WAN backbone.

multicast call. Similarly, significantly large local multicast group sizes will be likely to hide the remote multicast latency. The total multicast completion latency for the mixed case, denoted as $t_{mixed\_multicast}$ in this dissertation, can be expressed as:

$$t_{mixed\_multicast} = Max\left\lfloor t_{multicast}, t_{all\_remote\_multicast} \right\rfloor \quad (5\text{-}2)$$

where $t_{multicast}$ is the multicast completion latency in the local cluster as in previous chapters.

In summary, this case study reveals that the WAN latency is the dominant latency component for small group sizes. Moreover, the gateway-to-SAN link plays an important role on the overall system performance. The capacity of the gateway-to-SAN link determines if the gateway should be part of the SAN cluster or not. The remote completion latency dominates when local and remote group sizes are small, while the local completion latency is likely to dominate for larger group sizes. Hiding the local multicast latency for small local multicast group sizes is possible as it is small compared

to a remote multicast call, while significantly large local group sizes can hide the remote multicast latency.

Case Study 2: Large-file Data and Replica Staging

Grid and distributed scientific and engineering applications require transfers of large amounts of data between storage systems and access to large amounts of data by distributed applications and users. The *GridFTP* is a tool to distribute and manage large volumes of geographically dispersed data over Grids [55]. It is an enhanced version of FTP with parallel and partial data transfer, among other features. It uses point-to-point unicast communication, and the proposed multicast framework can be used to enhance the efficiency of GridFTP.

The case study begins with a user data or replica staging request. The request, consisting of the identification of the source data, receiver domain list, and number of nodes per domain, is then transmitted to the actual data holder (i.e., data storage site, database maintainer).

Upon receipt of the request by the data server (source) the top-level tree formation phase starts. The source sends a "*probe message*" to each domain gateway on the receiver list and waits for replies. Upon receiving explicit "*join messages,*" it forms the top-level tree based on the request-reply latency that is collected from each gateway. The source sends "*channel messages*" (S, E) to the first-layer gateways. Lower-layer gateways are listed for each gateway in the channel messages. The source awaits the "*multicast tree ready*" signal from first-layer gateways. Upon receipt of the "*multicast tree ready*" messages, it notifies the user that the tree is formed. The state and routing information of the top-level tree is stored in the gateways so that the tree can be used multiple times.

Gateways listen to incoming requests and upon intercepting a "*probe message*" they first extract the participant gateways list and start sending "*probe messages*" to each. When they have received replies back from all the other gateways, they send an explicit "*join message*" with the results from the other gateways to the source gateway. The control messages are exchanged in an all-to-all fashion for the probe messages, enabling each gateway to evaluate its neighbors and their relative distances. This information helps the source gateway obtain a graph of the participants list so it can build the most efficient top-level tree. First-level gateways receive "*channel messages*" (*S*, *E*) after sending out the "*join messages.*" Each "*channel message*" includes the list of respective child gateways, if exist, for each top-level gateway and for each branch of the tree. First-level gateways propagate the "*channel message*" down to their child gateways, while they also simultaneously form a local multicast group in their own subnets, if necessary. These local groups are built based on the user's preset criterion. They will then wait for the "*multicast tree ready signal*" from their child gateways. After receiving these messages, and after the local multicast group is formed (if requested), they send out a "*multicast tree ready signal*" to the source. Mid-level and leaf gateways perform similarly to the first level-gateways. Figure 5-15 shows the top-level tree formation actions for the outlined scenario.

After the top-level tree is formed, the source initiates the actual multicast data transmission to the first-level gateways. The source listens for the retransmission requests from the first-level gateways and replies if there are any. When the data streaming is finished it waits for the "*multicast complete*" signal from the first-layer

gateways, and upon receiving them from every child gateway it sends a "*multicast completed*" signal to the user.

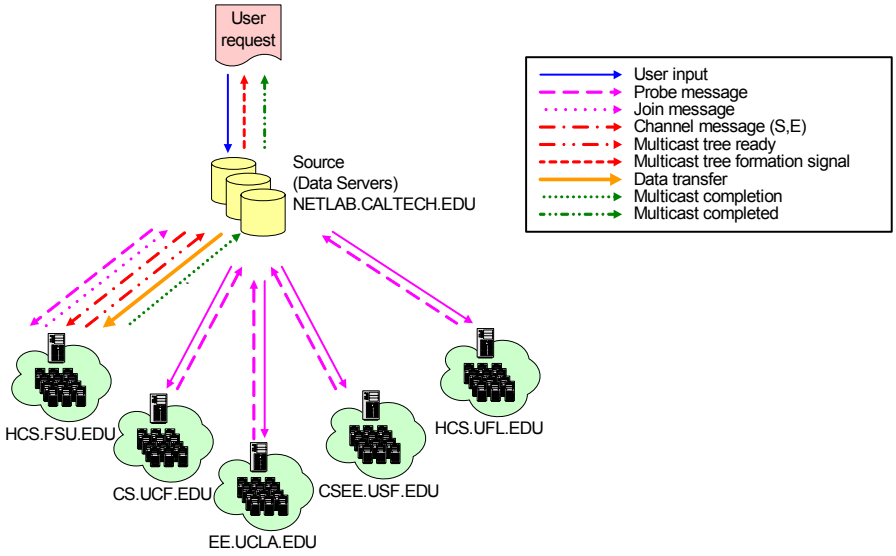


Figure 5-15. Top-level tree formation.

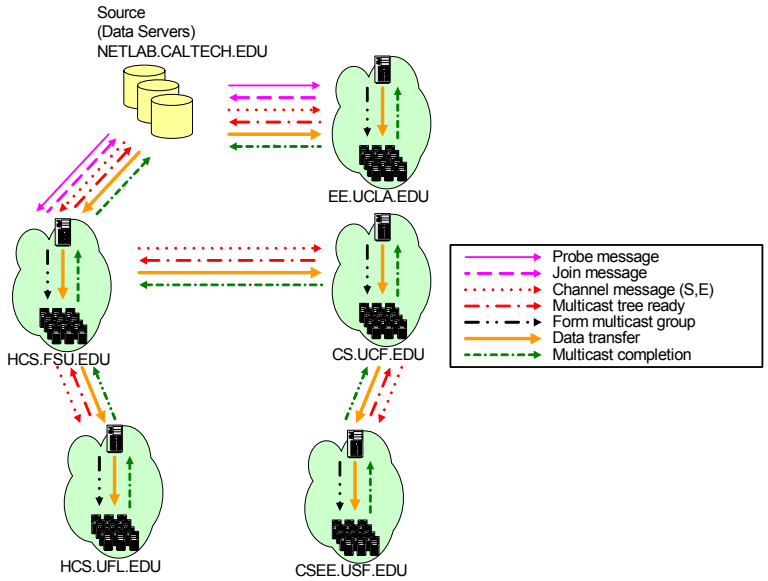


Figure 5-16. Multicast communication.

In the actual multicast data transfer phase, all gateways receive incoming multicast data from upper-level gateways. They immediately save a copy of the incoming data to

their local caches. If the incoming data is corrupted, they ask for a retransmission from upper-level gateways. Upon receipt of uncorrupted data, they simultaneously propagate the data to their child gateways and local groups. They listen and reply to any retransmission request from their own multicast group or child gateways and wait for the "*multicast completion*" signal from them. Upon receiving the completion signals, first-level gateways send "*multicast completion*" signals to the source. Figure 5-16 shows the multicast communication for the outlined scenario.

Figure 5-17 shows three network topology scenarios evaluated in this case study. Figure 5-17A shows the flat topology where all receiver domains are connected to the source with hypothetical direct links that are one-hop away for the six WAN-connected domains. Figures 5-17B and 5-17C show the 2-hop and 3-hop tree architectures, respectively. Figure 5-18 shows a snapshot of the simulation model.
Figure 5-19 shows the obtained multicast completion latencies for a Terabyte multicast file transfer over the WAN backbone. *Tree (L.C)* represents a tree architecture with local caching for retransmission, and *Tree (N.C)* represents a tree architecture with no local caching. Figures 5-19A and 5-19B show the case where the gateway is a dedicated node. Figures 5-19C and 5-19D show the case where the gateway is part of the local SAN clusters.

As can be seen from the figure, a flat topology achieves lowest completion latency at the cost of increased backbone impact. Although the flat topology shows good "*theoretical*" results, it is not practical, as all-to-all direct connectivity is not possible for most cases in real-world WAN networks. Moreover, with the flat topology there is a

possible congestion problem, as the root has to issue more transfers and most likely these transfers will share the same physical links to some degree.

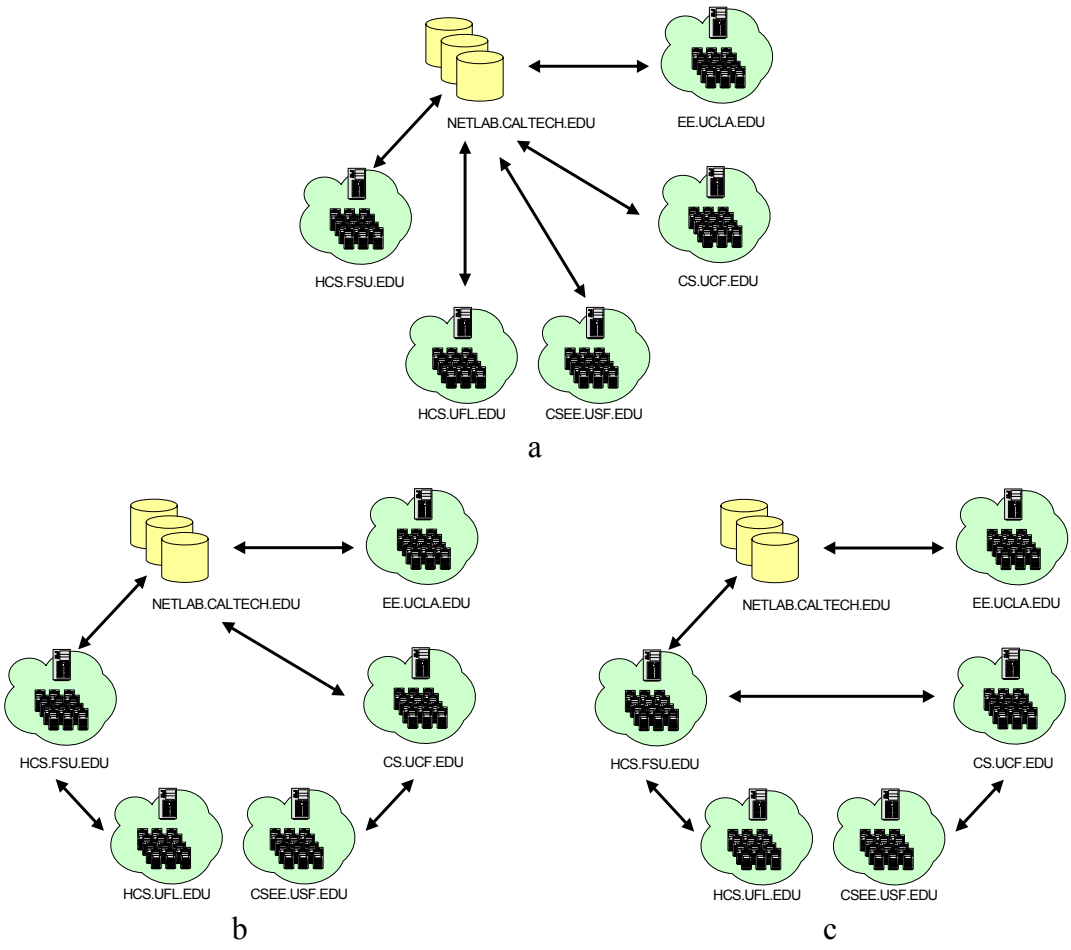


Figure 5-17. Evaluated network topology scenarios. A) Flat topology. B) Tree architecture for 2-hop topology. C) Tree architecture for 3-hop topology.

An alternative to the flat topology is the tree architecture. Previous chapters presented comparative multicast performance of various flat topology (i.e., separate addressing) and tree architecture (i.e., U-torus, M_d-torus, M_u-torus, binomial and binary trees) algorithms for SANs. Following this approach, the performance of tree-based multicast architectures is analyzed in comparison to flat topology for WANs. It is observed that the regular tree architecture (i.e., Tree (N.C)) is not a viable alternative to

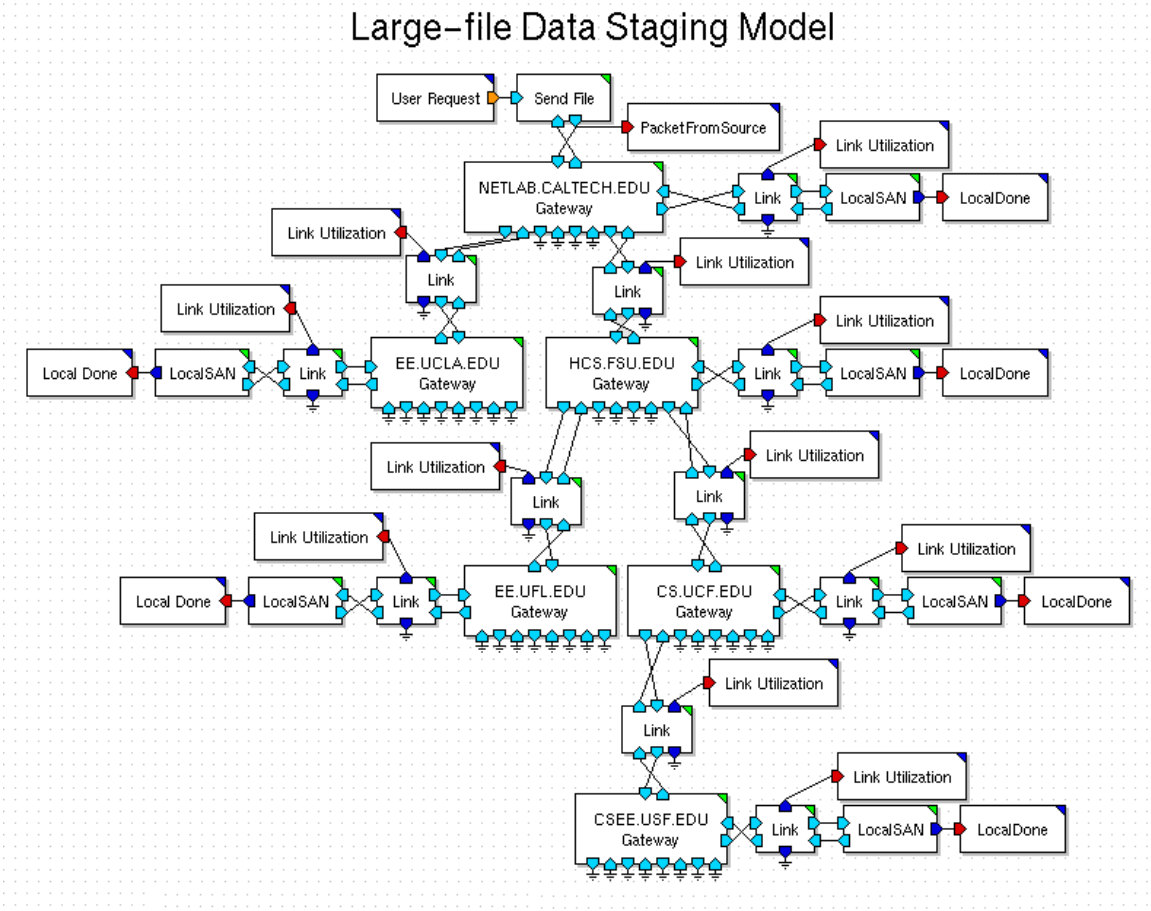
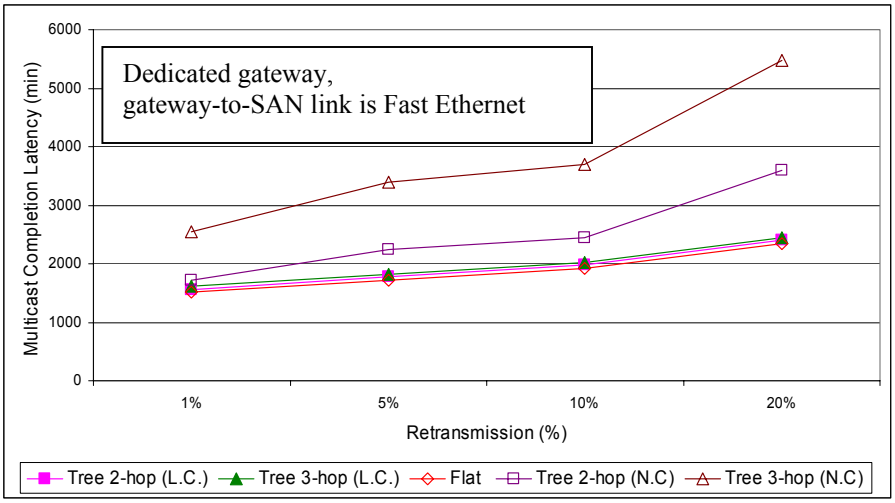


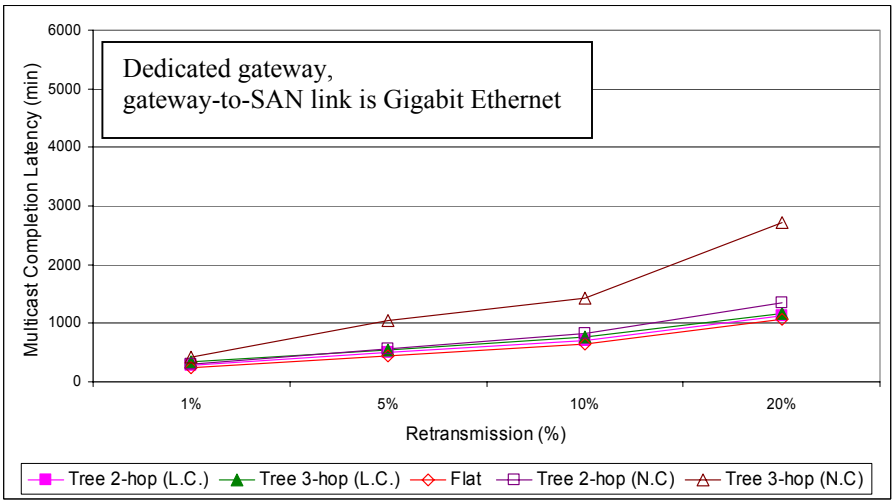
Figure 5-18. Snapshot of simulative model.

the flat topology because of the extensive completion latencies, as increasing the number of hops and the retransmission rate also increase the completion latency. By contrast, an enhanced version of the tree architecture that is proposed in the framework, Tree (L.C), provides similar performance levels to the flat topology. Furthermore, Tree (L.C) is not affected by increasing the number of hops and the retransmission rate, and it provides a lower WAN backbone impact.

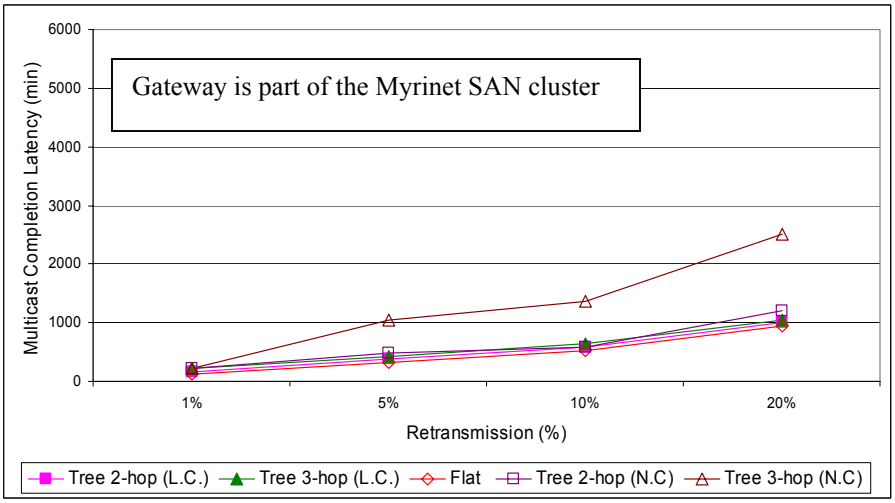
Figure 5-19 also shows useful information about the gateway placement problem. As can be seen from the figure, when the gateway is not part of the SAN cluster, and the gateway-to-SAN link is a slow interconnect (e.g., Fast Ethernet), this slow link becomes bottleneck in addition to the slow WAN links. However, if the gateway-to-SAN link is a



a

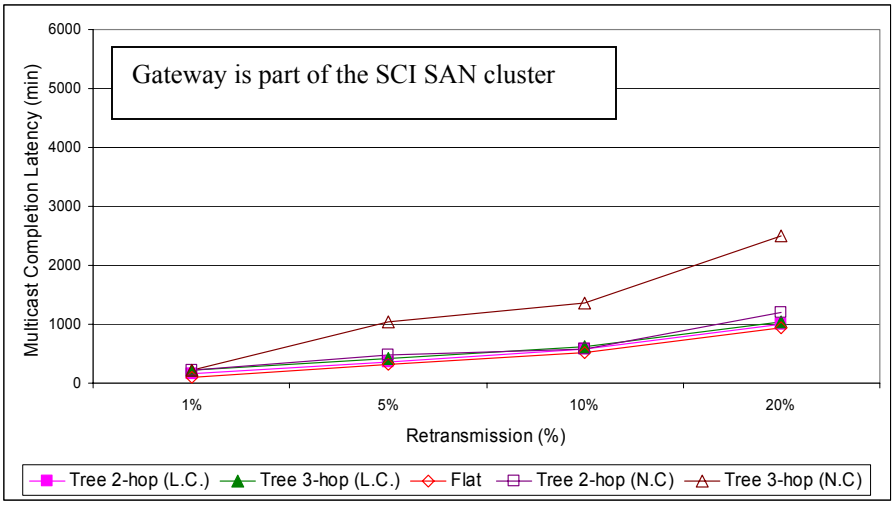


b



c

Figure 5-19. Continued.



d

Figure 5-19. Multicast completion latencies for the large-file transfer scenario. A) Gateway is a dedicated node and gateway-to-SAN interconnect is Fast Ethernet. B) Gateway is a dedicated node and gateway-to-SAN interconnect is Gigabit Ethernet. C) Gateway is part of the Myrinet SAN cluster. D) Gateway is part of the SCI SAN cluster.

fast one, such as Gigabit Ethernet, this problem can be eliminated to some degree, and for a small number of the hops Tree (N.C) provides good performance results. When the gateway is part of the SAN cluster, it is observed that the SAN links provide adequate link capacity for data transfers, and the only system bottleneck is the WAN link latencies.

Figure 5-20 shows a comparative analysis of the backbone impact of Tree (L.C) and Tree (N.C) architectures. The measurements are obtained from a 3-hop system configuration. The backbone impact is measured as the amount of additional traffic on the first hop of the WAN link backbone from the source. The baseline is defined as the minimum amount of traffic (0% retransmission case) required to transfer data.

As can be seen, the Tree (L.C) is constant and unaffected by increasing retransmission rates. This observation parallels with the results of the previous case. Moreover, Tree (L.C) provides the lowest WAN backbone impact for all retransmission rates. By contrast, the Tree (N.C) architecture is directly affected by increasing

retransmission rates and therefore places an increased impact on the source and the 1-hop WAN link. This difference is because of the route lengths that the retransmission requests and the actual retransmissions have to propagate over the WAN backbones in these two tree architectures.

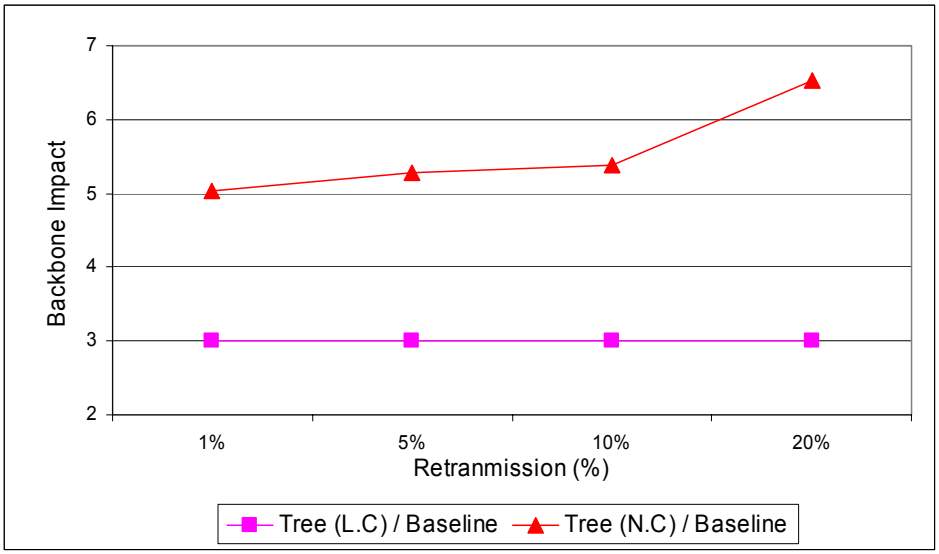


Figure 5-20. Comparative WAN backbone impacts of Tree (L.C) and Tree (N.C) architectures

Figure 5-21 shows a head-to-head multicast completion latency comparison of the Tree (L.C) and Tree (N.C) architectures with the four different interconnect technologies for a 3-hop topology with 5% retransmission rate. Tree (L.C) performs better than the Tree (N.C) for all interconnects. Explicitly, Tree (L.C) performs ~40%, ~50%, and ~60% better compared to the Tree (N.C) algorithm, for Fast Ethernet (FE), Gigabit Ethernet (GigE), and Myrinet and SCI, respectively. Among these four, Fast Ethernet is the worst choice as a gateway-to-SAN link. Also, Figure 5-21 reveals that the WAN backbones are the primary bottleneck as they can not supply enough data to saturate SCI and Myrinet links. For the Tree (N.C), when the gateway-to-SAN link is chosen to be a Fast Ethernet interconnect an additional bottleneck is observed, whereas for Tree (L.C)

both the Fast Ethernet and Gigabit Ethernet act as bottlenecks in addition to the WAN backbone links. Moreover, for Tree (L.C), more efficient multicast data staging is obtained when the gateway is part of the SAN cluster.

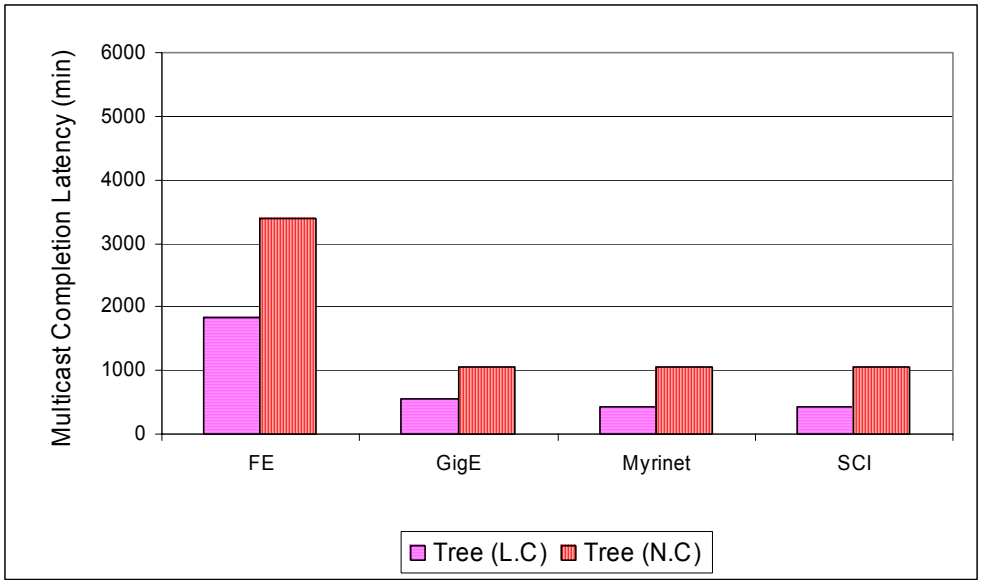
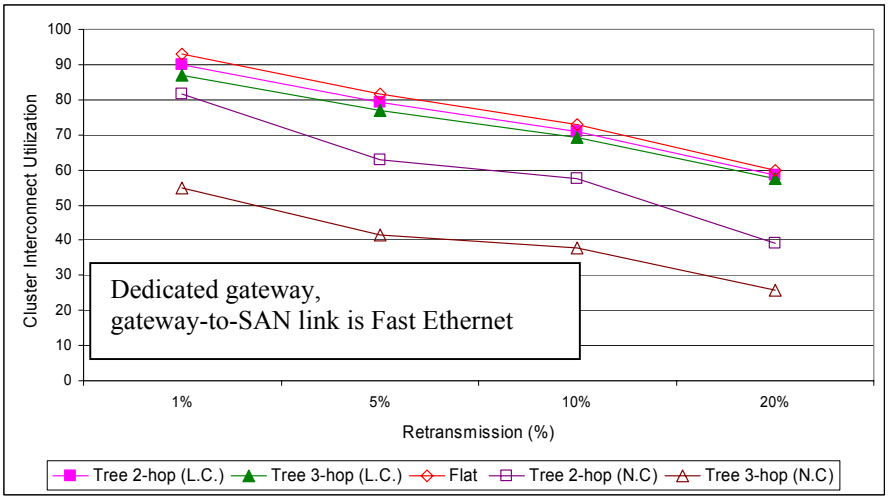


Figure 5-21. Head-to-head comparison of Fast Ethernet (FE), Gigabit Ethernet (GigE), Myrinet, and SCI interconnects as gateway-to-SAN links for Tree (L.C) and Tree (N.C) architectures

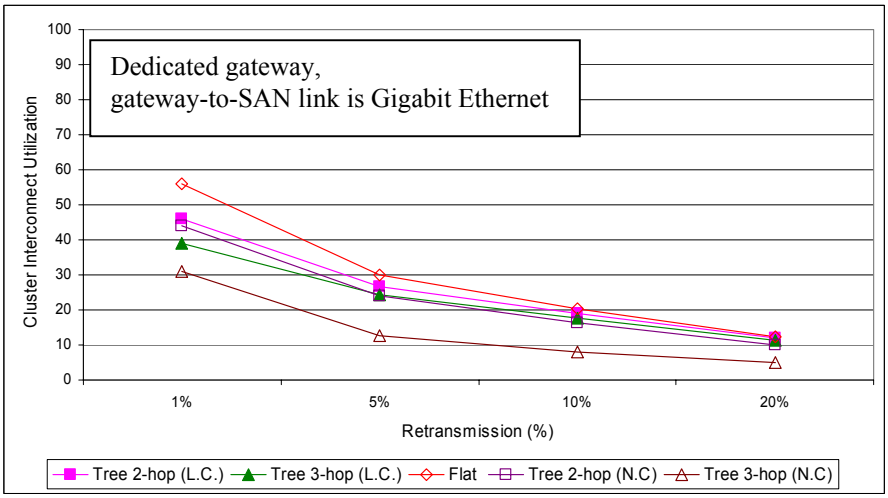
Figure 5-22 shows the local domain cluster interconnect utilizations. Figures 5-22A and 5-22B show the case where the gateway is a dedicated node. Figures 5-22C and 5-22D show the case where the gateway is part of the local SAN cluster. As can be seen from Figure 5-22A, where the gateway-to-SAN is connected with a slow interconnect such as Fast Ethernet, Tree (N.C) is less utilized than Tree (L.C). The Tree (N.C) pushes the utilization bottleneck from the local domain to the WAN backbone, and increasing the number of WAN backbone hops increases the WAN bottleneck problem. Moreover, increasing retransmission rates amplifies the problem, and the gateway-to-SAN Fast Ethernet link is underutilized. Overall, using a slow interconnect such as Fast Ethernet as the gateway-to-SAN link results in decreased throughput. Tree (L.C), with Fast Ethernet

as the gateway-to-SAN link is highly utilized, as the bottleneck is shared between the WAN and the local gateway-to-SAN link. Increasing rates pushes the bottleneck to the WAN links and the gateway-to-SAN link is less utilized. The flat results exhibit similar performance levels to those of the Tree (L.C) scheme because the data and retransmissions only travel one hop over the WAN backbones for both Tree (L.C) and flat architectures. Overall, Tree (L.C) provides faster data transfers to the local multicast group because of its better local link utilization. For the case where the gateway-to-SAN connection is a Gigabit Ethernet link, as given in Figure 5-22B, lower utilization is obtained compared to the Fast Ethernet case as the link capacity of Gigabit Ethernet is higher compared to Fast Ethernet. For the Gigabit Ethernet case, a reduced backbone impact because of the Tree (N.C) approach is observed as Tree (N.C) exhibits a similar level of performance to Tree (L.C). Finally, as can be seen, increasing retransmission rates decreases utilization. Figures 5-22C and 5-22D show the results for gateways that are part of the Myrinet or SCI SAN clusters. For both cases, the SAN links are highly underutilized for all trials because the system bottleneck is the WAN backbone. It can be concluded that, with today's backbone technology, the SAN links cannot be saturated by the amount of data the WAN links can provide.

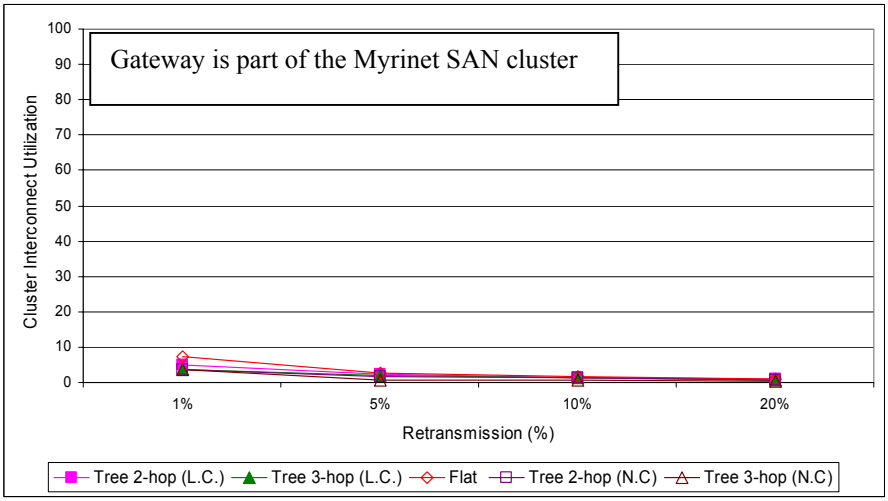
Case study 2 shows that, in general, the WAN links are the key system bottlenecks. For some cases the gateway-to-SAN interconnect can also be an additional bottleneck (i.e., Fast Ethernet as a gateway-to-SAN link). Lower data retransmission rates and higher QoS priority in the WAN links can help to reduce the WAN bottleneck. Moreover, using faster interconnects as the gateway-to-SAN link and placing the gateway as part of the SAN cluster also helps to reduce the bottleneck.



a

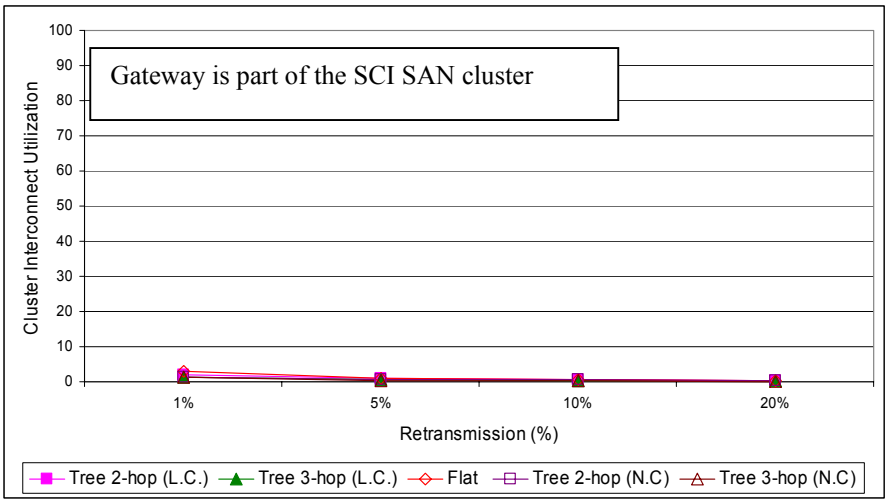


b



c

Figure 5-22. Continued



d

Figure 5-22. Interconnect utilizations for local clusters. A) Gateway is a dedicated node and gateway-to-SAN interconnect is Fast Ethernet. B) Gateway is a dedicated node and gateway-to-SAN interconnect is Gigabit Ethernet. C) Gateway is part of the Myrinet SAN cluster. D) Gateway is part of the SCI SAN cluster.

Summary

This chapter introduced a Layer-2/Layer-3 framework for cluster-to-cluster multicast over a WAN backbone. The proposed infrastructure supports multi-protocol, latency-sensitive multicast with minimal WAN backbone impact. The framework takes advantage of faster interconnects whenever possible, and as it requires a minimal amount of changes in the existing Grid/Globus architecture, it can be implemented easily. Two separate simulative case studies are also presented for further performance analysis of the proposed approach.

Simulation case studies show that the WAN backbone latency is the most dominant component of the multicast communication for Grid-connected clusters. The flat topology for Grids is not practical and introduces high backbone utilization levels. The proposed tree architecture with local caching provides lower latencies and reduces backbone impact compared to flat topologies. Moreover, the tree architecture with local

caching is unaffected by increasing the retransmission rate or the number of hops. The gateway-to-SAN link is also observed to be an additional bottleneck for some cases. For example, when a slow interconnect, such as Fast Ethernet, is used as a gateway-to-SAN link, it increases multicast completion latencies, and the WAN backbone utilization. Therefore, special attention is required when determining the placement of the gateway.

CHAPTER 6
CONCLUSIONS

Cluster computing is a cost-effective solution for computationally intensive problems, providing performance similar to that of supercomputers. Grid computing is the answer of the parallel and distributed processing community to many of the computationally demanding scientific and engineering problems of the world, bringing together geographically distributed computing resources, such as supercomputers and clusters, on the global-scale. Collective communication operations, such as multicasting, simplify and increase the functionality and efficiency of parallel and distributed tasks for Grid computing. Unfortunately, specialized high-performance cluster interconnects do not inherently support multicasting. Furthermore, current Grid multicasting schemes are unable to support any other parallel and distributed computing platform other than MPI, and they are not targeted for latency-sensitive applications.

This research investigates the multicast problem for Grid-connected SAN-based and IP-based clusters. Following a bottom-to-top approach, the research is divided into three distinct phases. The first phase of this research is focused on the experimental analysis, small-message latency modeling, and analytical projections of software-based Layer-2 multicast algorithms on high-performance torus networks. The second phase of this research investigates the multicast problem for interconnects with onboard NIC co-processors. Experimental analysis, small-message latency models, and analytical projections are performed for networks with onboard NIC coprocessors. Finally, a

universal, latency-sensitive multicast framework for Grid-connected clusters is introduced and evaluated based on extensions to the results obtained in first two phases.

In the first phase of this study, five different multicast algorithms for high-performance torus networks are evaluated. Direct torus networks are widely used in high-performance parallel computing systems as they are cost-effective and also provide good scalability in terms of bandwidth. The selected algorithms are analyzed on direct SCI torus networks. The performance characteristics of these algorithms are experimentally examined under different system and network configurations using various metrics, such as multicast completion latency, root-node CPU utilization, multicast tree creation latency, and link concentration and concurrency, to evaluate their key strengths and weaknesses. Based on the results obtained, small-message latency models for each algorithm are defined. The models are observed to be accurate. Projections for larger systems are also presented and evaluated.

It is observed that for SCI torus networks with small messages and multicast group sizes, the best approach, in terms of completion latency and host CPU utilization, is the separate addressing algorithm. However, for large message sizes and large multicast group sizes, more complex algorithms perform better. The M_d-torus algorithm performs better for larger system sizes because of its balanced dimensional partitioning method, providing the lowest completion latency and very low user-level CPU overhead. Moreover, for higher dimension torus networks (such as 3D or more), the M_d-torus algorithm appears to be the best performing protocol. In short, there is no definite multicast algorithm which best suits every possible networking scenario. Although SCI

hardware does not inherently support user-level multicasting, it can be achieved with reasonable performance levels for this high-performance interconnect.

The second phase of this research introduces a multicast performance analysis and modeling for high-speed indirect networks with NIC-based processors. This type of interconnect finds a wide implementation area in the parallel computing community because of its reprogramming flexibility and work offloading from the host CPU. Various degrees of host and NIC processor work sharing are evaluated in this phase of study, such as host-based, NIC-based, and NIC-assisted communication schemes. The selected schemes are experimentally analyzed for binomial and binary trees, serial forwarding and separate addressing multicast algorithms, using various metrics, such as multicast completion latency, root-node CPU utilization, multicast tree creation latency, and link concentration and concurrency, to evaluate their key strengths and weaknesses. Small-message latency models of these algorithms are developed and verified based on the experimental results. The models are observed to be accurate. Projections for larger systems are also presented and evaluated.

Experimental and latency modeling analysis revealed that for interconnects with onboard NIC coprocessors and for latency-sensitive applications that utilize small-messages, a host-based multicast communication scheme performs best. However, host-based multicasting results in the highest CPU utilization. NIC-based solutions provide the lowest and constant CPU utilizations for both small and large messages at the cost of increased completion latencies. NIC-assisted multicasting provides lower CPU utilizations than host-based ones, and comparable CPU-utilizations to the NIC-based algorithms. Furthermore, the NIC-assisted approach provides comparable multicast

completion latencies to host-based schemes for lower host CPU utilizations, and thus appears to be better choice for applications that demand a high level of computation-communication overlapping.

The third phase of this research tackles the problem of obtaining a universal, low-level multicast system for latency-sensitive applications for Grid computing. A framework for low-level multicast communication is proposed that can be used as a service for high-level Grid computing applications. The proposed framework introduces a Layer-2/Layer-3 framework for cluster-to-cluster multicast over a WAN backbone and uses and extends the results of previous two phases for improved overall multicast performance and efficiency. The infrastructure supports multi-protocol multicast with minimal WAN backbone impact. The framework takes advantage of faster interconnects, such as SANs, whenever possible. It also requires a minimal amount of changes in the existing Grid/Globus architecture. Two simulative case studies are performed for performance analysis of the proposed approach.

For Grid-connected clusters the WAN backbone latency is the most dominant component for a given multicast system. Simulation results show that the tree-based architecture with local caching provides lower latencies and lower backbone impact for all cases compared to a flat architecture and is unaffected by the increasing retransmission rate and number of hops, unlike the flat architecture. It is also observed that the gateway-to-SAN interconnect can also be an additional bottleneck and the capacity of the gateway-to-SAN link determines gateway placement. The remote completion latency dominates for small local and remote group sizes and significantly large local groups will likely hide the remote multicast latency.

This study is the first to implement and experimentally analyze and model user-level software-based multicast performance on direct SCI networks. Moreover, it is the first to present a full-scale comparative experimental analysis, latency modeling, and analytical projections of host-based, NIC-based, and NIC-assisted multicast communication schemes over various spanning tree and path-based algorithms for Myrinet networks. This research also is the first in presenting an experimental comparative multicast performance analysis of SCI and Myrinet interconnects. Furthermore, this research is the first to define a latency-sensitive universal multicast framework for Grid-connected SAN-based and IP-based clusters that can be used as a low-level service for higher-level applications. The proposed multicast framework is unique in the sense that it introduces a channel-based and group-based hybrid approach for obtaining a distributed and scalable multicast communication system over the WAN backbones while maintaining the efficiency and scalability in the local multicast groups.

LIST OF REFERENCES

1. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications*, Vol. 15, No.3, 2001.

2. M Barnett, 2003, *ATLAS experiment home page*, CERN, http://atlasexperiment.org/, Aug. 2003.

3. A.R. Whitney, K.A. Dudevoir, H.F. Hinteregger, J.P. Gary, B. Fink, L.N. Foster, C. Kodak, K. Kranacs, P. Lang, W.T. Wildes, S.L. Bernstein, L.A. Prior, P.A. Schulz, T.W. Lehman, and J. Sobieski, "The Gbps e-VLBI Demonstration Project". ftp://web.haystack.edu/pub/e-vlbi/demo_report.pdf.

4. P. Avery, and I. Foster, 2003, *iVDgL –International Virtual Data Grid Laboratory home page*, iVDgL, http://www.ivdgl.org, Jan. 2003.

5. P. Avery, and I. Foster, 2003, *GriPhyN - Grid Physics Network home page*, GriPhyN, http://www.griphyn.org, Jan. 2003.

6. R. Mondardini, 2003, *DataTAG project home page*, CERN, http://datatag.web.cern.ch/datatag, Jan. 2003.

7. T. DeFanti, 2003*, StarLight project home page*, University of Illinois at Chicago, http://www.startap.net/starlight, Jan. 2003.

8. M. Maimour, and C. Pham, "An Active Reliable Multicast Framework for the Grids," *Proceedings of the International Conference on Computational Science (ICCS 2002)*, Amsterdam, The Nederlands, pp588-597, Apr. 2002.

9. H. Eriksson, "The multicast backbone," *Communications of the ACM*, Vol. 8, pp. 54-60, 1994.

10. T. Bates, R. Chandra, D. Katz, and Y. Rekhter, "Multiprotocol extensions for BGP-4," *Internet Engineering Task Force (IETF) specification, RFC 2283*, Feb. 1998.

11. D. Estrin, D. Fariannaci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol independent multicast sparse-mode (PIM-SM): Protocol specification," *Internet Engineering Task Force (IETF) specification, RFC 2362*, Jun. 1998.

12. D. Farinacci, Y. Rekhter, P. Lothberg, H. Kilmer, and J. Hall, "Multicast source discovery protocol," *Internet Engineering Task Force (IETF) specification, RFC 0020*, Jun. 1998.

13. P.K. McKinley, H.Xu, A.H. Esfahanian, and L.M. Ni, "Unicast-Based Multicast Communication in Wormhole-Routed Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 12, pp. 1252-1265, 1994.

14. P.K. McKinley, Y.Tsai, and D.F. Robinson, "Collective Communication in Wormhole-Routed Massively Parallel Computers," *IEEE Computer*, Vol. 28, No. 2, pp. 39-50, 1995.

15. Y. Tseng, D.K. Panda, and T Lai, "A Trip-Based Multicasting Model in Wormhole-Routed Networks with Virtual Channels," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, No.2, pp. 138-150, 1996.

16. R. Kesavan and D.K. Panda, "Multicasting on Switch-Based Irregular Networks using Multi-Drop Path-Based Multi-Destination Worms," *Proceedings of Parallel Computer Routing and Communication*, *Second International Workshop (PCRCW'97)*, Atlanta, Georgia, pp. 179-192, Jun. 1997.

17. D. Gustavson and Q. Li, "The Scalable Coherent Interface (SCI)," *IEEE Communications*, Vol. 34, No. 8, pp. 52-63, Aug. 1996.

18. 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, Vol. 15, No. 1, pp. 29-36, Feb. 1995.

19. IEEE, "SCI: Scalable Coherent Interface," *IEEE Approved Standard 1596-1992*, 1992.

20. D.F. Robinson, P.K. McKinley, and B.H.C. Cheng, "Optimal Multicast Communication in Wormhole-Routed Torus Networks," *IEEE Transactions on Parallel and Distributed Systems,* Vol. 6, No. 10, pp. 1029-1042, 1995.

21. X. Lin and L.M. Ni, "Deadlock-Free Multicast Wormhole Routing in Multicomputer Networks," *Proc. of 18th Annual International Symposium on Computer Architecture*, Toronto, Canada, pp. 116-124, May 1991.

22. D.F. Robinson, P.K. McKinley, and B.H.C. Cheng, "Path-Based Multicast Communication in Wormhole-Routed Unidirectional Torus Networks," *J. of Parallel and Distributed Computing*, Vol. 45, No. 2, pp. 104-121, 1997.

23. L.M. Ni and P.K. McKinley, A Survey of Wormhole Routing Techniques In Direct Networks, *IEEE Computer*, Vol. 26, No. 2, pp. 62-76, 1993.

24. K. Omang and B. Parady, "Performance of Low-Cost Ultrasparc Multiprocessors Connected By SCI," *Technical Report, Department of Informatics*, University of Oslo, Norway, 1996.

25. M. Ibel, K.E. Schauser, C.J. Scheiman and M. Weis, "High-Performance Cluster Computing using SCI," *Proceedings of Hot Interconnects Symposium V*, Palo Alto, CA, Aug. 1997.

26. M. Sarwar and A. George, "Simulative Performance Analysis of Distributed Switching Fabrics for SCI-Based Systems," *Microprocessors and Microsystems*, Vol.24, No.1, pp. 1-11, 2000

27. D. Gonzalez, A. George, and M. Chidester, "Performance Modeling and Evaluation of Topologies for Low-Latency SCI Systems," *Microprocessor and Microsystems*, Vol.25, No.7, pp. 343-356, 2001

28. R. Todd, M. Chidester, and A. George, "Comparative Performance Analysis of Directed Flow Control for Real-Time SCI," *Computer Networks*, Vol.37, No.4, pp. 391-406, 2001

29. H. Bugge, "Affordable Scalability using Multicubes," in: H. Hellwagner, A. Reinfeld (Eds.), *SCI: Scalable Coherent Interface*, LNCS State-of-the-Art Survey, Springer, Berlin, Germany, 1999, pp. 167-174

30. L.P. Huse, "Collective Communication on Dedicated Clusters Of Workstations," *Proc. of 6th PVM/MPI European Users Meeting (EuroPVM/MPI '99)*, Sabadell, Barcelona, Spain, Sep. 1999, pp. 469-476.

31. H. Wang, and D.M. Blough, "Tree-Based Multicast in Wormhole-Routed Torus Networks," *Proc. PDPTA'98*, 1998.

32. D.E. Culler, R. Karp, D.A. Patterson, A. Sahay, K.E. Shauser, E. Santos, R. Subramonian, and T. von Eicken, "LogP: Towards a Realistic Model of Parallel Computation," *Proceedings of ACM 4th SIGPLAN Symposium on Principles and Practices of Parallel Programming*, San Diego, California, pp. 1-12, May 1993.

33. E. Deelman, A. Dube, A. Hoisie, Y. Luo, R. Oliver, D. Sundaram-Stukel, H. Wasserman, V.S. Adve, R. Bagrodia, J.C. Browne, E. Houstis, O. Lubeck, J. Rice, P. Teller, M.K. Vernon, "POEMS: End-to-end Performance Design of Large Parallel Adaptive Computational Systems," *Proceedings of First International Workshop on Software and Performance '98, WOSP '98*, Santa Fe, New Mexico, pp. 18-30, Oct. 1998.

34. M. Gerla, P. Palnati, and S. Walton, "Multicasting Protocols for High-Speed, Wormhole-Routing Local Area Networks," *Proceedings of SIGCOMM '96 Symposium*, pp. 184-193, Aug. 1996.

35. K. Verstoep, K. Landgendoen, and H. Bal, "Efficient Reliable Multicast on Myrinet," *Proceedings of 1996 International Conference on Parallel Processing*, pp. 156-165, Aug. 1996.

36. P. Kesavan and D.K. Panda, "Optimal Multicast with Packetization and Network Interface Support," *Proceedings of 1997 International Conference on Parallel Processing*, pp. 370-377, Aug. 1997.

37. R.A.F. Bhoedjang, T. Ruhl, and H.E. Bal, "Efficient Multicast on Myrinet Using Link-Level Flow Control," *Proceedings of 1998 International Conference on Parallel Processing*, pp. 381-389, Aug. 1998.

38. D. Buntinas, D. K. Panda, and P. Sadayappan, "Fast NIC-Based Barrier over Myrinet/GM," *International Parallel and Distributed Processing Symposium IPDPS'01*, pp. 52-60, San Francisco, CA, Apr. 2001.

39. D. Buntinas, D. K. Panda, J. Duato, and P. Sadayappan, "Broadcast/Multicast over Myrinet using NIC-Assisted Multidestination Messages," *Proceedings of Fourth International Workshop on Communication, Architecture, and Applications for Network-Based Parallel Computing, CANPC '00*, Toulouse, France, pp. 115-129, Jan. 2000.

40. 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, Feb. 1998.

41. D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, F. Berry, A. Merritt, E. Gronke, and C. Dodd, "The Virtual Interface Architecture," *IEEE Micro*, pp. 66-76, Mar./Apr. 1998.

42. C. Kurmann and T.M. Stricker, "A Comparison of two Gigabit SAN/LAN technologies: Scalable Coherent Interface versus Myrinet," *Scalable Coherent Interface: Technology and Applications*, Edited by Hermann Hellwagner & Alexander Reinefeld, pp. 29-42, Cheshire Henbury Publications, 1998.

43. M. Fischer, U. Brüning, J. Kluge, L. Rzymianowicz, P. Schulz, and M. Waack, "ATOLL a new switched, high-speed Interconnect in Comparison to Myrinet and SCI," *IPDPS 2000 (Int. Parallel and Distributed Processing Symposium)*, *PC NOW Workshop*, Cancun, Mexico, 2000.

44. I. Foster, and C. Kesselman, 2003, *Globus project home page*, Globus, http://www.globus.org, May 2003.

45. M. Livny, and M. Solomon, 2003, *Condor project home page*, University of Wisconsin, http://www.cs.wisc.edu/condor, May 2003.

46. D. Waitzman, C. Patridge, and S. Deering, "Distance vector multicast routing protocol (DVMRP)," *Internet Engineering Task Force (IETF), RFC 1075*, Nov. 1988.

47. Y. Rekhter and T. Li, "a border gateway protocol (BGP-4)," Internet Engineering Task Force (IETF), RFC 1771, Mar. 1995.

48. H. Holbrook and D. Cheriton, "IP multicast channels: EXPRESS support for large-scale single-source applications," *ACM SIGCOMM*, Cambridge, MA, Aug. 1999.

49. B.B. Lowekamp, A Beguelin, "ECO: Efficient Collective Operations for Communication on Heterogeneous Networks," *International Parallel Processing Symposium*, pp. 399-405, Honolulu, HI, 1996.

50. I. Foster, J. Geisler, W. Gropp, N. Karonis, E. Lusk, G. Thiruvathukal, and S. Tuecke, "Wide-Area Implementation of the Message Passing Interface," *Parallel Computing*, Vol. 24, No. 12, pp. 1735-1749, 1998.

51. N. Karonis, B. Toonen, and I. Foster, "MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface," *Journal of Parallel and Distributed Computing (JPDC),* Vol. 63, No. 5, pp. 551-563, May 2003.

52. G. Schorcht, I. Troxel, K. Farhangian, P. Unger, D. Zinn, C. Mick, A. George, and H. Salzwedel, "System-Level Simulation Modeling with MLDesigner," *Proc. 11th IEEE/ACM International Symposium On Computer and Telecommunication Systems*, Orlando, FL, 2003.

53. K. Shanmungen, V. Frost, and W. LaRue, "A Block-Oriented Network Simulator (BONeS)," *Simulation*, Vol. 58, No.2, pp. 83-94, 1992.

54. D. Culler, K. Keeton, L.T. Liu, A. Mainwaring, R. Martin, S. Rodrigues, K. Wright, and C. Yoshikawa, "The Generic Active Message Interface Specification," white paper, 1994.

55. GridFTP: Universal Data Transfer for the Grid, white paper, http://www.globus.org/datagrid/deliverables/C2WPdraft3.pdf.

BIOGRAPHICAL SKETCH

Mr. Hakki Sarp Oral received the B.S. degree in Electrical and Electronics Engineering from Istanbul Technical University, Turkey; and the M.S. degree in Electrical and Electronics Engineering from the Cukurova University, Turkey.

He is presently a Graduate Assistant in the Department of Electrical and Computer Engineering at the University of Florida and a group leader in the High-performance Computing and Simulation (HCS) Research Laboratory.