Multicast Virtual Topologies for Collective Communication in MPCs and ATM Clusters

*Y. Huang, C. C. Huang, and P. K. McKinley**

Department of Computer Science
Michigan State University
East Lansing, Michigan 48824
{huangyih,huangch,mckinley}@cps.msu.edu

Abstract

This paper defines and describes the properties of a multicast virtual topology, the **M-array**, and a resource-efficient variation, the **REM-array**. It is shown how several collective operations can be implemented efficiently using these virtual topologies, while maintaining low complexity. Because the methods are applicable to any parallel computing environment that supports multicast communication in hardware, they provide a framework for collective communication libraries that are portable and yet take advantage of such low-level hardware functionality. In particular, the paper describes the practical issues of using these methods in wormhole-routed massively parallel computers (MPCs) and in workstation clusters connected by Asynchronous Transfer Mode (ATM) networks. Performance results are given for both environments.

* This work was supported in part by DOE grant DE-FG02-93ER25167, and by NSF grants MIP-9204066, CDA-9222901, and CCR-9503838.

1 Introduction

The supercomputer market now includes many parallel architectures in which memory is physically distributed among processing nodes. Commercial **massively parallel computers** (MPCs) include both message passing systems, such as the Intel Paragon and TMC CM-5, as well as systems that support distributed shared memory, such as the Cray T3D. Many such systems use a cut-through switching technique, **wormhole-routing** [1], in which data is pipelined through routers in the network. An alternative platform for parallel scientific computing is the **cluster** [2], in which collections of off-the-shelf systems are connected to form a parallel computer. Many clusters are connected by high-speed switches, which provide large aggregate capacity compared to shared media. While distributed-memory systems may be programmed in a variety of ways, inevitably, nodes must communicate by sending messages through a network. Communication operations may be either **point-to-point**, which involve a single source and a single destination, or **collective**, in which more than two processes participate. Collective operations are usually defined in terms of a process group and include broadcast, scatter, gather, global reduction and scan, total exchange, and barrier synchronization.

Our research addresses the design of collective communication operations for parallel and distributed computing. Approaches to this problem range from the implementation of special-purpose hardware for individual collective operations, to the use of so-called **unicast-based** collective algorithms [3], which are implemented purely in software atop send and receive primitives. We focus on an approach that lies between these two ends of the spectrum, in which collective algorithms are designed to exploit "generic" hardware features, that is, hardware features that may be used in many collective operations as well as in other applications. In this paper, we address the problem of designing collective communication operations in systems that provide hardware support for one-to-many, or **multicast**, communication. Multicast is perhaps the most fundamental collective operation, in which a source node must deliver copies of a single message to each node in a specified set of destinations. A special case of multicast is **broadcast**, in which the destination set contains every node in the network.

Several parallel processing platforms already support multicast in hardware. Examples include both wormhole-routed MPCs, such as the nCUBE-2 [4] and CM-5 [5], and switch-based LANs, such as the Fore

Systems ASX100 ATM switch [6], which can be used to interconnect workstation clusters. For such parallel computing environments, collections of nodes that communicate across hardware-supported multicast channels can be viewed logically as a directed hypergraph. A **hypergraph** is a generalization of a graph in which each edge may be associated with an arbitrary subset of vertices, rather than exactly two vertices as in a graph. This paper proposes two multicast (or hyper-) topologies that can support several collective communication operations. The suggested topologies are **virtual**, in that they may be mapped onto existing physical architectures. Most importantly, these topologies provide a framework for the development of collective communication libraries that are portable across different parallel computing platforms.

The remainder of the paper is organized as follows. In Section 2, we review the methods in which hardware multicast can be implemented in both wormhole-routed MPCs and in clusters. Section 3 defines and describes the properties of two particular multicast topologies, the **M-array** and **REM-array**, which will be used throughout the paper. In Section 4, we show how several collective operations can be implemented efficiently on an M-array topology. Section 5 and Section 6, respectively, address the practical issues of using the M-array in wormhole-routed systems and in switch-based clusters; performance results are included in each section. Section 7 briefly discusses related work, and Section 8 presents our conclusions and discusses possible future areas of investigation.

2 Background

2.1 Collective Communication Operations

A collective operation is usually defined in terms of a group of processes. Collective operations may be used for process control, data movement, or global operations within the process group. The increasing interest in collective operations is evidenced by their inclusion in the Message Passing Interface (MPI) standard [7], and by their use in a wide variety of parallel algorithms [8].

2.2 Hardware Multicast in MPCs

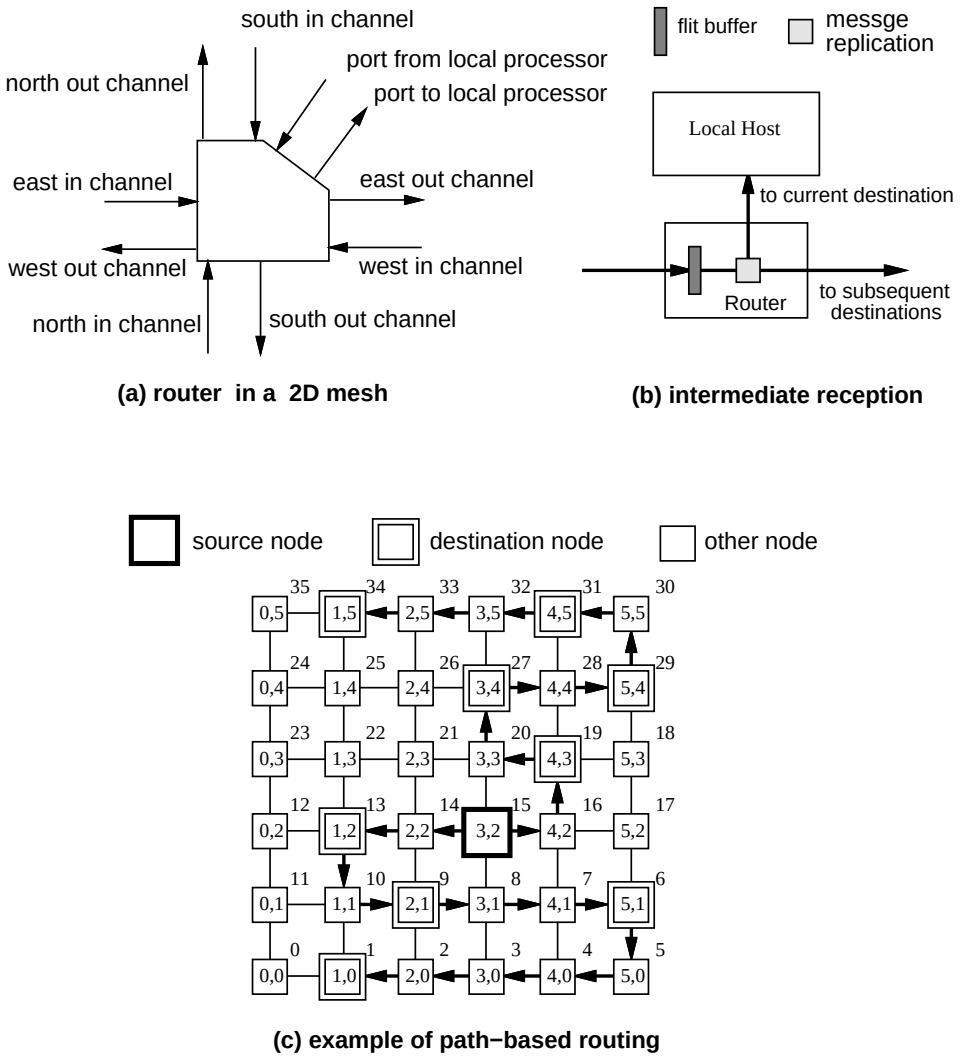
In wormhole routing [1], each message is divided into small pieces called **flits** that are pipelined through the network by way of **routers** at each node. Compared to the store-and-forward switching method that was used in early multicomputers, the pipelining operation of wormhole routing often reduces the effect of path length on communication latency. As a result, wormhole routing has been adopted in many MPC systems [9]. Figure 1(a) shows a node/router pair in a 2D mesh. Each router is connected to its local processor/memory by **internal** channels, or **ports**.

One method that has been proposed to support multicast in wormhole-routed systems is **intermediate reception** (IR). The IR capability allows a router to deliver an incoming message to the local host while simultaneously forwarding it to another router, as shown in Figure 1(b). Although IR requires a relatively minor modification to existing MPC routers, it can be used to deliver a message to multiple destinations in nearly the same time that is needed to send a message to a single destination. Such communication methods are termed **path-based** [10, 11], while the constituent messages are called **multi-destination worms** [12].

An important issue in wormhole-routed systems is the avoidance of deadlock among messages, which is typically handled in the routing algorithm. One approach to deadlock-free, path-based routing is to base routing decisions on an ordering imposed by a **Hamiltonian Path** (HP) in the network. Lin, et al. [10, 11] used the HP approach to develop a family of path-based multicast routing algorithms for mesh and hypercube networks. In the most basic of these algorithms, termed **dual-path**, the source node of a multicast generates at most two multi-destination worms, one for each direction from the source relative to the HP. Figure 1(c) shows the two message worms generated by the dual-path algorithm for an example multicast operation.

2.3 Hardware Multicast in ATM Clusters

Asynchronous Transfer Mode (ATM) is a connection-oriented switching technique, in that the route is established between source and destination prior to data transmission. This connection is referred to as a **virtual channel** (VC). ATM is based on fast packet switching of small **cells**, each containing a 5-byte header and a 48-byte data payload. Although



(a) router in a 2D mesh

(b) intermediate reception

(c) example of path−based routing

Figure 1. Architectural aspects of multicast in MPCs

ATM was primarily designed as the switching technique for future wide area multimedia telecommunications networks, ATM LANs are becoming increasingly popular. A major advantage of switch-based interconnects compared to networks based on a shared medium is that they can provide a very large aggregate bandwidth because multiple packets can be passed simultaneously through the switch at the full channel rate. The increasing popularity of ATM LANs has led to their use in cluster-based parallel computing [13, 14, 15].

Some ATM LANs support hardware multicast. For example, the Fore Systems ASX100 ATM switch [16] allows attached workstations to communicate across multicast virtual channels. Figure 2(a) shows the configuration of an ATM cluster testbed, which we have constructed using

three such switches. Each workstation is connected through fiber optic links to one of the switches. The fabric of these particular switches is a simple high-speed bus, permitting an incoming message to be delivered to multiple output ports on a cell-by-cell basis. While many shared-media LANs also support multicasting, a switched environment with this capability permits multiple multicasts to be transmitted concurrently across the network, a feature that can be useful in many collective operations [13].

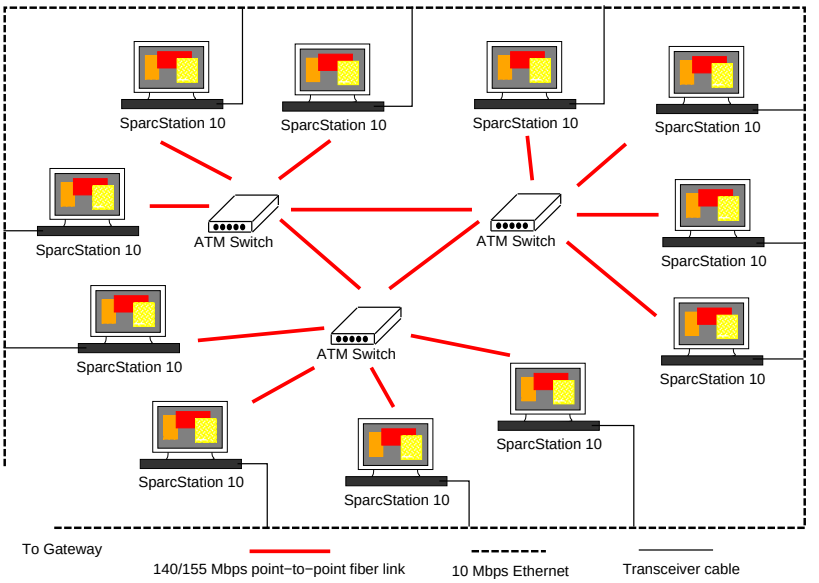


Figure 2. ATM cluster testbed

3 The M-Array and REM-Array Virtual Topologies

If certain patterns of communication are known in advance and used repeatedly over the lifetime of the application, performance may be improved by establishing a **virtual topology** when the application begins execution. A hardware-supported multicast service allows a virtual topology to contain not only edges, but also **hyperedges**, that is, edges associated with more than two endpoints. The tasks required to construct such a **multicast virtual topology** depend on the platform: In wormhole-routed MPCs, it may be sufficient to prepare destination lists for use in message headers; in ATM-based clusters, on the other hand, ATM virtual channels must be established.

In this section, we define two multicast virtual topologies. Both

topologies are **linear** in the sense that they are based on a total ordering of nodes. Several properties of the topologies are stated. Due to space limitations, we omit formal proofs; these may be found in [17].

Definition 1 *A multicast-based array (M-array) of length $N = 2^n$ is recursively defined as follows:*

1. *A single node is an M-array of length 2^0.*
2. *An M-array of length 2^{n+1} consists of two M-arrays of length 2^n, (referred to as left and right sub-arrays). The rightmost node of the left subarray is the source of a multicast channel whose destination nodes are all the nodes in the right sub-array. Similarly, the leftmost node of the right subarray is the source of a multicast channel whose destination nodes are all the nodes in the left sub-array.*

Definition 2 *A multicast channel in an M-array is said to be at* **level** i *if it contains 2^i destinations.*

Figure 3 shows M-array topologies of lengths 2, 4, and 8. By definition, an M-array of length 2^i contains channels at levels $0,\ 1,\ 2,\ \ldots,\ i-1$. In this paper, we will give examples in which the number of nodes is always a power of 2. However, the methods described here can easily be applied to networks with arbitrary numbers of nodes by simply reducing the numbers of destinations on certain multicast channels, using an REM-array.

It can be shown that an M-array of length N has $N \log N$ channel destinations in total. In some environments, it may be desirable to reduce the maximum number of destinations on a multicast channel. In an ATM network, for example, multicast channels with many destinations may require excessive time to be established. It is possible to reduce the total number of destinations to $O(N)$, while retaining efficient implementations of collective operations.

Definition 3 *A resource-efficient multicast-based array (REM-array) is constructed as follows.*

1. *Divide the array into $N/\log N$ segments, each of length $\log N$.*
2. *Using unicast channels, connect each segment to form a spanning binomial tree, with the rightmost node as the root of the tree.*

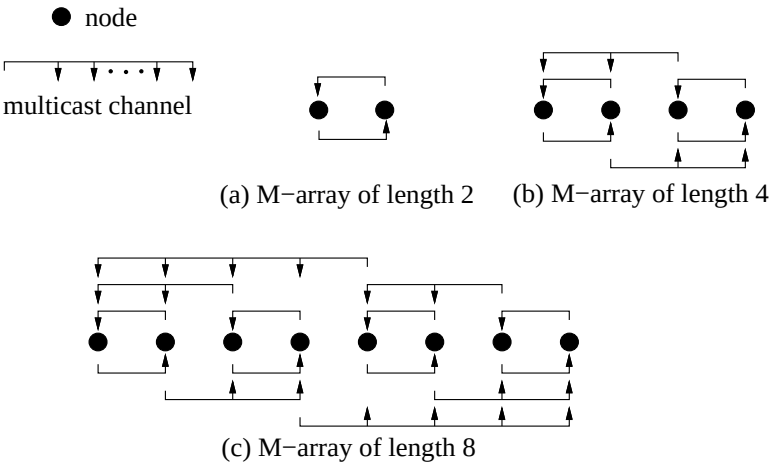


Figure 3. Examples of M-array topologies

3. *Connect the rightmost nodes of the segments according to the pattern of an M-array of length* $N/\log N$.

As an example, for $N = 16$, we have $N/\log N = 4$ segments, each of length $\log N = 4$. Each segment is simply a 4-node spanning binomial tree. The resulting topology is shown in Figure 4.

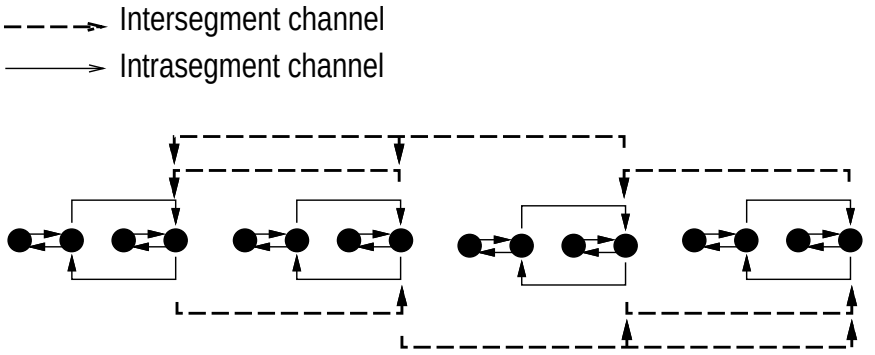


Figure 4. 16-node REM-array topology

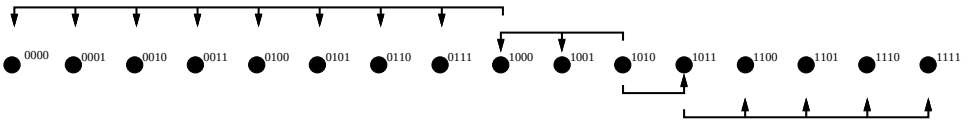
Like the linear array, the REM-array of length N has $O(N)$ total destinations. This property can be important in ATM networks with large process groups, where it is desirable to reduce the density of the topology.

4 Collective Communication Algorithms on M-Arrays

In this section, we show that a broad range of collective operations can be executed efficiently on the M-array; algorithms for the REM-array are described in [17]. All these algorithms are designed to execute such that, within every communication step, all involved channels are at the same level. Such algorithms are called **normal**. Normal algorithms minimize contention for ports and channels during execution.

We begin with the broadcast operation, where the M-array implementation may be especially useful for large groups. For a node to initiate a broadcast operation, it uses the right-to-left (R-L) channels to send data to the left and uses the left-to-right (L-R) channels to send data to the right. An example on a 16-node M-array is shown below, where the source is node 1010; the formal algorithm is straightforward [17].

Example 1 *Broadcast from node 1010 in a 16-node M-array.*

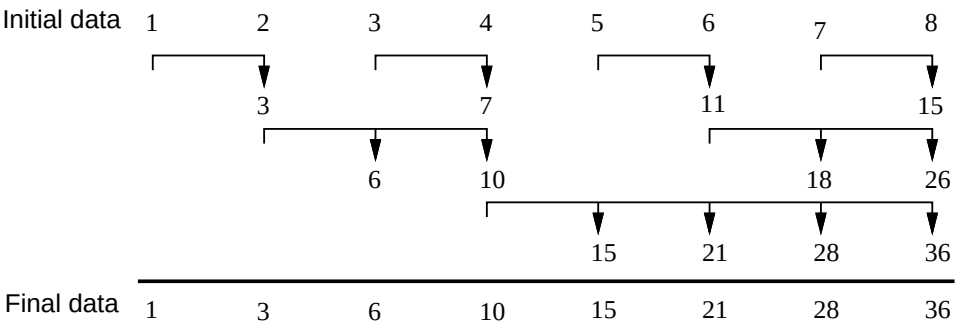


As with a unicast-based binomial tree algorithm, this broadcast algorithm requires $\log N$ steps to reach all destinations. However, because more destinations can be reached in earlier steps, the average number of steps required is less than that of a unicast-based binomial tree. Although it may appear counterintuitive to use a logarithmic broadcast algorithm in an environment that offers hardware-support multicast/broadcast, the M-array supports broadcast from **any** node in the group. This feature may be important for large groups in ATM-based environments, where taking advantage of constant-time group broadcast may require a separate multicast channel to be constructed by **every** potential source node.

The many-to-many collective operations on the M-array are all based on the prefix scan operation. The prefix operation is defined according to the ordering of the nodes. Given processes $P_0, P_1, \ldots, P_n$ and data items $d_0, d_1, \ldots, d_n$, an associative and commutative operator '$*$' is applied such that the result at process P_i is $d_0 * d_1 * \ldots * d_i$. The prefix scan operation on an M-array is straightforward. In order to show how the multicast channels are used, we give a simple example that calculates the prefix scan

of numbers 1, 2, 3, 4, 5, 6, 7, and 8, for the operator addition. Note that a **postfix** scan can be carried out in a similar manner by using the R-L channels. Both scan operations require exactly $\log N$ steps and are normal.

Example 2 *Prefix computation (sum) on an 8-node M-array.*



An N/N global reduction operation (all nodes should receive the result) for an associative and commutative operator can be carried out by combining the results of a prefix scan and a postfix scan. This technique avoids separate reduction and distribution phases. Care must be taken that the local data is included only once in the final result. Each node i retains intermediate data that it receives from the left and the right, then incorporates its local data. If we assume that a node can send and receive simultaneously, then the communication in the two directions can be carried out concurrently. Therefore, a combined prefix scan and postfix scan can be executed on an M-array of length N in $\log N$ communication steps.

Example 3 N/N *global reduction (in this case, addition) on an 8-node M-array.*

Initial	1	2	3	4	5	6	7	8
Prefix	0	1	3	6	10	15	21	28
Postfix	35	33	30	26	21	15	8	0
Final	1+0+35	2+1+33	3+3+30	4+6+26	5+10+21	6+15+15	7+21+8	8+28+0

Other operations use the same message-passing pattern. The barrier synchronization operation can be implemented as an N/N global reduction that applies the logical "AND" operator on a sequence of flags, each of which indicates whether or not its corresponding processor has reached the

barrier. However, the additional requirement must be made that a node will not forward the result until its flag is set. A node having received flags from both the left and the right suggests that all nodes have reached the barrier. All-to-all broadcast can be implemented in $\log N$ communication steps by carrying out an N/N global reduction with "message concatenation" as the operator.

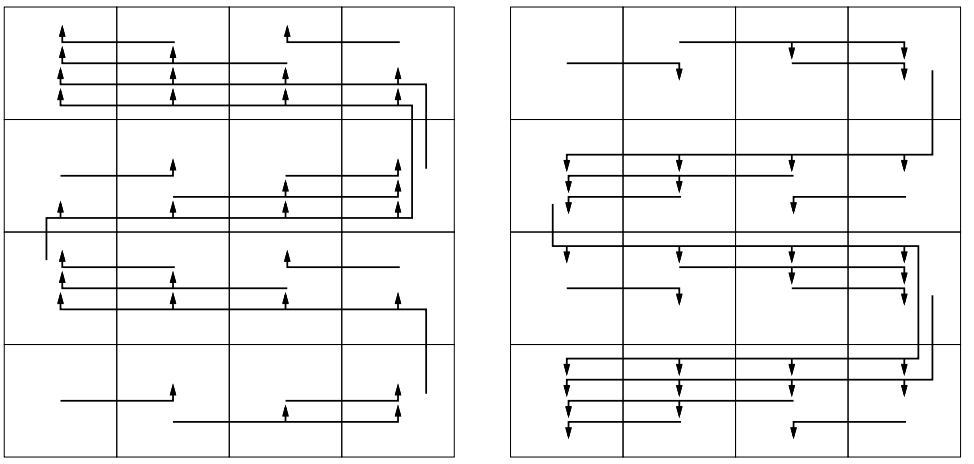
5 Application to Wormhole-Routed Networks

The availability of intermediate reception allows a wormhole-routed network to be used as a multicast topology. In a path-based approach [10, 11], the intermediate nodes of a multi-destination message lie on the path (defined by the underlying routing algorithm) towards the last destination.

5.1 M-array mappings

In a wormhole-routed 2D mesh with intermediate reception, the M-array approach may be applied in different ways. If the underlying routing is based on a Hamiltonian path [11] (see Figure 1(c)), then that path defines a total ordering on the N nodes, and the nodes in the mesh can be considered as an M-array of length N. Figure 5 illustrates this mapping in a 4×4 mesh. In this case, the collective communication algorithms for the M-array can also be applied directly. For large networks, however, some of the multicast paths may be very long: the highest-level multicast channels must cover half of the nodes, so their lengths are $N/2$ in a $\sqrt{N} \times \sqrt{N}$ mesh.

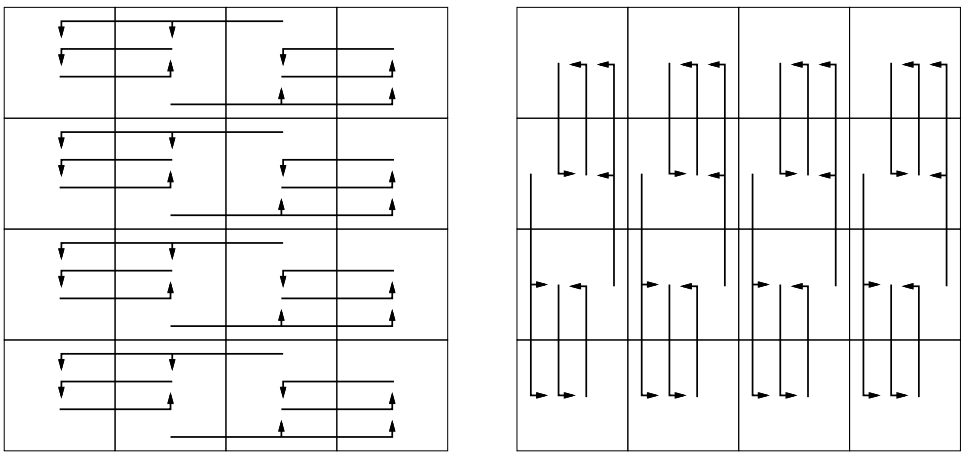
Another way to implement collective operations on $\sqrt{N} \times \sqrt{N}$ wormhole-routed mesh is to treat each row and each column as a separate M-array of length $\sqrt{N}$, effectively creating a two-dimensional multicast topology. This approach, illustrated in Figure 6, has two advantages over the single M-array method. First, the length of the longest path used by an M-array channel is $\sqrt{N}/2$. Second, because every constituent multi-destination message is transmitted exclusively within a row or a column, the paths of these messages are consistent with other underlying routing algorithms besides the Hamiltonian path method. The use of multi-destination worms for various "base routing" schemes, such as dimension-ordered routing and adaptive routing, is described in [12].



(a) forward channels (b) backward channels

Figure 5. Mapping of a 16-node M-array into a 4×4 2D mesh

Definition 4 *An* **M-mesh** *topology is a 2D grid in which M-arrays are constructed along every row and every column.*



(a) row−wise channels (b) column−wise channels

Figure 6. M-mesh virtual topology in a 4×4 2D mesh

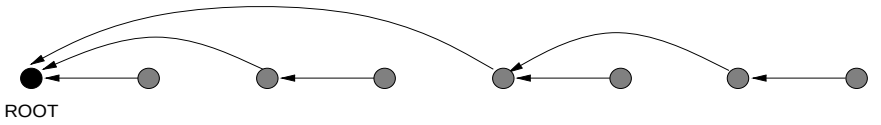
Due to space limitations, descriptions of M-mesh algorithms are omitted in this paper, but may be found in [17].

Table 1 presents analytical results for collective operations implemented in three different virtual topologies: the M-Array, the M-Mesh, and a unicast-based spanning binomial tree. In the table, S stands for the startup overhead at both sending and receiving sites, τ the per-bite channel

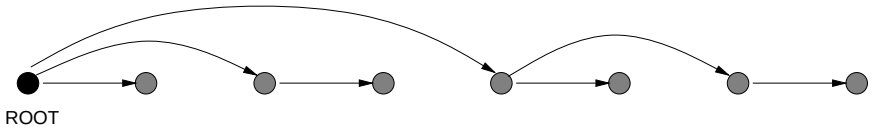
Operation	Approach	Complexity
Prefix	M-Array	$S \log N + 2\tau N$
	M-Mesh	$S \log N + 4\tau\sqrt{N}$
All-to-All Broadcast	M-Array	$S \log N + \tau MN$
	M-Mesh	$S \log N + \tau MN + \tau M\sqrt{N}$
	Tree	$2S \log N + \tau M\sqrt{N}(1 + \log \sqrt{N})(1 + \sqrt{N})$
Global Reduction	M-Array	$S \log N + 2\tau N$
	M-Mesh	$S \log N + 4\tau\sqrt{N}$
	Tree	$2S \log N + 4\tau\sqrt{N}$

Table 1. Summary of performances

transmission time, M the message length in all-to-all broadcast, and N the total number of nodes. The tree-based reduction and all-to-all broadcast operations use a spanning binomial tree in each row and column (see Figure 7). Details can be found in [17]. The tree-based implementation of the prefix operation incurs network contention, which makes its performance difficult to analyze.



(a) Upward links



(b) Downward links

Figure 7. Tree-based global reduction in a linear array

5.2 Simulation study

In order to better understand the potential performance benefits of intermediate reception and the M-array approach to collective communication, as well as the effects of path length and flit forwarding under different system parameters, we use simulation to compare three global reduction

approaches in Table 1. A simulation study was conducted, using a simulation tool called **MultiSim** [18], which we developed for the study of large-scale multiprocessors. MultiSim is based on an event-driven simulation package, CSIM [19]. Using MultiSim, we compared the three reduction algorithms using parameters reported for two different wormhole-routed systems, the Intel Touchstone DELTA and the MIT J-Machine; the relevant parameters are summarized in Table 2. In all our comparisons, we assume that the physical topology is a 2D mesh. The two parameters are S, the combined sending and receiving latencies, and τ, the per-byte channel time. The two systems differ substantially in the ratio of S to τ.

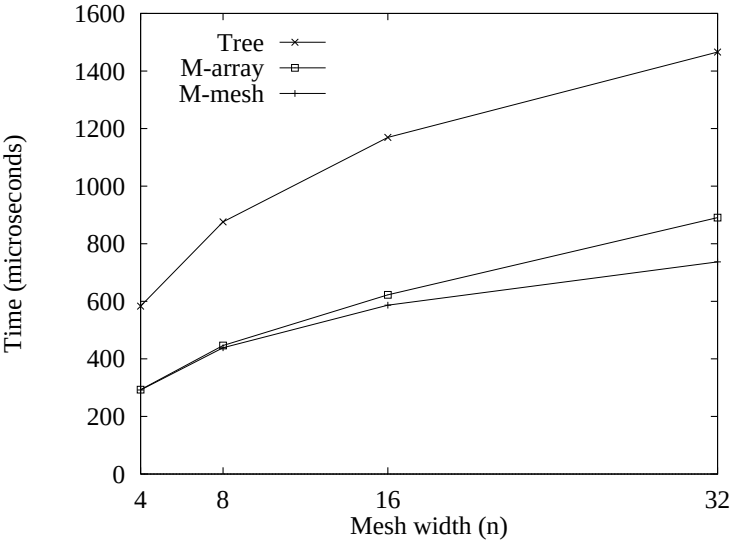
Table 2. Hardware parameters for wormhole-routed systems

System	Topology	$S(\mu sec)$	$\tau(\mu sec/byte)$	S/τ
DELTA	2D mesh	72.0	0.08	900.0
J-Machine	3D mesh	0.9	0.04	22.5

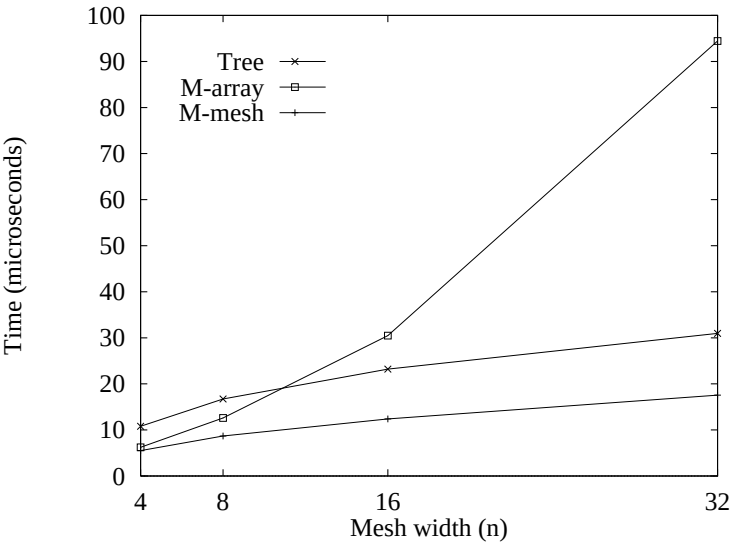
Figure 8(a) compares the algorithms using startup and per-flit values consistent with the Intel Touchstone DELTA. The unicast-based tree is negatively affected by the relatively large constant factor for start-up latency. The two hardware-supported approaches perform equally well in relatively small meshes. In larger meshes, the long path lengths of the HP-based M-array approach increase its execution time relative to the M-mesh approach.

Figure 8(b), which compares the algorithms using startup and per-flit values consistent with the MIT J-Machine, presents a noticeably different picture. First, we note that the absolute values of all three approaches are much lower than that of the fastest algorithm on the DELTA. More importantly, we see the effect of lower startup latency, particularly the lower ratio of S to τ, on the relative performance of the algorithms. The HP-based algorithm suffers from its $O(N)$ transmission delay. The relative performance of unicast-based tree reduction is much better for the J-machine, since its high startup overhead ($4S \log N$) is less important under these conditions. The M-mesh approach again outperforms the other approaches.

In summary, the availability of intermediate reception in wormhole-routed networks can improve the performance of collective operations. In particular, the M-array and M-mesh approaches were shown to improve performance both in theory and when specific system characteristics are taken into account. The M-mesh approach may be particularly useful



(a) DELTA parameters



(b) J-Machine parameters

Figure 8. Global-reduction comparisons for 2D meshes (simulation)

in large systems, where the M-array scheme can suffer due to long path lengths.

6 Application to ATM-Based Clusters

Our experience has shown that the cost of establishing ATM virtual channels can be very high relative to transmission time [13, 20]. Therefore, the M-array, which supports many collective operations with a single multicast virtual topology, is particularly well-suited for ATM LAN environments. Also, since the operations are based on disjoint sets of destinations, the probability of message contention is reduced compared to other implementations (even more so using an REM-array [17]). In this section, we briefly describe an implementation of the M-array on the ATM testbed in our laboratory.

6.1 ATM multicast channels

In our testbed, user-level processes executing on workstations may access the ATM network in one of two ways. First, they may use the standard TCP/IP protocol suite by invoking socket system calls. Alternatively, they may use the Application Program Interface (API), a set of socket-like system utilities that allow user-level processes to bypass TCP/IP and directly access AAL5, an international standard to support data communications over ATM networks. Bypassing the general-purpose TCP/IP protocols reduces overhead and allows the communications library routine to take advantage of the ATM switch. The remainder of this section discusses only AAL5-based software.

The API system calls for establishing ATM virtual channels are similar to Unix socket-related system calls in syntax. However, the ATM connections do not provide either reliability or even flow control, which must be built to our own interface software. Our present implementations of reliable ATM multicast channels and other collective operations execute as a user-level library based on concurrent **threads** [20]. The threads share a common address space and synchronize with one another using semaphores and mutual-exclusive locks provided in the SunOS 5.3 Threads library.

Figure 9 shows the configuration and interaction of threads associated with a multicast channel from one process to seven destination processes. Data transmitted on the multicast channel (depicted with shaded arrows) arrives at the destinations nearly simultaneously, due to the pipelining of ATM cells. The reliability and flow control are realized with a **combining-tree** of acknowledgment channels. Since this strategy introduces $\log(N +$

$1) - 1$ additional delays for N destinations, a sliding window protocol is used in order to improve throughput.

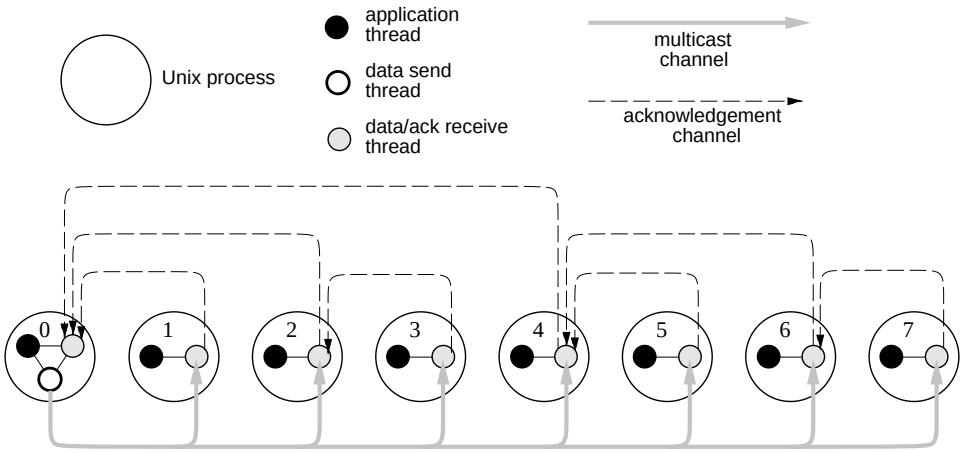


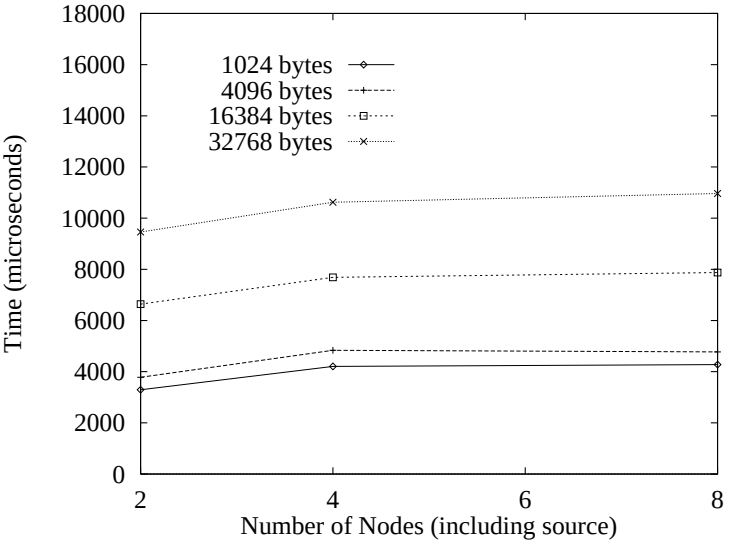
Figure 9. ATM thread configuration for 8-node multicast channel

6.2 Performance

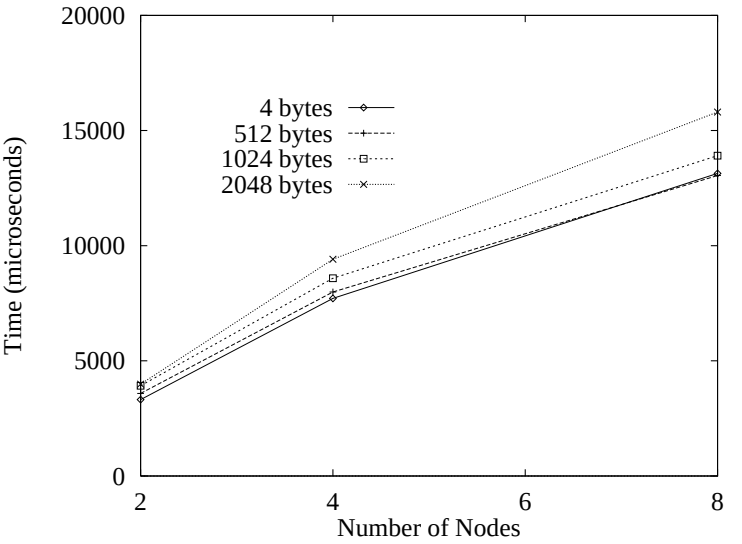
Using our ATM testbed, we have implemented and tested collective operations based on the M-array. Multiple reliable multicast channels, each as described above, are established to form the M-array topology. Independent thread sets are used to manage each channel. Here, we discuss only all-to-all broadcast; results for other collective operations can be found in [20]. We conducted a series of tests with different group sizes and message lengths in order to evaluate the performance of our ATM implementations of collective operations. The performance results were obtained through a series of tests in which nodes were synchronized using ATM multicast "pulses" (one-cell messages).

Figure 10(a) plots the average completion time among destinations for multicast operations with varied group sizes. Despite the logarithmic increase in acknowledgment time, the average completion time is relatively constant irrespective of the number of destinations, particularly from 4 to 8 destinations. This result illustrates that it is possible to take advantage of the hardware multicast capability, despite the implementation of reliability in user-level software.

We measured the performance of the all-to-all broadcast operation over different group sizes and different message sizes. Figure 10(b) plots the results; each data point is the average over 20 runs. Since the message



(a) average reliable multicast completion time



(b) average all-to-all completion time

Figure 10. Measured performance of ATM collective operations

size doubles with each level, the completion time can be approximated by $c_1 \log(N)S + c_2(MN)/b$, where S is the startup latency, b is the bit rate, M is the individual message size, and c_1 and c_2 are constants. Given the relatively high startup latency, the number of communication steps dominates the completion time, and we see that the curves are approximately logarithmic in the number of nodes. For larger messages, however, the corresponding curves would eventually become linear in the

number of nodes. Since every node must receive a message from every other node, this result cannot be avoided.

These results demonstrate that good performance can be achieved by using the M-array to implement collective operations on ATM networks, while limiting the effect of high setup costs and unidirectional multicast channels. In our environment, the M-array implementation benefits from the lack of message contention within the bus-based switch fabric. It is worth noting, however, that the M-array can be mapped onto other switch fabrics, such as the Butterfly and the Omega, without contention among channels; details are given in [17].

7 Related Work

The subject of collective communication in wormhole-routed networks has been extensively studied in that last several years, and the contributions are too numerous to list here; a survey can be found in [21]. The study of distributed parallel computing across ATM networks is growing rapidly, and efficient communication is a key issue [13, 14, 15].

Perhaps the work most closely related to multicast virtual topologies is the study of reconfigurable meshes [22]. In such studies, buses connecting nodes in processor arrays are assumed to be separable and are used as (dynamic) multicast media. Researchers have shown that this concept has great potential for developing fast algorithms. For example, in [22], it was shown that this architecture can implement N/N global reduction of the boolean OR operator in constant time. In some cases, the reconfigurable-bus scheme is even superior to the theoretical PRAM model.

The M-array and REM-array concepts are similar to the reconfigurable mesh work in that they rely on (nearly) constant-time, resource-disjoint multicast communication in order to implement efficient algorithms. However, the reconfigurable mesh approach appears to be more appropriate for fine-grained parallel environments, such as SIMD systems, while the M-array method is applicable to more coarse-grained parallel computing systems, such as distributed memory MIMD multicomputers and workstation clusters. Moreover, the multicast topologies used in reconfigurable meshes are dynamic. While the ability to dynamically change the virtual topology is a powerful tool, it may be difficult to implement in some environments. In fact, an underlying theme of the M-array approach is to use a **single**

static virtual topology to solve multiple problems, in our case, collective operations. This feature is particularly important when the approach is used in a connection-oriented hardware platform where setup costs can be high, such as an ATM network, or in process-group oriented software, where group construction is expensive.

8 Conclusions

In this paper, we have described the M-array and the REM-array, two multicast virtual topologies that can be used to implement collective communication operations in networks that provide multicast communication. This approach to collective communication is in contrast to designing operation-specific hardware, or using only software. Rather, the methods described here require only that the system support a hardware multicast capability.

Efficient methods were described for several communication operations, including broadcast, scan, and all-to-all broadcast. We also showed how these methods can be applied in two very different parallel computing environments: wormhole-routed massively parallel computers and ATM-based workstation clusters. In this manner, these multicast virtual topologies can be used to design communication libraries that are portable yet still take advantage of such low-level hardware functionality.

Our ongoing and future work in this area involves more extensive study of the operations on our ATM testbed, the study of these methods on different wormhole-routed topologies, the use of the resultant operations in parallel numerical algorithms.

Further Information

A number of related papers and technical reports of the Communications Research Group at Michigan State University are available via anonymous ftp and world-wide web. The respective addresses are `ftp://ftp.cps.msu.edu/pub/crg` and `http://www.cps.msu.edu/~mckinley/crgweb`.

References

[1] W. J. Dally and C. L. Seitz, "The torus routing chip," *Journal of Distributed Computing*, vol. 1, no. 3, pp. 187–196, 1986.

[2] H. T. Kung, R. Sansom, S. Schlick, P. Steenkiste, M. Arnould, F. J. Bitz, F. Christianson, E. C. Cooper, O. Menzilcioglu, D. Ombres, and B. Zill, "Network-based multicomputers: An emerging parallel architecture," in *Proceedings of Supercomputing'91*, pp. 664–673, IEEE CS Press, 1991.

[3] P. K. McKinley, H. Xu, A.-H. Esfahanian, and L. M. Ni, "Unicast-Based Multicast Communication in Wormhole-Routed Direct Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, pp. 1254–1265, December 1994.

[4] NCUBE Company, *NCUBE 6400 Processor Manual*, 1990.

[5] C. E. Leiserson, et al., "The network architecture of the connection machine CM-5," in *Proc. 4th ACM Symposium on Parallel Algorithms and Architectures*, pp. 272–285, June 1992.

[6] Fore Systems Inc., 1000 Gamma Drive, Pittsburgh, PA 15238, *ForeRunner ASX-100 ATM Switch Architecture Manual*, 1992.

[7] Message Passing Interface Forum, "Document for standard message-passing interface," Tech. Rep. CS-93-214, University of Tennessee, Nov. 1993.

[8] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Redwood City, California: Benjamin/Cummings, 1994.

[9] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, vol. 26, pp. 62–76, Feb. 1993.

[10] X. Lin and L. M. Ni, "Deadlock-free multicast wormhole routing in multicomputer networks," in *Proceedings of the 18th Annual International Symposium on Computer Architecture*, pp. 116–125, May 1991.

[11] X. Lin, P. K. McKinley, and L. M. Ni, "Deadlock-free multicast wormhole routing in 2D mesh multicomputers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, pp. 793–804, Aug. 1994.

[12] D. K. Panda, S. Singal, and P. Prabhakaran, "Multidestination message passing mechanism conforming to base wormhole routing scheme," in *Proceedings of the First International Parallel Computer Routing and Communication Workshop*, (Seattle, Washington), pp. 131–145, Springer-Verlag, May 1994.

[13] C. C. Huang and P. K. McKinley, "Communication issues in parallel computing across ATM networks," *IEEE Parallel and Distributed Technology*, vol. 2, no. 4, pp. 73–86, 1994.

[14] M. Lin, J. Hsieh, D. H. C. Du, J. P. Thomas, and J. A. MacDonald, "Distributed network computing over local ATM networks," *IEEE Journal of Selected Areas in Communications*, vol. 13, pp. 733–748, May 1995.

[15] T. von Eicken, V. Avula, A. Basu, and V. Buch, "Low-latency communication over ATM networks using active messages," in *Proceedings of Hot Interconnects II*, IEEE Computer Society Press, Los Alamitos, Calif., Aug. 1994. Also appears in IEEE Micro, February 1995.

[16] E. Biagioni and E. C. R. Sansom, "Designing a practical ATM LAN," *IEEE Network*, pp. 32–39, Mar. 1993.

[17] Y. Huang and P. K. McKinley, "Multicast virtual topologies for efficient collective communication," Tech. Rep. MSU-CPS-94-47, Department of Computer Science, Michigan State University, East Lansing, Michigan, Sept. 1994.

[18] P. K. McKinley and C. Trefftz, "MultiSim: A tool for the study of large-scale multiprocessors," in *Proc. 1993 International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Networks (MASCOTS)*, pp. 57–62, Jan. 1993.

[19] H. D. Schwetman, "Csim: A C-based, process-oriented simulation language," Tech. Rep. PP-080-85, Microelectronics and Computer Technology Corporation, 1985.

[20] C. Huang, Y. Huang, and P. K. McKinley, "A thread-based interface for collective communication on ATM networks," in *Proceedings of the 15th IEEE International Conference on Distributed Computing Systems*, (Vancouver, British Columbia), pp. 254–261, 1995.

[21] P. K. McKinley, Y.-J. Tsai, and D. Robinson, "Collective communication in wormhole-routed massively parallel computers." *IEEE Computer*, accepted to appear.

[22] R. Miller, V. Prasanna-Kumar, D. I. Reisis, and Q. F. Stout, "Parallel computations on reconfigurable meshes," *IEEE Transactions on Computers*, vol. 42, pp. 678–692, June 1993.

Biographical Sketches

Yih Huang received the B.S. and M.S. degrees in information engineering and computer science from Feng-Chai University, Taichung, Taiwan, in 1985 and 1987 respectively. He is currently a doctoral candidate in computer science, at Michigan State University.

His current research interests include collective communication operations in massively parallel computers and workstation clusters, and efficient multipoint connection support over ATM networks.

Chengchang Huang received the B.S. degree in information science from National Chiao Tung University, Hsinchu, Taiwan, in 1988. In 1992, he received the M.S. degree in computer science from Michigan State University, where he is presently a doctoral candidate in computer science.

His current research interests include collective communication libraries for parallel and distributed computing in workstation cluster environments, and high-speed network architectures and protocols. He is a member of ACM.

Philip K. McKinley received the B.S. degree in mathematics and computer science from Iowa State University in 1982, the M.S. degree in computer science from Purdue University in 1983, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign in 1989.

He has been an Assistant Professor in the Department of Computer Science at Michigan State University since 1990. He was a member of technical staff at Bell Laboratories in Naperville, Illinois from 1982-1990, on leave of absence 1985-1989. His current research interests include scalable architectures and software, communications libraries for parallel and distributed computing, multicast communication, gigabit network architectures and protocols, and parallel numerical algorithms. He is a member of IEEE, SIAM, and ACM.