Efficient Multicast on Irregular Switch-based Networks with Cut-through Switching*

Ram Kesavan Dhabaleswar K. Panda

Department of Computer and Information Science
The Ohio State University, Columbus, OH 43210
Phone: (614) 292-5199, Fax: (614) 292-2911
Email: {kesavan,panda}@cis.ohio-state.edu

Contact Author: Ram Kesavan

Abstract: The irregular switch-based network of workstations is fast becoming a cost-effective platform for high performance computing. This paper presents efficient multicasting with reduced link contention on irregular networks with switch-based cut-through interconnection and unicast message passing. First, it is proved that for an arbitrary irregular network with a typical deadlock-free, adaptive routing, it is not possible to create an ordered list of nodes to implement an arbitrary multicast in a link contention-free manner with minimal number of communication steps. Next, three different multicast algorithms are proposed with their respective node orderings to reduce link contention: switch-based ordering (SO), switch-based hierarchical ordering (SHO), and chain concatenation ordering (CCO). A variation of the binomial tree-based communication pattern with unicast message passing is used on the above orderings to implement multicast. Then, the problem of node contention is described in the case when multiple multicasts occur concurrently in a system. Using source-based information, the CCO algorithm is modified to propose a source-partitioned chain concatenation ordering (SPCCO) algorithm. It is also shown how the SPCCO algorithm reduces the effect of node contention at the cost of link contention.

Using detailed simulation experiments, the proposed multicast algorithms are compared with each other as well as with the naive random ordering (RO) algorithm for a range of system sizes, switch sizes, message lengths, input buffer sizes, degrees of connectivity, destination set sizes, and communication start-up times. For the case of single multicast, the CCO algorithm is shown to be the best to implement multicast with reduced link contention and minimum latency. For the case of multiple multicasts, the SPCCO algorithm is shown to be the best when the start-up overhead dominates the propagation overhead, and the CCO algorithm is shown to be the best otherwise. The results also highlight the importance of reducing link contention when designing efficient multicast, even for systems with large input buffers in the switches. Thus, these results demonstrate significant potential to be applied to current and future generation NOW systems with irregular interconnection.

Keywords: Parallel Computer Architecture, Cut-through Routing, Wormhole Routing, Multicast, Broadcast, Collective Communication, Switch-based Networks, Irregular Networks, Networks of Workstations.

*This research is supported in part by NSF Career Award MIP-9502294, NSF Grant CCR-9704512, an Ohio State University Presidential Fellowship, and an Ohio Board of Regents Collaborative Research Grant. A preliminary version of this paper has been presented at the International Symposium on High Performance Computer Architecture (HPCA-3), Feb. 1997 [18].

Contents

1 Introduction

Multicast/broadcast is a common collective communication operation as defined by the MPI standard [26]. Parallel systems supporting distributed memory or distributed-shared memory programming paradigms require fast implementation of multicast and broadcast operations in order to support various application and system level data distribution functions. Multicast and broadcast also get used for other collective communication operations like barrier synchronization and global combining [24, 30]. Since broadcast is a special case of multicast (multicast to all nodes in the system), we will consider multicast for the remainder of this paper. However, it must be noted that all the developed algorithms and theories in this paper apply to broadcast as well.

Current generation parallel systems like IBM SP2 [41], Intel Paragon [16], Cray T3E [35], nCube-3 [12], J-Machine [28], and Stanford FLASH use the cut-through switching technique due to its inherent advantages like low-latency communication and reduced communication hardware overhead [27]. These systems provide very small buffer space at each hop, which results in links getting held up by blocked worms. Also, these systems use regular network topologies (such as meshes [16], tori [35], hypercubes [3, 8], multistage interconnection networks [41], etc.) with various deadlock-free routing schemes. Such regular topologies have important mathematical properties that make message communication easier by making message routing simpler, lowering the average distance per communication, and/or increasing the bisection bandwidth [11]. For such regular cut-through networks, many multicast/broadcast algorithms have been proposed in the literature in recent years [2, 5, 9, 10, 17, 20, 23, 25, 32].

More recently, cut-through switching is being applied to switch-based interconnects like Myrinet [4] and ServerNet [15] to build networks of workstations, or NOWs (also called workstation clusters), for cost-effective parallel computing. In contrast to traditional parallel systems, these switches provide larger buffers at the input ports. This allows the trailing flits of a blocked worm to be pooled into the buffers, thus freeing links that would have otherwise been held up. Also, such switch-based networks typically have irregular topologies to allow the construction of scalable systems with incremental expansion capability. This flexibility allows easy addition and deletion of nodes to the computing environment making the overall environment more amenable to network reconfigurations and resistant to faults. However, these topologies do not possess many of the attractive mathematical properties of the regular topologies. This makes the routing schemes on such systems quite complicated. There are routing schemes [1, 7, 33, 34, 36] that have been proposed on such systems to achieve deadlock-free, adaptive routing. The complex nature of such routing schemes also leads to difficulty in implementing a multicast/broadcast operation in a contention-free manner.

Multicast algorithms are typically hierarchical in nature to achieve reduced latency. In these algorithms, some nodes work as intermediate nodes which receive a copy of the message from the source and forward it to other nodes. Typically, tree-structured algorithms are used to minimize the number of communication start-ups (steps) required for multicast [6, 25]. The efficiency of an algorithm is determined by the required number of start-ups for a multicast to complete and the degree of *link contention* experienced among the messages of the multicast. For regular networks with e-cube routing, the concept of a dimension-ordered chain has been developed [25] to implement contention-free multicast with minimum latency. However, for irregular cut-through networks with adaptive routing, developing such contention-free multicast algorithms is a non-trivial task.

The goal of this paper is to develop efficient multicast algorithms for irregular switch-based networks. Without loss of generality, we consider a deadlock-free routing scheme, called *up*/down** (UD) routing, similar to that used in DEC AN1 networks [34]. In addition to providing deadlock-freedom, this routing provides adaptive communication between nodes in an irregular network. With respect to such routing, we first prove that no ordered chain, similar to that proposed in [25], exists to implement contention-free multicast in $\lceil \log_2(d+1) \rceil$ steps for d destinations. Next, we develop multicast algorithms which (a) minimize the number of communication start-ups (steps) for a given number of destinations and (b) minimize contention among the communication steps.

We assume a system consisting of S switches with k ports per switch. We propose three different multicast algorithms with their respective orderings of destinations. The first algorithm, *switch-based ordering* (SO), groups the destinations based on the switches to which they are connected to generate an ordered list of destinations. This algorithm implements multicast with $\lceil \log_2(d+1) \rceil$ steps with contention among the steps. The second algorithm, *switch-based hierarchical ordering* (SHO), provides enhancement by using a two-step hierarchical multicast (inter-switch and intra-switch). This algorithm implements a multicast with up to $(\lceil \log_2 L_1 \rceil + \lceil \log_2 k \rceil)$ steps, where a leader node set of size L_1 is generated after grouping the destinations based on the switches. This algorithm guarantees that the final up to $\lceil \log_2 k \rceil$ intra-switch steps are contention-free. Finally, we propose a *chain concatenation ordering* (CCO) algorithm. For a given network and a set of destinations, this algorithm first determines chains of switches (defined as *partial-ordered-chains* or POCs) which can allow contention-free multicast within themselves. These POCs are concatenated to generate the overall ordered list in order to minimize contention.

Then, we analyze the performance of the proposed CCO algorithm for the scenario where multiple multicasts occur simultaneously in the system. This scenario is a common occurrence in parallel numerical and scientific applications, distributed shared memory systems, etc. In these operations, destination sets of different concurrent multicasts often overlap, leading to nodes participating concurrently in multiple multicasts. We discuss the problem of *node contention* in such multiple multicasts, and describe a technique to reduce such node contention [19, 20]. Using this technique of using source-based information, we propose a *source-partitioned chain concatenation ordering* (SPCCO) algorithm. We show how the SPCCO algorithm reduces node contention at the expense of increased link contention. In the remainder of this paper, we refer to link contention simply as contention, whereas we refer to node contention specifically as node contention.

We then compare the four proposed algorithms using extensive detailed simulation experiments.

In addition to comparing these algorithms with each other, we compare them against a naive *random ordering* (RO) algorithm which is used in MPICH [14], an implementation of MPI. We first use single multicast experiments to isolate the effect of each of the following parameters on the algorithms: system size, switch size, message length, input buffer size, degree of connectivity, destination set size, and communication start-up time. Finally, we study the latency of these schemes under increasing multicast load with a variation of a few selected parameters. This study gives us an understanding on how these schemes behave in realistic multiple multicast traffic. Another important issue that has never been studied is the relevance of reducing link contention for multicast algorithms on systems with switches having large input buffers. In other words, is it meaningful at all to consider link contention as a factor during the design of multicast algorithms on systems with large input buffers? Also, as the size of input buffers increases in current-day switches, does link contention become less and less of a factor?

Our simulation results clearly show that the CCO algorithm is capable of implementing multicast with reduced latency for the single multicast scenario. These results also show that the relative performance improvement of the CCO algorithm with respect to the other algorithms does not decrease with increase in input buffer size (even with input buffer size of four times the message length). This gives us strong evidence that reducing contention is very important while designing multicast algorithms for systems with large input buffer sizes. The multiple multicast experiment results show that the SPCCO and the CCO algorithms perform the best in terms of latency and throughput achievable in the network. The relative performance of these two algorithms depends on whether the communication start-up time dominates the message propagation time, or otherwise. Therefore, we conclude that the SPCCO and the CCO algorithms show significant potential to be applied to current and future generation NOW systems with irregular interconnection.

The rest of the paper is organized as follows. Section 2 provides an overview of irregular networks and some associated issues related to routing. Section 3 shows why implementing contention-free multicast in irregular networks is a non-trivial problem. Section 4 presents the three multicast algorithms in detail. Section 5 discusses the problem of node contention for multiple multicast traffic, and proposes the SPCCO algorithm. Simulation experiments and results, comparing the relative merits of the multicasting schemes, are presented in Section 6. Related works in the literature are described in Section 7. Finally, concluding remarks are made in Section 8.

2 Irregular Networks

In this section, we provide models for irregular switch-based networks and the associated cut-through switches. Issues related to deadlock-free, adaptive routing for such a network are discussed.

2.1 Network Model

Figure 1(a) shows a typical parallel system using switch-based interconnect with irregular topology. Such a network consists of a set of switches where each switch can have a set of ports. The system in the figure consists of eight switches with eight ports per switch. Some of the ports in each switch are connected to processors/workstations, some ports are connected to ports of other switches to provide connectivity between the processors, and some ports are left open for future connections. Such connectivity is typically irregular and the only thing that is guaranteed is that the network is connected. Thus, the interconnection topology of the network can be denoted by a graph G = (V,E) where V is the set of switches, and E is the set of bidirectional links between the switches [4, 34]. Figure 1(b) shows the interconnection graph for the irregular network in Fig. 1(a). It is to be noted that all links are bidirectional and multiple links between two switches are possible. A typical switch-based irregular network can be described by using the following parameters:

P - number of processors

S - number of switches

k - number of ports per switch

f - fraction of the total number of ports in the system which are connected to processors, $P = fSk$

c - percentage connectivity out of remaining $(1 - f)Sk$ ports for interconnection

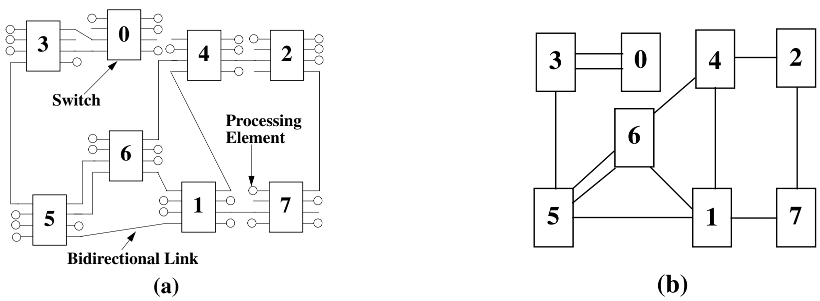

Figure 1: a) An example system with switch-based interconnect and irregular topology; b) corresponding interconnection graph G.

Without loss of generality, we assume $f = 0.5$ in this paper, so half the switch ports of the network are connected to processors. Such a configuration allows a system with a given number of processors to be built using less number of switches while allowing reasonable number of external communication ports per processor [15]. We vary c in our model to provide different types of irregular connectivity.

2.2 Switch Model

Figure 2 shows the architecture of a generic switch with k ports. Each port consists of one input and one output link. As shown in Fig. 1(a), a port can be connected to the port of another switch, a workstation, or kept open. A switch is wired to the workstation through a network interface card which typically is plugged into the I/O bus of the workstation. The switch can implement different types of switching techniques: cut-through or store-and-forward. In this paper we assume switches implementing cut-through switching. Each port consists of an input and an output buffer. Although these buffers only need to be big enough to capture the header flit of an incoming worm so that the routing decision can be made as soon as the header flit arrives, deeper buffers are usually required to perform flow control efficiently across long links. A k-port switch typically provides a $k \times k$ crossbar connectivity in order to enable a concurrent transfer of messages from the input buffers to any of the output buffers [4, 34, 38, 40, 41]. However, in many instances, some routing restrictions are used to achieve deadlock-free routing. We consider some of these issues in the following subsection.

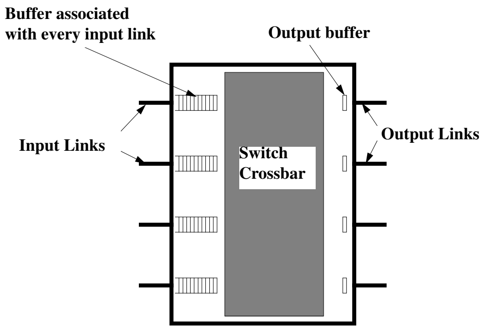

Figure 2: Organization of a typical k-port switch supporting cut-through switching.

2.3 Routing Issues

Several deadlock-free routing schemes have been proposed in the literature for irregular networks [4, 15, 33, 34]. Without loss of generality, in this paper we assume the routing scheme for our irregular network to be similar to that used in Autonet [34] due to its simplicity and its commercial implementation. Such routing allows adaptivity, and is deadlock-free.

In this routing scheme, a breadth-first spanning tree (BFS) on the graph G is first computed using a distributed algorithm. The algorithm has the property that all nodes will eventually agree on a unique spanning tree. Now, the edges of G can be partitioned into tree edges and *cross edges*. According to the property of BFS trees, a cross edge does not connect two switches which are at a difference of more than one level in the tree. Deadlock-free routing is based on a loop-free assignment of direction to the operational links. In particular, the "up" end of each link is defined as: 1) the end whose switch is closer to the root in the spanning tree; or 2) the end whose switch has the lower UID, if both ends are at switches at the same tree level. Links looped back to the same switch are omitted from the configuration. The result of this assignment is that the directed links do not form loops. Figure 3 shows the BFS spanning tree corresponding to the example irregular network shown in Fig. 1. The assignment of the "up" direction to the links on this network is illustrated. The "down" direction is along the reverse direction of the link.

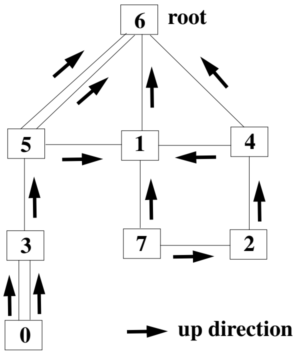

Figure 3: BFS spanning tree rooted at node 6 corresponding to the example irregular network shown in Fig. 1.

To eliminate deadlocks while still allowing all links to be used, this routing uses the following up/down rule: a legal route must traverse zero or more links in the "up" direction followed by zero or more links in the "down" direction. Putting it in the negative, a packet may never traverse a link along the "up" direction after having traversed one in the "down" direction. Details of this routing scheme can be found in [34]. This routing is also referred to as *up*/down** routing or UD routing.

In order to implement the above routing, each switch has an indexed forwarding table. When a worm reaches a switch, the destination address is captured from the header flit of the incoming worm. This address is concatenated with the incoming port number and the result is used to index the switch's forwarding table. The table lookup returns the outgoing port number that the worm should be routed through. The forwarding tables can be constructed to support both minimal and non-minimal adaptive routing. In this paper, we only consider minimal adaptive routing. Thus, the forwarding tables allow only legal routes with the minimum hop count. When multiple minimal routes exist from the source to the destination, the forwarding table entry shows alternative forwarding ports. The choice of the outgoing port is decided dynamically based on the ports which are free when the header flits arrive at the switch. In the case of multiple outgoing ports being free, the routing scheme randomly selects one of them.

3 Contention-Free Multicast in Irregular Networks

In this section, we discuss the significance of ordered chains to achieve contention-free multicast with optimal number of communication steps. We prove that there does not exist an ordered chain of nodes to implement contention free multicast with binomial tree-based message pattern on an arbitrary irregular network with the UD routing scheme discussed in Section 2.3.

3.1 Contention-Free Multicast with Ordered Chain

Typically, binomial tree-based algorithms have been used in the literature [24, 25] to implement multicast on meshes, tori, and hypercubes with optimal number of communication start-ups (steps). Such an approach requires $\lceil \log_2(d+1) \rceil$ communication steps for a multicast with d destinations. Besides the number of startups, an important factor which affects the overall multicast latency is the contention that messages undergo between different steps of the binomial tree-based algorithm. In [25] it has been shown that if an *ordered chain* can be generated among the nodes participating in the multicast, a link contention-free binomial multicast tree can be constructed. Let the symbol $<_d$ denote such an ordering. Such an ordered chain exhibits the following property:

Property 1 *If there exist four nodes w, x, y, and z in an ordered chain such that $w \ <_d \ x \ <_d \ y \ <_d \ z$, then messages between processors w and x will not contend for any links with messages between processors y and z, even for the boundary condition $x = y$ [25].*

3.2 Non-existence of Ordered Chain in Irregular Networks

Using the above property, the contention-free multicast problem in irregular networks reduces to generating an ordered chain among the participating nodes. However, in switch-based networks, concurrent communication between the processors connected to the same switch are contention-free. Thus, the above problem further reduces to generating an ordered chain among participating switches, where a participating switch is defined as a switch having at least one node connected to it which is participating in the multicast. In the worst case of a broadcast, an ordered chain consisting of all the switches in the network must be generated. This chain can be easily reduced to generate the ordered chain for any arbitrary multicast. The following theorem indicates that it is not always possible to generate such an ordered chain for an arbitrary irregular network.

Theorem 1 *Given an arbitrary irregular network using the deadlock-free, adaptive routing discussed in Section 2.3, there does not always exist an ordered chain satisfying Property 1 consisting of all the switches in the network.*

Proof: Consider an irregular network with the routing scheme as discussed in Section 2.3. Let the graph G reflect the connectivity between the participating switches for a broadcast. Let us take five switches $\{s_1, \ldots, s_5\}$ in the BFS spanning tree of G, such that the subgraph G$'$ in Fig 4(a) shows their relative positions in the BFS tree. Let there be no cross links incident on the switches s_1, s_2, s_4, s_5. It can be easily seen that the shortest valid route from switch s_i to switch s_j is along the links of G$'$, where $1 \leq i, j \leq 5$. We claim that any ordered chain in G containing the switches s_1 to s_5 must have either $[s_1, s_2] <_p s_3 <_p [s_4, s_5]$ or $[s_4, s_5] <_p s_3 <_p [s_1, s_2]$. We prove this by contradiction. In this discussion, the square brackets indicate that the relative ordering of the switches enclosed within square brackets is not important. If $[s_1, s_2, s_4] <_p s_3 <_p s_5$, then a message from a processor connected to the switch s_3 to a processor connected to the switch s_5 will contend for the link e with a message from a processor connected to the switch s_1 to a processor connected to the switch s_4. This scenario is shown in Fig 4(b). This violates the Property 1 of ordered chains. Similarly, it can be proved that $[s_1, s_2, s_5] <_p s_3 <_p s_4$, $[s_4, s_5, s_1] <_p s_3 <_p s_2$, and $[s_4, s_5, s_2] <_p s_3 <_p s_1$ cannot be true. Thus, any ordered chain in G containing the switches s_1 to s_5 must have either $[s_1, s_2] <_p s_3 <_p [s_4, s_5]$ or $[s_4, s_5] <_p s_3 <_p [s_1, s_2]$.

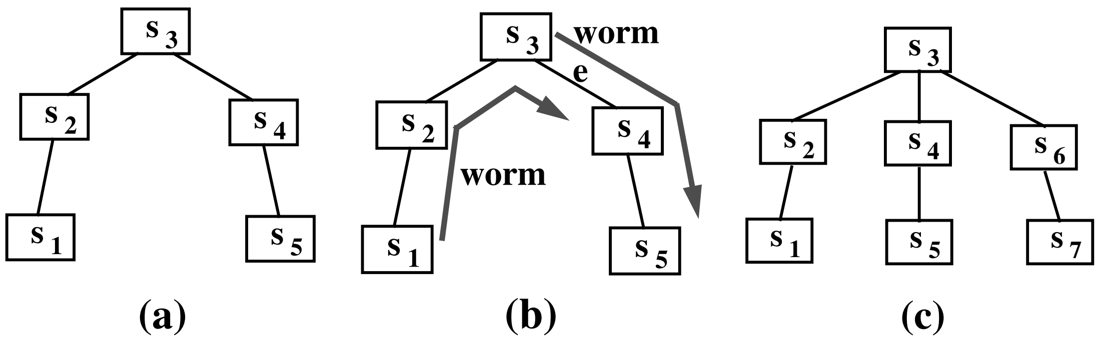

Figure 4: (a) The relative positions of five switches in the subgraph G′ of an example BFS tree, (b) a possible scenario of contention in G′, and (c) subgraph G″ with seven switches, which is part of another example BFS tree.

Now, let us take an example of seven switches s_1 to s_7 in the BFS spanning tree of G, such that the subgraph G″ in Fig 4(c) shows their relative positions in the BFS tree. Let there be no cross links incident on the switches s_1 to s_7 excluding s_3. Using the above reasoning, any ordered chain of G containing the switches s_1 to s_7 must satisfy all three of the following conditions: (a) either $[s_1, s_2] <_p s_3 <_p [s_4, s_5]$ or $[s_4, s_5] <_p s_3 <_p [s_1, s_2]$, (b) either $[s_4, s_5] <_p s_3 <_p [s_6, s_7]$ or $[s_6, s_7] <_p s_3 <_p [s_4, s_5]$, and (c) either $[s_6, s_7] <_p s_3 <_p [s_1, s_2]$ or $[s_1, s_2] <_p s_3 <_p [s_6, s_7]$. It can be easily observed that such an ordered chain is impossible to generate. Therefore, there exists no ordered chain for an arbitrary irregular graph with the routing discussed in Section 2.3. ∎

Since it is not possible to construct an ordered chain, it is a non-trivial problem to implement contention-free multicast with optimal number of communication steps in irregular networks. Thus, in the next section we propose alternative ordering schemes and the associated multicast algorithms to implement multicast with reduced contention as well as with minimum number of steps.

4 Multicast Algorithms

In this section, we present several multicast algorithms. A naive random ordering algorithm is introduced first. Then, we propose three new algorithms with capability for reduced contention during multicast. These multicast algorithms are illustrated with examples to demonstrate their performance and capability to reduce contention.

4.1 Random Ordering (RO) Algorithm

Let the source of a multicast be n_s and the destination processors be in a set D. The naive RO algorithm randomly orders the elements of the set $D \cup \{n_s\}$ into a list, L_0, and executes a binomial tree-based multicast on it. Current generation communication layers use such an algorithm for implementing multicast. For example, the popular MPICH implementation of the MPI standard uses this algorithm for supporting multicast [14, 26, 39].

This algorithm is very simple to implement and it takes $\lceil \log_2(|D|+1) \rceil$ communication start-ups (steps) to complete. Since the destinations and the source are ordered randomly, nothing can be said about the contention among messages of the multicast. Therefore, it is likely that this algorithm is prone to severe contention with an increasing in $|D|$.

Let us consider a sample multicast shown in Fig. 5 on the example irregular network of Fig. 1(a). Processor 0 is the source and {3,9,15,16,19,20,21} is the destination set of this sample multicast. Figure 7(a) shows the multicast tree generated using the RO algorithm for the sample multicast. It also shows the list L_0, which is a random ordering of the elements of $D \cup \{n_s\}$ for the multicast. For this sample destination set, the multicast takes three communication steps to complete.

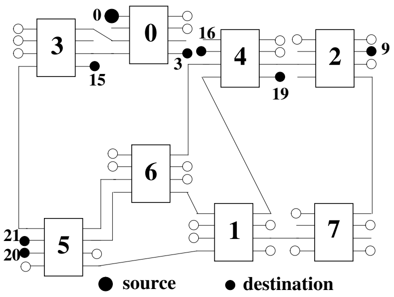

Figure 5: A sample multicast destination set on the example irregular network.

4.2 Switch-based Ordering (SO) Algorithm

The SO algorithm sorts the elements of $D \cup \{n_s\}$ into a list L_0 such that participating processors on the same switch appear adjacent to each other in L_0. This is done by doing a switch-based grouping of the processors and then randomly ordering these groups into the list L_0. Similar to the RO algorithm, a binomial tree-based multicast is now performed on L_0. Figure 7(b) shows the multicast tree generated using the SO algorithm for the sample multicast shown in Fig. 5 on an irregular network. It also details the list L_0 for the multicast. For this sample destination set, the multicast takes three communication steps to complete. A formal specification of the SO algorithm is given in Fig. 6.

Like the RO algorithm, the SO algorithm takes $\lceil \log_2(|D|+1) \rceil$ start-ups to complete. However,

The SO Algorithm

Inputs: G: irregular graph, B: a breadth first spanning tree on the switches of G, D: set of destinations, n_s: the source; $c(n, s)$: node n is connected to switch s.

Output: Send the message to all destinations with minimum number of start-ups.

Procedure:

1. $S \leftarrow \{s \mid s$ is a switch in G, and $\exists d, d \in D \cup \{n_s\}$ and $c(d, s)\}$. S is the set of participating switches.
2. $l \leftarrow < s_1, s_2, \ldots >$ such that $s_i \in S$, and l is an arbitrary ordering of all the switches in S.
3. $L_0 \leftarrow < n_{1,1}, n_{1,2}, \ldots, n_{2,1}, n_{2,2}, \ldots >$ where $\{n_{j,1}, n_{j,2}, \ldots\}$ are the participating processors connected to switch s_j in l.
4. Apply a binomial tree-based multicast on list L_0 with source n_s.

Figure 6: Outline of the SO algorithm.

it reduces contention compared to the RO algorithm. In the latter phases of the multicast, nodes send messages to their neighboring nodes in L_0. Due to the grouping, there is a higher probability of these communications taking place between processors on the same switch. This reduces inter-switch traffic considerably during the latter phases of the multicast when the number of messages is quite large. Intra-switch messages do not contribute to contention since these messages do not use inter-switch links. Therefore, the SO algorithm promises better performance compared to the RO algorithm.

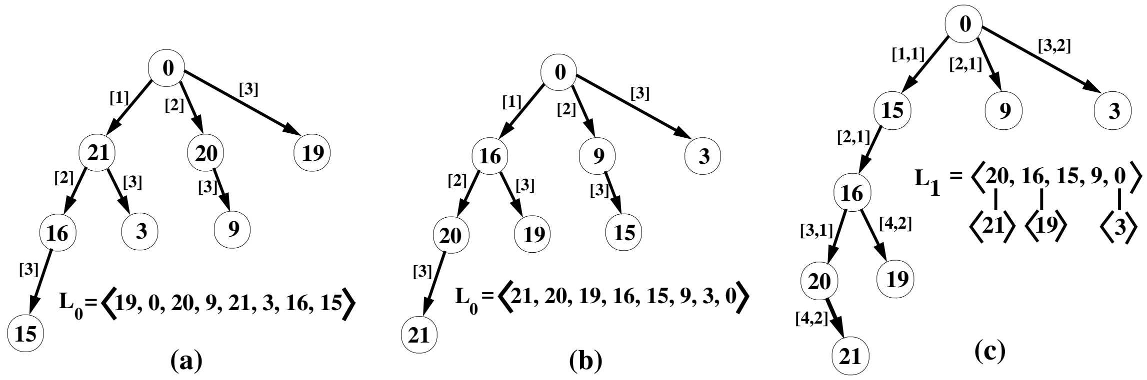

Figure 7: Multicast trees for the sample multicast destination set using algorithms: a) RO, b) SO, and c) SHO.

4.3 Switch-based Hierarchical Ordering (SHO) Algorithm

The SHO algorithm uses the concepts of *leader* and *hierarchy* to guarantee contention-freedom in the latter phases of the multicast. The set $D \cup \{n_s\}$ is partitioned into disjoint subsets such that each subset is represented by a *leader* node. This partitioning is done in a way such that all participating processors connected to a switch form a disjoint subset. For subsets not containing the source node n_s, the processor with the least UID within the subset is chosen as the leader node. The source node n_s is chosen as the leader node of its subset. A list L_1 is formed by randomly ordering all the leader nodes.

The SHO Algorithm

Inputs: G: irregular graph, B: a breadth first spanning tree on the switches of G, D: set of destinations, n_s: the source; $c(n, s)$: node n is connected to switch s.

Output: Send the message to all destinations with less contention (compared to the SO algorithm).

Procedure:

1. $S \leftarrow \{s \mid s$ is a switch in G, and $\exists d, d \in D \cup \{n_s\}$ and $c(d, s)\}$. S is the set of participating switches.
2. $l \leftarrow < l_1, l_2, \ldots >$ where $l_i = < d_{i,1}, d_{i,2}, \ldots >$ such that $d_{i,j} \in D \cup \{n_s\}$ and $c(d_{i,j}, s_i)$. l is an arbitrary ordering of l_i's.
3. $L_1 \leftarrow < n_1, n_2, \ldots >$ where if $c(n_s, s_i)$ then $n_i = n_s$, otherwise $\forall d \in l_i$ UID$(n_i) \leq$ UID(d).
4. Apply a binomial tree based multicast on list L_1 with source n_s.
5. $\forall n_i \in L_1$, apply a binomial tree-based multicast on list l_i with source n_i.

Figure 8: Outline of the SHO algorithm.

A formal specification of the SHO algorithm is given in Fig. 8. The multicast takes place in two stages. The first stage involves executing a binomial tree-based multicast on the elements of the list L_1 with n_s as the source. This stage takes $\lceil log_2 |L_1| \rceil$ start-ups to complete. It is to be noted that there is no contention-freedom guaranteed during this stage. During the second stage, each leader node does a binomial tree-based multicast over its associated subset members. This stage of the algorithm takes up to $\lceil \log_2(k-1) \rceil$ start-ups to complete. This is because there could be up to $k-1$ processors connected to each switch (one port is required for interconnection), and so, each subset could have up to $k-1$ elements. Since this stage of the multicast consists solely of intra-switch messages, they do not experience any contention with other messages. Therefore, the SHO algorithm has reduced contention compared to the SO algorithm. However, the SHO algorithm takes up to $\lceil log_2(|L_1|) \rceil + \lceil log_2(k-1) \rceil$ start-ups, which could be more than the number of start-ups for the SO algorithm for small values of $|D|$. This advantage is offset as the the size of

$|D|$ increases and the message length increases.

Figure 7(c) shows the multicast tree generated using the SHO algorithm for the sample multicast shown in Fig. 5. It also shows the list L_1 for the multicast. The communication steps are identified by $[i, j]$ where i corresponds to the step number (as in the examples for the RO and SO algorithms), and j corresponds to the stage number. For this sample destination set, the multicast takes four communication steps to complete.

4.4 Chain Concatenation Ordering (CCO) Algorithm

The above three algorithms do not attempt to reduce contention during the inter-switch multicast steps. In order to reduce such contention we use a new concept of partial ordered chain (POC) to order the participating switches.

4.4.1 Concept of a Partial Ordered Chain(POC)

A POC is formally defined as follows:

Definition 1 *A partial ordered chain (POC) is an ordered list of a subset of the switches in an arbitrary irregular network such that the nodes in the list satisfy Property 1.*

As proved in Theorem 1, there does not exist a global ordered chain among the switches of an arbitrary irregular network with the deadlock-free, adaptive routing discussed in Section 2.3. Therefore, we attempt to construct as many longest POCs as possible and concatenate them to form an overall ordering. Such a concatenated chain promises reduced contention among inter-switch messages during multicast steps. The following theorem suggests a method of constructing POCs on an irregular network with the routing scheme discussed in Section 2.3:

Theorem 2 *Let P be any ordered list of switches $< s_1, s_2, \ldots, s_n >$, where s_i is connected to s_{i+1} by a "down" tree link (from the BFS spanning tree) or a "down" cross link connecting switches at different levels of the BFS spanning tree. Then P forms a partial ordered chain (POC).*

Proof: Let us use the symbol $<_{poc}$ to denote the order in the above list P. Therefore $s_i <_{poc} s_{i+1}$. Let $E =< e_1, e_2, \ldots, e_{n-1} >$ denote the list of "down" links such that switch s_i is connected to the switch s_{i+1} by the "down" link e_i, where s_i and $s_{i+1} \in P$. A message from a processor connected to the switch s_i to a processor connected to the switch s_j where $s_i <_{poc} s_j$ will take only the links from P if all the links $e_i, e_{i+1}, \ldots, e_{j-1}$ are "down" tree links of the BFS spanning tree. Figure 9(a) shows the only other possible minimal route for worm $w_{i,j}$ from s_i to s_j. The switches in P are highlighted. This scenario cannot occur in a BFS tree because during the construction of the tree, either the cross link e_m would be a tree edge of sub-tree T_m or the cross link e_n would be a tree edge of sub-tree T_n. However, if there is a cross link e_c in the list $\{e_i, e_{i+1}, \ldots, e_{j-1}\}$, then a message from a processor connected to s_i to a processor connected to s_j can take links that are not in P as shown in Fig. 9(b). In the figure, worm $w_{i,j}$ takes links not in the list E. In any case, the links taken by worm $w_{i,j}$ cannot be taken by worm $w_{k,l}$ going from s_k to s_l, where $s_i <_{poc} s_j <_{poc} s_k <_{poc} s_l$. This is because the links e_i to e_{j-1} are at higher level than the links e_k to e_{l-1}, and the worms $w_{i,j}$ and $w_{k,l}$ take minimal paths. Therefore, P is a partial ordered chain. ∎

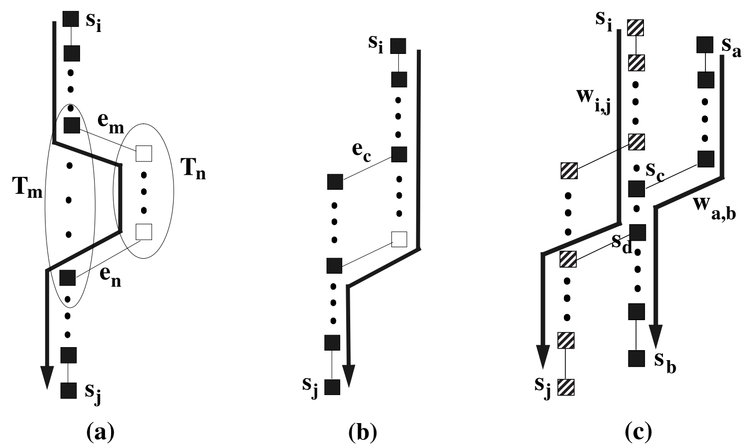

Figure 9: Possible minimal paths from s_i to s_j which take links not in P: a) all links between s_i and s_j in P are *down* tree links of the BFS spanning tree, b) $\exists$ one cross link between s_i and s_j in P, and c) an example of contention between messages of two disjoint POCs.

Now, given an arbitrary multicast destination set in an arbitrary irregular network, the results of Theorem 2 need to be used to construct longest possible POCs. The CCO algorithm, described in the next section, does this efficiently.

4.4.2 The Algorithm

The CCO algorithm constructs as many longest POCs as possible from the participating processors, concatenates the POCs and executes a binomial tree-based multicast on this concatenated list. Such an approach promises to minimize the contention because: a) messages within a POC do not contend with each other and b) a message within one POC contends with a message within another disjoint POC only if one of these messages takes links not contained in its POC. An example of the latter situation is given in Fig. 9(c). In the figure, switches in two POCs, P and P', are highlighted with different shading. The worm $w_{i,j}$ going from s_i to s_j takes links that are not in E. Therefore, there is contention between the worms $w_{i,j}$ and $w_{a,b}$ for the links between switches s_c and s_d.

A formal specification of the CCO algorithm is given in Fig. 10 as a six-step approach. In the first step, a depth-first-search (DFS) is applied on the BFS spanning tree discussed in Section 2.3, starting with the root node r and considering only the "down" links specified in Theorem 2. This is to facilitate the construction of the longest POCs. The step results in a DAG, T. Figure 11(a) shows the DAG, T, which is created when the above DFS is applied on the BFS tree in Fig. 3. Like in the SHO algorithm, a participating switch is defined as one with at least one participating

The CCO Algorithm

Inputs: G: irregular graph, B: a breadth first spanning tree on the switches of G, D: set of destinations, n_s: the source; $c(n, s)$: node n is connected to switch s.

Output: Send the message to all destinations with minimum contention and minimum number of start-ups.

Procedure:

1. Apply a DFS on B from the root node r of B. Only *down* tree links belonging to B, and *down* cross links which connect switches in different levels of B are used. Let the resulting DAG be T.
2. $S \leftarrow \{s \mid s$ is a switch in G, and $\exists d, d \in D \cup \{n_s\}$ and $c(d, s)\}$. S is the set of participating switches.
3. Reduce T to a DAG T' which contains all switches in S and no other switches. Reduction is done by removing all non-participating switches from T. Every time a switch s and all links incident on s are removed, directed *down* links are drawn from each parent of s to each child of s.
4. Calculate the weight wt of each switch in T' as follows:
   $wt(s) \leftarrow \sum_{s_i \in Desc(s)} \rho(s_i)$ where $Desc(s)$ is the set containing s and all the descendents of s in T', and $\rho(s)$ is the number of participating processors connected to the switch s.
5. $L \leftarrow null$; $i \leftarrow 0$
   while T' is not empty
   (a) Let p_1 be a switch with maximum weight in T'.
   $l_i \leftarrow < p_1, p_2, \ldots >$ such that p_{j+1} is the switch with the maximum weight among all children switches of p_j.
   (b) $T' \leftarrow T' - \{p \mid p \in l_i\}$
   (c) $l'_i \leftarrow < n_{1,1}, n_{1,2}, \ldots, n_{2,1}, n_{2,2}, \ldots >$ where $\{n_{j,1}, n_{j,2}, \ldots\}$ are the participating processors connected to switch p_j in l_i.
   (d) $L \leftarrow concat(L, l'_i)$; $i \leftarrow i + 1$
   end while
6. Apply a binomial tree-based multicast on list L with source n_s.

Figure 10: Outline of the CCO algorithm.

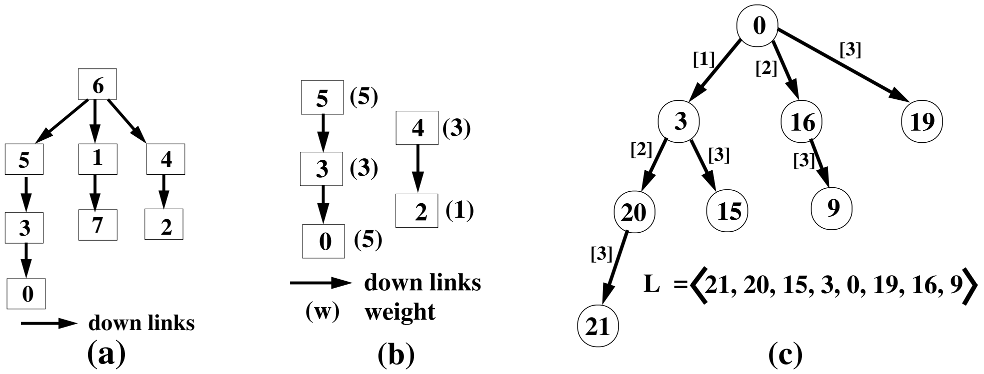

Figure 11: Illustrating the steps of the CCO algorithms on the multicast set of Fig. 5: a) DAG T created by step 1; b) DAG T' created according to step 3 and the weights for switches computed according to step 4, and c) the list L created by step 5 and the corresponding multicast tree.

processor connected to it. In the third step, the resultant DAG, T, from the DFS is reduced to a DAG, T', which contains only the participating switches. Figure 11(b) shows the T' created when the T from Fig. 11(a) is reduced according to the multicast described in Fig. 5.

In order to determine the longest POCs and concatenate them to form an overall ordered list, we carry out a weighted descendents approach. As indicated in step 4, each switch is given an appropriate weight according to the number of participating processors connected to it and to all its descendent switches. Figure 11(b) shows the corresponding weights of each switch in parentheses. The child with the largest weight indicates how to proceed while building the longest POC from the parent. After the weights have been calculated, chains of switches are stripped off from T' according to their weights in step 5. In other words, the heaviest chain gets stripped first from T', and the lightest the last. These chains are concatenated together in chronological order, and each switch is replaced by the participating processors connected to it to form L. The chains of switches stripped off from the T' in Fig. 11(b) are $l_1 =< 5, 3, 0 >$ and $l_2 =< 4, 2 >$, in chronological order. The switches in l_1 and l_2 are replaced by the participating processors connected to them to generate the POCs: $l_1' =< 21, 20, 15, 3, 0 >$ and $l_2' =< 19, 16, 0 >$. The POCs l_1' and l_2' are concatenated to form the list L. Finally, a binomial tree-based multicast is performed on this list L, as indicated in step 6. Figure 11(c) shows the resultant multicast tree generated in step 6, and the list L.

It can be observed that the CCO algorithm has significant potential to reduce contention compared to the SHO and the SO algorithms. It incorporates the grouping effect of the SO algorithm by reducing participating processors to participating switches. It counteracts the extra start-ups due to the hierarchical effect of the SHO algorithm by expanding the switches to the participating processors before the last step. By constructing as many longest POCs as possible and concatenating them together, the contention among messages within POCs is eliminated. The CCO algorithm also takes only $\lceil \log_2 |D| \rceil$ steps to complete. Thus, this algorithm promises potential to implement a multicast with minimum number of communication start-ups as well as reduced contention.

5 An Algorithm for Multiple Multicast

In this section, we consider how algorithms proposed for single multicasts (like the CCO algorithm) behave for the generalized case of multiple multicast. The problem of node contention is described, and a technique of using source based information is is applied to propose the Source-Partitioned-CCO (SPCCO) algorithm.

5.1 Contention in Multiple Multicast

Multiple multicast operations (i.e. two or more multicasts executing simultaneously) occur frequently in parallel systems. Examples include cache-invalidation in distributed shared memory systems, multiple broadcast in numerical and scientific applications (LU decomposition for example), multiple multicast/broadcast operations during concurrent barrier and reduction operations, etc. In these operations, destination sets of different concurrent multicasts often overlap, leading to nodes participating concurrently in multiple multicasts. In such a scenario, the source node of each multicast uses the same algorithm designed for single multicast and constructs its multicast tree independently. With overlapped destination sets, such construction of trees may result in node contention [19, 20]. Let us see how node contention arises when using the CCO algorithm for multiple multicast.

As discussed earlier, the CCO algorithm builds a low-contention ordering of all the nodes, and uses a binomial tree to deliver the multicast to the destinations. Let the chain concatenation ordering for a multicast be $\Phi =< D_{left}, D_{left+1}, \ldots, D_{right} >$ and let $D_{source} \in \Phi$ be the source node. The binomial multicast tree is built in the following manner. The source divides the chain into two halves by sending the message to the node D_{center}. The value of *center* is given as

$$center = \begin{cases} left + \lceil \frac{right-left}{2} \rceil & \text{if } source < \frac{left+right}{2} \\ left + \lfloor \frac{right-left}{2} \rfloor & \text{if } source > \frac{left+right}{2} \\ source - 1 & \text{if } source = \frac{left+right}{2} \end{cases}$$

Then, D_{source} and D_{center} recursively cover the other destinations in their respective halves of the chain. Figure 12(a) shows how a multicast message propagates within a sample CCO. The node D_{center}, which receives the first copy of the message, is positioned half-way in the chain, and is called the *half-node*. The algorithm recursively identifies *quarter-nodes*, *one-eighth-nodes* and so on as the intermediate nodes.

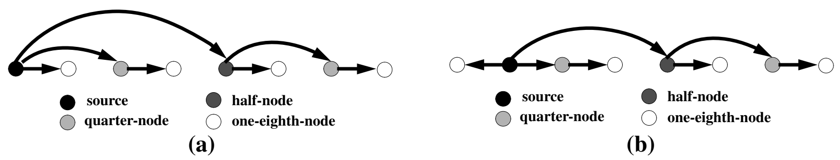

Figure 12: Multicast message pattern for sample CCOs for (a) multicast A and (b) multicast B. The sources, half-nodes, and quarter-nodes are highlighted.

Now, let us consider two multicasts A and B with identical source-destination sets. According to the CCO algorithm, both these multicasts will have the same CCO as shown in Fig. 12. Figures 12(a) and 12(b) show that both multicasts share the same half-node and quarter-nodes. The common half-node for A and B has to *sequentialize* the four message start-ups that it undergoes. This leads to node contention, and two of the messages are delayed. Similarly, if several multicasts have (nearly) identical chain concatenated orderings, they tend to share the same nodes at the key positions along the orderings, leading to hot spots. In the worst case of multiple multicast: *many-to-all broadcast*, each broadcast has the same chain concatenated ordering. Therefore, all the sources choose the same node half-way in the ordering to send their first messages to, the node quarter-way in the ordering to send their second messages to, and so on. This leads to severe node contention and high latency for the multiple multicasts. In an earlier work we presented a detailed analysis of node contention in the context of regular networks [19, 20].

A method to reduce node contention is to make each multicast choose unique intermediate nodes, as different as possible from the rest. With dynamic multicast patterns, all concurrent multicasts are unaware of one another. This means that a multicast has no information whatsoever about the source and destinations of the other multicasts. A good multicast algorithm should use some local information to make its tree as unique as possible. The local information that our new algorithm uses is the position of the source in the system which is unique for each multicast. This technique was proposed and used in [19, 20] to propose the SPUmesh algorithm for regular networks. We use the same technique to propose a new *Source Partitioned CCO* (SPCCO) algorithm for irregular networks.

5.2 Source-Partitioned-CCO (SPCCO) Algorithm

In this section, we propose and discuss the new SPCCO algorithm which reduces the effect of node contention in multiple multicasts.

5.2.1 The Algorithm

As the name suggests, the Source Partitioned CCO algorithm partitions the ordering according to the position of the source in the ordering. Let the concatenated chain ordering (created by the CCO algorithm) containing the source and destinations be Φ. A new ordering Φ' is obtained by a *rotate-left* operation on Φ till the source shifts to the beginning of Φ'. Now the binomial tree-based multicast is built on Φ'. The algorithm is formally presented in Fig. 13.

5.2.2 Reduced Node Contention

Changing ordering Φ to Φ' causes the multicast pattern to be dependent on the position of the source. In other words, each multicast chooses a different half-node depending on the position of its corresponding source node. This reduces the node contention for the centrally positioned

The SPCCO Algorithm

Input: D: set of destinations, n_s: the source.

Output: Send the message to all destinations, reduce node contention between concurrent multicasts

Procedure:

1. Apply the CCO algorithm to derive the chain concatenated ordering, $\Phi = < D_0, D_1, \ldots, D_n >$
2. Let $D_i = n_s$ where $D_i \in \Phi$
3. Create $\Phi_1 = < D_i, D_{i+1}, \ldots, D_n, D_0, D_1, \ldots, D_{i-1} >$
4. Apply a binomial tree-based multicast on Φ' with D_i as source

Figure 13: Outline of the SPCCO algorithm.

node. When Φ' is divided recursively at each stage of the algorithm, the above effect carries over. Therefore, node contention and latency is reduced for multiple multicast as compared to the CCO algorithm.

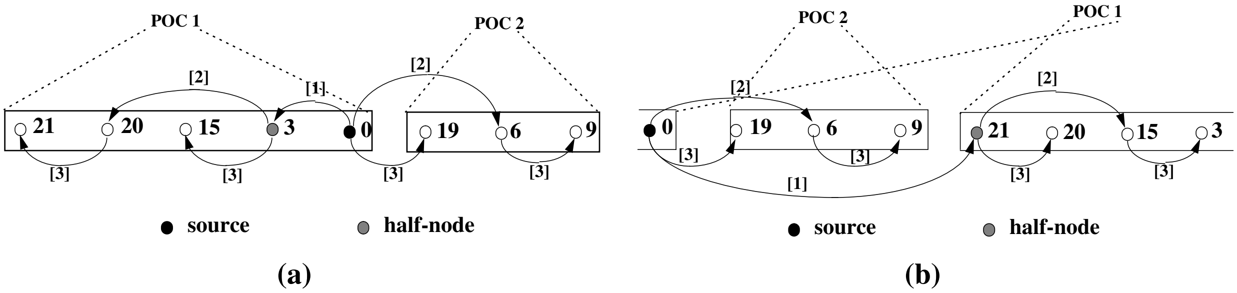

Figure 14: Multicast message patterns generated by (a) the CCO algorithm and (b) the SPCCO algorithm for the example multicast of Fig. 5.

Figures 14(a) and 14(b) show the respective multicast patterns using the CCO and the SPCCO algorithms for the sample multicast of Fig 11. The ordering, Φ, from Fig. 14(a) has been rotated left till the source, node 0, is at the beginning of the new ordering, Φ', shown in Fig. 14(b). It can be seen that the choice of the half-node and quarter-nodes is now based on the position of the source. Also, it should be noted that the new ordering generated by the SPCCO algorithm has all the POCs of the original ordering (generated by the CCO algorithm) intact, except the POC which contains the source node. This POC is now split into two parts, a part of it is at the beginning of the Φ' and the remaining is at the end of Φ'. Although it is not apparent in this example, the splitting of the POC might lead to an increase in inter-POC messages. As discussed earlier, intra-POC messages do not contend for links between themselves. However, inter-POC message are not guaranteed to be contention-free amongst themselves and with respect to other messages. Thus, a general increase in inter-POC messages (and therefore, an increase in link contention) is expected using the SPCCO algorithm. However, it is not clear how much this increase in link contention will offset the reduction in node contention for the case of multiple multicast. Section 6.4 studies this issue using detailed latency versus applied load simulation experiments.

6 Simulation Experiments and Results

In this section, we present results of simulation experiments to compare the three algorithms proposed in Sec. 4 and the SPCCO algorithm proposed in Sec. 5.

6.1 Experiments and Performance Measures

We used a C++/CSIM based simulation test-bed [31] for our experiments. The simulation test-bed is capable of modeling a large number of topologies, and can model a variety of flow control techniques ranging from wormhole routing to virtual cut-through. We assumed cut-through switching as the flow control technique. For all simulation experiments, we assumed system and technological parameters representative of the current trend in technology. The following default parameters were used: t_s (communication start-up time) = 10.0 microseconds, t_{phy} (link propagation time) = 12.5 nanoseconds, t_{route} (routing delay at switch) = 500 nanoseconds, t_{sw} (switching time across the router crossbar for a flit) = 12.5 nanoseconds, t_{inj} (time to inject a flit into network) = 12.5 nanoseconds and t_{cons} (time to consume a flit from network) = 12.5 nanoseconds. The default message length was assumed to be 128 flits, and the default input buffer size at each port was assumed to be 64 flits. In our earlier work [18], we had presented results assuming single-flit input buffers at each port (wormhole routing). Here, we present generalized results for cut-through switching with large input buffers at each port.

We used two types of experiments to measure the performance of the proposed multicasting schemes. In the first type of experiments, we measured the latency of single multicasts for each of the schemes to study the effect of different parameters on the relative latencies of the schemes. We assumed that exactly one multicast occurs in the system at any given time and that there is no other network traffic. The results from these experiments give us an estimate of the best possible performance of each of the schemes in isolation. Furthermore, the results help us isolate the effect of the various network parameters on the performance of each of the schemes. The destinations and network topologies were generated randomly. For each data point, the multicast latency was averaged over 30 different sets of destinations for each of 10 different network configurations. The 95% confidence intervals generated for the data points were observed to be extremely narrow. For our study, we varied each of the following parameters one at a time: the system size, the message length, the startup overhead time, the switch size, the input buffer size in the switches, and the degree of connectivity.

In a real parallel system however, it is unlikely that at any given moment the only traffic in the network is due to a single multicast. A more likely traffic scenario consists of multiple concurrent multicasts in the system. We used such traffic for our second type of experiments. We applied an increasing load consisting of multicast traffic alone and examined the load at which the network saturates with each of the multicasting schemes under the influence of the various parameters. As in [42], we used effective applied load as a measure of our stimulus. For a multicast of degree m and a load of B_i, the effective applied load is mB_i. For each data point, the multicast latency reported was got by taking the average of the latencies obtained from experiments run on 10 different network configurations which were randomly generated. We studied the performance of two different degrees of multicasts over the range of loads till saturation. We also varied each of the following parameters one at a time: the message length, the input buffer size in the switches, the switch size, and the startup overhead time. The next section discusses the irregular topologies used for the experiments and how they were generated.

6.2 Generating Random Irregular Topologies

For all experiments of the first type (single multicast) we assumed a default system configuration of 256 processors interconnected by 64 eight-port switches in irregular topologies. For all experiments of the second type (latency-throughput) we assumed a default system configuration of a 32-processor system interconnected by 8 eight-port switches in an irregular topology. The smaller system size was required to make the latency-throughput simulations manageable in terms of memory and processing time. However, it is clear that the results obtained for the smaller system will scale well to larger systems.

Let us look at the process used for generating the irregular topologies mentioned above. To generate a topology with s k-port switches and p nodes, we reduced the problem to that of generating interconnections among $(sk - p)$ switch ports so that the graph with the switches as vertices remains connected. It was assumed that all ports of a switch are full duplex. Links were not allowed between ports of the same switch. A random number generator was used to generate the port and switch to which a given switch port should be connected or to decide if the port should be connected to a processing node. In the preliminary version of this work [18], we assumed half the ports of each switch to be connected to processors. Here, we place no restriction on number of processor nodes connected to a switch. This allows us to create certain types of topologies where some switches are used purely for interconnection and have no processor nodes connected to them. Also, the degree of connectivity was used to fix the percentage of available switch ports (which did not have a processor connected to it) used for interconnection between switches. The default percentage connectivity was fixed at 75%.

6.3 Single Multicast Performance

We now present our results of the single multicast experiments on the proposed multicasting schemes. One by one, the effect of each parameter on the performance of the schemes is examined. As described earlier, 10 random topologies were generated for each experiment. Then, 30 random multicasts were generated for each multicast set size and for each topology, and each data point reported in the graphs is the average latency of these 300 multicasts.

6.3.1 Effect of System Size

First, we examined the effect of variation in system size on the performance of the proposed multicasting schemes. We simulated the RO, SO, SHO, CCO, and SPCCO algorithms on four different system configurations with 64, 128, 256 and 512 processors, respectively. The switch size was fixed at 8 ports, but the number of switches was 16, 32, 64, and 128 for each system configuration, respectively. All other parameters were maintained at their respective default values. Figure 15 shows these results. It can be observed that the CCO and SPCCO algorithms perform the best for all system sizes and destinations. As the system size increases, the benefits of the CCO and SPCCO algorithms become more prominent. For example, on a 512 processor system with 256 destinations, the reduction in multicast latency achieved by the CCO algorithm is around 35%, 17%, and 15% compared to the RO, SO, and SHO algorithms, respectively. The CCO algorithm performs marginally better than the SPCCO algorithm although this is not apparent in Fig. 15. Since the performance of the SPCCO algorithm is nearly identical to that of the CCO algorithm, we only present but do not discuss the SPCCO results in the remaining single multicast performance results.

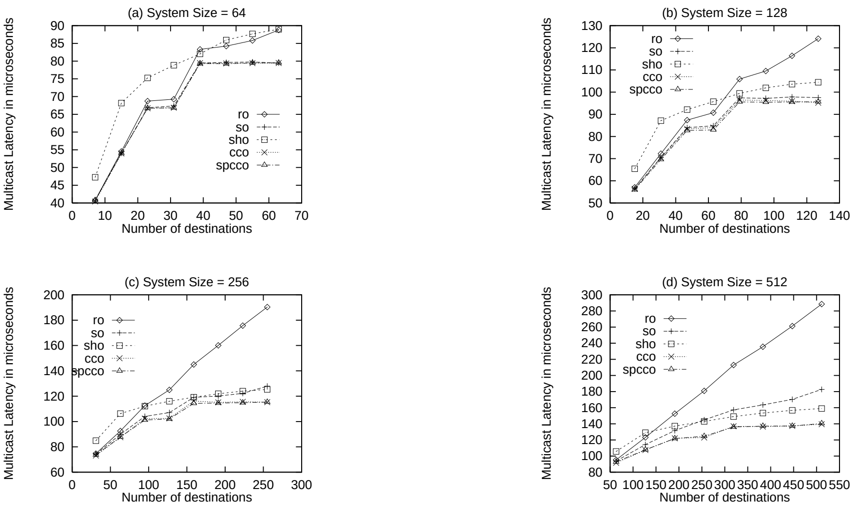

Figure 15: Multicast latency versus number of destinations for four different system configurations: 64, 128, 256, and 512 processors.

It can be observed that the RO algorithm performs the worst. The relative multicast latency using the RO algorithm increases considerably as we move to larger systems and larger number of destinations. The SO algorithm performs well for small sizes of destination sets. However, its latency also increases as we move to larger systems and larger number of destinations. The SHO algorithm does not perform well for smaller sizes of destination sets because of its additional start-up requirement. However, as the number of destinations increases, it performs reasonably well and its performance falls between the SO and CCO algorithms.

6.3.2 Effect of Message Length

We studied the impact of message length on the four algorithms. Five different message lengths - 64, 128, 256, 512, and 1024 flits, on a 256 processor system with default parameters were considered. Figure 16 shows the respective results. It can be easily observed that CCO > SHO > SO > RO for all message lengths, where > reflects the capability to implement the multicast with reduced latency. Also, the improvement in performance obtained by the CCO algorithm increases with increase in message length. This is because a longer message size accentuates the link contention between messages, and this leads to a larger difference in the performance of the algorithms. This is also reflected in the SHO algorithm outperforming the SO algorithm as the message length is increased.

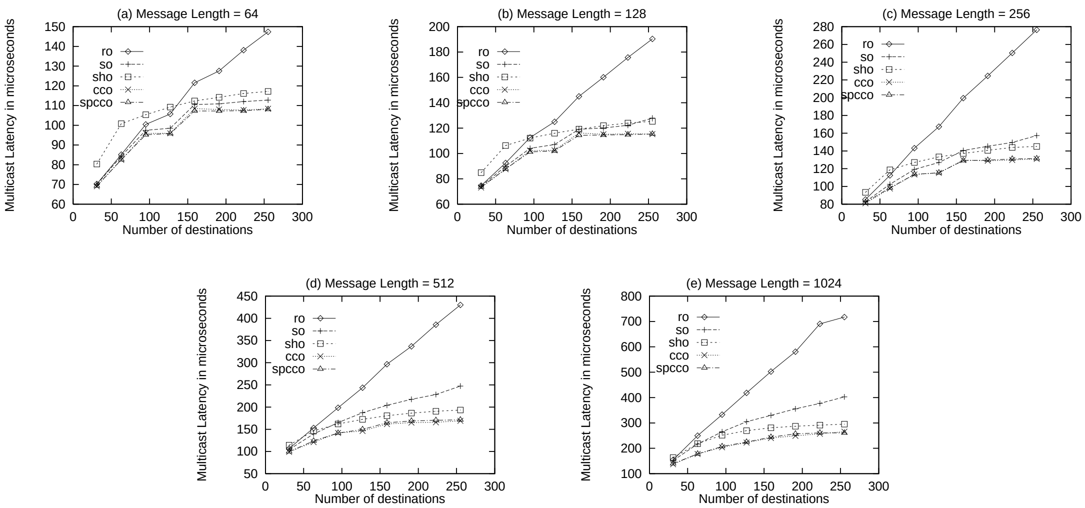

Figure 16: Multicast latency versus number of destinations for five different message lengths: 64, 128, 256, 512, and 1024 flits.

6.3.3 Effect of Communication Start-up Time

We studied the effect of communication start-up time on the performance of the four algorithms. The default 256 processor system configuration was used with four different communication start-up times: 1.0, 5.0, 10.0, and 20.0 microseconds. Figure 17 shows the respective multicast latencies. It can be observed that with higher communication start-up time, the CCO algorithm shows smaller benefits compared to the SO and SHO algorithms. This is expected, because for a given message length, higher start-up times reduce the contention between different phases of the multicast algorithm. The performance of the SHO algorithm worsens with increase in start-up time due to the extra start-up overhead of the SHO algorithm. However, it is clear that as the start-up time diminishes, the CCO algorithm clearly performs the best for all destination sizes. This is because decreasing the start-up time accentuates the link contention in the network.

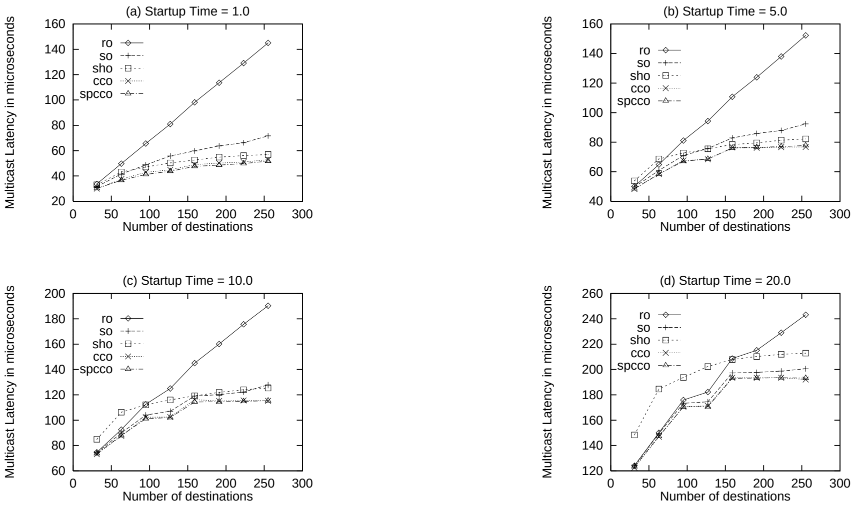

Figure 17: Multicast latency versus number of destinations for three different communication start-up times: 5.0, 10.0, and 20.0 microseconds.

Currently researchers are exploring multiple directions to design efficient network interface architectures [21, 43] and messaging layers [13, 29, 44, 45] to reduce communication start-up time. In this context the current results indicate that message contention in multicast will gradually dominate with reduction in communication start-up time. Thus, algorithms like the CCO hold great promise for implementing multicast with reduced latency in future systems.

6.3.4 Effect of Switch Size

We studied the effect of switch size on the performance of the four algorithms. The default 256 processor system was considered with three different switch sizes: 8, 16, and 32 ports. Figure 18 shows the performance results. It can be observed that with smaller switch size the CCO algorithm performs the best. As switch size increases, more number of communication steps become intra-switch steps. Since intra-switch steps are contention-free, it leads to reduced contention for the overall multicast and the algorithms start delivering equal performance. However, for larger number of destinations, contention still exists for the RO and SHO algorithms. Thus, for bigger switch size and larger number of destinations, either the SO or the CCO algorithm can be used.

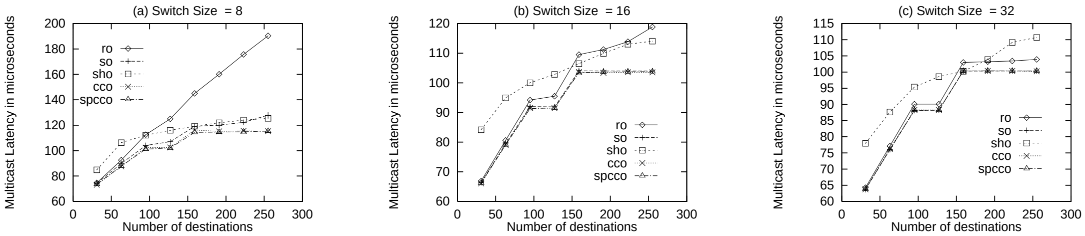

Figure 18: Multicast latency versus number of destinations for three different switch sizes: 8, 16, and 32 ports.

6.3.5 Effect of Input Buffer Size

In the preliminary version of this work [18], we showed that the CCO algorithm performs the best with wormhole routed switches, i.e. cut-through switches with single flit input buffer size. It is well known that increasing input buffer size will allow blocked worms to pool up at the buffers and release downstream links that would otherwise have remained reserved. This should allow other worms to use these freed links. Current day cut-through switches provide large input buffers [4, 15, 40]. This leads us question the very need for low contention multicast algorithms, since larger input buffers reduce link contention.

To answer this question, we studied the impact of input buffer size (in the switches) on multicast latency. The default system size of 256 processors was considered with five different input buffer sizes: 16 flits, 64 flits, 128 flits, 256 flits, and 512 flits. The default message length of 128 flits was used for these experiments. Figure 19 shows the associated performance results. These results show that even with an input buffer size of 512 flits (4 times the message length of 128 flits) the multicast latency of the CCO algorithm is clearly less than the other schemes. In fact, the multicast latencies of all four schemes does not vary much. This is because the increase in buffer space has only moved the contention from the inter-switch links to the input buffers of the switches. These results let us draw a very important conclusion: contention is still an important factor in the design of efficient multicast algorithms even for systems with large input buffers in switches.

6.3.6 Effect of degree of network connectivity

Finally, we studied the impact of degree of network connectivity on single multicast latency. The default system size of 256 processors was considered with three different degrees of network connectivity: 65%, 75%, and 90%. Figure 20 shows the associated performance results. With lesser connectivity, the number of communication links reduces in an irregular network leading to less number of adaptive paths and more contention for multicast. Under such circumstances, the CCO algorithm delivers the best performance. As the degree of connectivity increases, the contention effect reduces, but does not get completely eliminated. Thus, with higher connectivity, the CCO algorithm still performs better compared to other algorithms but, the benefits are reduced.

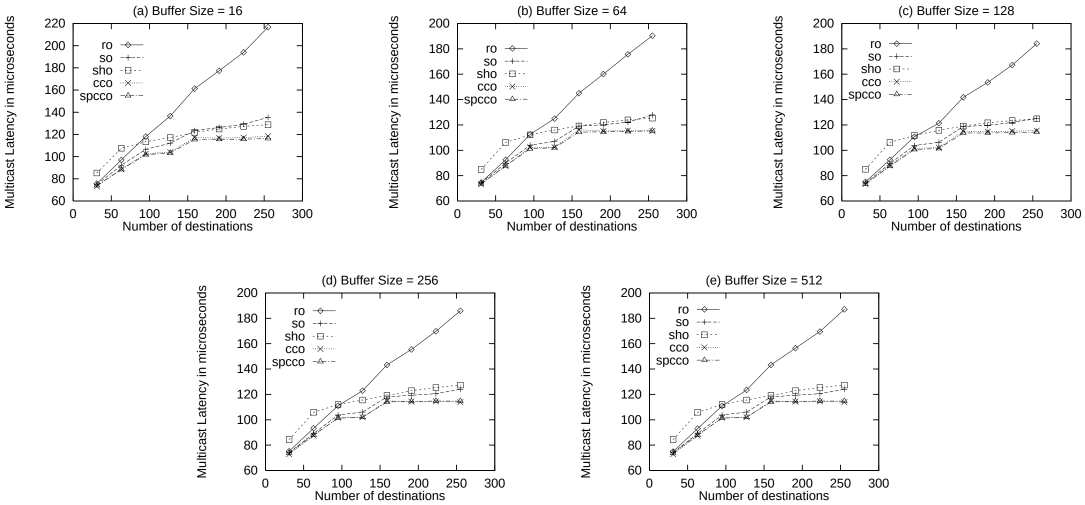

Figure 19: Multicast latency versus number of destinations for different input buffer size in the switches: 16, 64, 128, 256, and 512 flits.

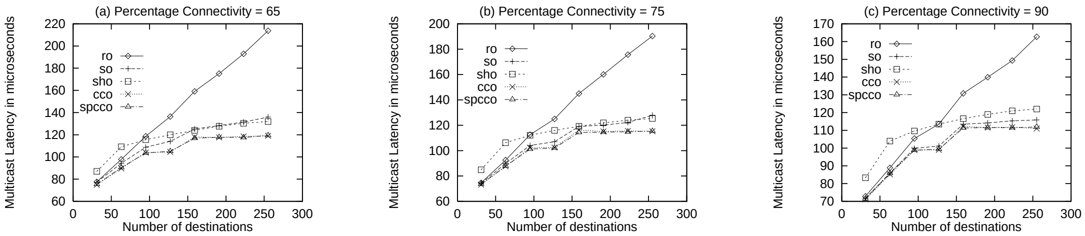

Figure 20: Multicast latency versus number of destinations for different degrees of network connectivity: 65%, 75%, and 90%.

6.4 Latency versus Applied Load for Multiple Multicast

We now present our results for multiple multicast latency under an increasing multicast load for the proposed algorithms. We used two different multicast degrees in our experiments: 15-way multicasts (i.e., multicasts with 15 destinations) and 27-way multicasts. As mentioned earlier, a 32-processor system was assumed for these experiments. For each of our experiments, our simulations were run for at least one million cycles, with measurements beginning after a cold-start time of 500,000 cycles. It is worth keeping in mind that for each of the networks, the maximum unicast throughput (assuming no software overheads and no contention for the I/O bus) with UD routing has been observed to be less than 0.18 in our simulations and in other work [33]. Also, each of the plots in this section show multicast latency against *effective applied load* as discussed in Sec. 6.1. Again, 10 random topologies were generated for each experiment, the results reported is an average over these 10 topologies. The SHO algorithm is not included in all the results reported in this section. This is because the SHO algorithm performed worse than all the remaining schemes due to the extra start-up overhead.

6.4.1 Effect of Message Length

Figure 21 shows the results of our experiments under variation of the message length: 64, 128, and 256 flits. For a smaller message length of 64 flits the SPCCO and CCO algorithms perform almost the same for a smaller degree (15), but the SPCCO algorithm outperforms the rest for a higher degree (27) for the same message length. It should be noted that with increasing message length, the applied load at which the CCO algorithm saturates starts catching up with that of the SPCCO algorithm (and even overtakes it in Fig. 21(c)). This is shown clearly in the increase in message length from 128 flits to 256 flits (Fig. 21(b) to 21(c), and Fig. 21(e) to 21(f)).

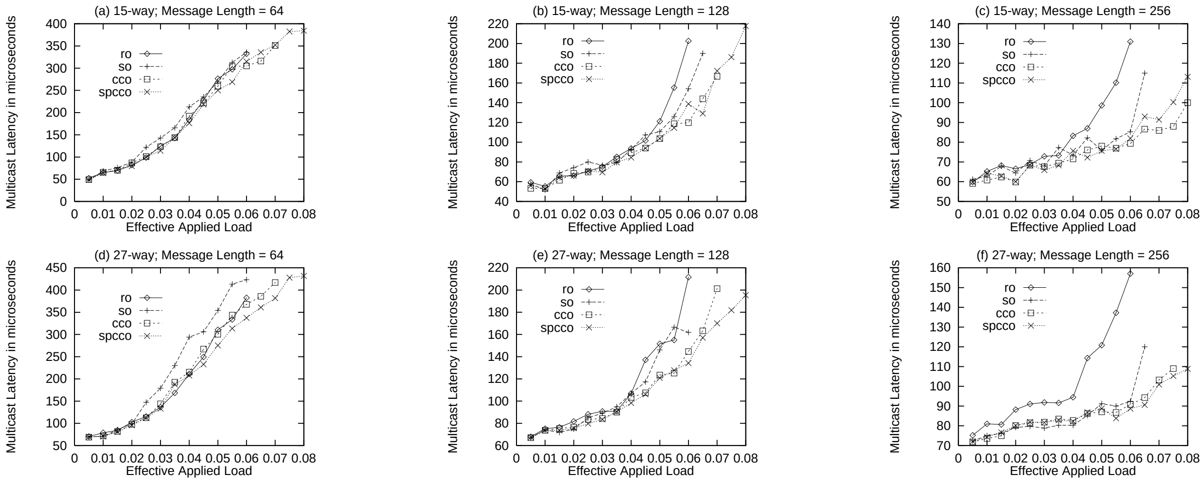

Figure 21: Multicast latency versus applied load for 15-way and 27-way multicasts with varying message length: 64, 128, and 256 flits.

These trends can be explained as follows. For smaller multicast destination sets, the degree of overlapping of the half-nodes, quarter-nodes, etc., of the various concurrent multicasts is not high enough to offset the link contention in the SPCCO algorithm. In other words, the node contention in the CCO algorithm with low degree of multicast (and less overlapping destination sets) is not high enough to offset the increased link contention in the SPCCO algorithm. However, with increase in the degree of multicast (27), the degree of overlapping between intermediate nodes of concurrent multicasts increases. This results in an increase in node contention for the CCO, SO, and RO algorithms. This resultant node contention is reduced in the SPCCO algorithm. Therefore, with increase in multicast degree, the performance of the SPCCO algorithm improves in comparison to the other algorithms. This can be clearly seen with the increase in multicast degree from Fig. 21(a) to Fig. 21(d), Fig. 21(b) to Fig. 21(e), and Fig. 21(c) to Fig. 21(f). This trend can also be seen in all the remaining results reported in this section.

With increase in message length, the link contention in the network increases. This is because longer messages hold up more network links for a longer period of time. This increase in link contention affects the SPCCO algorithm more than the CCO algorithm. At some point, the increase in link contention in SPCCO offsets the node contention in the CCO algorithm (as seen with the increase in message length from Fig. 21(b) to Fig. 21(c)). Therefore, with increase in message length, the performance of the CCO algorithm improves in comparison to the SPCCO algorithm, and the other algorithms as well.

Another point to be noted is that the latency-throughput curves do not have a well-defined knee to indicate the saturation point for a message length of 64 flits. This is because the start-up overhead is too large compared to the propagation time of 64 flit messages in the network. Therefore, the network does not saturate easily with this ratio of start-up overhead time to message propagation time in the network. With increase in message length, there is a reduction in the dominance of the start-up overhead time over the network propagation time. This results in the curves having a well defined curve to indicate the saturation point.

6.4.2 Effect of Input Buffer Size

Figure 22 shows the results of our experiments under variation of the input buffer size in the switches: 16, 64, and 128 flits. As in Fig. 21(a), Fig. 22(a) shows that the CCO algorithm outperforms the SPCCO algorithm for a lower degree of multicast (15) for a smaller buffer size. As explained in the above discussion, the SPCCO algorithm outperforms the CCO algorithm with increase in multicast degree (27). This can be clearly seen when comparing any of Figures 22(a), 22(b), and 22(c) with Figs. 22(d), 22(e), and 22(f), respectively. It can be seen from Fig 22(f) that the applied load at which the SPCCO algorithm saturates is around 15%, 33%, and 45% higher than the load at which the CCO, SO, and RO algorithms saturate, respectively.

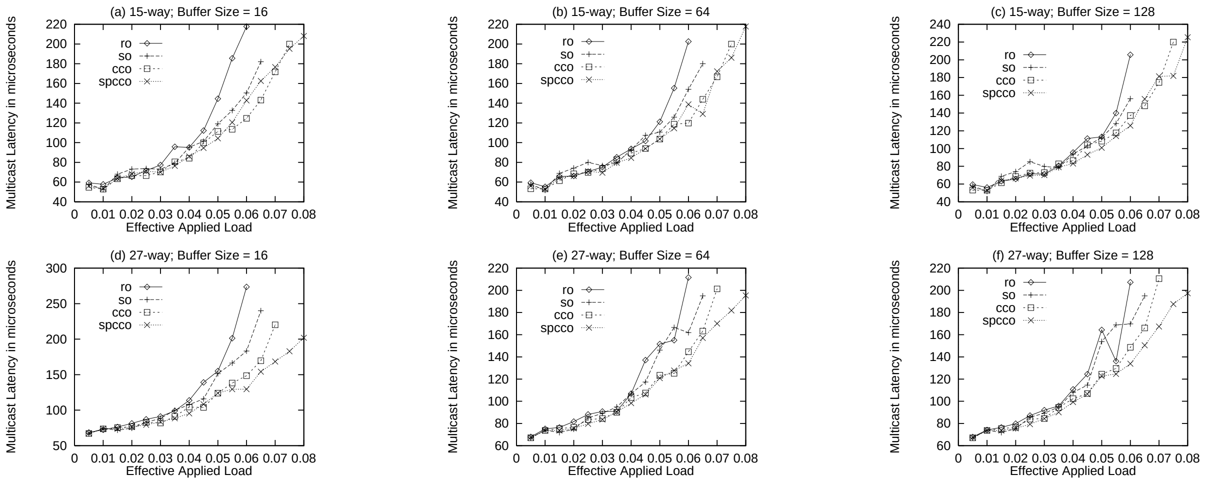

Figure 22: Multicast latency versus applied load for 15-way, and 27-way multicasts with varying input buffer size in the switches: 16, 64, and 128 flits.

With increase in buffer size, the relative performance of all the algorithms does not vary much. We saw in Sec. 6.3.5 that the relative single multicast latency performance of the algorithms is not effected by an increase in buffer size. This trend also holds true for multiple multicast traffic.

6.4.3 Effect of Switch Size

Figure 23 shows the results of our experiments under variation in switch size. In this experiment, we kept the degree of network connectivity as 100%. This is due to the fact that the switch size was required to be varied as 4, 8, and 16 ports. To maintain the same number of switch ports in the system, the number of switches for each of these configurations were 16, 8, and 4, respectively. Sixteen 4-port switches and 32 processors gives 32 ports free ports, which with 75% connectivity results in only 24 ports, i.e. 12 bidirectional links. It is obvious that 12 bidirectional links cannot connect a 16 switch system. Therefore, we assumed 100% connectivity in this experiment to allow 4-port switch configurations of the system. It is to be noted that lower degrees of connectivity will lead to higher link contention, and will, thus, favor the CCO algorithm over the SPCCO algorithm.

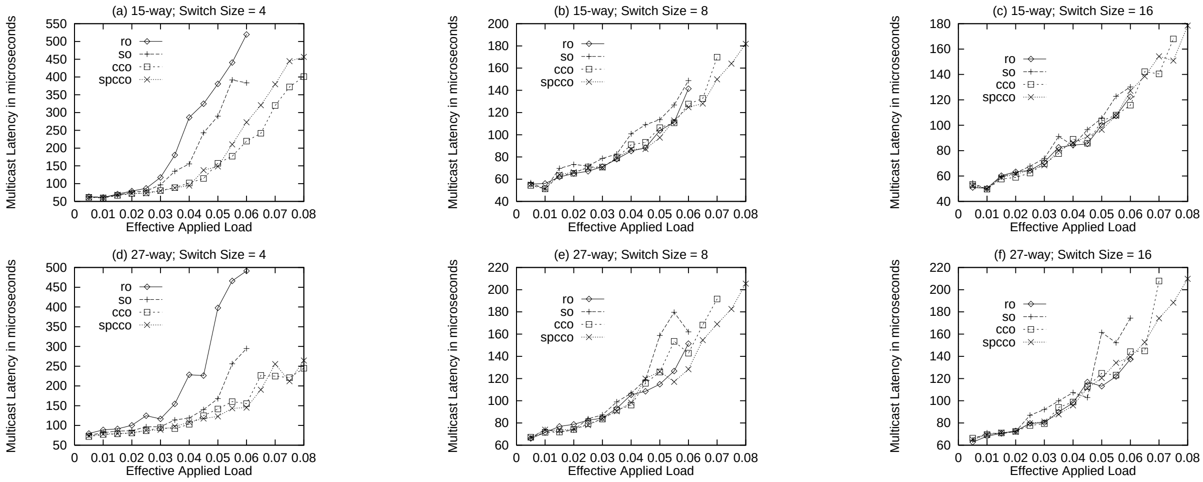

Figure 23: Multicast latency versus applied load for 15-way and 27-way multicasts with varying switch size: 4, 8, and 16 ports.

As expected, the performance of the SPCCO algorithm improves compared to that of the CCO algorithm with increase in degree of multicast. Also, an increase in switch size favors the SPCCO algorithm over the CCO algorithm. This is because an increase in switch size results in more number of communication steps becoming intra-switch steps. Since intra-switch steps are contention-free, it leads to reduced link contention for the multiple multicast traffic. This favors the SPCCO algorithm.

6.4.4 Effect of Communication Start-up Time

Figure 24 shows the results of our experiments under variation of the start-up overhead time: 5.0, 10.0, and 20.0 microseconds. As expected, the performance of the SPCCO algorithm improves compared to that of the CCO algorithm with increase in degree of multicast. With increase in start-up time, the SPCCO algorithm outperforms the CCO and other algorithms. The reason is as follows. A higher start-up time reduces the effect of link contention due to the fact that contention occurs during the propagation time of messages in the network. Thus, if the start-up overhead substantially dominates the propagation time, the effect of link contention is reduced. Also, with increasing start-up time, the effect of node contention is accentuated. Therefore, an increase in start-up time favors the SPCCO algorithm.

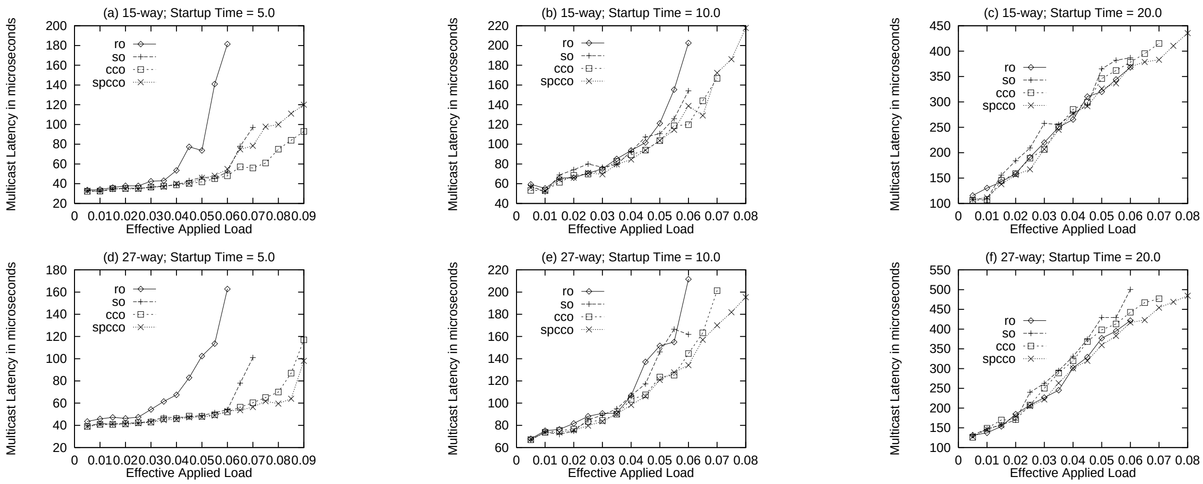

Figure 24: Multicast latency versus applied load for 15-way and 27-way multicasts with varying communication start-up time: 5, 10, and 20 microseconds.

It should also be noted that with an increase in start-up time, the latency-throughput curves do not have a well-defined knee to indicate the saturation point. This can be seen specially in the graphs with start-up time = 20 μs. This is due to the fact that the start-up overhead dominates the propagation time, and this results in the network not saturating easily. A similar trend is seen (and explained) for small message lengths in Sec. 6.4.1.

6.4.5 Evaluation with Zero Start-up Time

Figure 25 shows the results of our experiments under start-up time set to zero. These results are presented to give an idea of the throughput obtainable from the proposed multicast algorithms under the ideal assumption of zero start-up time. This assumption unfairly highlights the link contention in each of the algorithms, and gives a clear picture of how much the CCO algorithm succeeds in reducing link contention.

It can be clearly seen in Fig. 25 that the CCO algorithm substantially outperforms the remaining algorithms. In fact, the saturation applied load for the CCO algorithm is 50%, 50%, and 100% more than that for the SPCCO, SO, and RO algorithms, respectively.

6.5 Summary of Results

In summary, the CCO algorithm performs significantly better than the RO, SO, and SHO algorithms, and marginally better than the SPCCO algorithm for the case of single multicast. The

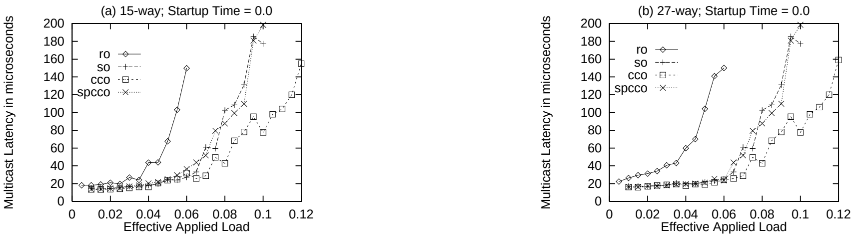

Figure 25: Multicast latency versus applied load for 15-way and 27-way multicasts with communication startup time equal to zero.

difference in performance of these algorithms increases with increase in message length, decrease in communication start-up time, decrease in switch size, and decrease in degree of network connectivity. Also, the CCO algorithm scales very well with system size - its relative performance with respect to the other algorithms improves with increase in system size. Also, the relative performance of these algorithms does not change with increase in buffer size. This leads to the important conclusion that contention is still an important factor in the design of efficient multicast algorithms for systems with large input buffers in switches. In the case of multiple multicast, (a) the SPCCO algorithm outperforms the CCO algorithm when node contention dominates - with higher degree of multicast and larger switches, and (b) the CCO algorithm outperforms the SPCCO algorithm when link contention dominates - with longer messages and lower communication start-up time. Therefore, when designing efficient collective communication support, it is recommended that either the SPCCO algorithm or the CCO algorithm be used judiciously depending on the technological parameters (like communication start-up time and switch size) and characteristics of the application (like message length and multicast degree).

7 Related Work

Several multicast schemes have been recently proposed and evaluated for networks of workstations with cut-through switching . In [33] Ni et. al. have proposed a deadlock-free, adaptive routing scheme for irregular networks with cut-through switches. The routing is based on Eulerian trails. In this paper, we have considered deadlock-free, adaptive routing scheme proposed in [34] due to its simplicity and commercial implementation.

In [43] Bal et. al. have proposed reliable multicast scheme using extra network interface support on Myrinet using Fast Messages [29]. In this work a credit-based scheme has been introduced for managing buffers at senders and receivers to support reliable multicast. Our emphasis on this paper has been on developing alternative multicast algorithms without using any additional network interface support and evaluating their relative performance. In [21, 37] we have shown how the CCO algorithm can be integrated with the smart network interface approach taken in [43] to build more efficient multicast algorithms with lower contention.

In [22] Hadas et. al. have proposed optimal contention-free multicasting using unicast messages.

Although this paper assumes the UD routing, there is a further restriction on the routes some messages can take; these routes are called *relaxed up-first paths.* This further restriction permits the construction of a contention-free multicast for irregular networks. However, the routing scheme is obviously not strict UD routing. The results presented in this paper provide unicast-based multicast solutions for systems supporting the strict UD routing without any constraints.

8 Conclusions

In this paper, we have shown efficient ways of implementing multicast on the emerging irregular switch-based networks with cut-through routing and unicast message passing. Using a typical deadlock-free, adaptive routing for irregular networks, first we have proved that it is not possible to construct a complete ordered chain of destinations to implement multicast in a contention-free manner with optimal number of communication steps. Then, we have proposed three new multicast algorithms (SO, SHO, and CCO) with their respective orderings of destinations. We have discussed the problem of node contention for multiple multicast traffic, and proposed the SPCCO algorithm for efficient multicast in such traffic.

These algorithms together with a naive random ordering (RO) have been evaluated through simulation for a wide range of system sizes, message lengths, switch sizes, input buffer sizes, degrees of connectivity, destination set sizes, and communication start-up times. The simulation results demonstrate the CCO algorithm to be the best for a wide range of system and technological parameters in the single multicast scenario. This algorithm implements multicast with the least amount of contention and minimum latency. The SO algorithm does better than the SHO algorithm for small sizes of destination sets. However, the SHO outperforms the SO as the system size and the number of destinations increase. Overall, for relatively large systems and large number of destinations, the four algorithms have been demonstrated to perform in the following order: CCO (best) $>$ SHO $>$ SO $>$ RO (worst). We have also clearly demonstrated that reducing link contention should be a major focus during the design of efficient multicast algorithms, even for systems with large input buffers in the switches. This is because increasing input buffers in switches only shifts the contention from the links to the buffers, but does not reduce the multicast latency. In the case of multiple multicast traffic, we have shown that the SPCCO outperforms the CCO algorithm with higher degree of multicast and larger switches, and the CCO algorithm outperforms the SPCCO algorithm with increase in message length and decrease in communication start-up time.

As the network/cluster of workstations platform gradually becomes a more popular alternative for high performance computing, the importance of efficient multicasting on such systems will prove to be critical to the overall performance of the system. With a wealth of research focussed at reducing the software start-up overhead at the host workstations, reducing contention while designing efficient multicast algorithms is unavoidable even for systems with large input buffers in switches. Therefore, the CCO and SPCCO algorithms demonstrate significant potential to be applied to current and future generation networks of workstations with irregular interconnection. Also, it will be an interesting exercise to extend this framework to see how other collective communication operations like barrier synchronization, complete exchange, etc., can be implemented on irregular networks with low latency.

Acknowledgements

The authors would like to thank Kiran Bondalapati who collaborated in the earlier version of this work [18]. The authors would also like to thank other members of the *Parallel Architecture and Communication* (PAC) research group in the department for providing comments, criticisms, and suggestions to this work.

Additional Information: A number of related papers and technical reports are available electronically through the home page of *Parallel Architecture and Communication* (PAC) research group. The URL is *http://www.cis.ohio-state.edu/~panda/pac.html*.

References

[1] B. Abali. A Deadlock Avoidance Method for Computer Networks. In *Proceedings of the First International Workshop on Communication and Architectural Support for Network-Based Parallel Computing (CANPC '97).* , pages 61–72, February 1997. Available as `Lecture Notes in Computer Science #1199, Springer-Verlag`.

[2] M. Barnett, S. Gupta, D. G. Payne, L. Shuler, R. van de Geijn, and J. Watts. Interprocessor Collective Communication Library (Intercom). In *Scalable High Performance Computing Conference*, pages 357–364, 1994.

[3] L.N. Bhuyan and D.P. Agrawal. Generalized Hypercube and Hyperbus Structures for a Computer Network. *IEEE Transactions on Computers*, C–33(4):323–333, Apr 1984.

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

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

[6] J. Bruck, R. Cypher, and C.-T. Ho. Multiple Message Broadcasting with Generalized Fibonacci Trees. In *Symposium on Parallel and Distributed Processing*, pages 424–430, 1992.

[7] L. Cherkasova, V. Kotov, and T. Rokicki. Fibre Channel Fabrics: Evaluation and Design. In *29th Hawaii International Conference on System Sciences*, February 1995.

[8] P. J. Chuang and N. F. Tzeng. Dynamic Processor Allocation in Hypercube Computers. In *Proc. 17th Annual Symposium on Computer Architecture*, pages 40–49, 1990.

[9] L. De Coster, N. Dewulf, and C.-T. Ho. Efficient Multi-packet Multicast Algorithms on Meshes with Wormhole and Dimension-Ordered Routing. In *International Conference on Parallel Processing*, pages III:137–141, Aug 1995.

[10] J. Duato. A Theory of Deadlock-Free Adaptive Multicast Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, pages 976–987, September 1995.

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

[12] B. Duzett and R. Buck. An Overview of the Ncube-3 Supercomputer. In *Proceedings of the Frontiers of Massively Parallel Computation*, pages 458–464, 1992.

[13] E. W. Felten, R. A. Alpert, A. Bilas, M. A. Blumrich, D. W. Clark, S. N. Damianakis, C. Dubnicki, L. Iftode, and K. Li. Early Experience with Message-Passing on the SHRIMP Multicomputer. In *International Symposium on Computer Architecture (ISCA)*, pages 296–307, 1996.

[14] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A High-Performance, Portable Implementation of the MPI, Message Passing Interface Standard. Technical report, Argonne National Laboratory and Mississippi State University.

[15] R. Horst. ServerNet Deadlock Avoidance and Fractahedral Topologies. In *Proceedings of the International Parallel Processing Symposium*, pages 274–280, 1996.

[16] Intel Corporation. *Paragon XP/S Product Overview*, 1991.

[17] S. L. Johnsson and C.-T. Ho. Optimum Broadcasting and Personalized Communication in Hypercubes. *IEEE Transactions on Computers*, pages 1249–1268, September 1989.

[18] R. Kesavan, K. Bondalapati, and D. K. Panda. Multicast on Irregular Switch-based Networks with Wormhole Routing. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA-3)*, pages 48–57, February 1997.

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

[20] R. Kesavan and D. K. Panda. Minimizing Node Contention in Multiple Multicast on Wormhole k-ary n-cube Networks. In *Proceedings of the International Conference on Parallel Processing*, pages I:188–195, Chicago, IL, Aug 1996.

[21] R. Kesavan and D. K. Panda. Optimal Multicast with Packetization and Network Interface Support. In *Proceedings of International Conference on Parallel Processing*, pages 370–377, Aug 1997.

[22] R. Libeskind-Hadas, D. Mazzoni, and R. Rajagopalan. Optimal Contention-Free Unicast-Based Multicasting in Switch-Based Networks of Workstations. In *Proceedings of the Merged 12th International Parallel Processing Symposium and the 9th Symposium on Parallel and Distributed Processing*, pages 358–364, April 1998.

[23] X. Lin and L. M. Ni. Deadlock-free Multicast Wormhole Routing in Multicomputer Networks. In *Proceedings of the International Symposium on Computer Architecture*, pages 116–124, 1991.

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

[25] 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*, 5(12):1252–1265, Dec 1994.

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

[27] L. Ni and P. K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Computer*, pages 62–76, Feb. 1993.

[28] M. D. Noakes, D. A. Wallach, and W. J. Dally. The J-Machine Multicomputer: An Architectural Evaluation. In *International Symposium on Computer Architecture*, pages 224–236, 1993.

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

[30] D. K. Panda. Issues in Designing Efficient and Practical Algorithms for Collective Communication in Wormhole-Routed Systems. In *ICPP Workshop on Challenges for Parallel Processing*, pages 8–15, 1995.

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

[32] D. K. Panda, S. Singal, and R. Kesavan. Multidestination Message Passing in Wormhole k-ary n-cube Networks with Base Routing Conformed Paths. *IEEE Transactions on Parallel and Distributed Systems*. In Press.

[33] W. Qiao and L. M. Ni. Adaptive Routing in Irregular Networks Using Cut-Through Switches. In *Proceedings of the International Conference on Parallel Processing*, pages I:52–60, Chicago, IL, Aug 1996.

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

[35] S. L. Scott and G. M. Thorson. The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus. In *Proceedings of the Symposium on High Performance Interconnects (Hot Interconnects 4)*, pages 147–156, August 1996.

[36] F. Silla, M. P. Malumbres, A. Robles, P. Lopez, and J. Duato. Efficient Adaptive Routing in Networks of Workstations with Irregular Topology. In *Proceedings of the First International Workshop on Communication and Architectural Support for Network-Based Parallel Computing (CANPC '97)*, pages 46–60, February 1997. Available as `Lecture Notes in Computer Science #1199, Springer-Verlag`.

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

[38] R. Sivaram, C. B. Stunkel, and D. K. Panda. HIPIQS: A High Performance Switch Architecture using Input Queuing. In *Proceedings of the 12th International Parallel Processing Symposium*, pages 134–143, April 1998.

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

[40] C. B. Stunkel, D. Shea, D. G. Grice, P. H. Hochschild, and M. Tsao. The SP1 High Performance Switch. In *Scalable High Performance Computing Conference*, pages 150–157, 1994.

[41] C. B. Stunkel, D. G. Shea, B. Abali, et al. The SP2 High-Performance Switch. *IBM System Journal*, 34(2):185–204, 1995.

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

[43] K. Verstoep, K. Langendoen, and H. Bal. Efficient Reliable Multicast on Myrinet. In *Proceedings of the International Conference on Parallel Processing*, pages III:156–165, Aug 1996.

[44] T. von Eicken, A. Basu, V. Buch, and W. Vogels. U-Net: A User-level Network Interface for Parallel and Distributed Computing. In *ACM Symposium on Operating Systems Principles*, 1995.

[45] T. von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser. Active Messages: A Mechanism for Integrated Communication and Computation. In *International Symposium on Computer Architecture*, pages 256–266, 1992.