Adaptive Multicast Routing in Wormhole Networks*

R. Libeskind-Hadas T. Hehre A. Hutchings M. Reyes K. Watkins
Department of Computer Science
Harvey Mudd College
Claremont, CA 91711

Abstract

Multicast communication has applications in a number of fundamental operations in parallel computing. An effective multicast routing algorithm must be free from both livelock and deadlock while minimizing communication latency. We describe two classes of multicast wormhole routing algorithms that employ the multi-destination wormhole hardware mechanism proposed by Lin *et al.* [12] and Panda *et al.* [17]. Specific examples of these classes of algorithms are described and experimental results suggest that such algorithms enjoy low communication latencies across a range of network loads.

1 Introduction

The distributed memory multiprocessor paradigm provides a promising means of constructing scalable parallel computers. These systems comprise a collection of *nodes*, where each node consists of a processor with its own local memory and a router which supports message communication between nodes. The routers are connected by *channels* according to a particular interconnection topology. Among the most common topologies for multicomputers are low-dimensional meshes. These topologies are scalable and have a number of features that make them particularly amenable to high-performance computing [4, 9]. For example, a two-dimensional mesh topology is used in the Intel Paragon and a three-dimensional mesh is used in the MIT J-Machine and the Cray T3D and T3E.

In order to minimize communication latency, the current generation of multicomputers employ the *wormhole routing* switching strategy [16]. Wormhole routing divides each packet into a number of constant size components or *flits*. The *header flit* contains the routing information for the message and leads the remaining flits through the network in a pipelined fashion to the destination. Once the header flit gains access to a channel, the current message "owns" that channel until the tail flit passes through it and relinquishes ownership of the channel.

Communication in the network can be either *unicast* or *multicast*. In unicast communication a message is sent from a source processor to a single destination processor, whereas in multicast communication a message is sent from a source processor to an arbitrary set of destination processors. Multicast communication has applications in a number of fundamental operations such as barrier synchronization [20], cache coherency in distributed shared-memory architectures [10], and clock synchronization [1], among others.

An effective multicast routing algorithm must satisfy several basic requirements [12]. First, the algorithm must be free from both deadlock and livelock. Deadlock arises when a cycle of blocked messages forms in which each blocked message waits for a channel belonging to the next blocked message in the cycle. Livelock occurs when a message proceeds through the network indefinitely, never arriving at its destination. Second, the routing algorithm should be supported by fast and simple routing hardware. Third, the algorithm should minimize communication latency.

Communication latency comprises three components: *startup latency*, *network latency*, and *blocking latency* [16]. The startup latency is the amount of time incurred by the operating system in preparing the message for injection into the network. The network latency consists of channel propagation and router latencies. Blocking latency accounts for all delays associated with contention for routing resources among the various worms in the network. In current generation machines, startup latency is typically on the order of several microseconds whereas channel and router latencies are typically on the order of a few nanoseconds [17]. Blocking latency, in contrast, depends heavily on message traffic patterns and thus on the routing algorithm itself.

Since startup latency appears to be the dominant term in communication latency, a number of efforts have been made to design multicast routing algorithms that minimize the number of startups required to deliver a message to all of its destinations [8, 12, 15, 19]. However, the number of startups alone is not necessarily an accurate indicator of the performance of a routing algorithm. Consider, for example, the Hamiltonian path-based routing algorithm proposed in [12]

*This work was supported in part by the National Science Foundation under grant CCR-9418311.

which requires at most two startups to deliver a message to an arbitrary set of destinations and the column path algorithm proposed in [2] which requires as many as $2n$ startups in an $n \times n$ mesh. By using at most two startups to deliver a message to an arbitrary number of destinations, the Hamiltonian-path based algorithm creates long worms that occupy a large number of routing resources. At high loads these long paths incur high network and blocking latencies which result in commmunication latencies that are considerably higher than those of the column path algorithm [2]. Our experimental results suggest that in many cases a trade-off exists between minimization of startup latencies and minimization of network and blocking latencies.

In this paper we consider adaptive multicast routing algorithms that balance startup latencies with network and blocking latencies, resulting in low communication latencies across a range of network loads and traffic conditions. In Section 2 we review existing multicast routing algorithms and propose two general classes of adaptive routing algorithms. In Section 3 we describe specific algorithms that are representative of these two classes. Experimental results are described in Section 4 and we conclude in Section 5 with observations and directions for future research.

2 Multicast Routing

Existing multicast routing algorithms can be classified as *unicast-based* and *multidestination-based*. In unicast-based routing algorithms, a source node sends messages to its set of destination nodes by sending a sequence of separate unicast messages to each of these destinations. Alternatively, a *multicast tree* can be used in which the source node sends the message to a subset of the destinations which then participate in recursively retransmitting the message to the remaining set of destinations. Examples of unicast-based multicast routing can be found in [3, 8, 15, 18]. A significant disadvantage of the unicast-based approach is the large number of startups required to send a message to a large set of destination nodes.

To address this problem, Lin *et al.* [12] and Panda *et al.* [17] have proposed a hardware supported multicast routing methodology known as *multidestination multicast routing*. In this scheme, a source node prepares a message for delivery to a set of destinations by first sorting the addresses of the destinations in the order in which they are to be delivered, and then placing this sorted list in the header flits of a worm. When the header enters a router with address α, the router checks to see if α is the next address in the header. If so, the address α is removed from the header and the data flits are forwarded both to the local processor at this node as well as to the next node on the path. Otherwise, the worm is forwarded only to the next destination on the path. In this way, the message is eventually delivered to every destination in the header. An important feature of multidestination routing is that a message can be delivered to multiple destinations with the same startup latency as a message sent to a single destination [12, 17].

Lin *et al.* have proposed a multidestination-based routing algorithm for the mesh known as the Hamiltonian path-based algorithm [11, 12]. Panda *et al.* [17] have proposed a general multidestination-based *Base Routing Conformed Path (BRCP)* methodology in which an underlying deadlock-free unicast routing algorithm is used in conjunction with the multidestination hardware mechanism. Specifically, to avoid deadlock each worm must follow a path that is valid with respect to the underlying unicast routing algorithm, but the worm may deliver the message to any number of destinations on that path using the multidestination hardware mechanism. For example, Boppana *et al.* proposed a BRCP algorithm called the *column path* algorithm which partitions a set of destinations into worms that can be reached on valid e-cube paths [2]. Panda *et al.* proposed the *hierarchical leader* algorithm that distributes the message to the destinations in a hierarchical fashion, similar to a multicast tree, such that each path taken is also a valid e-cube path. In other related work, Duato [5] has proposed an elegant theory of multicast routing which has been applied to develop several adaptive multicast algorithms [5, 6]. The results described in this paper can be used in conjunction with Duato's theory.

Since the Hamiltonian path-based algorithm requires substantially fewer startups than either the column path or the hierarchical leader algorithms, the Hamiltonian path-based algorithm incurs lower communication latencies than either of these BRCP schemes at low network loads [2, 17]. At higher network loads, however, the long paths taken by the Hamiltonian path-based routing algorithm incur high network and blocking latencies, resulting in overall communication latencies that greatly exceed those of the two BRCP schemes.

Our objective, therefore, is the design of multicast routing algorithms that balance startup latencies with network and blocking latencies to achieve low communication latencies over a range of network loads and traffic patterns. We show that BRCP algorithms based on adaptive unicast routing algorithms offer a very promising means of achieving this goal. In particular, we restrict our attention to non-hierarchical BRCP algorithms in which no intermediate nodes participate in absorbing and retransmitting a message to other destinations. Analytical and experimental results, described in detail in Section 4, suggest that in many cases non-hierarchical algorithms incur fewer startups and lower communication latencies than hierarchical algorithms. For example, for the demanding task of all-to-all broadcast in an $n \times n$ mesh, the non-hierarchical column path algorithm [2] incurs at most $2n$ startups per node while the hierarchical SCHL scheme [8] incurs up to $2n + 7$ startups. Moreover, in several experiments, the column path algorithm exhibited communication latencies that were over 50% lower than those of SCHL in a 16×16 mesh.

We consider two types of non-hierarchical adaptive BRCP algorithms: *pure BRCP algorithms* and *minimal BRCP algorithms.*

Definition 1 *Let U denote a deadlock-free unicast routing algorithm. A multicast routing algorithm M is said to be a* pure BRCP *routing algorithm with respect to U if it has the following property: For every source node s and set of destination nodes D, algorithm M partitions the set D into the minimum number of disjoint lists $P_1, \ldots, P_k$ such that the nodes in each list P_i, $1 \leq i \leq k$, can be reached on a single path from s that is valid with respect to routing algorithm U.*

While this approach minimizes the number of startups per message, it potentially permits long paths that hold a large number of routing resources. In contrast, minimal BRCP algorithms are defined to minimize the usage of routing resources by requiring that the path taken from the source to any node in its destination list be a minimal path.

Definition 2 *Let U denote a deadlock-free unicast routing algorithm. A multicast routing algorithm M is said to be a* minimal BRCP *routing algorithm with respect to U if it has the following property: For every source node s and set of destination nodes D, algorithm M partitions the set D into the minimum number of disjoint sorted lists $P_1, \ldots, P_k$ such that the nodes in each set P_i, $1 \leq i \leq k$, can be reached on a single path path from s such that:*

1. *The path is valid with respect to routing algorithm U.*
2. *For each destination $d \in P_i$, the path taken from s to d is a minimal path.*

Although minimal BRCP algorithms in general require more worms than pure BRCP algorithms, the restriction to minimal paths potentially reduces network and blocking latencies.

We demonstrate these classes by constructing both pure BRCP and minimal BRCP algorithms with respect to the *negative-first* unicast routing algorithm [7]. The negative-first algorithm was selected to demonstrate the effectiveness of adaptive BRCP schemes because it requires no virtual channels and can therefore be compared fairly with the column path and Hamiltonian path-based routing algorithms. Experimental results comparing these BRCP algorithms with the column path and Hamiltonian path-based algorithms are described in Section 4. More complex adaptive routing algorithms employing virtual channels are likely to achieve even lower communication latencies. We conclude in Section 5 with comments and directions for future research.

3 The Negative-First BRCP Algorithms

In this section we describe a pure BRCP routing algorithm and a minimal BRCP routing algorithm with respect to the negative-first unicast routing algorithm. To simplify the discussion, we restrict our attention to two-dimensional meshes, although all of the results described here generalize immediately to higher dimensions. We begin in Section 3.1 with a description of the negative-first unicast routing algorithm. In Section 3.2 we describe the pure negative-first BRCP multicast algorithm and in Section 3.3 we describe the minimal negative-first BRCP multicast algorithm.

3.1 Negative-First Unicast Routing

The negative-first routing algorithm was proposed as a partially adaptive unicast routing algorithm requiring no virtual channels [7]. Although the negative-first routing algorithm was originally described using the *turn model* [7], an equivalent alternative equivalent characterization is as follows [13].

Consider an $m \times n$ mesh. The bottom left node of the mesh is designated the *origin* and is given coordinates $(0, 0)$. The remaining nodes in the network are assigned integer coordinates with respect to the origin. We will henceforth use these integer coordinates to represent nodes. The unidirectional channels of the mesh are partitioned into two subnetworks. The *negative subnetwork* comprises all channels directed towards the origin (all west and south channels) and the *positive subnetwork* comprises all channels directed away from the origin (all east and north channels). A channel c_1 is said to be *contiguous* to channel c_2 if channel c_1 enters a node n and c_2 exits node n. A negative-first routing path p is any sequence of contiguous channels such that all routing in the negative subnetwork is completed before entering the positive subnetwork. This is formalized in the following definition.

Definition 3 *A* negative-first routing path originating at node *s is an ordered list of channels $p = [c_1, c_2, \ldots, c_k]$ such that*

1. *Channel c_1 exits node s.*
2. *Channel c_i is contiguous to channel c_{i+1}, $1 \leq i < k$.*

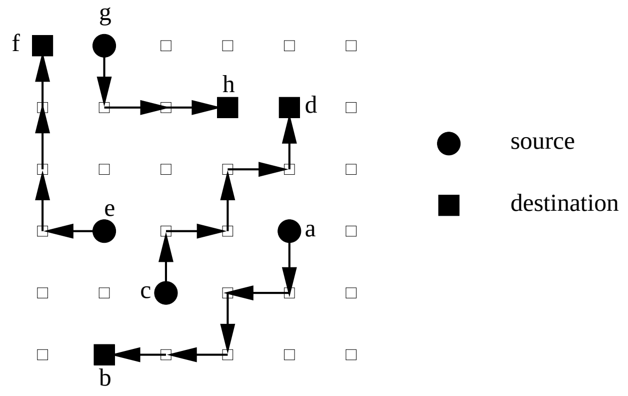

Figure 1: Examples of negative-first routing paths.

3. *If c_i is an element of the negative subnetwork then c_j is an element of the negative subnetwork for all j, $1 \leq j < i$.*

Several negative-first routing paths are shown in Figure 1. It is easily shown that negative-first routing is deadlock-free [7, 13].

3.2 Pure Negative-First BRCP Multicast Routing

We now describe a pure BRCP multicast algorithm with respect to negative-first routing. We begin with some definitions and lemmas that will be used throughout this discussion.

Definition 4 *Let $n_1 = (x_1, y_1)$ and $n_2 = (x_2, y_2)$ denote two nodes in the mesh. Node n_1* contains *n_2, denoted by $n_2 \preceq n_1$, if $x_2 \leq x_1$ and $y_2 \leq y_1$.*

It is easily verified that the containment relation is a partial ordering on the set of nodes in the mesh. Moreover, note that if two nodes n_1 and n_2 satisfy $n_2 \preceq n_1$ then any minimal path from n_1 to n_2 uses channels exclusively in the negative subnetwork and is a valid negative-first routing path by Definition 3. For example, in Figure 1, $b \preceq a$ and one possible minimal path from a to b in the negative subnetwork is indicated. Similarly, if $n_1 \preceq n_2$ then any minimal path from n_1 to n_2 uses channels exclusively in the positive subnetwork and is a valid negative-first routing path by Definition 3. An example is illustrated in the path from c to d in Figure 1. Finally, if n_1 is incomparable to n_2 (neither node contains the other), as is the case for pair of nodes e and f and for pair g and h in Figure 1, then there exists exactly one minimal negative-first routing path from n_1 to n_2.

Definition 5 *Let $L = [n_1, n_2, \ldots, n_k]$ be an ordered list of nodes. L is said to be* containment ordered *if $n_i \preceq n_{i+1}$, $1 \leq i < k$, and is said to be* reverse containment ordered *if $n_{i+1} \preceq n_i$, $1 \leq i < k$.*

The unary *reversal* operator and binary *concatenation* operator on lists, defined below for completeness, will also be used throughout this discussion.

Definition 6 *Let $L = [n_1, n_2, \ldots, n_k]$ be an ordered list of nodes. The* reversal *of L, denoted reversal(L), is the list $[n_k, n_{k-1}, \ldots, n_1]$.*

Definition 7 *Let $L_1 = [n_{1,1}, n_{1,2}, \ldots, n_{1,k}]$ and $L_2 = [n_{2,1}, n_{2,2}, \ldots, n_{2,\ell}]$ denote two ordered lists. The* concatenation *of L_1 and L_2, denoted $L_1 \circ L_2$, is the ordered list*
$[n_{1,1}, n_{1,2}, \ldots, n_{1,k}, n_{2,1}, n_{2,2}, \ldots, n_{2,\ell}]$.

Each worm in our multicast routing algorithm must conform to the negative-first routing algorithm. This notion is formalized in the following definition.

Definition 8 *Let s be a source node and let D denote the set of distinct destination nodes for source s. A* negative-first routing list *with respect to s is an ordered list of nodes $L = [n_1, n_2, \ldots, n_k]$ such that $n_i \in D$, $1 \leq i \leq k$, and the nodes in L can be visited in order on a negative-first routing path originating at s.*

Given a source node s and a set of destinations D, our objective is to partition D into the minimum number of negative-first routing lists. The following lemma states that a negative-first routing list can be formed from two containment ordered lists. This construction will be exploited in our optimal BRCP algorithm.

Lemma 1 *Let s be a source node, let D denote the set of destinations for source s, and let $L_1 = [n_{1,1}, n_{1,2}, \ldots, n_{1,k}]$ and $L_2 = [n_{2,1}, n_{2,2}, \ldots, n_{2,\ell}]$ denote two containment ordered lists whose nodes are distinct elements of D. If $n_{1,k} \preceq s$ then the list $L = reversal(L_1) \circ L_2$ is a negative-first routing list with respect to s.*

The full proof of this lemma can be found in [14]. A sketch of the proof is given below.

Proof Sketch: Since $L_1 = [n_{1,1}, n_{1,2}, \ldots, n_{1,k}]$ is a containment ordered list and $n_{1,k} \preceq s$, by definition of containment $n_{1,k}$ can be reached from s using channels exclusively in the negative subnetwork. Similarly, $n_{1,k-1}$ can be reached from $n_{1,k}$ and, in general, $n_{1,i-1}$ can be reached from $n_{1,i}$, $2 \leq i \leq k$, using channels exclusively in the negative subnetwork. Thus reversal(L_1) is a negative-first routing list with respect to s. Moreover, $n_{2,1}$, the first element in containment ordered list L_2 can be reached from $n_{1,1}$ by a sequence of zero more edges in the negative subnetwork followed by a sequence of zero or more edges in the positive subnetwork. Next, $n_{2,2}$ can be reached from $n_{2,1}$ and, in general, $n_{2,j}$ can be reached from $n_{2,j-1}$, $2 \leq j \leq \ell$, using edges exclusively in the positive subnetwork.

Algorithm: ***Column-Greedy***
Input: Destination set D in an $m \times n$ mesh and two nodes, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ such that $p_1 \preceq p_2$.
Output: Containment ordered list $L = [n_1, n_2, \dots, n_k]$ such that $p_1 \preceq n_1$ and $n_k \preceq p_2$.

Procedure:

1. Sort destination nodes in D in increasing lexicographic order. (Thus, nodes are sorted by increasing x-coordinates and nodes with equal x-coordinates are further sorted by increasing y-coordinates.)
2. Let L denote the containment ordered list under construction. Initialize $L = \emptyset$.
3. Let y denote the y-coordinate of the last element in L. Initialize $y = y_1$.
4. For each column x, $x_1 \leq x \leq x_2$, for all $d = (x_d, y_d) \in D$ in lexicographic order, if $x_d = x$ and $y_d \geq y$, then append d to L and set $y = y_d$.
5. Return list L.

Figure 2: The Column-Greedy Algorithm.

Algorithm: ***Row-Greedy***
Input: Destination set D in an $m \times n$ mesh and two nodes, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ such that $p_1 \preceq p_2$.
Output: Containment ordered list $L = [n_1, n_2, \dots, n_k]$ such that $p_1 \preceq n_1$ and $n_k \preceq p_2$.

Procedure:

1. Sort destination nodes in D in increasing lexicographic order. (Thus, nodes are sorted by increasing x-coordinates and nodes with equal x-coordinates are further sorted by increasing y-coordinates.)
2. Let L denote the containment ordered list under construction. Initialize $L = \emptyset$.
3. Let x denote the x-coordinate of the last element in L. Initialize $x = x_1$.
4. For each row y, $y_1 \leq y \leq y_2$, for all $d = (x_d, y_d) \in D$ in lexicographic order, if $y_d = y$ and $x_d \geq x$, then append d to L and set $x = x_d$.
5. Return list L.

Figure 3: The Row-Greedy Algorithm.

Therefore, by Definition 8, $L = \text{reversal}(L_1) \circ L_2$ is a negative-first routing list with respect to s. □

The pure negative-first algorithm constructs negative-first routing lists by concatenating the reversal of containment ordered lists with other containment ordered lists as suggested by Lemma 1. Two algorithms are used to construct these lists, the *column-greedy* algorithm and the *row-greedy* algorithm. These algorithms take as input the destination set D in the $m \times n$ mesh and two nodes in the mesh, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ such that $p_1 \preceq p_2$. Both algorithms return a containment ordered list of destinations, L, such that the nodes of L form a subset of D and every node in L contains p_1 and is contained in p_2. Therefore, reversal(L) can be used as a negative-first routing list in the negative subnetwork for any source containing p_2 or L can be used in the positive subnetwork for any source contained in p_1. The column-greedy and row-greedy algorithms are given in Figure 2 and Figure 3, respectively.

As an example of the containment ordered lists constructed by these algorithms, consider the mesh in Figure 5(a). The result of applying the column-greedy algorithm to this destination set is shown in Figure 5(b). The directed edges indicate the order in which the destinations are added to the containment ordered list and suggest one possible negative-first routing path for this containment ordered list. Similarly, the result of applying the row-greedy algorithm to the remaining destinations is shown in Figure 5(c).

The pure negative-first multicast routing algorithm is described in Figure 4. We now consider each step of this algorithm using Figure 5 as an example. The algorithm maintains two sets of containment ordered lists, $S_{positive}$ and S_{either}. The set $S_{positive}$ contains lists that must be routed in the positive subnetwork whereas S_{either} contains lists that can be routed in either the negative subnetwork or the positive subnetwork. The algorithm partitions the destination set D into four sets, NW, NE, SW, and SE, corresponding approximately to those nodes to the northwest, northeast, southwest, and southeast of the source node, as indicated in step 2 of the algorithm. For the example in Figure 5, these sets are $NW = \{(1,4),(1,5),(2,7)\}$, $NE = \{(5,4),(6,5),(6,7),(7,4)\}$, $SW = \{(0,2),(2,0),(2,2),(2,3),(3,1),(3,2),(4,2)\}$, and $SE = \{(5,0),(5,1),(5,2),(7,2)\}$.

Observe that since every node $n \in NW$ is strictly north and west of the source s, any negative-first routing list originating at s and including node n must use some channels in the positive subnetwork. Consequently, while destination nodes remain in the set NW, step 3 repeatedly invokes the column-greedy algorithm to find containment ordered lists that include these destinations and these lists are added to $S_{positive}$. For the mesh in Figure 5(a), we have seen that the first application of the column-greedy algorithm results in the containment ordered list represented in Figure 5(b). Since this list contains all of the destinations in set NW (as well as some additional destinations not in NW), we now enter step 4 of the algorithm.

Step 4 considers those nodes in set SE. Since every node $n \in SE$ is strictly south and east of the source, any negative-first routing list originating at s and including node n must also use channels in the positive subnetwork. Thus, analogously to step 3, the row-greedy algorithm is repeatedly invoked to find containment ordered lists that include these destinations and these lists are also added to set $S_{positive}$. For the mesh in Figure 5(b), we have seen that the first application

Algorithm: ***Pure Negative-First BRCP Multicast***
Input: Source node $s = (s_x, s_y)$ and destinations set D in an $m \times n$ mesh.
Output: Set S of negative-first routing lists.
Procedure:

1. Initialize $S = \emptyset$. In addition, initialize two sets of containment ordered lists $S_{positive} = \emptyset$ and $S_{either} = \emptyset$.

2. Partition D into four sets NW, NE, SW, and SE defined as follows:

$$NW = \{n = (x, y) | x < s_x, y > s_y\}$$

$$NE = \{n = (x, y) | x \geq s_x, y \geq s_y\}$$

$$SW = \{n = (x, y) | x \leq s_x, y \leq s_y\}$$

$$SE = \{n = (x, y) | x > s_x, y < s_y\}$$

3. While destinations remain in set NW use the column-greedy algorithm with $p_1 = (0, 0)$ and $p_2 = (m-1, n-1)$ to construct a containment ordered list L. Add L to $S_{positive}$ and remove all destinations in L from D.

4. While destinations remain in set SE use the row-greedy algorithm with $p_1 = (0, 0)$ and $p_2 = (m-1, n-1)$ to construct a containment ordered list L. Add L to $S_{positive}$ and remove all destinations in L from D.

5. While destinations remain in set NE use the column-greedy algorithm with $p_1 = (0, 0)$ and $p_2 = (m-1, n-1)$ to construct a containment ordered list L. Add L to $S_{positive}$.

6. While destinations remain in set SW use the column-greedy algorithm with $p_1 = (0, 0)$ and $p_2 = (s_x, s_y)$ to construct a containment ordered list L. Add L to S_{either}.

7. While there exists a containment ordered list $L_1 \in S_{either}$ and a containment ordered list $L_2 \in S_{positive}$, add the ordered list reversal$(L_1) \circ L_2$ to S and remove L_1 and L_2 from S_{either} and $S_{positive}$, respectively.

8. While S_{either} contains two or more lists, select two lists $L_1, L_2 \in S_{either}$, remove these two lists from S_{either} and add reversal$(L_1) \circ L_2$ to S. If S_{either} contains only one list, L, remove L from S_{either} and add reversal(L) to S.

9. Otherwise, if $S_{positive}$ is not empty then add each element of $S_{positive}$ to S.

10. Return set S.

Figure 4: The Pure Negative-First BRCP Multicast Algorithm.

of the row-greedy algorithm results in the containment ordered list represented in Figure 5(c). Since this list contains all of the destinations in set SE (as well as some additional destinations not in SE), we now enter step 5 of the algorithm.

Step 5 of the algorithm considers any destinations that still remain in NE. (Note that some nodes originally in NE may have already been included in the containment ordered lists found in the two previous steps of the algorithm.) Since the remaining NE destinations are north and east of the source, they too require paths in the positive subnetwork. The column-greedy algorithm is used (the row-greedy algorithm could also be used here) to construct containment ordered lists that are added to $S_{positive}$. In the example in Figure 5, the destination nodes $(5, 4)$ and $(6, 5)$ still remain after the previous steps. Thus, the column-greedy algorithm finds the containment ordered list shown in Figure 5(d). Since this list contains all remaining destination nodes in NE, we now enter step 6 of the algorithm.

Step 6 of the algorithm considers the remaining destinations in SW. Since these nodes are contained in s, any containment ordered list of such nodes can be routed in the positive subnetwork while the reversal of any such list can be routed in the negative subnetwork. Such lists are therefore added to S_{either}. In the example in Figure 5, the only remaining destinations in SW at this point are nodes $(3, 1)$, $(3, 2)$, and $(4, 2)$ and the resulting negative-first list found by the column-greedy algorithm is shown in Figure 5(e).

In the remaining steps of the algorithm, the containment ordered lists in S_{either} and $S_{positive}$ are assembled into negative-first routing lists. In our example, there exist three containment-ordered lists in $S_{positive}$ and one list in S_{either}. In step 7 of the algorithm, the single list in S_{either} (Figure 5(e)) is reversed and concatenated with an arbitrary list in $S_{positive}$ (such as the list in Figure 5(b)), resulting in the negative-first routing list indicated by an "A" in Figure 6. At this point $S_{positive}$ is empty. When one of the two sets becomes empty, step 7 ends and at at most one of step 8 or step 9 applies, depending on which set still contains lists. In our example, $S_{positive}$ is not empty, so step 9 applies. We now add the remaining containment ordered lists in $S_{positive}$ (those in Figure 5(c) and (d)) directly to S, resulting in the negative-first lists indicated by "B" and "C", respectively, in Figure 6.

Theorem 1 *The Pure Negative-First BRCP Routing Algorithm partitions the destination set into the minimum number of negative-first routing lists.*

The proof of this theorem is omitted in the interest of space. The detailed proof can be found in [14].

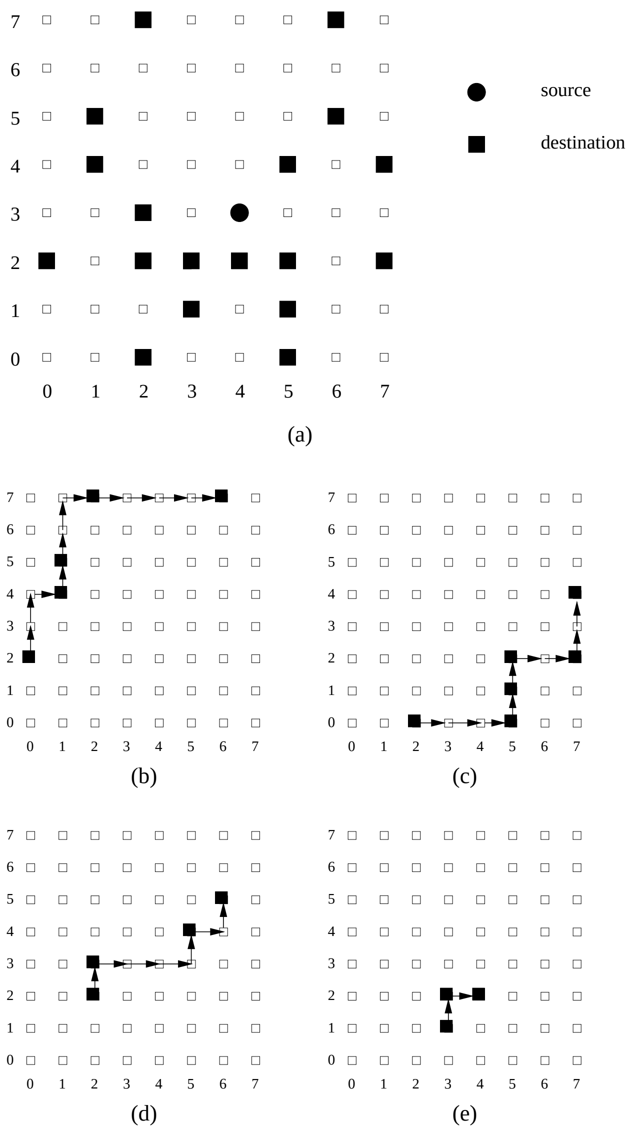

Figure 5: (a) An 8×8 mesh with a source and multiple destinations. (b) The containment ordered list constructed by the column-greedy algorithm. (c) The containment ordered list constructed by the row-greedy algorithm. (d) The next containment ordered list constructed by the column-greedy algorithm. (e) The last containment ordered list constructed by the column-greedy algorithm.

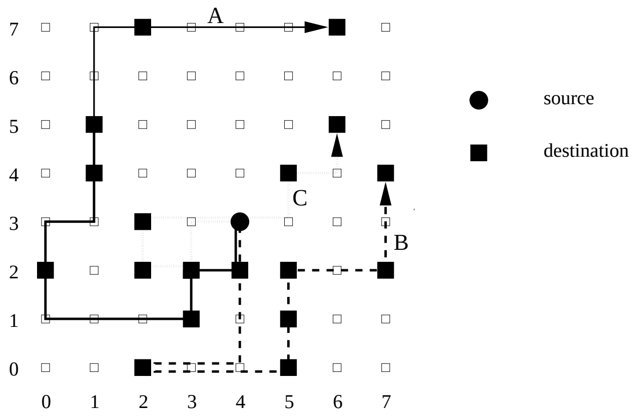

Figure 6: Negative-first routing lists found by the pure negative-first multicast BRCP algorithm.

3.3 Minimal Negative-First BRCP Multicast Routing

We now describe a minimal negative-first BRCP multicast routing algorithm. Given a source node $s = (s_x, s_y)$ and a set D of destinations, consider the partition of D into sets NW, NE, SW, and SE defined in Figure 4. For every destination in $n \in NW$, the only valid minimal path from s to n under negative-first routing is the path that routes west (negative subnetwork) to the column containing the destination and then north (positive subnetwork) to the destination. Thus, the minimal negative-first routing algorithm constructs one negative-first routing list for each column containing at least one destination in NW and the destinations in each such routing list are sorted by increasing y-coordinate. Similarly, for every destination $n \in SE$, the only valid minimal path from s to n under negative-first routing is the path that routes south (negative subnetwork) to the row containing the destination and then east (positive subnetwork) to the destination. Thus, the minimal negative-first routing algorithm constructs one negative-first routing list for each row containing at least one destination in SE and the destinations in each such routing list are sorted by increasing x-coordinate.

For every destination in $n \in SW$, the set of minimal paths from s to n is exactly the set of paths that use only the negative subnetwork. Similarly, for every destination in $n \in NE$, the set of minimal paths from s to n is exactly the set of paths that use only the positive subnetwork. The proof technique used to prove Theorem 1 implies that either the column-greedy or the row-greedy algorithms can be used to construct the minimum number of negative-first routing lists for destinations in SW and NE. The algorithm is summarized in Figure 7 and an example of the paths found by the algorithm is shown in Figure 8.

Theorem 2 *The Minimal Negative-first BRCP Rout-*

Algorithm: ***Minimal Negative-First BRCP Multicast***
Input: Source node $s = (s_x, s_y)$ and destinations set D in an $m \times n$ mesh.
Output: Set S of negative-first routing lists.
Procedure:

1. Initialize $S = \emptyset$.

2. Partition D into four sets NW, NE, SW, and SE defined as follows:

$$NW = \{n = (x, y) | x < s_x, y > s_y\}$$

$$NE = \{n = (x, y) | x \geq s_x, y \geq s_y\}$$

$$SW = \{n = (x, y) | x \leq s_x, y \leq s_y\}$$

$$SE = \{n = (x, y) | x > s_x, y < s_y\}$$

3. For each column x containing a destination in NW, sort the destinations in NW in column x by increasing y coordinate. Add this negative-first routing list to S and remove these destinations from NW.

4. For each row y containing a destination in SE, sort the destinations in SE in row y by increasing x coordinate. Add this negative-first routing list to S and remove these destinations from SE.

5. While destinations remain in SW use the column-greedy algorithm to construct a negative-first routing list. Specifically, use the column-greedy algorithm with $p_1 = (0, 0)$ and $p_2 = (s_x, s_y)$. The resulting list L is containment-ordered and thus reversal(L) is a negative-first routing list with respect to s. Add reversal(L) to S and remove these destinations from SW.

6. While destinations remain in NE use the column-greedy algorithm to construct a negative-first routing list. Specifically, use the column-greedy algorithm with $p_1 = (s_x, s_y)$ and $p_2 = (m - 1, n - 1)$. Add the resulting list L to S and remove these destinations from NE.

7. Return set S.

Figure 7: The Minimal Negative-First BRCP Multicast Algorithm.

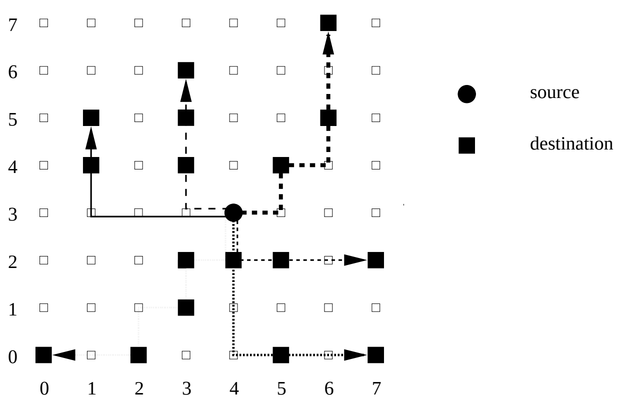

Figure 8: Negative-first routing lists found by the minimal negative-first multicast BRCP algorithm.

ing Algorithm partitions the destination set into the minimum number of negative-first routing lists such that each destination in a list is reached on a minimal path.

The proof of this theorem can be found in [14].

4 Experimental Results

In this section we describe simulation results comparing the negative-first BRCP multicast algorithms to several existing multicast algorithms. The simulations were conducted using the Mesh Architecture Routing Simulator (MARS) developed at Harvey Mudd College. MARS is an event-driven simulation package written in C++ and has an X-windows-based graphical user interface that allows visualization of network traffic and other network characteristics.

Except where indicated otherwise, the following system parameters were used in these experiments. The mesh size was 16×16 and each node had 4 injection channels and 4 consumption channels. The communication startup latency was 5 microseconds, link propagation time was 5 nanoseconds, and router latency was 20 nanoseconds. Messages were assumed to comprise a number of data flits selected uniformly from 10 to 100. Messages were injected into the network according to a negative binomial distribution. Each data point indicates the mean of at least 100 runs with additional runs used if necessary to ensure that the 95% confidence interval widths were within 5% of the mean.

The first set of experiments measure communication latency as a function of the average arrival rate for the two negative-first BRCP algorithms, the Hamiltonian path-based algorithm, and the column path algorithm. Figure 9(a), (b), and (c) shows these results for messages with 10, 20, and 30 destinations, respectively. In each case, the Hamiltonian path-based algorithm enjoys the lowest communication latencies at very low loads since it requires no more than two startups per message. However, at moderate and higher loads the large number of routing resources held by each message in the Hamiltonian path-based algorithm result in very high latencies. The column path algorithm incurs the highest communication latency at low loads due to the large number of startups required by this algorithm. At all but the very lowest loads, the two negative-first routing algorithms enjoy the best performance. Communication latency for the pure negative-first algorithm is initially lower due to the relatively small number of startups required while the performance of the minimal negative-first algorithm is superior at higher loads where the network and blocking latencies began to dominate the communication latency for the pure algorithm.

Figure 10 demonstrates the behavior of these algorithms for even larger numbers of destinations. Here

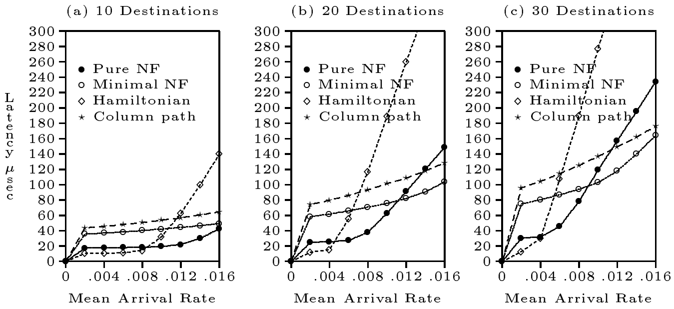

Figure 9: Latency versus arrival rates (in μsecs) for (a) 10, (b) 20, and (c) 30 destinations per message in a 16×16 mesh.

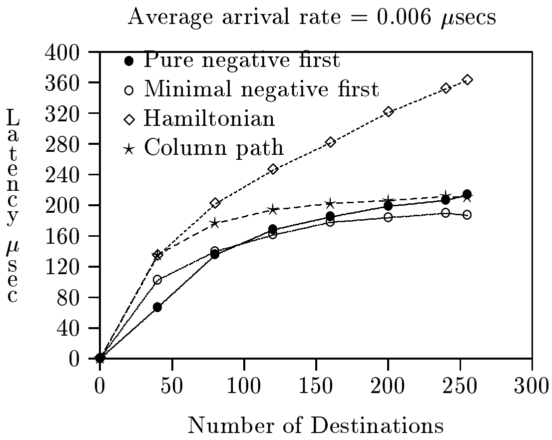

Figure 10: Latency versus number of destinations in a 16×16 mesh.

the average arrival rate is fixed (0.006) and the number of destinations ranges from 0 to 255. The two negative-first BRCP schemes again outperform both the Hamiltonian path-based algorithm and the column path algorithm. Observe that for large destination sets, the minimal negative-first algorithm again incurs lower communication latency than the pure negative-first algorithm in spite of the larger number of startups.

In the next set of experiments, we consider the *multiple multicast* traffic pattern in which multiple sources send multicast messages simultaneously. Multiple multicast operations arise in cache-invalidation in distributed shared memory systems, concurrent barrier operations, among other applications [8]. Kesavan and Panda have recently proposed several algorithms for multiple multicast [8]. Among these algorithms, the *source centered hierarchical leader (SCHL)* algorithm was shown to be the most effective. SCHL is a hierarchical BRCP algorithm whose underlying routing algorithm is e-cube. Since SCHL is hierarchical, the source node incurs only a small number of startups. However, since the source node for one message

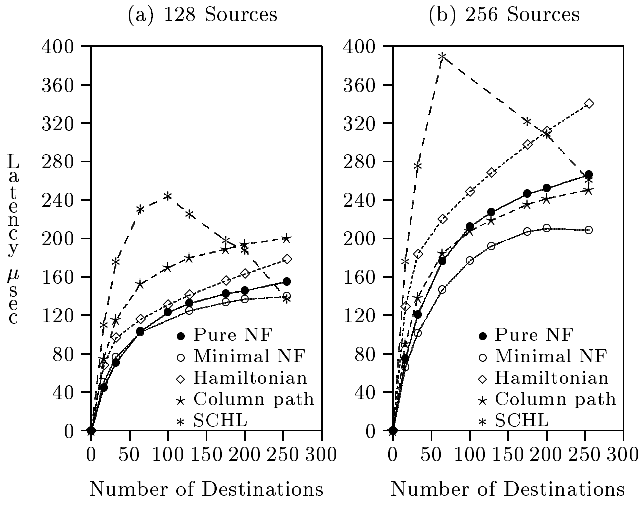

Figure 11: Latency versus number of destinations for multiple multicast in a 16×16 mesh.

may be required to participate in retransmitting other messages, the actual number of startups in such hierarchical schemes can in fact be higher than in pure and minimal BRCP schemes. For example, the maximum and mean number of startups required by each of the algorithms for all-to-all broadcast in a $n \times n$ mesh are shown in Table 1. The pure negative-first algorithm, for example, requires less than half as many startups as the SCHL algorithm in this case.

Experimental results for multiple multicast with 128 sources and 256 sources and varying numbers of destinations from 0 to 255 are shown in Figure 11. Each source receives a random set of destinations. In these experiments, the system parameters were modified to be consistent with those used in [8]. Specifically, each channel had only one injection channel, the messages were 50 flits long, and crossbar and injection latencies of 5 nanoseconds each were included. Note that communication latencies under SCHL begin decreasing at a certain point since additional destinations can be used to participate in absorbing and retransmitting the message, thereby reducing the total startup latency [8].

5 Conclusions and Future Research

In this paper we have considered two classes of adaptive multicast routing algorithms: pure BRCP and minimal BRCP algorithms. We constructed both pure and minimal BRCP algorithms with respect to negative-first routing and demonstrated their effectiveness experimentally. The pure BRCP algorithm was shown to incur lower communication latencies at low loads due to the relatively small number of startups incurred, while the minimal BRCP algorithm performed

	Pure Neg-First	Min Neg-First	Hamiltonian	Col-Path	SCHL
Max	n	$3n-2$	2	$2n$	$2n+7$
Mean	$\frac{41}{48}n - \frac{3}{8} - \frac{1}{6n}$	$\frac{5}{3}n - 2 + \frac{4}{3n}$	$2 - \frac{2}{n^2}$	$2n-2$	$2n + 4 - \frac{12}{n} + \frac{4}{n^2}$

Table 1: Maximum and mean number of startups for all-to-all broadcast.

better at higher loads where network and blocking latencies dominated communication latency.

Pure and minimal multicast routing algorithms based on fully adaptive unicast routing algorithms require virtual channels but are likely to result in even better performance. We are currently investigating such algorithms. Note also that it is possible to use a pure and a minimal algorithm simultaneously in the same network, if they are constructed with respect to the same deadlock-free unicast routing algorithm. Thus, a more sophisticated algorithm might allow each node to choose either the pure algorithm or the minimal algorithm, or even somthing in between, depending on factors such as the number of destinations and approximate network load. Finally, we are also considering fault-tolerant versions of pure and minimal adaptive routing algorithms.

Acknowledgements

The authors wish to thank Professor D. K. Panda for the experimental results for the SCHL algorithm.

References

[1] M. Azevedo and D. Blough. Fault-tolerant clock synchronization of large multicomputers via multidimensional interactive convergence. Technical Report ECE 94-12-03, University of California, Irvine, Dec. 1994.

[2] R. Boppana, S. Chalasani, and C. Raghavendra. On multicast routing in multicomputers. In *Proc. 1994 IEEE Symp. on Parallel and Distributed Processing*, 1994.

[3] L. De Coster, N. Dewulf, and C-T. Ho. Efficient multipacket multicast algorithms on meshes with wormhole and dimension-ordered routing. In *Proc. of the Int. Conf. on Parallel Processing*, pages 137–141, Aug. 1995.

[4] W. Dally. Virtual channel flow control. In *Proc. of the 17th Int. Symp. on Computer Architecture*, pages 60–68, May 1990.

[5] J. Duato. A theory of deadlock-free adaptive multicast routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(9):976–987, Sept. 95.

[6] E. Fleury and P. Fraigniaud. Strategies for multicasting in meshes. In *Proc. 23rd Int. Conf. on Parallel Processing (ICPP '94)*, Aug. 1994.

[7] C. Glass and L. Ni. Adaptive routing in mesh-connected networks. In *Proc. 12th Int. Conf. on Distributed Computing Systems*, pages 12–19, June 1992.

[8] R. Kesavan and D. Panda. Minimizing node contention in multiple multicast on wormhole k-ary n-cube networks. In *Proc. of the Int. Conf. on Parallel Processing*, Aug. 1996.

[9] F. T. Leighton. *Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes.* Morgan Kaufmann, 1992.

[10] K. Li and R. Schaefer. A hypercube shared virtual memory. In *Proc. 1989 Int. Conf. Parallel Processing*, volume 3, pages 125–132, 1989.

[11] X. Li, P. McKinley, and A-H. Esfahanian. Adaptive multicast wormhole routing in 2D mesh multicomputers. In *Proc. 1993 Parallel Architectures and Languages Europe Conference (PARLE'93)*, pages 228–241, June 1993.

[12] X. Li, P. McKinley, and L. Ni. Deadlock-free multicast wormhole routing in 2-D mesh multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, 5(8):793–804, Aug. 1994.

[13] R. Libeskind-Hadas and E. Brandt. Origin-based fault-tolerant routing in the mesh. In *Proc. of the First IEEE Symposium on High-Performance Computer Architecture*, pages 102–111, Jan. 1995.

[14] R. Libeskind-Hadas and A. Hutchings. Proofs of correctness and analysis of the pure and minimal negative-first multicast routing algorithms. Technical report, Harvey Mudd College, July 1994.

[15] P. McKinley, H. Xu, A-H. Esfahanian, and L. Ni. Unicast-based multicast communication in wormhole-routed networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(12):1252–1265, Dec. 1994.

[16] L. Ni and P. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer*, 26(2):62–76, Feb. 1993.

[17] D. Panda, S. Singal, and P. Prabhakaran. Multidestination message passing mechanism conforming to base wormhole routing scheme. In *Proc. 1994 Parallel Computer Routing and Communication Workshop*, number 853 in Lecture Notes in Computer Science, pages 131–145. Springer-Verlag, 1994.

[18] Y. Tsai and P. McKinley. A dominating set model for broadcast in all-port wormhole-routed 2D mesh networks. In *Proc. Eighth ACM Int. Conf. on Supercomputing*, July 1994.

[19] Y-C Tseng and S. K. S. Gupta. All-to-all personalized communication in a wormhole-routed torus. *IEEE Transactions on Parallel and Distributed Systems*, 7(6):498–505, May 1996.

[20] H. Xu, P. McKinley, and L. Ni. Efficient implementation of barrier synchronization in wormhole-routed hypercube multicomputers. *Journal of Parallel and Distributed Computing*, 16:172–184, 1992.