Optimal Contention-Free Unicast-Based Multicasting in Switch-Based Networks of Workstations*

Ran Libeskind-Hadas   Dominic Mazzoni   Ranjith Rajagopalan
Department of Computer Science
Harvey Mudd College
Claremont, CA 91711

Abstract

A unicast-based multicasting algorithm is presented for arbitrary interconnection networks arising in switch-based networks of workstations. The algorithm is optimal with respect to the number of startups incurred and is provably free from depth contention. Specifically, no two constituent unicasts for the same multicast contend for a common channel, even if some unicasts are delayed due to unpredictable variations in latencies. The algorithm uses an underlying partially adaptive deadlock-free unicast routing algorithm. Simulation results indicate that the algorithm behaves as predicted by its theoretical properties and provides a promising approach to unicast-based multicasting.

1 Introduction

In this paper we consider the problem of designing efficient multicast routing algorithms for wormhole-routed switch-based networks of workstations (NOWs). These networks comprise a collection of routing switches and workstations interconnected in some arbitrary topology. Currently, switches support only unicast (one-to-one) communication in hardware. However, multicast (one-to-many) communication is required in many applications as well as in a variety of system-level functions such as barrier synchronization, cache coherency in distributed shared-memory architectures, and clock synchronization, among others.

Since existing switches support only unicast communication in hardware, multicasting is currently supported by sending multiple unicast messages. In *separate addressing*, the source node simply sends a separate unicast message to each of the destination nodes. However, since message startup time is generally several orders of magnitude larger than network latency in wormhole-routed networks, it is desirable to minimize the number of startup phases or *communication steps* used to deliver the message to all of its destinations. An alternative approach, therefore, is to use a *multicast tree algorithm* [9] in which the source node sends the message to some subset of the destination nodes which then participate in distributing the message to other destinations until, eventually, all destinations receive the message.

A unicast-based multicast algorithm should have several properties [6]. First, the underlying unicast routing algorithm must be deadlock-free. Second, the number of communication steps required to deliver the message to all of its destinations should be minimized. Third, the constituent unicast messages of any single multicast should not contend among themselves for communication channels.

In this paper we describe a unicast-based multicast algorithm for arbitrary topologies which employs deadlock-free unicast routing, achieves the theoretical lower bound on the number of communication steps, and is provably free from contention among unicasts belonging to the same multicast message. To the best of our knowledge, this is the first algorithm for arbitrary topologies that satisfies all three of these properties.

The remainder of this paper is organized as follows. In Section 2 we review related work on unicast-based multicast algorithms. In Section 3 we describe our new multicast tree algorithm and prove that it has the properties described above. Section 4 describes simulation results and we conclude in Section 5.

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

2 Related Work

Although a number of hardware-supported multicast algorithms have been proposed recently for wormhole-routed switch-based networks, efficient software-supported algorithms are required to implement multicast communication in hardware which supports only unicast communication. In this section we describe recent results on unicast-based multicast routing algorithms.

It is easily verified that the theoretical lower bound on the number of communication steps required to complete a multicast to d destinations using unicast-based routing is $\lceil \log_2(d+1) \rceil$ [6]. Thus, any unicast-based algorithm that uses exactly $\lceil \log_2(d+1) \rceil$ communication steps is said to be *optimal*. Although in general it is not possible to avoid contention for channels among unicast messages belonging to different multicasts, in some cases scheduling techniques have been employed to guarantee freedom from contention among the constituent unicasts of the same multicast. *Step contention* occurs when two unicast messages in the same communication step contend for a common unidirectional channel [6]. However, since the nodes are not synchronized, a node scheduled to transmit the message in communication step i may become delayed and transmit the message at a later communication step, possibly contending with a unicast for the same message transmitted at communication step j, $j \geq i$. This type of contention, called *depth contention*, is a generalization of step contention. Thus, guaranteeing depth contention-freedom is stronger than guaranteeing step-contention freedom.

McKinley *et al.* described optimal depth contention-free unicast-based routing algorithms for meshes and hypercubes employing dimension-ordered routing [6]. De Coster *et al.* [3] described a class of depth contention-free algorithms for meshes using dimension-ordered routing that generalizes the algorithm in [6]. Robinson *et al.* subsequently proposed an optimal depth contention-free unicast-based routing algorithm for the torus [7]. Kesavan *et al.* described several unicast-based depth contention-free algorithms for arbitrary topologies that require a fairly small number of communication steps, but are not optimal [4]. Birchler *et al.* described a sufficient condition for optimal step contention-free communication in any topology [1].

In this paper we describe an optimal depth contention-free unicast-based multicast routing algorithm for arbitrary topologies. The algorithm is simple to implement and uses an underlying deadlock-free unicast routing algorithm that has been successfully employed in an existing switch-based network [8].

3 Depth Contention-Free Routing in Arbitrary Topologies

In this section we describe the depth contention-free multicast routing algorithm for arbitrary topologies. We begin in Subsection 3.1 with definitions and fundamental results related to depth contention. The algorithm and the proof of depth contention-freedom are given in Subsection 3.2.

3.1 Definitions and Fundamentals

A switch-based interconnection network can be represented by a directed graph $G = (V, E)$ where $V = V_1 \cup V_2$. The set V_1 represents the set of switches and the set V_2 represents the set of processors. In order to correctly model existing switch-based networks such as the DEC Autonet [8] and Myrinet [2], we assume that the graph is *symmetric*: For each directed edge $(u, v) \in E$ there exists a corresponding edge $(v, u) \in E$. In both Autonet and Myrinet, these symmetric channels can be used simultaneously with no contention [2, 8].

Since the graph is symmetric, vertex u is said to be *connected* to vertex v if there exists a pair of oppositely directed edges between the two vertices. Similarly, since the indegree of a vertex equals the outdegree of a vertex in a symmetric graph, the *degree* of a vertex refers to indegree or outdegree.

Each vertex in V_2 is connected to a single vertex in V_1, representing a connection between a processor and exactly one switch. In addition, pairs of vertices in V_1 may be connected, representing connections between pairs of switches. The degree of a vertex in V_1 is bounded by the number of ports in the corresponding switch. The degree of each vertex in V_2 is exactly 1.

A *directed trail* in the network is an alternating sequences of vertices and edges $v_0, e_0, v_1, e_1, \cdots v_{k-1}, e_{k-1}, v_k$, such that $e_i = (v_i, v_{i+1}) \in E$ and all edges are distinct. A *directed path* is a directed trail in which all vertices are distinct. A *directed circuit* is a directed trail in which the start and end vertices coincide. The definitions below from McKinley *et al.* [6] are needed to formalize the notion of depth contention. They are included here for completeness.

A unicast operation is defined as an ordered quadruple $(u, v, P(u, v), t)$ where u and v are the source and destination nodes, $P(u, v)$ is a directed path from u to v, and t is the positive integer communication step at which the unicast begins. Although vertices u and v can be in either V_1 or V_2, to simplify our discussion we assume that all sources and destinations are in V_2, representing messages between processors.

Two unicasts, $(u, v, P(u,v), t)$ and $(x, y, P(x,y), \tau)$, with $t = \tau$, are *feasible* if u, v, x, and y are all distinct. A *feasible unicast set* is a set of unicasts whose members are pairwise feasible. A multicast request can be represented by a set $M = \{v_0, v_1, \cdots, v_d\}$, where node v_0 is the source of the multicast and the other nodes are its destinations.

Definition 1 (McKinley, Xu, Esfahanian, and Ni) *An implementation $I(M)$ of a unicast-based multicast request M is a sequence of feasible unicast sets $U_1, U_2, \cdots, U_k$ satisfying the following conditions.*

1. *For each $j, 1 \le j \le k$, if $(u, v, P(u,v), j) \in U_j$, then both u and v belong to M.*
2. *The first unicast set is $U_1 = \{(v_0, u, P(v_0, u), 1)\}$, where $u = v_i$ for some $i, 1 \le i \le d$.*
3. *For every unicast $(u, v, P(u,v), t) \in U_t, 1 < t \le k$, there must exist a set U_j with $j < t$ which has $(w, u, P(w,u), j)$ as a member for some node w.*
4. *For every destination $v_i, 1 \le i \le d$, there exists one and only one integer j such that $1 \le j \le k$ and $(w, v_i, P(w, v_i), j)$ appears in U_j for some node w.*

Definition 2 (McKinley, Xu, Esfahanian, and Ni) *Given a multicast implementation $I(M) = \{U_1, U_2, \ldots, U_k\}$, a node v is in the* reachable set *of a node u, denoted R_u, if and only if $v = u$ or there exists a $j, 1 \le j \le k$, such that $(w, v, P(w,v), j) \in U_j$ for some node $w \in R_u$.*

A multicast implementation is said to be *depth contention-free* if no two constituent unicast messages of the multicast implementation contend for the same channel, even if one or both of them are sent after their scheduled communication steps due to delays. Observe that if two unicast paths are edge-disjoint then they cannot contend for a common channel, regardless of the communication steps in which the unicasts are sent. However, two unicasts paths that use a common channel may still be depth-contention free. For example, if one unicast delivers the message from u to v and another unicast delivers the message from v to y, then the first unicast will exit the network before the second one enters. More generally, if x is the source of the second unicast and x is in the reachable set of v or is in the reachable set of some node w which receives a unicast from u after the unicast from u to v has completed, then the unicast from u to v and the unicast originating from x cannot contend for channels. The following theorem formalizes these observations, providing a sufficient condition for a multicast implementation to be depth contention-free.

Theorem 1 (McKinley, Xu, Esfahanian, and Ni) *Given a multicast implementation $I(M)$, if at least one of the following four conditions holds for every pair of unicasts $(u, v, P(u,v), t)$ and $(x, y, P(x,y), \tau)$ in $I(M)$, where $t \le \tau$, then $I(M)$ is depth contention-free.*

1. *$x \in R_v$.*
2. *$P(u,v)$ and $P(x,y)$ are edge-disjoint.*
3. *$x = u$.*
4. *$x \in R_w$ and $(u, w, P(u,w), t + \ell) \in I(M)$, for some node w and positive integer ℓ.*

3.2 The Depth Contention-Free Algorithm

In this subsection we present an optimal depth contention-free multicasting algorithm for arbitrary topologies. This algorithm uses the deadlock-free up*/down* algorithm described by Schroeder *et al.* and employed in the DEC Autonet [8]. The up*/down* algorithm uses a rooted breadth-first search (BFS) spanning tree in the network. An edge (u, v) is in the *up subnetwork* if the level of u is larger than the level of v or, if u and v are at the same level, then the ID of u is larger than the ID of v. All other edges are in the *down subnetwork*. The up*/down* routing algorithm requires that all routing be completed in the up subnetwork before using any edges in the down subnetwork. The algorithm is easily verified to be deadlock-free [8].

While the original up*/down* algorithm only requires that each node have a unique ID, our algorithm requires a specific choice of ID's in order to guarantee depth contention-freedom. Specifically, we perform a postorder traversal of the breadth-first search tree and use the postorder label of each node as its ID. We henceforth use the notation ID(v) to denote the postorder numbering of vertex v.

Edges in the network can be partitioned into *tree edges* and *cross edges* depending on whether they are in the underlying breadth-first search tree or not. We define two types of routes with respect to a fixed BFS tree.

Definition 3 *A* strict up-first path *from u to v, denoted $P_1(u,v)$, is a directed path $(v_0, e_0, v_1, e_1, \cdots, e_{k-1}, v_k)$ such that $u = v_0$, $v = v_k$, each edge is in the BFS tree, and if e_i is an up edge, then for all $j \leq i$, e_j is an up edge.*

Thus, a strict up-first path is the unique path between two vertices that uses only tree edges. For example, Figure 1 shows a symmetric network with tree edges indicated by solid lines, cross edges indicated by dashed lines, and vertices labeled with postorder ID's. The root of the tree is vertex 8. The strict up-first path from vertex 1 to vertex 4 visits vertices in the sequence $1, 2, 8, 7, 5, 4$.

Definition 4 *A* relaxed up-first path *from u to v, denoted $P_2(u,v)$, is a directed path $(v_0, e_0, v_1, e_1, \cdots, e_{\ell-1}, v_\ell)$ such that $u = v_0$, $v = v_\ell$, if e_i is in an up edge then for all $j \leq i$, e_j is an up edge, and if $e_i = (v_i, v_{i+1})$ is a cross edge then v_i and v_{i+1} are both in the strict up-first path from u to v and v_i precedes v_{i+1} in the the strict up-first path.*

Thus, a relaxed up-first path from u to v is a path from u to v that uses only cross edges whose endpoints occur in the same relative order in the unique strict up-first path from u to v. For example, for the network in Figure 1 one relaxed up-first path from vertex 1 to vertex 4 visits vertices in the sequence $1, 2, 5, 4$. The cross edge from vertex 2 to vertex 5 is allowed because because vertex 2 occurs before vertex 5 on the strict up-first path from vertex 1 to vertex 4. However, the path $1, 2, 8, 3, 7, 5, 4$ is not a valid relaxed up-first path because vertex 3 does not appear on the strict up-first path from vertex 1 to vertex 4.

The optimal depth contention-free multicast algorithm uses the multicast tree technique as follows. Let M denote a multicast request with a source s and d destination nodes. The set $M - \{s\}$ is partitioned into two sets, M_1 and M_2 such that M_1 contains all destinations in M whose postorder ID's are greater than the postorder ID of s and M_2 contains the destinations in M whose postorder ID's are less than the postorder ID of s. A list L is constructed in which s is the first element, followed by the elements in M_1 sorted by increasing postorder ID's, followed by the elements in M_2 sorted by increasing postorder ID's. Let $L = v_0, v_1, \ldots, v_d$ where $s = v_0$. In the first communication step, v_0 sends the message to the node at the midpoint of the list, $v_{\lceil \frac{d+1}{2} \rceil}$, using any relaxed up-first path from v_0 to $v_{\lceil \frac{d+1}{2} \rceil}$. Now, v_0 is responsible for

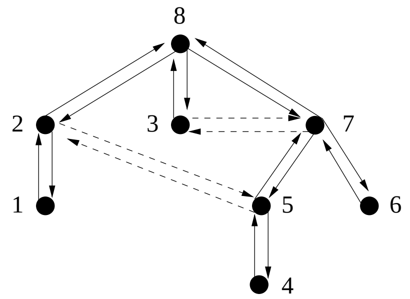

Figure 1: A symmetric network with tree edges indicated by solid lines, cross edges indicated by dashed lines, and vertices labeled with postorder ID's.

delivering the message to the first half of the nodes in L while $v_{\lceil \frac{d+1}{2} \rceil}$ is responsible for delivering the message to the second half of the nodes in L. Each of these two nodes recursively applies the algorithm by sending the message to the midpoint of each of their respective sublists using relaxed up-first paths. This process continues until after $\lceil \log_2(d+1) \rceil$ iterations every destination has received the message. This algorithm is referred to henceforth as the *postorder recursive doubling algorithm.*

Figure 2 shows the multicast tree that results from this algorithm when vertex 3 in Figure 1 broadcasts a message to all of the other vertices. The list L in this case is $3, 4, 5, 6, 7, 8, 1, 2$. Thus, in the first communication step vertex 3 sends the message to vertex 7 using any relaxed up-first path (which in this example is either the direct edge from vertex 3 to vertex 7 or the strict-up first path $3, 8, 7$.). In the second communication step, vertex 3 sends the message to the midpoint of its remaining sublist, vertex 5, while vertex 7 sends the message to the midpoint of its sublist, vertex 1. Finally, in the third communication step all vertices have received the message.

Theorem 2 *The postorder recursive doubling algorithm using relaxed up-first paths is an optimal depth contention-free unicast-based multicast algorithm.*

The proof of Theorem 2 proceeds in several steps. We begin by restricting our attention to the special case that all unicast messages are sent along strict up-first paths. Several lemmas lead to Theorem 3 which shows that for this special case, the postorder recursive doubling algorithm is depth contention-free. We then use this result to prove Theorem 2.

Lemma 1 *Let $(u, v, P_1(u,v), t)$ be a unicast belonging to a multicast $I(M)$ using the postorder recur-*

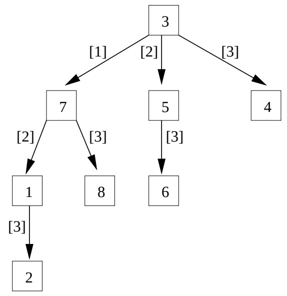

Figure 2: The multicast tree corresponding to vertex 3 broadcasting to the remaining vertices in Figure 1. Numbers in brackets represent scheduled communication steps.

sive doubling algorithm. Let y be any destination in the multicast request M. Then $y \in R_w$ and $(u, w, P_1(u, w), t + \ell) \in I(M)$ for some node w and positive integer ℓ if either of the following conditions hold.

1. *ID(u) < ID(y) < ID(v).*
2. *ID(v) < ID(u) and either ID(y) < ID(v) or ID(u) < ID(y).*

Proof: By definition of the postorder recursive doubling algorithm, if u sends a unicast to v at communication step t and y is a destination with ID(u) < ID(y) < ID(v) then at some time $t + \ell$, $\ell \geq 1$, either u sends the message directly to y or u sends the message to some intermediate node w and $y \in R_w$. An analogous argument applies for the second case. □

Definition 5 *A* postorder circuit *is a circuit $C = (v_0, e_0, v_1, e_1, \ldots, e_{n-1}, v_n)$, which follows a postorder traversal of the graph G starting from the root, v_0, where $v_0 = v_n$.*

Note that each directed edge in the BFS tree appears exactly once in the postorder circuit. However, a vertex may occur several times in the postorder circuit. For example, the postorder circuit for the tree in Figure 1 visits the vertices in the order 8, 2, 1, 2, 8, 3, 8, 7, 5, 4, 5, 7, 6, 7, 8.

Definition 6 *Let $C = (v_0, e_0, \cdots, e_{n-1}, v_n)$ be a postorder circuit and let $v \in V$. The* exit point *of vertex v is the element $v_i \in C$ such that $v_i = v$ and $v_j \neq v$ for all $j > i$.*

Observe that the exit point of a vertex indicates the index in the postorder traversal at which the vertex received its ID.

Definition 7 *A* postorder trail *from u to v, denoted $t(u, v)$, is the subsequence of the postorder circuit C beginning at the exit point of u and ending at the exit point of v. Thus, if ID(u) < ID(v), then the postorder trail is the subsequence of C $(v_i, e_i, v_{i+1}, e_{i+1}, \ldots, e_{j-1}, v_j)$, where v_i and v_j are the exit points of u and v, respectively. If ID(u) > ID(v), then the postorder trail has the form $(v_i, e_i, \ldots, v_{n-1}, e_0, v_0, \ldots, e_{j-1}, v_j)$, where v_i and v_j are the exit points of u and v, respectively.*

Lemma 2 *Given four vertices u, v, x, y the postorder trail $t(u, v)$ is edge-disjoint from the postorder trail $t(x, y)$ if one of the following conditions holds:*

1. *ID(u) < ID(v) < ID(x) < ID(y).*
2. *ID(y) < ID(u) < ID(v) < ID(x).*

Proof:

Case 1: ID(u) < ID(v) < ID(x) < ID(y). The postorder trail $t(u, v)$ has the form $(v_j, e_j, \ldots, e_{k-1}, v_k)$, where v_j is the exit point of vertex u and v_k is the exit point of v. The postorder trail $t(x, y)$ has the form $(v_\ell, e_\ell, \ldots, e_{m-1}, v_m)$, where v_ℓ is the exit point of vertex x and v_m is the exit point of vertex y. Assume that there exists an edge e_p which occurs in both $t(u, v)$ and $t(x, y)$. Then $j \leq p < k$ and $\ell \leq p < m$. However, $k < \ell$ since ID(v) < ID(x), implying that $p < k < \ell \leq p$, a contradiction. Thus, the two trails are edge disjoint.

Case 2: This case is analogous to Case 1 and is omitted due to space limitations. □

Lemma 3 *If postorder trails $t(u, v)$ and $t(x, y)$ are edge-disjoint then the strict up-first paths $P_1(u, v)$ and $P_1(x, y)$ are also edge-disjoint.*

Proof: Consider the path from u to v constructed by removing circuits from trail $t(u, v)$ until no circuits remain. Since there is a unique path between any two vertices in a tree, this path is necessarily $P_1(u, v)$. Thus, $P_1(u, v)$ uses a subset of the edges in $t(u, v)$. Similarly, $P_1(x, y)$ uses a subset of the edges in $t(x, y)$. Since $t(u, v)$ and $t(x, y)$ are edge-disjoint, $P_1(u, v)$ and $P_1(x, y)$ are necessarily edge-disjoint. □

Next we show that if we restrict all routing to use strict up-first paths, then the postorder recursive doubling algorithm is depth contention-free. Afterwards, we use this result to show that the algorithm remains depth contention-free even if relaxed up-first paths are used.

Theorem 3 *The postorder recursive doubling algorithm is depth contention-free if all paths are strict up-first paths.*

Proof: Consider a multicast implementation of the postorder recursive doubling algorithm in which all routing is performed using strict up-first paths. Let $(u, v, P_1(u, v), t)$ and $(x, y, P_1(x, y), \tau)$ denote two unicasts in this implementation where $t \leq \tau$. If $\mathrm{ID}(u)$, $\mathrm{ID}(v)$, and $\mathrm{ID}(x)$ are not distinct then since $\mathrm{ID}(u) \neq \mathrm{ID}(v)$, either $\mathrm{ID}(u) = \mathrm{ID}(x)$ or $\mathrm{ID}(v) = \mathrm{ID}(x)$ In the first case, $x = u$, satisfying condition 3 of Theorem 1. In the second case, $x = v$ and thus $x \in R_v$, satisfying condition 1 of Theorem 1. The remaining cases are that $\mathrm{ID}(u) \neq \mathrm{ID}(v) \neq \mathrm{ID}(x)$.

If $\mathrm{ID}(u) < \mathrm{ID}(v) < \mathrm{ID}(x)$ then either $\mathrm{ID}(y) < \mathrm{ID}(u) < \mathrm{ID}(v) < \mathrm{ID}(x)$ or $\mathrm{ID}(u) < \mathrm{ID}(v) < \mathrm{ID}(x) < \mathrm{ID}(y)$. To see this, observe that if $\mathrm{ID}(u) < \mathrm{ID}(y) < \mathrm{ID}(v) < \mathrm{ID}(x)$ then by Lemma 1 u sends the message to some node w and $y \in R_w$. Therefore, $\mathrm{ID}(x) < \mathrm{ID}(y)$, a contradiction. If $\mathrm{ID}(u) < \mathrm{ID}(v) < \mathrm{ID}(y) < \mathrm{ID}(x)$ then by Lemma 1 x sends the message to y before u sends the message to v, contradicting the assumption that $t \leq \tau$. Since $\mathrm{ID}(y) < \mathrm{ID}(u) < \mathrm{ID}(v) < \mathrm{ID}(x)$ or $\mathrm{ID}(u) < \mathrm{ID}(v) < \mathrm{ID}(x) < \mathrm{ID}(y)$, by Lemma 2, the postorder trails $t(u, v)$ and $t(x, y)$ are edge-disjoint. Thus, by Lemma 3, the strict up-first paths $P_1(u, v)$ and $P_1(x, y)$ are edge disjoint, satisfying condition 2 of Theorem 1. Analogous arguments apply for cases $\mathrm{ID}(x) < \mathrm{ID}(u) < \mathrm{ID}(v)$ and $\mathrm{ID}(v) < \mathrm{ID}(x) < \mathrm{ID}(u)$.

If $\mathrm{ID}(u) < \mathrm{ID}(x) < \mathrm{ID}(v)$ then, by Lemma 1, $x \in R_w$ and $(u, w, P_1(u, w), t + \ell)$ for some node w and positive integer ℓ. Thus, this case satisfies condition 4 of Theorem 1. Cases $\mathrm{ID}(x) < \mathrm{ID}(v) < \mathrm{ID}(u)$ and $\mathrm{ID}(v) < \mathrm{ID}(u) < \mathrm{ID}(x)$ similarly satisfy condition 4 of Theorem 1. □

We now use Theorem 3 to prove Theorem 2.

Proof of Theorem 2: Since the postorder recursive doubling algorithm uses the theoretical lower bound of $\lceil \log_2(d+1) \rceil$ steps to deliver a message to d destinations, the number of steps is optimal. We now show that the algorithm is depth contention-free.

Consider a multicast implementation of the postorder recursive doubling algorithm in which all routing is performed using relaxed up-first paths. Let $(u, v, P_2(u, v), t)$ and $(x, y, P_2(x, y), \tau)$ denote an arbitrary pair of unicasts in this implementation and let $(u, v, P_1(u, v), t)$ and $(x, y, P_1(x, y), \tau)$ denote the corresponding unicasts in an implementation using strict up-first paths. Observe that by definition of relaxed up-first paths, any tree channels used by $P_2(u, v)$ are also in $P_1(u, v)$ and similarly for $P_2(x, y)$ and $P_1(x, y)$. Thus, if unicasts $(u, v, P_2(u, v), t)$ and $(x, y, P_2(x, y), \tau)$ contend for a common tree channel, then unicasts $(u, v, P_1(u, v), t)$ and $(x, y, P_1(x, y), \tau)$ contend for the same tree channel, contradicting Theorem 3.

The remaining case is that unicasts $(u, v, P_2(u, v), t)$ and $(x, y, P_2(x, y), \tau)$ contend for a cross channel. Thus, $P_2(u, v)$ and $P_2(x, y)$ share some cross channel $e = (v_i, v_j)$. By the definition of relaxed up-first paths, v_i occurs before v_j on the strict up-first paths $P_1(u, v)$ and $P_1(x, y)$. Let $e_i = (v_i, v_{i+1})$ denote the first tree edge on the strict up-first path from u to v. Edge e_i is common to both $P_1(u, v)$ and $P_1(x, y)$, implying that unicasts $(u, v, P_1(u, v), t)$ and $(x, y, P_1(x, y), \tau)$ contend for edge e_i, contradicting Theorem 3. □

4 Simulation Results

In this section we describe simulation results for single multicast and multiple multicast traffic. The simulations were conducted using the Harvey Mudd MARS simulator, an event-driven flit-level wormhole routing simulator. Since the postorder recursive doubling algorithm allows for partial adaptivity in the routing paths, a selection policy was employed that prefers the outgoing channel to a node whose ID is closest to the ID of the destination. This policy implicitly prefers cross channels over tree channels.

The following system parameters were used in these experiments. Each switch was assumed to have 8 ports. In order to simulate physical proximity of connected switches, switches were randomly selected from points on an integer lattice and connected only to adjacent lattice points. Thus, at most 4 ports per switch were used for connections to other switches. In order to maximize the probability of contention between unicasts from *different* multicasts in our multiple multicast experiments, each switch was connected to exactly one processor.

The following latency parameters were used in all experiments. The communication startup latency, t_{startup} was 10 microseconds, router setup latency for each message header, t_{router}, was 20 nanoseconds, and the channel latency, t_{channel}, was 10 nanoseconds. Each message comprised 128 flits. The measured latency for a multicast message was the total elapsed time from startup at the source until the last constituent unicast was consumed by the last destination node. Each data point in our experiments is within 2%

of the mean or better, using 95% confidence intervals.

In the first set of simulations, message latency was measured for a single multicast with a varying number of destinations. The simulations were conducted for networks comprising 64 and 256 nodes. Figure 3 summarizes the results of these simulations. In addition to the latencies incurred by the postorder doubling algorithm, the theoretical lower bound on the startup times, $\lceil \log_2(d+1) \rceil \cdot t_{\mathrm{startup}}$, is also plotted. The small difference between the two curves accounts for actual routing time. These results confirm that the latency incurred by the recursive doubling algorithm is directly related to the theoretical lower bound on the number of required startups, but the additional network latency is largely insensitive to the number of destinations. Additional simulation results can be

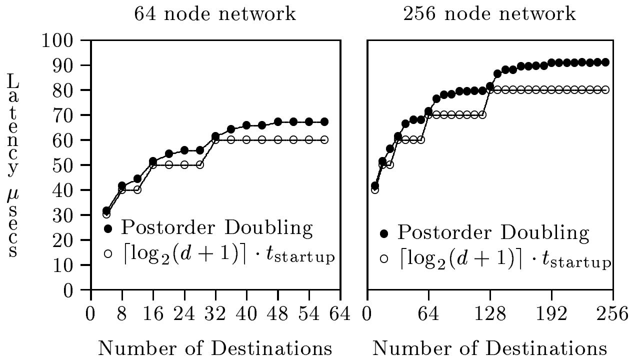

Figure 3: Latency versus number of destinations for a single multicast in 64 and 256 node networks.

found in [5].

5 Conclusion

In this paper we have presented a new depth contention-free unicast-based multicast algorithms for arbitrary topologies. The algorithm is optimal with respect to the number of startups, permits partially adaptive routing, and is provably deadlock-free. The technique used here to prove depth contention-freedom may have applications in developing other depth contention-free routing algorithms in both regular and irregular topologies which are currently being investigated. Simulation studies of the postorder doubling algorithm verify that single multicasts perform as predicted by the theoretical properties of the algorithm and multiple multicast latency increases approximately linearly with average arrival rate.

References

[1] B. Birchler, A-H. Esfahanian, and E. Torng. Sufficient conditions for optimal multicast communication. In *Proc. of Int. Conf. on Parallel Processing*, Aug. 1997.

[2] N. Boden et al. Myrinet: A gigabet-per-second local-area network. *IEEE Micro*, 15(1):29–36, February 1995.

[3] L. De Coster, N. Dewulf, and C-T. Ho. Efficient multi-packet 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] R. Kesavan, K. Bondalapati, and D. Panda. Multicast on irregular switch-based networks with wormhole routing. In *Third Int. Symposium on High Performance Computer Architecture*, pages 48–57, Feb. 1997.

[5] R. Libeskind-Hadas, D. Mazzoni, and R. Rajagopalan. Optimal contention-free unicast-based multicasting in switch-based networks of workstations. Technical Report HMC-CS-97-03, Harvey Mudd College, August 1997.

[6] 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.

[7] D. Robinson, P. McKinley, and B. Cheng. Optimal multicast communication in wormhole-routed torus networks. In *Proc. of Int. Conf. on Parallel Processing*, volume 1, pages 134–141, Aug. 1994.

[8] M. Schroeder et al. Autonet: A high-speed, self-configuring local area network using point-to-point links. Technical Report 59, DEC SRC, April 1990.

[9] H. Sullivan and T. R. Bashkow. A large scale, homogeneous, fully distributed parallel machine. In *Proc. of the 4th Ann. Symp. Comp. Architecture*, volume 5, pages 105–124, Mar. 1977.