Multicasting on Switch-based Irregular Networks using Multi-drop Path-based Multidestination Worms [1]

Ram Kesavan and Dhabaleswar K. Panda

Dept. of Computer and Information Science
The Ohio State University, Columbus, OH 43210-1277
Tel: (614)-292-5199, Fax: (614)-292-2911
E-mail: {kesavan,panda}@cis.ohio-state.edu

Contact Author: Dhabaleswar K. Panda

Abstract: This paper presents a novel concept of *multi-drop path-based* multidestination message passing on switch-based irregular networks. First, the multi-drop mechanism is defined with an associated header encoding scheme, and this mechanism is used to develop path-based multidestination worms. Next, a method is proposed to identify valid multidestination paths on arbitrary irregular networks with a typical deadlock-free routing. Then, the deadlock-free property of multi-drop path-based worms is emphasized. Using the above concepts, three multicast algorithms are proposed: multi-drop path-based greedy (MDP-G), multi-drop path-based less-greedy (MDP-LG), and multi-drop path-based binomial (MDP-B). The proposed algorithms are compared with each other and with the best unicast based algorithm, the CCO [5], for a range of system and technological parameters. The MDP-LG scheme is shown to be the best to implement multicast with reduced latency.

Keywords: Interconnection network, collective communication, multicast, broadcast, wormhole routing, cut-through routing, networks of workstations, and path-based routing.

[1] This research is supported in part by NSF Grant MIP-9309627 and NSF Career Award MIP-9502294.

Contents

1 Introduction

The cut-through/wormhole-routing switching technique is becoming the trend in building future parallel systems due to its inherent advantages like low-latency communication and reduced communication hardware overhead [9]. Intel Paragon, Cray T3D, Ncube, J-Machine, and Stanford DASH are systems using this switching technique. These systems have regular network topologies with various deadlock-free routing schemes. More recently, this switching technique is being applied to switch-based interconnects like Myrinet [1] and ServerNet [4] to build networks of workstations for cost-effective parallel computing. Such networks typically have irregular topology which provides flexibility to design scalable systems with incremental expansion capability. The irregularity also makes the routing on such systems quite complicated. Typically deadlock-free adaptive routing schemes [13, 14] are supported on such systems to achieve low-latency and high-bandwidth communication.

In order to to support the paradigms of distributed memory or distributed-shared memory, these systems need fast implementation of collective communication operations (*broadcast, multicast, global combine*, and *barrier synchronization*) [8, 11]. Traditionally, wormhole-routed systems support only *point-to-point* (unicast) message passing. This mechanism allows a message to have exactly one destination. This leads to collective communication operations being implemented as *multiple phases* of unicast message exchange [7, 5, 18].

In order to reduce the number of phases to implement collective communication operations a Hamiltonian Path-based scheme was proposed on MPP networks [6]. This scheme uses a modified router organization to support propagation of multidestination messages (a message with multiple destinations) and allows a message to be consumed at a destination node while being forwarded to the next node concurrently. Such an approach demonstrated that collective communication operations can be implemented on MPP networks with reduced latency. This concept was futher extended to allow multidestination worms to take row-column paths [2] and all paths conforming to the base routing using the *Base Routing Conformed Path* (BRCP) model [12].

Modern cut-through switches possess the capability of simultaneous replication of an incoming worm to multiple output ports [14]. Although this facility has been exploited for implementing broadcast by flooding the network, there has been little prior work on using this capability for arbitrary multicast. Recently, a multiport encoding scheme has been proposed to construct multidestination worms in multistage interconnection networks [15], and the associated architectural issues in supporting such worms have been discussed in [17]. This leads to a challenge of designing an efficient path-based multidestination message passing mechanism on irregular networks. This challenge includes design of an efficient header encoding scheme, a scheme for constructing such worms, determining valid paths in the system, and designing efficient multicast algorithms.

In this paper we take on the above challenges. First, the multi-drop mechanism with an associated header encoding is proposed. Next, the path-based multidestination message passing mechanism is developed on switch-based irregular networks. A method to identify valid multi-drop paths with respect to the Autonet routing is proposed, and proved to be deadlock-free. Then, we propose three new multicast algorithms using the proposed path-based multidestination message passing mechanism: a) the Multi-drop Path-based Greedy (MDP-G) algorithm, b) the Multi-drop Path-based Less-Greedy (MDP-LG) algorithm, and c) the Multi-drop Path-based Binomial (MDP-B) algorithm. These algorithms are evaluated over a range of system parameters, and compared with the best unicast-based multicast algorithm, the CCO algorithm [5]. The MDP-LG algorithm shows benefits of 50%, 40%, and 20% for multicast latency when compared to the CCO, MDP-G, MDP-B algorithms, respectively. These results clearly demonstrate that the multi-drop path-based multidestination mechanism promises significant potential to implement fast and efficient collective communication operations in future switch-based irregular systems.

The rest of the paper is organized as follows. Section 2 provides an overview of switch-based irregular networks, and deadlock-free routing on such networks. Section 3 describes the basic concepts of multidestination message passing. Section 4 presents the multi-drop path-based multidestination mechanism. A method of creating valid multi-drop worms on irregular networks is described in Section 5. Section 6 discusses three new multicast algorithms. Performance evaluation results of these algorithms are presented, and compared with those of the CCO algorithm in Section 7. Finally, concluding remarks are made in Section 8.

2 Switch-based Irregular Networks

In this section we provide a brief overview of switch-based irregular networks and focus on a typical deadlock-free routing on such networks.

2.1 Irregular Interconnection

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 bidirectional 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, some ports are connected to ports of other switches to provide connectivity between the processors, and some ports are left open for further 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 [1, 5, 14]. Figure 1(b) shows the interconnection graph for the irregular network in Fig. 1(a). The switches can implement different types of switching techniques: wormhole, virtual cut-through, or ATM. These switches can also implement different routing schemes [1, 4, 13, 14]. In this paper we focus on wormhole switches and assume the Autonet [14] routing scheme due to its simplicity and its commercial implementation. The scheme is briefly described below.

2.2 The Autonet Routing

In the Autonet routing scheme [14], a breadth-first spanning tree (BFS) on G is first computed using a distributed algorithm. 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 1(c) shows the BFS spanning tree embedded on graph G shown in Fig. 1(b). 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.

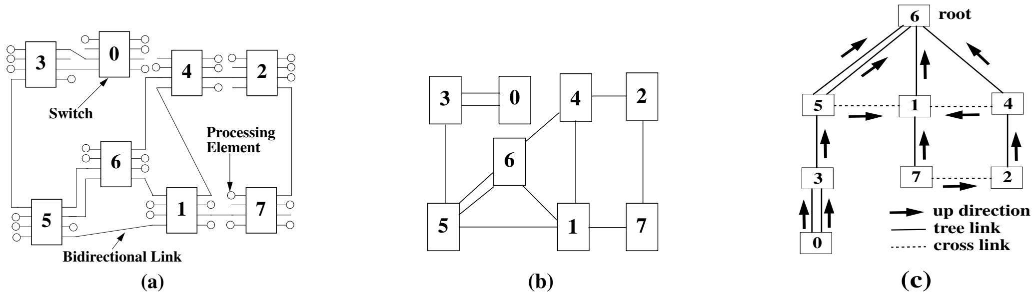

Figure 1: (a) An example system with switch-based interconnect and irregular topology, (b) the corresponding interconnection graph G, and (c) the BFS spanning tree rooted at node 6 of G.

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. Therefore, this routing is also called *up*/down** 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. During configuration the current implementation of Autonet switches constructs tables to support minimal adaptive routing [14]. In this paper, we assume that the forwarding tables allow only legal routes with the minimum hop count.

3 Multidestination Message Passing

In this section, we describe the basic concepts of multidestination message passing and path-based multidestination worms. Next, we discuss tree-based multidestination worms on switch-based irregular networks, and show the motivation for path-based multidestination worms on such networks.

3.1 Multidestination Message Passing and Path-based Worms

The concept of wormhole message passing with multiple destinations was first introduced by Lin and Ni for [6] Hamiltonian meshes. Then, the multidestination message passing was introduced [12] for k-ary n-cube networks with different base routing schemes. These schemes were proposed for regular direct networks typical of MPPs. In such networks each node consists of a router and a processor. The processor is connected to the router through a set of injection and consumption channels. The routers are connected in some given topology. A path-based multidestination worm in such networks carries the *ids* of multiple destinations in its header. After being injected into the network by the source, the worm is routed based on the first destination in the header. When the worm reaches the router of an intermediate destination, the address of the destination is removed from the header, and the next destination is used to route the worm onwards. As the data flits pass through routers corresponding to intermediate destinations, they are forwarded to the next router and absorbed to the processor. The final destination absorbs the complete worm.

3.2 Tree-based Multidestination Worms

The above path-based multidestination mechanism (with a header of destination ids) cannot be directly applied to switch-based networks because there is no one-to-one correspondence between switches/routers and processors on such networks. Thus a tree-based mechanism [3, 10, 15, 17] has been proposed which allows a multidestination worm to replicate at switches with multiple replicas of the worm following different paths in the network. In this approach, the switches possess the capability to replicate flits of an arriving multidestination worm to multiple output ports. The issues of synchronous and asynchronous replication of flits in a deadlock-free manner within the switch have been discussed in detail in [15, 17]. Figure 2 shows an example of a tree-based multidestination worm for MINs using the *multiport encoding scheme* [15]. It must be noted that such a worm branches out in a symmetric fashion. For example, switches s_1 and s_2 replicate to output ports 0, 2, and 3, and switches s_3, s_4, s_5, and s_6 replicate to output ports 0 and 2. The symmetry is required because all replicas of a multiport encoded multidestination worm created at a switch must possess the same header information.

It can be easily observed that it is difficult to create such multiport encoded worms for irregular networks. In [17] the authors show how precomputed *reachability information* at switches can be used to generate asymmetric multidestination worms with *bit-string encoded* headers. The authors have also extended this bit-string encoded mechanism to irregular switch-based networks [16]. However, computation of reachability information at each switch for an arbitrary switch-based irregular network may be quite a non-trivial task. Therefore, the question arises whether path-based multidestination mechanism can be adapted to switch-based irregular networks using the replication capability at the switches. The challenges to doing this are in deriving solutions to the following problems: a) the concept of path-based worms on a switch-based network, b) a suitable header encoding scheme, c) defining valid paths on a given base routing scheme, d) prevention of deadlock due to coexistence of such worms with unicast worms, and e) developing multicast algorithms. In the next sections we look at these problems, and propose solutions.

4 Multi-drop Path-based Multidestination Worms

In this section, we describe the concept of multi-drop path-based multidestination worms and a new header encoding scheme for such worms.

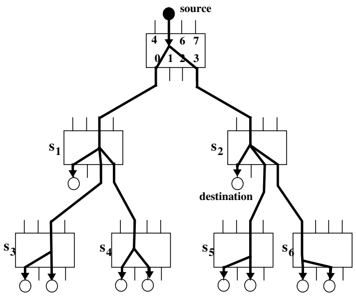

Figure 2: Example of tree-based multidestination worm for MINs using the multiport encoding mechanism [15].

4.1 The Multi-drop Mechanism

In this mechanism, the capability at switches to replicate incoming worms and to forward the copies to several output ports simultaneously is used, but the replication is restricted. The multi-drop mechanism is defined as follows.

Definition 1 *A multidestination worm is defined as a multi-drop worm if a) it can cover multiple destinations on a traversed switch and b) for every switch in which it replicates to multiple outgoing ports, no more than one of these ports is connected to another switch and the remaining are connected to processors.*

Such a mechanism allows easy construction of paths (of switches) in an irregular network. It simplifies handling of deadlock problems because a worm does not get replicated into several worms, branched over multiple paths. It also allows simplified header encoding by using a variation of the all-destination encoding [6, 12] where 'destination' ids are replaced by 'switch' ids and ids of 'destinations connected to a switch'. Figure 3(a) shows an example of a multi-drop path-based multidestination worm. If we abstract a switch-based irregular network to the interconnection graph G, then multi-drop worms can be abstracted to strict path-based multidestination worms on G. The path of a multi-drop worm is defined by the set of inter-switch links taken by the worm. From this point on, we refer to a path as a sequence of inter-switch links. Although the number of destinations covered in a single step by such a multi-drop multidestination worm can be less than that covered by a worm using the tree-based mechanism, this scheme is much simpler. It also requires minimal modification to the switch and results in low complexity for creation of multidestination worms.

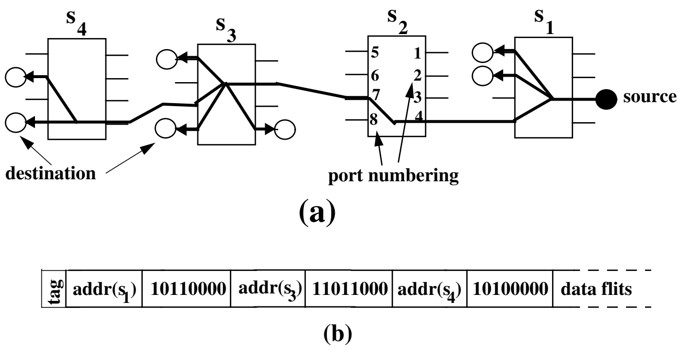

Figure 3: The Multi-drop mechanism: (a) example of a multidestination worm and (b) the header encoding.

4.2 The Multi-drop Header Encoding Scheme

This new scheme is a hybrid of the *all-destination* [6, 12] and *multiport* [15] encoding schemes. It makes use of the fact that switches too have unique addresses in a typical switch-based irregular network like Autonet [14]. The current implementation of the Autonet switch can route messages addressed to a particular switch. The multi-drop worm header encoding for each set of destinations on a switch contains the switch address followed by the multiport encoding of destinations for that switch. A tag bit can distinguish whether it is a unicast or multidestination worm. Figure 3(b) shows the multi-drop scheme encoded header of the worm shown in Fig. 3(a). Since switch s_1 is the first switch with destinations on it, the header encoding begins with the address of s_1. The following 8 bits encode the information about which output ports the worm replicates to. The bits correspond to output ports 1 to 8 from right to left. In the first switch, the worm is intended to be replicated to two destinations and one outgoing link to switch s_2. Similarily, information about switches s_3 and s_4 is encoded in the header.

In this scheme, the source does not completely specify the route taken by the worm - intermediate switches with no destinations are not included. Since the routing tables at the switches are used to route a worm, any adaptivity provided by the underlying routing scheme is fully utilized between any two consecutive participating switches. The length of the header of a worm only increases linearly with the number of participating switches in the worm, and not the number of hops traversed by the worm or the number of destinations. Also, this new scheme requires only a minor change in the routing logic of a switch which already provides the replication facility, like the Autonet switch.

Now, the challenge is in defining valid paths on a given base routing scheme, and prevention of deadlock due to coexistence of such worms with unicast worms. In the next section we solve this problem in the context of Autonet routing.

5 Creation of Multi-drop Worms on Irregular Networks

A multi-drop path-based multidestination worm needs to get routed in a *piece-wise* manner from the source switch to the final switch. This raises the following fundamental problems at any intermediate switch: a) it should be feasible for the multidestination worm to find a path leading to the next switch in its list and b) the piece-wise routing of a multidestination worm should not violate the routing scheme to prevent deadlock. In this section, we first propose a method to identify valid multi-drop multidestination paths which conform to the base Autonet routing on an arbitrary irregular graph. Then, we discuss the requirements for deadlock-freedom.

5.1 Valid Multidestination Paths with the Autonet Routing

Let us define the list of participating switches, $L_{sw}(w)$, of a multi-drop multidestination worm w as follows. $L_{sw}(w)$ consists of switches directly connected to the destinations of w, in the order in which the corresponding destinations will be covered. Let us denote the jth processor on the ith switch as $p_{i,j}$. If w starts from source processor $p_{s,1}$ and covers the processors $p_{1,1}, p_{1,2}, \ldots, p_{2,1}, p_{2,2}, \ldots, p_{n,m}$, in that order, then $L_{sw}(w)$ is given by the switches $< s_s, s_1, s_2, \ldots, s_n >$. Therefore, generating a valid multi-drop multidestination path for a worm w on an arbitrary switch-based irregular network N is equivalent to generating a valid path which covers the switches in $L_{sw}(w)$, in that order, on the interconnection graph G of that network.

Definition 2 *Let P_{sd} and P^m_{sd} be defined as the sets of all paths allowed between a source switch s to a destination switch d under the up*/down* routing and up*/down* minimal routing, respectively.*

Definition 3 *A partial ordered chain (POC) is 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.*

In our recent work [5], we defined the concept of a POC with the specific purpose of constructing an ordering among switches in an irregular network to reduce link contention between two messages of the same multicast using *unicast* message passing. In this paper, we use this concept to construct valid multi-drop multidestination paths.

Definition 4 *The path set of a multi-drop multidestination worm w, denoted as $PS^m(w)$ is defined as the set of all paths that can be taken by w when routed in a piece-wise manner using the routing tables which support up*/down* minimal routing.*

Lemma 1 *The path set of a multi-drop multidestination worm w, $PS^m(w)$, is given by $P^m_{s_s s_1} \times P^m_{s_1 s_2} \times \ldots \times P^m_{s_{n-1} s_n}$ where $L_{sw}(w) =< s_s, s_1, s_2, \ldots, s_n >$.*

Proof: The worm w starts from the switch s_s and covers $s_1, s_2, \ldots, s_n$, in that order. Since w is routed in a piece-wise manner using up*/down* minimal routing, it first takes one of the paths from $P^m_{s_s s_1}$. Then, it takes one of the paths from $P^m_{s_1 s_2}$, and so on. Thus, $PS^m(w) = P^m_{s_s s_1} \times P^m_{s_1 s_2} \times \ldots \times P^m_{s_{n-1} s_n}$. ■

Theorem 1 *$PS^m(w) \subseteq P_{s_s s_n}$ if $L_{sw}(w) =< s_s, s_1, s_2, \ldots, s_n >$ and if the switches $s_1, s_2, \ldots, s_n$ lie on some POC, P, in that particular order.*

Proof: It can be easily observed that all paths in $P^m_{s_s s_1}$ are up*/down* paths. If it can be shown that $\forall i, (1 \leq i < n)$, all paths in $P^m_{s_i s_{i+1}}$ are down* paths, then using the result from Lemma 1 it can be seen that all paths in $PS^m(w)$ are up*/down* paths. Let P be given by the ordered list of switches $< sw_1, sw_2, \ldots, sw_m >$. Then $\forall i, (1 < i < n)$, $s_i \in P$. Let us use the symbol $<_{poc}$ to denote the order in the above list P. Therefore, $\forall i, (1 \leq i < m)$, $sw_i <_{poc} sw_{i+1}$, and $\forall i, (1 \leq i < n)$, $s_i <_{poc} s_{i+1}$. Let $E =< e_1, e_2, \ldots, e_{n-1} >$ denote the list of "down" links such that switch sw_i is connected to the switch sw_{i+1} by the "down" link e_i, where sw_i and $sw_{i+1} \in P$. Let us consider two arbitrary switches s_a and s_{a+1} such that $1 \leq a < n$, $s_a = sw_i$, and $s_{a+1} = sw_j$. Therefore, $s_i <_{poc} s_j$. It can be easily observed that the path described by list of links $p_{i,j} =< e_i, e_{i+1}, \ldots, e_{j-1} >$ is one of the valid minimal paths between s_i and s_j. Also, each link in this path is a down link, and the path is $j - i$ links long. Any path from s_i to s_j that contains an up link is definitely longer than $p_{i,j}$ because an up link takes the worm higher up the BFS routing tree, whereas s_j is exactly $j - i$ levels lower than s_i in the BFS tree. Since a routed worm from s_a to s_{a+1} takes only down links, all paths in $P^m_{s_a s_{a+1}}$ are down* paths. Therefore, piece-wise routing from s_1 to s_n results in a down* path only. Therefore, all paths in $PS^m(w)$ are up*/down* paths. ■

Theorem 2 *A multi-drop path-based multidestination worm w, where $L_{sw}(w) =< s_s, s_1, s_2, \ldots, s_n >$ and the switches $s_1, s_2, \ldots, s_n$ lie on some POC, P, in that particular order, always follows a valid up*/down* path and introduces no deadlocks.*

Proof: This can be easily derived from Theorem 1, and the fact that non-minimal up*/down* routing is deadlock-free, and non-minimal and minimal up*/down* paths can coexist without deadlock. ■

It must be noted that the concept of valid multi-drop multidestination worms using POCs on Autonet routing can also be extended to the case when the routing tables support non-minimal paths. In this case, multi-drop multidestination worms are not routed using the routing tables at intermediate switches, instead the source specifies the exact route (all switches in a path) in the header encoding of the worm. Such a header encoding for the sample multi-drop worm in Fig. 3(a) is shown in Fig 4(a) (it includes the address of s_2 with a single port replication). It can be easily observed that the switch addresses in this encoding is redundant information, and the header encoding can be further simplified by dropping the switch addresses completely, and retaining only the multiport encoding, as shown in Fig. 4(b).

addr(s_1)	10110000	addr(s_2)	01000000	addr(s_3)	11011000	addr(s_4)	10100000	data flits

(a)

10110000	01000000	11011000	10100000	data flits

(b)

Figure 4: Header encoding of the multi-drop of Fig. 3(a) for routing tables supporting nonminimal routing: (a) each switch in the route is specified with its multiport encoding, and (b) an enhancement by eliminating the switch addresses.

5.2 Deadlock-freedom

It is shown in [15] that asynchronous replication with buffers at least as large as the largest multidestination worm ensures deadlock freedom on multistage interconnection networks for tree-based multidestination worms. The Autonet switch provides similar guarantee with input buffer size of 4096 bytes [14]. (This feature is provided to prevent deadlock that arises in the case of broadcast worms which use the replication mechanism at the switches to flood the network.) Since multi-drop worms form a subset of all tree-based worms, addition of multi-drop worms cannot cause deadlock in an Autonet switch-based network provided the message length is less than the given buffer size.

Interestingly, the buffer requirement for deadlock-freedom with multi-drop worms can be actually lowered depending on the scheduling policy used at the switch. There are two basic kinds of scheduling policies at a switch: *centralized* or *distributed*. When a worm arrives at the switch, the header flits are used to index into the routing table, and the appropriate output port(s) are read from the table to form the request. These requests can be handled at each input port independently (distributed) or all the requests can be handled centrally (centralized). The buffer requirement for deadlock-freedom should be larger than the maximum worm length in the case of the distributed scheduling policy, but much lesser in the case of the centralized scheduling policy.

The Autonet switch uses a central scheduling policy to schedule output ports among the various incoming worms, both unicast and broadcast. The central scheduling engine maintains a queue of requests for ports. If a broadcast request is for multiple output ports, then the engine only grants the request after it accumulates all the requested output ports in the free state. Therefore, the replication of flits to output ports only begins after all the required output ports have been captured by the worm. If the same centralized scheduling policy is used with the multi-drop mechanism the buffer size requirement for deadlock-freedom can be relaxed as shown below.

Theorem 3 *Multi-drop path based multidestination worms using the valid paths as defined in Section 5.1 with the Autonet central scheduling is deadlock-free with even single flit buffers at each input port of a switch.*

Proof: A multidestination worm cannot capture some ports of a switch while waiting on some other ports. This is because the centralized scheduler only grants all requested ports to a worm after they have been captured. Once a multi-drop worm reaches its first destination switch it can only proceed on 'down' links. This is because after reaching any switch which is adjacent to a destination the worm is forced to take 'down' links down a POC, as described in Theorem 1. Therefore no multi-drop worm can wait on an 'up' link after it has reached even one destination. This means there can exist no cycle of multi-drop worms which have reached at least one destination each, since there exists no cycle of 'up' links. This also holds for a combination of unicast and multi-drop worms since there can exist no worm holding on to a 'down' link and waiting on an 'up' link. Therefore, the multi-drop mechanism can be implemented without deadlock with single flit size buffers at input ports on any switch which uses the central scheduling policy. ∎

6 Multicast Algorithms

In this section we propose three multicast algorithms using the multi-drop path-based mechanism. All these algorithms use the POC concept to construct as many longest POCs as possible from the participating switches, and group these POCs in some hierarchy to implement multicast. Let us first look at how the POCs are constructed.

6.1 Construction of POCs

In [5], we described a method to construct as many longest POCs as possible from the participating switches of a multicast. Figure 12 in the Appendix formally describes the method.

We explain the method using an example multicast. The interested reader can refer to [5] for details.

Figure 5(a) shows the BFS graph of a sample irregular network. Let us assume that there are two processors connected to each switch of the irregular graph, and let the processors on the switch numbered i be labelled as $p_{i,1}$ and $p_{i,2}$. Let us consider a multicast with the source as $p_{10,1}$ and two destinations on each switch except switches 10 and 6. In the first step, a depth-first search (DFS) is applied on the BFS spanning tree, starting with the root node r and considering only the down links specified in Definition 3. This is to facilitate the construction of the longest POCs. Figure 5(b) shows the DFS tree, T, on the sample irregular network. A participating switch is one with at least one participating processor connected to it. The resultant DAG, T, from the DFS can be reduced to a DAG, T', which contains only the participating switches. Fig. 5(c) shows the reduced DAG T'. Each switch is given a weight equal to the number of participating processors connected to it and to all its descendent switches. Subsequently, chains of switches are stripped off from T' according to their weights. In other words, the heaviest chain gets stripped first from T', and the lightest the last. Figure 5(d) shows the obtained set of chains, C. In all these figures, the participating switches are highlighted.

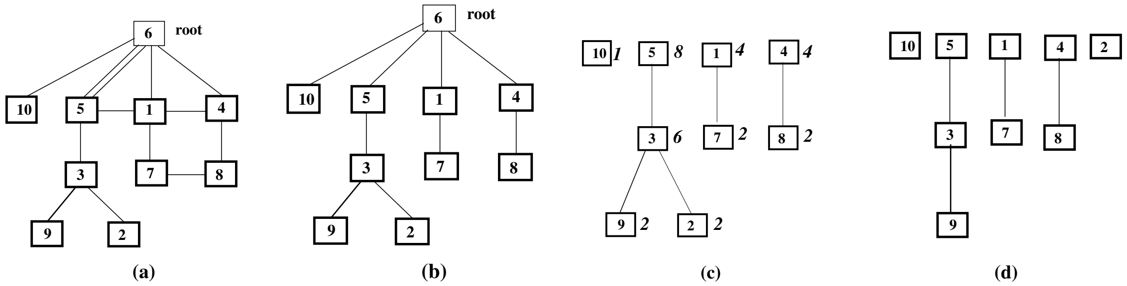

Figure 5: Illustrating steps for construction of POCs: a) sample BFS graph of an irregular network, b) DFS tree T, c) DAG T' and the associated switch weights, and (d) the set C of chains.

6.2 The Multi-drop Path-based Greedy (MDP-G) Algorithm

This algorithm uses a greedy approach on the set of POCs generated by the above steps, to implement multicast. First, the source node covers the destination processors on the switches of the heaviest chain with a single multi-drop worm. Then, in each successive step, the destinations that have been covered in previous steps cover destinations on chains that have not been covered yet (in C) using multi-drop worms. Figure 6(a) shows these steps for the example multicast: $p_{10,1}$ covers all the destinations on switches 5, 3, and 9 with a single multi-drop worm in the first step. In the second step, $p_{10,1}$ covers all destinations on switches 1 and 7 using a multidestination worm, $p_{5,1}$ covers all destinations on switches 4 and 8, and $p_{5,2}$ covers all destinations on switch

2. Therefore, the multicast completes in two steps.

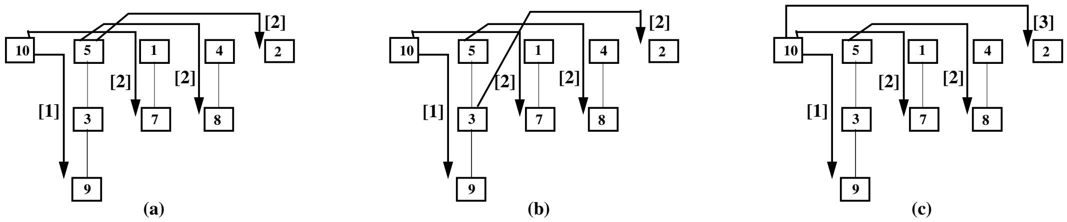

Figure 6: Illustrating the multicast algorithms for the sample multicast of Fig. 5: a) the MDP-G algorithm, b) the MDP-LG algorithm, and c) the MDP-B algorithm. The numbers in brackets denote step numbers.

Since at each step maximum possible destinations are being covered using multi-drop worms constructed with longest possible POCs, this algorithm takes minimum number of steps to execute. However, it has the drawback that the number of multi-drop worms starting from a single switch might be large leading to link contention.

6.3 The Multi-drop Path-based Less Greedy (MDP-LG) Algorithm

We propose the MDP-LG algorithm to address the above drawback. This algorithm also uses the output from the POC construction method, like the MDP-G algorithm. Following the first step, only one destination from each switch that has been covered chooses chains that have not been covered yet (in C), and covers them using multidestination worms. Since all destinations covered in the particular step do not send out worms this scheme is given the name Less Greedy. As shown in Fig. 6(b), $p_{10,1}$ covers all the destination on switches 5, 3, and 9 with a multi-drop worm in the first step. In the second step, $p_{10,1}$ covers all destinations on switches 1 and 7, $p_{5,1}$ covers all destinations on switches 4 and 8, and $p_{3,1}$ covers all destinations on switch 2. This algorithm may take more steps to complete, but it reduces link contention by making sure that only one multi-drop worm starts from a switch during one step.

6.4 The Multi-drop Path-based Binomial (MDP-B) Algorithm

We propose the MDP-B algorithm to further reduce link contention in the MDP-LG algorithm. Again, the MDP-B algorithm starts the same way as the MDP-G algorithm with the construction of the POCs. Following the first step, only the source processor and the leader processor of each chain are involved in sending worms. The leader processor is chosen as a destination processor on the first switch of the chain. The source covers all destinations on one chain in the first step using a multi-drop worm. In the second step, the source and the leader of the covered chain cover all destinations on two other chains using multi-drop worms, and so on. As shown in Fig. 6(c), $p_{10,1}$ covers all the destination on switches 5, 3, and 9 with a multi-drop worm in the first step. In the second step, $p_{10,1}$ covers all destinations on switches 1 and 7 with a single worm, and $p_{5,1}$ covers all destinations on switches 4 and 8 using a single worm. In the third step, $p_{10,1}$ covers all destinations on switch 2. Therefore, this algorithm requires $\lceil log_2 C \rceil$ steps. But, the link contention as compared to the MDP-G and MDP-LG algorithms is reduced, since there are no two destinations on the same chain trying to send out worms at the same time.

7 Simulation Experiments and Results

We performed simulation experiments to evaluate and compare performance of the MDP-G, MDP-LG, MDP-B, and CCO [5] algorithms for single multicast. A flit-level wormhole-routed simulator WORMULSim was used to model the irregular network. For all simulation experiments, we assumed system and technological parameters representing the current trend in technology. The following default parameters were used: t_s (communication start-up time) = 10 microseconds, t_{phy} (link propagation time) = 10 nanoseconds, t_{route} (routing delay at switch) = 40 ns, t_{sw} (switching time across the router crossbar for a flit) = 10 ns, t_{sched} (time for scheduler to process a request) = 20 ns, t_{inj} (time to inject a flit into network) = 10 ns, and t_{cons} (time to consume a flit from network) = 10 ns. For all experiments (unless otherwise stated) we assumed the following default system configuration: a 256 processor system interconnected by 64 eight-port switches and network having 75% connectivity, as defined in [5]. The default message length was assumed to be 128 flits.

We performed several experiments to study the impact of system size, message length, communication start-up time, switch size, and degree of connectivity on single multicast latency for varying sizes of destination sets. 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.

7.1 Effect of system size

We simulated the four multicast algorithms on four different system configurations with 64, 128, 256 and 512 processors, respectively. Figure 7 shows these results. It can be observed that the MDP-LG algorithm performs the best for almost all system sizes and destinations. As system size increases, the MDP-LG algorithm shows steady benefits compared to the CCO algorithm. For example, on a 512 processor system with 256 destinations, the reduction in multicast latency achieved by the MPD-LG algorithm is around 50%, 40%, and 20% compared to the CCO, MPD-G, and MDP-B algorithms, respectively.

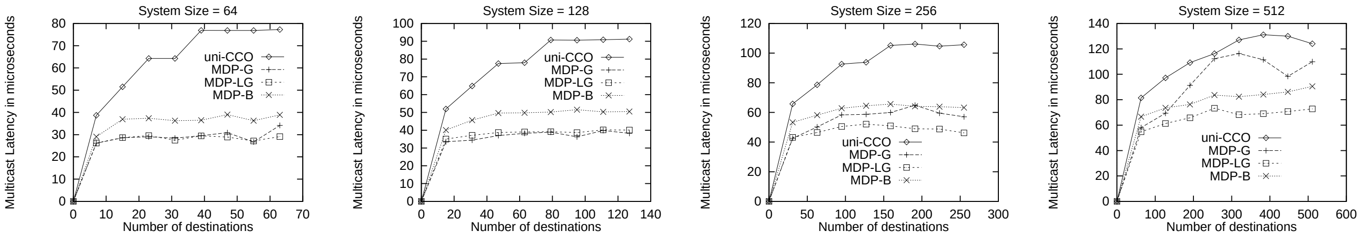

Figure 7: Multicast latency vs. number of destinations for four different system configurations: 64, 128, 256, and 512 processors.

As expected the multi-drop algorithms outperform the CCO algorithm. Since the number of destinations covered in one step is increased by using multidestination worms, the total number of steps for the completion of the multicast algorithm is reduced. As discussed in Section 6, the MDP-G suffers from link contention at higher destination set sizes since multiple destinations on the same switch send worms simultaneously, and this leads to severe link contention. The MDP-B algorithm incurs less link contention, but undergoes more number of steps, and hence it is outperformed by the MDP-LG which strikes a balance between number of steps and link contention.

7.2 Effect of message length

We studied the impact of message length on the four algorithms. Four different message lengths - 128, 256, 512, and 1024 flits, on a 256 processor system with default parameters were considered. Figure 8 shows the respective results. It can be easily observed that as message length increases the link contention effect dominates the results, and the MDP-B algorithm outperforms the remaining algorithms although it requires more steps.

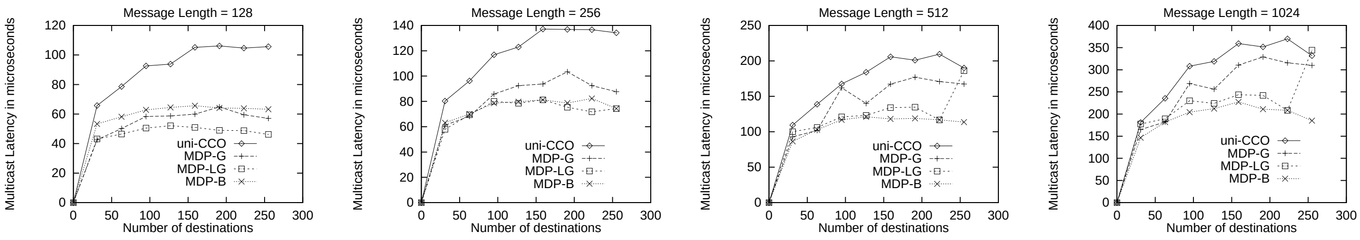

Figure 8: Multicast latency vs. number of destinations for four different message lengths: 128, 256, 512, and 1024 flits.

7.3 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 three different communication start-up times: 5.0, 10.0, and 20.0 microseconds. Figure 9 shows the respective multicast latencies. It can be observed that with higher communication start-up time, the MDP-G and MDP-LG algorithms outperform the other two algorithms. This is expected, because for a given message length, higher start-up times reduce the effect of link contention, and the MDP-G and MDP-LG algorithms take the minimum steps.

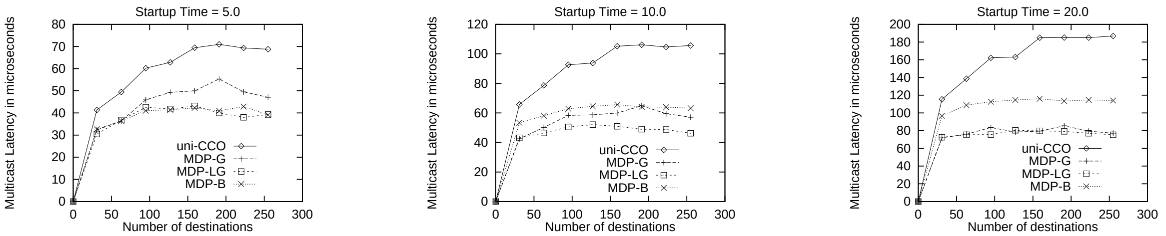

Figure 9: Multicast latency vs. number of destinations for three different communication start-up times: 5.0, 10.0, and 20.0 microseconds.

7.4 Effect of switch size

We studied the effect of switch size on the overall performance of the four algorithms. The default 256 processor system was considered with three different switch sizes: 8, 16, and 32 ports. Figure 10 shows the performance results. It can be observed that with increasing switch size the MDP-G 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 reduces the effect of link contention on the overall multicast latency. Therefore, the algorithm which completes in the minimum number of steps, the MDP-G algorithm, outperforms the other algorithms.

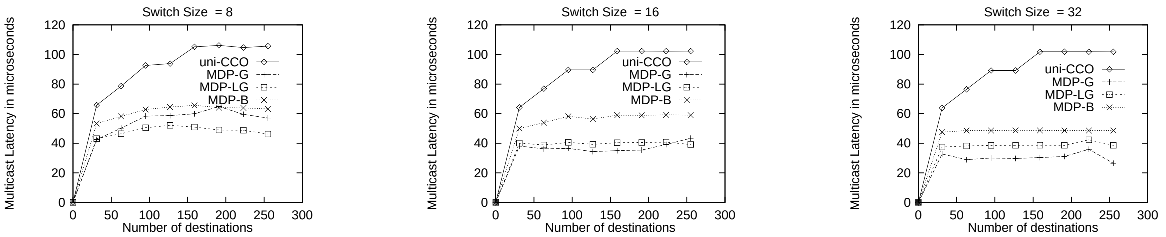

Figure 10: Multicast latency vs. number of destinations for three different switch sizes: 8, 16, and 32 ports.

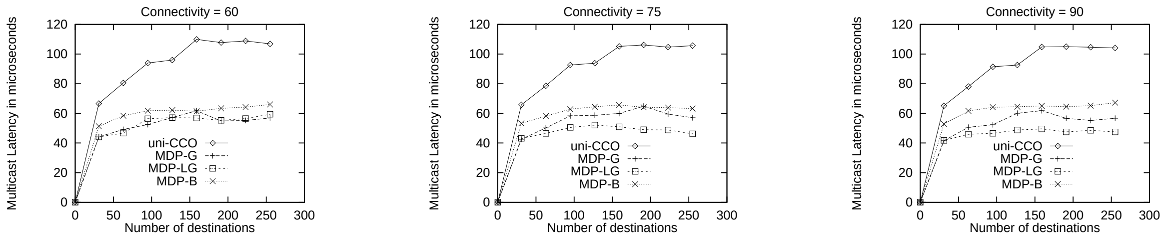

Figure 11: Multicast latency vs. number of destinations for different degrees of network connectivity: 60%, 75%, and 90%.

7.5 Effect of degree of network connectivity

Finally, we studied the impact of degree of network connectivity on multicast latency. The default system size of 256 processors was considered with three different degrees of network connectivity: 60%, 75%, and 90%. Figure 11 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 between messages. Under such circumstances, the MDP-B and MDP-LG algorithms deliver the best performance. As the degree of connectivity increases, the contention effect reduces, but does not get completely eliminated. Thus, with higher connectivity, the MPD-G and MDP-LG algorithms outperform the rest.

8 Conclusions

In this paper we have proposed a new multi-drop path-based multidestination message passing mechanism on switch-based irregular networks. A method to identify valid multi-drop paths with respect to a typical deadlock-free routing on arbitrary irregular networks has been proposed. Using this new mechanism three multicast algorithms, MDP-G, MDP-LG, and MDP-B, have been developed. These algorithms have been simulated and compared with each other and the best unicast-based multicast algorithm, the CCO algorithm. Compared to the CCO algorithm, the MDP-LG algorithm shows an improvement in multicast latency up to a factor of 60%. These results demonstrate significant potential for the multi-drop multidestination mechanism to be applied to current and future switch-based irregular networks.

Since our simulator currently models only pure wormhole routing (single flit buffer size at input ports of each switch), we believe that the observed benefits of the multi-drop multidestination mechanism were diminished due to the effect of link contention. We expect this effect to reduce with any realistic buffer size at input ports of switches, and increased benefits of the multi-drop mechanism. We are working on extending the simulator to model larger buffer sizes at the switches and results will be reported during the workshop. We are also looking at applying this new mechanism at other collective communication operations like barrier synchronization, gather, multiple multicast, etc.

Additional Information: A number of related papers can be obtained from `http://www.cis.ohio-state.edu/~panda/pac.html`. Alternatively, the papers can be obtained by anonymous ftp from: `ftp.cis.ohio-state.edu, cd pub/communication`.

Acknowledgements: The authors would 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.

References

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

[2] 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.

[3] C. M. Chiang and L. M. Ni. Deadlock-Free Multi-Head Wormhole Routing. In *Proceedings of the First High Performance Computing-Asia*, 1995.

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

[5] 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, 1997.

[6] 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.

[7] 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.

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

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

[10] Lionel Ni. Should Scalable Parallel Computers Support Efficient Hardware Multicasting? In *ICPP Workshop on Challenges for Parallel Processing*, pages 2–7, 1995.

[11] 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.

[12] D. K. Panda, S. Singal, and P. Prabhakaran. Multidestination Message Passing Mechanism Conforming to Base Wormhole Routing Scheme. In *Proceedings of the Parallel Computer Routing and Communication Workshop*, pages 131–145, 1994.

[13] 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.

[14] 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.

[15] R. Sivaram, D. K. Panda, and C. B. Stunkel. Efficient Broadcast and Multicast on Multistage Interconnection Networks using Multiport Encoding. In *Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing*, pages 36–45, Oct 1996. Available from http://www.cis.ohio-state.edu/~panda/paper.html.

[16] R. Sivaram, D. K. Panda, and C. B. Stunkel. Multicasting in Irregular Networks with Cut-Through Switches using Tree-based Multidestination Worms. Technical report, The Ohio State Univeristy, March 1997.

[17] 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 ACM Annual International Symposium on Computer Architecture (ISCA-24)*, June 1997. Available from http://www.cis.ohio-state.edu/~panda/paper.html.

[18] H. Xu, P. K. McKinley, and L. Ni. Efficient Implementation of Barrier Synchronization in Wormhole-routed Hypercube Multicomputers. *Journal of Parallel and Distributed Computing*, 16:172–184, 1992.

Appendix

Algorithm for POC Construction

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: Construct longest possible POCs from participating switches in G.

Procedure:

1. Apply the 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. $C \leftarrow null$; $i \leftarrow 0$
   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) $C \leftarrow concat(C, l_i)$; $i \leftarrow i + 1$
   end while

Figure 12: Algorithm for construction of POCs.