An Efficient Implementation of Tree-Based Multicast Routing for Distributed Shared-Memory Multiprocessors ⋆

M.P. Malumbres and José Duato

Departamento de Informtica de Sistemas y Computadores, Universidad Politécnica de Valencia, P.O.B. 22012. 46071 - Valencia, SPAIN
{mperez,jduato}@gap.upv.es

Abstract

This paper presents an efficient routing and flow control mechanism to implement multidestination message passing in wormhole networks. The mechanism is a variation of tree-based multicast with pruning to recover from deadlocks and it is well suited for distributed shared-memory multiprocessors (DSMs) with hardware cache coherence. It does not require any preprocessing of multicast messages reducing notably the software overhead required to send a multicast message. Also, it allows messages to use any deadlock-free routing function. The new scheme has been evaluated by simulation using synthetic loads. It achieves multicast latency reductions of 30% on average. Also it was compared with other multicast mechanisms proving its benefits. Finally, it can be easily implemented in hardware with minimal changes to existing unicast wormhole routers.

Key words: Interconnection networks, wormhole switching, tree-based multicast, adaptive routing, distributed shared-memory multiprocessors

1 Introduction

The performance of scalable multiprocessors is often determined by how effectively they support processor communication. In many cases, processor communication is slowed down by insufficient throughput and high latency in the interconnection network. It is important, therefore, to design cost-effective techniques that enhance network throughput and reduce message latency.

⋆ This work was supported by Spanish CICYT under Grant TIC97–0897–C04–01

Interprocessor communications can be classified into three types depending on the number of message destinations, namely one-to-one (unicast), one-to-many (multicast), and one-to-all (broadcast). Of these schemes, unicast and broadcast can be considered a special case of multicast. Multicast communications routinely appear in parallel programs. Typical examples include explicit distribution of data to several nodes or invalidation and update messages in distributed shared-memory multiprocessors [14] (DSMs). Similarly, the inverse of multicast, namely many-to-one messages, is also common. Examples include barrier synchronization and global reductions. It appears, therefore, that optimizing the multicast operation would improve the performance of scalable multiprocessors.

Multicast is supported in hardware by at least one commercial machine. Indeed, the NCube-2 multicomputer supports broadcast and, in addition, multicast messages whose destinations belong to a subcube of the hypercube [20]. This multicast scheme uses a tree-based multicast mechanism to reach nodes within a given subcube. This mechanism, however, is not deadlock-free. Currently, there is not any satisfactory commercial solution to hardware multicast.

Efficient support for multicast has been the subject of much previous academic research. The earliest studies [12,15] proposed optimal tree-based algorithms for multicast routing based on graph-modeling theory. However, aspects like topology, router design and deadlock problems were not considered. In [1], three deadlock-free multicast protocols were presented. These multicast protocols are specially designed for virtual cut-through networks [10] but no routing algorithm was proposed.

Later, deadlock-freedom was studied for multicast communications in multicomputer networks using wormhole switching [16,21]. The approach was to define a Hamiltonian path to route multidestination worms while avoiding deadlock. Multicast messages are propagated following one path that visits all destinations without branching at intermediate routers. This type of multicast is called path-based multicast. Routing algorithms like Dual-Path and Multi-Path were proposed for 2-D meshes.

New partially and fully adaptive path-based multicast wormhole routing algorithms called PM, FM, and LD were defined for 2-D meshes [17]. However, the design of deadlock-free adaptive multicast routing algorithms is complex. For this reason, a new theory and methodology for designing deadlock-free adaptive multicast algorithms was proposed in [5,7]. The theory makes the design of these algorithms easy and efficient. This theory is an extension of a previously proposed theory for deadlock-free unicast routing [6]. Alternative approaches to solve the same problem were proposed in [18,19].

Other works developed broadcast communication in wormhole networks using

SDP (Spanning set of Dimensional-disjoint Paths) [22] . This scheme uses several phases to transport one message to all the destinations. Two general solutions proposed are the 1-port and the n-port. They are deadlock-free and achieve better performance than path-based multicast.

After these studies, the BRCP (Base Routing Conformed Path) model was developed [23]. This is a new path-based message passing mechanism that transports multicast and broadcast messages and is deadlock-free. This mechanism routes multicast messages using the same routing algorithm as for unicast messages, so routing algorithms like e-cube, planar-adaptive, turn-model, or fully adaptive can be used. Multicast and broadcast messages are carried toward their destinations in several sequential steps using two protocols: Hierarchical Leader-based (HL) and Multiphase Greedy (MG). Finally, multidestination messages have been used to optimize barrier synchronization and global reduction [24,25]. All these schemes use path-based multicast based on the BRCP model.

Most of the work described above for whormhole networks uses path-based multicast. Unfortunately, path-based multicast has several inefficiencies, especially when messages are short. The first inefficiency is that each multicast message needs a message preparation phase to order the destinations. Usually, this preparation phase involves a split-and-order function with a software cost of $O(n * logn)$, where n is the number of destinations. This preparation phase may take more time than the transfer itself. This is the main limitation of path-based multicast schemes when the latency of multicast messages is an important issue.

The second inefficiency is that, in most of the proposed schemes, path-based multicast does not use a minimal path for all of the destinations of a multicast message. As a result, more network resources are used and network contention increases. The third inefficiency is that, to prevent deadlocks, some path-based multicast mechanisms use a routing function that follows a Hamiltonian path. As a result, unicast routing must use the same routing function and, therefore, it can not exploit the advantages of other unicast routing functions. This limitation has been removed by the BRCP model. Finally, path-based multicast routing requires several delivery channels at each node to avoid deadlock [23].

In this paper, we propose a new tree-based multicast mechanism that overcomes the limitations of the previously proposed mechanisms. First, the new mechanism does not require an initial ordering of the destinations. This makes the message preparation phase much faster. This, however, does not affect deadlock freedom: a pruning mechanism guarantees deadlock freedom. A second advantage is that it can use a minimal path for all the destinations of a multicast message. A third advantage is that it is able to reuse any deadlock-

free routing algorithm used by unicast messages. Therefore, the routing flexibility for unicast messages is also available to multicast messages. Finally, tree-based multicast with pruning does not require several delivery channels to guarantee deadlock freedom.

Simulation results of networks under synthetic loads show that the new scheme can significantly reduce the latency of multicast messages. Furthermore, it has a higher performance than the other multicast mechanisms when the multicast traffic is composed of short messages like in DSMs. Finally, the new scheme can be easily implemented in hardware with minimal changes to existing wormhole routers.

The rest of this paper is organized as follows. Section 2 describes the new multicast mechanism. Section 3 analyzes deadlock avoidance. Section 4 explains the additional hardware needed to implement tree-based multicast and its impact on the critical path of a conventional router. Section 5 evaluates the scheme and compares it with other schemes. Finally, section 6 presents conclusions and future work.

2 Tree-Based Multicast with Pruning

The new scheme proposed in this paper is named Tree-Based Multicast with Pruning. Tree-based multicast has traditionally been considered a good mechanism for multidestination message routing. This mechanism was successfully used to broadcast and multicast messages in store-and-forward networks. However, with the arrival of wormhole switching, it became very prone to congestion and deadlocks in the interconnection network. As a consequence, other multidestination routing mechanisms like path-based multicast have been studied. In path-based routing, a multicast message visits destinations in an ordered and sequential way. With an adequate ordering of the destinations, a deadlock-free routing algorithm for unicast messages, and several delivery channels, deadlock-freedom is guaranteed [23].

Path-based mechanisms, however, are inefficient because the multicast destinations need to be ordered in the message preparation phase. This ordering involves a software cost of $O(n * logn)$, where n is the number of destinations. This cost may increase the total latency of each message considerably.

For this reason, we have reconsidered tree-based multicast, which does not necessarily require a destination ordering phase. Furthermore, if the base routing algorithm is minimal, each multicast message reaches all its destinations following minimal paths. Unfortunately, deadlocks may occur. This issue is addressed in Section 3 by prunning some branches of the tree. In the follow-

ing, we first describe the mechanism in detail and then consider its limitations and benefits.

2.1 Description of Tree-Based Multicast with Pruning

Each multicast message has several destinations, and for each destination it needs an address flit in the message header. In other papers, several schemes for encoding the destination addresses have been proposed [2]. These address encoding conventions reduce the number of flits in multicast messages. They are useful when short multicast messages with many destinations are used, in order to reduce the overhead involved with addressing information. In this paper, however, we consider the traditional all-destination scheme where each destination uses one address flit because the average number of destinations in coherence messages for DSMs is very small.

The operation of the new tree-based multicast mechanism is similar to the traditional one in some respects. In a multicast message, each address flit is routed along several intermediate nodes until it reaches its destination. These nodes decide the best path to follow. They can open a new path if they find that paths reserved by address flits already processed are not minimal. Therefore, a multicast message will be able to expand as many branches as needed in its advance toward the destinations. As we will see later, however, this does not increase the switch complexity at each node, since it is not necessary to connect an input with several outputs simultaneously.

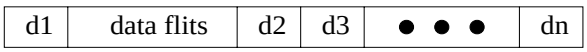

Fig. 1. Message format for tree-based multicast with pruning.

One of the differences in our scheme with respect to other multicast schemes is the way in which we organize the information in a message. Figure 1 shows the format of a multicast message. In this figure, we can see that the first flit of each message, d_1, corresponds to the first destination address. It is followed by the data flits of the message, and the remaining destination addresses, $d_2, d_3, ..., d_n$.

For our scheme to work correctly, the data flits must be stored in an auxiliary buffer at each intermediate node, even if it is not a destination. This is the reason why this mechanism is suitable for very short messages. However, the required buffer size is very small because multicast messages are very short in DSMs (just one flit in invalidation protocols). This buffer is associated with every input channel of every node. When the first address flit of a message arrives at an intermediate node, the following data flits are copied to the corresponding auxiliary buffer. This buffer stores the data until the tail of the

message leaves the node. Thus, when a destination address flit, d_i where $i > 1$, is routed at a node n_k, two things may happen:

- (a) If d_i opens a new path at node n_k, that is, it does not follow any path previously established by destination addresses d_1, ..., d_{i-1} in the same message, then node n_k must inject the data flits after sending d_i without interruption. Recall that the data flits are stored in the auxiliary buffer. Thus, d_i establishes a new branch in the multicast tree. The same operation is performed when the first address flit of any new branch of a multicast tree, d_i, arrives at an intermediate node.
- (b) If d_i uses a path previously established by d_j (with $j < i$) in the same message, then it simply crosses node n_k following that path. Data flits are not injected after transmitting d_i, because they were sent after transmitting the destination address flit d_j.

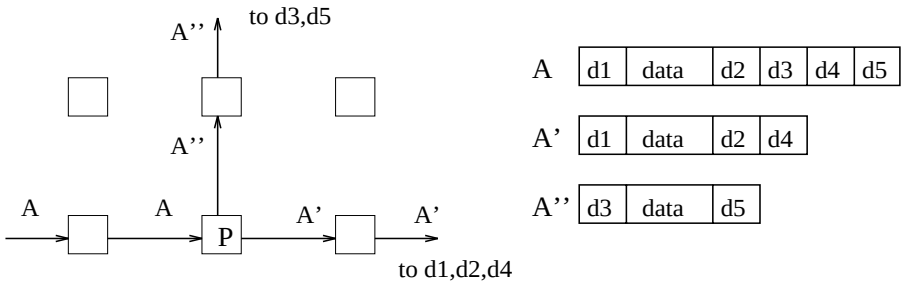

Fig. 2. A multicast branching example: The original multicast worm, A, is splited into two worms, A' and A", at node P.

It is important to note that the message format remains the same during the advance of the multicast worm. For each new branch, a "new" multicast message is expanded on-the-fly with the same format as the original one. In figure 2 we show an example of multicast worm propagation. The original worm, A, has five destinations, represented by the destination address flits $d_1, ..., d_5$, and one block of data flits. When this worm arrives at node P, the router supplies a free east channel for the destination address flit d_1. When d_1 crosses node P, the data flits are transmitted to the output channel selected by the destination address d_1, producing a new worm A'. These data flits are also copied into the auxiliary buffer associated with the corresponding input channel at node P.

After the data flits have crossed node P, d_2 is the next destination address flit that must be routed. The router decides that it must follow the path previously reserved by d_1 (to the east). Therefore, d_2 is transmitted to the output channel reserved by d_1. The next destination address flit that arrives at router P is d_3. Now, the router decides that a north channel must be used to reach that destination. Thus, a new branch is produced and the destination address flit d_3 is sent north, creating the A" branch. Next, the data flits (stored in the auxiliary buffer) are injected behind destination address flit d_3. When all the

data flits have crossed the switch at node P, the router continues with the next destination address d_4. This process is repeated until the last flit, d_5, leaves node P. When this occurs, the input channel and the associated buffer are freed.

2.2 Limitations of the Scheme

The main limitation of the proposed multicast mechanism is the data size of multicast messages. The data size must be small, typically a few flits only, because data need to be stored in an auxiliary buffer at each input channel of each node traversed by the message.

However, the size of invalidation or update multicast messages in distributed shared-memory multiprocessors with hardware cache coherence protocols is very small, typically one word. Thus, the proposed tree-based multicast mechanism is well suited for those architectures.

Anyway, if we consider a virtual cut-through switching network, this limitation would be overcome [26]. In this case, multicast messages of arbitrary length must be split into fixed-lenght packets, which can be easily stored at intermediate nodes allowing the proposed branching process. However, buffers must be large enough to store a whole packet. This constraint also applies to unicast routing.

2.3 Benefits of the Scheme.

The proposed multicast scheme has several advantages over the traditional path-based multicast routing proposed for wormhole networks: it does not requiere an initial ordering of the destinations, it can route all the multicast messages using minimal paths for all the destinations, it can use any deadlock-free routing algorithm for unicast messages and it does not require several delivery channels.

In this section, we would like to highlight the most important one: the impact of initial destination ordering on start-up time for multicast messages. Start-up time plays an important role when multicast messages are considered. In path-based multicast schemes, a preparation step is required for multicast messages, which orders the destination set to avoid deadlocks. This software-based ordering represents a considerable percentage of the total message latency, especially when the network is very fast and the messages are very short. Our proposed multicast scheme does not need any preparation for multicast messages. Indeed, the start-up time for a tree-based multicast message will be the

same as for unicast messages. We believe that this parameter will be crucial in the implementation of hardware multicast on real interconnection networks.

3 Deadlock Recovery in Tree-Based Multicast with Pruning

When the network is lightly loaded, tree-based routing works well for multicast and broadcast messages. For example, it is able to transport multicast messages with low latencies using a minimum number of channels. However, when the network is heavily loaded, the branches generated by tree-based routing increase contention and may cause deadlocks in the interconnection network.

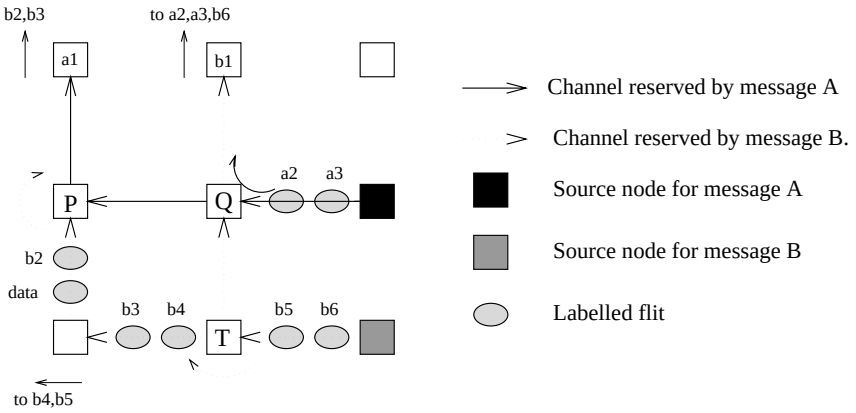

Fig. 3. A deadlock between two multicast messages A and B.

We propose resolving deadlocks and reducing contention by controlling multicast branches through a pruning mechanism. When one of the branches of a multicast message is blocked at a given node, a pruning of all the other branches of the message is performed at that node. Moreover, as flow control in wormhole switching stops flits in previous nodes, pruning is also performed at those nodes. Then, the pruned branches can freely advance and release channels that could block other messages.

For example, Figure 3 shows a portion of a 2-D mesh using XY routing where there are two multicast worms blocking each other. Message A has three destinations: a_1, a_2, and a_3. Message B has six destinations: $b_1, b_2, ..., b_6$. The first address flit of A, a_1, crossed Q and P and reached its destination. The first address flit of B, b_1, also reached its destination crossing node Q. The blocking state is reached when the header flits of each message, a_2 and b_2, cannot advance because the other message is using the requested channel. This blocking situation is indicated by arrows in Figure 3. Indeed, the path towards a_1 can not be freed because node Q is waiting to see if there are any more destinations in the message after a_2 that could use the same path to a_1. Similarly, the path towards b_1 cannot be freed because node T is waiting to see if there

are more destinations after b_5 that can use the same path to b_1. The result is deadlock.

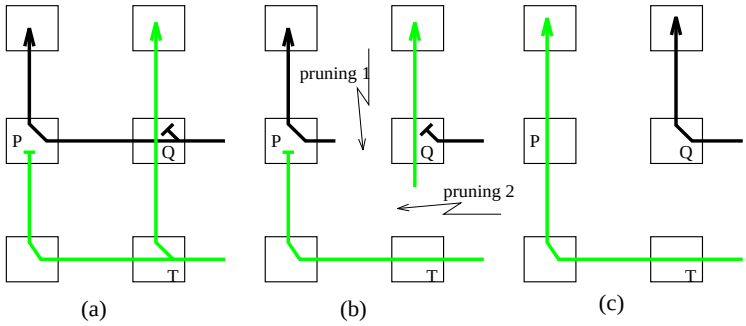

Fig. 4. Resolving the deadlock between multicast messages A and B.

To recover from deadlock, the pruning mechanism is used at nodes Q and T. Figure 4 presents an abstraction of the deadlock situation in Figure 3. The pruning scheme is represented here in three steps: (a) deadlock configuration; (b) pruning action; (c) routing destination address flits a_2 and b_2 at nodes Q and P, respectively.

When node Q routes a_2 and finds that there is no free output channel for it, a pruning of all the other branches of message A is performed at this node. In this case, the branch opened by a_1 is pruned, so that it can freely advance toward its destination (Figure 4(b), pruning1). Such pruning will release the channel at node P that is blocking the advance of message B. Then, message B will advance, eventually releasing the channel at node Q requested by message A. In the event that flow control also stopped the advance of flits at node T, message B would also prune its branch destined for b_1 (Figure 4(b), pruning2). This pruning is not necessary to recover from deadlock but shows that nodes performing a pruning do not need to synchronize.

We now show that the pruning mechanism described above is able to recover from deadlocks produced as a consequence of using tree-based multicast, assuming that the routing function for unicast messages is deadlock-free.

Lemma 1: Given an interconnection network I and a deadlock-free routing function R for unicast messages, a tree-based multicast mechanism based on R is deadlock-free if there is a total pruning function P that is used when a multicast message can not advance.

Proof: Let us assume that the base routing function R is deadlock-free for unicast message passing. A multicast message without branches has the same behavior as unicast messages regarding deadlock. Both multicast messages without ramifications and unicast messages use the same propagation scheme: the first flit establishes the path toward the destination and all the remaining flits follow it without branching. Since R is deadlock-free, there is not any deadlocked configuration for unicast messages. Now, we will proceed by contra-

diction. Suppose that tree-based multicast with pruning is not deadlock-free. Therefore, we will be able to find a deadlocked configuration. In a deadlocked configuration, no flit can advance. Also, a multicast message with branches cannot exist, because when multicast messages block, all the branches are prunned. Effectively, suppose that we have a multicast message with B branches at node Q. If pruning is performed at that node, the result will be equivalent to B unicast (or multicast without branches) messages. Thus, in a deadlocked configuration, no multicast message has branches. Therefore, we conclude that there is a deadlocked configuration for the routing function R that contains only unicast messages, contrary to the initial assumption. •

The pruning mechanism uses a total pruning scheme at the nodes where an address flit is blocked: all message branches are cut. This pruning scheme will be efficient if multicast messages have few destinations. However, in general, a total pruning is not necessary to guarantee the absence of deadlocks; a partial pruning would be sufficient. Currently, we are developing a partial pruning algorithm.

4 Router Design

In this section we describe the hardware extensions required to add support for tree-based multicast routing to a unicast router. A typical unicast router consists of a routing control unit, a switch, and several input and output channels with their corresponding channel controllers. The routing control unit selects the output channel for a message as a function of its destination node, the current node, and the output channel status. In most routers, the routing control unit can only process one message header at a time. The switch is a crossbar. Thus, it allows multiple messages traversing it simultaneously without interference. We will use the same crossbar as in unicast routers in order to minimize the changes in hardware. Physical channels can be split into several virtual channels. Virtual channels are assigned to the physical link using a demand-slotted round-robin arbitration scheme. Each virtual channel has an associated buffer at both input and output sides.

The input and output channel controllers manage virtual channel multiplexing and process every arrived flit. The input controllers send header flits to the routing control unit to establish the path through the crossbar. The routing control unit sets the crossbar. The subsequent flits are transferred from the input buffer to the corresponding output channel buffer. When the last flit of a message arrives at the input channel and traverses the crossbar to the corresponding output channel, the release notification is sent to the output channel controller. Then the path through the crossbar is released and when the last flit is transmitted, the output channel is freed. We assume that all the

operations inside each router are synchronized by its local clock signal.

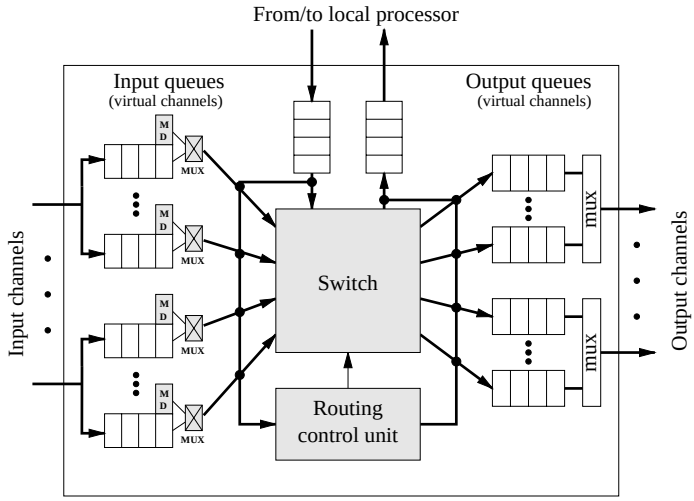

Fig. 5. Multicast extensions to an ordinary unicast router model.

Figure 5 shows the considered router model with some new parts as well as some modified. The most important modification will be at the input channel controller. The router requires an extra auxiliar buffer to store the multicast data flit (MD) as explained in the previous section. This buffer copies the data flit when it is stored at the queue head. The router needs a control signal to enable/disable the copy. Also a table of reserved output channels per input channel, a prunning register and some signals will be needed in the routing control and switch units.

When a message arrives, the message header has information about the message type (multicast or unicast) and the first destination. If it is a unicast message, it is processed as usually. If it is a multicast message, we know that the next flit will be the multicast data flit that must be stored in the auxiliary buffer. At this time the input channel controller changes to multicast mode. Thus, after routing and forwarding the header flit, it enables the copy of the next flit (multicast data flit) to the auxiliary buffer. That data flit is also transmitted following the header. At the same time, the input channel controller signals to the routing control unit the presence of new headers that have to be routed.

The following message flits are header flits. Each header is sent to the routing control unit, which checks if any of the reserved output channels for previous headers in the same message is valid to reach its destination. If a previously reserved output channel is selected by the routing function, then the header flit crosses to the output channel as if it were a data flit. However, if there is not any valid reserved output channel for this header, then a new output channel is reserved (if it is free), the reserved output channel set is updated, and a branching process is triggered at the input channel controller. This process consists of sending the header stored in the input buffer first, and

then sending the multicast data flit from the auxiliary buffer. If a branching process is required for the last header flit, then the release signal is not set until the data flit stored in the auxiliary buffer crosses the switch.

If the routing control unit does not find a free channel when routing a header after forwarding the multicast data flit, then a total pruning of the branches is required in order to prevent deadlocks as explained in previous section. This can be easily implemented using an XOR operation between the routing table entry and the reserved output channels associated to the current input channel. The result will activate the release procedure for the reserved output channels on the respective output channel controllers. The selection of the pruned output channels can be performed in paralell with the routing of next headers. Threfore, this procedure does not increase the routing cycle time.

The additional hardware does not change the timing for the switch and routing control units. The only overhead for the routing control unit is the overhead in the selection function, which has to check first if there is a reserved output channel among those suplied by the routing function (AND operation). This additional check slightly increases the selection delay (one gate delay). Also, the pruning action is very simple and can be made after the routing cycle, as stated below. Note that the timing for pruning is not critical because the messages involved in that operation are blocked.

5 Evaluation

We have developed a flit-level simulator for interconnection networks that supports unicast routing, path-based multicast routing and tree-based multicast routing with pruning. It takes as input parameters, the switching technique (wormhole or virtual cut-through), topology, routing algorithm, message size, message distribution, number of destinations for multicast messages, network size and number of virtual channels. In our experiments, we run several simulations to analyze the behavior of tree-based multicast routing against unicast routing and path-based multicast routing algorithms like Dual-Path [21] and PM [17]. The multicast mechanisms are evaluated for 2-D mesh, 2-D torus and 3-D torus topologies of different sizes. In the following, we first describe the simulation parameters and then compare the new scheme to path-based multicast and unicast routing.

5.1 Simulation Parameters

All multicast messages have only one data flit, a typical data size for invalidation and update messages in distributed shared-memory multiprocessors. The number of destinations of each multicast message varies between 4 and 25. A uniform distribution is used to construct the destination set of each multicast message. Deterministic routing algorithms are used for tree-based multicast and unicast in all the simulations. In particular, we use the dimension-order routing algorithm. For 2-D meshes, this is the popular XY routing algorithm. This routing algorithm does not require virtual channels. However, although tori also use dimension-order routing, two virtual channels per physical channel are required to avoid deadlock [4]. The routing algorithms for Dual-Path and PM have been described in [21,17]. In the multicast experiments of sections 5.2 and 5.3, traffic consists of multicast messages only. In section 5.4 we present simulations with a traffic pattern composed of unicast and multicast messages.

In all simulations, we have assumed that each physical channel has a bandwidth of one flit per clock cycle. Furthermore, both the switch and the circuit implementing the routing algorithm require one clock cycle to process a flit. In section 5.2, each router has four injection and delivery channels. Multiple delivery channels are required to avoid deadlock in path-based multicast algorithms [23]. However, in sections 5.3, and 5.4 each router has only one injection channel and one delivery channel. Several injection and delivery channels are not necessary to guarantee deadlock-freedom for tree-based multicast and unicast routing schemes. Each physical channel has queues with capacity for two flits at each end and one auxiliary queue at the input side with capacity for one flit. When physical channels are split into virtual channels, each virtual channel has two-flit queues at both ends and a one-flit auxiliary queue at the input side.

5.2 Comparative Evaluation of Different Multicast Mechanisms

In this section, we compare different multicast schemes on a 8×8 2-D mesh topology, using the simulation parameters. We compare our tree-based mechanism to two path-based schemes, namely the Dual-Path and the PM routing algorithm. In addition, we also compare it to unicast routing. Dual-Path uses a deterministic routing algorithm based on Hamiltonian paths, while PM is a partially adaptive routing algorithm based on the turn model [8]. We include the unicast mechanism as a reference.

We note that the startup time, needed to generate multicast messages in path-

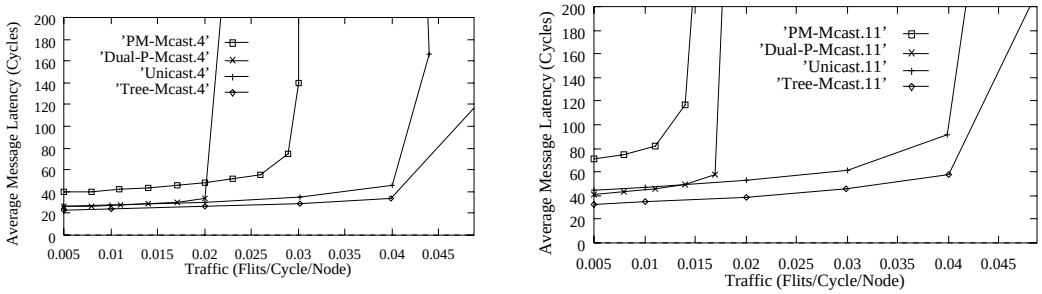

Fig. 6. Comparative evaluation of different multicast mechanisms for an 8×8 2-D mesh.

based multicast schemes, has not been considered when computing the network latency. If we included such startup time, the Dual-Path and PM schemes would increase their latency by the amount of cycles required to perform the message preparation phase of each message. Therefore, the results presented in this section for the Dual-Path and PM algorithms are optimistic.

Figure 6 shows the average message latency for the different multicast mechanisms using traffic composed of multicast messages with 4 or 11 destinations. Each plot has a label that indicates the associated multicast mechanism and the number of destinations of each message.

The PM algorithm (PM-Mcast in Figure 6) has a higher latency than the other algorithms for traffic conditions bellow saturation. However, it achieves a better throughput than the Dual-Path routing algorithm (Dual-P-Mcast in Figure 6) when messages have relatively few destinations (leftmost chart). When the number of destinations increases (rightmost chart), however, the PM algorithm has the worst latency and throughput. The reason is that the PM algorithm first routes a multicast message to its west-most destination following a deterministic path. This increases the distance travelled by multicast messages, thus increasing latency. Then the message reaches all the remaining destinations using a minimal adaptive routing function between sucessive destinations. On this second stage, the multicast message cannot go west in order to prevent deadlocks. Also, the overall path is not minimal. As a result, depending on the spatial distribution of the source and destination nodes, this mechanism will consume excessive network resources. Therefore, contention will increase and the network latency of multicast messages will increase too.

The Dual-Path algorithm achieves better results than the PM algorithm when the number of destinations grows. In general, Dual-Path divides each multicast message into two independent messages that follow different paths. Each of these messages will cover less destinations and use less network resources than PM. This is why DualPath is better than the PM algorithm, especially for many destinations. However, it reaches the saturation point very quickly if we compare it to unicast (Unicast in the figures) and the tree-based mechanism (Tree-Mcast in the figures). The reason is that DualPath almost only

uses the channels in the Hamiltonian path. It does not take full advantage of the remaining channels in the network. Additionally, the overall path is not minimal, consuming more resources than necessary.

Finally, we can see that tree-based multicast routing performs much better than path-based multicast routing for all traffic loads and number of destinations, even without considering the startup latency. Therefore, in what follows, we now focus on comparing our scheme to unicast routing. While Figure 6 assumed multiple delivery channels, we now consider the more interesting case of a single injection and delivery channel per node.

5.3 Tree-Based Multicast Versus Unicast

Tree-based multicast with pruning and unicast have been compared on several topologies. Figures 7, 8, and 9 show the average message latency (in clock cycles) as a function of traffic for a 8×8 2-D mesh, a 8×8 2-D torus, and an 8×8×8 3-D torus, respectively. In these figures, each plot has a label that indicates the associated multicast mechanism and the number of destinations for each message. As above, traffic is measured as the number of flits per cycle that are received by each node.

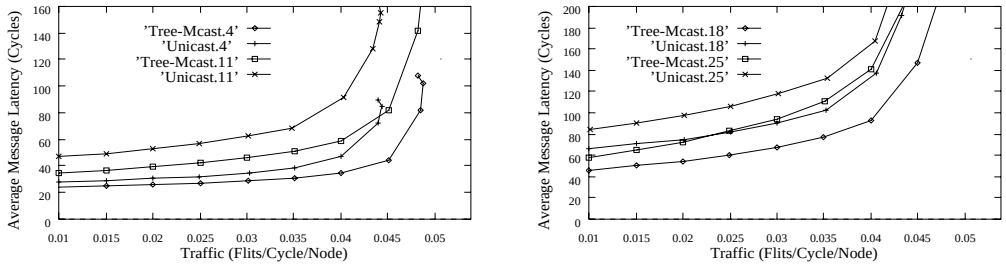

Fig. 7. Tree-based multicast versus unicast on a 8×8 2-D mesh with 4 to 25 message destinations.

On a 8×8 2-D mesh, the proposed tree-based multicast mechanism obtains latency reductions of up to 30% with respect to unicast routing. For both low and high number of destinations, it can be observed that, at low traffic, the difference between the proposed tree-based multicast mechanism and unicast routing increases proportionally with the number of destinations. Also, the saturation point for tree-based multicast occurs at a slightly higher traffic than for unicast routing.

On an 8×8 2-D torus, the improvement achieved by tree-based multicast is not as large as in meshes. The difference between both mechanisms is smaller. We can see that when messages have a small number of destinations, tree-based multicast still achieves a higher throughput. As the number of destinations increases, however, tree-based multicast degrades faster than unicast. When

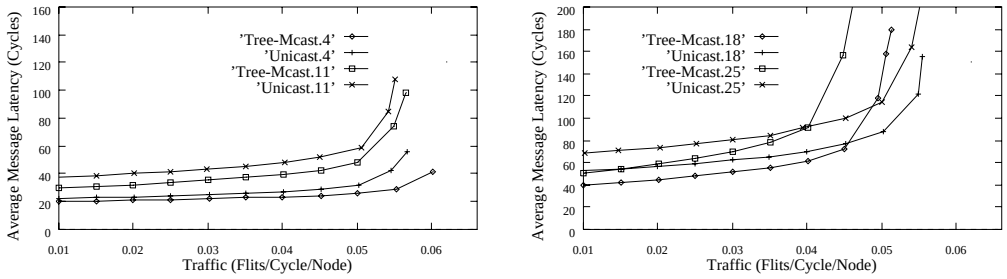

Fig. 8. Tree-based multicast versus unicast on a 8×8 2-D torus with 4 to 25 message destinations.

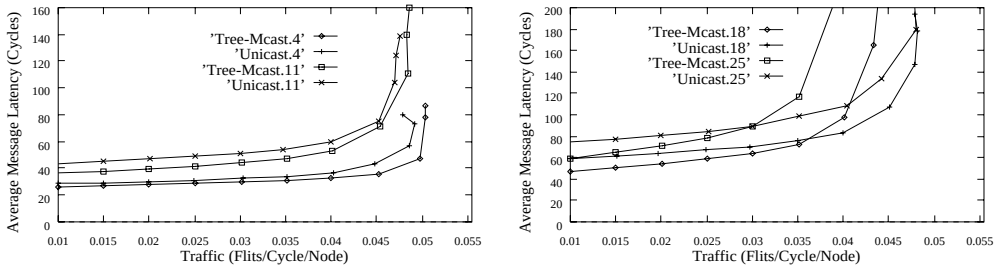

Fig. 9. Tree-based multicast versus unicast on a 8×8×8 3-D torus with 4 to 25 message destinations.

traffic load is high, and messages have many destinations, the unicast mechanism obtains better throughput than tree-based multicast. We believe that the reason is that tree-based multicast produces many branches, increasing contention considerably. The pruning mechanism recovers from deadlock, but pruning is not performed until flits have stopped. Therefore, some channels remain busy for longer than if messages were sent using unicast routing. However, tree-based multicast still achieves a lower latency than unicast routing almost up to the saturation point because data flits are transmitted only once.

We believe that the different behavior in meshes and tori is mainly due to the higher average number of branches per multicast message in tori networks. Note that multicast messages sent from nodes in upper (lower) rows of a 2-D mesh produce few branches in the north (south) direction.

On a 8×8×8 3-D torus the results are similar to the ones obtained for a 2-D torus. When traffic load is high and messages have many destinations, the unicast mechanism obtains even better results than those obtained with a 2-D torus when compared to tree-based multicast. The reason is that the number of dimensions is higher in a 3-D torus than in a 2-D torus. As a result, more branches are produced, especially when the number of destinations is high, increasing contention considerably. Nevertheless, tree-based multicast achieves lower latency than multiple unicast almost up to the saturation point.

In order to check the scalability of the proposed multicast scheme we have run several simualtions using larger networks. In particular, we show in Figures 10

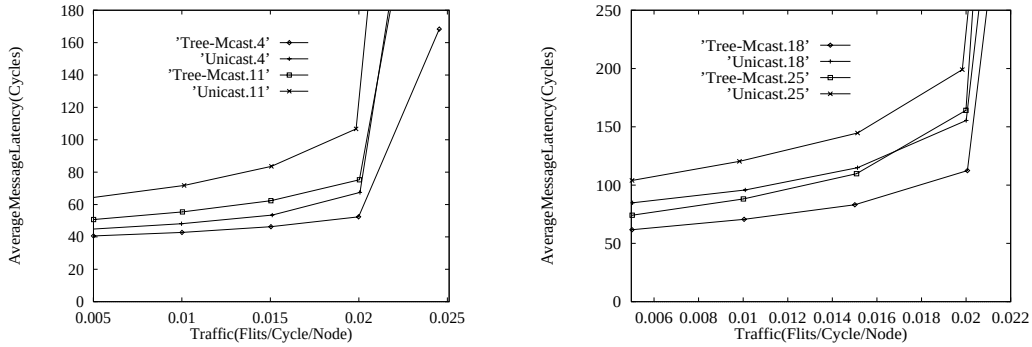

Fig. 10. Comparative evaluation of different multicast mechanisms for a 16×16 2-D mesh.

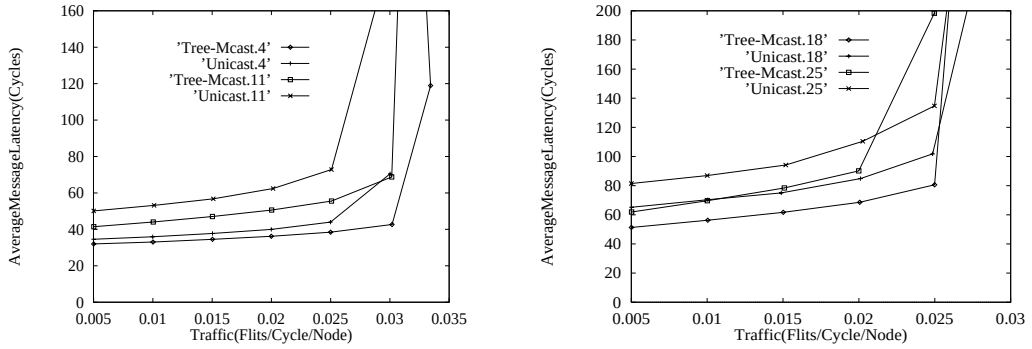

Fig. 11. Comparative evaluation of different multicast mechanisms for a 16×16 2-D torus.

and 11 the results for 2-D mesh and 2-D torus networks with 256 nodes. As can be seen, the plots are very similar to those obtained with smaller networks. So we can conclude that our tree-based hardware multicast scheme has a good scalability.

In general, we can see that, under tree-based multicast, performance decreases when the number of destinations in each multicast message increases. This occurs because multicast messages cannot travel long paths without branching. As a consequence, more pruning will be performed. Note that pruning is also performed when buffers are filled and flits stop advancing. Usually, address flits stop advancing several clock cycles before pruning is performed. Therefore, the bandwidth of the channels occupied by stopped flits is wasted. Thus, when traffic load increases and the number of destinations in each multicast message is high, tree-based multicast will decrease performance with respect to unicast routing. It is obvious that contention reduces the performance of this kind of multicast scheme. We can see this effect in 2-D torus and 3-D torus. Pruning is necessary to recover from deadlocks and to reduce contention. However, 2-D meshes exhibit good performance for any number of destinations, in all the cases we tried.

5.4 Tree-Based Multicast Evaluation with Mixed Traffic

In this section we analyze the performance using mixed traffic consisting of unicast and multicast messages. In Figure 12, we show the average message latency of tree-based multicast and unicast routing. The traffic pattern consists of 40% of unicast messages with 8 data flits per message and 60% of multicast messages with one data flit per message. This pattern may be representative of the traffic in a distributed shared-memory multiprocessor where updates and invalidations produce multicast messages and cache misses are served by unicast messages, each one containing a cache line (8 data flits).

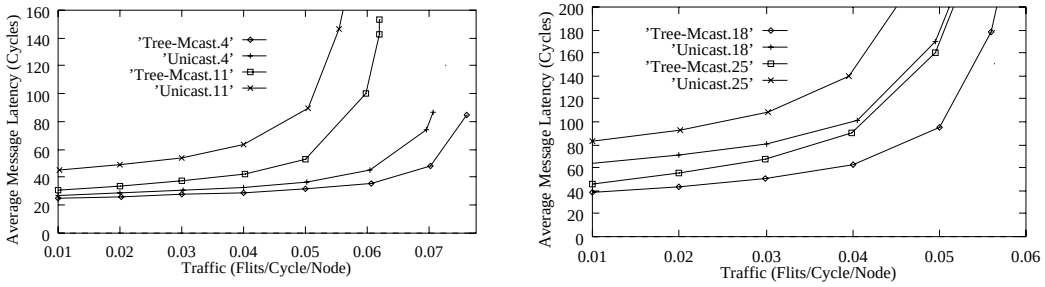

Fig. 12. Tree-based multicast routing versus unicast routing on a 8×8 2-D mesh using a mixed traffic pattern.

It can be seen in Figure 12 that, under this load, tree-based multicast still behaves much better than unicast routing. Thus, we expect the new multicast scheme to have a good behavior under real traffic. Although not shown, 2-D torus and 3-D torus topologies achieve similar results when this traffic pattern is used.

6 Conclusions and Future Work

This paper presents a fast hardware-supported multicast mechanism for wormhole networks. The advantages of the new scheme are that multicast messages do not need a pre-processing step that orders the destinations, messages reach their destinations following minimal paths if the base routing algorithm is minimal, it works for any topology, and it can reuse the routing algorithm for unicast messages. We call the new scheme tree-based multicast with branch pruning. The new scheme is deadlock-free and is particularly efficient for short messages, like those used to transmit invalidations and updates in DSMs.

We have evaluated the new scheme by running simulations with synthetic multicast loads. Multicast messages routed with the new scheme have significantly lower latency than if they are routed with path-based schemes. Furthermore, for high traffic loads, the network reaches substantially higher throughput.

We also show that tree-based multicast with branch pruning is better than unicast. The exception is when the interconnection network is close to saturation and the number of multicast destinations is very high. The reason for the latter effect is likely to be an excessive pruning in the new scheme that can be improved with a limited-pruning scheme.

We plan to analyze the performance of the new scheme by performing controlled pruning when the interconnection network is close to saturation and the number of multicast destinations is very high. In addition, we are in the process of refining our evaluation. Indeed, we plan to use SPLASH2 applications with multicast invalidation messages and a detailed simulation model of a DSM multiprocessor to evaluate the impact of the proposed scheme on overall execution time. We also plan to study efficient many-to-one communication schemes in order to develop a better support for acknowledgment messages in DSMs.

References

[1] G.T.Byrd, N.P.Saraiya, and B.A.Delagi, Multicast communication in multiprocessor systems, *Proc. Int. Conf. of Parallel Processing*, vol. I, pp. 196–200, 1989.

[2] C.M.Chiang and L.M.Ni, Multi-address encoding for multicast, *Proc. of the Parallel Computer Routing and Communication Workshop*, pp. 146–160, May 1994.

[3] D.Dai and D.K. Panda. Reducing Cache Invalidation Overheads in Wormhole DSMs Using Multidestination Message Passing, in *Int. Conf. on Parallel Processing*, Aug 1996.

[4] W.J.Dally and C.L.Seitz, Deadlock-free message routing in multiprocessor interconnection networks, *IEEE Trans. Comp.*, vol C-36, no. 5, pp. 547–553, May 1987.

[5] J. Duato, A new theory of deadlock-free adaptive routing in wormhole networks, *IEEE Trans. Parallel Distributed Syst.*, vol. 4, no. 12, pp. 1320–1331, Dec. 1993.

[6] J. Duato, A new theory of deadlock-free adaptive routing in wormhole networks, *Proc. 5th IEEE Int. Symp. on Parallel and Distributed Processing*, IEEE Computer Society Press, pp. 64-71, Sept. 1993

[7] J. Duato, A new theory of deadlock-free adaptive multicast routing in wormhole networks, *IEEE Trans. Parallel Distributed Syst.*, Vol. 6, No 9, pp. 976-987, 1995

[8] C.J.Glass and L.M.Ni, The turn model for adaptive routing, *Proc. 19th Annu. Int. Symp. Computer Architecture*, pp. 278–287, May 1992.

[9] J.L. Hennessy and D.Patterson. *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 1990.

[10] P.Kermani and L.Kleinrock, Virtual Cut-through: a new computer communication switching technique, *Comput. Networks*, vol. 3, pp. 267–286, 1979.

[11] J.Kuskin et al, The Stanford FLASH Multiprocessor, in *Proc. of the Int. Symp. on Computer Architecture*, 1994.

[12] Y.Lan, A.H.Esfahanian, and L.M.Ni, Multicast in hypercube multiprocessors, *Journal of Parallel and Distributed Computing*, pp. 30-41, Jan 1990.

[13] D.Lenoski et al, The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor, in *Proc. of the 17th Annual Symp. on Computer Architecture*, May 1990.

[14] D.Lenoski, J.Laudon et al, The Stanford DASH multiprocessor, *IEEE Computer*, 25(3):63-79, March 1992.

[15] X.Lin and L.M.Ni, Multicast communication in multicomputer networks, *Proc. Int. Conf. on Parallel Processing*, vol. III, pp. 114–118, Aug 1990.

[16] X.Lin and L.M. Ni, Deadlock-free multicast wormhole routing in multicomputer networks, in *Proc. 18th Annu. Int. Symp. Comput. Architecture*, May 1991.

[17] X.Lin, P.K. McKinley, and A.H Esfahanian. Adaptive multicast wormhole routing in 2D-mesh multicomputers, in *Proc. Parallel Architectures Lang. Europe 93*, June 1993.

[18] X.Lin, P.K. McKinley, and L.M. Ni. The message flow model for routing in wormhole-routed networks, in *Proc. 1993 Int. Conf. Parallel Processing*, Aug. 1993.

[19] X.Lin, P.K. McKinley, and L.M. Ni, The message flow model for routing in wormhole-routed networks, in *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no 7, pp. 755-760, Jul. 1995

[20] NCUBE Company, NCUBE 6400 Processor manual, 1990

[21] L.M. Ni and P.K. McKinley, Performance evaluation of multicast wormhole routing, in *Proc. Int. Conf. Parallel Processing*, I:435-442, 1991.

[22] D.K.Panda and S.Singal, Broadcasting in k-ary n-cube Networks using Multidestination Worms, *Proc. of Int. Conf. on Parallel Processing*, 1994.

[23] D.K.Panda, S.Singal, and P.Prabhakaran. Multidestination message passing mechanism conforming to base wormhole routing scheme, in *Proc. of the Parallel Computer Routing and Communication Workshop*, pp. 131-145, 1994.

[24] D.K.Panda, Global Reduction in wormhole k-ary n-cube networks with multidestination exchange worms, *Proc. of Int. Parallel Processing Symposium*, pp. 652–659, Apr. 1995.

[25] D.K.Panda, Fast Barrier synchronization in wormhole k-ary n-cube networks with multidestination worms, *Int. Symp. on High Performance Computer Architecture*, pp. 200–209, 1995.

[26] C.Stunkel, R.Sivaram, and D. Panda, Implementing multidestination worms in switch-based parallel systems: Architectural alternatives and their impact, *Proc. of the 24th ACM Annual International Symposium on Computer Architecture*, June 1997.

[27] S.C. Woo et al., The SPLASH-2 Programs: Chracterization and Methodological Considerations, in *Int. Symp. on Computer Architecture*, 1995.