Tree-Based Multicasting on Wormhole Routed Multistage Interconnection Networks*

Vara Varavithya and Prasant Mohapatra
Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011
E-mail: {vara,prasant}@iastate.edu

Abstract

In this paper, we propose a tree-based multicasting algorithm for Multistage Interconnection Networks. We first analyze the necessary conditions for deadlocks in MINs. Based on these observations, an asynchronous tree-based multicasting algorithm is developed in which deadlocks are prevented by serializing the initiations of branching operations that have potential for creating deadlocks. The serialization is done using a technique based on grouping of the switching elements. The preliminary simulation results are encouraging as it lowers the latency by almost a factor of 4 when compared with the software multicasting approach proposed earlier.

1 Introduction

Multistage Interconnection Networks (MINs) have been extensively studied and adopted as an interconnection fabric for multiprocessor systems. Multiprocessor systems increase their computing speed by performing several computations concurrently. These activities often require coordination and synchronization between processing elements through interprocessor communication which can be either one to one (unicast) or within a group of processors (collective). Most contemporary systems use wormhole routing for interprocessor communication.

An important communication primitive in collective operations is the multicast communication. Multicast communication is concerned with sending a single message from a source node to a set of destination nodes. The software-based approach has a higher latency as it incurs several communication steps [1]. To enhance the multicast performance, the multicast operations need to be supported at the hardware level. The multistage interconnection networks inherit the tree structure which can be effectively used to efficiently support multicast communication. In tree-based multicasting, a multihead message is sent out of the switch and the multiple headers are forwarded either synchronously or asynchronously. The tree-based multicasting scheme proposed by Chiang and Ni [2] requires the multicast headers in different branches to be forwarded synchronously. In asynchronous tree-based multicasting, multiple headers are allowed to be forwarded independent of each other. The asynchronous approach is preferred because of the ease of implementation. However, it is more prone to deadlocks. One approach to implement the asynchronous multicasting is through the use of buffers at each switch to prevent deadlocks [3, 4].

In this paper, we first analyze necessary conditions for deadlocks in the tree-based multicasting operations. Based on the study of deadlock problems, an asynchronous tree-based multicasting (ATBM) scheme is proposed for multicasting in MINs using the wormhole switching technique. To prevent deadlocks, the switches are grouped based on the layered networks and multiple tree operations are serialized within the groups. The proposed scheme is different from the previously proposed asynchronous [3] schemes in the sense that the multicast communication can be completed in a single start-up step and only small buffers are required for each input channels. The simulation results show that our approach performs significantly better than the software multicasting scheme while incurring low hardware overheads compared to the previously proposed multicasting schemes.

This paper is organized as follows. Preliminaries are presented in Section 2. The deadlock issues and the proposed algorithm are discussed in Section 3. Performance evaluations are given in Section 4. The concluding remarks are listed in Section 5.

2 Preliminaries

In this paper, we consider unidirectional MINs. The processor's communication channels are connected to the input ports of the MIN. The output ports of the MIN are connected (wrapped around) to the input ports of the processors. A MIN using $b \times b$ switches with S stages and R rows has $N = b^S$ processing nodes. A switch at row i and stage j is labeled as (i, j) where $[i \in (0, 1, \ldots, R-1)]$ and $[j \in (0, 1, \ldots, S-1)]$. In this paper, we consider the *baseline* networks as the basis of our discussions. However, other classes of MINs can also employ the same technique developed in our work.

To support the tree-based multicast operations, a replication mechanism is required. The degree of replication can vary between 2 and b (broadcast at the

*This research was supported in part by the National Science Foundation under the grants MIP-9628801, CCR-9634547, and CDA-9617375.

switch). The flit can be copied to the multiple output ports simultaneously or in sequence. The replicated flits can be forwarded either independently or synchronously.

The network throughput and communication latency are usually considered as the performance metrics of MINs. Throughput is the number of the packets delivered per unit time. In multicast communication, the multicast latency refers to the time between the message initiation and the reception of entire message at all the destinations.

3 Asynchronous Tree-Based Multicasting (ATBM) Algorithm

3.1 Deadlock Issues

Deadlocks in tree-based multicasting generally involve multiple multicast messages that perform the tree operations at the same stage. Consider the switch level deadlock configuration shown in Figure 1 (a). Message A from the upper port of switch 2 acquires the lower buffer of switch 3 and requests for the lower port of switch 4. Message B from the lower port of switch 2 holds the lower buffer of switch 4 and requests for the lower port of the switch 3. Both messages perform the switch broadcast operation at the same switch. Since both messages request the buffers that are held by the other, a deadlock cycle is formed. A variety of priority schemes are proposed in [2] to resolve this scenario.

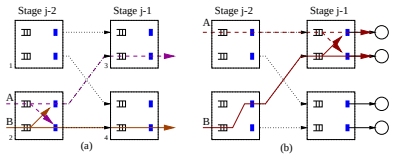

Figure 1: Switch level deadlock configurations.

If the switches in the last stage have the capability to process more than one incoming message concurrently, the throughput is increased. The number of incoming messages that the processor can concurrently absorb is usually dependent on the number of consumption channels. If the number of consumption channels is equal to the number of input ports, the messages that arrive the last stage will eventually be consumed. Figure 1 (b) shows a possible deadlock configuration developed from the tree operation at the last stage. This problem can be solved using multiple (b) consumption channels.

To illustrate the general concept behind the formation of deadlock cycles, the abstract deadlock configurations are shown in Figure 2. In Figure 2 (a), two messages perform a switch broadcast operation at the stage zero. These two messages request the same buffers at stage three. Deadlock occurs when messages acquire common buffers in one branch and

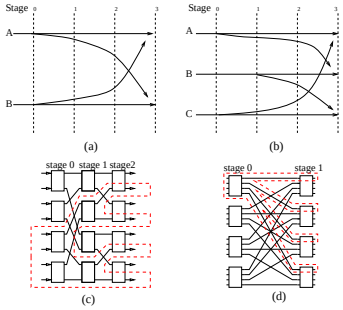

Figure 2: Abstraction of multicast deadlocks and network partitioning in MINs.

request the common buffers in the other branch. Figure 2 (b) show a possible deadlock that involves more than two messages.

A MIN using $b \times b$ switches can be viewed as a set of b overlapped networks. For example, we can partition the buffers and channels of a MIN using 2×2 switches into two separate sets, as shown in Figure 2 (c). These two separate sets have no buffers or channels in common. Similarly, the 16 nodes with 4×4 switches can be partitioned into four separate sets of buffers and channels as shown in Figure 2 (d).

Lemma 1: A buffer deadlock cycle can be formed by the multicast messages only when they belong to the same overlapped network.

Proof: Messages generated from different overlapped networks will not request for the same buffers. Therefore, the deadlock configuration needs to involve at least two messages in the same overlapped network. It is also important to note that deadlocks involving consumption channels can be formed.

Lemma 2: The deadlock configuration must involve at least two multicast messages that perform tree operations at the same stage.

Proof: When a switch broadcast operation is performed at the j^{th} stage switch, the upper branch message may use switches only in $\frac{N}{b^j}$ rows in the later stages (baseline network). If messages A and B perform broadcast operations only at j^{th} and $(j+1)^{th}$ stages, respectively, A can request the same buffer(s) in B's path. However, message B will not request the same buffers in the other branch of the message A. Therefore, *two multicast messages performing broadcast operations in different stages cannot create deadlock.*

3.2 The Proposed Algorithm

We propose a method that modifies the routing operations to prevent deadlocks in the asynchronous tree-based multicasting operation in MINs. On the basis of the necessary conditions for deadlock discussed in Section 3.1, we can form groups of switches at every stage such that only the multicast messages using the switches in the same group have a potential to

create deadlock configurations. Two multicast messages within the group may request the same buffers at any stage. From Lemmas 1 and 2, we can conclude that deadlocks are possible only when there are multiple tree operations in the same overlapped network at the same stage. From these observations we are motivated toward an asynchronous tree-based multicasting scheme that serializes the tree operations at every stage within a group of switches thus disallowing concurrent tree operation in a group at the same stage. Using this serialization technique, the deadlock cycles are prevented.

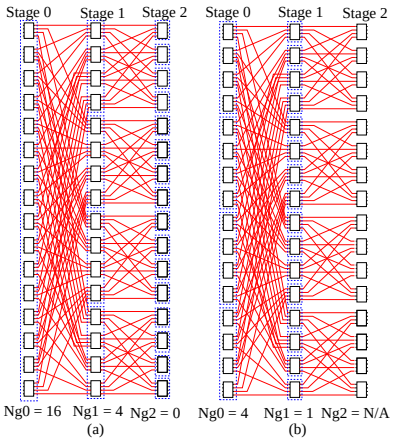

Figure 3: Switch grouping of the 64 nodes in a baseline network.

We divide the switch grouping methods into two cases: one for single consumption channel, and the other for b consumption channels. For the single consumption channel model, each switch constitutes a group at the last stage since multiple multicast messages at the last stage can only request the same consumption channels at the same switch. The tree operations that are initiated in the same switch (at the last stage for this case) can create the deadlock. Therefore, they are serialized to prevent the cyclic dependency. The last stage switches in the b consumption channels model do not have to be grouped since once the message reaches the input port, it will be eventually consumed by the processor. In the stage $S-2$, there are b switches in a group in the single consumption channel model. The number of switches in the group N_{gj} at stage ($j \in [0, 1, ..., j, ..., S-2]$) for the single consumption channel model is given by, $N_{gj} = b^{(S-1)-j}$ and for the b consumption channels model, $N_{gj} = b^{(S-2)-j}$. For the baseline networks, the (i, j) switch belongs to group $[k, j]$, i.e., the switch belong to group k at stage j. The group label $[k, j]$ can be calculated using the equation, $k = \lfloor \frac{i}{N_{gj}} \rfloor$.

The grouping example for the single consumption channel model is shown in Figure 3 (a) for 64 nodes using 4×4 switches baseline network. The dashed lines represent the switches in the same group at each stage of the baseline network. Figure 3 (b) shows the grouping for b consumption channels model. It is important to note that the number of switches in the group at each stage for b consumption channels model is significantly less than the single consumption therefore it allows more concurrent tree operations.

Routing Algorithm(message_header)

```
1. If (unicast_message)
       forward a flit to the output port specified
       in the routing_tag;
2. If (multicast_message)
       perform routing calculation;
       If (tree operation)
           wait to obtain the token;
           hold the token;
           replicate and forward the header flit;
           Release the token after the header flits
           arrive at the destination(s);
       else
           forward the header flit to the output ports;
```

Figure 4: The ATBM routing algorithm.

Additional hardware is required for serializing the tree operations in the switches of the same group. A control line can be implemented at each stage that interconnects the switches of a group. The token can be passed from one switch to another through the control lines in a round robin fashion within a group. When a switch is ready to do a multicast operation, it waits until it gets the token. The switch holds the token while multicasting a message and releases it after the header of message arrives at the destination. As the number of stages in MINs is usually small, the waiting time for the token will be within limits and multiple multicast messages can proceed if there is no blocking. This hardware modification is simpler and faster than the feedback mechanism required in synchronous multicasting [2] which requires the permutation of the whole multicast tree. The formal description of the algorithm is given in Figure 4.

4 Performance Evaluation

A flit-level wormhole routed network simulator is developed to evaluate the performance of ATBM algorithm. We design the experiments to evaluate the performance of the 2×2 and 4×4 switch-based baseline MINs networks. MINs of 16, 32, 64, and 256 nodes are considered in our study. The multicast header is assumed to fit in one flit using bit string encoding scheme [5]. The simulation parameters are set to the current trend of the technology.

To simulate a realistic environment, we generate a mixture of unicast and multicast traffic. The number of multicast destinations in a multicast operation

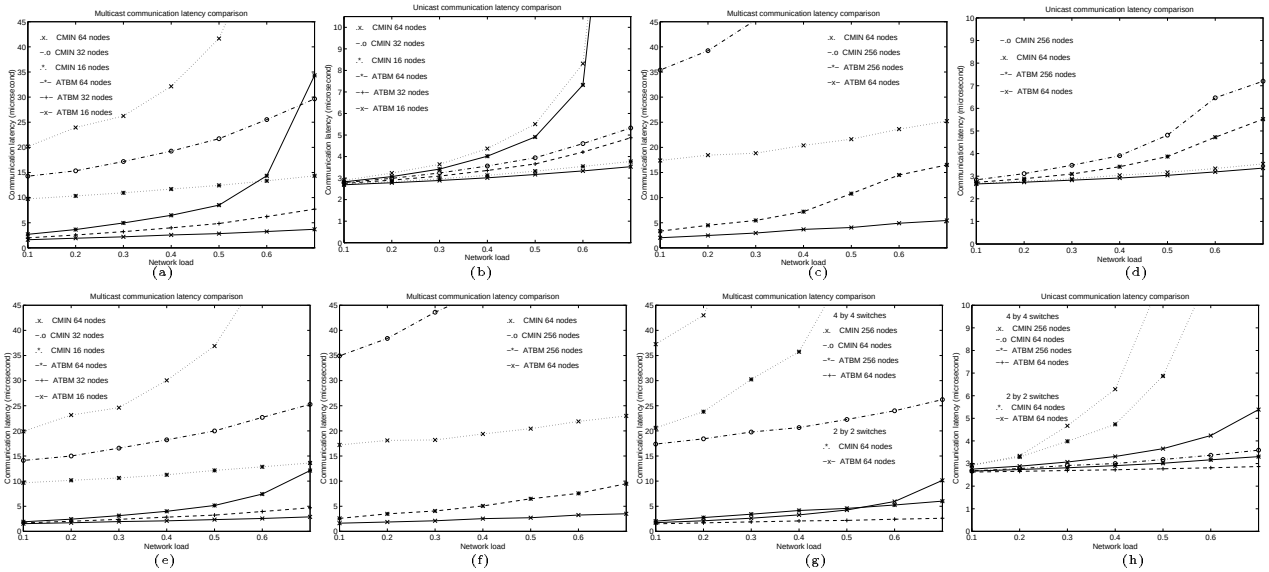

Figure 5: Performance comparison of ATBM and CMIN schemes.

is assumed to have an average of $\frac{N}{2}$ with a standard deviation of $\frac{N}{4}$. The size of unicast and multicast messages are fixed to 128 flits and 64 flits, respectively.

We compare the ATBM scheme with the CMIN algorithm [1] to show the performance improvement from supporting the multicast at the hardware level. The CMIN algorithm is the binomial unicast-based multicast algorithm which requires $\lceil \log_2(d+1) \rceil$ communication steps to complete the multicast operation to d destinations. The multicast tree structure is constructed such that the blocking is minimized.

Figure 5 (a) shows the multicast communication latency for baseline networks using 2×2 switches. The single consumption channel model is assumed. The mixture of the traffic is set to 20% multicast traffic with 80% unicast traffic. The ATBM algorithm has less multicast latency by a factor of four compared to the software multicast scheme. Figure 5 (b) shows the comparison of unicast latency. Both unicast and multicast latencies of the ATBM algorithm are lower than the respective latencies of the CMIN algorithm. At heavy load, the multicast operations tend to be blocked at the early stage. Therefore, they do not congest the network.

The simulation results for baseline MINs using 4×4 switches are shown in Figure 5 (c) and (d). Similar trends can be observed for both multicast and unicast latency. As discussed in Section 4, the b consumption channel model can enhance the performance of the ATBM scheme. Figure 5 (e) and (f) show the multicast latency under the b consumption channel model with a little performance improvement for the ATBM scheme compared to the one port model results. The proportion of multicast and unicast traffic is set to 50% for the simulation results in Figure 5 (h) and (g). The ATBM scheme gives even better performance for high ratio of multicast traffic.

5 Conclusions

We have examined the deadlock problems associated with asynchronous wormhole switching – based multicasting in MINs. The proposed tree-based approach avoids deadlock by serializing the messages that are prone to deadlocks. The deadlock prone messages are identified using a grouping technique. The grouping is based on the topological interconnection of the switches. The proposed technique uses less hardware while providing the same or better performance than previously proposed hardware multicasting scheme for MINs.

References

[1] C. Chiang and L. M. Ni, "Efficient software multicast in wormhole-routed unidirectional multistage networks," *Proc. Symp. of Parallel and Distributed Processing*, pp. 106–113, 1995.

[2] C. Chiang and L. M. Ni, "Deadlock-free multihead wormhole routing," *Proc. 1st High Performance Computing-Asia*, 1995.

[3] R. Sivaram, D. K. Panda, and C. B. Stunkel, "Fast broadcast and multicast on wormhole multistage networks using multiport encoding," *Proc. 8th Symp. on Parallel and Distributed Processing*, pp. 36–45, Oct. 1996.

[4] M. Malumbres, J. Duato, and J. Torrellas, "An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors," *Proc. 8th Symp. on Parallel and Distributed Processing*, pp. 186–189, Oct. 1996.

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