Broadcast on Clusters of SMPs with Optimal Concurrency

Yuzhong Sun, *David A. Bader*[#], *Xiaola Lin*[*], and *Yibei Ling*[+]

Streaming21 Inc.
Los Gatos, CA

[*]Dept. of Computer Science
City University of Hong Kong

[#] Dept. of Electrical and Computer Engineering, University of New Mexico

[+]Applied Research Laboratories
Telcordia Technologies, NJ

Abstract

In this paper, we present a hierarchical method for broadcast on clusters of symmetric multiprocessors (CSMPs) connected by switches with one-port model. We focus on the inter-switch broadcast that forms the core part of a broadcast on CSMPs. The proposed broadcast method is based on single-source shortest path minimum-cost spanning tree (SSS-MST). Two heuristic algorithms, from-up-to-down and from-down-to-up, are proposed to achieve the maximum concurrency using the information of the underlying network topology and the costs of links. Performance evaluation is also conducted to show the superiority of the proposed methods.

I. Introduction

Implementing an efficient broadcast in CSMP could be a challenge due to its irregular underlying interconnects. The symmetric multiprocessors (SMPs) in a CSMP are usually connected by a commodity network, such as Myrinet [BC95]. In a switch-based network, multiple SMP nodes are connected to one switch, and all switches are connected by certain topology. A message is transferred along a path of the switches from a source to a destination. A broadcast can be performed in a CSMP, first in the root switch where the source is located, then among switches, and finally within each switch except the root switch. Therefore, we can build a three-level hierarchy of a broadcast in CSMPs: the broadcast in a single SMP node, the broadcast within the switches called *intra-switch broadcast*, and the broadcast among switches called *inter-switch broadcast*. Apparently, the inter-switch broadcast forms the core part of a broadcast in a CSMP.

In this paper, we focus on the tree-based, inter-switch broadcasts in CSMPs. The tree-based techniques for broadcast have been proposed to designs various broadcast algorithms in many different interconnects, such as hypercube [JH89], and 2D-mesh [BMT92], wormhole-routed networks [MXEN94], Myrinet [BPDS00], and arbitrary topology [RM99]. Distinguished from the existing tree-based broadcasting algorithms, the proposed broadcast algorithm aims at achieving the maximum step concurrency, minimum steps of message passing among switches, and minimum cost for a broadcast to get high broadcasting throughput.

In the proposed method, several techniques are applied to perform a broadcast based on a minimum-cost spanning tree (MST). A single-source MST (SSS-MST) has been used to serve as a backbone for broadcast due to its minimum cost for broadcast under all-port model. Most CSMPs, however, only support one-port model. Thus, the SSS-MST method cannot provide support to step concurrency in inter-switch broadcast under one-port model. In the proposed method, link sharing is allowed with respect to its bandwidth. We use the static configuration information including topologies and link bandwidth to construct the SSS-MSTs for broadcast, following by two major techniques to optimize the step concurrency in the two directions: from up to down and from down to up. To show the superiority of the proposed method, we conducted the performance evaluation for the algorithm, and present the experimental results of the proposed algorithm.

II Broadcast in CSMPs

In this paper, we assume that a CSMP consists of a set of SMP nodes interconnected by a set of switches. The underlying network can be modeled as a graph in which communication is subject to the latency and the bandwidth properties of the underlying network. An SMP node (or SMP) is a shared-memory multiprocessor. Each SMP contains a number of identical processors, and has uniform access to the shared memory and other resources in the SMP. The CSMPs adopt one-port model such that each node can send and receive at most one message simultaneously. A *broadcast* is an operation in which a source (or root) processor broadcasts a message to all other processors in a broadcast group. All the processors in a broadcast group are referred as *working members*. Each SMP is connected to one switch in the network. A *working* SMP refers to an SMP that has at least one processor in the broadcast group. Similarly, a *working* switch is defined as the switch that connects at least one working SMP. We select an arbitrary working member as a leader for each working switch. Given a switch-based CSMP, we may have a graph $G = (V, E)$, where *V* corresponds to a set of switches on the CSMP, and *E* corresponds to a set of links between any pair of switches. Each link may consist of a couple of channels that may simultaneously transfer different messages [BPDS00]. The three-level hierarchy of a broadcast in CSMPs includes the broadcast in a single SMP node (Level 0), *intra-switch broadcast* (Level 1), *inter-switch broadcast* (Level 2).

The proposed hierarchical broadcast covers the two important techniques used in [MSP99]: switch-ordered ring and link scheduling. The essence in the switch-ordered ring technique is to order the processors by switch to eliminate potential link contention. For each switch, only one local processor sends message to another processor on the other switch and only one remote processor receives message from a local processor. On each level, each working SMP (working switch) should select a leader processor (leader SMP). Only these leader processors (leader SMPs) can communicate with other SMPs (switches). The proposed method puts no restriction on the underlying topology for inter-switch broadcast except the switch-ordered ring in [MSP99]. The topology for inter-switch broadcast may be arbitrary (e.g., ring and tree). In the broadcast hierarchy, Level 0 and Level 1 can be implemented easily in hardware in SMPs and switches respectively.

2.1 Using Single-Source Shortest-Path MST for Inter-Switch Broadcast

One of the design objectives for optimal broadcast is to have high broadcast throughput. An inter-switch broadcast should have as little negative impact as possible on the underlying interconnect for switches. A good switch routing should minimize the number of switches traversed by a message while making efficient use of the links as possible to reduce possible congestion and latencies.

In the proposed hierarchical algorithm, single-source shortest path minimum-cost spanning trees (SSS-MST) is applied to order the switches in the inter-switch broadcast on switch-based CSMPs. The SSS-MST problem is defined in the inter-switch broadcast as follows: Given a $G_w = (V, E)$, we intend to find a shortest path from a given source vertex $s \in V$ to every vertex $v \in V$, and deduce an SSS-MST T_w. We apply the well-known Dijkstra's algorithm in [D59] to produce a T_w for the G_w. The complexity of the Dijktras' algorithm is $O(|V[G_w]|^2)$.

2.2 Heuristics for Optimal Inter-Switch Broadcast

The SSS-MST-based inter-switch broadcast may suffer from sequential effect under one-port model, which arises from a node with multiple children. The sequential effects prevent the step concurrency on an SSS-MST under all-port model.

The key to reduce the sequential effect is to find appropriate candidates of parents for nodes affected in the broadcast tree to increase the step concurrency in each step. The link connecting any two switches consists of a couple of channels. Each channel can transmit any messages independent of the other channels in the link. In the case that multiple working processors are connected to one switch, all of these working processors in one switch can serve as senders which forward simultaneously to another connected switch along those independent channels on the link between them.

The following two major techniques are proposed in this paper to optimize the step concurrency in the two directions.

Down-to-up algorithm: The down-to-up algorithm is to calculate the longest path *p* from one node to its descendants in the G_w. Any descendant of this node can receive the message broadcasted from the node in the weight $w(p)$ of this path. The $w(p)_v$ covers link weights and processor start-ups in node *v*. It is desirable to find the maximum $w(p)$ from any node *v* to all its descendants. According to the $w(p)_v$ of node *v*, we can decide a sending order for nodes affected by a sequential effect. The down-to-up algorithm can globally optimize the send order for the children incident on any node *u*.

For a switch-based cluster of SMPs, the procedure of transferring of a message from one switch to another can be divided into the two phases: the start-up and transmission. The latency incurred by the start-up is usually larger or even much larger than that of the transmission. The down-to-up algorithm can guarantee that the node with the longest level of the descendant in G_w always has the highest send priority. This strategy can overlap the transmission time for the descendants of a node to reduce the overall execution time for the broadcast.

```
For an $SG_w$ and the its height l:
Assume that the root is at the level 1, the step for the root is zero, and each leaf i has that $w(p)_i = 0$.
Step 1:  while ($l \neq 1$) Then {
Step 2:  Each node u in the level l sends the sum $\sum_u$ of its $w(p)_u$, and the weight of the link to its parent to its parent. Notice that the $\sum_u$ should include the start-up of the node u if it isn't a leaf;
Step 3:  Each node v in the level $l-1$ selects the maximum $\sum_u$ from all its children as its $w(p)_v$;
Step 4:  Each node u in the level l decides the send order for all its children in the descending order of the $w(p)$'s of all its children;
Step 5:  Decrease l by one. }
Step6:  Do the Depth-First-Search algorithm from the root and give each node a step number representing that the node should send a message.
```

Figure 2.1 The down-to-up algorithm under on-port model.

Up-to-Down algorithm: This algorithm is used for the SSS-MST in an inter-switch broadcast. There are two rounds for the up-to-down algorithm. The first round executes the breadth-first-search (BFS) over the SSS-MST starting from the root of the SSS-MST to find all the nodes with at least two descendants. Under one-port model, the nodes may suffer from sequential effects. During the BFS, all nodes searched are sorted according to their upper bounds on the weights of the shortest paths from the source node, which form a set S_H. This operation can be easily done because the SSS-MST, in fact, is a partly-sorted tree in the sense that the upper bound of a node should be greater than the upper bounds of any its predecessors. Therefore, when a node *v* with *m* children is searched out, we will select $m-1$ appropriate nodes in the current S_H as the direct predecessors of those $m-1$ children. If $S_H \neq \phi$, then each of the children will have different direct predecessor. This procedure is called *paralleling procedure*. In the second round, the well-known depth-first-search algorithm is used to find all sequential effects of the second case over the SSS-MST handled by the first round. For each of the found sequential effects, a paralleling procedure is applied to reduce the sequential effect.

The gap between the start-up and the transmission may significantly affect the finding of a predecessor for a node in the paralleling procedure to remove sequential effects. Thus, when making a choice of the predecessor for a node in the paralleling procedure, we should consider the two factors: the start-up and the bandwidth between switches. The former has been calculated in the down-to-up algorithm, while the latter has been given in the form of the weights on the graph G_w. According to our assumption, the $V(G_w)$ only covers all the working switches on the cluster of SMP. The shortest path between two nodes *u* and *v* is calculated in the G_w, not for the network topology. The weight for a link between the nodes *u* and *v* are calculated by the default routing $R_{cluster}$ on the network topology. For a switch-based cluster, the $R_{cluster}$ may be shortest or the others selected by the users. Therefore, selecting the predecessors for the paralleling procedure on one node *v* can be done in S_H corresponding to the node *v* by the shortest-paths from the node *v* to the source crossing these predecessors. The weights for these shortest-paths are the sums of the weights of the path from the node *v* to these predecessors by the $R_{cluster}$, and the weights of the shortest-paths from these predecessors to the source on the G_w.

A *window-slipped* technique is applied to gradually increase the range for the search. If the node *v* has *m* descendants, we specify a window of $m-1$ nodes and slip the window from the node *v* gradually until the $m-1$ predecessors are found. Meanwhile, when we use the window-slipped technique, some contention for usage of the links may occur. Therefore, we also should consider the bandwidth of the links calculated by the $R_{cluster}$. We should guarantee that the number of paths contented for one link should not exceed the number of channels on the link. The up-to-down algorithm is given in Figure 2.2.

Given a graph G for the inter-switch topology on a CSMP, and source s and its SG_w and the SSS-MST handled by the down-to-up algorithm:

Step 1: Execute the BFS algorithm to find all the possible sequential effects of the first case and use the paralleling procedure to remove them;
Step 2: Execute the DFS algorithm to find all the possible sequential effects of the second case paralleling procedure to remove them.

Figure 2.2 The up-to-down algorithm for removing all the possible sequential effects on the SSS-MST.

Figure 2.3 depicts the algorithm for the paralleling procedure to remove the sequential effects.

Given a graph G and $R_{cluster}$ for the inter-switch topology on a CSMP, and source s and its G_w and the SSS-MST handled by the down-to-up algorithm, assume $S_H = \phi$ and S_H is the holding set, each element of which holds the message.

Step 1: At node v searched out by the BFS in the Step 1 of the up-to-down algorithm or the DFS in the Step 2 of the up-to-down algorithm, $S_H = S_H + \{v\}$ and sort the S_H in the ascending order of the weights of the paths from the nodes in S_H to the source s.
Step 2: if node v is a sequential-effected node with m children in the temporal axis (e.g., it is either the first case or second case), we build a window of the size $m-1$. Using the size, the $m-1$ predecessors of the node v are selected as candidates for the direct parents of the $m-1$ children in the right-to-left direction on the S_H. Then check each $u_i, 0 < i < m$, of the candidates:
Step 3: for each u_i, if the step number of the node u_i < the step number of the node v, then the node u_i is invalid, got Step 7; otherwise do Step 4-6.
Step 4: calculate the sum $\sum u_i$ of the $d(u_i)$ on the SSS-MST handled by the down-to-up

algorithm and the weight of the path from the node v to the node u_i by the $R_{cluster}$.

Step 5: if the start-up of the node u_i plus the weight from the node u_i to the node v is less than the sum of the multiplication of the start-up of the node v by the step number in which the node u_i receives a message from the node v, and, the sum of the weights of all the links whose step members are less than the step number of the link between the nodes u_i and v, then,

Step 6: if each link along the path from the node u_i to the node v doesn't exceed the number of channels of this link, then the node u_i is a valid predecessor.

Step 7: if the number of valid predecessors is less than $m-1$, the window is left-shifted on the S_H, go to the Step 3 to check each candidates in the new window until all the overall S_H is searched.

Step 8: Sort the valid predecessors k ($0 \le k < m$) in the ascending order of the weights of the path from the source s to the node v traversing these predecessors. The node v sends a message along the link with the least step number j. The children denoted by the links of the step numbers from $j+1$ to $j+k$ are mapped to the sorted predecessors. Those links are removed while the step numbers of the node v is subtracted by k for those greater than $j+k$. Each valid predecessor is added a new step number equal to the least step number of the node v, while a new link is added between the predecessor and its mapped child of the node v on the G_w, its weight is the weight of the path from the predecessor to the child crossing the node v.

Figure 2.3 The algorithm for the paralleling procedure to remove sequential effects.

III. Performance Analysis

We conducted an experimental comparison of the proposed hierarchical broadcast algorithm and the MPI standard broadcast function MPI_Bcast(), varying the numbers of SMP nodes and switches to measure the execution times for these two broadcast algorithms on the parallel architecture, a Linux Supercluster. The MPI_Bcast uses a tree-based algorithm to broadcast the message from the root processor to blocks of processes in a group denoted by MPI_COMM_WORLD. A linear algorithm is then used to broadcast the message from the first process in a block to all other processes. The MPI_Bcast does not consider the optimization of the broadcast tree using the topological and bandwidth information. The MPI_Bcast uses the two-level hierarchy that is determined by MPIR_BCAST_BLOCK_SIZE.

The Myrinet-connected Linux Supercluster, called Roadrunner, is located at the Albuquerque High Performance Computing Center (AHPCC) of the University of New Mexico, USA [B99]. Roadrunner is an Alta Technology Corporation 64-node AltaCluster containing 128 Intel 450 MHz Pentium II processors. The supercluster runs the 2.2.12 Linux operating system in SMP mode with communications between nodes provided via a high-speed Myrinet network (full-duplex 1.2 Gbps). Each node contains components similar to those in a commodity PC, for instance, a 100 MHz system bus, 512KB cache, 512MB ECC SDRAM, and a 6.4 GB hard drive. The Myrinet topology consists of four octal 8-port SAN switches (M2M-OCT-SW8 Myrinet-SAN), each with 16 front-ports attached to each of 16 nodes, and 8 SAN back-ports attached to the other switches. We used the MPICH-GM implementation of MPI for Myrinet.

To compare the MPI_Bcast and the proposed hierarchical broadcast algorithms, we run two threads on each SMP nodes consisting of two processors. The thread 0 on each SMP node takes responsibility for the intra- and inter-switch broadcasts. The MPI_Bcast does not support the thread broadcast among threads on one SMP node. In our experiments, the SMP broadcast (Level 0) broadcast in the hierarchy is omitted for simplicity. The *self-congestion* technique is used for the congested message transmission suffer from the necessary congestion. Large-size messages (200,000 bytes) have been used in our experiments.

We chose four cases to compare the MPI_Bcast and the proposed algorithms. The sizes of the SMP nodes in the four cases are 64, 44, 38, and 28 respectively. Except the case of 64 SMP nodes, the other three cases obtain their configurations by the automatic system schedule, *Portable Batch System* (PBS), that provide a resource list for each batch job. The resource lists of jobs in the PBS give insight to the dynamic configurations of those jobs such as how many switches are occupied and how many SMP nodes are attached to each of those switches. In each case, we do a series of experiments based on the same configuration but varying the number of SMP nodes involved in one broadcast, i.e., the range of the SMP nodes are 8 to 64 in the case of 64 SMP nodes. Thus, in each case, we can do the experiment in various sub-configurations.

The following experiments indicate that the proposed broadcast algorithm outperforms the MPI_Bcast in the above four cases, shown in Figure 5.1, 5.2, 5.3, and 5.4 respectively. In general, the proposed broadcast reduces the execution time for broadcast by 20% compared to the MPI_Bcast method. When the number of switches is greater than or equal to three, the proposed broadcast algorithm outperforms the MPI_Bcast by 30%. Even in the experiments using one switch, the execution time of the MPI_Bcast is often about 25% higher than that of the proposed algorithm, which shows that the proposed intra-switch broadcast algorithm outperforms the linear or tree-liked algorithm used by the MPI_Bcast.

V. Related Work

The technique of using tree in broadcasting has been proposed for a long time. Many researchers developed tree-based broadcast algorithm by taking advantage of the topological characteristics of the underlying interconnects, e.g., 2D-mesh in [BMT92], hypercube in [JH89]. These algorithms cannot be ported to other network topologies due to their dependence on the underlying topologies. Some researchers presented portable tree-based broadcast algorithms for various networks [BC95][JH89].

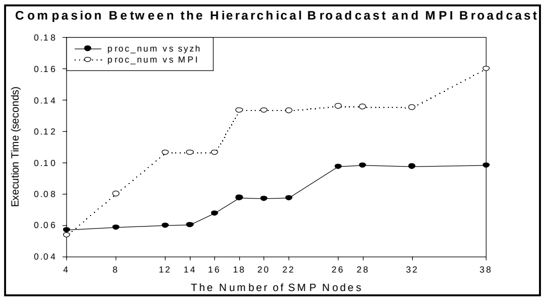

Figure 5.1 The experiments using up to four switches, up to 64 SMP nodes.

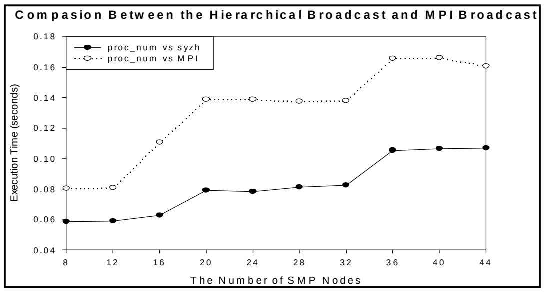

Figure 5.2 The experiments using up to three switches, up to 44 SMP nodes.

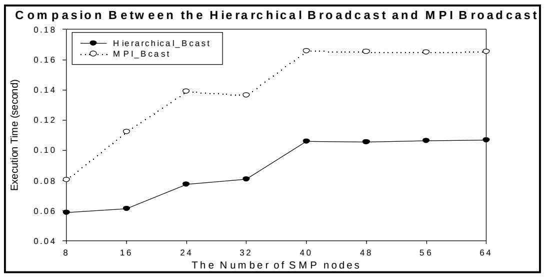

Figure 5.3 The experiments using up to four switches, up to 38 SMP nodes.

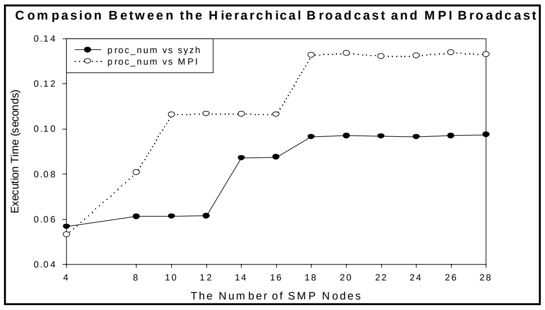

Figure 5.4 The experiments using up to four switches, up to 28 SMP nodes.

McKinley *et al* in [MXEN94] implemented multicast communication in wormhole-routed direct interconnects in the absence of hardware multicast support. The method exploits the properties of the switching technology on *n*-dimensional mesh and hypercube and use deterministic dimension-ordered routing of unicast messages. The algorithm cannot guarantee the minimum time on either other regular topologies such as Torus and star graphs or irregular topologies. Jacunski *et al* in [MSP99] presents an all-to-all broadcast on switch-based clusters of workstations. The switch-order ring and link scheduling techniques are proposed to approach to the optimal properties such as the degree of pipelining of communication components, the minimal transmission latencies and minimal the node. The algorithm is not contention-free for the usage of the links. Buntinas *et al* in [BPDS00] constructs optimal multicast trees by a simple top-down greedy algorithm under the postal model. Raghavan and manimaran in [RM99] proposes a re-arrangeable algorithm for the construction of delay-constrained dynamic multicast trees. The method uses the concept of quality factor to describe the usefulness of a portion of the multicast tree to the overall multicast tree.

VII. Conclusions

In this paper, we have proposed an efficient broadcast algorithm for the switch-based clusters of SMPs. The proposed method is based on single-source shortest path minimum-cost spanning tree. We have used the down-to-up and up-to-down heuristics to optimize the step concurrency while remaining the minimum cost and optimal number of steps for a broadcast. The experiments we conducted show that the proposed hierarchical broadcast outperforms the popular MPI broadcast.

References

[BC95] N. J. Boden *et al.*, "Myrinet: A Gigabit-per-Second Local Area Network," *IEEE Micro*, PP. 29-35, Feb. 1995.

[JH89] S. L. Johnsson, and C. T. Ho, "Optimal Broadcasting and Personized Communication in Hypercube," *IEEE Transactions on Computers*, vol.38, no.9, pp.1249-1268, Sept. 1989.

[BMT92] J. C. Bermond, P.Michallon, and D. Trystram, "Broadcasting in Wraparound Meshes With Parallel Monodirectional Links," *Parallel Computing*, vol.18, pp.639-648, 1992.

[MXEN94] 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*, vol.5, no.12, pp.1254-1265, 1994.

[BPDS00] D. Buntinas, D. K. Panda, J. Duato, P. Sadayappan, "Broadcast/Multicast over Myrinet using NIC-Assisted Multidestination Messages," *Proc. of Fourth Int'l Workshop on Communication, Architecture, and Applications for Network-Based Parallel Computing*, Jan 2000.

[RM99] S.Raghavan, and G. Manimaran, "A Rearrangeable Algorithm for the Construction of Delay-Constrained Dynamic Multicast Trees," *IEEE Transactions on Networks*, vol.7, no.4, Aug. 1999.

[MSP99] M. Jacunski, P. Sadayappan, and D. K. Panda, "All-toAll Broadcast on Switch-Based Clusters of Workstation," *Proc. of International Parallel Processing Symposium*, April 1999.

[D59] E. W. Dikstra, "A Note on Two Problems in connexion with graphs," *Numerische Mathematik*, vol.1, 269-271, 1959.

[B99] D. A. Bader *et al.*, "Design and Analysis of the Alliance/University of New Mexico Roadrunner Linux SMP SuperCluster," *Proc. of the 1st IEEE Computer Society International Workshop on Cluster Computing*, 1999.