Broadcast on Clusters of SMPs with Optimal Concurrency

Authors

Yuzhong Sun and David Bader
The University of New Mexico
Dept. of Electrical and Computer Engineering

X. Lin
The University of Hong Kong
Dept. of Electrical and Computer Engineering
Disclaimer

The High Performance Computing, Education & Research Center (HPCERC) provides a focus for high performance computing and communication at the University of New Mexico (UNM). HPCERC is committed to innovative research in computational and computer science with emphasis on both algorithm development and application. As part of this commitment, HPCERC sponsors this technical report series. The technical reports are subject to internal review by the HPCERC. However, the material, as presented, does not necessarily reflect any position of the HPCERC. Further, neither UNM, nor the AHPCC, makes any warranty or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information contained in this report.

Frank L. Gilfeather, Executive Director, HPCERC
Brian T. Smith, Chief Scientist, HPCERC
John S. Sobolewski, Chief Technologist, HPCERC
Ernest D. Herrera, Associate Director, HPCERC
Susan Atlas, Associate Director, AHPCC Computer Systems Research
Bob Ballance, Associate Director, AHPCC Science & Engineering Research
Broadcast on Clusters of SMPs with Optimal Concurrency

Yuzhong Sun+, David Bader+, and X. Lin*

+Department of Electrical and Computer Engineering
University of New Mexico

*Department of Electrical and Electronic Engineering
University of Hong Kong

Abstract—Broadcast is an important collective communication for either users’ programs or the underlying high-performance computing platforms. In this paper, we present a hierarchical method for broadcast over clusters of SMPs (CSMPs) connected by switches under the one-port model. A broadcast over CSMPs consists of three levels, one for a sub-broadcast in each SMP node, one for a sub-broadcast within each of switches called intra-switch broadcast, and one for a sub-broadcast among switches called inter-switch broadcast. Regarding high-performance of switches, our concern focuses on inter-switch broadcasts. The new inter-switch broadcast is based on Single-Source Shortest path Minimum-cost Spanning Tree (SSS-MST). In general, a broadcast over an SSS-MST may not outperform due to sequential effects arisen from poor usage of bandwidth per link and nodes having receiving broadcasted messages under the one-port model. In our new algorithm, we obtain the optimal concurrency in each step of broadcasts such that as many messages as possible are forwarded simultaneously in each step of SSS-MST with the optimal number of steps and minimum cost. The two heuristics, from-up-to-down and from-down-to-up algorithms, are proposed to obtain this maximum concurrency using the static topological information and costs for links. Additionally, for regular programs a local update technique is applied to adapt to the dynamic changes in topology and available bandwidth per link over the underlying interconnects.

Keywords: broadcast, MST, concurrency, cluster of SMPs, Myrinet.
I. Introduction

Broadcast is a critical collective communication for the performances of users’ programs and the underlying high-performance platforms. Clusters of Symmetric Multiprocessors (CSMP) are becoming an increasingly popular high-performance computing platforms due to the commodity availability of multiprocessor nodes, mature SMP operating systems, low-latency and high-bandwidth interconnects, and superior price-performance ratio [SBIS98][J98]. Compared to point-to-point communication (e.g., MPI Send and Recv) and accesses to shared memory (e.g., write or read), that we call single communications, a broadcast operation always suffers from a long execution time. In the case of CSMPs, it is more difficult to obtain efficient broadcast algorithms compared to traditional Massively Parallel Processors (MPPs) and SMPs due to a relaxed coupling of commodity processors and remote memory by commodity interconnects in a good price-performance ratio. The performance gap between broadcast and single communications may deteriorate over CSMPs because a broadcasted message may traverse multiple levels (i.e., SMP, Switch, Switch interconnect) with different performance characters.

In this paper, we focus on one of broadcast patterns, called one-to-all broadcast such that a node denoted by the root broadcasts a message to all the other nodes in a broadcast group. A broadcast group may refer to all the nodes distributed to one user program similar to the concept, MPI_COMM_WORLD [GLDS96]. SMPs in a CSMP are connected by a commodity interconnect such as Myrinet [BC95]. Myrinet is switch-based such that multiple SMP nodes are connected to one Myrinet switch and then all Myrinet switches are connected by an interconnect. A message is transferred along a path of Myrinet switches from a source to a destination. An example of such a Myrinet switch-based Linux Supercluster is the Roadrunner in AHPCC [BMMMK99]. When a user runs a program on CSMPs, he or she should be distributed a working subset of processors. The two Ids can locate each working processor, one referred to as the SMP it is connected to, one as the switch this SMP is linked to. Naturally, a broadcast can be done over a CSMP, first in the root switch the root is located at, second among Switch, and last again in each switch except the root switch. Thereby, we can build a three-level hierarchy
of a broadcast over CSMPs, one for a broadcast on a single SMP node, one for a broadcast in switches called \textit{intra-switch}, and one for a broadcast among switches called \textit{inter-switch}. The similar hierarchy for a switch-based cluster is applied to all-to-all broadcast in [JSP99]. The hierarchical broadcast technique is used in [MFET98] to organize heterogeneous machines as a hierarchy of processor-and-memory.

In this paper, we don’t consider how a commodity SMP node implements a broadcast. Instead, we assume each commodity SMP node provides a default broadcast algorithm among its processors, that we can use. Regarding the high-performance switches like Myrinet switches [BC95], our focus isn’t on intra-switch broadcasts due to high-speed hardware crossbar implementations which make doing an intra-switch broadcast trivial. Our concern is concentrated on inter-switch broadcasts among SMPs in CSMPs. On a CSMP, we randomly select a working SMP node as a \textit{leader} for a SMP node. Obviously, the root SMP node becomes naturally a leader forever. With each working switch represented by its leader, we can deduce a weighted directed switch graph (SG) according to the topology of the underlying interconnect and bandwidth per link in a CSMP. In the past literature, the technique of using trees are widely applied to designs of various broadcast algorithms on many different interconnects, e.g., star graphs in [TS97], hypercube in [JH89], and 2D-mesh in [BMT92], wormhole-routed networks [MXEN94], Myrinet [BPDS00], channel networks [BFHI97], and arbitrary topology [RM99].

Distinguished by the existing tree-based broadcasting algorithms, our new broadcast algorithm satisfies a couple of objectives:

1) the maximum number of nodes sending at any step denoted by maximum step concurrency,
2) the optimal steps of message passing among switches on the SG,
3) the minimum cost for a broadcasting to get high broadcasting throughput,
4) the efficient usage of bandwidth per link and topological configuration, and
5) dynamical update to adjust to the changes in the underlying interconnects.
To implement these design objectives, three techniques are applied to accelerate broadcasting on CSMPs based on a Minimum-cost Spanning Tree (MST). A Single-source MST (SSS-MST) is applied to serve as a backbone for broadcasting due to its minimum cost for broadcasting under the all-port model. The practical CSMPs always only support the one-port model. Thus, SSS-MSTs provide no support to step concurrency in inter-switch broadcasting under the one-port model. Indeed, under the all-port model SSS-MSTs may not guarantee the maximum step concurrency. Notice that a broadcast using SSS-MST does not suffer congestion in a way that one link can not be shared simultaneously by more than one message transfer. This limitation may waste bandwidth on modern high-speed interconnects such as Myrinet.

In our techniques, we allow for sharing links with respect to their bandwidths. We use the static configuration information including topologies and bandwidth per link to constitute SSS-MSTs following by two major techniques to optimize the step concurrency in the two directions, from up to down and from down to up. In regular program, identical broadcasts may iterate in sequences of supersteps. We design a locally-updated technique such that each member (a SMP node involved in a broadcast) may update a subtree of the SMST where itself serves as a root by using the real-time information it collected in the prior broadcast. The update frequency may be the one of broadcasts.

The paper is organized as follows. Section II describes a model for clusters of SMPs. Our new broadcast algorithm is proposed in Section III. In Section IV, the correctness and performance analyses of this new algorithm are presented. The following Section V gives the experimental results of the new algorithm. In Section VI, we introduce the related work on broadcast. The conclusions are given in Section VII.

II. Model for Clusters of SMPs

The new broadcast algorithm is based on the SIMPLE model [BJ99] developed by Bader and JáJá. SIMPLE is a methodology and programming environment for
developing high-performance programs for SMP clusters and makes efficient use of the hybrid shared-memory and message-passing environment. Next, we briefly introduce the SIMPLE model.

Programming methodologies for CSMP typically fall into two categories. The first, distributed shared memory (DSM) systems such as TreadMarks from Rice University [AC96], provides a software layer which simulates coherent shared memory between nodes by internally using messaging to move around specific data or referenced memory pages. The second, based on message-passing primitives such as MPI [GL96], enforces a shared-nothing paradigm between tasks, and all communication and coordination between tasks are performed through the exchange of explicit messages, even between tasks on a node with physically shared memory. However, the SIMPLE model differs from both of these approaches, in that a hybrid methodology is advocated which maps directly to the underlying architectural aspects. As such, the shared memory programming on shared-memory nodes is combined with message-passing communication between these nodes. Figure 2.1 illustrated the SIMPLE model.

The underlying CSMP architecture consists of a collection of SMP nodes interconnected by a network that can be modeled as a complete graph on which communication is subject to the restrictions imposed by the latency and the bandwidth properties of the network. The architecture of one SMP is similar to the definition in [BJ99]. Each SMP node contains a number of identical processors, each typically with its own on-chip cache and a larger off-chip cache, which have uniform access to a shared memory and other resources such as the network interface. Similar to the BSP model [V90], we view a parallel algorithm as a sequence of local computation interleaved with communication steps (see Figure 2.1). The SIMPLE model allows the incorporation of heterogeneous components, for instance, SMP nodes with different processor counts or speeds or with different memory size.
Figure 2.1 On the left, we show a message-passing algorithm where each task uses a sequential algorithm during computation phases. On the right, the SIMPLE approach replaces each computation step with an optimal SMP algorithm, and each communication step with the hybrid SMP cluster equivalent. Notice that the SIMPLE methodology allows both irregular and regular algorithms.

In this paper, we consider a switch-based clusters of SMPs like the Linux Supercluster AHPCC Roadrunner [BMMMK99]. The CSMPs are under the one-port model such that at a time each node can send and receive at most one message simultaneously. In this paper we consider one-to-all broadcast such that a processor broadcasts a common message to all the processors in a broadcast group. This processor is denoted by the root processor. All the processors in a broadcast group are working members. Each processor should belong to one SMP. Each SMP is connected to one switch. A working SMP refers to a one that has at least one processor in a broadcast group. Similarly, a working switch means a one that has at least one working SMP. In
each working switch, we select an arbitrary working member as a leader for each working switch. Given a switch-based CSMP, thereby we can deduce a graph $SG = (V, E)$, where $V$ is a set of switches on the CSMP, and $E$ is a set of links between any pair of switches. In switch-based CSMPs, each link may consist of a couple of channels each of that may simultaneously and independently transfer different messages (i.e., Myrinet). Using the number of channels per link, we can calculate the weight for each link. Then, we can further deduce a weighted and directed graph $SG_w = (V, E)$, with weight function $w: E \rightarrow R$ mapping links to real-valued weights. In this paper, without loss of generality, we take the reciprocal of the number of channels over one link as the weight of the link. The weight of path $p = \{v_0, v_1, \ldots, v_k\}$ is the sum of the weights of its constituent links: $w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$. We define the shortest-path weight from $u$ to $v$ by $\delta(u, v) = \begin{cases} \min\{w(p) : u \rightarrow^p v\}, & \text{if there is path from } u \text{ to } v, \\ \infty, & \text{otherwise}. \end{cases}$ A shortest path from vertex $u$ to vertex $v$ is then defined as any path $p$ with weight $w(p) = \delta(u, v)$. According to the weights of links, all links incident on each node are divided to two groups, one for costive links with relative higher weights, and one for cheap links with relative lower weights in the graph $SG_w$.

![Figure 2.2](image-url) A hierarchy for a broadcast in a switch-based CSMP.
Based on the SIMPLE model in Figure 2.1, we constitute a hierarchy for broadcasting on CSMPs in Figure 2.2. This hierarchy shows a natural and simple way for broadcasting on switch-based CSMPs. Consider a broadcasting from a processor as a root. Using this hierarchy, the procedure for this broadcasting can be divided into three rounds. In the first round, the root processor broadcasts the message to all the members at the same SMP through the hardware-implemented crossbar or bus, which occurs on the level 0. In the second round, the members holding the broadcasted message at this root SMP may broadcast the message to all the other SMPs at the same switch, which happens on the level 1. This round corresponds to the intra-switch broadcast. Then, in the third round, the message is transferred from the switches holding it to every other working switch without this message, which is done on the level 2 and corresponds to the inter-switch broadcast. Notice that the hierarchy is reversible and embedded because the final destination is working processors. Obviously, this hierarchy is also extendible to the arbitrary number of levels.

III. Broadcast on CSMPs

3.1 Hierarchical Broadcast Algorithm for CSMPs

In this section, we present a switch-ordered general broadcast algorithm according to the hierarchy for broadcasting in Figure 2.2. Our 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 either switch, only one local processor sends to a processor on the other switch and only one remote processor receives from a local processor. As described in Section 2, a hierarchical broadcast covers this idea by using the concept leader. On each level, each working SMP (working switch) should select a leader processor (leader SMP) in the hierarchical algorithm. Only these leader processors (leader SMPs) can communicate with other SMPs (switches). Meanwhile, the hierarchical algorithm doesn’t put any limits on the underlying topology for inter-switch broadcast instead of the switch-ordered ring in [MSP99]. The topologies for inter-switch broadcast may be arbitrary (e.g., ring and tree). Also, the link scheduling technique is
similar to the U-mesh algorithm in [MXEN94] to permit every processor to remain busy with useful work of the broadcast by scheduling the link usage to avoid link contention. As we show in Section 2, no link contention may be a too strict limit on the high-bandwidth switch-based CSMPs. Thus, without loss of generality, our hierarchical broadcast removes this limit of no link contention, and doesn’t guarantee the busyness of each working processor. Therefore, the hierarchical broadcast may become a backbone for design of any broadcast algorithm by admitting to add various optimization techniques into it.

The hierarchical algorithm is described in Figure 3.1. The three levels of broadcasting in Figure 2.2 are interweaved or embedded in the hierarchical algorithm. When the working components (e.g., working SMP or working switch) on the level 1 or level 2 receive the broadcasted messages, these components will trigger the prior level to broadcast the received messages in the component if the component doesn’t hold these messages prior. Notice that the hierarchical algorithm doesn’t specify how to do the intra-switch and inter-switch broadcatings. These two broadcatings are open to leave enough design space to any improvement. Obviously, the U-mesh algorithm [MXEN94], and the switch-order and link scheduling techniques in [MSP99], and NIC-assisted multidestination messages in [BPDS00] can be easily applied to the intra- and inter-switch broadcatings. For example, the U-mesh algorithm can be directly applied to the intra- and inter-switch broadcasting. Meanwhile, the optimization with the intra- and inter-switch broadcasting is open to future development. The hierarchical algorithm uses the switch priority in a way that when a working processor receives a message, it first broadcasts the message in its working SMP and then this working SMP does a broadcast in its working switch. At last, the working switch will forward the message to other working switches. The advantages of this priority are 1) efficient usage of high-speed switches compared to the relative slow communication among switches 2) efficient increase in step concurrency. Kesavan et al presented a switch-based hierarchical ordering (SHO) and chain concatenation ordering (CCO) algorithm in [KBP97]. Our hierarchical algorithm is different from the SHO in we don’t partition all the members in a broadcast group into disjoint subsets each of them is represented by a leader. In the
hierarchy in Figure 2.2, each switch constitutes a subset that is represented by a leader. In fact, the CCO orders the switches by finding longest partial order chains (POCs). Therefore, the SHO and CCO can be directly applied to our hierarchical algorithm at the same time or respectively.

For a switch-based cluster of SMP, and source processor p with a broadcasted message m:

**Step 1:** The working processor with the message m does a broadcast within the SMP it located at; /* at level 0 */

**Step 2:** *(2.1)* Each working SMP with the message m does a broadcast at the switch it located at. During this process, only the leader processor can serve as a root and can communicate with other SMPs and switches; /* at level 1, the intra-switch broadcast is done. */

*(2.2)* One working SMP does the **Step 1** if it has the working processor without the message m; /* at level 1, the intra-switch broadcast is done. */

**Step 3:** *(3.1)* Each working switch with the message m transfers the message m to some working switch without the message m over the interconnect of switches on the CSMP until all working switches hold the message m. During this process, only the leader processor can serve as a root and can communicate with other SMPs and switches. /* at level 2, the inter-switch broadcast is done. */

*(3.2)* If one working switch without the message m receives the message m, it does the **Step 2**.

**Step 4:** Exit if all working processors receive the message m.

![Figure 3.1 A hierarchical broadcast for switch-based CSMP.](image)

In the broadcast hierarchy, the two levels, level 0 and level 1 can be implemented in a trivial way due to the high-performance hardware-based SMP and switches. In the hierarchical algorithm, the broadcast in working SMP and intra-switch broadcast are trivial. For the level 0, we can directly use the commands provided by SMPs. We can use
any existing techniques to implement the optimal intra-switch broadcast (e.g., U-mesh
algorithm) and may not produce difference in performance of the intra-switch broadcast.
Different from the U-mesh algorithm, we consider there are no partial relationships
between nodes in one switch. The simple intra-switch broadcast with the optimal steps
under the one port-model is shown as follows. This algorithm guarantees that each node
holding the broadcasted message at step \( i \) should be busy forwarding this message at step
\( k \), \( k > i \), until all nodes involved in the intra-switch broadcast receive this message.

*The Algorithm for Intra-Switch Broadcast:*

*For a source \( a_0 \) and a broadcast set \( A = \{a_0, a_1, ..., a_{n-1}\} \) in one switch:*

*Step 1: If Mynode == \( a_0 \) then send the message to \( 2^k \), \( k \geq 0 \) until \( 2^k \geq n \);*

*Step 2: Else receive the message from Mynode \(-\lceil \log_2(Mynode) \rceil \) and

Then send the message to Mynode + \( 2^{\lceil \log_2(Mynode) \rceil} \), where \( K \geq 1 \) and

\[ (Mynode + 2^{\lceil \log_2(Mynode) \rceil}) < n . \]

3.2 Using Single-Source Shortest-path MST for Inter-Switch Broadcasting

One of design objectives for optimal broadcast is a high broadcast throughput.
This design objective implies that a inter-switch broadcast should have as little negative
impact as possible on the underlying interconnect for switches. This point is especially
more important to the inter-switch broadcast than to the broadcasts on the level 0 and
level 1 due to the available bandwidth on the level 2 is always less than those available
bandwidths on the levels 0 and 1. For example, on a Myrinet switch-based cluster, when
a message is transferred from one switch to another switch, the latency for it is the sum of
two crossbar times for traversing a crossbar and one transmission along the link between
the two switches. Instead, the latency for transmissions between different two SMPs
connected to the same switch is just one crossbar time. Crossbar times are completely
dependent on hardware implements of switches. However, the transmission times of
messages among switches are also dependent on switch routing. Good switch routing
may minimize the number of switches traversed by a message while taking as many cheap links as possible to reduce possible congestion and latencies.

In our hierarchical algorithm, single-source shortest path minimum-cost spanning trees (SSS-MST) is applied to order the switches in the inter-switch broadcasting on switch-based CSMPs, due to its satisfaction of the above design objective. The SSS-MST problem is defined in the inter-switch broadcasting as follows.

Given a \( SG_w = (V, E) \), we intend to find a shortest path from a given source vertex \( s \in V \) to every vertex \( v \in V \), and deduce a SSS-MST \( T_w \).

Building an \( SG_w \) for a switch-based CSMP is introduced in Section 2. In this paper, we apply the well-known Dijkstra’s algorithm in [D59] to produce a \( T_w \) for the \( SG_w \). This algorithm can be simply described as follows:

\[
\text{Dijkstra}(SG_w, w, s) \quad /* s \text{ is the source; } w \text{ is the weights of links; } Adj[u] \text{ is the neighbors of the vertex } u. */
\]

1. \( d(s) \leftarrow 0, T \leftarrow V[SG_w] \);
2. \( \text{for } v \in V \setminus \{s\} \text{ do } d(v) \leftarrow \infty \text{ od;} \)
3. \( \text{While } T \neq \phi \text{ do} \)
4. \( \text{Find some } u \in T \text{ such that } d(u) \text{ is minimal;} \)
5. \( T \leftarrow T \setminus \{u\} \)
6. \( \text{for } v \in T \cap Adj[u] \text{ do } d(v) \leftarrow \min(d(v), d(u) + w_{uv}) \text{ od} \)
7. \( \text{od.} \)

where \( d[v] \) is an upper bound on the weight of a shortest path from source \( s \) to \( v \).

The complexity of the Dijkstra’s algorithm is \( O(|V[SG_w]|^2) \).

Under the all-port model, the inter-switch broadcast using the SSS-MST produces the minimum execution time. In fact, this minimum execution time is the lower bound for inter-switch broadcasts in either all-port or one-port model. In fact, under the one-port model, the performance of the SSS-MST broadcast may be very poor. In the next
subsection, we will propose some techniques to improve the one-port SSS-MST algorithm so that we can approach to the lower bound on the execution time of inter-switch broadcasts. Notice that the minimum steps of message passing in the inter-switch broadcast is one of our design objectives. The SSS-MST-based inter-switch broadcast cannot guarantee simultaneously this design objective. The Umesh algorithm in [MXEN94] produces a lower bound on the number of steps for message passing in the inter-switch broadcast. The Umesh, however, approaches hardly to the lower bound on the minimum time given by the SSS-MST broadcast algorithm because the Umesh cannot consider the path weights, and the link contention in one step under the one-port or all-port model. In this paper, we assume the execution time of the Umesh is the upper bound on the execution time for inter-switch broadcasts. We intend to put our inter-switch algorithm on some point between the lower and upper bound on the execution time of inter-switch broadcasts and simultaneously approach to these two design objectives.

**Figure 3.2** An Example of $SG_w$ in (a), its SSS-MST under the all-port model in (b), and a possible send order with least-weight priority under the one-port model in (c). This example shows that SSS-MST cannot guarantee the minimum number of message passing steps. In fact, the source $S_0$ sends messages in a sequential order to the other three switches under the one-port model. Call the node $S_0$ as a sequential-effect node and the other nodes as the affected nodes.
**Proposition 1.** For a $SG_w$, and source switch $s$, given the all-port model, the inter-switch broadcast based on the SSS-MST on the $SG_w$ has the minimum execution time for any interconnect of switches.

**Proof:** This is trivial just according to the Dijkstra’s algorithm and the definition of all-port model because each node can receive and send any arbitrary number of messages at a time. ■

### 3.3 Heuristics for Optimal Inter-switch Broadcasting

SSS-MST based inter-switch broadcast may suffer from *sequential effects* under the one-port model. These sequential effects destroy the step concurrency on an SSS-MST under the all-port model. For example, in Figure 3.2, the node $S_0$ has the connectivity of three under the all-port model, which makes the node $S_0$ send a common message simultaneously to the nodes $S_1, S_2$, and $S_3$ in an only single step. Under the one-port model, unfortunately, the node $S_0$ has to *sequentialize* the sends to these three nodes and the three steps are required to complete the overall inter-switch broadcast. Compared to the broadcast under the all-port model, the performance of the broadcast under the one-port model decreases twice. This case of sequential effects arises from a node with multiple children under the one-port model. We can further generalize this case of the sequential effect to get a general definition of sequential effects. Consider the Figure 3.2.c. In fact, under the one-port model, the three children, $S_1, S_2$, and $S_3$ of the node $S_0$ form a *partial order chain* in the temporal axis although they have disjoint paths to the node $S_0$.

**Definition 1.** *Sequential effect* in a subset of the vertex set of an arbitrary graph exists if and only if the nodes in the subset form a partial order chain in the temporal axis. In an SSS-MST, it has two cases shown in Figures 3.2 and 3.3. The first node in the partial order chain is called the sequential effect node while the other nodes are called the affected nodes.
For an SSS-MST for the inter-switch broadcast, there are two cases to possibly produce sequential effects under the one-port model. The first case is described in Figure 3.2 such that a node has at least two children on the SSS-MST. The second case is described in Figure 3.3.a such that at least three nodes are on a line while the intermediate nodes should have an only single child. The two cases of sequential effects make the SSS-MST inter-switch broadcast’s performance significantly poor under the one-port model. It can be concluded that the step concurrency of the SSS-MST based inter-switch broadcasting may approaches to that of the Umesh under the one-port model if and only if we can remove all the sequential effects from the SSS-MSTs. A way to do so is shown in Figure 3.3.b. The basic idea is that at step $i$ we find a predecessor of the sequential effect node with the broadcasted messages for an affected node in a sequential effect. This predecessor should not be affected by any sequential effects at step $i$ and be different from the direct parent of the node affected by the sequential effect. For example, in Figure 3.3.b, we find a new parent $A$ for the affected node $C$ in the sequential effect. Thereby, the nodes $A$ and $B$ send the broadcasted messages to the nodes $C$ and $D$ in parallel in a step. The key to this idea is to find appropriate candidates of parents for nodes affected by sequential effects so as to increase the step concurrency in each step. The advantage of this technique is that we can make the SSS-MST based inter-switch broadcast is close to the Umesh algorithm to any specified extent regarding the tradeoff between link contention and bandwidth per link.

This technique is especially suitable for Myrinet or optically switch-based clusters of SMPs. The link between any neighborhood two switches consists of a couple of channels. Each channel can transmit any messages independent on the other channels in the link. In the case that multiple working processors are connected to one switch, in fact, 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.
Figure 3.3 An example of the second case for the sequential effect on a SSS-MST for the inter-switch broadcasting. The nodes B, C, and D is on a line while C has a single child D. Thus, The nodes B, C, and D form a partial order chain in the temporal axis shown in Figure 3.3.c under the one-port model. Call the node B as a sequential effect node and the nodes C and D the affected nodes.

Figure 3.4 An example of the down-to-up algorithm to optimize the SSS-MST for the inter-switch broadcasting under the one-port model. Figure 3.4.a is an SSS-MST. Figure 3.4.b is a broadcast scheme by the least-weight priority while the Figure 3.4.c is the broadcast tree by the down-to-up algorithm. Compared to the two schemes, in this case, the number of steps is decreased by 25 percent. The level of the children for the node B is two because it has the two children located respectively at the levels 3 and 4 if we assume that the node A is located at the level 1.
**Down-to-up algorithm.** The down-to-up algorithm is to calculate the longest path $p$ from any one node to its children including the direct and indirect in the $SG_w$. Obviously, this path $p$ implies that any child of this node can receive the message broadcasted by this node in the weight $w(p)$ of this path. $w(p)$, covers link weights and processor start-ups. Our concern is to find this maximum $w(p)$ from any node $v$ to all its children, which is denoted by. According to the $w(p)_v$ of each node $v$, we can decide a send order for nodes affected by a sequential effect instead of the least-weight priority. The least-weight priority in the example of Figure 3.2 can optimize the local send order for the nodes incident only on the node $u$, for example, not regarding the weights of the paths to its grandchildren from the children of the node $u$. However, our 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 one can be divided into the two phases, one for the start-up, and another for the transmission. In the start-up phase, the working processor should finish a series of operations such as calculating the routing for the message, calculating the cyclic-redundancy-check (CRC) for detecting corrupted packets, assembling the routing information, CRC, the data, and other necessary bits by the transmission protocol together into a data frame, transferring this data frame into the DMA on the working processor and further from the DMA to the router. In fact, when a user transfers a message, some high-level communication protocol is applied as an interface between the user’s high-level language program such as C/C++ and the underlying interconnects such as Myrinet. Such an interface may be MPI or OpenMP. Obviously, this intermediate interface may attach the extra latency for the transferring such as copies of the data from the user space into the data space. It is often that these copies may require some help of the kernel, which also contributes the additional latency to the transferring. However, the transmission is very simple to route the data frame built in the start-up between switches according to the calculated routing information in the start-up by fully hardware-based implementations such as high-performance Myrinet switch. Therefore, in most cases, the latency by the start-up is larger or even far larger than the one by the transmission. The
down-to-up algorithm can be easily extended to include the start-up factor in the way that before a node sends its $\Sigma$ to its parent, we add its start-up to the $\Sigma$. According to the $\Sigma$ with the start-up, the down-to-up algorithm can guarantee that the node with the longest level of the children on the $SG_w$ always should have the highest send priority. This strategy can overlap the transferring times for the children of a node so as 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
the $w(p)_i = 0$.
Step 1: while ($l \neq 1$) Then {
Step 2: Each node $u$ in the level $l$ sends the sum $\Sigma_u$ of its $w(p)_u$, and the
weight of the link to its parent to its parent. Notice that the $\Sigma_u$
should include the start-up of the node $u$ if the $u$ isn’t a leaf;
Step 3: Each node $v$ in the level $l - 1$ selects the maximum $\Sigma_u$ 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. } 
Step 6: Do the Depth-First-Search (DFS) algorithm from the root and give each
node a step number representing that the node should send a message at

Figure 3.5 The down-to-up algorithm under the on-port model for the inter-switch
broadcasting. Notice that a node may have multiple step numbers because it is permitted
to send messages at different steps.

Up-to-Down algorithm: This algorithm is done over the SSS-MST for an inter-
switch broadcast. There are two rounds for the up-to-down algorithm. The first round
executes the Breadth-First-Search (BFS) [Moore59] over the SSS-MST beginning with
the root of the SSS-MST so as to find all the nodes with at least two children. Obviously,
under the one-port model, those nodes will suffer from sequential effects. During the BFS, all nodes searched are sorted according to their upper bounds on the weights of shortest paths from the source node, which form a set called $S_H$. This operation can be trivially 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 for if $S_H \neq \emptyset$ such that each of children will have different direct predecessor. This procedure is called the parallelized procedure. In the second round, the well-known Depth-First-Search algorithm [Jun99] 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 parallelized procedure is applied to break off the sequential effect.

![Diagram](image)

**Figure 3.6** An example for application of up-to-down algorithm to optimize the SSS-MST. The original inter-switch $SG_w$ is displayed in Figure 3.2.a. Figure 3.6.a shows an original SSS-MST; Figure 3.6.b displays the SSS-MST broadcast under the one-port model; Figure 3.6.c shows the optimized SSS-MST.

The gap between the start-up and the transmission may affect significantly finding a predecessor for a node in the parallelized procedure for removing sequential effects. Thus, when making a choice of the predecessor for a node in the parallelized procedure, we should consider the two factors, the start-up and the bandwidth between switches. The
latter has been given in the form of the weights on the graph $SG_w$. The former has been calculated in the down-to-up algorithm. Notice that the $V(SG_w)$ may not cover all the switches on a cluster of SMP. According to our assumption, the $V(SG_w)$ only cover all the working switches on the cluster of SMP. The shortest path between two nodes $u$ and $v$ is calculated on the $SG_w$ not on the topology of all the switches on the cluster. The weight for a link between the nodes $u$ and $v$ are calculated by the default routing $R_{cluster}$ on the topology of all the switches on the cluster. For a switch-based cluster, the $R_{cluster}$ may be shortest or others selected by the users. Thereby, choosing predecessors for the parallelized procedure on one node $v$ can be done in the $S_H$ 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 $SG_w$. A window-slipped technique is applied to gradually increase the range for the search. If the node $v$ has the $m$ children, 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 links may occur. Therefore, we also should consider the bandwidths for 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. When we are concerned with the link contention, the cheap link priority can be used.

---

*Figure 3.7* The up-to-down algorithm for removing all the possible sequential effects on the SSS-MST.

---

*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 parallelized procedure to remove them;

*Step 2:* Execute the DFS algorithm to find all the possible sequential effects of the second case parallelized procedure to remove them;
Given a graph $G$ and the $R_{\text{cluster}}$ 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, assume $S_{H} = \emptyset$ and call $S_{H}$ the holding set each element of which holds the message.

Step 1: on a 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 the 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 $m-1$. Use 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; else then

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 $R_{\text{cluster}}$.

Step 4: 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 5: if the 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 6: if the number of valid predecessors is less than \( m - 1 \), the window is left-shifted on the \( S_n \), go to the Step 3 to check each candidates in the new window until all the overall \( S_n \) is searched.

Step 7: Sort the valid predecessors \( k \ (0 \leq k < m) \) 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 the \( j + k \) 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 \( SG_w \) whose weight is the weight of the path from the predecessor to the child crossing the node \( v \).

**Figure 3.8** The algorithm for the parallelized procedure to remove sequential effects.

**Local-update algorithm:** In a cluster of SMPs connected by high-performance switches such as Myrinet switches, regarding its scalability, sometimes some links in the \( SG_w \) may change dynamically the available bandwidth to all users who contend for those links or due to some faults on these links or switches. In this case, the weights of these links may dynamically change so that the SSS-MST may be necessary to change to adapt to those changes. An adaptive communication algorithm for distributed heterogeneous systems are propose in [BVR98] using the directory services to provide information on current network performance by Globus[FF97], based on which an all-to-all broadcast is developed in [BVR98]. In this paper, we do not use any directory services such as Globus. Instead, we just make efficient use of one-to-one communications on our broadcasting algorithm. When a node sends a message to another node according to the broadcast tree built by the down-to-up and up-to-down algorithms, it will measure the completion time of the communication. This measure can be used as the current valid weight for the link between the two nodes. Notice that a link on the broadcast tree may be
a single physical link on the switch topology for the cluster of SMPs, or, a path of some physical links.

An alternative method for these dynamic changes in the configuration of the switch topology is to find a new path for a pair of the nodes if the weight of the link of the pair increases to a threshold on $S_{G_w}$. This method may be intensive-computation due to search for a set of candidates undetermined in advance. In the worst case, the set may be very large. Importantly, the weight-increased link may be a critical one that means no other alternative paths exist between the two nodes. In this case, this method may not work. Another alternative is to rebuild the broadcast tree. This method may face the intensive-computation problem since this update should be completed on-the-fly. Regarding the up-to-down and down-to-up algorithms, we present a simple non-intensive-computation method to handle this dynamic configuration. Our method is locally updated. That is, when a node finds the increase in the weight of a link to its child, it will do the two things. First, the node checks if the change is valuable to respond to. The rule is simple that if the change in the weight affects the send order of some sequential effect whose the sequential-effect node is the first such predecessor of the two nodes. In fact, what should be done is to re-calculate the maximum $w(p)$ for the sequential-effect node similar to the down-to-up algorithm. Second, if it is valuable, the node will select one of its predecessors as the new direct parent of the child so that the time saved by starting in advance the communication to the child can tradeoff as much the increase in the weight as possible. The complexity for this method is at most $O(V)$ plus one one-to-one communication, where $V$ is the number of switches involved in the inter-switch broadcast.

IV. Correctness and Performance Analysis

Now we address the correctness of the hierarchical broadcast algorithm. Indeed, we are concerned with the correctness of the inter-switch broadcast algorithm. The proof of the intra-switch broadcast algorithm is trivial. For the inter-switch broadcast, our concern is whether the broadcast tree remains shortest-path after removing sequential effects. The following Proposition 2 comes directly from the Corollary 25.11[CLR90].
**Proposition 2.** If the Dijkstra’s algorithm runs on a weighted, directed graph \( G = (V, E) \) with nonnegative weight function \( w \) and source \( s \), then at termination, the predecessor subgraph \( G_x \) is a shortest-paths tree rooted at \( s \).

Figure 4.1 The temporal abstractions of sequential effects and the parallel procedure to remove sequential effects. Figure 4.1.a describes a sequential effect consisting of three nodes. Figure 4.1.b shows how to remove the sequential effect by choice of one predecessor \( A^p \) of \( A \). The broadcast root is denoted by \( s \).

**Proposition 3.** Given a weighted, directed graph \( G = (V, E) \) with nonnegative weight function \( w \), source \( s \), and the shortest-paths tree \( G_x \) rooted at \( s \), the \( G_x \) remains a shortest-paths tree rooted at \( s \) after removing sequential effects by the parallel procedure shown in Figure 3.8 if the routing \( R_{router} \) over \( G \) is shortest-path.

**Proof:** Consider the three nodes \( A \), \( B \) and \( C \). Assume that the three nodes form a sequentially temporal order, \( A \rightarrow B \rightarrow C \) shown in Figure 4.1.a. Obviously, this temporal abstraction covers the two cases of sequential effects given in Figures 3.2 and 3.3. Without loss of generality, removing this sequential effect is to choose a predecessor \( A^p \) of \( A \), \( A^p \in V(G_x(A)) \) shown in Figure 4.1.b where \( G_x(A) \) is a shortest-paths subtree of \( G_x \) with \( A \) serving as a leaf. We conclude that \( \delta(s, A^p) \leq \delta(s, A) \). Since the
$R_{\text{router}}$ gives the shortest path between $A^p$ and $C$, $\delta(s, A^p) + \delta(A^p, C) \leq \delta(s, A) + \delta(A, C)$ . Thus, the new path from $s$ to $C$ remains shortest. Additionally, removing sequential effects does not affect weights of the paths from $s$ to all descendants of nodes $B$ and $C$. For the node $A^p$, since it is idle at step $i$ shown in Figure 4.1, the weights of the paths from $s$ to all its descendants remain unchanged. Therefore, removing sequential effects remains the shortest-paths feature of $G_x$. 

It is important to maintain the messages involved in a broadcast operation contention-free along their paths. The contention-free limit, however, may keep broadcast operations from efficient usage of the high bandwidth of the high-speed switch-based interconnects. For example, in Myrinet, a link always consists of multiple channels each of which can simultaneously server different transmissions of messages involved in a broadcast operation. Further, the Wavelength Division Multiplexing (WDM) [CKW90] for optimal networks can partition the optimal bandwidth of a single fiber into a number of channels and allow the transmission of multiple data streams concurrently along the same fiber on the different channels, i.e., different wavelengths. Thus, relaxing the strict contention-free limit is practical in switch-based interconnects. In this paper, we propose a new relaxed contention-free limit, link-controlled contention-free such that the number of messages sharing the same link is not greater than the number of the channels of the link. Based on this link-controlled contention-free concept, we develop the up-to-down algorithm to parallelize the sequential effects on a single-source MST. Notice that we don’t consider the contentions of accesses to shared memory in each of SMPs and the contentions in intra-switch broadcasts, but consider the contentions only in inter-switch broadcasts.

**Proposition 4.** The hierarchical broadcast algorithm is stepwise link-controlled contention-free.

**Proof:** Consider an inter-switch broadcast. A single-source MST is stepwise contention-free under the all-port model. However, the one-port model may break this limitation when we use the up-to-down algorithm to remove sequential effects in the two cases (see Figure 3.2 and 3.3) due to choosing predecessors for the nodes in a sequential effect.
They are the only sources to break the link-controlled contention free. However, a predecessor is chose only if the path from this predecessor to one node in the sequential effect should satisfy the number of simultaneous transmissions of messages over any link of the path should be equal to or less than the number of available channels of the link at one step of a broadcast operation. Thus, at any step, the link-controlled contention-free is guaranteed.

Important, the switch-based interconnect of a cluster of SMPs may be shared by multiple users running their programs. As shown in the proof of the Proposition 4, the hierarchical broadcast algorithm considers the availability of bandwidth per link. Indeed, when multiple programs simultaneously run on a cluster of SMPs, the scheduler of the cluster always make those programs fairly share the single switch-based interconnect. Therefore, all the channels of links may not be distributed to a single program at a time regarding to the “quality of services” (QoS) requirements of other programs. In the hierarchical broadcast, choosing predecessors is based on the current available bandwidth resource. Meanwhile, we provide the static and dynamic methods to adopt the broadcast to the availability of bandwidth of links. This feature is critical to QoS application such as real-time image and video applications.

**Proposition 5.** The hierarchical broadcast is stepwise link-controlled optimal in the sense that as the maximum number of nodes with broadcasted messages are busy to forward these messages at each step under the link-controlled contention free limitation with the minimum cost of the inter-switch broadcast.

**Proof:** Consider the intra- and inter-switch broadcasts. First, we calculate the step numbers for the two broadcasts. Second, we calculate the costs for the two broadcasts.

1) For the intra-switch broadcast, from construction of the algorithm, each destination node receives the message exactly once. Assume the number of nodes in the intra-switch to be $m$. For $m-1$ destinations, the source node sends at most $\lceil \log_2 m \rceil$ messages sequentially. A destination node receiving the message that was sent in the $i$th step will send at most $\lceil \log_2 m \rceil - i$ messages. Thus, the height of the broadcast tree, that is, the number of steps required for all nodes to receive the
message, is \( \lceil \log_2 m \rceil \) which is minimum for \( m - 1 \) destinations. The cost for the intra-switch is only dependent on the physical architecture of switches.

2) For the inter-switch broadcast, the single-source shortest-path MST gives the lower bound on broadcasts under the all-port model. In this paper, we assume the one-port model. From construction of the up-to-down algorithm, removing sequential effects can shorten the length of the part of the tree covered by the sequential effects, that is the only operation to change the tree. Thus, the final broadcast tree has the height not greater than the one of the original single-source shortest-path MST, that is the number of steps required for all the primary nodes of switches to receive the message. Indeed, the height of the original single-source shortest-path MST is the upper bound on the number of steps for the inter-switch broadcast. At the extreme case of the unlimited number of channels per link, the hierarchical broadcast can remove all the sequential effects using the up-to-down algorithm. Refer to the two cases of sequential effects in Figures 3.2 and 3.3. Thereby, the lower bound on the number of steps for the inter-switch broadcast is \( \lceil \log_2 n \rceil \), where \( n \) is the number of switches involved in the inter-switch broadcast. The original SSS-MST can be changed to be a new broadcast tree whose each node except leafs has at least two children at different steps similar to the intra-switch broadcast algorithm.

Consider the cost for an inter-switch broadcast. From construction of the down-to-up algorithm, when the MST under the all-port model is transformed to the one under the one-port model each node sorts all its children in the descending order of the cumulative weights of its children. The cumulative weight of a node \( i \) refers to the greatest path weight from \( i \) to any of its descendants. Thus, the sequentialized MST by the one-port model remains the minimum cost under the one-port model. Then, the up-to-down algorithm removes the maximum number of sequential effects under the link-controlled contention-free rule. Notice this process does not affect the greatest path weight of the root. Thus, the inter-switch algorithm has the minimum cost. ■
**Proposition 6.** The lower bound on the number of steps for an inter-switch broadcast under the one-port model is \( \lceil \log_2 n \rceil \) if the number of channels per link is unlimited; its upper bound is the height of the single-source shortest-path MST sequentialized by the one-port model, given the \( n \) switches involved in the inter-switch broadcast.

**Proof:** Refer to the proof of Proposition 5.

**Proposition 7.** Regarding the dynamic update algorithm, the complexity of the inter-switch broadcast algorithm is \( O(V^2 + (S/2 + \kappa + 2) \cdot V) \) for an inter-switch graph \( G = (V, E) \), where \( \kappa \) is the average degree in \( SG_w \), and \( s \) the number of sequential effect nodes.

**Proof:** First, we consider the complexity of the Dijkstra’s algorithm for building a single-source shortest-path MST \( SG_w(V, E_w) \) for the topology \( G(V, E) \) of the switches. The running time of the Dijkstra’s algorithm is \( O(V^2) \). The down-to-up algorithm begins with calculation of cumulative weights for each node of \( SG_w \). Each node should compare the weights of all its children. Thus, this calculation runs within \( O(\kappa \cdot V) \). Then, the DFS algorithm is executed to determine the send order of children for each node. In this case, the DFS can run in \( O(V) \). The up-to-down algorithm executes the BFS and DFS algorithms to find sequential effects and to remove them. Regardless removing sequential effects, the runtime for the BFS and DFS is \( O(2 \cdot V) \). Removing a sequential effect node can be done in at most \( O(V) \) if the holding set \( S_H \) for this sequential effect node is equal to \( V \). Of course, this case is the worst. The average size of \( S_H \) should be the half of the size of \( V \). The \( s \) sequential effects can be removed in \( O(s \cdot \kappa) \). Thus, the complexity of the inter-switch broadcast algorithm is the sum of these complexities.

**V. Experimental Results**
VI. Related Work

The technique of using trees in broadcasting has a long and rich history due to the optimal steps for broadcasting. Many researcher developed tree-based broadcast algorithm by using the unique topological characteristics of the underlying interconnects, e.g., 2D-mesh in [BMT92], star graphs in [TS97], hypercube in [JH89]. These algorithms can not be ported to other interconnects due to dependence on the underlying interconnects. Some researchers presented portable tree-based broadcast algorithms such as for wormhole-routed networks [MXEN94], Myrinet [BPDS00], channel networks [BFHI97], arbitrary topology [RM99], all-optical networks [BPD99].

McKinley et al in [MXEN94] implemented multicast communication in wormhole-routed direct interconnects in the absence of hardware multicast support by exploiting the properties of the switching technology on n-dimensional mesh and hypercube with minimum time using deterministic dimension-ordered routing of unicast messages. On the two interconnects by U-cube tree for hypercube and U-mesh tree for mesh, their algorithms deliver a multicast message to $m-1$ destinations in $\lceil \log_2 m \rceil$ message-passing steps while avoiding contention among the constituent unicast messages. Their algorithm can not guarantee to be of minimum time on either other regular topologies such as Torus and star graphs or irregular topologies. Under the all-port model and unlimited bandwidth per link, their algorithm give always the minimum time for message passing in broadcasting over any arbitrary topology. Therefore, their algorithm cannot adapt to intra-switch broadcast on CSMPs connected possibly by arbitrary topologies under the one-port model. However, their algorithm implies the significance of step concurrency to obtaining minimum time for broadcasting. In this paper, we call their algorithm Ubcast.

Jacunski et al in [MSP99] presented an all-to-all broadcast on switch-based clusters of workstations. They considered the switch-order ring and link scheduling techniques to approach to the optimal properties:1)degree of pipelining of communication components are significant, 2)transmission latencies are minimal, and 3)node idleness is minimized. On irregular interconnects, an ordering of nodes for the ring algorithm is
found by performing an in-order traversal of the BFS tree that was generated to determine interconnect routing characteristics. Their algorithm cannot allow for contention for links. In fact, for the inter-switch broadcasting on Myrinet switches, the nodes may be connected by a couple of channels which are combined into one link on the graph SG. This point implies that along a tree, one node may receive messages not from its direction parents but from certain its predecessor by efficient usage of bandwidth per link.

Buntinas et al in [BPDS00] constructed optimal multicast trees by a simple top-down greedy algorithm under the postal model. Using a special enhancement to NIC over Myrinet, the multidestination message transmission was applied to accelerate the procedure of multicasts. Their algorithm implies that it is possible to find a way such that we can implement a software multidestination message technique by using non-direct predecessor transmission by optimize the multicast tree by up-down and down-up greedy algorithms. Raghavan and manimaran in [RM99] proposed a rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees by using the concept quality factor describing the usefulness of a portion of the multicast tree to the overall multicast tree. Mitzenmacher in [M00] analyzed the period of update to information.

VII. Conclusions

Acknowledges

References


