All-to-all broadcast on switch-based clusters of workstations

Matt Jacunski, P. Sadayappan, and D. K. Panda

Department of Computer and Information Science
The Ohio State University, Columbus, OH 43210
Phone: (614) 292-0053, Fax: (614) 292-2911
Email: {jacunski,panda,saday}@cis.ohio-state.edu

Abstract: *This paper presents efficient all-to-all broadcast for arbitrary irregular networks with switch-based wormhole interconnection and unicast message passing. First, all-to-all broadcast is considered within a single switch cluster. Both combining and non-combining algorithms are compared via analytical modeling and simulation. The characteristics of optimal all-to-all broadcast operation are considered and applied to development of multi-switch algorithms. The single switch algorithms are considered on two switch clusters and a near-optimal algorithm is developed which schedules use of interconnecting links where the potential for link contention exists. Finally, the Link Scheduling concept is extended to handle arbitrary irregular networks. Operation of this algorithm is simulated on a 128-node irregular network showing a 27.1% improvement in performance compared to other available algorithms.*

Contents

1 Introduction

The availability of switch-based commodity gigabit networking technologies such as Myrinet [2] and the high performance of personal computers (and workstations) makes parallel computing over clusters a promising alternative to the use of custom designed parallel computers. An important issue to be addressed is the development of efficient implementations of portable parallel programming models such as the Message Passing Interface (MPI) [14] for switch-based clusters of workstations.

MPI defines primitives for point-to-point communication as well as various collective communication operations between processes. While considerable research has addressed the development of efficient algorithms for collective communication operations over regular networks such as meshes and hypercubes [1, 3, 4, 6, 9, 11, 12, 13, 15], much less work has been done for optimizing collective communication over irregular switch-based clusters [10, 16]. In this paper we address the collective communication operation of all-to-all broadcast (called All_Gather in MPI) for switch-based clusters with arbitrary topology.

The paper is organized as follows. In Sec. 2 we describe the communication model assumed in this paper. In Sec. 3, all-to-all broadcast is considered within a single-switch cluster. Both combining and non-combining algorithms are compared via analytical modeling and simulation. Sec. 4 treats the two-switch case and develops a near-optimal algorithm. The general case of multiple switches with arbitrary topology is considered in Sec. 5. Conclusions are provided in Sec. 6.

2 Communication Model

In this section, we provide an irregular switch-based cut-through network model. The associated issues related to deadlock-free, adaptive routing for such a network are discussed.

2.1 Network Model

Figure 1(a) shows a typical parallel system using switch-based interconnect with irregular topology. Such a network consists of a set of switches where each switch can have a set of ports. The system in the figure consists of eight switches with eight ports per switch. Some of the ports in each switch are connected to processors/workstations, some ports are connected to ports of other switches to provide connectivity between the processors, and some ports are left open for future connections. Such connectivity is typically irregular and the only thing that is guaranteed is that the network is connected. Thus, the interconnection topology of the network can be denoted by a graph G = (V,E) where V is the set of switches, and E is the set of bidirectional links between the switches [2, 18]. Figure 1(b) shows the interconnection graph for the irregular network in Fig. 1(a). It is to be noted that all links are bidirectional and multiple links between two switches are possible.

2.2 Switch Model

Figure 2(a) shows the architecture of a generic switch with k ports. Each port consists of one input and one output link. As shown in Fig. 1(a), a port can be connected to the port of another switch, a workstation, or kept open. A switch is wired to the workstation through a network interface card which typically is plugged into the I/O bus of the workstation. The switch can implement different types of

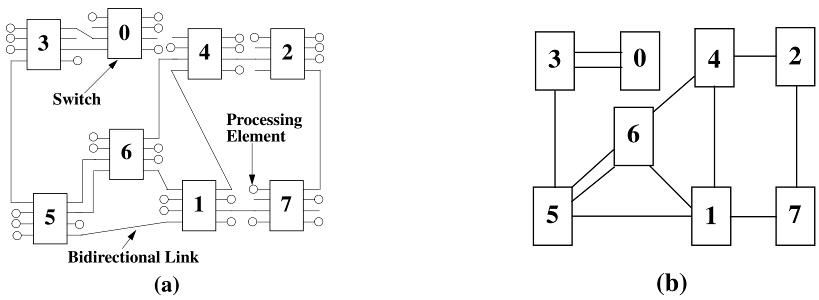

Figure 1: a) An example system with switch-based interconnect and irregular topology; b) corresponding interconnection graph G.

switching techniques: cut-through or store-and-forward. In this paper we assume switches implementing cut-through switching. Each port consists of an input and an output buffer. Although these buffers only need to be big enough to capture the header flit of an incoming worm so that the routing decision can be made as soon as the header flit arrives, deeper buffers are usually required to perform flow control efficiently across long links and to reduce link contention. A k-port switch typically provides a $k \times k$ crossbar connectivity in order to enable a concurrent transfer of messages from the input buffers to any of the output buffers [2, 18, 19, 20, 21]. However, in many instances, some routing restrictions are used to achieve deadlock-free routing. We consider some of these issues in the following subsection.

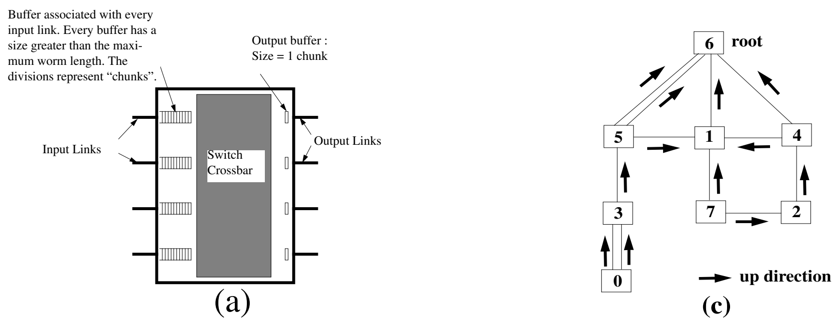

Figure 2: (a) Organization of a typical k-port switch supporting wormhole routing. (b) BFS spanning tree rooted at node 6 corresponding to the example irregular network shown in Fig. 1.

2.3 Routing Issues

Several deadlock-free routing schemes have been proposed in the literature for irregular networks [2, 8, 17, 18]. Without loss of generality, in this paper we assume the routing scheme for our irregular network to be similar to that used in Autonet [18] due to its simplicity and its commercial implementation. Such routing allows adaptivity, and is deadlock-free.

In this routing scheme, a breadth-first spanning tree (BFS) on the graph G is first computed using a distributed algorithm. The algorithm has the property that all nodes will eventually agree on a unique spanning tree. Now, the edges of G can be partitioned into tree edges and *cross edges*. According to the property of BFS trees, a cross edge does not connect two switches which are at a difference of more than one level in the tree. Deadlock-free routing is based on a loop-free assignment of direction to the operational links. In particular, the "up" end of each link is defined as: 1) the end whose switch is closer to the root in the spanning tree; or 2) the end whose switch has the lower UID, if both ends are at switches at the same tree level. Links looped back to the same switch are omitted from the configuration. The result of this assignment is that the directed links do not form loops. Figure 2(b) shows the BFS spanning tree corresponding to the example irregular network shown in Fig. 1. The assignment of the "up" direction to the links on this network is illustrated. The "down" direction is along the reverse direction of the link.

To eliminate deadlocks while still allowing all links to be used, this routing uses the following up/down rule: a legal route must traverse zero or more links in the "up" direction followed by zero or more links in the "down" direction. Putting it in the negative, a packet may never traverse a link along the "up" direction after having traversed one in the "down" direction. Details of this routing scheme can be found in [18]. This routing is also referred to as *up*/down** routing or UD routing.

In order to implement the above routing, each switch has an indexed forwarding table. When a worm reaches a switch, the destination address is captured from the header flit of the incoming worm. This address is concatenated with the incoming port number and the result is used to index the switch's forwarding table. The table lookup returns the outgoing port number that the worm should be routed through. The forwarding tables can be constructed to support both minimal and non-minimal adaptive routing. In this paper, we only consider minimal adaptive routing. Thus, the forwarding tables allow only legal routes with the minimum hop count. When multiple minimal routes exist from the source to the destination, the forwarding table entry shows alternative forwarding ports. The choice of the outgoing port is decided dynamically based on the ports which are free when the header flits arrive at the switch. In the case of multiple outgoing ports being free, the routing scheme randomly selects one of them.

2.4 Parameters used in analysis of algorithms

P	number of nodes participating in the operation
m	length of message data each node begins operation with
S	number of switches in the network
k	number of ports on each switch
	node
L	message transmission time from source to destination (LogP model)
o	overhead incurred by a node when sending or receiving a message (LogP model [5])
g	gap - the minimum time between successive message transmissions (LogP model)
g_s	steady state gap - the minimum time between successive message transmissions while a node must both send and receive messages alternately

3 All-to-all broadcast on a single switch

All-to-all broadcast among nodes connected to a single switch may be performed without link contention by a variety of methods. Three such algorithms are presented below.

3.1 Algorithms

3.1.1 Logical Ring (LR) algorithm

This algorithm organizes all participating nodes in a logical ring by rank order. In the first step of the required $(P-1)$ steps, each node sends its message data to the subsequent node in the ring and receives a message from the preceding node in the ring. (In all remaining steps, the message just received is forwarded to the following node and another message is received from the preceding node.) This algorithm has the characteristic that all communication from any single node's perspective is with only two other nodes.

LR algorithm

```
for i = 1 to (P-1)
    send message MSG[ ((myindex+1)-i)mod(P) ] to (myindex+1)mod(P)
    receive message MSG[ (myindex-i)mod(P) ] from (myindex-1)mod(P)
```

Figure 3: Pseudo-code description of the LR algorithm for P node all-to-all broadcast.

3.1.2 Simultaneous Broadcast (SB) algorithm

The SB algorithm performs P simultaneous broadcasts, one originating from each node. Each broadcast is performed by sending the node's local message data to each of the $(P-1)$ other nodes participating in the all-to-all operation. Each node will receive $(P-1)$ messages.

SB algorithm

```
for i = 1 to (P-1)
    send message MSG[ myindex ] to (myindex+i)mod(P)
    if incoming message present
        receive message MSG[ (myindex-i)mod(P) ] from (myindex-i)mod(P)
        num_received ++
while num_received < (P-1)
    receive message MSG[ (myindex-num_received)mod(P) ] from (myindex-i)mod(P)
    num_received ++
```

Figure 4: Pseudo-code description of the SB algorithm for P node all-to-all broadcast.

Because each node sends only its local data, sends may be performed asynchronously to receiving. Transmission may overlap startup costs for all steps after the first. Each node begins by sending messages consecutively to the destinations $(rank + step)$, checking after each send for the arrival of an incoming message. This continues until the network can accept no more messages or an incoming message is detected. At this point, receives and sends begin alternating until all messages have been sent. The node then continues to receive until all required messages have been received. Figure 4 gives a pseudo-code description of the SB algorithm for P node all-to-all broadcast. Figure 3 gives a pseudo-code description of the LR algorithm for P node all-to-all broadcast.

3.1.3 Combining algorithm

The combining algorithm consists of $\lceil log_2 P \rceil$ steps during which every node sends a message and receives a message. The message to be sent consists of the local message data combined with message data received in previous steps. The size of the message sent and the message received will double at each step (with the possible exception of the last step). The sum of the sizes of all messages sent or received by each node is $(P-1)m$. Beginning at $step = 0$, the address of the destination at each step will be $(rank + 2^{step})$ mod P. The address of the source node at each step will be $(rank - 2^{step})$ mod P. Figure 5 gives a pseudo-code description of the Combining algorithm for P node all-to-all broadcast.

Combining algorithm

> **for** i $= 1$ to $\lceil log_2 P \rceil - 1$
>
> > **send** message containing (2^{i-1}) data blocks to (myindex+2^{i-1})mod(P)
> >
> > **receive** message containing (2^{i-1}) data blocks from (myindex-2^{i-1})mod(P)
>
> **send** message containing $(P - 1 - num\_sent)$ data blocks to (myindex+$2^{\lceil log_2 P \rceil - 1}$)mod(P)
>
> **receive** message containing $(P - 1 - num\_recv)$ data blocks from (myindex-$2^{\lceil log_2 P \rceil - 1}$)mod(P)

Figure 5: Pseudo-code description of the Combining algorithm for P node all-to-all broadcast.

3.2 Analytical comparison of algorithms (on single switch)

There are three components to consider when analyzing the performance of the all-to-all operation on switch based networks: startup costs, transmission costs, and the degree to which communication components are pipelined. The lower bound on the required number of startups for the all-to-all operation is $\lceil log_2 P \rceil$. The lower bound on the transmission time is proportional to the number of messages each processor must receive: $(P-1)L$ where L is the transmission latency for a message with size equal to a single node's data block.

For the algorithms discussed above, the lower bound on transmission time is attained. That is, each node receives only the $(P-1)$ messages it requires. Also, at no point does more than one node send to the same destination node. This communication pattern may be executed without link contention on a single switch. Therefore, the factors which differentiate the performance of these algorithms on a single switch are the number of message startups required and the amount of pipelining of the communication components which is attained.

The SB algorithm and the LR algorithm each require $(P-1)$ startups to complete. The communication pattern for the ring algorithm requires completion of a receive before the next send may begin, eliminating the possibility of any communication component pipelining. The SB algorithm, however, repeatedly sends the same message and may send to multiple destinations without the delay incurred by the ring algorithm due to message forwarding. This means that the transmission cost of a previously started message may overlap with a subsequent message startup. Therefore, the SB algorithm will always complete in less time than the ring algorithm on a single switch.

Implementation of the SB algorithm is such that messages are consecutively sent until the presence

Case 1		Condition when $t_{SB} < t_{combining}$
$L > 2o \Rightarrow$	$g_s = L$ $g = L$	$\lceil log_2 P \rceil > 1$ (always true)
Case 2		
$o < L < 2o \Rightarrow$	$g_s = 2o$ $g = L$	$\frac{2P-4-2\lceil log_2 P \rceil)}{p-3} < \frac{L}{o}$
Case 3		
$L < o \Rightarrow$	$g_s = 2o$ $g = o$	$\frac{2P-3-2\lceil log_2 P \rceil}{p-2} < \frac{L}{o}$

Table 1: Description of conditions where the SB algorithm outperforms the combining algorithm on a single switch.

of an incoming message to be received is detected. Assuming that all nodes begin the operation at the same time, the number of messages, n, sent before the first incoming message is detected depends on the relationship between the latency, L, and the gap, g. For a wormhole routed network consisting of only a single switch, this will be $n = 1$. In general, $n = \left\lceil \frac{L}{g} \right\rceil$.

The time required to complete the SB algorithm on a single switch

$$\begin{aligned} t_{SB} &= 2o + L + (P - 2 - n)g_s + ng \\ &= 2o + L + (P - 3)g_s + g \end{aligned}$$

The values of gap, g, and steady state gap, g_s, depend on the relative values of L and o. These are given in Table 1.

The communication pattern for the combining algorithm requires a reduced number of message startups but no pipelining of communication components is possible. For each message, the startup overhead, transmission time, and receive overhead occur sequentially. The completion time for the combining algorithm is

$$t_{combine} = \lceil log_2 P \rceil 2o + (P - 1)L$$

As the startup cost decreases relative to the transmission cost, the performance differences between these algorithms decreases. The algorithm cost becomes dominated by the transmission cost which is the same for each of these algorithms. For instances of all-to-all broadcast where the startup cost dominates, the combining algorithm clearly outperforms the SB and ring algorithms. However, for cases where the startup cost and transmission latency are comparable (or neither is negligible), Table 1 defines which algorithm requires less time to complete in terms of the number of participating nodes and the ratio of transmission latency to send/receive overhead.

3.3 Simulation experiments and results

Simulation experiments were performed to evaluate and compare performance of the algorithms discussed above for all-to-all broadcast on a single switch. A flit-level wormhole-routed simulator WORMULSim was used to model the network as discussed in Section 2. For these experiments, the following parameters were used: t_s (communication startup time) varied with experiments, t_{phy} (link propagation time = 16.0 nanoseconds, t_{route} (routing delay at switch = 500 nanoseconds, t_{sw} (switching time across the router crossbar for a flit) = 16.0 nanoseconds, t_{inj} (time to inject a flit into the network) = 16.0 nanoseconds, and t_{cons} (time to consume a flit from the network) = 16.0 nanoseconds.

The analytical predictions of the previous section were verified. Figure 3.3 shows the performance of 2 message sizes, 128 flits and 1024 flits. In the case of the 128 flit messages, the combining algorithm performs best except for very small numbers of nodes. In the case of 1024 flit messages, however, the SB algorithm outperforms the combining algorithm. For all startup times of 5, 10, 20, and 30 microcesconds, this crossover was observed at some message size.

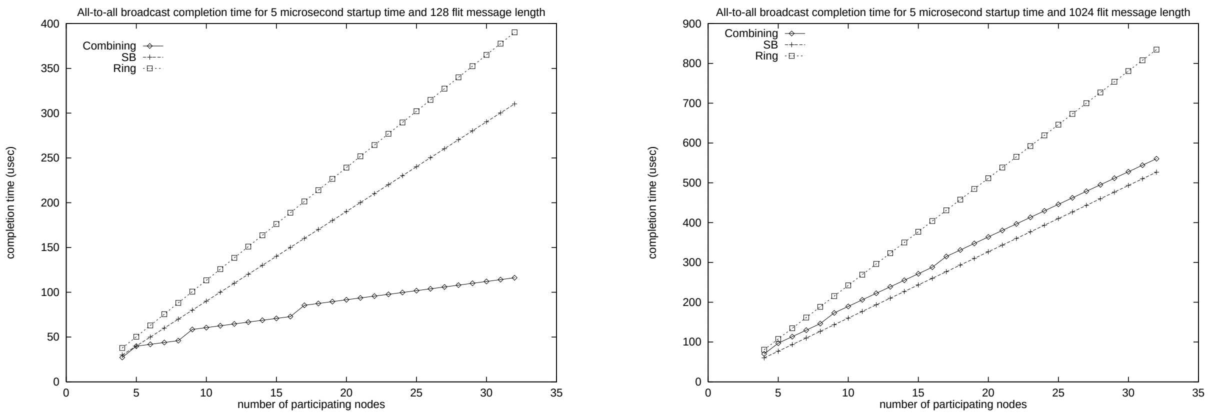

Figure 6: Single switch all-to-all broadcast completion times with 5 microsecond startup for (a) 128 flit message and (b) 1024 flit message.

3.4 Extension of algorithms to multiple switch networks

We have characterized the performance of several all-to-all broadcast algorithms for single switch networks. As send and receive overheads decrease, the SB algorithm approaches optimal performance: total transmission latency is minimal and pipelining of communication components is maximized. In the remainder of this paper, we will explore adaptations of the SB algorithm to attain efficient, near optimal performance on multi-switch networks. We will first consider two switch networks in the next section, then irregular networks with an arbitrary number of switches in the following section.

4 All-to-all broadcast on 2-switch networks

In this section, we discuss the execution of the all-to-all operation on networks consisting of two switches. The purpose of this is not that this is a common network topology. We discuss two switch networks to illustrate the issues affecting algorithm performance introduced with reduced connectivity (and therefore potential link contention) without the complexities of the general irregular networks explored in the next section.

Executing the all-to-all operation on a network consisting of two switches with a single interconnecting link introduces the potential for link contention. Communication patterns which approach optimal performance will have the following properties: (a) degree of pipelining of communication components will be significant, (b) transmission latency will be minimal, and (c) node idleness is minimized.

4.1 Algorithms

The combining and SB algorithms operate as described in the previous section. There is slight modification to the ring algorithm and we introduce a new algorithm which is a modification of the SB algorithm.

4.2 Switch-Ordered Ring (SO-R) algorithm

The operation of the ring algorithm remains essentially the same as for the single switch case except for ordering of the nodes by switch to eliminate potential link contention. For either switch, only one local node should send to a node on the other switch and only one remote node should receive from a local node. This communication pattern encounters no link contention and therefore is expected to perform about as well as the ring algorithm on a single switch. The only additional cost encountered is the additional switching time for the messages being sent between switches.

4.3 2-switch Link Scheduling (LS2) algorithm

We introduce a new algorithm, the 2-switch Link Scheduling (LS2) algorithm, which attempts to attain near-optimal performance as defined above. This algorithm is a special case of the more general Link Scheduling (LS) algorithm formally described in the next section. The basic idea is that use of the interconnecting link is scheduled among nodes in a way that permits every node to remain busy with useful work. The algorithm is described below for the two possible cases: the balanced case where there are same number of nodes on each of the two switches, and the unbalanced case where there are unequal numbers of nodes on each switch.

In general, the algorithm executes in two phases. During the first phase, nodes take turns using the interconnecting link to transfer its message data to a node on the other switch. During the steps when it is not a nodes turn to use the interconnecting link, it participates in a local all-to-all broadcast of the local message data. The second phase consists of performing another local all-to-all broadcast, this time distributing the remote data received during the first phase.

4.3.1 Balanced node distribution

Consider 2 switches with N nodes on each switch ($P = 2N$). These nodes are assigned an index, 1 to N, within the switch to which they belong. The algorithm completes in two phases. During the first phase, the N nodes on each switch perform an N-step modified SB all-to-all broadcast within each switch. A standard SB all-to-all operation takes $N - 1$ steps. The modification is that at step number i, the node with index i does not participate in the local all-to-all but rather sends its local message across the interconnecting link to the node at index $(N - i)$ on the other switch (a remote node). Likewise, at step i, the node with index $(N - i)$ receives from the remote node with index i. In this way, at the end of the N-step first phase the following has been accomplished: (a) a local switch all-to-all broadcast operation

LS2 algorithm

N_{here} = the number of nodes on this switch

N_{there} = the number of nodes on other switch

$local\_src = (myindex + 1)mod(N_{here})$

$local\_dest = (myindex - 1)mod(N_{here})$

NEXT_MSG = local message data

for stage = 1 to $\lceil \frac{N_{there}}{N_{here}} \rceil$

MSG = NEXT_MSG

for step = 0 to $(N_{here} - 1)$

if $(step == myindex)$

send MSG to remote index $(N_{dest})mod(step + (stage * N_{there}))$

else

send MSG to $(local\_dest)$

$local\_dest = (local\_dest - 1)mod(N_{here})$

if $(step == N_{here} - myindex)$

recv NEXT_MSG from remote index $(step)mod(N_{src})$

else

recv message from $(local\_src)$

$local\_src = (local\_src + 1)mod(N_{here})$

for i = 1 to $(N_{here} - 1)$

send NEXT_MSG to local index $(myrank + i)mod(N_{here})$

recv message from local index $(myrank - i)mod(N_{here})$

Note: For clarity, the algorithm is presented to illustrate communication pattern (as if sends and receives alternate). In implementation, sends and receives may occur asynchronously. The only coordination required is to ensure that sending does not continue to the next stage in phase 1 until the NEXT_MSG has been received.

Figure 7: Pseudo-code description of the LS2 algorithm for P node all-to-all broadcast.

		receive pattern				send pattern			
step	PHASE 1	PE 1	PE 2	PE 3	PE 4	PE 1	PE 2	PE 3	PE 4
1	stage 1	2	3	4	R1	R7	1	2	3
2		3	4	R2	1	4	R6	1	2
3		4	R3	1	2	3	4	R5	1
4		R4	1	2	3	2	3	4	R4
5	stage 2	2	3	4	R5	-	1	2	3
6		3	4	R6	2	4	-	1	2
7		4	R7	1	1	3	4	-	1
8		-	1	2	3	2	3	4	-
9	PHASE 2	2	3	4	-	-	1	2	3
10		3	4	-	2	-	4	1	2
11		4	-	2	3	-	3	4	1

Figure 8: Communication pattern for LS2 algorithm on switch with 4 nodes and remote switch has 7 nodes. Numbers indicate local sources or destinations. Remote sources and destinations are indicated as Rx where x is the remote index. The messages sent locally in stage 2 contain the remote data blocks received during stage 1.

has completed, (b) all local switch message data has been transferred to remote nodes, and (c) all remote message data has been received by local nodes. Since each local node now holds exactly one remote message which must be distributed within the switch, the second phase is simply a local (within-switch) SB all-to-all broadcast operation, distributing the data received in the first phase. The second phase completes in $N-1$ steps.

4.3.2 Unbalanced node distribution

Consider 2 switches, one with N nodes attached and the other with N^* nodes attached where $N > N^*$ and $N + N^* = P$. The LS2 algorithm for unbalanced node distributions also has two phases. The first phase on the switch with fewer nodes, however, will have multiple stages.

For the switch with N nodes, the first phase consists of an N-step modified SB all-to-all broadcast. Each of the N steps involves one of the local nodes sending to a node in the other switch (a remote node). The target addresses of these inter-switch sends will be $(N^* - \mathit{step})\mathit{mod}(N^*)$. A local node is involved in receiving from a remote node during only the first N^* steps of the first phase. After the N^* step of the first phase, all N^* remote messages have been received so the node scheduled to receive from outside the switch has nothing to receive and therefore continues with the next step. At the completion of the first phase, (a) a local switch all-to-all broadcast operation has completed, (b) all local message data has been transferred to remote nodes, and (c) all remote message data has been received by N^* of the local nodes. In the second phase, each of the N^* nodes which received an inter-switch message performs a broadcast to all other local nodes. The second phase still requires $N-1$ steps (because there are $N-1$ destinations for each broadcast) but each node needs to receive a maximum of only N^* messages. Therefore, this second phase will require less time to complete than that of the balanced scenario.

For the switch with N^* nodes, the first phase consists of $\lceil \frac{N}{N^*} \rceil$ stages, each with N^* steps. During the first of these stages, the following occurs: (a) messages are sent by local nodes to remote nodes in a scheduled fashion as previously described, (b) the first N^* remote messages are received, one per

Algorithm	1-switch (reference)	Balanced 2-switch	Unbalanced 2-switch
Combining	139.69	249.16	593.14
SB	137.79	1079.47	1082.24
RO-Ring	205.78	1204.24	1088.22
SO-Ring	205.78	206.84	206.84
LS2	-	141.64	184.24

Table 2: Completion time in microseconds for balanced 2-switch network (16 nodes per switch) and unbalanced 2-switch network (11 nodes on one switch, 21 nodes on the other) for 256 flit messages and 5 microsecond startup time.

local node, and (c) a modified SB all-to-all broadcast is executed, distributing the local messages locally. During each of the following stages of the first phase, modified SB all-to-all broadcasts are performed, distributing messages received from remote nodes in the previous stage while another N^* remote messages are received. This continues until a total of N messages have been received. The second phase consists of the distribution of the remote messages received in the final stage of the first phase.

4.3.3 Analysis of LS2 algorithm

In the balanced case where there are the same number of nodes on each of two switches, the LS2 algorithm is near-optimal. That is, (a) communication components are overlapped for all possible steps except one, (b) all nodes remain busy with useful work during the entire execution of the algorithm, and (c) there is no unnecessary communication. The only point at which communication overlap does not occur is at the transition from the first phase to the second phase. Whichever node receives from the remote switch in the last step of the first phase must wait until receipt of that message is complete before beginning the second phase.

This performance degrades when the numbers of nodes on the two switches are unequal. The LS2 algorithm does not make optimal use of the interconnecting link when the node distribution is not balanced. Once there are no more messages to be sent from one switch, the link becomes idle in that direction. It is possible to revise the algorithm to make use of this link during the entire execution of the algorithm with improved results.

4.4 Simulation experiments and results

We consider two scenarios to illustrate the performance issues and possible solutions for all-to-all communication on 2 switches. Each consists of (2) 32-port switches, each with a single bidirectional interconnecting link. The balanced scenario divides the nodes equally between the switches, 16 on each. The unbalanced divides the nodes asymmetrically: 21 nodes on one switch and 11 on the other.

Table 2 shows the performance results for the two scenarios considered here. For these simulations, t_s (communication startup time) = 5 microsecond and the message length is 256 flits. Results for the single switch case with these parameters are given for reference.

5 All-to-all broadcast on irregular networks of arbitrary size

For irregular networks of arbitrary size and complexity, the potential for link contention increases. The rank ordering ring (RO-R) algorithm, the default implementation supplied by MPICH [7], becomes less attractive as the potential for link contention increases. Two algorithms which perform well for arbitrary irregular networks are discussed below.

5.1 Switch-Ordered Ring (SO-R) algorithm

For an irregular network, an efficient ordering of nodes for the ring algorithm is found by performing an in-order traversal of the BFS tree that was generated to determine network routing characteristics. Although this ordering does not eliminate potential link contention completely, the potential is greatly reduced.

The lists of nodes on each switch are concatenated in the order found traversing the BFS tree to determine the node ordering for the SO-R algorithm. This ordering has a simple implementation and provides good performance for arbitrary irregular networks. Performance of the SO-R algorithm for a 10-switch, 128 node irregular topology is shown in Table 3.

5.2 Link Scheduling (LS) algorithm

Here we extend the concepts of the LS2 algorithm presented in the previous section to handle arbitrary irregular networks. This algorithm will also permit an enhancement to the LS2 algorithm which improves performance.

Essentially, the LS algorithm involves nodes on each switch performing three tasks: (a) sending currently held data blocks to nodes on another switch (the remote destination switch), (b) distributing these currently held data blocks among the nodes on the local switch, and (c) receiving data blocks from another switch (the source switch). Since nodes on the local switch communicate only with nodes on two other switches, data blocks must be forwarded around the network until all non-local data blocks have been received at each switch. There is a steady stream of data blocks needed by nodes on the local switch flowing into the switch and a steady stream of data blocks required at the remote destination switch flowing out of the local switch.

The LS algorithm consists of a setup phase plus two operational phases. The setup phase entails determining the switch ordering by traversing the routing BFS tree. Each node finds its remote source and remote destination switch from this ordering. The logical organization of the switches may be thought of as a ring.

In the first phase, there are one or more stages. During each stage, a local modified SB all-to-all operation occurs while sending remote messages to nodes on the remote destination switch and receiving remote messages from nodes on the the remote source switch. The data distributed by the modified local all-to-all broadcast is the local data in the first stage and the data received in the previous stage thereafter. Figure 9 shows the communication pattern for a particular switch in a 15-node irregular network with 4 local nodes, 5 nodes on the remote source switch, and 3 nodes on the remote destination switch. Phase 1 completes in a number of steps which is a multiple of the number of local nodes and when all required remote messages have been sent and received. To maintain a flow of only the required data blocks into and

LS algorithm communication pattern

		receive pattern				send pattern			
step	PHASE 1	PE 1	PE 2	PE 3	PE 4	PE 1	PE 2	PE 3	PE 4
1	stage 1	2	3	4	S1	D3	1	2	3
2		3	4	S2	1	4	D2	1	2
3		4	S3	1	2	3	4	D1	1
4		S4	1	2	3	2	3	4	D1
5	stage 2	S5	1	2	3	2	3	4	D2
6		4	S5	1	2	3	4	D3	1
7		3	4	S4	1	4	D3	1	2
8		2	3	4	S3	D2	1	2	3
9	stage 3	2	3	4	S2	D1	1	2	3
10		3	4	S1	1	4	D1	1	2
11		4	S1	1	2	3	4	D2	1
12		-	1	2	3	2	3	4	D3
13	PHASE 2	4	-	2	3	-	3	4	1
14		3	4	-	2	-	4	1	2
15		2	3	4	-	-	1	2	3

Figure 9: LS algorithm communication patterns for a switch with 4 local nodes. In this example, there are 5 nodes on the source switch and 3 nodes on the destination switch. Assume a total of 15 nodes participating in the all-to-all broadcast so this switch must send a total of 12 messages to the destination switch and receive a total of 11 messages from the source switch. Source and destination nodes for receives and sends are identified by their index in the switch. Messages originating from the remote source switch are identified by Sx and messages destined for the remote destination switch are identified as Dx where x is the node index.

out of the local switch, the sequence of remote receiving and remote sending reverses for every stage. The first node to receive a remote message in one phase must be the first to send it to the remote destination in the subsequent phase.

The second phase consists of any nodes which received a remote message in the final stage of the first phase performing a local broadcast of that message data.

LS algorithm

1. Determine switch ordering by performing an in-order traversal of the routing BFS tree.
2. Perform required stages of phase 1, forwarding data blocks received in the previous stage.
3. Any nodes receiving in the final stage of phase 1 broadcast to the other local nodes.

Figure 10: High level description of the LS algorithm for P node all-to-all broadcast.

5.3 Simulation experiments and results

We simulated execution of the three algorithms presented in the single switch section as well as the SO-R and LS algorithms on an irregular 128-node system. We again used a 5 microsecond startup time and 256 flit message length. The 128-node irregular network consisted of 10 switches, each with 7 to 21 nodes

Algorithm	128-node irregular topology
Combining	20542.14
SB	28537.30
RO-Ring	22632.85
SO-Ring	1868.07
LS	1361.36

Table 3: Completion time in microseconds for 128-node irregular network with 256 flit messages and 5 microsecond startup.

attached. There were 1 to 5 external links to a switch with no more than 2 links between and 2 switches. The results of these simulations is shown in Table 3. The LS algorithm completes in 27.1% less time than the SO-R algorithm. This improvement reflects the increased communication component pipelining possible with the LS algorithms. Another significant result is the 94% improvement over the rank ordered ring (RO-R) algorithm. This result highlights the degradation in performance possible when the algorithm is depends on the mapping of logical rank rather than adjusting based on knowledge of the underlying network topology.

6 Conclusions

In this paper, we have shown efficient ways of implementing all-to-all broadcast on the emerging clusters of workstations based on commodity switch-based networks. We also characterized the performance of standard implementations of this operation, suggested simple enhancements which provide significant benefit, and introduced new algorithms which approach optimal performance in some cases.

For single switch clusters, the LR, SB, and Combining algorithms were considered. It was shown that selection of the best performing algorithm is a function of both network characteristics as well as the message length and number of participating nodes in the all-to-all broadcast operation. These algorithms were compared both analytically and through simulation.

An enhancement to the ring algorithm was proposed which gave significant performance improvements for 2-switch clusters. A 2-switch Link Scheduling algorithm was proposed which was link-contention free, node-contention free, and enabled pipelining of communication components. Execution of this algorithm was simulated for both balanced and unbalanced 2-switch networks with associated performance improvements over other algorithms.

Finally, two algorithms were proposed for arbitrary irregular networks. The Switch-Ordered Ring algorithms provides an efficient ordering for the logical ring of nodes, and the general Link Scheduling algorithm was introduced. The LS algorithm uses the same switch ordering as the SO-R algorithm but enables pipelining of communication components with associated performance improvements.

References

[1] M. Barnett, S. Gupta, D. G. Payne, L. Shuler, R. van de Geijn, and J. Watts. Interprocessor Collective Communication Library (Intercom). In *Scalable High Performance Computing Conference*, pages 357–364, 1994.

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

[3] R. V. Boppana, S. Chalasani, and C. S. Raghavendra. On Multicast Wormhole Routing in Multicomputer Networks. In *Symposium on Parallel and Distributed Processing*, pages 722–729, 1994.

[4] L. De Coster, N. Dewulf, and C.-T. Ho. Efficient Multi-packet Multicast Algorithms on Meshes with Wormhole and Dimension-Ordered Routing. In *International Conference on Parallel Processing*, pages III:137–141, Aug 1995.

[5] D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a Realistic Model of Parallel Computation. In *Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 262–273, 1993.

[6] J. Duato. A Theory of Deadlock-Free Adaptive Multicast Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, pages 976–987, September 1995.

[7] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A High-Performance, Portable Implementation of the MPI, Message Passing Interface Standard. Technical report, Argonne National Laboratory and Mississippi State University.

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

[9] S. L. Johnsson and C.-T. Ho. Optimum Broadcasting and Personalized Communication in Hypercubes. *IEEE Transactions on Computers*, pages 1249–1268, September 1989.

[10] R. Kesavan, K. Bondalapati, and D. K. Panda. Multicast on Irregular Switch-based Networks with Wormhole Routing. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA-3)*, pages 48–57, February 1997.

[11] R. Kesavan and D. K. Panda. Minimizing Node Contention in Multiple Multicast on Wormhole k-ary n-cube Networks. In *Proceedings of the International Conference on Parallel Processing*, pages I:188–195, Chicago, IL, Aug 1996.

[12] X. Lin and L. M. Ni. Deadlock-free Multicast Wormhole Routing in Multicomputer Networks. In *Proceedings of the International Symposium on Computer Architecture*, pages 116–124, 1991.

[13] P. K. McKinley, H. Xu, A.-H. Esfahanian, and L. M. Ni. Unicast-based Multicast Communication in Wormhole-routed Networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(12):1252–1265, Dec 1994.

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

[15] D. K. Panda, S. Singal, and R. Kesavan. Multidestination Message Passing in Wormhole k-ary n-cube Networks with Base Routing Conformed Paths. Technical Report OSU-CISRC-12/95-TR54, The Ohio State University, December 1995. *IEEE Transactions on Parallel and Distributed Systems.* In Press.

[16] J. Y. L. Park, H. A. Choi, N. Nupairoj, and L. M. Ni. Construction of Optimal Multicast Trees Based on the Parameterized Communication Model. In *Proceedings of the International Conference on Parallel Processing*, Chicago, IL, Aug 1996.

[17] W. Qiao and L. M. Ni. Adaptive Routing in Irregular Networks Using Cut-Through Switches. In *Proceedings of the International Conference on Parallel Processing*, pages I:52–60, Chicago, IL, Aug 1996.

[18] M. D. Schroeder et al. Autonet: A High-speed, Self-configuring Local Area Network Using Point-to-point Links. Technical Report SRC research report 59, DEC, Apr 1990.

[19] R. Sivaram, C. B. Stunkel, and D. K. Panda. HIPIQS: A High Performance Switch Architecture using Input Queuing. In *Proceedings of the 12th International Parallel Processing Symposium*, pages 134–143, April 1998.

[20] C. B. Stunkel, D. Shea, D. G. Grice, P. H. Hochschild, and M. Tsao. The SP1 High Performance Switch. In *Scalable High Performance Computing Conference*, pages 150–157, 1994.

[21] C. B. Stunkel, D. G. Shea, B. Abali, et al. The SP2 High-Performance Switch. *IBM System Journal*, 34(2):185–204, 1995.