High-Performance MPI Broadcast Algorithm for Grid Environments Utilizing Multi-lane NICs

Tatsuhiro Chiba[§], Toshio Endo[§], Satoshi Matsuoka[§,♯]

[§]Tokyo Institute of Technology
2-12-1 Ookayama, Tokyo, 152-8550, Japan
chiba-t@matsulab.is.titech.ac.jp
endo@gsic.titech.ac.jp

[♯]National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan
matsu@is.titech.ac.jp

Abstract

The performance of MPI collective operations, such as broadcast and reduction, is heavily affected by network topologies, especially in grid environments. Many techniques to construct efficient broadcast trees have been proposed for grids. On the other hand, recent high performance computing nodes are often equipped with multi-lane network interface cards (NICs), most previous collective communication methods fail to harness effectively. Our new broadcast algorithm for grid environments harnesses almost all downward and upward bandwidths of multi-lane NICs; A message to be broadcast is split into two pieces, which are broadcast along two independent binary trees in a pipelined fashion, and swapped between both trees. The salient feature of our algorithm is generality; it works effectively on both large clusters and grid environments. It can be also applied to nodes with a single NIC, by making multiple sockets share the NIC. Experimentations on a emulated network environment show that we achieve higher performance than traditional methods, regardless of network topologies or the message sizes.

1 Introduction

With proliferation of large-scale grid infrastructure such as TeraGrid[18], applications that have traditionally been run on a single super computer are now being run on machines in the grid. Also, multiple applications are being combined on the grid for multi physics applications. In both situations, grid-enabled substrates of MPI are often used, such as MPICH-G2[7] and GridMPI[12].

In such environments, one of the key features for performance is collective operations such as broadcast and reduction. Since the performance of collective operations heavily depends on the network topology, the collective algorithm should take the topology into account, especially in grid environments where the network characteristics are heterogeneous. Thus several collective operation algorithms for grid have been proposed so far[9, 16]. Their main focus is to reduce the amount of communication among difference sites, because the wide area networks (WANs) are worse in the latency and the bandwidth.

Our premise is that we can further improve the performance of collective operations, by considering the recent trend of network technologies. WANs are becoming *long-and-fat*; their bandwidths are often wider than that of the NICs on computing nodes[13]. Moreover *multi-lane* networks are now becoming popular; cluster nodes are often equipped with several NICs, connected to independent links. In order to achieve high performance, it is desirable to harness the bandwidth of such networks effectively.

We propose an efficient algorithm for MPI collective operations on grids, assuming that grid resources are composed of groups of clusters with multi-lane local interconnects, linked by long-and-fat WAN pipes. The key techniques for high performance are *multi-lane broadcast trees* and *pipelined broadcast* [14] with multi-lane trees, we can fill almost all downward and upward bandwidths of all nodes. As described later, this topology effectively suits multi-lane networks, though we expect it also works well

on single lane networks. By assuming long-fat WANs, pipelined broadcast can be executed without any stalls.

One of the features of the proposed algorithm is that it is fast regardless of the message sizes, whereas efficiency of many previous algorithms heavily depended on the message sizes. For example, MPICH[4], and its grid variants use different collective algorithms for small and large messages. This paper shows that our algorithm is effective both for small and large messages.

2 Network Model

This section describes the network environments that this paper assumes and discusses some issues that we pay attention to, through the explanation of simple broadcast algorithms.

2.1 Network Environment

We consider the network environments where several clusters as grid nodes are connected by WAN of a high bandwidth for execution of MPI applications. Every node has two NICs of uniform performances respectively, and can communicate with two nodes at the same time independently. We assume all NICs can perform full duplex transmission.

We also assume that switches in clusters have sufficient performances, so that all nodes in a single cluster can communicate with each other without performance degradation (In large clusters, where switches form tree topology, this assumption may be inappropriate. In such cases, we can take each edge switch as the root of cluster and adapt multi-site algorithm as we described later). For simplicity, we assume that the bandwidth of every NIC is $b/2$, thus the bandwidth of each node is b. Under this assumption, our algorithm works efficiently when the WAN bandwidth between arbitrary two clusters is b or larger. This assumption is becoming realistic because of the improvement of the WAN performance in recent years.

Each node is assumed to be able to aggregate two NICs into one link. For simplicity, we do not consider the costs associated with link aggregation and pipelining mentioned below.

Although our algorithm is designed for nodes with two NICs, it is applicable to nodes with a single NIC, by putting two sockets or more on each NIC pair. Such an approach is adopted by SplitStream[2], which is a peer-to-peer multicasting system.

2.2 Simple Algorithms and Problems

We first consider the broadcast (Bcast) algorithm among p nodes. The root node 0 transmits a message of size M to all other $p-1$ nodes ($1 \cdots p-1$).

One of the simplest Bcast algorithms is based on a single chain. In this method, node 0 sends the message to node 1, and then node 1 sends to node 2, and so on. Let l be the communication delay and b be the bandwidth of each node (we assume aggregating two NICs for each transmission), then the total cost of this algorithm is $(p-1)\cdot(l+M/b)$, which is worse than that of other proposed algorithms. This method can be easily improved by introducing pipelined communication; when each node receives parts of messages, it can send them to the successor node immediately, before the whole message becomes available. When the overhead due to pipelining is disregarded, the cost of algorithm is improved to $(p-1)l+M/b$. However, it is inefficient when the number of nodes p is large, because the cost increases linearly with p.

Another simple algorithm is based on the binary tree. By using two NICs, each node can send the message to two child nodes simultaneously. The cost of this algorithm is $\log p(l+2M/b)$, because the bandwidth for each child is $b/2$. Pipelining technique is also applicable to this algorithm, and the cost would be $\log p \cdot l + 2M/b$. While it is better than the 'single chain' when M is small, it is twice as worse as single chain with larger M. This is because this algorithm fails to utilize the bandwidth of NICs efficiently. Each node uses only one NIC to receive a message, while the other is idle. Moreover the leaf node of binary tree doesn't use NICs for sending at all and their number dominates, accounting for about half of the nodes in a cluster. Contrastingly, the single chain method, we observe every node uses both NICs for sending and receiving, excluding the last node.

Our algorithm solves the problems of both, and works efficiently in grid environments. More, it exhibits the following features:

- The cost does not increase linearly with p, but with $\log p$, since it is based on a binary tree.
- Almost all bandwidths of each node are used, and pipelining without stalling can be naturally realized. As a result, its cost is equivalent to single chain algorithm when M is very large.
- It works efficiently in grid environments, because the amount of the communication among the sites is compareable to M.

3 Proposed Algorithm

We propose a Bcast algorithm based on the *multi-lane tree topology*. We describe the algorithm for a single site first, and then enhance the algorithm for two or more sites. Finally, features of the algorithm are described.

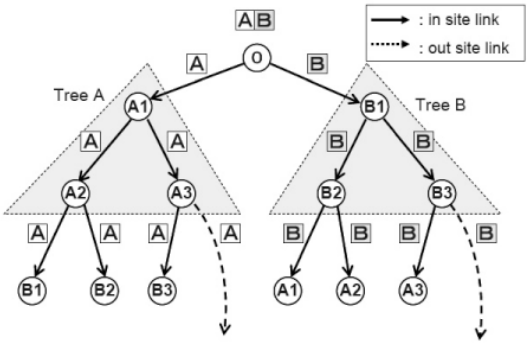

Figure 1. Multilane tree in a single site

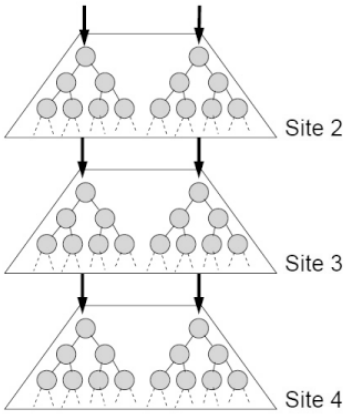

Figure 2. Multilane tree in multiple sites

3.1 Single Site Algorithm

Our algorithm constructs two binary trees as in Figure 1, and we call this topology *multi-lane tree topology*. To make the explanation simple, we assume the following preconditions. We let the number of nodes be $4n-1$, where n is an integer. Each node has two physical NICs.

The root node of Bcast is called node 0. The other nodes are divided equally into two groups, each of which has $2n-1$ nodes. Then two binary trees called $Tree_A$ and $Tree_B$ are created as shown in the figure; the leaves of trees are flat, but that condition is not requisite. We renumber nodes in $Tree_A$ as $A_1, A_2, \cdots, A_{2n-1}$ and assume the child nodes of A_i are A_{2i} and A_{2i+1}. At a result, n nodes of $A_n \cdots A_{2n-1}$ are the leaves of $Tree_A$. $Tree_B$ is similarly constructed.

The outline of our Bcast algorithm based on this topology is as follows.

(i) Node 0 divides the message M to be broadcast into two pieces of the same size, namely M_A and M_B. Then M_A is sent to A_1, which is the root of $Tree_A$ by using one link. Simultaneously, M_B is sent to B_1 similarly.

(ii) When each node receives a message fragment, it forwards it to two child nodes in a pipelined style along the tree. The node uses one link for each child. Note that at this point, each node leaves one of the downward (receiving) links idle. In the leaf nodes, both upward (sending) links are also idle. These idle links are utilized to swap messages between two trees in the next stage.

(iii) In the next stage, we send M_A to nodes in $Tree_B$ and M_B to nodes in $Tree_A$ as follows. Node $A_{n+i}(0 \leq i < n)$, which is one of leaf nodes of $Tree_A$, sends message M_A to two nodes in $Tree_B$, B_{2i+1} and B_{2i+2} by using two upward links. The figure describes this situation by drawing nodes of $Tree_B$ below leaf nodes of $Tree_A$. Because the number of leaf nodes in $Tree_A$ is n, they can cover all the $2n-1$ nodes in $Tree_B$. Conversely, the transmission from leaves of $Tree_B$ to $Tree_A$ is similarly done. We can do all communication in a pipelined style without any stalls through (i), (ii) and (iii).

(iv) When all nodes receive both all of M_A and M_B, they are combined and Bcast is complete.

3.2 Multiple Sites Algorithm

The algorithm is enhanced for grid environments consisting of several sites. The root site including the root node of Bcast is called site 1, and the rest is called site 2, 3 and so on. The number of nodes in each site can be different.

First, two binary tree $Tree_A$ and $Tree_B$ are constructed respectively in each site as in the single site algorithm. The root node 0, however, exists only on site 1. Then we connect sites in a chained style, as shown in Figure 2.

Single site algorithm is run on each site, and forwarding of the message among sites is done as follows. Note that node A_{2n-1}, which is one of the leaf nodes of $Tree_A$ of site c, leaves one of upward links idle in the single site algorithm. So it can transmit M_A to the root node of $Tree_A$ in site $c+1$. Similarly, node B_{2n-1} in site c sends to the root of $Tree_B$ in site $c+1$.

In this algorithm, pipelined transmission without stalls is still possible over the whole phase, as long as bandwidth of each WAN link between site pairs is equal to or larger than b, which is the bandwidth of each node.

One might consider that it would be difficult to obtain sufficient WAN bandwidth with a single TCP connection for long-and-fat WAN pipes. In the actual implementation, it would be possible to put multiple TCP connections between sites, or to use high performance communication software

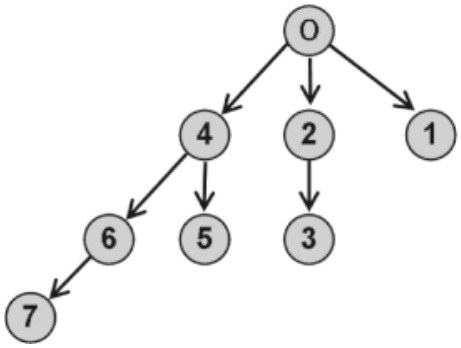

Figure 3. Binomial Tree

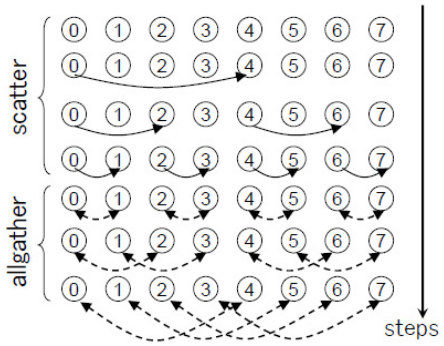

Figure 4. van de Geijn algorithm

for WAN such as Scalable TCP [8]. Evaluation in a real environment that uses such techniques is one of our future works.

3.3 Features of Algorithm

Our algorithm has the following characteristics. The message to be broadcast is divided into two, and each message is transmitted along independent two trees in a pipelined style; every node uses two links both for sending and receiving. As such, we can utilize almost all available bandwidth of all the NICs fully by assigning each link to each NIC. Furthermore, pipelined communication without stalls enables very efficient broadcasting.

4 Other Bcast Algorithms

This section compares our proposal algorithm with simple algorithms in Section 2, and well-known algorithms by using a simple cost model.

4.1 Binomial Tree

Binomial tree algorithm is a well known algorithm and used to broadcast small messages in MPICH. Figure 3 shows structure of a binomial tree. In the first step, the root sends the whole message to node $p/2$. Next, the root and $p/2$ send the message to $p/4$ and $(3/4)p$, respectively. Then the algorithm is continued recursively.

While this algorithm is preferred for small messages, it is not suitable for large messages. This is because some nodes send the whole message several times.

4.2 van de Geijn Algorithm

Next, we describe the Bcast algorithm proposed by van de Geijn[1]. As described in Figure 4, this algorithm consists of two phases.

Table 1. Estimated costs of Bcast algorithms in a single site

chain (pipelined)	$(p-1)\cdot l + M/b$
binary (pipelined)	$\log p \cdot l + 2M/b$
binominal	$\log p(l + M/b)$
Van de Geijn	$2\log p \cdot l + 2p/(p-1)\cdot M/b$
multilane (pipelined)	$\log p \cdot l + M/b$

(1) Scatter: The message is divided and scattered among all nodes by using a binomial tree.

(2) AllGather: Divided messages are collected by using recursive doubling technique. Then the whole message becomes available in every node.

This algorithm is used for the Bcast communication of messages of 512KB or less in MPICH-G2 [15, 11, 19].

4.3 Comparison by Cost Model

Table 1 shows the summary of the estimated costs of Bcast algorithms that have been described so far. It includes single chain, binary tree, binomial tree, van de Geijn and our multi-lane algorithm. In binomial tree and van de Geijn, the effects of pipelining are not considered because pipelining is difficult to apply to these algorithms.

Since the depth of binomial tree is $\log p$, the cost is $\log p(l + M/b)$. In the van de Gein algorithm, the cost of scatter phase is $(l+M/2b)+(l+M/4b)+(l+M/8b)+\cdots = \log p \cdot l + p/(p-1)\cdot M/b$. The cost of allgather phase is the same as the scatter phase. Therefore, the total cost is $2\cdot\log p + 2p/(p-1)\cdot M/b$.

When the message size is small, binary, binomial and multi-lane algorithm would work efficiently, because the

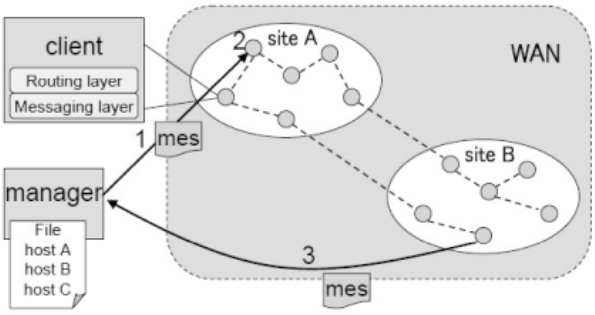

Figure 5. Overview of our simulator

cost when $M = 0$ is $\log p \cdot l$, which is smaller than other two algorithms. When the message size is large, single chain and multi-lane seem better, because the bandwidth term M/b is smaller than other algorithms. In summary, our multi-lane algorithm is expected to work efficiently both for small and large messages.

5 Collective Operation Simulator

We conduct our experiment in a controlled environment with a simulator we will describe below.

5.1 Requirements of the Simulator

To compare our Bcast algorithm and existing algorithms, the following functions are needed in the simulator.

- Simulation of networks: configuration of the number of sites and nodes, bandwidth, latency.
- Simulation of algorithm: switching among multiple collective algorithms.
- Simulation of multiple NICs: simulation of multiple NICs, and their separate and aggregated usage.

5.2 Implementation of Our Simulator

Our collective operations simulator has been implemented to satisfy the above requirements. Figure 5 shows the overview of the simulator. Clients that perform the simulated collective operations consist of the communication layer and the routing layer. In the routing layer, configuration file is read via instruction from the manager, and communication is conducted along a topology according to the algorithm and network settings.

We used the Overlay Weaver, which is an overlay network construction toolkit[17], to implement the communication and routing layers. Because the implementation of the routing layer and the communication layer is separated, users can easily implement new algorithms on the overlay network. In the simulator, each client creates two communication threads provided by the Overlay Weaver messaging service, to simulate two NICs.

Table 2. PrestoIII Cluster

OS	Debian/Linux(kernel 2.6.16)
CPU	Opteron242(1.6GHz) * 2
Memory	2GB DDR(PC2100)
NIC	1000Base-T

To simulate the specified network topology, the simulator pauses momentarily at each message send and receive. The pause time is calculated from the specified bandwidth and latency between source node and destination node of the message.

The Bcast operation is simulated as follows.

(i) When each process starts, it reads the configuration file and executes initialization task with the configuration information, such as the number of nodes and sites, intra and inter-site latency, bandwidth. The topology information such as binary tree is also read. After initialization, the manager node sends a message to the client of node 0 to inform the current time T_{start}, the collective algorithm, and the message size.

(ii) When each node receives a message, the node forwards it to the next nodes, which are determined according to the specified algorithm. To simulate network latency, the receiver node pauses for the calculated latency. Additionally, if the algorithm is not pipelined such as for binomial tree, the server node pauses to emulate the sending cost M/b.

(iii) When a leaf node receives the message at the end, it pauses for M/b to simulate the receiving cost of the whole message. This is required in pipelined algorithms. Then the node sends an acknowledge message to the manager node. When the manager node receives the messages from all the leaf nodes, it records the current time as T_{end}. Total Bcast cost is calculated as $T_{com} = T_{end} - T_{start}$.

6 Performance Evaluation

To show the effectiveness of the proposed multi-lane algorithm, we compare Bcast algorithms on the simulator described above. First, we measure the total bandwidth when multiple links are present between nodes over WAN by using GtrcNET-1[10, 6], which is a fully programmable hardware for network experiments. Second, we evaluate single

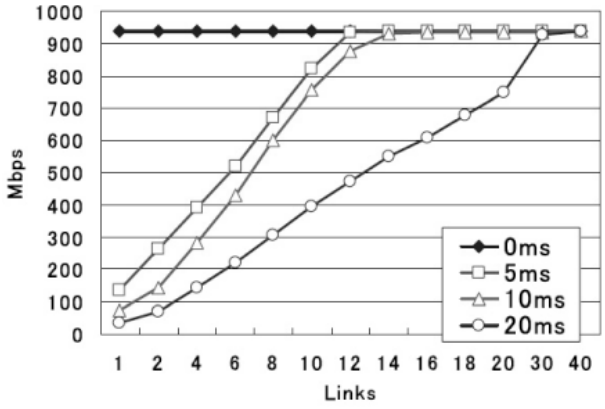

Figure 6. total bandwidth by using multiple TCP links for delay of 0 to 20ms

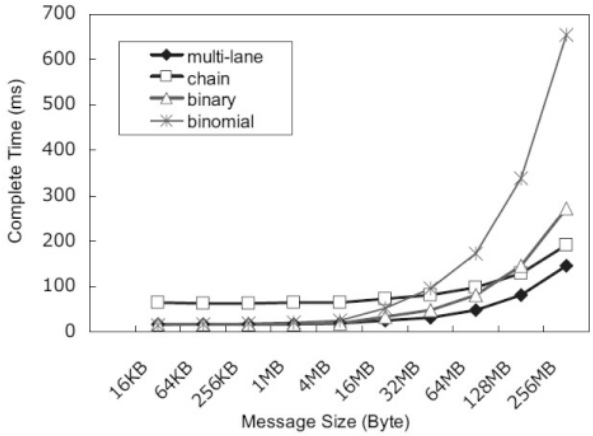

Figure 7. Results of Bcast on one site(32 nodes)

chain, binary tree , binomial tree and multi-lane algorithm, all of which are pipelined. Two network environments are simulated; a single cluster and a grid environment that consists of two clusters. We use the PrestoIII cluster at the Tokyo Institute of Technology for the experiments(Table 2).

We evaluate the algorithms on two sites as follows. We let l be intra-cluster latency, and L be inter-cluster latency. We assume both sites have p' nodes.

Single chain First, nodes in the first site are connected as a single chain. The tail node of the first site is connected to the head node of the second site. The cost of this algorithm on two sites is evaluated as $2(p'-1)\cdot l + L + M/b$.

Binary tree First, the root node sends a message to one of nodes (we call it representative node) in the second site. While the representative node is receiving the message, it starts the binary tree algorithm as usual. In the first site, after the root node finishes the transmission to the second site, it starts the binary tree algorithm. Thus the cost on two sites is evaluated as $\log p \cdot l + L + 3M/b$.

Binomial Tree The root sends a message to representative node of the second site at cost of $L + M/b$. Then each site starts binomial tree algorithm locally. So total cost on two sites is evaluated as $L + M/b + \log p' \cdot (l + M/b)$.

6.1 Performance of multiple TCP links over WAN

As we have described in Section 3.2, our multiple site algorithm assumes that sufficient bandwidth is available among different sites. To estimate the feasibility of this assumption, we have conducted a preliminary experiment by using two PrestoIII nodes, which are connected by gigabit ethernet, as follows. We insert an artificial latency between two nodes by using a hardware network emulator GtrcNET-1[10]. Then we measure the total effective bandwidth between two nodes with multiple TCP links.

Figure 6 shows the result with varying number of links. We see that the effective bandwidth heavily decreases with large latencies when a single TCP connection is used. For instance, the bandwidth with 20ms delay is about 40Mbps, which is only 4% of physical bandwidth. On the other hand, the performance is improved almost linearly as we increase the number of TCP connections. With sufficient number of connections, we can utilize more than 90% performance of physical bandwidth between WAN nodes. From this result, we see that our assumption on WAN bandwidth is realistic.

6.2 Results for a single site

We show the results of each Bcast algorithm for a single site environment with 32 nodes. Figure 7 shows the running times with varying message sizes.

When message sizes are small, we observe that multi-lane is similarly fast compared with binary tree and binomial tree. Binary and Binomial are slightly faster than multi-lane. On the other hand, the running time of binary and binomial tree become longer as the message size gets larger, so binary is twice and binomial is four times as slow as multi-lane. It is even slower than single chain when the message size is 128MB or more. The forwarding cost of single chain is extent in which 50ms was always added to multi-lane without depending on the message size.

In summary, the proposed multi-lane algorithm works efficiently both for small and large messages. Although we

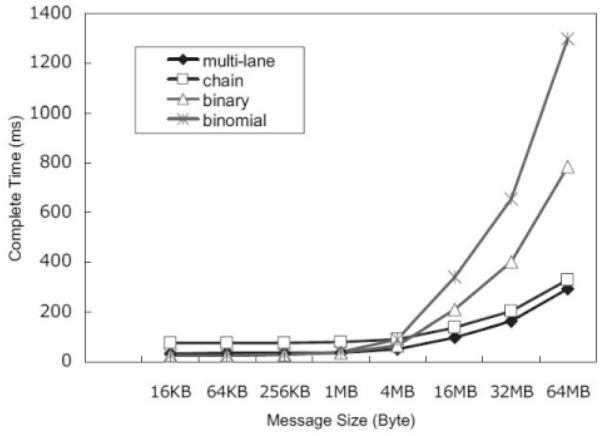

Figure 8. Results of Bcast on two sites (32 nodes in total, inter-site latency is 10ms)

observe it is slightly inferior to binary tree with small messages, the difference is very small.

6.3 Results for two sites

We show the results in a two site environment in Figure 8. The experiments conducted in the same manner as the above experiments. Each simulated site has 16 nodes, and we have 32 nodes in total. The latency between sites has been set as $L(= 10ms)$.

The results show similar tendency to the single site case, though each algorithm suffers from the inter-site latency L. For binary tree, the total time is much larger then the case of one site, and it is about three times larger that of single chain and multi-lane. This is because the usual binary tree algorithm is suspended until the inter-site transmission is completed. Binomial Tree is also similar, and because the forwarding cost from root nodes in two sites are high, the total time is about four times larger that of multi-lane. Our multi-lane algorithm works effectively also on two site environments, regardless of the message size.

7 Related Works

Many collective algorithms have been proposed for grid environments such as MagPIe[9] and MPICH-G2[7]. Most of them construct tree topologies, by using network information. Their main focus is the layered structure of WAN and LAN, and the goal is to reduce communication over sites. For example, a flat tree is used in WAN, and binomial tree is used in LAN. However, they do not mention effective use of upward links, by using multiple links and multi-lane.

A system that harnesses multiple links is SplitStream[2], which is a multicast system for peer-to-peer environments targeted towards transmitting large-scale data including movies. In this system, multiple tree structures are constructed dynamically on top of a distributed hash table (DHT), and each node participates in all the trees. The stream is divided into several sub streams, and the bandwidth of each node can be effectively by flooding each sub stream along each tree. Our single site algorithm is similar to this algorithm. However, since the main focus of SplitStream is peer-to-peer environments where nodes frequently participate and leave, it differs from our research in the following points. In SplitStream, the topology and the number of links of each node is adjusted dynamically, thus the bandwidth that is actually used changes probabilistically. Second, it does not take into account the layered structure of networks explicitly. On the other hand, this paper focuses to reduce the amount of communication among sites by distinguishing WAN and LAN.

Balanced multicasting[3] utilizes multiple NICs to improve effective bandwidth over WAN. The focus of this method is different from ours; it creates multicasting trees dynamically by using bandwidth information provided by monitoring systems.

GridMPI[5] is an MPI implementation designed for grid environments of high latency and bandwidth. The paper[13] proposes collective operation algorithms that are aware of high WAN bandwidth. Their algorithms are based on van de Geijn, and they utilize multiple TCP connections by multiple nodes over WAN, in order to improve WAN communication costs. On the other hand in their work, it is difficult to introduce pipelined transmission as in original van de Geijn algorithm. In our algorithm, the bandwidth term of the total cost function is kept low by using pipelined transmission without stalls.

8 Conclusion

We have proposed a Bcast communication algorithm for clusters and grid environments. By using multi-lane trees and pipelined transmission, it can use the bandwidth of every node fully including upward links. It is designed for environments where each node is equipped with multiple NICs, but we expect it also works well on single lane networks.

Through the experiments on the simulator, we have compared our algorithm with single chain, binary and binomial tree algorithm. As a result, we have observed that our algorithm works very efficiently regardless of message sizes, whereas performances of many existing Bcast algorithm heavily depend on message size. Ours achieves equivalent performance to binary tree with small messages. With large messages, it is superior to other algorithms, and two times

faster than others.

The followings are future work. We plan to compare with more advanced algorithms, such as van de Geijn algorithm and its extended version by Matsuda et al. In the experiments of this paper, we have ignored some performance degradation factors, such as degradation of bandwidth on WAN and costs introduced by pipelining. We would like to evaluate algorithms in more realistic environments, by using real grid environments. We have assumed that all nodes are homogeneous; they have the same number of NICs and their performances are identical. Therefore an extension of the algorithm for heterogeneous environment is also desirable. In addition to Bcast, we plan to design and evaluate other collective communications, including AllGather, AlltoAll, and so on. We expect the approach in this paper, multiple trees and pipelining, would be also applicable to those operations.

Acknowledgement

This work is partly supported by Grant-in-Aid for Scientific Research (No. 18049028 and No. 17700050) from Ministry of Education, Culture, Sports and Technology, Japan.

References

[1] M. Barnett, L. Shuler, S. Gupta, D. G. Payne, R. A. van de Geijn, and J. Watts. Building a high-performance collective communication library. In *Supercomputing*, pages 107–116, 1994.

[2] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *19th ACM Symposium on Operating Systems Principles*, 2003.

[3] M. den Burger, T. Kielmann, and H. E. Bal. Balanced multicasting: High-throughput communication for grid applications. In *ACM/IEEE Conference on Supercomputing*, 2005.

[4] I. Foster and N. Karonis. A grid-enabled MPI: Message passing in heterogeneous distributed computing systems. In *Proceedings of SC'98*, 1998.

[5] GridMPI. `http://www.gridmpi.org/`.

[6] GtrcNET. `http://projects.gtrc.aist.go.jp/gnet/`.

[7] N. T. Karohis. MPICH-G2: A grid-enabled implementation of the message passing interface. *Journal of Parallel and Distributed Computing (JPDC)*, 63(5):551–563, 2003.

[8] T. Kelly. Scalable TCP : Improving performance in high speed wide area networks. In *First Internation Workshop on Protocols for Fast Long Distance Networks*, 2003.

[9] T. Kielmann, R. F. H. Hofman, H. E. Bal, A. Plaat, and R. A. F. Bhoedjang. MagPIe: MPI's collective communication operations for clustered wide area systems. *ACM SIGPLAN Notices*, 34(8):131–140, 1999.

[10] Y. Kodama, T. Kudoh, R. Takano, H. Sato, O. Tatebe, and S. Sekiguchi. GNET-1: Gigabit ethernet network testbed. In *IEEE International Conference on Cluster Computing*, pages 185–192, 2004.

[11] S. Lacour. MPICH-G2:collective operations performance evaluation, optimizations. Technical report, Argonne National Laboratory Mathmatics Computer Science Division, 2001.

[12] M.Matsuda, T.Kudoh, Y.Kodama, R.Takano, and Y.Ishikawa. TCP adaptation for MPI on long-and-fat networks. In *IEEE International Conference on Cluster Computing (Cluster2005)*, 2005.

[13] M.Matsuda, Y.Ishikawa, T.Kudoh, Y.Kodama, and R.Takano. Efficient MPI collective operations for clusters in long-and-fast networks. In *IEEE International Conference on Cluster Computing (Cluster2006)*, 2006.

[14] P. Patarasuk, A. Faraj, and X. Yuan. Pipelined broadcast on ethernet switched clusters. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[15] J. Pjesivac-Grbovic, T. Angskun, G. Bosilca, G. E. Fagg, E. Gabriel, and J. J. Dongarra. Performance analysis of MPI collective operations. In *19th International Parallel and Distributed Processing*, 2005.

[16] H. Saito, K. Taura, and T. Chikayama. Collective operations for wide-area message passing systems using adaptive spanning trees. In *6th IEEE/ACM International Workshop on Grid Computing*, pages 40–48, 2005.

[17] K. Shudo. Overlay weaver: An overlay construction toolkit. *AIST Today*, (21):15, 2006.

[18] TeraGrid. `http://www.teragrid.org/`.

[19] R. Thakur and W. Gropp. Improving the performance of collective operations in MPICH. In *Euro PVM/MPI*, 2003.