

Two-Tree Collective Communication in Grid Computing Systems

Kwangho Cha¹, Dongsoo Han¹, and Chansu Yu²

¹ School of Engineering, Information and Communications University,
58-4 Hwa-am, Yu-seong, Daejeon, 305-732 KOREA
{khoa, dshan}@icu.ac.kr

² Department of Electrical and Computer Engineering, Cleveland State University,
2121 Euclid Avenue, Cleveland, Ohio
c.yu91@csuohio.edu

Abstract. This paper studies the collective communication in the grid computing environment, which is characterized by the combination of heterogeneous networks as well as uneven, long communication delay. Efficient collective communication requires communication schedule, which in turn requires network information. When the network information is not accurate or network faults occur, the performance of collective communication can be markedly degraded. This paper proposes TTCC (Two-Tree Collective Communication) for scheduling collective communication in the grid. It provides an efficient and reliable schedule even in this unfavorable network condition by maintaining two disjoint communication trees. Benefits of the proposed method are manifested via simulation, where the performance degradation with TTCC is much slower than those using conventional scheduling algorithms.

Index terms : broadcast, collective communication, grid, heterogeneous network, network information, and NWS.

1 Introduction

The grid computing, which combines parallel and distributed computing with high speed networking, attracts a lot of attention recently as a viable solution to provide an enormous computing power based on existing computing resources scattered in wide area networks [1]. Some pioneering grid projects such as Globus [2] and Legion [3] have been launched to prove its feasibility as well as to explore related issues. Those issues are mostly related with uncertainties inherent to the grid environment, such as heterogeneity, unpredictable structure, dynamic behavior, and multiple administrative domains [4].

Since the resources in the grid can be shared, the available performance that each resource can deliver varies with time. Thus, resource performance prediction is an indispensable facility for efficient operation of the computational grid. One recent study shows that the resource performance predictor consists of three

basic functions: measuring resource performance throughout the system, predicting the future performance of each resource, and distributing the predicted information to all interested schedulers [5]. Considering that there may be non-negligible spatial distance between resources as well as between the predictor and the schedulers, one cannot be sure about the correctness of the information. It is particularly true because resource performance is measured periodically in a relatively large interval in order not to affect the target system. The issue of correctness or freshness of information is much more important in wide area network applications. It often leads to extremely bad behavior when a scheduling is based on old inaccurate information [6]. Nevertheless, it is usually used as a basis for application scheduling.

In a message-passing parallel application, communication scheduling is an important factor to enhance the overall performance because the grid environment is characterized by a large latency among computational resources. Among them, collective communication plays an important role in the development of parallel programs by simplifying the programming task [7, 8].

For collective communication, a communication tree is generally constructed but the construction process takes exponential time in heterogeneous environment. Several heuristic algorithms, such as FEF (Fastest-Edge First), or ECEF (Earliest Completing Edge First), have been suggested in the literature [7, 9]. They achieved polynomial time complexity to find a suboptimal solution. However, the performance impact due to inaccurate information and unpredictable network faults has not been analyzed. It may seriously influence the performance and the effect can be persistent during the lifetime of the corresponding communication group.

This paper proposes Two-Tree Collective Communication (TTCC) method as an efficient scheduling method in the presence of information unreliability in the computational grid. It uses a pair of communication trees, *i.e.* Original and Redundant Tree (OT and RT), to provide the consistent performance regardless of inaccurate resource performance information.

Extensive simulation study has been conducted to compare TTCC and ECEF [9], the best performing scheduling algorithm in the literature. A network simulator [9] is augmented with the redundant tree generation algorithm and the information error generating module. From the simulation, we have found that as the information inaccuracy and the number of faulty nodes increases, the performance of ECEF is dramatically degraded, while TTCC shows more graceful performance degradation. For example, when the number of node is 100, the performance degradation of ECEF is in the range of 0% to 485%, while that of TTCC is in 19% to 104%.

This paper is organized as follows. Section 2 introduces related work on collective communication. Network information service such as NWS is presented in this section. Section 3 describes our proposed scheme, TTCC. Performance evaluation of the proposed scheme is described in Section 4. Finally, we draw conclusion in Section 5.

2 Related Work

In this section, we overview recent works on resource performance prediction with an emphasis on the prediction reliability. Then, we introduce the previous research on communication scheduling for collective communication in the computational grid.

2.1 Resource Performance Prediction

NWS (Network Weather Service) is prevail network information tool used in Globus, AppLes, and Legion projects [5, 10]. NWS analyzes the history of network performance and predicts the future network performance based on nine methodologies. It then dynamically selects the best performing predictor that shows the least mean prediction error. Their experimental study with six super-computing sites showed that generally the best predictor is in not obvious and varies from resource to resource. They also found that even an isolated Ethernet segment from general internet traffic still displays considerable performance variation [5]. In wide area networks, variation in network performance is much wider and the prediction is usually followed by a considerable error. One recent measurement study on delay and hop-count of the internet shows also that there is no strong correlation between delay and hop-count, meaning that the physical distance or hop count alone cannot predict the network performance reliably [11].

2.2 Scheduling of Collective Communication

Collective communication has been studied under various computing environments not only homogeneous [12, 13] but also heterogeneous [7, 9, 14]. In a homogenous parallel system, a communication tree for collective communication can be easily constructed because parameters for collective communication such as latency or overhead are constant [12].

Unlike homogeneous system, communication parameters of a heterogeneous system take different values, indicating various computing nodes and network features from physical media to protocols. Most of all, due to various kinds of network elements, it is impossible to construct a uniform communication tree [7]. Under this environment, the problem of finding an optimal communication tree is known to be NP-hard and requires exponential time complexity, $O(N^2 2^{2N})$, where N is the number of computational nodes. Several heuristic algorithms such as SPOC (Speed-Partitioned Ordered Chain), FNF (Fastest-Node First) [7], FEF (Fastest-Edge First), and ECEF (Earliest Completing Edge First) [9] have been suggested in the literature.

FEF approach selects a node which can be reached from the root along the fastest edge with the minimum communication cost. It repeats the same procedure but selects one reachable from those already selected. The time complexity of FEF is $O(N^2 \log N)$.

ECEF is similar to FEF but uses a different edge-selection policy. It considers the ready time of the sender and selects the edge with minimum sum of the communication cost and ready time. Look-ahead algorithm adds a look-ahead value to ECEF. This look-ahead value represents the next communication cost. Time complexities of ECEF and Look-ahead algorithms are $O(N^2 \log N)$ and $O(N^4)$, respectively.

3 Two-Tree Collective Communication (TTCC)

Heuristic algorithms introduced in section 2, could be reasonable solution for collective communication in heterogeneous environments. But the effect of inaccurate network information and unpredictable problems among nodes are not properly handled in the algorithms. As a result, the scheduling based on the algorithms is not efficient, when the inaccuracy rate of information grows. In this section, we propose Two-Tree Collective Communication (TTCC) as a reliable scheduling method in the face of information unreliability in the computational grid.

As the frequency of the network performance measuring should be limited in practical, and exactly predicting the future network performance is difficult, obtaining accurate network information at some point is non-trivial. The proposed scheme, TTCC, is an approach to mitigate the problem in the heuristic algorithms for generating schedule by properly handling the network information.

In the next subsection, we explain the communication model of TTCC. The detailed explanation of TTCC and its pseudo code follows in subsection 3.2. The simulation method to test the performance of TTCC is described in subsection 3.3.

3.1 Communication Model

The significant network information for ECEF is the communication cost between two nodes, and this is given by $latency + (\frac{message\ size}{bandwidth})$ [9]. Like the communication model of P. B. Bhat et al. [9], the communication model of this paper is mainly based on both the bandwidth and the latency of a network and it can be illustrated by fully connected graph and an asymmetric matrix.

Our model is also based on their assumption that a node executes at most one send and one receive operations simultaneously. But we add a new assumption for TTCC that a node can identify the signal from another sender, while it receives data from an original sender node. If a receiving node takes a signal from secondary sender node, it stops receive operation for a while, and compares the receiving speeds from the two sender nodes. After that it selects suitable sender nodes and resumes receiving operation. We consider the time to determine the proper sender from the two nodes as a constant α , some portions of average receiving time between two nodes in a broadcast operation.

3.2 Two-Tree Collective Communication

According to the work of P. B. Bhat et al. [9], while the performance of ECEF is similar to that of Look-ahead algorithm, the time complexity of ECEF is better than that of Look-ahead algorithm. Thus we design TTCC by modifying ECEF.

If ECEF has multiple communication trees for scheduling, a node \mathbf{A} can have multiple receiving paths. And if excessive discrepancy happens in any of paths to \mathbf{A} , it may take a path guaranteeing fast data arrival using other trees. By selecting best paths partially during execution time, the rapid performance degradation of ECEF can be prevented.

Figure 1 shows the performance improvement of using multiple trees in certain conditions³. As the number of trees increase, the degree of improvement is diminished, and the scheduling time and the number of node selection increase. Hence we use only two trees for our new approach. In making a schedule, TTCC takes around twice as much time as that of ECEF. But, since their total performances are mainly dominated by the communication time, we ignore the time of TTCC in schedule generating in this paper.

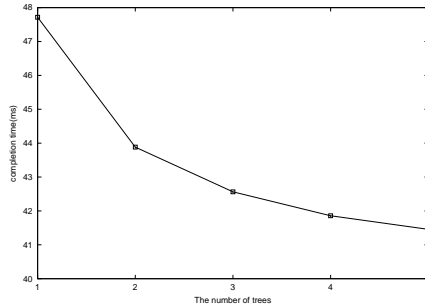


Fig. 1. Performance improvement with multiple trees.

```

procedure Redundant Tree (G:weighted bi-directional graph which has
                           vertex v and edge E)
// G = (V,E)
// E = {eij | i ∈ E, j ∈ E, and i ≠ j}
// Y is used for T0
// R is used for T1
begin
  Y := ∅
  R := ∅

  Y := Y ∪ {v0} // v0 is the vertex representing root node
  V := V - {v0}

  for i=1 to n-1 // n is the number of vertex of G
  begin
    eyz = ECEF(V,V,E) // select the edge which satisfy ECEF rule,
                       // y=v, z=v
    Y := Y ∪ {z}
    V := V - {z}

    E := E - {eyz}
    E := E - {ezy} // for redundant tree
  end

  R := R ∪ {v0}

  for i=1 to n-1
  begin
    eyz = ECEF(V,V,E)

    R := R ∪ {z}
    V := V - {z}

    E := E - {eyz}
  end
end

```

Fig. 2. Pseudo code for generating redundant tree.

To embody this idea that using two trees in scheduling into an algorithm, the ECEF algorithm is modified. In the algorithm, original tree, \mathbf{T}_0 is generated by ECEF, and in the same way the paths for redundant tree \mathbf{T}_1 is also generated. Because \mathbf{T}_1 is generated after making \mathbf{T}_0 , if there is no error in the network information, the performance using \mathbf{T}_0 is better than that using \mathbf{T}_1 , but \mathbf{T}_1

³ For this simulation the same parameters introduced in section 4 are used. The figure shows the result, when the number of nodes is 70, standard deviation of the error distribution is 0.25, and α is 0.

can compensate for finding the more efficient paths on the fly. Figure 2 shows pseudo code for generating redundant tree.

If two trees have recursive paths, that is one from node \mathbf{A} to node \mathbf{B} and the other from node \mathbf{B} to node \mathbf{A} , it is possible for a node to send back data to its sender. Since this is not normal situation, when an edge is selected for \mathbf{T}_0 , its counter edge is removed to disable its usage. Undoubtedly, if \mathbf{T}_0 is a flat tree, it is not possible to construct \mathbf{T}_1 with our pseudo code, but this is rare to happen. Therefore, in this paper, such a case that \mathbf{T}_0 has all direct edges from the root node, is not considered.

After making these two trees, communication is performed. In receiving phase, if a node receives data through a path, it disconnects another receiving path. In sending phase, it tries to send data through all paths of two trees from the paths of \mathbf{T}_0 to \mathbf{T}_1 .

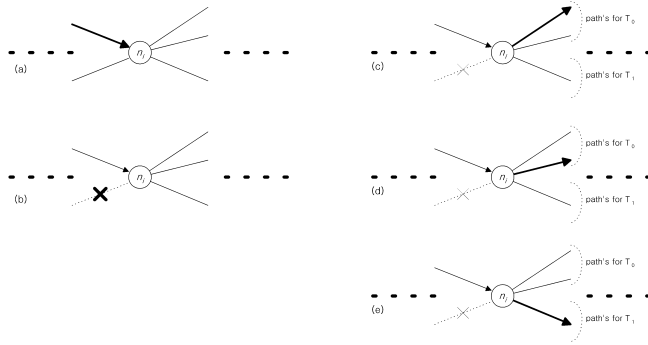


Fig. 3. A simple example illustrating how a node performs TTCC; (a)~(b):receiving phase, (c)~(e):sending phase.

Figure 3 shows redundant trees and the actions taken in each phase of a node. Once a node \mathbf{n}_i receives data through a path from its sender completely(a), it ignores the data from another input path (b). If \mathbf{n}_i receives data from another node, before completely receiving from an original node, it selects a suitable one, after monitoring the result during the time α . When this receiving event is over, it enters into sending phase. When the schedule is generated, it is expected that the performance using \mathbf{T}_0 is better than that using \mathbf{T}_1 , so \mathbf{n}_i selects the paths in \mathbf{T}_0 (c~d), before it selects a path in \mathbf{T}_1 (e).

The simple example of TTCC is in the below. Assume that Figure 4(a) is the real status of a network with which the collective communication will be performed. If there exist some discrepancies between a real network performance and its obtained network information, Figure 4(b) can be reported as network information.

With this situation, ECEF generates the schedule tree like Figure 5(a). This tree is an original tree, also called \mathbf{T}_0 , and Figure 5(b) is a redundant tree, \mathbf{T}_1 .

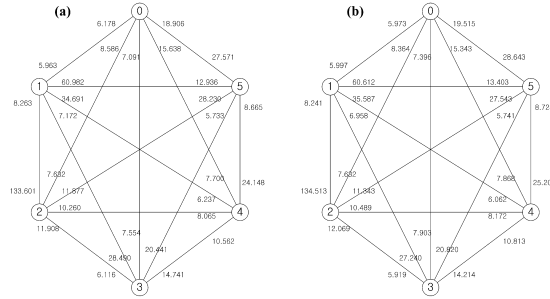


Fig. 4. The real status of a network and its network information; (a):The real status of a network, (b):Network information for (a).

With these two trees, the final solution assembled by selecting fast edges from T_0 and T_1 , can be generated on the fly, like Figure 5(c). When node 0 completes the send operation with T_0 , it starts to send the data to node 5 with T_1 . Since the communication time between node 0 and node 5 is shorter than that between node 3 and node 5, node 5 selects the path from node 0 instead of from node 3, after the time α .

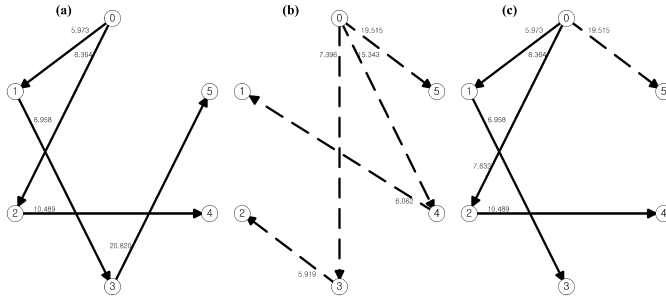


Fig. 5. The trees of TTCC for Figure 4; (a): T_0 , (b): T_1 , (c):The final solution for Figure 4.

Figure 6 shows timing diagrams for broadcasting operation, based on ECEF and TTCC of Figure 4(a) respectively. It shows that the completion time of the collective communication with TTCC could be better than that with ECEF.

3.3 TTCC simulator

To simulate TTCC, specific simulator⁴ is designed. In this subsection, the conceptual structure of TTCC simulator is introduced. The simulator has two procedures, one for generating schedule, and the other for measuring the execution

⁴ The simulator written in C is based on the simulator of P. B. Bhat et al. [9].

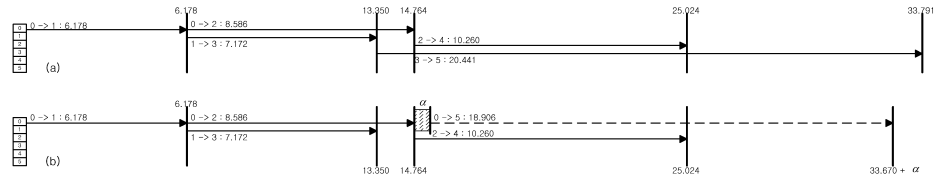


Fig. 6. Timing diagrams; (a):original scheduling tree(ECEP), (b):final scheduling tree(TTCC).

time of generated schedule. The performance of each algorithm is estimated from the measured execution time.

For the simulation, two communication cost matrices are used. At first, the original assumed matrix \mathbf{M}_0 , which represents the real network condition, is generated. With this \mathbf{M}_0 , within a certain distribution of error rate, distorted matrix \mathbf{M}_1 is build up.

In the first procedure, generating schedule, the matrix \mathbf{M}_1 is used for making up the schedule for both ECEP and TTCC. In the next procedure, to get the performance in the real condition, not \mathbf{M}_1 but \mathbf{M}_0 is used to calculate the execution time of each scheduling algorithm. The result of simulation using the TTCC simulator is explained in the next section.

4 Performance Evaluation

We simulated TTCC with the simulator introduced in the previous section. In this section, we describe the performance of TTCC based on the result of the simulation.

The NWS, prevailing network information monitoring tool, provides useful network information. However, it is reported that some minor errors frequently occurs [5, 15]. According to the histograms of error distribution [15], many small errors are detected frequently, moreover large errors are also detected too often to ignore. For reflecting this error distribution to the TTCC simulation, the normal distribution⁵, $N(\mu, \sigma)$ is applied to random number generation module for the distorted network cost matrix. As the error range can be reflected by standard deviation (σ), various standard deviations, from 0 to 0.5, are used in this simulation. Considering the result of [15], the general situation of standard deviation of error information is in the range from 0 to 0.3, so we assumed that the range from 0.3 to 0.5 to represent network faults.

The simulation parameters used for this experiment are as follows.

- The message size is **1MB**.
- The network latency is in the range of **10usec to 1msec**.
- The network bandwidth is in the range of **10KB/s to 200MB/s**.

⁵ To generate the random number following the normal distribution, **CSIM18** is used [16].

With these parameters, the experiment is performed varying the number of nodes and the experiment is repeated 1000 times. Figure 7 compares the performance between ECEF and TTCC under two standard deviations in error distribution. As the number of nodes increase, the execution time increases gradually. On the other hand, we can see steep changes in Figure 8. It shows the performance is degraded as the standard deviation of an error distribution increases. This reveals that the change of the error distribution is much more serious to the performance degradation than due to the number of nodes.⁶

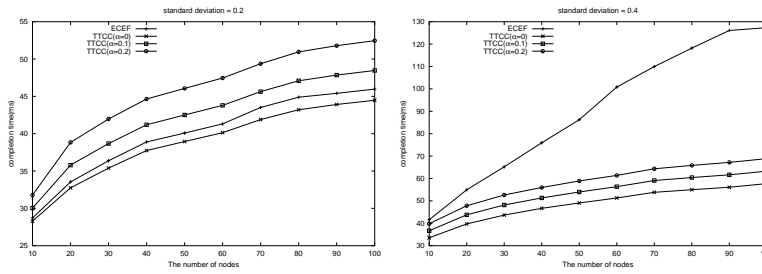


Fig. 7. TTCC performance on various standard deviations for error distribution.

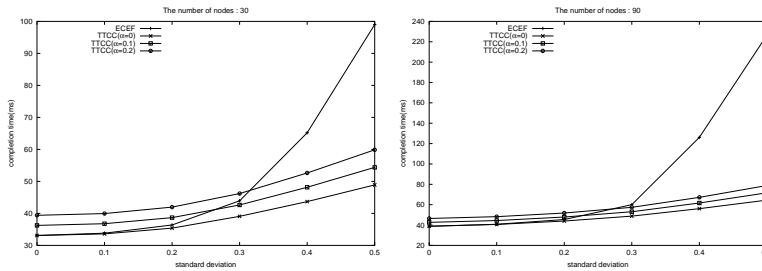


Fig. 8. TTCC performance on various numbers of nodes.

While ECEF is better than TTCC, when the network information is correct, TTCC shows better performance than ECEF with inaccurate network information. Particularly, when the standard deviation of the error distribution

⁶ α is a time to determine the suitable sender node for a receiver node. If a node can select the suitable node promptly, supporting by specific hardware, α will be 0. In general situation, we assumed it takes some portions of average communication time among nodes, and α is 0.1 means it takes 10% of average communication time for a node to select a suitable sender node.

increases, the scheduling with TTCC shows much better performance than that with ECEF. This situation can be found also in Figure 9.

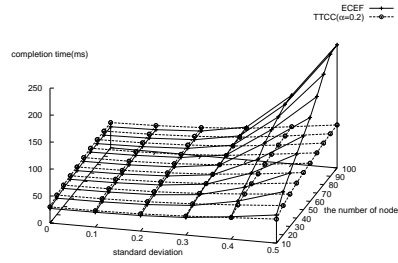


Fig. 9. The result of the experiment for TTCC performance.

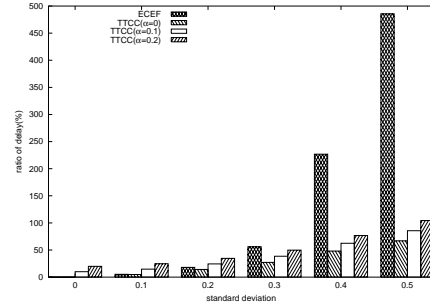


Fig. 10. The ratio of delay; The number of node is 100.

Figure 10 shows the change of the delay displayed by the ratio. The ratio of delay is represented by $\frac{P(E_i) - P(E_0)}{P(E_0)}$, where $P(E_i)$ is the execution time when the standard deviation of error distribution is i . The graph shows the degree of the performance degradation as the standard deviation of error rate increases. For example, when the standard deviation is 0.3, the performance degradation of ECEF is 55%, while that of TTCC is 49%, and when the standard deviation is 0.4, they are 226% and 76% respectively.

5 Conclusion

Previous studies of collective communication in distributed heterogeneous systems are mainly focused on the efficiency of communication based on correct network information. But uptodate network information is sometimes difficult to obtain. In this paper, TTCC is suggested to generate a reliable communication scheduling even in unfavorable network conditions. Redundant tree is constructed additionally and it is used to find better scheduling path during the execution time. Simulation result shows that although TTCC has non-negligible overhead with general situation, it could construct better scheduling tree with the inaccurate network information. Particularly, with higher number of nodes and high error rate, while ECEF suffers from serious performance degradation, TTCC shows tolerable performance.

This paper is based on only simulation, and some factors, such as the time to generate schedule or the frequency of generating schedule, are not completely considered. But those are important factors to get the exact result, so we are planning to study on these factors.

References

1. I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: enabling scalable virtual organizations," *The International Journal of Supercomputer Applications*, 2001.
2. I. Foster, and C. Kesselman, "The Globus Project: A status report," *7th Heterogeneous Computing Workshop*, pp. 4 ~ 18, 1998.
3. A. S. Grimshaw, and W. A. Wulf, "Legion - A view from 50,000 feet," *5th IEEE International Symposium on High Performance Distributed Computing*, pp. 89 ~ 99, 1996.
4. G. Aloisio, M. Cafaro, C. Kesselman and R. Williams, "Web access to supercomputing," *IEEE Computing in Science & Engineering*, pp. 66 ~ 72, November/December 2001.
5. R. Wolski, "Dynamically forecasting network performance using the network weather service," *Journal of Cluster Computing*, 1998.
6. M. Mitzenmacher, "How useful is old information," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 11, No. 1, pp.6 ~ 20, Jan. 2000.
7. M. Banikazemi, V. Moorthy, and D. K. Panda, "Efficient collective communication on heterogeneous networks of workstations," *International Conference on Parallel Processing*, pp. 460 ~ 467, 1998.
8. M. Bernaschi and G. Iannello, "Collective communication operations: experimental results vs. theory," *Concurrency: Practice and Experience*, pp. 10(5):359 ~ 386, 1998.
9. P. B. Bhat, C. S. Raghavendra, and V. K. Prasanna, "Efficient collective communication in distributed heterogeneous systems," *19th IEEE International Conference on Distributed Computing Systems*, pp. 15 ~ 24, 1999.
10. R. Wolski, N. T. Spring, and J. Hayes, "The network weather service: a distributed resource performance forecasting service for metacomputing," *Journal of Future Generation Computing Systems*, pp. 15(5~6):757 ~ 768, 1999.
11. A. Fei, G. Pei, R. Liu and L. Zhang, "Measurements on delay and hop-count of the internet," *3rd Global Internet Mini-Conference in conjunction with IEEE Globecom*, 1998.
12. A. Bar-Noy and S. Kipnis, "Designing broadcasting algorithms in the postal model for message passing systems," *4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 13 ~ 22, 1992.
13. D. Culler, R. Karp, D. Patterson, A. Sahay, K.E. Schauer, E. Santos, R. Subramonian, and T. von Eicken, "LogP: Towards a realistic model of parallel computation," *4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 1 ~ 12, 1993.
14. M. Banikazemi, J. Sampathkumar, S. Prabhu, D. K. Panda, and P. Sadayappan, "Communication modeling of heterogeneous networks of workstations for performance characterization of collective operations," *International Workshop on Heterogeneous Computing*, pp. 125 ~ 131, 1999.
15. C. Krintz, R. Wolski, "JavaNws: the network weather service for the desktop," *Proceedings of Java Grande*, pp. 116 ~ 125, 2000.
16. CSIM18 web page, <http://www.mesquite.com/>.